JOSHUA PARKER

SOURCE-CODE

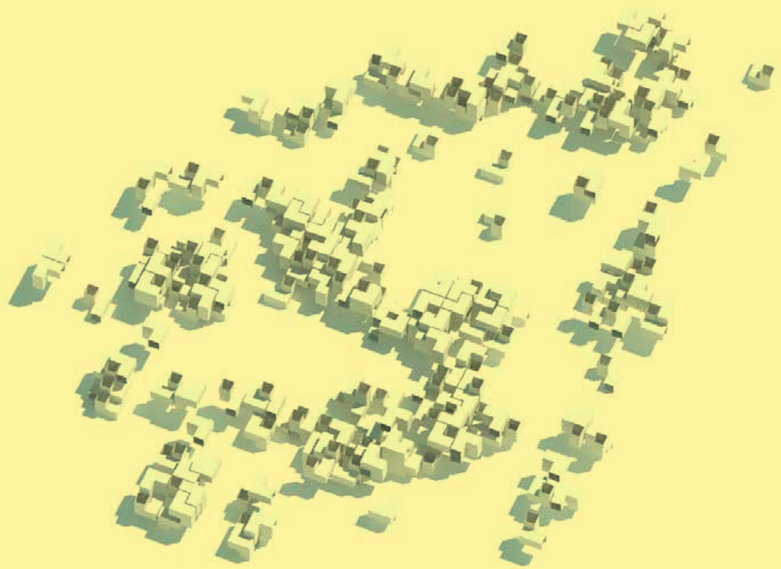
FOR SAMPLE WORKS

JOSHUA PARKER, M. Arch., Syracuse University School of Architecture (SUSOA), B.S. Electrical Engineering, University of Washington (UW), Certification, Institute of Advanced Architecture of Catalonia (IaaC)

CONTENTS

PROTOCELL, SOURCE CODE

SOURCE CODE FOR PROTOCELL GENERATIVE HOUSING SYSTEM. THE ALGORITHM, INITIALLY IMPLEMENTED IN RHINOSCRIPT, CAN BE DIVIDED INTO THREE ISOLATED STAGES. THE FIRST DISTRIBUTES CELLULAR MASS ACCORDING A CA LOGIC AND OPTIMIZER THAT ADJUSTS MASSING FOR CIRCULATION PATHS AND OTHER ENCODED CONSTRAINTS LIKE SITE, TOPOGRAPHIC FEATURES, ETC. THE FIRST STAGE REQUIRES AS INPUT AN INITIAL MASSING CONDITION AS CELLULAR AUTOMATA PROCEDES AS INCREMENTAL REFINEMENT.[1] FIRST STAGE IS ALSO RESPONSIBLE FOR CONSTRUCTING A CELL NETWORK, IE. EACH CELL MAINTAINS STATE AND KEEPS TRACK OF NEIGHBOORING CELLS. THIS NETWORK IS PASSED TO STAGE TWO ALONG WITH FURTHER BUILDING CONSTRAINTS LIKE PROGRAM, FLOORSPACE ALLOCATION, ETC. THE SECOND STAGE SUBDIVIDES SINGLE-CELL MASS INTO MULTI-CELL CLUSTERS OR LINKED CELL CHAINS. THIS IS ACCOMPOLISHED BY ESTABLISHING AN INITIAL CONDITION OF RANDOMLY PLACED CONDITIONAL UNITS AND DEFINING A FITNESS FUNCTION THAT EVALUATES THE PLACEMENT AND SITUATION OF THE UNIT AND SIMPLY DISCARDS LOSERS AND IGNORES WINNERS. THIS IS A WEAK GENETIC OPTIMIZATION ALGORITHM THAT IS MORE BRUTE FORCE THAN ANYTHING ELSE, BUT IT DOES FIND SOLUTIONS IN REASONABLE AMOUNT OF TIME. THE FINAL STAGE EMBEDS BUILDING INTELLEGENCE INTO CELL CLUSTERS, PLACES UTILITY CORE, AND ADAPTS UNIT TO BASIC BUILDING CONSTRAINS... ALL STAGES ARE INTENDED TO BE A GUIDED META-DESIGN PROCESSES, IN WHICH DESIGNERS PARTICIPATE IN REAL TIME BY MAKING SUBJECTIVE DECISIONS AND FEEDING THEM BACK INTO THE SYSTEM.




```
'--------------------------------------------------------
'Smart cells v1
'
'
'--------------------------------------------------------

Option Explicit
'Script written by Josh Parker
'Script copyrighted by OPEN Architecture
'Script version Tuesday, February 15, 2011 3:25:36 AM

Call Main()
Sub Main()

	Dim arrPts, intSteps, intLevels, intSpacing, oMajRule
	Dim oSubstr, oSnakeFarm

	'get initial data
	arrPts=Rhino.GetObjects("select points to populate",1)
	If Not IsArray(arrPts) Then
		Rhino.Print "no points selected"
		Exit Sub
	End If
	intSpacing=Rhino.GetInteger("enter cell spacing",5)
	intSteps=Rhino.GetInteger("enter number of steps",2)
	intLevels=Rhino.GetInteger("enter number of levels",2)

	'create substrate
	Set oSubstr = New Substrate
	'substrate "constructor"
	Call oSubstr.Build(intSpacing, intSteps, intLevels)
	'add Cells from points
	Call oSubstr.AddCellsByColor(arrPts)
	'link each cell to its neighbors
	Call oSubstr.ConnectCells()
	'group cells by type
	Call oSubstr.GroupCells()
	'color cell pts by cell type
	Call oSubstr.ColorByType()

	'create snakes
	Set oSnakeFarm = New SnakeFarm
	'locate snakes in substrate
	Call oSnakeFarm.FindSnakes(oSubstr, 4)
	'spine the snakes
	Call oSnakeFarm.SpineSnakes()
	'skin the snakes
	Call oSnakeFarm.SkinSnakes((CDbl(intSpacing)/2)*.9)

	'create massing
	'Set oMass = New massing
	'build mass from snake formation
	'oMass.build(oSnakes)


End Sub


'--------------------------------------------------------
'Cell CLASS
'
'
'--------------------------------------------------------

Option Explicit
'Script written by Josh Parker
'Script copyrighted by OPEN Architecture
'Script version Tuesday, February 15, 2011 3:25:36 AM

Class Cell

	Public m_intType
	Public m_strPt

	'array of neighbors
	Public m_arrNeighbors
	Public m_arrVonNeumanns

	'moore subsets
	Public m_arrType0Ms
	Public m_arrType1Ms
	Public m_arrType2Ms

	'vonneumann subsets
	Public m_arrType0VNs
	Public m_arrType1VNs
	Public m_arrType2VNs

	'neighbor pointers
	Public m_objNorth
	Public m_objSouth
	Public m_objEast
	Public m_objWest
	Public m_objTop
	Public m_objBottom

	'is cell part of snake
	Public m_boolIsOpen


	Private Sub Class_Initialize
		'Rhino.Print "Initialize cell"
		m_boolIsOpen = True
	End Sub

	Private Sub Class_Terminate
		'Rhino.Print "Terminate cell"
	End Sub

	Public Sub Build(strPt, intType)

		m_intType = intType
		m_strPt = strPt

	End Sub

	Public Sub Connect(arrCells, intRadius)

		'check arg is valid non-empty array
		If IsEmpty(arrCells) Or IsNull(arrCells) Then
			Rhino.Print "Connect arg, arrCells is empty or
null"
			Exit Sub
		ElseIf Not IsArray(arrCells) Then
			Rhino.Print "Connect arg, arrCells is not an ar-
ray, its a: " + TypeName(arrCells)
			Exit Sub
		End If

		Call FindNeighbors(arrCells, intRadius)
		'Call PointAtNeighbors()
		Call GroupNeighbors()
		Call GroupVonNeumanns()

	End Sub

	'find neighbors and create ordered array of pointers---------
	Private Sub PointAtNeighbors()

		Dim c, p1, p2

		'sort neighbors
		For Each c In m_arrNeighbors

			'get coordinate of target and point
			p1=Rhino.PointCoordinates(m_strPt)
			p2=Rhino.PointCoordinates(c.m_strPt)

			If p2(1) > p1(1) Then
				'y is >, its on north
				Set m_objNorth = c
			ElseIf p2(1) < p1(1) Then
				'y is <, its on south
				Set m_objSouth = c
			ElseIf p2(0) > p1(0) Then
				'x is >, its on east
				Set m_objEast = c
			ElseIf p2(0) < p1(0) Then
				'x is <, its on west
				Set m_objWest = c
			ElseIf p2(2) > p1(2) Then
				'z is >, its on top
				Set m_objTop = c
			ElseIf p2(2) < p1(2) Then
				'z is <, its on bottom
				Set m_objBottom = c
			Else
				Rhino.Print "ignore, same cell"
			End If

		Next
```

```
End Sub

'Get points within radius from cell pt--------------------
Private Sub FindNeighbors(arrCells, intRadius)

	Dim c, d, i, j, closeCells(), adjacentCells(), p1, p2

	i=0
	j=0
	For Each c In arrCells
		'calc distance to cell c
		p1=Rhino.PointCoordinates(m_strPt)
		p2=Rhino.PointCoordinates(c.m_strPt)
		d=Rhino.Distance(p1,p2)
		'if its within radius, its adjacent
		If d<=(intRadius*1.1) And d>0 Then
			ReDim Preserve adjacentCells(i)
			Set adjacentCells(i) = c
			i=i+1
		End If
		'if its within 1.5radius, its close
		If d<=(intRadius*1.5) And d>0 Then
			ReDim Preserve closeCells(j)
			Set closeCells(j) = c
			j=j+1
		End If
	Next

	'set if any
	If i>0 Then
		m_arrVonNeumanns = adjacentCells
	Else
		Rhino.Print "no von neumanns"
		Rhino.ObjectColor m_strPt, RGB(0, 0, 255)
	End If

	'set if any
	If j>0 Then
		m_arrNeighbors = closeCells
	Else
		Rhino.Print "no moores"
		Rhino.ObjectColor m_strPt, RGB(0, 255, 0)
	End If


End Sub


	'Group neighbors by type----------------------------------
	Public Sub GroupNeighbors()
		'Rhino.Print "Grouping Neighbors"

		Dim type0(), type1(), type2()
		Dim i, a, b, c
		a=0
		b=0
		c=0

		For i=0 To Ubound(m_arrNeighbors)

			'group by type
			Select Case m_arrNeighbors(i).m_intType
				Case 0
					ReDim Preserve type0(a)
					Set type0(a) = m_arrNeighbors(i)
					a=a+1
				Case 1
					ReDim Preserve type1(b)
					Set type1(b) = m_arrNeighbors(i)
					b=b+1
				Case 2
					ReDim Preserve type2(c)
					Set type2(c) = m_arrNeighbors(i)
					c=c+1
				Case Else
					Rhino.Print "GroupNeighbors()
found bad cell"
			End Select
		Next

		'copy if smth to copy, else leave empty
		If a > 0 Then m_arrType0Ms = type0
		If b > 0 Then m_arrType1Ms = type1
		If c > 0 Then m_arrType2Ms = type2

	End Sub


'Group neighbors by type----------------------------------
Public Sub GroupVonNeumanns()
	On Error Resume Next
	'Rhino.Print "Grouping Neighbors"

	Dim type0(), type1(), type2()
	Dim i, a, b, c
	a=0
	b=0
	c=0

	If IsNull(m_arrVonNeumanns) Then Exit Sub

	For i=0 To Ubound(m_arrVonNeumanns)

		'group by type
		Select Case m_arrNeighbors(i).m_intType
			Case 0
				ReDim Preserve type0(a)
				Set type0(a) = m_
arrVonNeumanns(i)
				a=a+1
			Case 1
				ReDim Preserve type1(b)
				Set type1(b) = m_
arrVonNeumanns(i)
				b=b+1
			Case 2
				ReDim Preserve type2(c)
				Set type2(c) = m_
arrVonNeumanns(i)
				c=c+1
			Case Else
				Rhino.Print "GroupVonNeumanss()
found bad cell"
		End Select
	Next

	'copy if smth to copy, else leave empty
	If a > 0 Then m_arrType0VNs = type0
	If b > 0 Then m_arrType1VNs = type1
	If c > 0 Then m_arrType2VNs = type2

End Sub

'color by type--------------------------------------------
Public Sub UpdateColor()

	Dim color

	'set color var by cell type
	Select Case m_intType
		Case 0
			color = RGB(255, 0, 0)
		Case 1
			color = RGB(0, 255, 0)
		Case 2
			color = RGB(0, 0, 255)
		Case Else
			color = RGB(255, 255, 255)
	End Select

	'set object color
	Rhino.ObjectColor m_strPt, color
End Sub

'apply ruleset--------------------------------------------
Public Sub Apply(oRuleSet)

	Dim newType

	'calculate new type
	'newType = oRuleSet.Apply(m_objNorth, m_objSouth, m_ob-
jEast, m_objWest, m_objTop, m_objBottom)
	newType = oRuleSet.Apply2(m_arrNeighbors)
	'set new type
	m_intType = newType

End Sub

' ------------------------------------------
Public Sub MajorityRuleStep()

	'count number of a,b,c types within neighbors
```



```
		'set type to that type

	End Sub

End Class

'-------------------------------------------------------
'MajorityRule CLASS
'
'
'-------------------------------------------------------
Option Explicit
'Script written by <insert name>
'Script copyrighted by <insert company name>
'Script version Thursday, February 17, 2011 11:50:25 PM

Class MajorityRule


	Public Function Apply(n,s,e,w,t,b)

		Dim cells, counts, c, i

		cells = Array(n,s,e,w,t,b)
		counts = Array(0,0,0)

		'count number of each type
		For Each c In cells
			Rhino.Print TypeName(c)
			If Not IsEmpty(c) Then
				Select Case c.m_intType
					Case 0
						counts(0)=counts(0)+1
					Case 1
						counts(1)=counts(1)+1
					Case 2
						counts(2)=counts(2)+1
					Case Else
						'do nothing
				End Select
			Else
				'is null b/c no cell
				Rhino.Print "skip " + TypeName(c)
			End If
		Next

		'return most numerous type
		j=0
		For i=0 To Ubound(counts)
			If counts(i) = Max(counts) Then
				Apply = i
			End If
		Next

	End Function

	'This one is better
	Public Function Apply2(arrCells)

		Dim cells, counts, c, i, j, maj(), iRand

		'cells = Array(n,s,e,w,t,b)
		counts = Array(0,0,0)

		'count number of each type
		For Each c In arrCells
			If Not IsEmpty(c) Then
				Select Case c.m_intType
					Case 0
						counts(0)=counts(0)+1
					Case 1
						counts(1)=counts(1)+1
					Case 2
						counts(2)=counts(2)+1
					Case Else
						'do nothing
				End Select
			Else
				'is null b/c no cell
				Rhino.Print "skip " + TypeName(c)
			End If
		Next

		'return most numerous type
		For i=0 To Ubound(counts)
			If counts(i) = Max(counts) Then
				ReDim Preserve maj(j)
				maj(j) = i
				j=j+1
			End If
		Next

		'if tie
		If(Ubound(maj)>0) Then
			'get random max
			iRand=RndInt(0,Ubound(maj))
			Apply2 = maj(iRand)
		Else
			Apply2 = maj(0)
		End If

	End Function

End Class
Option Explicit
'Script written by Josh Parker
'Script copyrighted by OPEN Architecture
'Script version Tuesday, February 15, 2011 3:25:36 AM

'-------------------------------------------------------
'SNAKEFARM CLASS
'
'
'-------------------------------------------------------
Class SnakeFarm

	Public m_arrSnakes
	Public m_intSnakeCnt

	Public Sub FindSnakes(oSubstr, intSize)

		Dim snakeCells, newSnakes()
		Dim i, failed, invalid

		'check args
		If IsNull(oSubstr) Then
			Rhino.Print "FindSnakes arg, oSubstr is null"
			Exit Sub
		ElseIf Not TypeName(oSubstr) = "Substrate" Then
			Rhino.Print "FindSnakes arg, oSubstr not a Sub-
strate, its a: " + TypeName(oSubstr)
			Exit Sub
		End If

		i=0
		failed = 0
		invalid = 0

		'find snakes
		Do While failed<500
			'get next set of cells
			snakeCells = NextSnake(oSubstr, intSize)
			'if not invalid or null, make a snake and add to
farm
			If IsNull(snakeCells) Then
				'found a short snake, increment failed
count
				failed=failed+1
				Rhino.Print "found short snake, short
count: " + CStr(failed)
			Else
				Rhino.Print "found snake, size: " +
CStr(Ubound(snakeCells)+1)
				ReDim Preserve newSnakes(i)
				'create new snake
				Set newSnakes(i) = New Snake
				'build snake
				Call newSnakes(i).build(snakeCells)
				i=i+1
			End If
		Loop

		If i>0 Then m_arrSnakes = newSnakes

	End Sub

	'find snake-----------------------------------------
	Public Function NextSnake(oSubstr, intSize)

		Dim oRndCell, arrOpenCells, snakeCells(), snake-
Cells2(), i, n
		Dim arrBB, strBB, arrVol, strSpine
```

```
		'check for valid args
		If IsNull(oSubstr) Or IsEmpty(oSubstr) Then
			Rhino.Print "NextSnake arg, oSubstr is null or
empty"
			NextSnake = Null
			Exit Function
		ElseIf Not IsArray(oSubstr.m_arrType0) Then
			Rhino.Print "NextSnake arg, oSubstr.m_arrType0
is not an array, its a: " + TypeName(oSubstr.m_arrType0)
			NextSnake = Null
			Exit Function
		End If

		'get open cells
		arrOpenCells = OpenCells(oSubstr.m_arrType0)

		If IsNull(arrOpenCells) Then
			'found a short snake, break & return null
			Rhino.Print "no more snakes"
			NextSnake = Null
			Exit Function
		End If

		'get random cell from open cells
		Set oRndCell = RndCell(arrOpenCells)

		'create snake
		ReDim Preserve snakeCells(0)
		Set snakeCells(0)=oRndCell
		snakeCells(0).m_boolIsOpen = False
		i=1
		'while snake size is not met
		Do While i < intSize
			'if no where for snake to grow, break
			If IsNull(NextCell(snakeCells(i-1))) Then
				'found a short snake
				Rhino.Print "NextSnake call to Next-
Cell() returned null, its just means its a short snake"
				'release cells and return null and exit
				ReleaseCells(snakeCells)
				NextSnake = Null
				Exit Function
			End If

			'get next cell
			Set n = NextCell(snakeCells(i-1))
			'add to tmp array
			ReDim Preserve snakeCells(i)
			Set snakeCells(i) = n
			'close cell
			snakeCells(i).m_boolIsOpen = False
			i=i+1
		Loop

		'if snake is 2d, reject it, release cells, and return
null
		If CellsCoplanar(snakeCells) Then
			Rhino.Print "NextSnake produced 2d snake, re-
ject"
			'release cells and try again
			ReleaseCells(snakeCells)
			'try again recursively
			NextSnake = NextSnake(oSubstr,intSize)
			Exit Function
		End If

		'return valid snake
		NextSnake = snakeCells

	End Function

	'addSegs-------------------------------------------------
	Public Sub ReleaseCells(arrCells)

		Dim c

		'check arg is valid non-empty array
		If IsEmpty(arrCells) Or IsNull(arrCells) Then
			Rhino.Print "ReleaseCells arg, arrCells is empty
or null"
			Exit Sub
		ElseIf Not IsArray(arrCells) Then
			Rhino.Print "ReleaseCells arg, arrCells is not
an array, its a: " + TypeName(arrCells)
			Exit Sub
		End If

		For Each c In arrCells
			c.m_boolIsOpen = True
		Next

	End Sub

	'addSegs-------------------------------------------------
	Public Function NextCell(currCell)

		Dim arrOpenCells

		'check args
		If IsNull(currCell) Or IsEmpty(currCell) Then
			Rhino.Print "currCell is null or empty"
			NextCell = Null
			Exit Function
		ElseIf Not TypeName(currCell) = "Cell" Then
			Rhino.Print "currCellis not a Cell, its a: " +
TypeName(currCell)
			NextCell = Null
			Exit Function
		End If

		'check args
		If IsNull(currCell.m_arrType0VNs) Or IsEmpty(currCell.m_
arrType0VNs) Then
			Rhino.Print "currCell.m_arrType0VNs is null or
empty"
			NextCell = Null
			Exit Function
		ElseIf Not IsArray(currCell.m_arrType0VNs) Then
			Rhino.Print "currCell.m_arrType0VNs is not an
array, its a: " + TypeName(currCell.m_arrType0VNs)
			NextCell = Null
			Exit Function
		End If

		'get open von neumanns
		arrOpenCells = OpenCells(currCell.m_arrType0VNs)

		If Not IsNull(arrOpenCells) Then
			'pick one at random
			Set NextCell = RndCell(arrOpenCells)
		Else
			Rhino.Print "NextCell found no open von neu-
manns"
			NextCell = Null
		End If

	End Function

	'----------------------------------------------------
	Public Sub SpineSnakes()

		Dim snake, strNewSpine

		For Each snake In m_arrSnakes

			strNewSpine = StrokeCells(snake.m_arrCells)

			If Not IsNull(strNewSpine) Then
				snake.m_strSpine = strNewSpine
			Else
				'smth is wrong
				Rhino.Print "Null Spine, this should
never happen"
				Exit Sub
			End If

		Next
	End Sub

	'are the cell pts coplanar??
	Private Function CellsCoplanar(arrCells)

		Dim i, arr3dPts(), strPt

		For i=0 To Ubound(arrCells)
			'get pts
			ReDim Preserve arr3dPts(i)
			strPt = arrCells(i).m_strPt
			arr3dPts(i)=Rhino.PointCoordinates(strPt)

		Next
```



```
		'return true if pts are coplanar
		CellsCoplanar = Rhino.PointsAreCoplanar(arr3dPts, 1)

	End Function

	'returns polyline from ordered cell array----------------
	Private Function StrokeCells(arrCells)

		Dim i, arr3dPts(), strPt, strShortSpine, strLongSpine

		For i=0 To Ubound(arrCells)
			'get pts for next snake
			ReDim Preserve arr3dPts(i)
			strPt = arrCells(i).m_strPt
			arr3dPts(i)=Rhino.PointCoordinates(strPt)
			'label joints
			Call Rhino.AddText(CStr(i), arr3dPts(i))

		Next

		'create polyline
		If Not IsEmpty(arr3dPts) Then
			'create spine
			strShortSpine = Rhino.AddPolyline(arr3dPts)
			'extend curve to accound for snub ends
			strLongSpine = Rhino.ExtendCurveLength(strShortS
pine, 0, 2, 2.5)
			'assign to snake
			'm_arrSnakes(i).m_strSpine = strLongSpine
		Else
			Rhino.Print "arr3dPts is empty"
			StrokeCells = Null
		End If

		'return polyline
		StrokeCells = strLongSpine

	End Function

	Public Sub SkinSnakes(dblRadius)

		Dim i, strP0, strP1, arrP0, arrP1
		Dim arrNorm, arrPlane, arrPlane2, strCirc, arrBB
		Dim strCrossSect, strSpine, arrSweeps, strSkin

		For i=0 To Ubound(m_arrSnakes)

			'get orgin
			'strP0 = m_arrSnakes(i).m_arrCells(0).m_strPt
			'strP1 = m_arrSnakes(i).m_arrCells(1).m_strPt
			'arrP0 = Rhino.PointCoordinates(strP0)
			'arrP1 = Rhino.PointCoordinates(strP1)
			'get normal vector
			'arrNorm = Rhino.VectorCreate (arrP0, arrP1)
			'create unit plane from normal vector
			'arrPlane = Rhino.PlaneFromNormal (arrP0, ar-
rNorm)

			'get spine
			strSpine = m_arrSnakes(i).m_strSpine
			'curve frame
			arrPlane2 = Rhino.CurvePerpFrame(strSpine,
-2.45)

			If Not IsArray(arrPlane2) Then
				Rhino.Print TypeName(arrPlane2)
				Rhino.ObjectColor m_strSpine, RGB(0,
255, 0)
			End If
			'add cirlcle
			strCirc = Rhino.AddCircle (arrPlane2, dblRadius)
			'get bounding box of circle
			arrBB = Rhino.BoundingBox(strCirc, arrPlane2)
			'create crosssection curv
			strCrossSect = Rhino.AddPolyline(arrBB)
			'sweep spine
			arrSweeps = Rhino.AddSweep1(strSpine, strCross-
Sect)
			'get skin guid
			'Rhino.Print "arrSweeps type: " +
TypeName(arrSweeps(0))
			'strSkin = arrSweeps(0)
			'assine skin to snake
			'm_arrSnakes(i).m_strSkin = strSkin
		Next
	End Sub
End Class

'-----------------------------------------------------
'SNAKE CLASS
'
'
'-----------------------------------------------------
Class Snake

	Public m_strSpine
	Public m_strSkin
	Public m_arrCells
	Public m_intSegs

	Public Sub Build(arrCells)

		Dim i

		m_arrCells = arrCells
		m_intSegs = Ubound(arrCells)+1

		'For i=0 To Ubound(arrCells)
		'	arrCells(i).m_boolIsOpen = False
		'Next
	End Sub

End Class

'-----------------------------------------------------
'[SNAKE]SKIN CLASS
'
'
'-----------------------------------------------------
Class Skin

	Private arrScales

End Class

'-----------------------------------------------------
'[SNAKE]SPINE CLASS
'
'
'-----------------------------------------------------
Class Spine

	'order array of pts
	Private arrCells

	'render snake method

End Class

'-----------------------------------------------------
'[SNAKE]SCALE CLASS
'
'
'-----------------------------------------------------
Class Scale

	Private strType
	Private oSrf
	'maybe pointers to neighboors (ids)
	'or shit maybe just reference the obj

End Class


Option Explicit
'Script written by Josh Parker
'Script copyrighted by OPEN Architecture
'Script version Tuesday, February 15, 2011 3:25:36 AM

Class Substrate

	Public m_arrCells
	Public m_intSpacing
	Public m_intSteps
	Public m_intLevels

	'subsets
	Public m_arrType0
	Public m_arrType1
```

```
Public m_arrType2

Private Sub Class_Initialize
	'Rhino.Print "Initialize substrate"
End Sub

Private Sub Class_Terminate
	'Rhino.Print "Terminate substrate"
End Sub

'seed substrate-------------------------------------
Public Sub Build(intSpacing, intSteps, intLevels)

	m_intSpacing = intSpacing
	m_intSteps = intSteps
	m_intLevels = intLevels

End Sub

'create random-type cell array from points-----------
Public Sub AddCells(arrPts)
	Rhino.Print "Adding Cells"

	Dim newCells(), i

	If IsNull(arrPts) Then
		Rhino.Print "AddCells arg, arrPts, is null"
		Exit Sub
	End If

	For i=0 To Ubound(arrPts)
		'incr cell array by 1
		ReDim Preserve newCells(i)
		'create new cell i
		Set newCells(i) = New Cell
		'cell "constructor"
		newCells(i).Build arrPts(i), RndInt(0,2)
	Next

	'"copy" array
	m_arrCells = newCells

End Sub

'create random-type cell array from points-----------
Public Sub AddCellsByColor(arrPts)
	Rhino.Print "Adding Cells By color"

	Dim newCells(), i, intType

	If IsNull(arrPts) Then
		Rhino.Print "AddCells arg, arrPts, is null"
		Exit Sub
	End If

	For i=0 To Ubound(arrPts)
		'incr cell array by 1
		ReDim Preserve newCells(i)
		'create new cell i
		Set newCells(i) = New Cell
		'cell "constructor"
		Call newCells(i).Build(arrPts(i),
TypeFromColor(arrPts(i)))
	Next

	'"copy" array
	m_arrCells = newCells

End Sub

'connect to neighbors-----------------------------------
Public Sub ConnectCells()
	Rhino.Print "Connecting Cells"

	Dim i
	For i=0 To Ubound(m_arrCells)
		Rhino.Print "Connecting Cells " + CStr(i) + " of
" + CStr(Ubound(m_arrCells))
		'connect cell to its neighbors
		m_arrCells(i).Connect m_arrCells, 5
	Next

End Sub

'Group cells by type-------------------------------------
Public Sub GroupCells()
	Rhino.Print "Grouping Cells"

	Dim type0(), type1(), type2()
	Dim i, a, b, c
	a=0
	b=0
	c=0

	'Rhino.Print "m_arrCells Type: " + TypeName(m_arrCells)
	'Rhino.Print "m_arrCells Size: " + CStr(Ubound(m_ar-
rCells))
	For i=0 To Ubound(m_arrCells)
		'group by type
		'Rhino.Print "Select Type: " + CStr(m_
arrCells(i).m_intType)
		Select Case m_arrCells(i).m_intType
			Case 0
				ReDim Preserve type0(a)
				Set type0(a) = m_arrCells(i)
				a=a+1
			Case 1
				ReDim Preserve type1(b)
				Set type1(b) = m_arrCells(i)
				b=b+1
			Case 2
				ReDim Preserve type2(c)
				Set type2(c) = m_arrCells(i)
				c=c+1
			Case Else
				Rhino.Print "GroupCells() found
bad cell"
		End Select
	Next

	'copy if smth to copy, else leave null
	If a > 0 Then m_arrType0 = type0
	If b > 0 Then m_arrType1 = type1
	If c > 0 Then m_arrType2 = type2

	'Rhino.Print "m_arrType0: " + CStr(Ubound(m_arrType0))
	'Rhino.Print "m_arrType1: " + CStr(Ubound(m_arrType1))
	'Rhino.Print "m_arrType2: " + CStr(Ubound(m_arrType2))

End Sub

'color substrate by type---------------------------------
Public Sub ColorByType()
	Rhino.Print "Coloring Cells"

	Dim i, color
	For i=0 To Ubound(m_arrCells)
		'color pt by type
		Call m_arrCells(i).UpdateColor()
	Next
End Sub

'-------------------------------------------------------------
Public Sub Apply(oRuleSet, intSteps)
	Rhino.Print "Applying ruleset"

	Dim i, j, iRand
	For j=0 To intSteps
		'this should be replaced  by random cell picker
		For i=0 To Ubound(m_arrCells)
			'get point(string id) at random
			iRand=RndInt(0,Ubound(m_arrCells))
			'color pt by type
			Call m_arrCells(iRand).
Apply(oRuleSet)
			Next

		Next

		'done so regroup cells by type
		GroupCells()

End Sub

'-------------------------
'----------------SCRAP--
'-------------------------

'return reference to random cell
Public Function RndCellold(intType)

	RndCellold = m_arrCells(RndInt(0, Ubound(m_arrCells)))
```



```
	End Function

	'return reference to random cell
	Public Function RndType0old()

		RndType0old = m_arrType0(RndInt(0, Ubound(m_arrType0)))

	End Function

	'return reference to random open type0 cell
	'this function will eventually have to timeout
	Public Function RndOpenType0old()

		Dim c
		Set c = m_arrType0(RndInt(0, Ubound(m_arrType0)))
		If c.m_boolIsOpen Then
			'return c
			RndOpenType0old = c
		Else
			'not found try again
			RndOpenType0old = RndOpenType0()
		End If

	End Function

End Class
Option Explicit
'Script written by Josh Parker
'Script copyrighted by OPEN Architecture
'Script version Tuesday, February 15, 2011 3:25:36 AM

Call Main()
Sub Main()

	Dim arrPts, intSteps, intLevels, intSpacing, oMajRule
	Dim oSubstr, oSnakeFarm

	'get initial data
	arrPts=Rhino.GetObjects("select points to populate",1)
	If Not IsArray(arrPts) Then
		Rhino.Print "no points selected"
		Exit Sub
	End If
	intSpacing=Rhino.GetInteger("enter cell spacing",5)
	intSteps=Rhino.GetInteger("enter number of steps",2)
	intLevels=Rhino.GetInteger("enter number of levels",2)

	'create substrate
	Set oSubstr = New Substrate
	'substrate "constructor"
	Call oSubstr.Build(intSpacing, intSteps, intLevels)
	'add Cells from points
	Call oSubstr.AddCells(arrPts)
	'link each cell to its neighbors
	Call oSubstr.ConnectCells()


	'create cellular automata obj
	Set oMajRule = New MajorityRule
	'apply ruleset to each cell x times
	Call oSubstr.Apply(oMajRule, 1)
	'color cell pts by cell type
	Call oSubstr.ColorByType()

	'create snakes
	Set oSnakeFarm = New SnakeFarm
	'locate snakes in substrate
	Call oSnakeFarm.FindSnakes(oSubstr, 4)
	'render snakes?
	Call oSnakeFarm.RenderSnakes()

	'create massing
	'Set oMass = New massing
	'build mass from snake formation
	'oMass.build(oSnakes)


End Sub
Option Explicit
'Script written by <insert name>
'Script copyrighted by <insert company name>
'Script version Wednesday, February 16, 2011 8:20:37 PM

Function RndInt(lowest, highest)
	Randomize
	'RndInt=Int((highest - lowest) * Rnd + lowest)
	RndInt=Int((highest - lowest + 1) * Rnd + lowest)

End Function

'return reference to random open type0 cell
'this function will eventually have to time-
out--------------------------
Function OpenCells(arrCells)

	Dim i, j, arrOpenCells()

	'check arg is valid non-empty array
	If IsNull(arrCells) Then
		Rhino.Print "OpenCells arg is null"
		OpenCells = Null
		Exit Function
	ElseIf Not IsArray(arrCells) Then
		Rhino.Print "OpenCells arg is not an array, its a: " +
TypeName(arrCells)
		OpenCells = Null
		Exit Function
	End If

	j=0
	For i=0 To Ubound(arrCells)
		'if open, add to array
		If arrCells(i).m_boolIsOpen Then
			ReDim Preserve arrOpenCells(j)
			Set arrOpenCells(j) = arrCells(i)
			j=j+1
		End If
	Next

	'return subsets if any open, else null
	If j>0 Then
		'Rhino.Print "OpenCells found some open cells in array"
		OpenCells = arrOpenCells
	Else
		Rhino.Print "OpenCells found no open cells in array"
		OpenCells = Null
		Exit Function
	End If

End Function

'return reference to random open type0 cell-------------------------
Function RndCell(arrCells)

	'check arg is valid non-empty array
	If IsNull(arrCells) Then
		Rhino.Print "RndCell arg is null"
		RndCell = Null
		Exit Function
	ElseIf Not IsArray(arrCells) Then
		Rhino.Print "RndCell arg is not an array, its a: " +
TypeName(arrCells)
		RndCell = Null
		Exit Function
	End If

	'arg is non-zero sized array to return random value from it
	Set RndCell = arrCells(RndInt(0, Ubound(arrCells)))

End Function

'return reference to random open type0 cell-------------------------
Function TypeFromColor(strPt)

	Dim c, typ

	c = Rhino.ObjectColor(strPt)

	'set color var by cell type
	Select Case c
		Case RGB(255, 0, 0)
			typ = 0
		Case RGB(0, 255, 0)
			typ = 1
		Case RGB(0, 0, 255)
			typ = 2
		Case Else
			Rhino.Print "pt is not valid color: " + c
			Exit Function
```

```
	End Select

	'return type
	TypeFromColor = typ
End Function

'------------------------
'----------------SCRAP--
'------------------------

'If you need to repeat a sequence of random numbers,
'you should call the Rnd function with a negative
'number as an argument immediately prior To using
'Randomize With any numeric argument.
Sub RepeatNumbers()
	Dim arr(9, 3)
	Dim loopCtr, intCtr
	Dim strMsg

	For loopCtr = 0 To 3
		Rnd -1
		Randomize(100)
		For intCtr = 0 To 9
			strMsg = strMsg & Rnd() & " "
		Next
		strMsg = strMsg & vbCrLf
	Next

	MsgBox strMsg
End Sub

'return reference to random open type0 cell
'this function will eventually have to timeout
Function RndOpenCell(arrCells)

	Dim c
	Set c = arrCells(RndInt(0, Ubound(arrCells)))
	If c.m_boolIsOpen Then
		'return c
		RndOpenCell = c
	Else
		'not found try again
		RndOpenCell = RndOpenCell(arrCells)
	End If

End Function
Option Explicit
'Script written by Josh Parker
'Script copyrighted by OPEN Architecture
'Script version Tuesday, February 15, 2011 3:25:36 AM

Class Village

	Private arrStructure
	Private arrTrees
	Private arrGrass
	Private arrPave

End Class

Class Snake

	Private oSpine
	Private oSkin

End Class

Class Skin

	Private arrScales

End Class

Class Spine

	'order array of pts
	Private arrCells

	'render snake method

End Class

Class Scale

	Private strType
```



```
	Private oSrf
	'maybe pointers to neighboors (ids)
	'or shit maybe just reference the obj

End Class
```

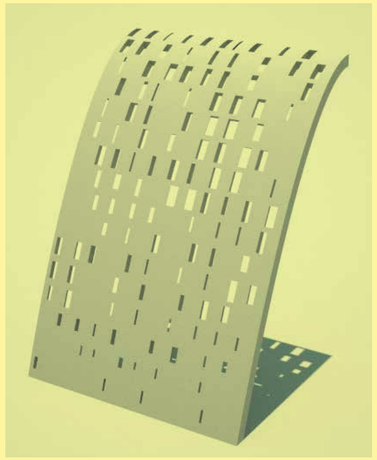
UPLIFT HANGER, SOURCE CODE

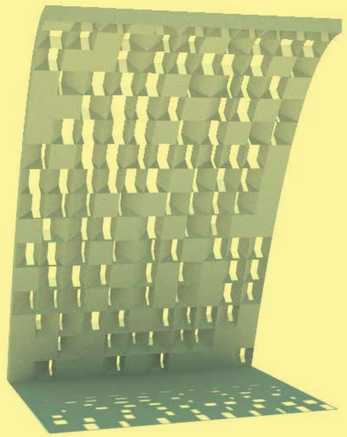
Code written for Interactive paneling tool developed for use in design of structurally integrated prefabricated concrete panel system for design competition of airship hanger in anhui, china for competition submission completed in collaboration with OPEN Architecture and the Chinese Academy of Building Research (CABR)

Uses the following processing libraries:

controlP5 - gui toolkit
- http://www.sojamo.de/libraries/controlP5/
Proscene - 3d scene library
- http://code.google.com/p/proscene/
0bjloader - 3d object loader
- http://code.google.com/p/saitoobjloader/

test sections of concrete massing generated by overlaying algorithmic patterns on top of solar data gathered from eco-tect.






```
/**************************************************

Aper[na]tures
Author: Joshua Parker
Date: 2011

Interactive paneling tool developed for use in
design of structurally integrated prefabricated
concrete panel system for design competition
of airship hanger in anhui, china for competition
submission completed in collaboration with OPEN
Architecture and the Chinese Academy of Building
Research (CABR)

Implementated with Simple Model-View-Controller
(MVC) pattern:

Gui - is a wrapper for controlP5, which supplies
  2d user interface toolkit
MyControlListener - recieves ui events, ids and
  routes them to controller
Controller - contains methods for updating data
  model & controlling view(s)*
View - contains the scene(empty space and camera),
  imported 3d base model, and data model defining
  panel system. View calls draws to screen via
  viewState interface, though only one viewState
  is implemented: View3d, a basic 3d view.
Hanger - data model defining panel system, composed
  of subclass skin, which is composed of panels.
  Panels are basically two sets of parameters: one
  which defines the panel's size and position in
  space, and another that defines the size and
  relative position of a single rectangular void
  in the panel. Panel also contains data from ecotect
  about the amount of direct and diffuse radiation
  each panel recieves.

Uses the following processing libraries:

controlP5 - gui toolkit
- http://www.sojamo.de/libraries/controlP5/
Proscene - 3d scene library
- http://code.google.com/p/proscene/
0bjloader - 3d object loader
- http://code.google.com/p/saitoobjloader/

Inheritted License: GPL, V3

Fourth Natures Lab | www.fnl.com

**************************************************/
import processing.core.*;
import controlP5.*;
import processing.opengl.*;
import controller.Listener;
import controller.Controller;
import model.Hanger;
import view.*;
import gui.*;

public class Apernatures extends PApplet{

        private static final long serialVersionUID = 1L;

        //CONSTANTS
        int U = 37; //number of modules in U direction
        int V = 42; //number of modules in V direction
        float D = 10f; //panel lengh/width dimension, D
(meters)
        float T = .5f; //thickness, T of frame (meters)
        float SUBDIVS = 20f; //subdivide panel for snapping

        //path to tabular data from ecotect analyses
        String PATH = "data/ecotectData.txt";

        //objects
        Listener myListener;
        Controller myController;
        Hanger myhanger;
        View myview;
        Gui mygui;

        public void setup(){
          size(1280, 1020, PGraphicsOpenGL.OPENGL);
          background(0);
          noStroke();
          //smooth();

          //MODEL: Parametric Data for panel system
          float snap = (float)((D-(T*2))/SUBDIVS);
          myhanger = new Hanger(this, U, V, D, T, snap,
PATH);

          //VIEW: draws a representation of model to
screen
          myview = new View(this, myhanger);

          //GUI: needs to access view scene for drawing 2d
gui on top of 3d scene
          mygui = new Gui(this, myview);

          //CONTROLLER: updates model & views in response
to ui events routed to it
          myController = new Controller(this, myhanger,
mygui, myview);

          //LISTENER/ROUTER: recieves ui events, ids and
routes them to controller
          myListener = new Listener(this, myController);

        }

        public void draw(){
          background(50);

          //Continually redraw view and gui via controller
          myController.update();
        }

        //required to run as application
        public static void main(String args[]) {
          PApplet.main(new String[] { "--present", "Apernat-
ures" });
        }

        //send ui events to listener, so it route them to
controller
        public void controlEvent(ControlEvent theEvent)
{myListener.controlEvent(theEvent);}
        public void mousePressed(){myListener.
mousePressed(mouseX, mouseY);}
        public void keyPressed(){myListener.
keyPressed(key);}

}

/**************************************************

Controller Class

MVC Controller contains methods for updating data
model & controlling view(s)*

Part of:

Aper[na]tures
Author: Joshua Parker
Date: 2011

Inheritted License: GPL, V3

**************************************************/
package controller;

import processing.core.PApplet;
import processing.core.PVector;
import model.Hanger;
import view.View;
import gui.Gui;

public class Controller{

        //The parent PApplet
        PApplet p;

        //PROPS
        private Hanger myhanger;
        private View myview;
        private Gui mygui;

        //CONSTRUCTOR
        public Controller(PApplet parent, Hanger Myhanger,
Gui Mygui, View Myview){
                p = parent;
          myhanger = Myhanger;
          mygui = Mygui;
          myview = Myview;
        }

        // PUBLIC METHODS /////////////////////////////////
//////////////////

        //gui noise increment
        public void setNoiseInc(float v){
                myhanger.setInc(v);
        }

        //set noise seed
        public void setNoiseSeed(int v){
                myhanger.setSeed(v);
        }

        //set coefficient with index i
        public void setPanelXcf(int i, float C){
                myhanger.setXcf(i, C);
        }

        //
        public void moveCameraByTheta(float v){
        PVector v0;
        float r = 500;
        float phi = 0;
        float theta = 0;
        //r = myview.scene.camera().sceneRadius();
        //r = 500;
        v0 = myview.scene.camera().position();
        phi = (float)Math.atan(v0.y/v0.x);
        theta = PApplet.radians(v); //azimuth 0-2PI
        setCameraPosition(phi, theta, r);
          }

          //
          public void moveCameraByPhi(float v){
        PVector v0;
        float r = 500;
        float phi = 0;
        float theta = 0;
        //r = myview.scene.camera().sceneRadius();
        //r = 500;
        v0 = myview.scene.camera().position();
        theta = (float)Math.acos(v0.z/r);
        phi = PApplet.radians(v); //azimuth 0-2PI
        setCameraPosition(phi, theta, r);
          }

          //
          public void setCameraPosition(float phi, float
theta, float r){
                float x = r*(PApplet.sin(theta)*PApplet.
cos(phi));
                float y = r*(PApplet.sin(theta)*PApplet.
sin(phi));
                float z = r*PApplet.cos(theta);

                PVector v = new PVector(x,y,z);
                PVector v2 = new PVector(0,0,0);
                PVector v3 = new PVector(0,0,-1);

                myview.scene.camera().setPosition(v);
                myview.scene.camera().setOrientation(theta,
phi);
                myview.scene.camera().setUpVector(v3,true);
                myview.scene.camera().lookAt(v2);
          }

          //
          public void focusGui(){
                myview.scene.disableMouseHandling();
                myview.scene.disableKeyboardHandling();
          }

          //
          public void focusScene(){
                myview.scene.enableMouseHandling();
                myview.scene.enableKeyboardHandling();
          }
```

```
  //
  public void toggleTexture(){
if(!myview.bTexture){
  showTexture();
} else {
  hideTexture();
}
  }

  //
  public void toggleMaterial(){
          if(!myview.bMaterial) {
                  showMaterial();
          } else {
                  hideMaterial();
          }
  }

  //
  public void toggleFill(){
          if(!myview.getFill()) {
                  showFill();
          } else {
                  hideFill();
          }
  }

  //
  public void showTexture(){
          myview.model.enableTexture();
          myview.bTexture = true;
  }

  //
  public void hideTexture(){
          myview.model.disableTexture();
          myview.bTexture = false;
  }

  //
  public void showMaterial(){
          myview.model.enableMaterial();
          myview.bMaterial = true;
  }

  //
  public void hideMaterial(){
          myview.model.disableMaterial();
          myview.bMaterial = false;
  }

  //
  public void showFill(){
          myview.enableFill();
  }

  //
  public void hideFill(){
          myview.disableFill();
  }

  //
  public void update(){
          myhanger.update(); //update the model
          myview.draw(); // draw the 3d scene
          mygui.draw(); // then draw the gui on
top
  }
}
/*************************************************

Listener Class

Listener recieves ui events, parses them and routes
them to controller for handling.

Part of:

Aper[na]tures
Author: Joshua Parker
Date: 2011

Inheritted License: GPL, V3

*************************************************/
package controller;

import controlP5.ControlEvent;
import controlP5.ControlListener;
import processing.core.PApplet;

public class Listener implements ControlListener{

          //The parent PApplet
          PApplet p;

          //PROPS
          Controller myController;

          //CONSTRUCTOR
          public Listener(PApplet parent, Controller MyCon-
troller){
                  p = parent;
                  myController = MyController;
          }

          // PUBLIC METHODS ////////////////////////////////
////////////////////

          //route slider events to controller via private
handlers
          public void controlEvent(ControlEvent theEvent) {

                  switch(theEvent.getController().getPar-
ent().getId()){
                  case(1):
                          //route g1 (shortcut info)
event!
                          //empty so break
                          break;
                  case(2):
                          //route g2 (noise sliders)
event!
                          g2ControlEvent(theEvent);
                  break;
                  case(3):
                          //route g3 (effect sliders)
event!
                          g3ControlEvent(theEvent);
                  break;
                  case(4):
                          //route g4 (camera dials)
event!
                          g4ControlEvent(theEvent);
                  break;
                  }
          }

          //route mouse events to controller
          //this should be replaced with focus on hover
          public void mousePressed(int mx, int my) {
                  if (mx < 200) {
                          myController.focusGui();
                  } else {
                          myController.focusScene();
                  }
          }

          //route key press events to controller
          public void keyPressed(char key) {

                  if(key == 't') {
                          //toggle the texture listed in
.mtl file
                          myController.toggleTexture();
                  }
                  else if(key == 'm') {
                          //toggle the material listed
in .mtl file
                          myController.toggleMaterial();
                  }
                  else if(key == 'f') {
                          //toggle the fill for new
shapes
                          myController.toggleFill();
                  }

          // PRIVATE METHODS ////////////////////////////////
////////////////////

          //parse group 2 (noise sliders) event!
*****************************

          private void g2ControlEvent(ControlEvent theEvent)
{

                  float v;
                  switch(theEvent.getController().getId())
{
                  case(1):
                          v = theEvent.getController().
getValue();
                  myController.setNoiseInc(v);
                  break;
                  case(2):
                          v = theEvent.getController().
getValue();
                  myController.setNoiseSeed((int)(v));
                  break;
                  }
          }

          //parse group 3 (effect sliders) event!
*****************************
          private void g3ControlEvent(ControlEvent theEvent)
{

                  float v;
                  switch(theEvent.getController().getId())
{
                  case(5):
                          v = theEvent.getController().
getValue();
                  myController.setPanelXcf(0,v);
                  break;
                  case(6):
                          v = theEvent.getController().
getValue();
                  myController.setPanelXcf(1,v);
                  break;
                  case(7):
                          v = theEvent.getController().
getValue();
                  myController.setPanelXcf(2,v);
                  break;
                  case(8):
                          v = theEvent.getController().
getValue();
                  myController.setPanelXcf(3,v);
                  break;
                  case(9):
                          v = theEvent.getController().
getValue();
                  myController.setPanelXcf(4,v);
                  break;
                  case(10):
                          v = theEvent.getController().
getValue();
                  myController.setPanelXcf(5,v);
                  break;
                  case(11):
                          v = theEvent.getController().
getValue();
                  myController.setPanelXcf(6,v);
                  break;
                  case(12):
                          v = theEvent.getController().
getValue();
                  myController.setPanelXcf(7,v);
                  break;
                  case(13):
                          v = theEvent.getController().
getValue();
                  myController.setPanelXcf(8,v);
                  break;
                  case(14):
                          v = theEvent.getController().
getValue();
                  myController.setPanelXcf(9,v);
                  break;
                  case(15):
                          v = theEvent.getController().
getValue();
                  myController.setPanelXcf(10,v);
                  break;
                  case(16):
                          v = theEvent.getController().
getValue();
                  myController.setPanelXcf(11,v);
                  break;
                  case(17):
                          v = theEvent.getController().
getValue();
                  myController.setPanelXcf(12,v);
                  break;
                  case(18):
                          v = theEvent.getController().
getValue();
                  myController.setPanelXcf(13,v);
                  break;
                  case(19):
                          v = theEvent.getController().
getValue();
                  myController.setPanelXcf(14,v);
                  break;
                  case(20):
                          v = theEvent.getController().
getValue();
                  myController.setPanelXcf(15,v);
                  break;
                  case(21):
                          v = theEvent.getController().
getValue();
                  myController.setPanelXcf(16,v);
                  break;
                  case(22):
                          v = theEvent.getController().
getValue();
                  myController.setPanelXcf(17,v);
                  break;

                  }
          }

          //parse group 4 (camera dials) event! ***********
*********************
          private void g4ControlEvent(ControlEvent theEvent)
{

                  float v;
                  switch(theEvent.getController().getId())
{
                  case(13):
                          v = theEvent.getController().
getValue();
                  myController.moveCameraByPhi(v);
                  break;
                  case(14):
                          v = theEvent.getController().
getValue();
                  myController.moveCameraByTheta(v);
                  break;
                  }
          }

}

/*************************************************

Gui Class

Gui is a wrapper for controlP5 library, which
provides user interface toolkit. more at:
http://www.sojamo.de/libraries/controlP5/

Part of:

Aper[na]tures
Author: Joshua Parker
Date: 2011

Inheritted License: GPL, V3

*************************************************/
package gui;

import processing.core.PApplet;
import processing.core.PFont;
import controlP5.*;
import remixlab.proscene.*;
import view.*;

public class Gui
{
          //CONSTANTS
          private static final int DISABLE_DEPTH_TEST = 4;
          private static final int ENABLE_DEPTH_TEST = -4;

          //parent PApplet
          PApplet p;

          //cp5 objs
          private ControlP5 cp5;
          private Accordion accordion;

          //composit view
          private View myview;

          //CONSTRUCTOR
          public Gui(PApplet parent, View Myview){

                  //initialize
                  p = parent;
                  cp5 = new ControlP5(p);
                  cp5.setAutoDraw(false);
                  myview = Myview;

                  //setup gui elements
                  setupP5Gui();
          }

          // PUBLIC METHODS ////////////////////////////////
////////////////////

          public void draw() {

                  //save scene (so you can draw 2d gui on
top)
                  saveState(myview.scene);

                  //draw gui
                  cp5.draw();

                  //restore scene
                  restoreState(myview.scene);
          }

          // PRIVATE METHODS ////////////////////////////////
////////////////////

          //setup p5 gui elements
          private void setupP5Gui(){

                  int BarH = 20;
                  int BgClr = p.color(50); //bg of block
                  int BarClr = p.color(50); //bar color
                  int HoverClr = 30; //bar color on hover
                  int LabelClr = p.color(255);

                  int slideH = 10;
                  int slideW = 100;
                  int slideX = 10;
                  int slideSpc = 5;
                  int slideDY = slideH+slideSpc;

                  PFont arial = p.createFont("ArialMT-30",
11);
                  PFont arial10 =
p.createFont("ArialMT-30", 10);

                  //add footer
                  cp5.addTextlabel("label")
                                  .setText("by Fourth
Natures Lab | www.fnl.com")
                                  .setPosition(2,p.
height - 20)
                                  .setFont(arial10)
                                  .setHeight(20)
                                  .setWidth(200)
                                  ;

                  // ADD GROUPS ************************
*************************************************

                  // group 1 (shortcut info)
                  controlP5.Group g1 = cp5.
addGroup("myGroup1")
                                  .setId(1).
setLabel("Aper[na]ture")
                                  .setBackground-
Height(80)
                                  .setBarHeight(BarH)
.setBackgroundColor(BgClr)
                                  .setColorBackground(BarClr)
                                  .setColorForeground(HoverClr)
                                  .setColorLabel(LabelClr)
                                  ;

                  // group 2 (noise sliders)
                  controlP5.Group g2 = cp5.
addGroup("myGroup2")
                                  .setId(2)
                                  .setLabel("Noise")
                                  .setBackground-
Height(40)
                                  .setBarHeight(BarH)
.setBackgroundColor(BgClr) //bg of block
.setColorBackground(BarClr) //bar color
.setColorForeground(HoverClr) //bar color on hover
.setColorLabel(LabelClr) //label text color
                                  ;

                  // group 3 (effect sliders)
                  controlP5.Group g3 = cp5.
addGroup("myGroup3")
                                  .setId(3)
                                  .setLabel("x = f()")
                                  .setBackground-
Height(300)
                                  .setBarHeight(BarH)
.setBackgroundColor(BgClr) //bg of block
.setColorBackground(BarClr) //bar color
.setColorForeground(HoverClr) //bar color on hover
.setColorLabel(LabelClr) //label text color
                                  ;

                  // group 4 (camera dials)
                  controlP5.Group g4 = cp5.
addGroup("myGroup4")
                                  .setId(4)
                                  .setLabel("view")
                                  .setBackground-
Height(200)
                                  .setBarHeight(BarH)
.setBackgroundColor(BgClr) //bg of block
.setColorBackground(BarClr) //bar color
.setColorForeground(HoverClr) //bar color on hover
.setColorLabel(LabelClr) //label text color
                                  ;

                  // ADD CONTROLLERS: sliders and dials **
*************************************************

                  //add group 1 controllers

                  cp5.addTextarea("txt")
                  .moveTo(g1)
                  .setPosition(5,5)
                  .setSize(195,90)
                  .setFont(arial)
                  .setLineHeight(14)
                  .setColor(p.color(128))
                  .setColorBackground(p.color(50))
                  .setColorForeground(p.color(255,100))
                  .setText("shortcuts: \n"
                                  +"f - toggle fills
\n"
                                  +"e - toggle per-
spective \n"
                                  +"c - close controls
\n"
                                  +"o - open controls
\n");
```

```
//add group 2 controllers

cp5.addSlider("noise inc")
.setId(1)
.moveTo(g2)
.setPosition(slideX,slideDY*1)
.setSize(slideW,slideH)
.setRange(0.0f,1.0f)
.setValue(.4f);

cp5.addSlider("noise seed")
.setId(2)
.moveTo(g2)
.setPosition(slideX,slideDY*2)
.setSize(slideW,slideH)
.setRange(0,50)
.setValue(8);

//add group 3 controllers

cp5.addSlider("total radiation")
.setId(5)
.moveTo(g3)
.setPosition(slideX,slideDY*1)
.setSize(slideW,slideH)
.setRange(0,10)
.setValue(0);

cp5.addSlider("direct radiation")
.setId(6)
.moveTo(g3)
.setPosition(slideX,slideDY*2)
.setSize(slideW,slideH)
.setRange(0,10)
.setValue(0);

cp5.addSlider("diffuse radiation")
.setId(7)
.moveTo(g3)
.setPosition(slideX,slideDY*3)
.setSize(slideW,slideH)
.setRange(0,10)
.setValue(0);

cp5.addSlider("by grade")
.setId(8)
.moveTo(g3)
.setPosition(slideX,slideDY*4)
.setSize(slideW,slideH)
.setRange(0,10)
.setValue(0);

cp5.addSlider("by elevation")
.setId(9)
.moveTo(g3)
.setPosition(slideX,slideDY*5)
.setSize(slideW,slideH)
.setRange(0,10)
.setValue(0);

cp5.addSlider("noise")
.setId(10)
.moveTo(g3)
.setPosition(slideX,slideDY*6)
.setSize(slideW,slideH)
.setRange(0,20)
.setValue(0);

cp5.addSlider("x scale")
.setId(11)
.moveTo(g3)
.setPosition(slideX,slideDY*7)
.setSize(slideW,slideH)
.setRange(-1.0f,1.0f)
.setValue(1.0f)
.setNumberOfTickMarks(21)
.snapToTickMarks(true)
.showTickMarks(false);

cp5.addSlider("x offset")
.setId(12)
.moveTo(g3)
.setPosition(slideX,slideDY*8)
.setSize(slideW,slideH)
.setRange(-1,1)
.setValue(.1f)
.setNumberOfTickMarks(100)
.snapToTickMarks(true)
.showTickMarks(false);

cp5.addSlider("x trim")
.setId(13)
.moveTo(g3)
.setPosition(slideX,slideDY*9)
.setSize(slideW,slideH)
.setRange(0,3)
.setValue(0)
.setNumberOfTickMarks(11)
.snapToTickMarks(true)
.showTickMarks(false);

cp5.addSlider("w scale")
.setId(17)
.moveTo(g3)
.setPosition(slideX,slideDY*13)
.setSize(slideW,slideH)
.setRange(-3,3)
.setValue(1)
.setNumberOfTickMarks(100)
.snapToTickMarks(true)
.showTickMarks(false);

cp5.addSlider("w offset")
.setId(18)
.moveTo(g3)
.setPosition(slideX,slideDY*14)
.setSize(slideW,slideH)
.setRange(-1,1)
.setValue(0)
.setNumberOfTickMarks(100)
.snapToTickMarks(false)
.showTickMarks(false);

cp5.addSlider("y scale")
.setId(14)
.moveTo(g3)
.setPosition(slideX,slideDY*10)
.setSize(slideW,slideH)
.setRange(-3,3)
.setValue(1)
.setNumberOfTickMarks(100)
.snapToTickMarks(false)
.showTickMarks(false);

cp5.addSlider("y offset")
.setId(15)
.moveTo(g3)
.setPosition(slideX,slideDY*11)
.setSize(slideW,slideH)
.setRange(-1,1)
.setValue(.1f)
.setNumberOfTickMarks(100)
.snapToTickMarks(true)
.showTickMarks(false);

cp5.addSlider("y trim")
.setId(16)
.moveTo(g3)
.setPosition(slideX,slideDY*12)
.setSize(slideW,slideH)
.setRange(0,3)
.setValue(0)
.setNumberOfTickMarks(11)
.snapToTickMarks(true)
.showTickMarks(false);

cp5.addSlider("h scale")
.setId(20)
.moveTo(g3)
.setPosition(slideX,slideDY*16)
.setSize(slideW,slideH)
.setRange(-3,3)
.setValue(1)
.setNumberOfTickMarks(100)
.snapToTickMarks(true)
.showTickMarks(false);

cp5.addSlider("h offset")
.setId(21)
.moveTo(g3)
.setPosition(slideX,slideDY*17)
.setSize(slideW,slideH)
.setRange(-1,1)
.setValue(0)
.setNumberOfTickMarks(100)
.snapToTickMarks(true)
.showTickMarks(false);

cp5.addSlider("h trim")
.setId(22)
.moveTo(g3)
.setPosition(slideX,slideDY*18)
.setSize(slideW,slideH)
.setRange(0,3)
.setValue(0)
.setNumberOfTickMarks(11)
.snapToTickMarks(true)
.showTickMarks(false);

//add group 4 controllers

cp5.addKnob("azimuth")
.setId(13)
.moveTo(g4)
.setPosition(20,20)
.setRadius(35)
.setAngleRange(2*PApplet.PI)
.setStartAngle((float)(.5*PApplet.PI))
.setRange(0,360)
.setValue(0)
.setNumberOfTickMarks(16)
.snapToTickMarks(true)
.setColorForeground(p.color(255))
.setColorBackground(p.color(40))
.setColorActive(p.color(255,255,0))
.setDragDirection(Knob.HORIZONTAL);

cp5.addKnob("elevation")
.setId(14)
.moveTo(g4)
.setPosition(110,20)
.setRadius(35)
.setAngleRange(PApplet.PI)
.setStartAngle((float)(1.5*PApplet.PI))
.setRange(1,179)
.setValue(0)
.setNumberOfTickMarks(8)
.snapToTickMarks(true)
.setColorForeground(p.color(255))
.setColorBackground(p.color(40))
.setColorActive(p.color(255,255,0))
.setDragDirection(Knob.HORIZONTAL);


//Add Sidebar Accordion:  ************************************************
accordion = cp5.addAccordion("acc")
                        .setPosition(0,0)
                        .setWidth(200)
                        .addItem(g1)
                        .addItem(g2)
                        .addItem(g3)
                        .addItem(g4)
                        ;

//initially open all groups
accordion.open(0,1,2,3);

//allow multiple groups open at a time.
accordion.setCollapseMode(Accordion.MULTI);

//Add keyboard shortcuts ************************************
cp5.mapKeyFor(new ControlKey() {public void keyEvent() {accordion.open(0,1,2,3);}}, 'o');
cp5.mapKeyFor(new ControlKey() {public void keyEvent() {accordion.close(0,1,2,3);}}, 'c');

        }//end init()
```



```
        //function from https://forum.processing.org/topic/proscene-and-2d-drawing
        private void saveState(Scene scene) {

                // Disable depth test to draw 2d on top
                p.hint(DISABLE_DEPTH_TEST);

                //  Set processing projection and modelview matrices to draw in 2D:
                // 1. projection matrix:
                float cameraZ = ((p.height/2.0f) / PApplet.tan(PApplet.PI*60.0f/360.0f));
                scene.pg3d.perspective(
                                        PApplet.PI/3.0f,
                                        scene.camera().aspectRatio(),
                                        cameraZ/10.0f,
                                        cameraZ*10.0f);

                // 2 model view matrix
                scene.pg3d.camera();
        }

        //function from https://forum.processing.org/topic/proscene-and-2d-drawing
        private void restoreState(Scene scene) {

                // 1. Restore processing projection matrix
                switch (scene.camera().type()) {
                case PERSPECTIVE:
                        scene.pg3d.perspective(
                                                scene.camera().fieldOfView(),
                                                scene.camera().aspectRatio(),
                                                scene.camera().zNear(),
                                                scene.camera().zFar());
                        break;
                case ORTHOGRAPHIC:
                        float[] wh = scene.camera().getOrthoWidthHeight();
                        scene.pg3d.ortho(
                                                -wh[0], wh[0], -wh[1], wh[1],
                                                scene.camera().zNear(),
                                                scene.camera().zFar());
                        break;
                }
                // 2. Restore processing modelview matrix
                scene.pg3d.camera(
                                scene.camera().position().x,
                                scene.camera().position().y,
                                scene.camera().position().z,
                                scene.camera().at().x,
                                scene.camera().at().y,
                                scene.camera().at().z,
                                scene.camera().upVector().x,
                                scene.camera().upVector().y,
                                scene.camera().upVector().z);
                //Re-enble depth test
                p.hint(ENABLE_DEPTH_TEST);
        }

}
/*************************************************

Hanger Class

Hanger is data model defining panel system, composed
of subclass skin, which is composed of panels.
Panels are basically two sets of parameters: one
which defines the panel's size and position in
space, and another that defines the size and
relative position of a single rectangular void
in the panel. Panel also contains data from ecotect
about the amount of direct and diffuse radiation
each panel recieves.

Part of:

Aper[na]tures
Author: Joshua Parker
Date: 2011

Inheritted License: GPL, V3

 *************************************************/
package model;

import processing.core.PApplet;
import model.Skin;

public class Hanger{

        //The parent PApplet
        PApplet p;

        //composit skin
        public Skin myskin;

        //CONSTRUCTOR
        public Hanger(PApplet parent,
                        int U,
                        int V,
                        float D,
                        float T,
                        float snap,
                        String PATH){

                p = parent;

                //create skin
                myskin = new Skin(p, U, V, D, T, snap);

                //pass data path to skin so it can load data into panels
                myskin.loadData(PATH);

                //update skin pattern
                update();

        }//end constructor

        // PUBLIC METHODS //////////////////////////////////
        ////////////////////

        //set noise increment and update model
        public void setInc(float Inc){
                myskin.setInc(Inc);
        }

        //set noise seed and update model
        public void setSeed(int Seed){
                myskin.setSeed(Seed);
        }

        //set pattern coeffecients and update model
        public void setXcf(int i, float C){
                myskin.setXcf(i, C);
        }

        //update Hanger Model
        public void update(){
                //update skin pattern
                myskin.update();
        }

}
/*************************************************

Panel Class

Panel is subclass of Hanger data model defining panel
system, composed of subclass skin, which is composed
of panels.
Panels are basically two sets of parameters: one
which defines the panel's size and position in
space, and another that defines the size and
relative position of a single rectangular void
in the panel. Panel also contains data from ecotect
about the amount of direct and diffuse radiation
each panel recieves.

Part of:

Aper[na]tures
Author: Joshua Parker
Date: 2011

Inheritted License: GPL, V3

 *************************************************/
package model;

import processing.core.*;

public class Panel
{
        //The parent PApplet
        PApplet p;

        //panel properties
        public PVector[] coords;
        public float x,y,w,h;
//normalized
        public float px, py, pd;                //centerline of frame
        public float t;
//thickness of frame
        public float ix, iy, id;                //inside line, ie. panel
        public float hx, hy, hw, hh;    //perforations, ie. holes
        public float snap;
//panel subdivisions
        public int pClr;
//panel color

        //imported data
/*      private float avgDailyTotal;
        private float avgDailyDirect;
        private float avgDailyDiffuse;
        private float grade;
        private float elevation;*/

        //normalized parameters
        private float totalNorm;
        private float directNorm;
        private float diffuseNorm;
        private float gradeNorm;
        private float elevationNorm;
        public float noiseVal;

        //parameter weights
        private float[] xcf;

        //constructor
        Panel(PApplet parent,
                                float px,
                                float py,
                                float pd,
                                float ix,
                                float iy,
                                float id,
                                float t,
                                float snap){

                p = parent;

                //println("new panel");

                this.px = px;
                this.py = py;
                this.pd = pd;
                this.ix = ix;
                this.iy = iy;
                this.id = id;
                this.t = t;
                //hx = this.hx;
                //hy = this.hy;
                //hw = this.hw;
                //hh = this.hh;
                this.snap = snap;
```

```
                coords = new PVector[4];
                xcf = new float[18];

        }

        // PUBLIC METHODS /////////////////////////////////
////////////////////

        //load data into panel
        public void loadData(String avgDailyTotal,
                             String avgDailyDirect,
                             String avgDailyDiffuse,
                             String grade,
                             String elevation,
                             String totalNorm,
                             String directNorm,
                             String diffuseNorm,
                             String gradeNorm,
                             String elevationNorm){

/*                      this.avgDailyTotal = Float.
parseFloat(avgDailyTotal);
                        this.avgDailyDirect = Float.
parseFloat(avgDailyDirect);
                        this.avgDailyDiffuse = Float.
parseFloat(avgDailyDiffuse);
                        this.grade = Float.parseFloat(grade);
                        this.elevation = Float.
parseFloat(elevation);*/

                        this.totalNorm = Float.
parseFloat(totalNorm);
                        this.directNorm = Float.
parseFloat(directNorm);
                        this.diffuseNorm = Float.
parseFloat(diffuseNorm);
                        this.gradeNorm = Float.
parseFloat(gradeNorm);
                        this.elevationNorm = Float.
parseFloat(elevationNorm);

        }//end loadData()

        //
        public void setXcf(int i, float C){
                xcf[i]=C;
        }

        //find hole
        public void perforate(){

                //name coefficients
                float totalWeight = xcf[0];
                float directWeight = xcf[1];
                float diffuseWeight = xcf[2];
                float gradeWeight = xcf[3];
                float elevationWeight = xcf[4];
                float noiseWeight = xcf[5];
                float xScale = xcf[6];
                float xOff = xcf[7];
                //float xTrim = xcf[8];
                float yScale = xcf[9];
                float yOff = xcf[10];
                //float yTrim = xcf[11];
                float wScale = xcf[12];
                float wOff = xcf[13];
                //float wTrim = xcf[14];
                float hScale = xcf[15];
                float hOff = xcf[16];
                //float hTrim = xcf[17];

                //calc smth like a weighted avg of pat-
tern components
                float pattern =  (totalNorm * totalWeight
+
                                  directNorm * direct-
Weight +
                                  diffuseNorm * dif-
fuseWeight +
                                  gradeNorm  * grade-
Weight +
                                  elevationNorm * el-
evationWeight +
                                  noiseVal * noise-
Weight +
                                  1)
                                                  /(xcf[0] +xcf[1]
+xcf[2] +xcf[3] +xcf[4] +xcf[5] +1);

                //calc normalized scale effects and off-
set x, y, w, z
                //VALUE RANGE: 0.0 - 1.0
                float wv = (float)((((pattern-.5) * wS-
cale) + .5) + wOff);
                float hv = (float)((((pattern-.5) * hS-
cale) + .5) + hOff);
                float xv = (float)((((pattern-1) * xScale)
+ 0) + xOff);
                float yv = (float)((((pattern-1) * yScale)
+ 0) + yOff);

                //calculate real hole dimensions and
displacement in meters
                hw = wv * id;
                hh = hv * id;
                hx = xv * id + t;
                hy = yv * id + t;

                //snap and normalize coords to subgrid
                w = (int)(PApplet.round(hw/snap));
                h = (int)(PApplet.round(hh/snap));
                x = (int)(PApplet.round(hx/snap));
                y = (int)(PApplet.round(hy/snap));

                //convert back to absolute coords
                hw = w*snap;
                hh = h*snap;
                hx = px + x*snap;
                hy = py + y*snap;

                //load coordinate array (relative to
basegrid)
                //adjust order for vertice mapping
                coords[3] = new PVector(x,y);
                coords[2] = new PVector(x+hw, y);
                coords[1] = new PVector(x+hw, y+hh);
                coords[0] = new PVector(x, y+hh);


        }//end perforate

        //
        public void colorize(){

                //calc smth like a weighted avg of pat-
tern components
                float pattern =  (totalNorm * xcf[0] +
                                  directNorm * xcf[1]
+
                                  diffuseNorm * xcf[2]
+
                                  gradeNorm  * xcf[3]
+
                                  elevationNorm *
xcf[4] +
                                  noiseVal * xcf[5] +
                                  1)/
                                  (xcf[0] +xcf[1]
+xcf[2] +xcf[3] +xcf[4] +xcf[5] +1);

                int c1 = p.color(50, 50, 50);
                int c2 = p.color(200, 255, 200);
                float v = pattern * 255;

                float deltaR = p.red(c2)-p.red(c1);
                float deltaG = p.green(c2)-p.green(c1);
                float deltaB = p.blue(c2)-p.blue(c1);

                int c = p.color(
(p.red(c1)+(v)*(deltaR/255)),
(p.green(c1)+(v)*(deltaG/255)),
(p.blue(c1)+(v)*(deltaB/255))
                                  );
                pClr = c;
        }

}//end class
/************************************************
```

Skin Class

Skin is subclass of Hanger data model defining panel system, composed of subclass skin, which is composed of panels.
Panels are basically two sets of parameters: one which defines the panel's size and position in space, and another that defines the size and relative position of a single rectangular void in the panel. Panel also contains data from ecotect about the amount of direct and diffuse radiation each panel recieves.

Part of:

Aper[na]tures
Author: Joshua Parker
Date: 2011

Inheritted License: GPL, V3

```
************************************************/
package model;

import processing.core.PApplet;
import model.Panel;

public class Skin
{
        //The parent PApplet
        PApplet p;

        //registration pt
        public float x, y;

        //composit panels
        public Panel[] panels;

        //skin properties
        int u;                          //number of modules
in U direction
        int v;                          //number of modules
in V direction
        float dim;                      //panel lengh/width
dimension, D (meters)
        float thick;        //thickness, T of frame (me-
ters)

        //noise vars
        float inc;
        int seed;

        //CONSTRUCTOR
        Skin(PApplet parent,
                        int U,
                        int V,
                        float D,
                        float T,
                        float snap){

                p = parent;

                //noise vars
                inc = 0.4f;
                seed = 8;

                this.u = U;
                this.v = V;
                this.dim = D;
                this.thick = T;

                //initialize ld array for u x v panels
                panels = new Panel[u*v];

                //create u x v panels in ld array
                int n = 0;
                for(int i=0; i<v; i++) {
                        for(int j=0; j<u; j++) {

                                //calc panel dimen-
sion and reg pts
                                float pd = D;
                                float px = j*D;
                                float py = i*D;

                                //calc inner frame
dimension and reg pts
                                float id = pd-(T*2);
                                float ix = px+T;
                                float iy = py+T;

                                //frame thickness
                                float thick = T;

                                //create panel
                                panels[n] = new
Panel(
p,
px,
py,
pd,
ix,
iy,
id,
thick,
snap);
                                //incr counter
                                n++;
                        }
                }

                //generate initial pattern
                update();

        }//end Skin()

        // PUBLIC METHODS /////////////////////////////////
////////////////////

        //load data
        public void loadData(String PATH){

                //holders for rows and items in a row
respectively
                String loadedStrings[];
                String splitString[];

                //load data into an array of rows
                loadedStrings = p.loadStrings(PATH);

                int n = 0;
                for(int i=0; i<v; i++) {

                        //get data for first row, i+1
b/c first row of text file
                        //is column headers
                        splitString = PApplet.
split(loadedStrings[i+1], '\t');

                        for(int j=0; j<u; j++) {

                                //load up panel
                                panels[n].loadData(
splitString[2],
splitString[5],
splitString[8],
splitString[10],
splitString[12],
splitString[3],
splitString[6],
splitString[9],
splitString[11],
splitString[13]);
                                n++;
                        }
                }
        }//end loadData()

        //set a coefficient
        public void setXcf(int id, float C){
                int n = 0;
                for(int i=0; i<v; i++) {
                        for(int j=0; j<u; j++) {
                                panels[n].setXcf(id,
C);
                                n++;
                        }
                }
        }

        //set noise increment
        public void setInc(float Inc){
                inc = Inc;
        }

        //set noise seed
        public void setSeed(int Seed){
                seed = Seed;
        }

        //update pattern
        public void update(){
                //generate noise pattern
                generateNoise();
                //assign colors to panels
                colorize();
                //generate perforation from ecotect data
and noise
                perforate();
        }

        //generate noise
        public void generateNoise(){

                //noise constants
                float vx = 0.0f;
                float vy = 0.0f;
                //float inc = 0.4;
                //float k = 0;
                p.noiseSeed(seed);

                int n = 0;
                for(int i=0; i<v; i++) {
                        for(int j=0; j<u; j++) {

                                //calc noise
                                panels[n].noiseVal =
p.noise(vx, vy);

                                //increment panel
                                n++;

                                //increment noise x
param
                                vx = vx + inc;
                        }
                        //increment noise y param
                        vy = vy + inc;
                }
        }//end generateNoise()

        //find holes
        public void perforate(){
                int n = 0;
                for(int i=0; i<v; i++) {
                        for(int j=0; j<u; j++) {
                                panels[n].perfo-
rate();
                                n++;
                        }
                }
        }//end perforate

        //find holes
        public void colorize(){
                int n = 0;
                for(int i=0; i<v; i++) {
                        for(int j=0; j<u; j++) {
                                panels[n].color-
ize();
                                n++;
                        }
                }
        }//end perforate

}//end class
/************************************************
```

View Class

View - contains the scene(empty space and camera), imported 3d base model, and data model defining panel system. View calls draws to screen via viewState interface, though only one viewState is implemented: View3d, a basic 3d view.

Part of:

Aper[na]tures
Author: Joshua Parker
Date: 2011

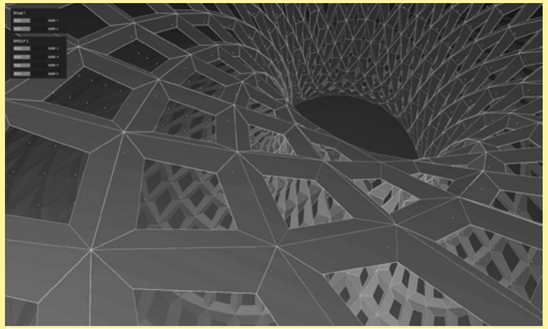
Inheritted License: GPL, V3

```
************************************************/
package view;

import processing.core.PApplet;
import remixlab.proscene.*;
import saito.objloader.*;
import model.Hanger;
import view.View3d;
import view.ViewState;

public class View{

        //parent PApplet
        PApplet p;

        //composit objs
        public Scene scene;
        public OBJModel model;
        public Hanger myhanger;

        //OBJModel flags - not used
        public boolean bTexture = true;
        public boolean bStroke = false;
        public boolean bMaterial = true;

        //states
        //private ViewState myView3d;
        //private ViewState myViewUnroll;

        public ViewState viewState;

        public View(PApplet parent, Hanger Myhanger){
                p = parent;

                //3d scene
                this.scene = new Scene(parent);
                this.scene.setAxisIsDrawn(false);
                this.scene.setGridIsDrawn(false);
                this.scene.enableFrustumEquationsUp-
date();
                //imported 3d model; no need to use mtls
or specify shapemode b/c
                //not drawing model, but just using the
face vertices and normals
                this.model = new OBJModel(p, "bs10.
obj");
                //this.model = new OBJModel(p, "bs10.
obj", "relative", LINES);
                //this.model.shapeMode(LINES);
                //this.model.disableMaterial();
                //this.model.enableDebug();
                this.model.scale(1);
                this.myhanger = Myhanger;

                //view state
                setViewState(new View3d(p, this));

        }

        // PUBLIC METHODS /////////////////////////////////
```

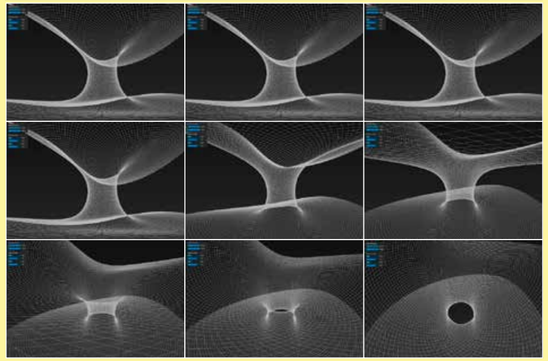
```
////////////////////

	//set view state
	public void setViewState(ViewState newState){
		this.viewState = newState;
	}

	//get current fill
	public boolean getFill(){
		return this.viewState.getFill();
	}

	//turn off strokes, turn on surface fill
	public void enableFill(){
		this.viewState.enableFill();
	}

	//turn off surface fill, turn on strokes
	public void disableFill(){
		this.viewState.disableFill();
	}

	//draw the view
	public void draw() {
		this.viewState.draw();
	}
}
/************************************************

View3d Class

View3d implements a basic 3d view

Part of:

Aper[na]tures
Author: Joshua Parker
Date: 2011

Inheritted License: GPL, V3

************************************************/
package view;

import processing.core.*;
import saito.objloader.*;
import model.*;

public class View3d implements ViewState{

	//The parent PApplet
	PApplet p;

	//composit view
	public View myview;

	//flags
	boolean bFill = false;

	//CONSTRUCTOR
	View3d(PApplet parent, View Myview){
		p = parent;
		this.myview = Myview;
	}

	// PUBLIC METHODS /////////////////////////////////
////////////////////

	//turn off strokes, turn on surface fill
	public void enableFill(){
		bFill = true;
	}

	//turn off surface fill, turn on strokes
	public void disableFill(){
		bFill = false;
	}

	//get current fill
	public boolean getFill(){
		return bFill;
	}

	//draw the 3d view
	public void draw() {

		//setup lighting
		p.directionalLight(126, 126, 126, 0, 0,
-1);
		//p.ambientLight(102, 102, 102);
		p.lights();
		p.stroke(10,10,10);

		//set camera position and target
/*		PVector v = new PVector(0,0,-1);
		PVector v2 = new PVector(0,0,0);
		myview.scene.camera().
setUpVector(v,true);
		myview.scene.camera().lookAt(v2);*/

		//just want vertices so do not draw the
3d model!!! this will slow sketch way down
		//myview.model.draw();

		//load faces in array
		Segment segment = myview.model.getSeg-
ment(0);

		Face[] faces = segment.getFaces();

		//for each face...
		for (int i = faces.length-1; i >= 0;
i--) {

			//get face and corresponding
panel
			Face nextFace = faces[i];
			Panel nextPanel = myview.my-
hanger.myskin.panels[i];

			//get vertices and normals of
face and relative coordinates
			//of panel opening
			PVector[] vs = nextFace.get-
Vertices();
			PVector[] ns = nextFace.get-
Normals();
			PVector[] cs = nextPanel.co-
ords;

			//get panel color
			int clr = nextPanel.pClr;

			//map panel space coordinates
to world space coordinates.
			PVector[] pts = mapFace(vs,
ns, cs);

			//stroke normal
			//line(vs[0].x, vs[0].y,
vs[0].z, vs[0].x+ns[0].x, vs[0].y+ns[0].y, vs[0].z+ns[0].z);

			//set fill and stroke
			if(bFill){
				p.fill(clr);
				p.noStroke();
			}else{
				p.noFill();
				p.stroke(10, 10,
10);
			}

			//draw four quad polygons
			p.beginShape();
			p.vertex(vs[0].x, vs[0].y,
vs[0].z);
			p.vertex(vs[1].x, vs[1].y,
vs[1].z);
			p.vertex(pts[1].x, pts[1].y,
pts[1].z);
			p.vertex(pts[0].x, pts[0].y,
pts[0].z);
			p.endShape();

			p.beginShape();
			p.vertex(vs[1].x, vs[1].y,
vs[1].z);
			p.vertex(vs[2].x, vs[2].y,
vs[2].z);
			p.vertex(pts[2].x, pts[2].y,
pts[2].z);
			p.vertex(pts[1].x, pts[1].y,
pts[1].z);
			p.endShape();

			p.beginShape();
			p.vertex(vs[2].x, vs[2].y,
vs[2].z);
			p.vertex(vs[3].x, vs[3].y,
vs[3].z);
			p.vertex(pts[3].x, pts[3].y,
pts[3].z);
			p.vertex(pts[2].x, pts[2].y,
pts[2].z);
			p.endShape();

			p.beginShape();
			p.vertex(vs[3].x, vs[3].y,
vs[3].z);
			p.vertex(vs[0].x, vs[0].y,
vs[0].z);
			p.vertex(pts[0].x, pts[0].y,
pts[0].z);
			p.vertex(pts[3].x, pts[3].y,
pts[3].z);
			p.endShape();

		}//end for loop

	}//end draw

	// PRIVATE METHODS /////////////////////////////////
////////////////////

	//find hole vertices on face from a face
vertices(vs),
	//face normals(ns), and face-space coordinates(cs)
	private PVector[] mapFace(PVector[] vs, PVector[]
n, PVector[] cs){

		PVector[] pts = new PVector[0];

		//for each corner, map 2d coord to 3d pt
		for (int i = vs.length-1; i >= 0; i--) {
			PVector pt = mapPt(vs[0],
n[i], cs[i]);
			pts = (PVector[])PApplet.
append(pts, pt);
		}
		return pts;

	}

	//find singe 3d pt from a vertice(v), normal(n),
	//and 2d relative coordinate(c)
	private PVector mapPt(PVector v, PVector n, PVec-
tor c){

		//get angle of normal
		float a = (float)Math.atan(n.z/n.y);
		a = PApplet.radians(90-PApplet.
abs(PApplet.degrees(a)));

		float x,y,z;

		if(n.y < 0){
			x = v.x + c.x;
			y = v.y - c.y*PApplet.cos(a);
			z = v.z - c.y*PApplet.sin(a);
		}else{
			x = v.x + c.x;
			y = v.y - c.y*PApplet.cos(a);
			z = v.z + c.y*PApplet.sin(a);
		}

		PVector pt3d = new PVector(x, y, z);

		return pt3d;
	}

}
/************************************************

ViewState Class

ViewState is an interface for various views

Part of:

Aper[na]tures
Author: Joshua Parker
Date: 2011

Inheritted License: GPL, V3

************************************************/
package view;

public interface ViewState{

	public void enableFill();
	public void disableFill();
	public boolean getFill();
	public void draw();
}
```

ZHUHAI PAVILION, SOURCE CODE

A CUSTOM PANELING TOOL WAS BUILT WITH JAVA/PROCESSING USING OPEN SOURCE GEOMETRY LIBRARY (IGEO, BY SATURO SUGIHARA) TO INTEROGATE FORM AND EXPLORE THE POSSIBILITY OF PERFORATING THE SHELL IN PLACES TO CREATE VISUAL AND PHYSICAL ACCESS THROUGH VERTICAL SURFACES. TOOL DEFINES A UV DIAGRAD AND MAPS PERFORATION PATTERNS AN INPUT SURFACE USING SIMPLE ALGORITHMS AND EXPOSES DESIGN PARAMETERS. THIS ALLOWED FOR REAL-TIME COLLABORATIVE USE BY ARCHITECTS, ENGINEERS, AND CLIENT.



gradient paneling condition
fedback to surface.



continuous surface subdivi-
sion.


```
/*************************************************

DiagridPaneler

Author: Joshua Parker
Date: 2011

Uses the following processing libraries:

controlP5 - gui toolkit
- http://www.sojamo.de/libraries/controlP5/
Proscene - 3d scene library
- http://code.google.com/p/proscene/
Objloader - 3d object loader
- http://code.google.com/p/saitoobjloader/

Inheritted License: GPL, V3

*************************************************/
import igeo.*;
import processing.core.*;
import controlP5.*;
import processing.opengl.*;
import controller.Listener;
import controller.Controller;
import migeo.MyIG;
import migeo.MyPIGraphicsGL;
import model.Box;
import model.Model;
import view.*;
import gui.*;

public class DiagridPaneler extends PApplet{

	private static final long serialVersionUID = 1L;

	//path to tabular data from ecotect analyses
	String PATH = "data/ecotectData.txt";

	//objects
	Listener myListener;
	Controller myController;
	Model mymodel;
	View myview;
	Gui mygui;

	IG myig;

	public void setup(){
	  size(1920, 1200, "migeo.MyPIGraphicsGL");
	  //IG.bg(50, 40, 70, 90);
	  IG.bg(255, 255, 255, 255);

	  //MODEL: Parametric Data for panel system
	  //mybox = new Box(this);
	  mymodel = new Model(this);

	  //VIEW: draws a representation of model to screen
	  //NOTE: This is a representation of the data model ONLY!
	  //      It create shapes and geometry and igeo renders to
panel panes
	  myview = new View(this, mymodel);

	  //GUI: needs to access view scene for drawing 2d gui on top
of 3d scene
	  //TBD: This should wrap all iu elements, ie. igeo panel,
sidebar, etc.
	  mygui = new Gui(this /*, IG.current().panel*/);

	  //CONTROLLER: updates model & views in response to ui events
routed to it
	  myController = new Controller(this, mymodel, mygui, myview);

	  //LISTENER/ROUTER: recieves ui events, ids and routes them to
controller
	  myListener = new Listener(this, myController);
	}

	public void draw(){
	  //update model, view and gui via controller
	  myController.update();
	}

	//required to run as application
	public static void main(String args[]) {
	PApplet.main(new String[] { "--present", "DiagridPaneler" });
	}

	}

	//send ui events to listener, so it can route them to control-
ler
	public void controlEvent(ControlEvent theEvent){myListener.
controlEvent(theEvent);}
	public void mousePressed(){myListener.mousePressed(mouseX,
mouseY);}
	public void keyPressed(){myListener.keyPressed(key);}

}



/*************************************************

Controller Class

MVC Controller contains methods for updating data
model & controlling view(s)*

Part of:

Aper[na]tures
Author: Joshua Parker
Date: 2011

Inheritted License: GPL, V3

*************************************************/
package controller;

import igeo.IG;
import processing.core.PApplet;
import processing.core.PVector;
//import model.Box;
import model.Model;
import view.View;
import gui.Gui;

public class Controller{

	//The parent PApplet
	PApplet p;

	//PROPS
	private Model mymodel;
	private View myview;
	private Gui mygui;

	//CONSTRUCTOR
	public Controller(PApplet parent, Model Mymodel, Gui Mygui,
View Myview){
		p = parent;
	    mymodel = Mymodel;
	    mygui = Mygui;
	    myview = Myview;
	}

	// PUBLIC METHODS //////////////////////////////////////////
///////

	//
	public void setUCount(int n){
		//
		mymodel.unum = n;
	}

	//
	public void setVCount(int n){
		//
		mymodel.vnum = n;
    }

	//
	public void setPatternUStart(int v){
		//
		mymodel.uMin = (int) (v*.01*mymodel.unum);
	}

	//
	public void setPatternUStop(int v){
		//
		mymodel.uMax = (int) (v*.01*mymodel.unum);
```

```
	}

	//
	public void setPatternVStart(int v){
		//
		mymodel.vMin = (int) (v*.01*mymodel.vnum);
	}

	//
	public void setPatternVStop(int v){
		//
		mymodel.vMax = (int) (v*.01*mymodel.vnum);
	}

	// PUBLIC METHODS /////////////////////////////////////////////
///////

	//
	public void focusGui(){
		//myview.scene.disableMouseHandling();
		//myview.scene.disableKeyboardHandling();
	}

	//
	public void focusScene(){

		//myview.scene.enableMouseHandling();
		//myview.scene.enableKeyboardHandling();
	}

	//
	public void toggleTexture(){
	 /*if(!myview.bTexture){
			showTexture();
	 } else {
			hideTexture();
	 }*/
	}

	//
	public void toggleMaterial(){
//		if(!myview.bMaterial) {
//			showMaterial();
//		} else {
//			hideMaterial();
//		}
	}

	//
	public void toggleFill(){
		/*if(!myview.getFill()) {
			showFill();
		} else {
			hideFill();
		}*/
	}

	//
	public void showTexture(){
		//myview.model.enableTexture();
		//myview.bTexture = true;
	}

	//
	public void hideTexture(){
		//myview.model.disableTexture();
		//myview.bTexture = false;
	}

	//
	public void showMaterial(){
		//myview.model.enableMaterial();
		//myview.bMaterial = true;
	}

	//
	public void hideMaterial(){
		//myview.model.disableMaterial();
		//myview.bMaterial = false;
	}

	//
	public void showFill(){
		//myview.enableFill();
	}

	//
	public void hideFill(){
		//myview.disableFill();
	}
	//
	public void topView(){
		//IG.topView();
		IG.current().panel.currentPane().getView().setTop();
		//IG.view(0).setTop();
	}
	//
	public void perspectiveView(){
		//IG.perspectiveView();
		IG.current().panel.currentPane().getView().setAxonometric();
	}
	//
	public void gotoView1(){
		myview.setView1();
	}
	//
	public void gotoView2(){
		myview.setView2();
	}

	//
	public void update(){

		//mybox.update(); //update the model
		mymodel.update();
		myview.draw(); // draw the 3d scene
		mygui.draw(); // then draw the gui on top
	}
}
/*************************************************

Listener Class

Listener recieves ui events, parses them and routes
them to controller for handling.

Part of:

Aper[na]tures
Author: Joshua Parker
Date: 2011

Inheritted License: GPL, V3

*************************************************/
package controller;

import igeo.IG;
import controlP5.ControlEvent;
import controlP5.ControlListener;
import processing.core.PApplet;

public class Listener implements ControlListener{

	//The parent PApplet
	PApplet p;

	//PROPS
	Controller myController;

	//CONSTRUCTOR
	public Listener(PApplet parent, Controller MyController){
		p = parent;
		myController = MyController;
	}

	// PUBLIC METHODS /////////////////////////////////////////////
///////

	//route slider events to controller via private handlers
	public void controlEvent(ControlEvent theEvent) {

		switch(theEvent.getController().getParent().getId()){
		case(0):
			//route g1 (shortcut info) event!
			g1ControlEvent(theEvent);
			break;
		case(1):
			//route g2 (noise sliders) event!
			g2ControlEvent(theEvent);
			break;
```



```
		}
	}

	//route mouse events to controller
	//this should be replaced with focus on hover
	public void mousePressed(int mx, int my) {
		if (mx < 200) {
			myController.focusGui();
		} else {
			myController.focusScene();
		}
	}

	//route key press events to controller
	public void keyPressed(char key) {

		if(key == 't') {
			//toggle the texture listed in .mtl file
			myController.toggleTexture();
		}
		else if(key == 'w') {
			//toggle the material listed in .mtl file
			//myController.toggleMaterial();
			//currentMousePane.getView().mode().setDrawMode(true,false,false);
			IG.current().panel.currentPane().getView().mode().setDrawMode(true,false,false);
		}
		else if(key == '0') {
			//toggle the fill for new shapes
			System.exit(0);
		}
		else if(key == '1') {
			//toggle the fill for new shapes
			myController.gotoView1();
		}
		else if(key == '2') {
			//toggle the fill for new shapes
			myController.gotoView2();
		}
		else if(key == 'a') {
			//toggle the fill for new shapes
			myController.topView();
		}
		else if(key == 's') {
			//toggle the fill for new shapes
			myController.perspectiveView();
		}
	}

	// PRIVATE METHODS /////////////////////////////////////////////
///////

	//parse group 2 (noise sliders) event!
*******************************
	private void g1ControlEvent(ControlEvent theEvent){

		float v;
		switch(theEvent.getController().getId()){
		case(1):
			v = theEvent.getController().getValue();
			myController.setUCount((int)(v));
		break;
		case(2):
			v = theEvent.getController().getValue();
			myController.setVCount((int)(v));
		break;
		}
	}

	//parse group 2 (noise sliders) event!
*******************************
		private void g2ControlEvent(ControlEvent theEvent){

			float v;
			switch(theEvent.getController().getId()){
			case(4):
				v = theEvent.getController().getValue();
				myController.setPatternUStart((int)(v));
			break;
			case(5):
				v = theEvent.getController().getValue();
				myController.setPatternUStop((int)(v));
			break;
			case(6):
				v = theEvent.getController().getValue();
				myController.setPatternVStart((int)(v));
			break;
			case(7):
				v = theEvent.getController().getValue();
				myController.setPatternVStop((int)(v));
			break;
			}
		}

}

package gui;

import igeo.IG;
import igeo.IVec;
import igeo.gui.IGraphicMode;
import igeo.gui.IGraphicMode.GraphicType;
import processing.core.PApplet;
import processing.core.PFont;
import controlP5.ControlP5;
import controlP5.Knob;

public class ControlPanel {

	//parent PApplet
	PApplet p;

	private ControlP5 cp5;

	//
	controlP5.Group[] groups;

	//count elements
	//int NumOfTabs;
	int numOfAccordings;
	int numOfGroups;
	int numOfElements;

	//current Elements
	//String curTab;
	//String curAccording;
	//String curGroup;
	//String curElement;

	int x;
	int y;
	int w;
	int h;

	int[] ppt = {0,0};
	int[] gpt = {0,0};
	int[] ept = {0,0};

	int dx;
	int dy;

	//style
	//int CPMargin;
	int marginTop = 40;
	int marginLeft = 20;
	int marginBottom = 10;
	int groupWidth = 200;
	//int groupHeight;
	//int accordingMargin;
	//int groupPadding;
	int headerPaddingTop = 10;
	int headerPaddingLeft = 5;
	int elementPaddingTop = 5;
	int elementPaddingLeft = 10;
	int elementWidth = 120;
	int elementHeight = 15;


	int BarH;
	int BgClr; //bg of block
	int BarClr; //bar color
	int HoverClr; //bar color on hover
	int LabelClr;

	int slideH;
	int slideW;
	int slideX;
	int slideSpc;
	int slideDY;
```

```
	PFont arial;
	PFont arial10;

	//cp5 objs
	//private Cp5UI cPanel;

	public ControlPanel(int X, int Y, int W, int H, PApplet parent)
{

		p = parent;

		this.x = X;
		this.y = Y;
		this.w = W;
		this.h = H;

		gpt[0] = x;
		gpt[1] = y;

		this.dx = 25;
		this.dy = 30;

		cp5 = new ControlP5(p);
		cp5.setAutoDraw(false);

		numOfAccordings = 0;
		numOfGroups = 0;
		numOfElements = 0;

		groups = new controlP5.Group[2];

		BarH = 20;
		BgClr = p.color(50); //bg of block
		BarClr = p.color(50); //bar color
		HoverClr = 30; //bar color on hover
		LabelClr = p.color(255);

		slideH = 10;
		slideW = 100;
		slideX = 10;
		slideSpc = 5;
		slideDY = slideH+slideSpc;

		arial = p.createFont("ArialMT-30", 11);
		arial10 = p.createFont("ArialMT-30", 10);

	}

	// PUBLIC METHODS /////////////////////////////////////////////
///////

	public void draw() {
		//switch to 2d draw gui, then switch back
		//also need to disable active pane so it
		//doesn't draw in 1of4 grid pane in gridview
		disable3d();
		cp5.draw();
		enable3d();
	}

	// PRIVATE METHODS /////////////////////////////////////////////
////////

	private void disable3d(){
		IGraphicMode mode2D = new IGraphicMode();
		mode2D.setGraphicType(GraphicType.J2D);
		//myview.panel.currentPane().getView().setMode(mode2D);
		IG.graphicMode(mode2D);
	}

	private void enable3d(){
		IGraphicMode modeGL = new IGraphicMode();
		modeGL.setGraphicType(GraphicType.GL);
		//myview.panel.currentPane().getView().setMode(modeGL);
		IG.graphicMode(modeGL);
	}

	//public void addTab(){}
	public void addAccording(){

	}

	/*//add group
	public void addGroup(String name){

		//calc bgheight, placeholder for now <----!!!
		int bgHeight = 50;

		//
		groups[numOfGroups] = cp5.addGroup(name)
			.setId(numOfGroups)
			.setLabel(name)
			.setPosition(gpt[0] + marginLeft, gpt[1] + mar-
ginTop)
			.setWidth(groupWidth)
			//.setBackgroundHeight(bgHeight)
			.setBarHeight(BarH)
			.setBackgroundColor(BgClr)
			.setColorBackground(BarClr)
			.setColorForeground(HoverClr)
			.setColorLabel(LabelClr);

		numOfGroups++;

		gpt[1] = gpt[1] + elementHeight + marginTop;
		ept[0] = 0 + elementPaddingLeft;
		ept[1] = 0 + elementPaddingTop;

		//return this;
	}*/

	//add group
	@SuppressWarnings("deprecation")
	public void addGroup(String name){

		//calc bgheight, placeholder for now <----!!!
		//int bgHeight = 50;

		push();

		//
		groups[numOfGroups] = cp5.addGroup(name)
			.setId(numOfGroups)
			.setLabel(name)
			.setPosition(gpt[0], gpt[1])
			.setWidth(groupWidth)
			//.setBackgroundHeight(bgHeight)
			.setBarHeight(BarH)
			.setBackgroundColor(BgClr)
			.setColorBackground(BarClr)
			.setColorForeground(HoverClr)
			.setColorLabel(LabelClr)
			.hideBar();

		numOfGroups++;

		//
		cp5.addTextlabel(name+"-label")
				.setId(numOfElements)
		 .setText(name)
		 .setHeight(elementHeight)
		 .setPosition(ept[0]+headerPaddingLeft,
ept[1]+headerPaddingTop)
		 //.setColorValue(0xffffff00)
		 .setFont(p.createFont("Arial-bold",12))
		 .moveTo(groups[numOfGroups-1])
		 ;

		numOfElements++;

		step();

		//return this;
	}

	public void endGroup(){

		//
		groups[numOfGroups-1].setBackgroundHeight(ept[1]);

		pop();
	}

	//add sep function for floating pt values
	public void addSlider(String label, int minV, int maxV, int de-
faultV){

		//
		cp5.addSlider(label)
			.setId(numOfElements)
			.moveTo(groups[numOfGroups-1])
```



```
			.setPosition(ept[0]+elementPaddingLeft,
ept[1]+elementPaddingTop)
			.setSize(elementWidth,elementHeight)
			.setRange(minV,maxV)
			.setValue(defaultV);

		numOfElements++;

		step();

	}

	/*//add sep function for floating pt values
		public void addSlider(String label, int minV, int maxV,
int defaultV){

			//
			cp5.addSlider(label)
				.setId(numOfElements)
				.moveTo(groups[numOfGroups-1])
				.setPosition(ept[0],ept[1])
				.setSize(elementWidth,elementHeight)
				.setRange(minV,maxV)
				.setValue(defaultV);

			numOfElements++;

			gpt[1] = gpt[1] + elementHeight;
			ept[1] = ept[1] + elementHeight + elementPad-
dingTop;


			groups[numOfGroups-1].setBackgroundHeight(ept[1]
+ elementPaddingTop);

		}*/

	public void addKnob(String label, int minV, int maxV, int de-
faultV){

		//add to style sheet
		//...

		//
		cp5.addKnob(label)
			.setId(numOfElements)
			.moveTo(groups[numOfGroups-1])
			.setPosition(20,20)
			.setRadius(35)
			.setAngleRange(2*PApplet.PI)
			.setStartAngle((float)(.5*PApplet.PI))
			.setRange(minV,maxV)
			.setValue(defaultV)
			.setNumberOfTickMarks(16)
			.snapToTickMarks(true)
			.setColorForeground(p.color(255))
			.setColorBackground(p.color(40))
			.setColorActive(p.color(255,255,0))
			.setDragDirection(Knob.HORIZONTAL);

		numOfElements++;
	}
	public void addTextArea(String label, String text){

		//add to style sheet
		//...

		//
		cp5.addTextarea(label)
			.moveTo(groups[numOfGroups-1])
			.setPosition(5,5)
			.setSize(195,90)
			.setFont(arial)
			.setLineHeight(14)
			.setColor(p.color(128))
			.setColorBackground(p.color(50))
			.setColorForeground(p.color(255,100))
			.setText(text);

		numOfElements++;
	}

	private void push(){
		//push into group
		//gpt[0] += dx;
		//gpt[1] += dy;

		//reset element registration pt
		ept[0] = 0;
		ept[1] = 0;
	}
	private void step(){
		//set to next element in group
		ept[1] += dy;
	}
	private void pop(){
		//pop out of group
		//gpt[0] -= dx;
		gpt[1] += ept[1] + marginBottom;
	}

}
/*************************************************

Gui Class

Gui is a wrapper for controlP5 library, which
provides user interface toolkit. more at:
http://www.sojamo.de/libraries/controlP5/

Part of:

Aper[na]tures
Author: Joshua Parker
Date: 2011

Inheritted License: GPL, V3

*************************************************/
package gui;

import igeo.IG;
import igeo.gui.IGraphicMode;
import igeo.gui.IGraphicMode.GraphicType;
import processing.core.PApplet;
import processing.core.PFont;
import controlP5.*;
//import remixlab.proscene.*;
//import view.*;

public class Gui
{

	//parent PApplet
	PApplet p;

	//cp5 objs
	private ControlPanel cPanel;

	//CONSTRUCTOR
	public Gui(PApplet parent){

		//initialize
		p = parent;

		cPanel = new ControlPanel(20,20, 200, p.height, p);

		cPanel.addGroup("Resample surface");
		cPanel.addSlider("u Samples", 5, 200, 20);
		cPanel.addSlider("v Samples", 5, 200, 20);
		cPanel.endGroup();
		cPanel.addGroup("Pattern u/v range");
		cPanel.addSlider("u start", 0, 100, 25);
		cPanel.addSlider("u stop", 0, 100, 75);
		cPanel.addSlider("v start", 0, 100, 25);
		cPanel.addSlider("v stop", 0, 100, 75);
		cPanel.endGroup();

	}

	// PUBLIC METHODS /////////////////////////////////////////////
///////

	public void draw() {
		//switch to 2d draw gui, then switch back
		//also need to disable active pane so it
		//doesn't draw in 1of4 grid pane in gridview
//		disable3d();
//		cp5.draw();
//		enable3d();

		cPanel.draw();
	}
```

```
	/*// PRIVATE METHODS /////////////////////////////////////////
//////////

	private void disable3d(){
		IGraphicMode mode2D = new IGraphicMode();
		mode2D.setGraphicType(GraphicType.J2D);
		//myview.panel.currentPane().getView().setMode(mode2D);
		IG.graphicMode(mode2D);
	}

	private void enable3d(){
		IGraphicMode modeGL = new IGraphicMode();
		modeGL.setGraphicType(GraphicType.GL);
		//myview.panel.currentPane().getView().setMode(modeGL);
		IG.graphicMode(modeGL);
	}*/

}
package migeo;

import igeo.IG;

public class MyIG extends IG{

	/** Processing Graphics using OpenGL to be put in size() method
in Processing */
    public static final String GL = "igeo.p.MyPIGraphicsGL";

}
package migeo;

import java.awt.Color;

import javax.media.opengl.GL;

import processing.core.PApplet;

import com.sun.opengl.util.j2d.Overlay;

import igeo.IG;
import igeo.gui.IGraphicsGL;
import igeo.gui.IGridPanel;
import igeo.gui.IPaneLight;
import igeo.gui.IPanel;
import igeo.gui.IView;
import igeo.p.PIConfig;
import igeo.p.PIGraphicsGL;
import igeo.p.PIInput;

public class MyPIGraphicsGL extends PIGraphicsGL{



    //public PIGraphicsGL(){ super(); }
        public IPaneLight pane;

    /**
    setParent is called by Processing in the initialization process of
Processing.
    Here the initialization proces of iGeo is also done.
    @param parent parent PApplet of Processing.
         */
        public void setParent(PApplet parent){

            super.setParent(parent);

            // initialize root GUI
            //panel = new IGridPanel(0,0,parent.getWidth(),parent.
getHeight(),2,2);
            panel = new IPanel(0,0,parent.getWidth(),parent.
getHeight());


            IView v=null;
            v = IView.getDefaultPerspectiveView(0,0,parent.
getWidth(),parent.getHeight());
            v.enableGL(); // here?
          v.enableRotationAroundTarget(); // here?
          v.setTarget(0,0,0); //

          pane = new IPaneLight(0,0,parent.getWidth(),parent.
getHeight(),v,panel);
            panel.addPane(pane);
            panel.currentMousePane = pane;
            panel.setVisible(true);

            // initialize iGeo
            IG ig = IG.init(panel);

            ig.server().graphicServer().enableGL(); //
            //ig.setBasePath(parent.sketchPath("")); // not sketch-
Path

            if(!parent.online){ // only when running local
                ig.setBasePath(parent.dataPath("")); // for default
path to read/write files
            }

            ig.setInputWrapper(new PIInput(parent));

            parent.addMouseListener(panel);
            parent.addMouseMotionListener(panel);
            parent.addMouseWheelListener(panel);
            parent.addKeyListener(panel);
            parent.addFocusListener(panel);
            parent.addComponentListener(panel);

            //igg = new IGraphics();
            igg = new IGraphicsGL();

            //noSmooth();

            if(PIConfig.drawBeforeProcessing) parent.
registerPre(this);
            else parent.registerDraw(this);
            parent.registerPost(this);

            if(PIConfig.resizable){ parent.frame.setResizable(true);
}

            super.hints[DISABLE_OPENGL_2X_SMOOTH]=true;  //
            super.hints[ENABLE_OPENGL_4X_SMOOTH]=true;  //
        }

        public void setGLProperties(){
            gl.glEnable(GL.GL_MULTISAMPLE); //
            gl.glEnable(GL.GL_POINT_SMOOTH); //
            gl.glEnable(GL.GL_LINE_SMOOTH); //
            gl.glEnable(GL.GL_POLYGON_SMOOTH); //

            gl.glEnable(GL.GL_ALPHA_TEST); //
            //gl.glEnable(GL.GL_BLEND); //
            //gl.glDisable(GL.GL_BLEND); //
            //gl.glBlendFunc(GL.GL_SRC_ALPHA, GL.GL_ONE_MINUS_SRC_
ALPHA); //
            //gl.glBlendFunc(GL.GL_SRC_ALPHA, GL.GL_ONE); //

            gl.glHint(GL.GL_LINE_SMOOTH_HINT, GL.GL_NICEST); //
            gl.glHint(GL.GL_POINT_SMOOTH_HINT, GL.GL_NICEST); //
            gl.glHint(GL.GL_POLYGON_SMOOTH_HINT, GL.GL_NICEST); //

            //gl.glEnable(GL.GL_NORMALIZE); //
            //gl.glEnable(GL.GL_AUTO_NORMAL); //
            //gl.glShadeModel(GL.GL_SMOOTH); //

            //gl.glLightModeli(GL.GL_LIGHT_MODEL_TWO_SIDE, 1); //

            //gl.glEnable(GL.GL_DEPTH_TEST); // already enabled in
super
            //gl.glDisable(GL.GL_DEPTH_TEST); // ? for transparency

            //gl.glEnable(GL.GL_LIGHTING); // test!
            //gl.glEnable(GL.GL_LIGHT1); // test!

        }


        public void pre(){ drawIG(); }
        public void draw(){ drawIG(); }

        /**
           Drawing all the iGeo objects through IPanel.
           Overlay is also used to draw 2D graphics on top of OpenGL
3D graphics.
        */
        public synchronized void drawIG(){

            int[] viewport=null;
            if(PIConfig.restoreGLViewport){
                viewport = new int[4];
                gl.glGetIntegerv(GL.GL_VIEWPORT, viewport, 0);
            }

            gl.glMatrixMode(GL.GL_MODELVIEW);
          gl.glPushMatrix();

          gl.glMatrixMode(GL.GL_PROJECTION);
          gl.glPushMatrix();

            if(PIConfig.resetGLDepthBefore) gl.glClear(GL.GL_DEPTH_
BUFFER_BIT);

            //gl.glClear(GL.GL_COLOR_BUFFER_BIT | GL.GL_DEPTH_BUF-
FER_BIT);

            //gl.glClear(GL.GL_COLOR_BUFFER_BIT);

            setGLProperties();

            if(igg.getGraphics()==null){
                setOverlay();
            }

            //igg.setGraphics(overlay.createGraphics());

            igg.getGraphics().clearRect(0,0,parent.
getWidth(),parent.getHeight()); //

            //overlay = new Overlay(drawable); //
            //Graphics2D g = overlay.createGraphics();
            //igg.setGraphics(g);

            igg.setGL(gl);

            panel.draw(igg);

            if(PIConfig.resetGLDepthAfter) gl.glClear(GL.GL_DEPTH_
BUFFER_BIT);

            gl.glMatrixMode(GL.GL_PROJECTION);
          gl.glPopMatrix();

            gl.glMatrixMode(GL.GL_MODELVIEW);
          gl.glPopMatrix();


            // bring the original viewport back
            if(PIConfig.restoreGLViewport && viewport!=null){
                gl.glViewport(viewport[0], viewport[1], view-
port[2], viewport[3]);
            }

            if(overlay!=null){
                overlay.markDirty(0,0,parent.getWidth(),parent.
getHeight());
                overlay.drawAll();
            }

            //g.dispose();
            //igg.getGraphics().dispose();
        }

        public void setOverlay(){
            overlay = new Overlay(drawable); //
            igg.setGraphics(overlay.createGraphics());
            igg.getGraphics().setBackground(overlayBG);
        }

        public void setSize(int w, int h){
            super.setSize(w,h);
            setOverlay(); // update overlay
        }


        public void post(){
            if(overwritePAppletFinish) parent.finished=finished;
            if(overwritePAppletLoop) if(looping) parent.loop();
else parent.noLoop();
        }

        public void loop(){ if(!looping) looping=true; }
        public void noLoop(){ if(looping) looping=false; }

        public void start(){ if(finished) finished=false; }
        public void stop(){ if(!finished) finished=true; }

        /*
        public void mousePressed(MouseEvent e){
        }
        public void mouseReleased(MouseEvent e){
        }
        public void mouseClicked(MouseEvent e){
        }
        public void mouseEntered(MouseEvent e){
        }
        public void mouseExited(MouseEvent e){
        }
        public void mouseMoved(MouseEvent e){
        }
        public void mouseDragged(MouseEvent e){
        }
        public void mouseWheelMoved(MouseWheelEvent e){
        }
        public void keyPressed(KeyEvent e){
        }
        public void keyReleased(KeyEvent e){
        }
        public void keyTyped(KeyEvent e){
        }
        public void focusLost(FocusEvent e){
        }
        public void focusGained(FocusEvent e){
        }
        */
        /*
        public void componentHidden(ComponentEvent e){
        }
        public void componentMoved(ComponentEvent e){
        }
        public synchronized void componentResized(ComponentEvent e){
            int w = e.getComponent().getBounds().width;
            int h = e.getComponent().getBounds().height;
            setSize(w,h);
        }
        public void componentShown(ComponentEvent e){
        }
        */
    }

/************************************************

FramePanel Class

Part of:

Aper[na]tures
Author: Joshua Parker
Date: 2011

Inheritted License: GPL, V3

************************************************/
package model;

import igeo.IVec;
import processing.core.PApplet;

public class FramePanel {

    //The parent PApplet
    PApplet p;

    double MAXTHICK = .2;

    public IVec center;
    public IVec normal;
    public IVec[] corners = new IVec[4];
    public double openness; //0.0 - 1.0
    public double thinness; //0.0 - 1.0

    public IVec[] pts = new IVec[5];
    public IVec[] normals = new IVec[5];

    public FramePanel(IVec[] Pts, IVec[] Normals, double Openness,
PApplet parent){

        p = parent;

        //pts = Pts;
        //normals = Normals;

        center = Pts[0].dup();
```

```
                normal = Normals[0].dup();

                corners[0] = Pts[1].dup();
                corners[1] = Pts[2].dup();
                corners[2] = Pts[3].dup();
                corners[3] = Pts[4].dup();

                openness = Openness;
                thinness = .5;

        }

}

/*************************************************

Model Class

Part of:

Aper[na]tures
Author: Joshua Parker
Date: 2011

Inheritted License: GPL, V3

*************************************************/
package model;

import java.util.ArrayList;

import processing.core.PApplet;
import igeo.IBox;
import igeo.ICurve;
import igeo.IFieldVisualizer;
import igeo.IG;
import igeo.IImageMap;
import igeo.IPoint;
import igeo.IRandom;
import igeo.ISurface;
import igeo.ISurfaceNormalField;
import igeo.IVec;

//import igeo.IVec;
//import processing.core.PApplet;

public class Model {

        //The parent PApplet
        PApplet p;

        public ISurface surf;
        public IImageMap map;
        public int unum, vnum;
        public int uMax, uMin, vMax, vMin;
        public double openMax, openMin;
        public ArrayList<FramePanel> panels;

        public Model(PApplet parent){
                p = parent;

                //load assets and create data model here. SHOULD NOT
NOT BE GEOMETRY HERE!!! SHOULD CREATE LOADER CLASS FOR THAT.
                //IG.open("testsrf-zhuhaiPavilion-untrimTop.3dm");
                IG.open("testsrf-zhuhaiPavilion-wormhole1.2.3dm");
                ISurface[] surfs = IG.surfaces();
                map = new IImageMap("map1.jpg");

                surf = surfs[0];
                unum = 20;
                vnum = 20;
                uMax = 15;
                uMin = 5;
                vMax = 15;
                vMin = 5;
                openMax = .8;
                openMin = .2;

        }

        //update Model
        public void update(){
                //
                double uinc = 1.0/unum, vinc = 1.0/vnum;
                panels = new ArrayList<FramePanel>();
                for(int i=0; i <= unum; i++){
                  for(int j=0; j < vnum; j++){
                    if( (i+j)%2 == 0 ){

                      IVec c = surf.pt( i*uinc, j*vinc );
                      IVec nc = surf.normal( i*uinc, j*vinc ).len(.2);

                      IVec p1 = surf.pt( (i-1)*uinc, j*vinc );
                      IVec n1 = surf.normal( (i-1)*uinc, j*vinc
).len(.2);

                      IVec p2 = surf.pt( i*uinc, (j-1)*vinc );
                      IVec n2 = surf.normal( i*uinc, (j-1)*vinc
).len(.2);

                      IVec p3 = surf.pt( (i+1)*uinc, j*vinc );
                      IVec n3 = surf.normal( (i+1)*uinc, j*vinc
).len(.2);

                      IVec p4 = surf.pt( i*uinc, (j+1)*vinc );
                      IVec n4 = surf.normal( i*uinc, (j+1)*vinc
).len(.2);

                      IVec[] pts = {c,p1,p2,p3,p4};
                      IVec[] normals = {nc,n1,n2,n3,n4};

                      /*IVec xaxis = new IVec(1,0,0);
                      IVec yaxis = new IVec(0,1,0);
                      double[] ref = nc.projectTo2Vec(xaxis, yaxis);
                      IVec vRef = new IVec(ref[0],ref[1],ref[2]);
                      double openness = (nc.angle(vRef)/PApplet.
PI)*(openMax-openMin)+openMin;*/

                      double openness;
                      if(i<uMax & i>uMin & j<vMax & j>vMin){
                              openness = map.get( i*uinc, j*vinc
)*(openMax-openMin)+openMin;
                      }else{
                              openness = 1;
                      }

                      panels.add(new FramePanel(pts, normals, open-
ness, p));

                    }
                  }
                }

        }

}

/*************************************************

View Class

View - contains the scene(empty space and camera),
  imported 3d base model, and data model defining
  panel system. View calls draws to screen via
  viewState interface, though only one viewState
  is implemented: View3d, a basic 3d view.

Part of:

Aper[na]tures
Author: Joshua Parker
Date: 2011

Inheritted License: GPL, V3

*************************************************/
package view;

import processing.core.PApplet;
import model.Model;
import view.View1;
import view.ViewState;

public class View{

        //parent PApplet
        PApplet p;

        //data model
        public Model mymodel;

        //states
        private ViewState view1;
        private ViewState view2;

        //current state
        public ViewState curState;

        public View(PApplet parent, Model Mymodel){

                p = parent;

                this.mymodel = Mymodel;

                view1 = new View1(p, this);
                view2 = new View2(p, this);

                //view state
                setViewState(view1);
        }

        // PUBLIC METHODS ////////////////////////////////////////////
///////

        //set view1 state
        public void setView1(){
                setViewState(view1);
        }

        //set view2 state
        public void setView2(){
                setViewState(view2);
        }

        //draw the view
        public void draw() {
                this.curState.draw();
        }

        // PRIVATE METHODS ///////////////////////////////////////////
////////

        //set view state
        private void setViewState(ViewState newState){
                this.curState = newState;
        }
}
/*************************************************

View3d Class

View3d implements a basic 3d view

Part of:

Aper[na]tures
Author: Joshua Parker
Date: 2011

Inheritted License: GPL, V3

*************************************************/
package view;

import java.util.ArrayList;

import model.FramePanel;
import model.Model;
import processing.core.*;
import igeo.IBox;
import igeo.IBrep;
import igeo.ICurve;
import igeo.IG;
import igeo.IMesh;
import igeo.IPoint;
import igeo.ISurface;
import igeo.IVec;

public class View1 implements ViewState{

        //The parent PApplet
        PApplet p;

        //composit view
        public View view;
        public Model model;

        //CONSTRUCTOR
        View1(PApplet parent, View Myview){

                p = parent;

                this.view = Myview;

                //moved to model...
        }

        // PUBLIC METHODS ////////////////////////////////////////////
///////

        //draw the 3d view
        public void draw() {

                //clear geometry in the scene
                IG.cur().server().clear();

                //IG.current().panel.currentPane().getView().mode().
setDrawMode(true,false,false);

                //draw
                for(FramePanel panel : view.mymodel.panels){

                        //outer and inner corner pts of panel
                        IVec[] outer = new IVec[4];
                        IVec[] inner = new IVec[4];

                        //locate inner corners (already have outer)
                        for(int i=0; i<outer.length; i++){
                                outer[i] = panel.corners[i].dup();
                                inner[i] = panel.center.dup()
                                                .sub(outer[i])
        //get miter vector
                                                .scale(panel.openness)
//scale by openness
                                                .add(outer[i]);
        //position at corner
                        }

                        if(panel.openness==1){
                                new ISurface(outer[0], outer[1], out-
er[2], outer[3]);
                        }else{
                                //create mitered frame faces
                                for(int j=0; j<outer.length; j++){

                                        new ISurface(outer[j],
outer[(j+1)%4], inner[(j+1)%4], inner[j])
                                                .clr(panel.openness, 0,
0);
                                }
                        }
                }
        }//end draw

        // PRIVATE METHODS ////////////////////////////////////////////
////////

}
/*************************************************

View3d Class

View3d implements a basic 3d view

Part of:

Aper[na]tures
Author: Joshua Parker
Date: 2011

Inheritted License: GPL, V3

*************************************************/
package view;

import processing.core.*;
import saito.objloader.*;
import migeo.*;
import model.*;
import processing.core.PVector;
import igeo.IBrep;
import igeo.ICurve;
```

```
import igeo.IG;
import igeo.IMesh;
import igeo.IPoint;
import igeo.ISurface;
import igeo.IVec;

public class View2 implements ViewState{

	//The parent PApplet
	PApplet p;

	//composit view
	public View view;
	public Model model;

	//CONSTRUCTOR
	View2(PApplet parent, View Myview){

		p = parent;

		this.view = Myview;

		//moved to model...
	}

	// PUBLIC METHODS /////////////////////////////////////////////
///////

	//draw the 3d view
	public void draw() {

		//clear geometry in the scene
		IG.cur().server().clear();
		IG.bg(25, 25, 50, 50);

		//draw
		for(FramePanel panel : view.mymodel.panels){

			new ICurve(panel.corners[0], panel.corners[1]).
clr(150);
			new ICurve(panel.corners[1], panel.corners[2]).
clr(150);
		}


	}//end draw

	// PRIVATE METHODS ////////////////////////////////////////////
////////

}
/************************************************

ViewState Class

ViewState is an interface for various views

Part of:

Aper[na]tures
Author: Joshua Parker
Date: 2011

Inheritted License: GPL, V3

************************************************/
package view;

public interface ViewState{

	  //public void enableFill();
	  //public void disableFill();
	  //public boolean getFill();
	  public void draw();
	}
```

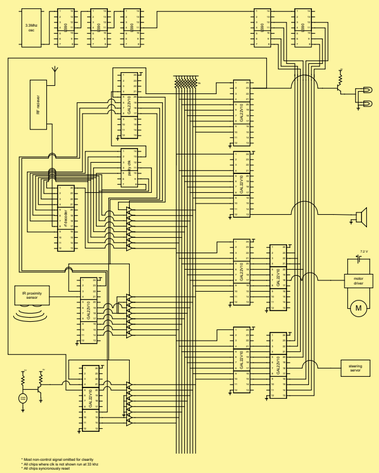
BEING THERE, SOURCE CODE

THE CAR PROJECT CONSISTS OF TWO MAIN MODULES. ONE OF THE MAIN MODULES IN THE USER INTERFACE CIRCUIT WHICH HANDLES ALL DATA COMING IN FROM THE USER AND SENDS IT OFF TO THE RF TRANSMITTER. THE OTHER MAIN MODULE RECEIVES THE INCOMING DATA FROM THREE SEPARATE SOURCES: THE RF RECEIVER, LIGHT SENSOR, AND PROXIMITY SENSOR. USING THESE SIGNALS IT CAN SUCCESSFULLY CONTROL THE STEERING, HORN, LIGHTS, AND MOTOR. THE LATTER CIRCUIT IS ACTUALLY ON THE RC CAR. THE USER WILL ALSO HAVE A REAL-TIME VIDEO FEED AVAILABLE TO BE ABLE TO WATCH WHERE THEY ARE GOING. THIS TRANSLATES TO A RC CAR CONTROLLED BY THE USER VIA VIDEO GAME CONTROLLER. THE STEERING WHEEL CONTROLLER WILL SEND DATA TO A PIC MICROCONTROLLER, WHICH WILL BE THE MAIN CONTROL UNIT ON THE USER SIDE OF THE SYSTEM. A DIGITAL READOUT OF THE CURRENT GEAR AND WELL AS STATUS LIGHTS WILL BE PRESENT. SRAM WILL BE AVAILABLE TO RECORD INPUT FROM THE USER. THE CAR WILL RESPOND TO COMMANDS BY THE USER VIA RF COMMUNICATION.

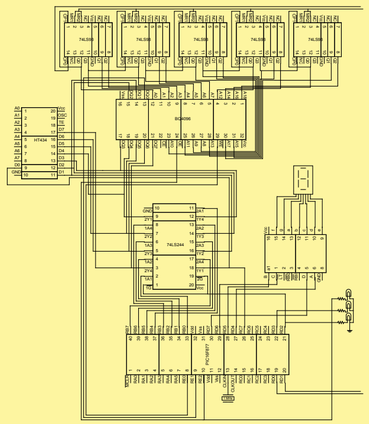
CONTRIBUTORS:
KEVIN LARSSON
JOSH PARKER
MATT ZOBEL

ADVIISORS:
JAMES K. PECKOL

1. main board schematic
2. control board schematic



1.



2.

The user module is controlled by the PIC microcontroller. The PIC accepts user input via the steering wheel, gas pedal, brake pedal, and buttons. Since the wheel, and pedals are potentiometer devices the PIC sample these measurements using the internal A/D converter. These measurements are then encoded internally in the PIC to compress some of the bits that are used. The PIC samples these devices continually and sends out three separate frames in sequence over RF transmission. The first frame contains speed information, the second contains turning information, and the third contains button push information.

To help detect errors a parity checker is used to calculate even parity and the parity bit is sent along with the transmission. The frames are sent to the RF encoder and then onto the RF transmitter for transmission to the car at 434MHz.

The PIC also controls some user displays according to the current state of the car. There is a gear display near the user that display the current gear the car is in. The PIC will increment or decrement the gear according to user input. The valid gears are 1,2,3,and R. In addition, the PIC controls 3 LED's that indicate the current status of the memory. The Green LED indicates that the car is under control of the user. The yellow LED indicates the memory is currently being written to. The Red LED means the memory is being read, nullifying any user input.

To control the memory the PIC uses 5 cascaded 74LS93 chips for addressing. The PIC sends out clock signals to the 74LS93 chips to increment them each time an address is written or read. The PIC is also responsible for clocking the SRAM to look at the current address and either read or write and is responsible for adding a flag bit to the last instruction written to memory. This is all done via the 8 bit parallel bus.

The car receives the data from the user module via RF transmission at 434MHz. The incoming data is sent through the RF decoder and onto the bus and into the parity checker. The Bus Master waits to get a valid transmit signal from the RF decoder. When Bus Master sees this signal he makes sure that the parity was correct. If so he strobes everyone listening to the bus to capture the new information.

Bus Master is also in charge of flipping the bus around when the proximity sensor chip or the light sensor chip have something to write to the bus. When the requests are made the Bus Master will flip the bus at the appropriate time and then flip it back when they are done.

After the Bus Master strobes the other nodes to capture the information on the bus, each node looks at the type of frame it is, as shown on Figure 5. Using bits 6 and 7 each node knows what type of frame to look at, if it is the wrong frame, it just ignores it. Speed decoder, steering decoder, horn controller, and light controller all listen to the bus continually.

Speed decoder and steering decoder decode the five-bit data stream in a speed frame or steering frame. With the input from the bus they decode the data into an eight-bit BCD value corresponding to the actual PWM signal that the generators will generate. For example, in a steering frame a 10000 sent by the user module gets decoded into a 42% duty cycle, therefore, the module puts out a 4 and 2 in BCD. The generators use 2 cascaded 74LS90 chips that are counting in BCD to do a comparison on. When the BCD input equals the BCD count, the PWM signal goes down and a PWM generation is produced.

The proximity sensor node and light sensor node have a little more intelligence. These nodes are responsible for writing to the bus to control the four nodes that are listening. If the light sensor senses a discrepancy between the status of the light and the status of the sensor, the light sensor node asks the bus master for bus control. When the control is granted the light sensor will write information to the bus to turn the light on or off appropriately.

he proximity sensor node works similarly to the light sensor. When the node receives information that the car is near a wall the node asks for control of the bus. When the control is granted, the proximity sensor node writes information to the bus, then to the speed controller, telling it to throw it into reverse. When the wall is no longer detected the proximity sensor then writes to the bus again telling the speed controller to go to idle. It must be noted that the bus is completely tied up by the proximity sensor node when the wall is near the car.

Using this implementation resulted in a successful project that operated correctly. Timing was the key ingredient and although we had some issues with timing in the beginning, in the end, it worked very well.

CUPL CODE

Busmaster.pld

```
Name       busmstr;
Partno     GAL22V10;
Date       05/06/02;
Revision   XX;
Designer   XXXXX;
Company    XXXXX;
Assembly   XXXXX;
Location   XXXXX;
Device     g22v10;

/*****************************************************************/
/*                                                               */
/*                     WINDOW/UNKNOWN                          */
/*                                                               */
/*****************************************************************/

/**  Inputs  **/

Pin  1         =  clk;
Pin  2         =  reset;

Pin  4         =  parity;
Pin  5         =  strobein;
Pin  6         =  proxsense;
Pin  7         =  lightsense;
Pin  8         =  validtransmit;

/**  Outputs  **/

Pin  17        =  lightenable;
Pin  18        =  proxenable;
Pin  19        =  mstrenable;
Pin  20        =  strobeout;
Pin  [21..23]  =  [s2..s0];


FIELD buscntrl = [s2..s0];

$DEFINE idle                'b'000
$DEFINE paritycheck         'b'001
$DEFINE strobe              'b'010
$DEFINE proxdetect          'b'011
$DEFINE proxwait            'b'100
$DEFINE lightdetect         'b'101
$DEFINE lightwait           'b'110
$DEFINE proxdetect2         'b'111


/** Declarations and Intermediate Variable Definitions **/

s0.ar          = reset;
s1.ar          = reset;
s2.ar          = reset;

s0.sp          = 'b'0;
s1.sp          = 'b'0;
s2.sp          = 'b'0;

s0.oe          = 'b'1;
s1.oe          = 'b'1;
s2.oe          = 'b'1;

sequence buscntrl
{

       present idle          /* 000 */
               default
next idle;
               if(proxsense)
next proxdetect;
               if(validtransmit & !proxsense)
next paritycheck;
               if(lightsense & !proxsense & !strobein)
next lightdetect;
       present paritycheck    /* 001 */
               default
next paritycheck;
               if(proxsense)
next proxdetect;
               if(!parity & !proxsense)
next idle;
               if(parity & !proxsense)
next strobe;

       present strobe               /* 010 */
               default
next idle;
               if(proxsense)
next proxdetect;

       present proxdetect     /* 011 */
               default
next proxdetect;
               if(strobein)
next proxdetect2;

       present proxdetect2    /* 111 */
               default
next proxdetect2;
               if(strobein)
next proxwait;

       present proxwait       /* 100 */
               default
next idle;

       present lightdetect    /* 101 */
               default
next lightdetect;
               if(proxsense)
next proxdetect;
               if(strobein & !proxsense)
next lightwait;

       present lightwait      /* 110 */
               default
next idle;
}


/**  Logic Equations  **/

idl            = !s2 & !s1 & !s0;
parchk         = !s2 & !s1 & s0;
str            = !s2 & s1 & !s0;
proxd          = !s2 & s1 & s0;
proxd2         = s2 & s1 & s0;
proxw          = s2 & !s1 & !s0;
lightd         = s2 & !s1 & s0;
lightw         = s2 & s1 & !s0;

strobeout = str;
mstrenable =  !(idl # parchk # str);         /* Drive Tri-State--Low
Enable */
proxenable =  !(proxd # proxw # proxd2);     /* Drive Tri-State--Low
Enable */
lightenable = !(lightd # lightw);


honker.pld

Name       honker;
Partno     GAL22V10;
Date       05/27/02;
Revision   XX;
Designer   XXXXX;
Company    XXXXX;
Assembly   XXXXX;
Location   XXXXX;
Device     g22v10;

/*****************************************************************/
/*                                                               */
/*                     WINDOW/UNKNOWN                          */
/*                                                               */
/*****************************************************************/

/**  Inputs  **/

Pin  1         =  clk;
```



```
Pin  2         =  reset;
Pin  3         =  strobe;
Pin  [4..10]   =  [bus1..bus7];


/**  Outputs  **/

Pin 13         = out_freq;
Pin 22         = s0;
Pin 23         = honkactive;


FIELD buscntrl = [s0];

$DEFINE off                 'b'0
$DEFINE on                  'b'1


/** Declarations and Intermediate Variable Definitions **/

s0.ar          = reset;
s0.sp          = 'b'0;
s0.oe          = 'b'1;


sequence buscntrl
{

       present off
               default
next off;
               if(strobe & bus2 & bus7 & !bus6)
next on;

       present on
               default
next on;
               if(strobe & !bus2 & bus7 & !bus6)
next off;
}


/**  Logic Equations  **/

honkactive = s0 & out_freq;

lighter.pld

Name       lighter;
Partno     GAL22V10;
Date       05/27/02;
Revision   XX;
Designer   XXXXX;
Company    XXXXX;
Assembly   XXXXX;
Location   XXXXX;
Device     g22v10;

/*****************************************************************/
/*                                                               */
/*                     WINDOW/UNKNOWN                          */
/*                                                               */
/*****************************************************************/

/**  Inputs  **/

Pin  1         =  clk;
Pin  2         =  reset;
Pin  3         =  strobe;
Pin  [4..10]   =  [bus1..bus7];


/**  Outputs  **/

Pin 22         = s0;
Pin 23         = lightactive;


FIELD buscntrl = [s0];

$DEFINE off                 'b'0
$DEFINE on                  'b'1


/** Declarations and Intermediate Variable Definitions **/

s0.ar          = reset;
s0.sp          = 'b'0;
s0.oe          = 'b'1;


sequence buscntrl
{

       present off
               default
next off;
               if(strobe & bus1 & bus7 & !bus6)
next on;

       present on
               default
next on;
               if(strobe & bus1 & bus7 & !bus6)
next off;
}


/**  Logic Equations  **/

lightactive = s0;

lightsense.pld

Name       lsense;
Partno     GAL22V10;
Date       05/06/02;
Revision   XX;
Designer   XXXXX;
Company    XXXXX;
Assembly   XXXXX;
Location   XXXXX;
Device     g22v10;

/*****************************************************************/
/*                                                               */
/*                     WINDOW/UNKNOWN                          */
/*                                                               */
/*****************************************************************/

/**  Inputs  **/

Pin  1         =  clk;
Pin  2         =  reset;
Pin  3         =  !flipenable;
Pin  4         =  !lightsense;
Pin  5         =  lightstatus;
Pin  6         =  honkstatus;


/**  Outputs  **/

Pin  14        =  strobe;
Pin  15        =  bus1;
Pin  16        =  bus2;
Pin  17        =  fliprequest;
Pin  [18..20]  =  [s0..2];
Pin  21        =  bus6;
Pin  22        =  bus7;



FIELD lightcntrl = [s2..s0];

$DEFINE idle                'b'000
$DEFINE flipreq             'b'001
$DEFINE datasend            'b'010
$DEFINE strobey                   'b'011
$DEFINE wait                'b'100


/** Declarations and Intermediate Variable Definitions **/

s0.ar     = reset;
s1.ar     = reset;
s2.ar     = reset;

s0.sp     = 'b'0;
s1.sp     = 'b'0;
s2.sp     = 'b'0;
```

```
s0.oe     = 'b'1;
s1.oe     = 'b'1;
s2.oe     = 'b'1;

sequence lightcntrl
{

       present idle
              default
next idle;
              if((lightsense $ lightstatus))
next flipreq;


       present flipreq
              default
next flipreq;
              if(flipenable)
next datasend;


       present datasend
              default
next strobey;


       present strobey
              default
next wait;


       present wait
              default
next idle;

}

/**  Logic Equations  **/

strobe          = !s2 & s1 & s0;
bus1            = 'b'1;
bus2            = honkstatus;
bus6            = 'b'0;
bus7            = 'b'1;
fliprequest     = !s2 & !s1 & s0;


Proximity Sense.pld


Name       psense;
Partno     GAL22V10;
Date       05/06/02;
Revision   XX;
Designer   XXXXX;
Company    XXXXX;
Assembly   XXXXX;
Location   XXXXX;
Device     g22v10;

/*************************************************************/
/*                                                           */
/*                      WINDOW/UNKNOWN                       */
/*                                                           */
/*************************************************************/

/**  Inputs  **/

Pin  1         =  clk;
Pin  2         =  reset;
Pin  3         =  !flipenable;
Pin  4         =  proxsense;

/**  Outputs  **/

Pin  14        =  strobe;
Pin  [15..19]  =  [bus1..5];  /* Tie Bus 6 & 7 to Pin 18 */
Pin  [20..22]  =  [s0..2];
Pin  23        =  fliprequest;


FIELD proxcntrl = [s2..s0];

$DEFINE idle                'b'000
$DEFINE flipreq             'b'001
```



```
$DEFINE rev_send            'b'010
$DEFINE rev_strobey         'b'011
$DEFINE rev_wait            'b'100
$DEFINE idle_send           'b'101
$DEFINE idle_strobey        'b'110
$DEFINE idle_wait           'b'111

/** Declarations and Intermediate Variable Definitions **/

s0.ar     = reset;
s1.ar     = reset;
s2.ar     = reset;


s0.sp     = 'b'0;
s1.sp     = 'b'0;
s2.sp     = 'b'0;

s0.oe     = 'b'1;
s1.oe     = 'b'1;
s2.oe     = 'b'1;

sequence proxcntrl
{

       present idle
              default
next idle;
              if(proxsense)
next flipreq;


       present flipreq
              default
next flipreq;
              if(flipenable)
next rev_send;


       present rev_send
              default
next rev_strobey;


       present rev_strobey
              default
next rev_wait;

       present rev_wait
              default
next rev_wait;
              if(!proxsense)
next idle_send;

       present idle_send
              default
next idle_strobey;

       present idle_strobey
              default
next idle_wait;

       present idle_wait
              default
next idle;

}


/**  Logic Equations  **/

strobe = (!s2 & s1 & s0) # (s2 & s1 & !s0);
bus1   = 'b'0;
bus2   = (!s2 & s1 & !s0) # (!s2 & s1 & s0) # (s2 & !s1 & !s0);
bus3   = (!s2 & s1 & !s0) # (!s2 & s1 & s0) # (s2 & !s1 & !s0);
bus4   = 'b'0;
bus5   = (s2 & !s1 & s0) # (s2 & s1 & !s0) # (s2 & s1 & s0);
fliprequest = !s2 & !s1 & s0;


Speed Generator.pld


Name       spedpw90;
Partno     GAL22V10;
```

```
Date       05/27/02;
Revision   XX;
Designer   XXXXX;
Company    XXXXX;
Assembly   XXXXX;
Location   XXXXX;
Device     g22v10;

/*************************************************************/
/*                                                           */
/*                      WINDOW/UNKNOWN                       */
/*                                                           */
/*************************************************************/

/**  Inputs  **/

Pin  1         =  clk;
Pin  2         =  reset;

Pin  [3..6]    =  [MSB3..MSB0];
Pin  [7..10]   =  [LSB3..LSB0];
Pin  [13..16]  =  [clkMSB0..clkMSB3];
Pin  [20..23]   =  [clkLSB0..clkLSB3];

/**  Outputs  **/
Pin  18        =  !a0;        /* this is the PWM output to the motor
*/
Pin  19        =  temp;

FIELD pwm    = [a0];

/** Declarations and Intermediate Variable Definitions **/

a0.ar          = reset;
a0.sp          = 'b'0;
a0.oe          = !reset;




temp =                (LSB0 $ clkLSB0) #
                      (LSB1 $ clkLSB1) #
                      (LSB2 $ clkLSB2) #
                      (LSB3 $ clkLSB3) #
                      (MSB0 $ clkMSB0);

sequence pwm
{
       present 'b'0
              default
next 'b'0;
              if(!(   temp            #
                      (MSB1 $ clkMSB1) #
                      (MSB2 $ clkMSB2) #
                      (MSB3 $ clkMSB3) ))
next 'b'1;


       present 'b'1
              default
next 'b'1;
              if(!clkLSB0 & !clkLSB1 & !clkLSB2 & !clkLSB3 &
                 !clkMSB0 & !clkMSB1 & !clkMSB2 & !clkMSB3)
next 'b'0;

}

Speed Decoder.pld


Name       speeder;
Partno     GAL22V10;
Date       05/27/02;
Revision   XX;
Designer   XXXXX;
Company    XXXXX;
Assembly   XXXXX;
Location   XXXXX;
Device     g22v10;

/*************************************************************/
/*                                                           */
/*                      WINDOW/UNKNOWN                       */
/*                                                           */
/*************************************************************/
```

```
/**  Inputs  **/

Pin  1         =  clk;               /* this is the Strobe off the
bus, bus bit 0 */
Pin  2         =  reset;

Pin  [4..10]   =  [bus1..bus7];


/**  Outputs  **/

Pin [14..17]   = [LSBdutycycle0..LSBdutycycle3];
Pin [18..21]   = [MSBdutycycle0..MSBdutycycle3];


/* intermediates */

MSBdutycycle3.ar = reset;
MSBdutycycle2.ar = reset;
MSBdutycycle1.ar = reset;
MSBdutycycle0.ar = reset;

LSBdutycycle3.ar = reset;
LSBdutycycle2.ar = reset;
LSBdutycycle1.ar = reset;
LSBdutycycle0.ar = reset;

MSBdutycycle3.sp = 'b'0;
MSBdutycycle2.sp = 'b'0;
MSBdutycycle1.sp = 'b'0;
MSBdutycycle0.sp = 'b'0;

LSBdutycycle3.sp = 'b'0;
LSBdutycycle2.sp = 'b'0;
LSBdutycycle1.sp = 'b'0;
LSBdutycycle0.sp = 'b'0;

MSBdutycycle3.oe = 'b'1;
MSBdutycycle2.oe = 'b'1;
MSBdutycycle1.oe = 'b'1;
MSBdutycycle0.oe = 'b'1;

LSBdutycycle3.oe = 'b'1;
LSBdutycycle2.oe = 'b'1;
LSBdutycycle1.oe = 'b'1;
LSBdutycycle0.oe = 'b'1;


speed_frame = !bus6 & !bus7;  /* Speed Frame, 00 */

MSBdutycycle3.d = (speed_frame & 'b'0) # (!speed_frame & MSBdutycy-
cle3);

MSBdutycycle2.d = (speed_frame & 'b'1) # (!speed_frame & MSBdutycy-
cle2);

MSBdutycycle1.d = (speed_frame & (    (bus5 & bus4 & (bus3 # bus2)))) #
(!speed_frame & MSBdutycycle1);

MSBdutycycle0.d = (speed_frame & (    (bus5 & (!bus4 # (bus4 & !bus3 &
!bus2))))) # (!speed_frame & MSBdutycycle0);


LSBdutycycle3.d = (speed_frame & (    (bus5 & bus4 & !bus3 & !bus2))) #
(!speed_frame & LSBdutycycle3);

LSBdutycycle2.d = (speed_frame & (    (bus3 & (!bus4 # bus2)))) #
(!speed_frame & LSBdutycycle2);

LSBdutycycle1.d = (speed_frame & (    (((!bus4 & bus2) # (bus4 & bus3 &
!bus2)))) # (!speed_frame & LSBdutycycle1) ;

LSBdutycycle0.d = (speed_frame & (    (bus1))) # (!speed_frame & LSBdu-
tycycle0);


Steering Decoder.pld


Name       steerer;
Partno     GAL22V10;
Date       05/27/02;
Revision   XX;
Designer   XXXXX;
Company    XXXXX;
Assembly   XXXXX;
```

```
Location   XXXXX;
Device     g22v10;

/*****************************************************************/
/*                                                               */
/*                     WINDOW/UNKNOWN                          */
/*                                                               */
/*****************************************************************/

/**  Inputs  **/

Pin  1         =  clk;               /* this is the Strobe off the
bus, bus bit 0 */
Pin  2         =  reset;

Pin  [4..10]   =  [bus1..bus7];


/**  Outputs  **/

Pin [14..17]   = [LSBdutycycle0..LSBdutycycle3];
Pin [18..21]   = [MSBdutycycle0..MSBdutycycle3];


MSBdutycycle3.ar = reset;
MSBdutycycle2.ar = reset;
MSBdutycycle1.ar = reset;
MSBdutycycle0.ar = reset;

LSBdutycycle3.ar = reset;
LSBdutycycle2.ar = reset;
LSBdutycycle1.ar = reset;
LSBdutycycle0.ar = reset;

MSBdutycycle3.sp = 'b'0;
MSBdutycycle2.sp = 'b'0;
MSBdutycycle1.sp = 'b'0;
MSBdutycycle0.sp = 'b'0;

LSBdutycycle3.sp = 'b'0;
LSBdutycycle2.sp = 'b'0;
LSBdutycycle1.sp = 'b'0;
LSBdutycycle0.sp = 'b'0;

MSBdutycycle3.oe = 'b'1;
MSBdutycycle2.oe = 'b'1;
MSBdutycycle1.oe = 'b'1;
MSBdutycycle0.oe = 'b'1;

LSBdutycycle3.oe = 'b'1;
LSBdutycycle2.oe = 'b'1;
LSBdutycycle1.oe = 'b'1;
LSBdutycycle0.oe = 'b'1;


steering_frame = bus6 & !bus7;  /* Directional Frame, 01 */

MSBdutycycle3.d = (steering_frame & 'b'0) # (!steering_frame  & MSBdu-
tycycle3);

MSBdutycycle2.d = (steering_frame & bus5) # (!steering_frame  & MSBdu-
tycycle2);

MSBdutycycle1.d = (steering_frame & !bus5) # (!steering_frame  & MSB-
dutycycle1);

MSBdutycycle0.d = (steering_frame & ( (!bus5 & (bus4 # (bus3 & bus2)))#
                                      (bus5 & bus4 & (bus3 # bus2)))) #
(!steering_frame  & MSBdutycycle0);


LSBdutycycle3.d = (steering_frame & ( (bus5 & bus4 & !bus3 & !bus2)
#
                                      (!bus5 & !bus4 & bus3 & !bus2)
#
                                      (!bus5 & bus4 & bus3 & bus2)))  #
(!steering_frame  & LSBdutycycle3);

LSBdutycycle2.d = (steering_frame & ( (bus5 & bus4 & bus3 & bus2)   #
                                      (bus5 & !bus4 & bus3)         #
                                      (!bus5 & !bus4 & !bus3)
#
                                      (!bus5 & bus4 & (bus3 $ bus2))))#
(!steering_frame & LSBdutycycle2);

LSBdutycycle1.d = (steering_frame & ( (bus5 & bus4 & bus3 & !bus2)  #
                                      (bus5 & !bus4 & bus2)         #
                                      (!bus5 & !bus3 & (bus4 $ bus2))))
# (!steering_frame  & LSBdutycycle1);

LSBdutycycle0.d = (steering_frame  & (       (bus1))) # (!steering_
frame  & LSBdutycycle0);

Steering Generator.pld

Name       sterpw90;
Partno     GAL22V10;
Date       05/27/02;
Revision   XX;
Designer   XXXXX;
Company    XXXXX;
Assembly   XXXXX;
Location   XXXXX;
Device     g22v10;

/*****************************************************************/
/*                                                               */
/*                     WINDOW/UNKNOWN                          */
/*                                                               */
/*****************************************************************/

/**  Inputs  **/

Pin  1         =  clk;
Pin  2         =  reset;

Pin  [3..6]    =  [MSB3..MSB0];
Pin  [7..10]   =  [LSB3..LSB0];
Pin  [13..16]  =  [clkMSB0..clkMSB3];
Pin  [20..23]   =  [clkLSB0..clkLSB3];

/**  Outputs  **/
Pin  18        =  !a0;        /* this is the PWM output to the servos
*/
Pin  19        =  temp;

FIELD pwm    = [a0];

/** Declarations and Intermediate Variable Definitions **/

a0.ar          = reset;
a0.sp          = 'b'0;
a0.oe          = !reset;



temp =                 (LSB0 $ clkLSB0) #
                       (LSB1 $ clkLSB1) #
                       (LSB2 $ clkLSB2) #
                       (LSB3 $ clkLSB3) #
                       (MSB0 $ clkMSB0);

sequence pwm
{
       present 'b'0
               default
next 'b'0;
               if(!(   temp            #
                       (MSB1 $ clkMSB1) #
                       (MSB2 $ clkMSB2) #
                       (MSB3 $ clkMSB3) ))
next 'b'1;


       present 'b'1
               default
next 'b'1;
               if(!clkLSB0 & !clkLSB1 & !clkLSB2 & !clkLSB3 &
                  !clkMSB0 & !clkMSB1 & !clkMSB2 & !clkMSB3)
next 'b'0;

}



PIC CODE

#define MAXGEAR 3
```



```
#define IDLE 16

char TRISC@0x87;
char TRISD@0x88;
char TRISE@0x89;
char PORTC@0x07;   // Button inputs from steering wheel
char PORTD@0x08;   // Gear output display and PIC output enable
Tristate
char PORTE@0x09;   // Enable signals to 512kx8 memory
char ADCON0@0x1f;
char ADCON1@0x9f;
char ADRESH@0x1e;



int  AdcWheel,AdcGas,AdcBrake;
char AdcWheelState,AdcGasState,AdcBrakeState,AdcSpeedState;
char CurrentGear,GearUp,GearDn;
char CurrentButtons,ButtonChange;
char MemWrite,MemRead,MemLast;

fAdc(char Channel);

void fGear(char Current);

void fButtons(void);

fOut(char Data, char State);

fStrobe(void);

bit_set(char Data, char Index);

main()
{



    TRISB = 0x00;       //Set PORTB to all output
    PORTB = 0x00;
    TRISD = 0x00;       //Set PORTD to all output
    PORTD = 0x03;       //b00000011
    TRISE = 0x00;       //Set PORTE to all output
    PORTE = 0x05;       //b00000101
    TRISA = 0xff;       //Set PORTA to all input
    TRISC = 0xff;       //Set PORTC to all input


    ADCON0 = 0x81;      //Configure and turn on A/D Module
    ADCON1 = 0x02;      //Set PORTA to analog and RIGHT justify result
                        //with PORTE on I/O

    CurrentGear = 1;
    GearUp = 0;
    GearDn = 0;
    MemRead = 0;
    MemWrite = 0;
    MemLast = 0;
    ButtonChange = 0;
    CurrentButtons = 0;

    int iter;

    delay_ms(2);

    while(1)
    {

        iter = 0;

        for(iter;iter<3;iter++)
        {

            fGear(PORTC);

            if(iter == 0)
            {
                AdcGasState   = fAdc(1);
                AdcBrakeState = fAdc(2);


                if((AdcBrakeState < IDLE) && (CurrentGear > 0))
                {
                    AdcSpeedState = IDLE;
                    CurrentGear = 1;
                }
                else if((AdcGasState > IDLE) && (AdcBrakeState ==
IDLE) &&
                        (CurrentGear > 0))
                    AdcSpeedState = AdcGasState;
                else if((AdcBrakeState < IDLE) && (CurrentGear == 0))
                    AdcSpeedState = AdcBrakeState;
                else
                    AdcSpeedState = IDLE;

                PORTB = fOut(AdcSpeedState,iter);

            }

            else if(iter == 1)
            {
                AdcWheelState = fAdc(0);
                PORTB = fOut(AdcWheelState,iter);
            }

            else
            {
                fButtons();
                PORTB = fOut(CurrentButtons,iter);
                if(MemLast == 1)          // Send one last data frame
with
                    set_bit(PORTE,1);     // a flag to determine end of
data
                else
                    clear_bit(PORTE,1);
            }

            if((MemWrite == 1) || (MemLast == 1) || (MemRead == 1))
            {
                clear_bit(PORTD,1);       // Send a Clock pulse to the
74LS93
                set_bit(PORTD,1);         // for a new address

                clear_bit(PORTD,0);       // Count Addresses on
74LS93s
            }
            else
                set_bit(PORTD,0);         // Reset Address on 74LS93s
for new
                                          // sequence

            delay_ms(50);                 // Shortest amount of delay
for RF
                                          // Transmission

            if((MemWrite == 1) || (MemLast == 1))
            {
                clear_bit(PORTE,0);       // Send a Write enable sig-
nal to
                set_bit(PORTE,0);         // the 512kx8
            }

            fStrobe();                    // Data is valid, send to
car



        }
    }
}



fAdc(char Channel)
{

    char Adc0 = 0;
    char Adc1 = 0;
    char Adc2 = 0;

    switch(Channel)            //Select which ADC channel
    {
        case 0:
            ADCON0 = 0x81;    //ADC 1 Enable AtoD
            break;
        case 1:
            ADCON0 = 0x89;    //ADC 2 Enable AtoD
            break;
        case 2:
            ADCON0 = 0x91;    //ADC 3 Enable AtoD
            break;
        default:
```

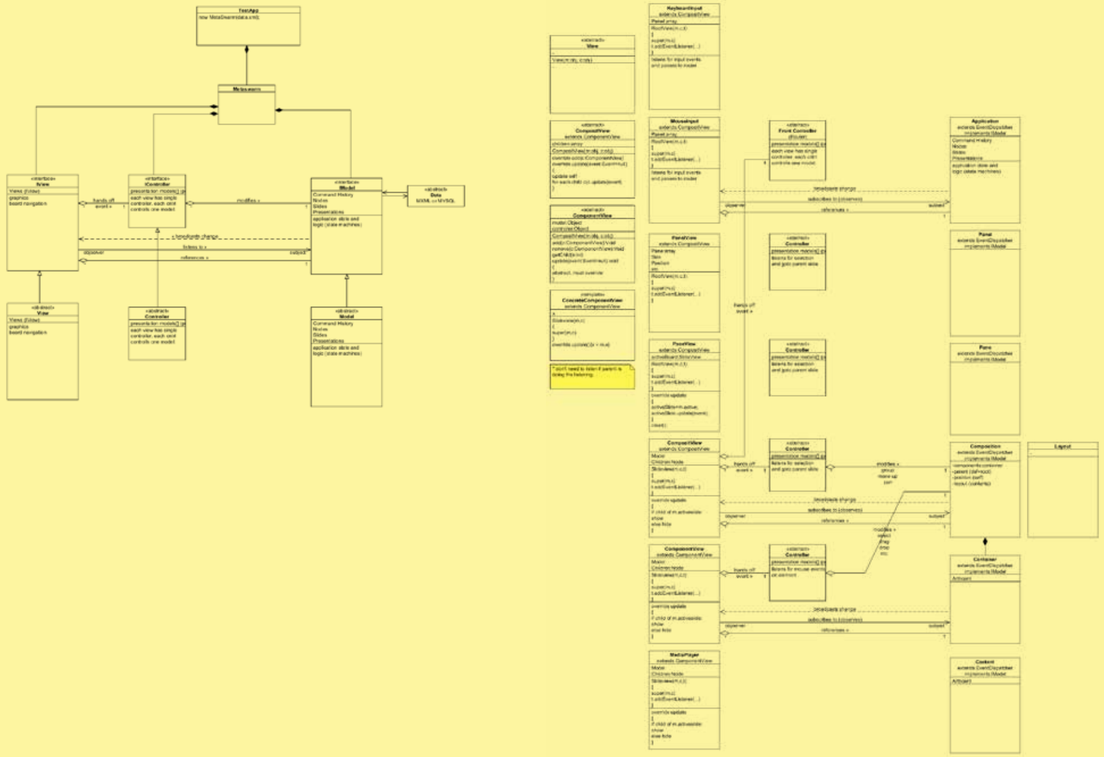
```
            ADCON0 = 0x81;    //ADC 1 Enable AtoD
            break;
    }

    delay_us(100);

    set_bit(ADCON0,2);

    while(bit_set(ADCON0,2) == 1);



    switch(Channel)
    {
        case 0:    // Calculate ADC for Wheel position

            Adc0 = ADRESH;

            if(Adc0 >= 96)
                return 31;
            else if((Adc0 >= 68) && (Adc0 < 96))
                return ((Adc0 - 34)/2);
            else if((Adc0 >= 60) && (Adc0 < 68))
                return IDLE;
            else if((Adc0 >= 12) && (Adc0 < 60))
                return ((Adc0 - 12)/3);
            else if(Adc0 < 12)
                return 0;
            else
                return IDLE;
            break;

        case 1:    // Calculate ADC for Gas pedal position

            Adc1 = ADRESH;

            if(CurrentGear > 0)
            {
                if((Adc1 >= 220) && (CurrentGear  <= 1))  // Max 224
Min 181
                    return (IDLE + (4 * (CurrentGear - 1)));
                else if((Adc1 >= 220) && (CurrentGear > 1))
                    return (17 + (4 * (CurrentGear - 1)));
                else if((Adc1 >= 208) && (Adc1 < 220))
                    return (18 + (4 * (CurrentGear - 1)));
                else if((Adc1 >= 195) && (Adc1 < 208))
                    return (19 + (4 * (CurrentGear - 1)));
                else
                    return (20 + (4 * (CurrentGear - 1)));
            }
            else
                return IDLE;
            break;

        case 2:    // Calculate ADC for Brake pedal position

            Adc2 = ADRESH;

            if(Adc2 < 110)    // Up 80    Dn 200
                return IDLE;
            else if((Adc2 >= 110) && (Adc2 < 155))
                return 6;
            else if((Adc2 >= 155) && (Adc2 < 175))
                return 5;
            else if(Adc2 >= 175)
                return 4;
            else
                return IDLE;
            break;

        default:

            return IDLE;
            break;

    }

}



void fGear(char Current)   // Calculates which gear the car should be
{                          // in according to paddle pushes

    if(((bit_set(Current,4) != 1) && (CurrentGear < MAXGEAR)) && (Gea-
rUp == 0))
    {
        CurrentGear = CurrentGear + 1;
        GearUp = 1;
    }
    else if((bit_set(Current,4) == 1) && (GearUp == 1))
        GearUp = 0;

    if(((bit_set(Current,5) != 1) && (CurrentGear > 0)) && (GearDn ==
0))
    {
        CurrentGear = CurrentGear - 1;
        GearDn = 1;
    }
    else if((bit_set(Current,5) == 1) && (GearDn == 1))
        GearDn = 0;

    if(CurrentGear == 0)        //Use upper 4 bits of PORTD
    {
        set_bit(PORTD,7);      //  1100XXXXb  Output a 12 for Reverse
        set_bit(PORTD,6);
        clear_bit(PORTD,5);
        clear_bit(PORTD,4);
    }
    else if(CurrentGear == 1)
    {
        clear_bit(PORTD,7);    // 0001XXXXb  Output a 1 for 1st
        clear_bit(PORTD,6);
        clear_bit(PORTD,5);
        set_bit(PORTD,4);
    }
    else if(CurrentGear == 2)
    {
        clear_bit(PORTD,7);    // 0010XXXXb  Output a 2 for 2nd
        clear_bit(PORTD,6);
        set_bit(PORTD,5);
        clear_bit(PORTD,4);
    }
    else if(CurrentGear == 3)
    {
        clear_bit(PORTD,7);    // 0011XXXXb  Output a 3 for 3rd
        clear_bit(PORTD,6);
        set_bit(PORTD,5);
        set_bit(PORTD,4);
    }
    else
    {
        clear_bit(PORTD,7);    // 0001XXXXb  Output a 1 for 1st
        clear_bit(PORTD,6);
        clear_bit(PORTD,5);
        set_bit(PORTD,4);
    }

}


void fButtons(void)
{

    CurrentButtons = ~PORTC;
    clear_bit(CurrentButtons,4);

    MemLast = 0;

    if(MemRead == 0)    // WRITE TO MEMORY
    {
        if((bit_set(CurrentButtons,2) == 1) && (ButtonChange == 0))
        {
            ButtonChange = 1;
            MemWrite = 1;
            set_bit(PORTD,3);
        }
        else if((bit_set(CurrentButtons,2) == 0) && (ButtonChange ==
1))
            ButtonChange = 2;
        else if((bit_set(CurrentButtons,2) == 1) && (ButtonChange ==
2))
            ButtonChange = 3;
        else if((bit_set(CurrentButtons,2) == 0) && (ButtonChange ==
3))
        {
            ButtonChange = 0;
            MemLast = 1;
            MemWrite = 0;
            clear_bit(PORTD,3);
        }
    }
```



```
    if((MemWrite == 0) && (MemLast == 0))  // READ FROM MEMORY
    {


        if((bit_set(CurrentButtons,3) == 1) && (ButtonChange == 0))
        {
            set_bit(PORTD,2);         // Disable PIC data output
            delay_us(25);
            ButtonChange = 1;
            MemRead = 1;
            clear_bit(PORTE,2);       // Enable Mem data output
        }
        else if((bit_set(CurrentButtons,3) == 0) && (ButtonChange == 1))
            ButtonChange = 2;
        else if((bit_set(CurrentButtons,3) == 1) && (ButtonChange == 2))
            ButtonChange = 3;
        else if((bit_set(CurrentButtons,3) == 0) && (ButtonChange == 3))
        {
            set_bit(PORTE,2);         // Disable Mem data output
            delay_us(25);
            ButtonChange = 0;
            MemRead = 0;
            clear_bit(PORTD,2);       // Enable PIC data output
        }

        if((MemRead == 1) && (bit_set(CurrentButtons,7) == 0))
        {
            ButtonChange = 3;         // Reached end of memory
            set_bit(PORTE,2);         // return to User control
        }
    }

}


fOut(char Data, char State)    //Formats the Data char to be ready
{                              //for output by placing an address
    char tempData = Data << 1; //at the MSB and shifting left 1 to
    clear_bit(tempData,0);     //make room for a parity bit

    switch(State)
    {
        case 0:
            clear_bit(tempData,7);
            clear_bit(tempData,6);
            return tempData;
            break;

        case 1:
            clear_bit(tempData,7);
            set_bit(tempData,6);
            return tempData;
            break;

        case 2:
            set_bit(tempData,7);
            clear_bit(tempData,6);
            return tempData;
            break;

    }
}

fStrobe(void)
{
    set_bit(PORTB,0);
    clear_bit(PORTB,0);
}


bit_set(char Data, char Bit)  //Checks to see if the selected
{                             //bit in a char is high or low
    switch(Bit)
    {
        case 0:

            if((Data % 2) == 1)
                return 1;
            else
                return 0;
            break;

        case 1:

            if(((Data % 4) / 2) == 1)
```

META-SWARM, SOURCECODE

WORKING PROTOTYPE SOURCE CODE FOR METASWARM VISUAL COMMUNICATION PLATFORM. INITIALLY DEVELOPED UI IN JAVA, THEN REBUILT IN ACTIONSCRIPT 3. IMPLEMENTED WITH VARIENT OF MODEL-VIEW-CONTROLLER DESIGN PATTERN: VIEW CONTAINS GRAPHIC ELEMENTS. EACH VIEW HAS A SINGLE CONTROLLER; EACH CONTROLLER CONTROLS ONE MODEL. MODEL HOLDS APPLICATION STATE AND LOGIC (STATE MACHINES). FLOW IS AS FOLLOWS: VIEWS HAND UI EVENTS TO CONTROLLER, WHICH MODIFIES MODEL. VIEW LISTENS FOR MODEL UPDATES AND UPDATES UI IN TURN.




```
package  {
    import flash.display.DisplayObject;
    import flash.display.Graphics;
    import flash.display.Shape;
        import flash.display.Sprite;
        import flash.events.Event;
        import flash.events.KeyboardEvent;
        import flash.events.MouseEvent;
        import flash.ui.Keyboard;
        import flash.net.*;
        import flash.utils.*;
        import flash.text.TextField;
        import metaswarm.ui.hoverlabel.HoverLabel;
        import metaswarm.ui.sidebar.Sidebar;
        import metaswarm.swarm.Swarm;
        import metaswarm.build.Builder;
        import metaswarm.control.Controller;
        import metaswarm.build.Preloader;
        import metaswarm.control.Router;
        import metaswarm.ui.Ui;

        [SWF(width='1920',height='1200',backgroundColor='#F7F7F7',frame
Rate='25')]

    public class Metaswarm extends Sprite
    {
                //declarations
                private var _preloader:Preloader;
                private var _builder:Builder;
                private var _router:Router;
                private var _controller:Controller;
                private var _ui:Ui;

        public function Metaswarm():void
                {
                        ui = new Ui();
                        builder = new Builder(this);
                        preloader = new Preloader(builder);
                        controller = new Controller(ui); //removed hov-
erLabel param
                        router = new Router(controller);

                        init();
                }

                private function init():void
                {
                        //this.stage.quality = "low";

                        //register ui listeners
                        addEventListener(Event.ENTER_FRAME,
                                router.onEnterFrame);
                        stage.addEventListener(KeyboardEvent.KEY_DOWN,
                                router.onkeyDown);
                        stage.addEventListener(KeyboardEvent.KEY_UP,
                                router.onkeyUp);
                        addEventListener(MouseEvent.CLICK,
                                router.onMouseClick);
                        addEventListener(MouseEvent.MOUSE_DOWN,
                                router.onMouseDown);
                        addEventListener(MouseEvent.MOUSE_UP,
                                router.onMouseUp);
                        addEventListener(MouseEvent.MOUSE_OVER,
                                router.onMouseOver);
                        addEventListener(MouseEvent.MOUSE_OUT,
                                router.onMouseOut);

                        //add ui stage
                        addChild(ui);
                }

                public function get controller():Controller
                {
                        return _controller;
                }

                public function set controller(value:Controller):void
                {
                        _controller = value;
                }

                public function get router():Router
                {
                        return _router;
                }

                public function set router(value:Router):void
                {
                        _router = value;
                }

                public function get builder():Builder
                {
                        return _builder;
                }

                public function set builder(value:Builder):void
                {
                        _builder = value;
                }

                public function get preloader():Preloader
                {
                        return _preloader;
                }

                public function set preloader(value:Preloader):void
                {
                        _preloader = value;
                }

                public function get ui():Ui
                {
                        return _ui;
                }

                public function set ui(value:Ui):void
                {
                        _ui = value;
                }


    }//end class
}//end pkg
*****************************************************
package     {
        import flash.display.DisplayObject;
        import flash.display.Sprite;
        import metaswarm.node.components.dot.Dot;
        import metaswarm.node.components.icon.Icon;
        import metaswarm.node.nodestates.DraggingState;
        import metaswarm.node.nodestates.DroppedState;
        import metaswarm.node.nodestates.IdleState;
        import metaswarm.node.nodestates.PickedState;
        import metaswarm.node.nodestates.PoppedState;
        import metaswarm.node.nodestates.ThumbDraggingState;
        import metaswarm.node.components.databox.DataBox;
        import metaswarm.node.components.tag.Tag;
        import metaswarm.node.components.thumb.Thumb;
        import metaswarm.node.NodeState;

        public class Node extends Sprite{

                //omit constants for now
                //var nodeLoader:NodeLoader;

                public var dat:Array; //again using key word

                //var dat:Array;
                public var xpos:Array;
                public var ypos:Array;
                //var xtmp:Array;
                //var ytmp:Array;

                //internal composite objects
                public var thumb:Thumb;
                public var icon:Icon;
                public var dot:Dot;
                public var tag:Tag;
                public var dataBox:DataBox;
                //var img:Image;

                //internal states
                public var idle:NodeState;
                public var picked:NodeState;
                public var dragging:NodeState;
                public var dropped:NodeState;
                public var tdragging:NodeState;
                public var popped:NodeState;

                //internal state holder
                public var nodeState:NodeState;
```

```
		public var prevState:NodeState;

		//constructor:
		public function Node(){

			//rediculous but don't know how else to initial-
ize an exact-sized 2d array
			xpos = new Array(new Array(), new Array(), new
Array(), new Array(), new Array(), new Array(), new Array(), new Ar-
ray(), new Array());

			ypos = new Array(new Array(), new Array(), new
Array(), new Array(), new Array(), new Array(), new Array(), new Ar-
ray(), new Array());


			//nodeLoader = new NodeLoader(this);

			idle = new IdleState(this);
			picked = new PickedState(this);
			dragging = new DraggingState(this);
			tdragging = new ThumbDraggingState(this);
			dropped = new DroppedState(this);
			popped = new PoppedState(this);

			dot = new Dot();
			icon = new Icon();
			thumb = new Thumb();
			tag = new Tag();
			dataBox = new DataBox();
			//img = new Image();

			init();
		}
		public function init():void{
			nodeState = idle;
		}
		/*function load():void{
			//trace("node.load()");
			nodeLoader.load();
		}
		function make():void{
			//trace("node.make()");
			nodeLoader.make();
		}
		function go():void{
			//trace("node.go()");
			nodeLoader.go();
		}*/
		function go():void{
			//trace("node.go()");
			nodeLoader.go();
		}*/

		//should this be delegated???--no!
		public function popout():void{
			nodeState.popout();
		}

		public function popin():void{
			nodeState.popin();
		}

		public function close():void{
			nodeState.close();
		}

		public function isIdle():Boolean{
			//trace("isidle?");
			if(nodeState == idle){
				return true;
			}else{
				return false;
			}
		}

		public function isPicked():Boolean{
			if(nodeState == picked){
				return true;
			}else{
				return false;
			}
		}

		public function isDropped():Boolean{
			if(nodeState == dropped){
				return true;
			}else{
				return false;
			}
		}

		public function isPopped():Boolean{
			if(nodeState == popped){
				return true;
			}else{
				return false;
			}
		}

		/*function loadData(dat:Array):void{
			tag.dat = dat;
			meta.dat = dat;
			meta.format();
			var path:String = String(dat[0] + dat[1]);
			//img.loadImg(path);
		}

		function loadImg():void{
			var path:String = String(meta.dat[0] + meta.
dat[1]);

			img.loadImg(path);
		}*/

		public function select():void{
			//trace("node delegate select");
			nodeState.select();
		}

		public function deSelect():void{
			nodeState.deSelect();
		}

		public function toggleSelect():void{
			nodeState.toggleSelect();
		}

		public function drag():void{
			//trace("node delegate drag");
			nodeState.drag();
		}

		public function drop():void{
			//trace("node delegate drop");
			nodeState.drop();
		}

		//change to setTarget
		public function setTarget(pid:uint, sid:uint):void{
			//trace("node set Target");
			nodeState.setTarget(xpos[pid][sid], ypos[pid]
[sid]);
		}

		public function rePosition():void{
			//trace("node delegate rePosition");
			nodeState.rePosition();
		}

		public function reSize():void{
			//trace("node delegate reSize");
			nodeState.reSize();
		}

		public function startFocus():void{
			//trace("");
			nodeState.startFocus();
		}

		public function stopFocus():void{
			//trace("");
			nodeState.stopFocus();
		}

		public function startHover():void{
			//trace("");
			nodeState.startHover();
		}

		public function stopHover():void{
			//trace("");
			nodeState.stopHover();
		}

		public function incrTag():void{
			tag.incr();
		}

		public function decrTag():void{
			tag.decr();
		}
	}
}


**************************************************
package metaswarm.build {
    import flash.display.*;
	import flash.events.*;
	import flash.ui.Keyboard;
	import flash.net.*;
	import flash.utils.*;
	import flash.text.*;
	import metaswarm.ui.hoverlabel.LabelMaker;
	import metaswarm.ui.sidebar.SidebarMaker;
	import metaswarm.swarm.patternmaker.PatternMaker;
	import metaswarm.swarm.SwarmLoader;
	import metaswarm.ui.UiLoader;
	import metaswarm.*;

	//import fl.controls.Label;

    public class Builder extends Sprite
    {
		private var _gnurbwurk:Metaswarm;
		private var _uiLoader:UiLoader;

		public function Builder(_gnurbwurk:Metaswarm):void
		{
			this._gnurbwurk = _gnurbwurk;
			this._uiLoader = new UiLoader(_gnurbwurk.ui);

			init();
		}
		private function init():void
		{
			//
		}
		public function load(dat:Array):void
		{
			//load parsed data into objects
			_uiLoader.load(dat);
		}
		public function make():void
		{
			//make views
			_uiLoader.make();
		}
		public function go():void
		{
			//add grandchildren to display
			_uiLoader.go();
		}

	}//end class
}//end pkg
package metaswarm.build {
  import flash.display.*;
  import flash.text.*;
  import flash.events.*
  import flash.net.*;

  public class DataLoader extends Sprite {

	private var _loadProg:TextField;
	private var _preloader:Preloader;

    public function DataLoader(preloader:Preloader)
	{
		this._preloader = preloader;
		this._loadProg = new TextField();

		init();
	}

    private function init():void
	{
	//loadProg.width = stage.stageWidth;
	//loadProg.height = stage.stageHeight;
	addChild(_loadProg);
    }

	public function preload():void {

		var loader:URLLoader = new URLLoader();

		// Instruct the loader to read the file as plain text -
This line is not
		// necessary because the dataFormat is DataFormat.TEXT
by default.
		loader.dataFormat = URLLoaderDataFormat.TEXT;

		// Register an event handler for when the data is fin-
ished downloading
		loader.addEventListener(Event.COMPLETE, loadComplete);

		// Listen for the progress event to check download
progress
	loader.addEventListener(ProgressEvent.PROGRESS, han-
dleProgress);

		// Load the HTML text from the example.html file
		loader.load(new URLRequest("index7.txt"));
	}

	private function handleProgress(event:ProgressEvent):void
	{
	// Calculate the percentage by multiplying the loaded-to-total
	// ratio by 100
	var percent:Number = Math.round(event.bytesLoaded
                                    / event.bytesTotal * 100 );

	_loadProg.text = " Loaded: " + event.bytesLoaded + "\n"
                   + "  Total: " + event.bytesTotal + "\n"
                   + "Percent: " + percent;

		//trace(loadProg.text);
	}


    private function loadComplete(event:Event):void
	{
		var loader:URLLoader = URLLoader(event.target);

		// assign to output. The data property of the URLLoader
is the file contents.
		parseData(loader.data);
	}

	private function parseData(loadedData:String):void
	{
		var parsedData:Array = new Array();
		//var parsedMenu:Array = new Array();

		// Split the string into an array of words using a
space as the delimiter.
		var lines:Array = loadedData.split("\n");
		var words:Array = new Array();

		// Loop through the array and do something with each
word.
		// In this example, just output the values.
		for ( var i:int = 0; i < lines.length; i++ ) {
			words = lines[i].split("\t");
			parsedData.push(words);
			//trace(parsedData[i][1]);
		}
		//gnurbwurk.parsedData = parsedData;
		_preloader.preloadComplete(parsedData);
		//gnurbwurk.main.load(parsedData);
		//gnurbwurk.main.make();
		//gnurbwurk.main.go();
		//gnurbwurk.main.makePatterns();
	}

  }//end class
}//end pkg
package metaswarm.build {
    import flash.display.DisplayObject;
    import flash.display.Graphics;
    import flash.display.Shape;
	import flash.display.Sprite;
	import flash.events.Event;
	import flash.events.KeyboardEvent;
	import flash.events.MouseEvent;
	import flash.ui.Keyboard;
	import flash.net.*;
	import flash.utils.*;

    public class Preloader extends Sprite
```

```
    {
            private var _dataLoader:DataLoader;
            //var main:Main;
            private var _builder:Builder;

        public function Preloader(_builder:Builder):void{
                this._builder = _builder;
                this._dataLoader = new DataLoader(this);
                init();
            }

            //loads external data
            private function init():void{
                _dataLoader.preload();
            }

            public function preloadComplete(parsedData:Array):void{
                //trace("gw.preloadComplete");
                _builder.load(parsedData);
                _builder.make();
                _builder.go();
            }

    }//end class
}//end pkg
package metaswarm.control {
    import flash.display.*;
    import metaswarm.ui.hoverlabel.HoverLabel;
    import metaswarm.swarm.Swarm;
    import metaswarm.ui.Ui;
    import metaswarm.ui.UiState;

    public class Controller extends Sprite{

            //private var _uiState:UiState;
            private var main:Ui;

            public function Controller(main:Ui){
                //_uiState = main.uiState;
                this.main = main;

                init();
            }
            private function init():void{
                //
            }

            private function init():void{
                //
            }

            //delegated meth-
ods************************************************
            public function onEnterFrame():void{
                main.uiState.onEnterFrame();
            }
            public function onRightKey():void{
                main.uiState.onRightKey();
            }
            public function onLeftKey():void{
                main.uiState.onLeftKey();
            }
            public function onUpKey():void{
                main.uiState.onUpKey();
            }
            public function onDownKey():void{
                main.uiState.onDownKey();
            }
            public function onSpaceKey():void{
                main.uiState.onSpaceKey();
            }
            public function onKey1():void{
                trace("Controller -> onKey1");
                main.uiState.onKey1();
            }
            public function onKey2():void{
                trace("Controller -> onKey2");
                main.uiState.onKey2();
            }
            public function onKey3():void{
                main.uiState.onKey3();
            }
            public function onKey4():void{
                main.uiState.onKey4();
            }
            public function onKey5():void{
                //swarm.showUniqueTags();
            }
            public function onCtrlClick(node:Node,
button:Object):void
            {
                main.uiState.onCtrlClick(node, button);
            }
            public function onShftClick(node:Node,
button:Object):void
            {
                main.uiState.onShftClick(node, button);
            }
            public function onAltClick(node:Node,
button:Object):void
            {
                //can't use alt b/c window uses it to focus
toolbar
            }
            public function onCtrlShftClick(node:Node,
button:Object):void
            {
                main.uiState.onCtrlShftClick(node, button);
            }
            public function onMouseClick(node:Node,
button:Object):void
            {
                main.uiState.onMouseClick(node, button);
            }

            //public function onDataBoxClick(node:Node,
button:Object):void{}
            public function onMouseDown(node:Node,
button:Object):void
            {
                main.uiState.onMouseDown(node, button);
            }
            public function onMouseUp(node:Node):void{
                main.uiState.onMouseUp(node);
            }

            //over
            public function onMouseOver(node:Node,
button:Object):void{
                main.uiState.onMouseOver(node, button);
            }
            public function onCtrlOver(node:Node,
button:Object):void{
                main.uiState.onCtrlOver(node, button);
            }
            public function onShftOver(node:Node,
button:Object):void{
                main.uiState.onShftOver(node, button);
            }
            public function onCtrlShftOver(node:Node,
button:Object):void{
                main.uiState.onCtrlShftOver(node, button);
            }

            //out
            public function onMouseOut(node:Node):void{
                main.uiState.onMouseOut(node);
            }
            public function onCtrlOut():void{
                main.uiState.onCtrlOut();
            }
            public function onShftOut():void{
                main.uiState.onShftOut();
            }
            public function onCtrlShftOut():void{
                //
            }


        }//end class
}//end pkg
package metaswarm.control {
    import flash.display.*;
        import flash.events.*;
        import flash.ui.Keyboard;
        import flash.net.*;
        import flash.utils.*;
        import flash.text.*;

        //import fl.controls.Label;

    public class Router extends Sprite
    {
            private var _controller:Controller;
            //var dropped:Subset;
            //var picked:Subset;
```



```
            //var idle:Subset;

            //these should be scoped to methods
            private var _ctrlPressed:Boolean;
            private var _shftPressed:Boolean;
            private var _altPressed:Boolean;
            private var _mousedOver:Boolean;

            private var _focusedNode:Node;
            private var _focusedButton:Object;

        public function Router(controller:Controller):void
            {
                this._controller = controller;

                _ctrlPressed = false;
                _shftPressed = false;
                _altPressed = false;
                _mousedOver = false;

                _focusedNode = new Node();
                _focusedButton = new Object();

                init();
            }
            private function init():void{
                //can't access this.stage until main is added to
stage
                //addEventListener(Event.ADDED_TO_STAGE, onAd-
dedToStage);
            }

            //delegated methods ********************************
**********************
            public function onEnterFrame(event:Event):void{
                _controller.onEnterFrame();
                //cmdTag.x = mouseX;
                //cmdTag.y = mouseY;
            }

            public function onkeyDown(event:KeyboardEvent):void
            {
                if (event.keyCode == Keyboard.RIGHT){
                    _controller.onRightKey();
                }else if (event.keyCode == Keyboard.LEFT){
                    _controller.onLeftKey();
                }else if (event.keyCode == Keyboard.UP){
                    _controller.onUpKey();
                }else if (event.keyCode == Keyboard.DOWN){
                    _controller.onDownKey();
                }else if (event.keyCode == Keyboard.SPACE){
                    _controller.onSpaceKey();
                }else if (event.keyCode == 49){
                    trace("pressed 1");
                    _controller.onKey1();
                }else if (event.keyCode == 50){
                    trace("pressed 2");
                    _controller.onKey2();
                }else if (event.keyCode == 51){
                    trace("pressed 3");
                    _controller.onKey3();
                }else if (event.keyCode == 52){
                    trace("pressed 4");
                    _controller.onKey4();
                }else if (event.keyCode == 53){
                    trace("pressed 5");
                    _controller.onKey5();
                }else if (event.ctrlKey == true || event.shift-
Key == true){

                    _ctrlPressed = event.ctrlKey;
                    _shftPressed = event.shiftKey;

                    if(_mousedOver){
                        //just clear for now
                        _controller.onShftOut();

                        if(!_shftPressed && _ctrlPressed)
{
                            _controller.onCtrlOver(_
focusedNode, _focusedButton);
                        }else if(_shftPressed && !_ctrl-
Pressed){
                            _controller.onShftOver(_
focusedNode, _focusedButton);
                        }else if(_shftPressed && _ctrl-
Pressed){
                            _controller.onCtrlShft-
Over(_focusedNode, _focusedButton);
                        }else{
                            //trace("neither ctrl nor
shift was pressed and yet here we are");
                        }
                    }else{
                        //trace("you pressed an invalid key: " +
event.keyCode);
                    }
                }

                public function onkeyUp(event:KeyboardEvent):void
                {
                    //reset ctrl and shft key states
                    _ctrlPressed = event.ctrlKey;
                    _shftPressed = event.shiftKey;

                    //retest for focus conditions
                    if(_mousedOver){

                        //just clear for now
                        _controller.onShftOut();

                        if(!_shftPressed && _ctrlPressed){
                            _controller.onCtrlOver(_focused-
Node, _focusedButton);
                        }else if(_shftPressed && !_ctrlPressed){
                            _controller.onShftOver(_focused-
Node, _focusedButton);
                        }else if(_shftPressed && _ctrlPressed){
                            _controller.onCtrlShftOver(_fo-
cusedNode, _focusedButton);
                        }else{
                            //trace("neither ctrl nor shift
was pressed and yet here we are");
                            _controller.onMouseOver(_focused-
Node, _focusedButton);
                        }
                    }
                }

                //maybe get rid of this since it competes with mouseUp
                public function onMouseClick(event:MouseEvent):void{
                    //trace(getQualifiedClassName(event.target));
                    var grandparentClass:String =
getQualifiedClassName(event.target.parent.parent);

                    if(grandparentClass == "Node"){
                        if(_ctrlPressed && _shftPressed){
                            _controller.
onCtrlShftClick(event.target.parent.parent, event.target);
                        }else if(_ctrlPressed == true){
                            _controller.onCtrlClick(event.
target.parent.parent, event.target);
                        }else if(_shftPressed == true){
                            _controller.onShftClick(event.
target.parent.parent, event.target);
                        }else{
                            _controller.onMouseClick(event.
target.parent.parent, event.target);
                        }
                    }else{
                        //trace("uh oh, grandparent is not a
Node but a: " + grandparentClass);
                    }
                }

                public function onMouseDown(event:MouseEvent):void
                {
                    //var parentClass:String =
getQualifiedClassName(event.target.parent);
                    var grandparentClass:String =
getQualifiedClassName(event.target.parent.parent);

                    if(!_ctrlPressed && !_shftPressed && !_alt-
Pressed && grandparentClass == "Node"){
                        _controller.onMouseDown(event.target.
parent.parent, event.target);
                    }else{
                        //break if ctrl, shft, or alt is pressed
                        //trace("don't drag");
                    }
                }
```

```
		public function onMouseUp(event:MouseEvent):void
		{
			//var parentClass:String =
getQualifiedClassName(event.target.parent);
			var grandparentClass:String =
getQualifiedClassName(event.target.parent.parent);

			if(!_ctrlPressed && !_shftPressed && !_alt-
Pressed && grandparentClass == "Node"){
				_controller.onMouseUp(event.target.par-
ent.parent);
			}else{
				//break if ctrl, shft, or alt is pressed
				//trace("don't drag");
			}
		}

		public function onMouseOver(event:MouseEvent):void
		{
			var grandparentClass:String =
getQualifiedClassName(event.target.parent.parent);

			if(grandparentClass == "Node"){
				_mousedOver = true;
				_focusedNode = event.target.parent.par-
ent;
				_focusedButton = event.target;
				if(!_ctrlPressed && !_shftPressed){
					_controller.onMouseOver(event.
target.parent.parent, event.target);
				}else if(_ctrlPressed && !_shftPressed){
					_controller.onCtrlOver(event.tar-
get.parent.parent, event.target);
				}else if(!_ctrlPressed && _shftPressed){
					_controller.onShftOver(event.tar-
get.parent.parent, event.target);
				}else if(_ctrlPressed && _shftPressed){
					_controller.onCtrlShftOver(event.
target.parent.parent, event.target);
				}
			}else{
				//trace("uh oh, grandparent is not a
Node but a: " + grandparentClass);
			}
		}

		public function onMouseOut(event:MouseEvent):void
		{
			var grandparentClass:String =
getQualifiedClassName(event.target.parent.parent);

			if(grandparentClass == "Node"){
				_mousedOver = false;
				//just clear for now
				_controller.onShftOut();
				_controller.onMouseOut(event.target.par-
ent.parent);
			}else{
				//trace("uh oh, grandparent is not a
Node but a: " + grandparentClass);
			}
		}

	}//end class
}//end pkg
package metaswarm.effects
{
	// createcomps_effects/myEffects/Rotation.as
	import mx.effects.TweenEffect;
	import mx.effects.EffectInstance;
	import mx.effects.IEffectInstance;

	public class ZoomIn extends TweenEffect
	{
		// Define parameters for the effect.
		public var angleFrom:Number = 0;
		public var angleTo:Number = 360;

		// Define constructor with optional argument.
		public function Rotation(targetObj:* = null) {
			super(targetObj);
			//instanceClass = RotationInstance;
		}

		// Override getAffectedProperties() method to return
"rotation".
		override public function getAffectedProperties():Array
{
			return ["rotation"];
		}

		// Override initInstance() method.
		override protected function initInstance(inst:IEffectIn
stance):void {
			super.initInstance(inst);
			//RotationInstance(inst).angleFrom = angleFrom;
			//RotationInstance(inst).angleTo = angleTo;
		}
	}
}
package metaswarm.node	{
  import flash.display.*;
  import flash.text.*;
  import flash.events.*
  import flash.net.*;

  public class NodeLoader extends Sprite {

	private var node:Node;

    public function NodeLoader(node:Node)
	{
		this.node = node;
		//this.loadProg = new TextField();

		init();
    }
	private function init():void{
		//
	}

	public function load(dat:Array):void{
		//trace("nodeLoader load")
		node.dat = dat;
		node.dot.load();
		node.tag.load(dat);
		node.icon.load();
		//node.tag.load(node.dat);
		node.dataBox.load(dat);
		//node.thumb.load(node.dat[0] + node.dat[1]);
	}
	public function make():void{
		//trace("nodeLoader make")
		node.dot.make();
		node.tag.make();
		node.icon.make();
		node.dataBox.make();
		//node.thumb.make();
	}
	public function go():void{
		//nodes are initially idle
		//node.nodeState = node.idle;
		//node.addChild(node.thumb);
		node.addChild(node.dot);
		node.addChild(node.icon);
		node.addChild(node.tag);
		node.addChild(node.dataBox);
		//node.addChild(node.thumb); //just added this so I can
remove bitmap loader from display list
	}

  }//end class
}//end pkg
package metaswarm.node	{

	//fsm interface
	public interface NodeState{ //<---------------------- public
breaks program

		//internal interface methods
		//function filter():void;
		function select():void;
		function deSelect():void;
		function toggleSelect():void;
		function drag():void;
		function drop():void;
		//function dropThumb():void;
		function popout():void;
		function popin():void;
		function close():void;
		function setTarget(tgx:Number,
```



```
tgy:Number):void;
		function rePosition():void;
		function reSize():void;
		function startFocus():void;
		function stopFocus():void;
		function startHover():void;
		function stopHover():void;
		//function incrTag():void;
		//function decrTag():void;
	}
}
package metaswarm.node.components	{
	import flash.display.DisplayObject;
	import flash.display.Graphics;
	import flash.display.Shape;
	import flash.display.Sprite;

	public class Particle extends Sprite{

		//public bc used in subclass plot functions
		//public static const DEF_CLR:Color	= #000000;
		//public static const FOC_CLR:Color	= #FF0000;
		private static const EASE:Number	= .25;
		private static const SNAP:Number	= 1;

		//Target Coordinates
		internal var tgx:Number;
		internal var tgy:Number;

		//Previous Coordinates
		public var px:Number;
		public var py:Number;

		//distance to target
		private var dx:Number;
		private var dy:Number;

		//velocity
		private var vx:Number;
		private var vy:Number;

		//atTarget:Boolean; unnec. used expression instead

		//hidden:Boolean; use inherited prop visible instead
		//infocus:Boolean; not needed because mouse events
		//are object oriented. that is they test for focus
		//internally

		//x
		//y
		//visible : Boolean

		//still unclear about constructors, see pg 180 of FofEd
anim.
		public function Particle(tgx:Number=0,

tgy:Number=0,

dx:Number=0,

dy:Number=0,

vx:Number=0,

vy:Number=0):void
		{
			this.x = 0;
			this.y = 0;
			this.tgx = tgx;
			this.tgy = tgy;
			this.dx = dx;
			this.dy = dy;
			this.vx = vx;
			this.vy = vy;
			this.px = 0;
			this.py = 0;
			//init();
		}

		/*public function init():void {
			//nothing for now
		}*/

		public function isVisible():Boolean {
			return this.visible;
		}

		//use move(x,y) instead
		/*public function position(xNew:Number,
yNew:Number):void
		{
			this.x = xNew;
			this.y = yNew;
		}*/

		public function goToTarget():void {
			//trace("x,y before: " + dx + " " + dy);
			//trace("tgs before: " + tgx + " " + tgy);
			dx = tgx - x;
			dy = tgy - y;
			if(Math.sqrt(dx*dx + dy*dy) < SNAP){
				x = tgx;
				y = tgy;
				//removeEventListener(Event.Enter_FRAME,
onEnterFrame);
			}else{
				vx = dx * EASE;
				vy = dy * EASE;
				x += vx;
				y += vy;
			}
			//trace("after: " + dx + " " + dy);
			//trace("tgs after: " + tgx + " " + tgy);
		}

		public function snapToTarget():void{ //<-------------
wtf
			x = tgx;
			y = tgy;
		}

		public function setTarget(a:Number, b:Number):void
{//<-------------wtf
			tgx = a;
			tgy = b;
		}

		public function setPrev(a:Number, b:Number):void {//<--
-----------wtf
			px = a;
			py = b;
		}

		//getFocus():Interactive Object
		//hitTestPoint(x:Number, y:Number, shapeFlag:Boolean =
false):Boolean
		//startDrag()
		//stopDrag()
	}
}
package metaswarm.node.components.databox	{
	import flash.display.*;
	import flash.text.*;

	public class ButtonDisplayState extends Sprite {

		private var typeOfButton:String;
		private var bgColor:uint;
		private var lineColor:uint;
		private var size:uint;

		public function ButtonDisplayState(typeOfButton:String,
lineColor:uint, bgColor:uint, size:uint) {
			this.typeOfButton = typeOfButton;
			this.bgColor = bgColor;
			this.lineColor = lineColor;
			this.size = size;
			draw();
		}

		private function draw():void {
			graphics.beginFill(bgColor);
			graphics.drawRect(0, 0, size, size);
			graphics.endFill();
			graphics.lineStyle(0, lineColor);
			//graphics.beginFill(lineColor);

			//. . . . .
			//. a . b .
			//. . . . .
			//. c . d .
			//. . . . .
```

```
                var ax:Number = size*(1/4);
                var ay:Number = size*(1/4);
                var bx:Number = size*(3/4);
                var by:Number = size*(1/4);
                var cx:Number = size*(1/4);
                var cy:Number = size*(3/4);
                var dx:Number = size*(3/4);
                var dy:Number = size*(3/4);


                switch (typeOfButton) {
                        case "close"://x
                                //trace(0);
                                graphics.moveTo(ax, ay);
                                graphics.lineTo(dx, dy);
                                graphics.moveTo(bx, by);
                                graphics.lineTo(cx, cy);
                                break;
                        case "popout"://>
                                //trace(1);
                                graphics.moveTo(ax, ay);
                                graphics.lineTo(dx, dy);
                                graphics.moveTo(bx, by);
                                graphics.lineTo(dx, dy);
                                graphics.lineTo(cx, cy);
                                break;
                        case "popin"://<
                                //trace(2);
                                graphics.moveTo(ax, ay);
                                graphics.lineTo(dx, dy);
                                graphics.moveTo(bx, by);
                                graphics.lineTo(ax, ay);
                                graphics.lineTo(cx, cy);
                                break;
                        default:
                                trace("Not a valid type of but-
ton");
                }

                //graphics.endFill();

                /*var label:TextField = new TextField();
                var fmt:TextFormat = new TextFormat(txt, size/2,
0xCCCCCC);

                //label.x = dbx.closeButton.x;
                //label.y = dbx.closeButton.y;
                label.autoSize = TextFieldAutoSize.LEFT;
                //label.width = size;
                //label.height = size;
                label.setTextFormat(fmt);
                label.text = txt;
                addChild(label);
                label.mouseEnabled = false;*/
        }
}
package metaswarm.node.components.databox   {
        import flash.display.*;
        import flash.text.*;

        public class CustomSimpleButton extends SimpleButton {
                /*private var upColor:uint   = 0xFFCC00;
                private var overColor:uint = 0xCCFF00;
                private var downColor:uint = 0x00CCFF;
                private var black:uint = 0x231F20;*/

                private static const BLACK:uint = 0x231F20;
                private static const WHITE:uint = 0xFFFFFF;
                private static const BRIGHTRED:uint = 0xFF3333;
                private static const BRIGHTBLUE:uint = 0x66CDFF;

                public var size:uint;
                public var type:String;

                public function CustomSimpleButton(type:String) {
                        this.size = 15;
                        this.type = type;
                        downState      = new ButtonDisplayState(type,
BRIGHTBLUE, BLACK, size);
                        overState      = new ButtonDisplayState(type,
BRIGHTRED, BLACK, size);
                        upState        = new ButtonDisplayState(type,
WHITE, BLACK, size);
                        hitTestState   = new ButtonDisplayState(type,
WHITE, BLACK, size);
                        hitTestState.x = -(size / 4);
                        hitTestState.y = hitTestState.x;
                        useHandCursor  = true;
                }

                public function make():void {
                        /*downState      = new
ButtonDisplayState(downColor, size, txt);
                        overState      = new
ButtonDisplayState(overColor, size, txt);
                        upState        = new ButtonDisplayState(upColor,
size, txt);
                        hitTestState   = new ButtonDisplayState(upColor,
size, txt);
                        hitTestState.x = -(size / 4);
                        hitTestState.y = hitTestState.x;
                        useHandCursor  = true;*/
                }
        }
}


package metaswarm.node.components.databox   {
        import flash.display.*;
        import flash.text.*;
        import flash.events.*;
        import flash.utils.*;
        import metaswarm.node.components.Particle;

        public class DataBox extends Particle{

                //need an idea for select and filter functions
                //e.g. select matching text with id eq to this.
                //this is id of array element no textBox
                //var id:uint;

                public var dat:Array;

                public var t:TextField = new TextField();

                public var closeButton:CustomSimpleButton;
                public var popoutButton:CustomSimpleButton;
                public var popinButton:CustomSimpleButton;
                //var loadExtButton:CustomSimpleButton;

                public var dataBoxMaker:DataBoxMaker;

                public function DataBox():void{
                        dataBoxMaker = new DataBoxMaker(this);
                        init();
                }
                public function init():void {
                        visible = false;
                        focusRect = true;
                        tabEnabled = true;

                        //addEventListener(MouseEvent.CLICK, onMouse-
Click);
                }
                public function load(dat:Array):void{
                        this.dat = dat;
                }
                public function make():void {
                        dataBoxMaker.make();
                }

                private function onMouseClick(event:MouseEvent):void{
                        //trace(getQualifiedClassName(event.target));

                        /*if(getQualifiedClassName(event.target) ==
"::CustomSimpleButton"){
                                switch (event.target.type) {
                                        case CLOSE:
                                                trace("close");
                                                break;
                                        case POPOUT:
                                                trace("popout");
                                                parent.thumb.load(dat[0]
+ dat[1]);

                                                parent.thumb.make();
                                                parent.thumb.x = this.
width;

                                                parent.thumb.y = this.y;
                                                parent.addChild(parent.
thumb);

                                                break;
                                        case POPIN:
                                                trace("popin");
                                                break;
                                        default:
                                                trace("Not a valid type
of button");
                                }
                        }*/
                }

        }//end class
}//end pkg
package metaswarm.node.components.databox   {
        import flash.display.*;
        import flash.text.*;

        class DataBoxButton extends SimpleButton {
                /*private var upColor:uint   = 0xFFCC00;
                private var overColor:uint = 0xCCFF00;
                private var downColor:uint = 0x00CCFF;
                private var black:uint = 0x231F20;*/

                private static const BLACK:uint = 0x231F20;
                private static const WHITE:uint = 0xFFFFFF;
                private static const BRIGHTRED:uint = 0xFF3333;
                private static const BRIGHTBLUE:uint = 0x66CDFF;

                var size:uint;
                var type:String;

                public function DataBoxButton(type:String) {
                        this.size = 15;
                        this.type = type;
                        downState      = new DataBoxButtonState(type,
BRIGHTBLUE, BLACK, size);
                        overState      = new DataBoxButtonState(type,
BRIGHTRED, BLACK, size);
                        upState        = new DataBoxButtonState(type,
WHITE, BLACK, size);
                        hitTestState   = new DataBoxButtonState(type,
WHITE, BLACK, size);
                        hitTestState.x = -(size / 4);
                        hitTestState.y = hitTestState.x;
                        useHandCursor  = true;
                }

                public function make() {
                        /*downState      = new
ButtonDisplayState(downColor, size, txt);
                        overState      = new
ButtonDisplayState(overColor, size, txt);
                        upState        = new ButtonDisplayState(upColor,
size, txt);
                        hitTestState   = new ButtonDisplayState(upColor,
size, txt);
                        hitTestState.x = -(size / 4);
                        hitTestState.y = hitTestState.x;
                        useHandCursor  = true;*/
                }
        }
}


package metaswarm.node.components.databox   {
        import flash.display.*;
        import flash.text.*;

        class DataBoxButtonState extends Sprite {
                private var typeOfButton:String;
                private var bgColor:uint;
                private var lineColor:uint;
                private var size:uint;

                public function DataBoxButtonState(typeOfButton:String,
lineColor:uint, bgColor:uint, size:uint) {
                        this.typeOfButton = typeOfButton;
                        this.bgColor = bgColor;
                        this.lineColor = lineColor;
                        this.size = size;
                        draw();
                }

                private function draw():void {
                        graphics.beginFill(bgColor);
                        graphics.drawRect(0, 0, size, size);
                        graphics.endFill();
                        graphics.lineStyle(0, lineColor);
                        //graphics.beginFill(lineColor);

                        //. . . . .
                        //. a . b .
                        //. . . . .
                        //. c . d .
                        //. . . . .
                        var ax:Number = size*(1/4);
                        var ay:Number = size*(1/4);
                        var bx:Number = size*(3/4);
                        var by:Number = size*(1/4);
                        var cx:Number = size*(1/4);
                        var cy:Number = size*(3/4);
                        var dx:Number = size*(3/4);
                        var dy:Number = size*(3/4);


                        switch (typeOfButton) {
                                case "close"://x
                                        //trace(0);
                                        graphics.moveTo(ax, ay);
                                        graphics.lineTo(dx, dy);
                                        graphics.moveTo(bx, by);
                                        graphics.lineTo(cx, cy);
                                        break;
                                case "popout"://>
                                        //trace(1);
                                        graphics.moveTo(ax, ay);
                                        graphics.lineTo(dx, dy);
                                        graphics.moveTo(bx, by);
                                        graphics.lineTo(dx, dy);
                                        graphics.lineTo(cx, cy);
                                        break;
                                case "popin"://<
                                        //trace(2);
                                        graphics.moveTo(ax, ay);
                                        graphics.lineTo(dx, dy);
                                        graphics.moveTo(bx, by);
                                        graphics.lineTo(ax, ay);
                                        graphics.lineTo(cx, cy);
                                        break;
                                default:
                                        trace("Not a valid type of but-
ton");
                        }

                        //graphics.endFill();

                        /*var label:TextField = new TextField();
                        var fmt:TextFormat = new TextFormat(txt, size/2,
0xCCCCCC);
                        //label.x = dbx.closeButton.x;
                        //label.y = dbx.closeButton.y;
                        label.autoSize = TextFieldAutoSize.LEFT;
                        //label.width = size;
                        //label.height = size;
                        label.setTextFormat(fmt);
                        label.text = txt;
                        addChild(label);
                        label.mouseEnabled = false;*/
                }
        }
}
package metaswarm.node.components.databox   {
        import flash.display.*;
        import flash.text.*;

        public class DataBoxMaker extends Sprite{

                private static const CLOSE:String = "closeDataBox";
                private static const POPOUT:String = "popout";
                private static const POPIN:String = "popin";

                public var dbx:DataBox;

                public function DataBoxMaker(dataBox:DataBox):void{
                        this.dbx = dataBox;
                        init();
                }
                public function init():void {
                        //
                }
                public function make():void {
                        makeTextField();
                        //makeButtons();
                }
```

```
		private function makeTextField():void {
			/*h1 = new TextField();
			h2 = new TextField();
			h3 = new TextField();
			l = new TextField();*/
			//var t:TextField = new TextField();

			var h1Fmt:TextFormat = new TextFormat("Technic",
18, 0x333333);
			//var h2Fmt:TextFormat = new TextFormat();
			var h3Fmt:TextFormat = new TextFormat("Technic",
14, 0x06c);
			var lFmt:TextFormat = new TextFormat("Technic",
8, 0x333333);

			/*h1Fmt.TextFormat(font:String = null,
					size:Object = null,
					color:Object = null,
					bold:Object = null,
					italic:Object = null,
					underline:Object = null,
					url:String = null,
					target:String = null,
					align:String = null,
					leftMargin:Object = null,
					rightMargin:Object = null,
					indent:Object = null,
					leading:Object = null);*/

			//alwaysShowSelection : Boolean


			//dbx.t.autoSize = TextFieldAutoSize.LEFT;
			dbx.t.height = 100;
			dbx.t.width = 100;
			//dbx.t.background = true;
			//dbx.t.backgroundColor = 0x231F20;
			dbx.t.alpha = 50;
			//dbx.t.border = true;
			//dbx.t.borderColor = 0x000000;
			//t.height =
			//t.width =
			dbx.t.mouseEnabled  = false;
			//t.mouseWheelEnabled = true;
			//t.multiline = true;
			dbx.t.wordWrap = true;

			//t.numLines [read-only]
			//t.length [read-only]
			//t.scrollH : int
			//t.scrollV : int
			//t.selectable = true;
			//t.tabEnabled = true;
			//t.thickness : Number
			/*//t.width =
			//t.height =
			t.background = true;
			t.backgroundColor = 0xCCCCCC; //light gray
			t.border = true;
			t.borderColor = 0x333333; //dark gray
			t.autoSize = TextFieldAutoSize.LEFT;
			//addChild(this);*/
			dbx.t.text = "";

			var c0:uint = 0;
			var c1:uint = c0 + dbx.dat[0].length + dbx.
dat[1].length +1;
			var c2:uint = c1 + dbx.dat[2].length + 1;
			var c3:uint = c2 + 10;

			dbx.t.appendText(dbx.dat[0] + dbx.dat[1] +
'\n');
			dbx.t.appendText(dbx.dat[2] + '\n');
			dbx.t.appendText("__________" + '\n');
			//t.text += dat[0] + dat[1] + '\n';
			//t.text += dat[2] + '\n';
			//t.text += "__________" + '\n';

			for(var i:uint=3; i<dbx.dat.length; i++){
				dbx.t.appendText(dbx.dat[i] + ", ");
				//t.text += dat[i] + ", ";
			}

			dbx.t.setTextFormat(h1Fmt, c0, c1);
			dbx.t.setTextFormat(h3Fmt, c1, c2);
			dbx.t.setTextFormat(h3Fmt, c2, c3);
			dbx.t.setTextFormat(lFmt, c3, dbx.t.text.
length);

			dbx.t.selectable = false;
			dbx.t.mouseEnabled = false;
			//this.id = 0;
			dbx.addChild(dbx.t);
			//initTag();
		}

		private function makeButtons():void {

			dbx.closeButton = new CustomSimpleButton(CLOSE);
			dbx.popoutButton = new
CustomSimpleButton(POPOUT);
			dbx.popinButton = new CustomSimpleButton(POPIN);
			//dbx.loadExtButton:CustomSimpleButton = new
CustomSimpleButton("e");

			//var myButton:Button = new Button();
			//dbx.close.label = "x";
			//dbx.close.emphasized = true;
			//dbx.close.width = 15;
			//close.move(20, 20);

			/*dbx.closeButton.make();
			dbx.popoutButton.make();
			dbx.popinButton.make();
			dbx.loadExtButton.make();*/

			dbx.closeButton.x = dbx.t.width-dbx.closeButton.
width;
			//dbx.closeButton.y = dbx.t.height;

			dbx.popoutButton.x = dbx.t.width-dbx.popoutBut-
ton.width;
			dbx.popoutButton.y = dbx.t.height-15;

			dbx.popinButton.x = dbx.t.width-(15 * 2);
			dbx.popinButton.y = dbx.t.height-15;

			//dbx.loadExtButton.x = dbx.t.width-(15 * 3);
			//dbx.loadExtButton.y = dbx.t.height-15;


			dbx.addChild(dbx.closeButton);
			//dbx.loadExtButton.y = dbx.t.height-15;


			dbx.addChild(dbx.closeButton);
			dbx.addChild(dbx.popoutButton);
			dbx.addChild(dbx.popinButton);
			//dbx.addChild(dbx.loadExtButton);

		}
	}
}
package metaswarm.node.components.dot    {
	import flash.display.DisplayObject;
    import flash.display.Graphics;
    import flash.display.Shape;
	import flash.display.Sprite;
	import metaswarm.node.components.Particle;

	public class Dot extends Particle{

		public var dotMaker:DotMaker;
		public var dotButton:DotButton;

		public function Dot():void{
			dotMaker = new DotMaker(this);
			init();
		}
		public function init():void {
			//visible = true;
			//focusRect = true;
			//tabEnabled = true;
		}

		public function load():void{
			//
		}
		//might dep on load later, so leave here!
		public function make():void{
			dotMaker.make();
		}

	}
}
```



```
package metaswarm.node.components.dot    {
	import flash.display.*;

	public class DotButton extends SimpleButton{

		private static const BLACK:uint = 0x231F20;
		private static const WHITE:uint = 0xFFFFFF;
		private static const BRIGHTRED:uint = 0xFF3333;
		private static const BRIGHTBLUE:uint = 0x66CDFF;

		public var size:uint;
		public var type:String;

		public function DotButton() {
			this.size = 3;
			this.type = "dot";
			downState      = new DotButtonState(BRIGHTBLUE,
WHITE, size);
			overState      = new DotButtonState(BRIGHTRED,
WHITE, size);
			upState        = new DotButtonState(BLACK,
WHITE, size);
			hitTestState   = new DotButtonState(BLACK,
WHITE, size); //change to overstate
			hitTestState.x = -(size / 4);
			hitTestState.y = hitTestState.x;
			useHandCursor  = true;
		}

		public function forceOver():void{
			upState = overState;
		}

		public function restoreOver():void{
			upState = hitTestState;
		}

	}
}
package metaswarm.node.components.dot    {
	import flash.display.*;
	import flash.text.*;

	//should this extend shape instead????
	public class DotButtonState extends Sprite {
		//private var typeOfButton:uint;
		private var bgColor:uint;
		private var lineColor:uint;
		private var size:uint;

		public function DotButtonState(lineColor:uint,
bgColor:uint, size:uint) {
			//this.typeOfButton = typeOfButton;
			this.bgColor = bgColor;
			this.lineColor = lineColor;
			this.size = size;
			draw();
		}

		private function draw():void {
			//trace("draw the dot button state");
			graphics.beginFill(bgColor);
			graphics.drawRect(-size, -size, size*2, size*2);
			graphics.endFill();

			graphics.lineStyle(0, lineColor);
			graphics.moveTo(x, y-size);
			graphics.lineTo(x, y+size);
			graphics.moveTo(x-size, y);
			graphics.lineTo(x+size, y);
		}
	}
}
package metaswarm.node.components.dot    {
	import flash.display.DisplayObject;
    import flash.display.Graphics;
    import flash.display.Shape;
	import flash.display.Sprite;

	public class DotMaker extends Sprite{

		private var _dot:Dot;

		public function DotMaker(dot:Dot):void{
			_dot = dot;
			init();
		}
		public function init():void{
			//
		}
		public function make():void{
			_dot.dotButton = new DotButton();
			_dot.addChild(_dot.dotButton);
		}
	}
}
package metaswarm.node.components.icon   {
	import flash.display.DisplayObject;
    import flash.display.Graphics;
    import flash.display.Shape;
	import flash.display.Sprite;
	import metaswarm.node.components.Particle;


	public class Icon extends Particle{

		private static const Z_EASE:Number    = 1;
		private static const Z_SNAP:Number    = 1;

		//Target zoom
		private var tgw:Number;

		//radial distance to zoom target
		private var dw:Number;

		//zoom velocity
		private var vw:Number;

		//should be a customSimpleButton not a maker
		public var iconMaker:IconMaker;
		public var iconButton:IconButton;

		public function Icon(tgw:Number=0,
                                                dw:Number=0,
                                                vw:Number=0):void
		{
			this.x = 0;
			this.y = 0;
			this.tgw = tgw;
			this.dw = dw;
			this.vw = vw;

			iconMaker = new IconMaker(this);

			init();
		}
		public function init():void {
			visible = false;
			focusRect = true;
			tabEnabled = true;
			doubleClickEnabled = true;
			//mouseEnabled = false;
		}

		public function load():void{
			//
		}
		//might depend on load later, so keep here
		//rather than put in init
		public function make():void{
			iconMaker.make();
		}

		public function scaleToTarget():void {
			dw = tgw - this.width;
			if(Math.abs(dw - Z_SNAP)<1){
				this.width = tgw;
				this.height = tgw;
				//removeEventListener(Event.Enter_FRAME,
onEnterFrame);
			}else{
				vw = dw * Z_EASE;
				this.width += vw;
				this.height += vw;
			}
		}

		public function setTargetWidth(a:Number):void {
			tgw = a;
		}
```

```
	}
}

package metaswarm.node.components.icon   {
	import flash.display.*;

	public class IconButton extends SimpleButton{

		private static const BLACK:uint = 0x231F20;
		private static const WHITE:uint = 0xFFFFFF;
		private static const BRIGHTRED:uint = 0xFF3333;
		private static const BRIGHTBLUE:uint = 0x66CDFF;

		public var size:uint;
		public var type:String;

		public function IconButton() {
			this.size = 3;
			this.type = "icon";
			downState      = new IconButtonState(BRIGHTBLUE,
BLACK, size);
			overState      = new IconButtonState(BRIGHTRED,
BLACK, size);
			upState        = new IconButtonState(WHITE,
BLACK, size);
			hitTestState   = new IconButtonState(WHITE,
BLACK, size);
			hitTestState.x = -(size / 4);
			hitTestState.y = hitTestState.x;
			useHandCursor  = true;
		}

		public function forceOver():void{
			upState = overState;
		}

		public function restoreOver():void{
			upState = hitTestState;
		}

		/*//should be draw()
		public function make():void {
			icon.graphics.beginFill(0x0000ff);
			icon.graphics.lineStyle(0);
			icon.graphics.drawRect(0, 0, 10, 10);
			icon.graphics.endFill();
		}*/
	}
}
package metaswarm.node.components.icon   {
	import flash.display.*;
	import flash.text.*;

	//should this extend shape instead????
	public class IconButtonState extends Sprite {
		//private var typeOfButton:uint;
		private var bgColor:uint;
		private var lineColor:uint;
		private var size:uint;

		public function IconButtonState(lineColor:uint,
bgColor:uint, size:uint) {
			//this.typeOfButton = typeOfButton;
			this.bgColor = bgColor;
			this.lineColor = lineColor;
			this.size = size;
			draw();
		}

		private function draw():void {
			//trace("draw the icon button state");
			graphics.beginFill(bgColor);
			graphics.lineStyle(0, lineColor);
			//graphics.drawRect(0, 0, size, size);
			graphics.drawRect(-size, -size, size*2, size*2);
			graphics.endFill();
		}
	}
}
package metaswarm.node.components.icon {
	import flash.display.DisplayObject;
    import flash.display.Graphics;
    import flash.display.Shape;
	import flash.display.Sprite;

	public class IconMaker extends Sprite{
		public var icon:Icon;

		public function IconMaker(icon:Icon):void{
			this.icon = icon;
			init();
		}
		public function init():void {
			//
		}
		public function make():void {
			//trace("make the icon button");
			icon.iconButton = new IconButton();
			icon.addChild(icon.iconButton);
		}
	}
}
package metaswarm.node.components.tag   {
	import flash.display.*;
	import flash.text.*;
	import metaswarm.node.components.Particle;

	public class Tag extends Particle{

		//add formatting constants here
		public var tagMaker:TagMaker;
		public var tagButton:TagButton;
		//zoom velocity
		//private var val:String;
		public var t:TextField;
		//var fmt:TextFormat;
		public var uniqArr:Array = new Array();
		public var isUniq:Boolean = new Boolean();

		//need an idea for select and filter functions
		//e.g. select matching text with id eq to this.
		//this is id of array element no textBox
		public var id:uint;

		//cnt not nec, can just use array length
		//var cnt:uint;
		public var dat:Array;
		//var dat:Array = ["cherry", "orange", "soda", "pop"];

		public function Tag():void
		{
			t = new TextField();
			tagMaker = new TagMaker(this);
			//fmt = new TextFormat( );
			init();
		}
		public function init():void {
			visible = false;
			//focusRect = false;
			//tabEnabled = false;
			//tag.mouseEnabled = false;
		}

		public function load(dat:Array):void{
			this.dat = dat;
		}
		public function make():void{
			tagMaker.make();
		}

		public function incr():void{
			if(id < dat.length-1){
				id++;
			}else{
				id=0;
			}
			t.text = dat[id];
			isUniq = uniqArr[id];
			//t.text = String(this.tgx) + ", " +
String(this.tgy);
		}

		public function decr():void{
			if(id > 0){
				id--;
			}else{
				id=dat.length-1;
			}
			t.text = dat[id];
			isUniq = uniqArr[id];
			//t.text = String(this.tgx) + ", " +
String(this.tgy);
		}

		/*public function getDatAt(id):void{
			return dat[id];
		}*/

	}//end class
}//end pkg
package metaswarm.node.components.tag   {
	import flash.display.*;

	public class TagButton extends SimpleButton{

		private static const BLACK:uint = 0x231F20;
		private static const WHITE:uint = 0xFFFFFF;
		private static const BRIGHTRED:uint = 0xFF3333;
		private static const BRIGHTBLUE:uint = 0x66CDFF;

		public var size:uint;
		public var type:String;

		public function TagButton() {
			this.size = 5;
			this.type = "tag";
			downState      = new TagButtonState(BRIGHTBLUE,
BLACK, size);
			overState      = new TagButtonState(BRIGHTRED,
BLACK, size);
			upState        = new TagButtonState(WHITE,
BLACK, size);
			hitTestState   = new TagButtonState(WHITE,
BLACK, size);
			hitTestState.x = -(size / 4);
			hitTestState.y = hitTestState.x;
			useHandCursor  = true;
		}

		public function forceOver():void{
			upState = overState;
		}

		public function restoreOver():void{
			upState = hitTestState;
		}

		/*//should be draw()
		public function make():void {
			icon.graphics.beginFill(0x0000ff);
			icon.graphics.lineStyle(0);
			icon.graphics.drawRect(0, 0, 10, 10);
			icon.graphics.endFill();
		}*/
	}
}
package metaswarm.node.components.tag   {
	import flash.display.*;
	import flash.text.*;

	//should this extend shape instead????
	public class TagButtonState extends Sprite {
		//private var typeOfButton:uint;
		private var bgColor:uint;
		private var lineColor:uint;
		private var size:uint;

		public function TagButtonState(lineColor:uint,
bgColor:uint, size:uint) {
			//this.typeOfButton = typeOfButton;
			this.bgColor = bgColor;
			this.lineColor = lineColor;
			this.size = size;
			//draw();
		}

		//need ref to tag, can't do this yet
		/*private function draw():void {
			//tag.tagButton = new TagButton();
			//tag.addChild(tag.tagButton);

			var fmt:TextFormat = new TextFormat( );

			fmt.font = "Arial";
			fmt.size = size;
			//tag.t.embedFonts = true;
			//tag.t.antiAliasType = AntiAliasType.ADVANCED;
			tag.t.text = "";
			tag.t.setTextFormat(fmt);
			tag.t.selectable = false;
			tag.t.mouseEnabled = false;
			tag.id = 0;

			tag.t.x += 5;
			tag.t.y -= 25;
			//t.width =
			//t.height =
			tag.t.background = true
			tag.t.backgroundColor = bgColor; //light gray
			//tag.t.border = true;
			tag.t.borderColor = lineColor; //dark gray
			tag.t.autoSize = TextFieldAutoSize.LEFT;
			//addChild(this);

			tag.t.text = tag.dat[tag.id];

			tag.addChild(tag.t);
		}*/

	}
}
package metaswarm.node.components.tag   {
	import flash.display.*;
	import flash.text.*;

	public class TagMaker extends Sprite{

		private var tag:Tag;

		public function TagMaker(tag:Tag):void{
			this.tag = tag;
			init();
		}
		public function init():void {
			//
		}

		/*public function make():void {
			//trace("make the tag button");
			tag.tagButton = new TagButton();
			tag.addChild(tag.tagButton);
		}*/

		public function make():void {
			//tag.tagButton = new TagButton();
			//tag.addChild(tag.tagButton);

			var fmt:TextFormat = new TextFormat( );

			fmt.font = "Arial";
			fmt.size = 5;
			//tag.t.embedFonts = true;
			//tag.t.antiAliasType = AntiAliasType.ADVANCED;
			tag.t.text = "";
			tag.t.setTextFormat(fmt);
			tag.t.selectable = false;
			tag.t.mouseEnabled = false;
			tag.id = 0;

			tag.t.x += 5;
			tag.t.y -= 25;
			//t.width =
			//t.height =
			tag.t.background = true
			tag.t.backgroundColor = 0xCCCCCC; //light gray
			tag.t.border = true;
			tag.t.borderColor = 0x333333; //dark gray
			tag.t.autoSize = TextFieldAutoSize.LEFT;
			//addChild(this);

			tag.t.text = tag.dat[tag.id];

			tag.addChild(tag.t);
		}
	}
}
package metaswarm.node.components.thumb   {
  import flash.display.*;
  import flash.text.*;
  import flash.events.*
  import flash.net.*;
  import flash.geom.Matrix;
```

```
  public class BitmapLoadr extends Sprite {

        public var loadProg:TextField;
        public var thumb:Thumb;

        private var loader:Loader; // The bitmap loader
        //var parsedData:Array;
    //var output:String;
        //private static const PATH:String = "../index.txt";

    public function BitmapLoadr(thumb:Thumb) {
                this.thumb = thumb;
                init();
    }
    private function init():void {
                loader = new Loader();
                loadProg = new TextField();
        //loadProg.width = stage.stageWidth;
        //loadProg.height = stage.stageHeight;
        thumb.addChild(loadProg);
    }
        public function loadBmp(path:String):void {
                // Listen for the progress event to check download
progress
        loader.contentLoaderInfo.addEventListener(ProgressEvent.PROG-
RESS, handleProgress);
                // listen for load complete
                loader.contentLoaderInfo.addEventListener(Event.COM-
PLETE, onComplete);
                // Register to be notified when the bitmap has been ini-
tialized
                loader.contentLoaderInfo.addEventListener(Event.INIT,
onInit);

                //loader.load(new URLRequest(path));
                var request:URLRequest = new URLRequest(path);
                loader.load(request);
        }

        // Triggered when the bitmap has been loaded and initialized
        private function onInit(event:Event):void {
                //trace("onInit");
                var loadedBitmap:Bitmap = Bitmap(loader.content);

                var loadedBitmapData:BitmapData = loadedBitmap.bitmap-
Data;

                // Set the amount by which to scale the bitmap
                var scaleFactor:Number = .5;

                /*//specify the height here for now <<<<<<<<<**********
*********************************
                var newHeight:int = 100;

                // Calculate the new dimensions of the scaled bitmap
                scaleFactor = newHeight/loadedBitmapData.height;
                var newWidth:int = loadedBitmapData.width * scaleFac-
tor;*/

                //specify the height here for now <<<<<<<<<***********
*******************************
                var newWidth:int = 200;

                // Calculate the new dimensions of the scaled bitmap
                scaleFactor = newWidth/loadedBitmapData.width;
                var newHeight:int = loadedBitmapData.height * scaleFac-
tor;

                // Create a new BitmapData object, sized to hold the
scaled bitmap
                var scaledBitmapData:BitmapData = new
BitmapData(newWidth, newHeight,

        loadedBitmapData.transparent);

                // Create a transformation matrix that will scale the
bitmap
                var scaleMatrix:Matrix = new Matrix();
                scaleMatrix.scale(scaleFactor, scaleFactor);

                //loadedBitmapData.draw(loadedBitmap, new Matrix())
                // Draw the scaled bitmap into the new BitmapData ob-
ject
                scaledBitmapData.draw(loadedBitmapData, scaleMatrix);

                // Replace the original BitmapData object with the
                // new, scaled BitmapData object
                //loadedBitmap.bitmapData = scaledBitmapData;

                // Create a new Bitmap using the BitmapData
        // and display it.
        //var newImage:Bitmap = new Bitmap(loadedBitmapData);

                thumb.bmpButton.bmp.bitmapData = scaledBitmapData;
                //addChild(thumb.bmp);

                //thumb.bmp = loadedImage;
                thumb.bmpButton.addChild(thumb.bmpButton.bmp);
        }

        private function handleProgress(event:ProgressEvent):void {
        // Calculate the percentage loaded
        var percent:Number = Math.round(event.bytesLoaded
                                        / event.bytesTotal * 100 );

        loadProg.text = " Loaded: " + event.bytesLoaded + "\n"
                        + "  Total: " + event.bytesTotal + "\n"
                        + "Percent: " + percent;

                //trace(loadProg.text);
        }

        private function onComplete(event:Event):void {
                //done with loadProg so remove it
        thumb.removeChild(loadProg);
        }

      }//end class
    }//end pkg
    package metaswarm.node.components.thumb   {
            import flash.display.*;
            import flash.text.*;

            public class BmpButton extends Sprite{

                    public var type:String;

                    public var bmp:Bitmap;

                    public function BmpButton() {
                            this.type = "bmp";

                            bmp = new Bitmap();
                    }

                    public function make():void {
                    }
            }
    }
    package metaswarm.node.components.thumb   {
            import flash.display.*;
            import flash.events.*;
        import flash.net.URLRequest;
            import flash.text.*;
            import metaswarm.node.components.Particle;

            public class Thumb extends Particle{

                    public var path:String;

                    public var ldr:BitmapLoadr;
                    public var bmpButton:BmpButton;
                    public var bmpLoaded:Boolean;
                    public var thumbMaker:ThumbMaker;

                    public var header:TextField = new TextField();
                    public var footer:TextField = new TextField();

                    public var closeButton:ThumbButton;
                    public var popoutButton:ThumbButton;
                    public var popinButton:ThumbButton;

                    public function Thumb():void {
                            ldr = new BitmapLoadr(this);
                            bmpButton = new BmpButton();
                            bmpLoaded = false;
                            thumbMaker = new ThumbMaker(this);
                            init();
                    }
                    public function init():void {
                            //addChild(ldr); //no no no
                            visible = false;
```



```
                            //focusRect = true;
                            //tabEnabled = true;
                    }

                    public function load(path:String):void{
                            this.path = path;
                            if(!bmpLoaded){
                                    ldr.loadBmp(path);
                                    bmpLoaded = true;
                            }
                    }
                    public function make():void{
                            thumbMaker.make();
                    }
            }
    }
    package metaswarm.node.components.thumb   {
            import flash.display.*;
            import flash.text.*;

            public class ThumbButton extends SimpleButton {
                    /*private var upColor:uint   = 0xFFCC00;
                    private var overColor:uint = 0xCCFF00;
                    private var downColor:uint = 0x00CCFF;
                    private var black:uint = 0x231F20;*/

                    private static const BLACK:uint = 0x231F20;
                    private static const LIGHTGREY:uint = 0xfcfcfc;
                    private static const WHITE:uint = 0xFFFFFF;
                    private static const BRIGHTRED:uint = 0xFF3333;
                    private static const BRIGHTBLUE:uint = 0x66CDFF;

                    public var size:uint;
                    public var type:String;

                    public function ThumbButton(type:String) {
                            this.size = 15;
                            this.type = type;
                            downState      = new ThumbButtonState(type,
BRIGHTBLUE, BLACK, size);
                            overState      = new ThumbButtonState(type,
BRIGHTRED, BLACK, size);
                            upState        = new ThumbButtonState(type,
WHITE, BLACK, size);
                            hitTestState   = new ThumbButtonState(type,
WHITE, BLACK, size);
                            hitTestState.x = -(size / 4);
                            hitTestState.y = hitTestState.x;
                            useHandCursor  = true;
                    }

                    public function make():void {
                            //
                    }
            }
    }
    package metaswarm.node.components.thumb   {
            import flash.display.*;
            import flash.text.*;

            public class ThumbButtonState extends Sprite {
                    private var typeOfButton:String;
                    private var bgColor:uint;
                    private var lineColor:uint;
                    private var size:uint;

                    public function ThumbButtonState(typeOfButton:String,
lineColor:uint, bgColor:uint, size:uint) {
                            this.typeOfButton = typeOfButton;
                            this.bgColor = bgColor;
                            this.lineColor = lineColor;
                            this.size = size;
                            draw();
                    }

                    private function draw():void {
                            graphics.beginFill(bgColor);
                            graphics.drawRect(0, 0, size, size);
                            graphics.endFill();
                            graphics.lineStyle(1, lineColor);
                            //graphics.beginFill(lineColor);

                            //. . . . .
                            //. a . b .
                            //. . . . .
                            //. c . d .
                            //. . . . .
                            var ax:Number = size*(1/4);
                            var ay:Number = size*(1/4);
                            var bx:Number = size*(3/4);
                            var by:Number = size*(1/4);
                            var cx:Number = size*(1/4);
                            var cy:Number = size*(3/4);
                            var dx:Number = size*(3/4);
                            var dy:Number = size*(3/4);


                            switch (typeOfButton) {
                                    case "closeThumb"://x
                                            //trace(0);
                                            graphics.moveTo(ax, ay);
                                            graphics.lineTo(dx, dy);
                                            graphics.moveTo(bx, by);
                                            graphics.lineTo(cx, cy);
                                            break;
                                    case "popout"://>
                                            //trace(1);
                                            graphics.moveTo(ax, ay);
                                            graphics.lineTo(dx, dy);
                                            graphics.moveTo(bx, by);
                                            graphics.lineTo(dx, dy);
                                            graphics.lineTo(cx, cy);
                                            break;
                                    case "popin"://<
                                            //trace(2);
                                            graphics.moveTo(ax, ay);
                                            graphics.lineTo(dx, dy);
                                            graphics.moveTo(bx, by);
                                            graphics.lineTo(ax, ay);
                                            graphics.lineTo(cx, cy);
                                            break;
                                    default:
                                            trace("Not a valid type of but-
ton");
                            }
                    }
            }
    package metaswarm.node.components.thumb   {
            import flash.display.DisplayObject;
        import flash.display.Graphics;
        import flash.display.Shape;
            import flash.display.Sprite;
            import flash.text.*;

            public class ThumbMaker extends Sprite{

                    private static const CLOSETHUMB:String = "closeThumb";
                    private static const POPOUT:String = "popout";
                    private static const POPIN:String = "popin";

                    private var thumb:Thumb;

                    public function ThumbMaker(thumb:Thumb):void{
                            this.thumb = thumb;
                            init();
                    }
                    public function init():void {
                            //
                    }
                    public function make():void {
                            //
                            makeHeader();
                            //makeFooter();
                            makeButtons();
                            //thumb.bmp.y = thumb.header.height;
                            /*thumb.closeButton = new
ThumbButton(CLOSETHUMB);
                            thumb.closeButton.x = thumb.width-thumb.close-
Button.width;
                            thumb.addChild(thumb.closeButton);*/
                    }

                    private function makeHeader():void {

                            var fmt:TextFormat = new TextFormat("Technic",
12, 0x66CDFF);

                            //thumb.header.height = 15;
                            //thumb.header.width = 125;
                            thumb.header.background = true;
                            thumb.header.backgroundColor = 0x231F20;
```

```
                //thumb.header.alpha = 50;
                //thumb.header.border = true;
                //thumb.header.borderColor = 0x000000;

                thumb.header.autoSize = TextFieldAutoSize.LEFT;
                //addChild(this);*/
                thumb.header.text = "123.jpg";

                thumb.header.setTextFormat(fmt);

                thumb.header.selectable = false;
                thumb.header.mouseEnabled = false;
                //this.id = 0;
                thumb.addChild(thumb.header);
                //initTag();
        }

        private function makeFooter():void {

                var fmt:TextFormat = new TextFormat("Technic",
10, 0x000000);

                thumb.footer.height = 15;
                thumb.footer.width = 125;
                thumb.footer.background = true;
                thumb.footer.backgroundColor = 0xffffff;

                //thumb.header.autoSize = TextFieldAutoSize.
LEFT;
                thumb.footer.text = "footer";

                thumb.footer.setTextFormat(fmt);

                thumb.footer.selectable = false;
                thumb.footer.mouseEnabled = false;
                //this.id = 0;

                //thumb.footer.x = 0;
                thumb.footer.y = thumb.footer.width;

                thumb.addChild(thumb.footer);
                //initTag();
        }

        private function makeButtons():void {

                //this should be in thumb
class*************************
                thumb.closeButton = new ThumbButton(CLOSETHUMB);
                thumb.popoutButton = new ThumbButton(POPOUT);
                thumb.popinButton = new ThumbButton(POPIN);
                //thumb.loadExtButton:CustomSimpleButton = new
CustomSimpleButton("e");

                var buttonSize:uint = thumb.closeButton.size;
                var thumbWidth:uint = 200;

                //var myButton:Button = new Button();
                //thumb.close.label = "x";
                //thumb.close.emphasized = true;
                //thumb.close.width = 15;
                //close.move(20, 20);

                /*thumb.closeButton.make();
                thumb.popoutButton.make();
                thumb.popinButton.make();
                thumb.loadExtButton.make();*/

                thumb.closeButton.x = thumbWidth - buttonSize;
                //thumb.closeButton.y = thumb.header.height;

                thumb.popoutButton.x = thumbWidth - button-
Size*2;
                //thumb.popoutButton.y = (thumb.header.height +
thumb.bmp.height) - 15;
                //thumb.popoutButton.y = buttonSize;

                thumb.popinButton.x = thumbWidth - buttonSize*3;
                //thumb.popinButton.y = (thumb.header.height +
thumb.bmp.height) - 15;
                //thumb.popinButton.y = buttonSize*2;

                //thumb.loadExtButton.x = thumb.header.width-(15
* 3);
                //thumb.loadExtButton.y = thumb.header.
height-15;
                thumb.addChild(thumb.bmpButton);

                thumb.addChild(thumb.closeButton);
                thumb.addChild(thumb.popoutButton);
                thumb.addChild(thumb.popinButton);
                //thumb.addChild(thumb.loadExtButton);


        }
    }
}
package metaswarm.node.nodestates   {
        import metaswarm.node.NodeState;

        public class DraggingState implements NodeState{

                private var node:Node;

                public function DraggingState(node:Node)
                {
                        //trace("construct DraggingState");
                        this.node = node;
                }
                public function select():void{}

                public function deSelect():void{}

                public function toggleSelect():void{}

                public function drag():void{}

                public function drop():void{
                        //trace("PickedState drag: goto dragging
state");
                        node.icon.visible = false;

                        node.icon.stopDrag();
                        node.thumb.stopDrag();

                        node.thumb.x = node.icon.x;
                        node.thumb.y = node.icon.y;
                        node.thumb.setTarget(node.icon.x, node.icon.y);
                        node.thumb.visible = true;

                        //goto dropped state
                        node.nodeState = node.dropped;
                }

                //public function dropThumb():void{}

                public function popout():void{}

                public function popin():void{}

                public function close():void{}

                public function setTarget(tgx:Number,
tgy:Number):void{
                        //trace("DraggingState switch");
                        //node.dot.setTarget(tgx, tgy);
                }

                public function rePosition():void{
                        //trace("DraggingState rePosition");
                        //node.dot.goToTarget();
                        //node.thumb.x = node.icon.x;
                        //node.thumb.y = node.icon.y;

                        //dragging icon so snap dot to icon
                        /*node.dot.x = node.dataBox.x;
                        node.dot.y = node.dataBox.y;*/


                }

                public function reSize():void{
                        //trace("DraggingState reSize");
                        node.icon.scaleToTarget(); //???
                }

                public function startFocus():void{}

                public function stopFocus():void{}
```



```
                public function startHover():void{}

                public function stopHover():void{}
        }
}
package metaswarm.node.nodestates   {
        import metaswarm.node.NodeState;

        public class DroppedState implements NodeState{

                private var node:Node;

                public function DroppedState(node:Node)
                {
                        //trace("construct DroppedState");
                        this.node = node;
                }

                public function select():void{}
                public function deSelect():void{}
                public function toggleSelect():void{}

                public function drag():void{
                        //trace("DroppedState drag: goto dragging
state");

                        //start dragging
                        node.thumb.startDrag();

                        //goto to dragging state
                        node.nodeState = node.tdragging;
                        node.prevState = node.dropped;
                }

                public function drop():void{}

                //public function dropThumb():void{}

                public function popout():void{
                        //trace("DroppedState popout");

                        //position dataBox next to thumb and show
                        //node.dataBox.x = node.thumb.x + node.thumb.
width;
                        node.dataBox.x = node.thumb.x + 125;
                        node.dataBox.y = node.thumb.y;
                        node.dataBox.visible = true;

                        //goto popped state
                        node.nodeState = node.popped;
                }

                public function popin():void{}

                public function close():void{
                        //trace("DroppedState close");

                        //position dot at target
                        node.dot.snapToTarget();

                        //position icon at thumb position
                        node.icon.x = node.thumb.x;
                        node.icon.y = node.thumb.y;
                        node.icon.setTarget(node.dot.x, node.dot.y);
                        node.icon.visible = true;

                        //hide thumb, but should i dispose of bitmap???
                        node.thumb.visible = false;

                        //go back to idle
                        node.nodeState = node.picked;
                }

                public function setTarget(tgx:Number,
tgy:Number):void{
                        //trace("DroppedState switch");

                        //preview mainstate needs this to tile
                        //node.thumb.setTarget(tgx, tgy);

                        //set dot target incase thumb is closed,
                        //dot it will get back in place in swarm
                        node.dot.setTarget(tgx, tgy);
                }

                public function rePosition():void{
                        //trace("DroppedState rePosition");

                        //preview mainstate need this to tile
                        node.thumb.goToTarget();
                }

                public function startFocus():void{}
                public function stopFocus():void{}
                public function reSize():void{} //icon is not visible
so don't bother
                public function startHover():void{} //change these
names to hover or smth
                public function stopHover():void{}

        }
}
package metaswarm.node.nodestates   {
        import flash.geom.ColorTransform;
        import metaswarm.node.NodeState;

        public class IdleState implements NodeState{

                private var redTransform:ColorTransform;
                private var blkTransform:ColorTransform;

                private var node:Node;

                function IdleState(node:Node)
                {
                        //trace("construct IdleState");
                        this.node = node;

                        redTransform = new ColorTransform( );
                        redTransform.color = 0xFF0000;
                        blkTransform = new ColorTransform( );
                        blkTransform.color = 0xCCCCCC;
                }
                public function select():void{
                        //trace("IdleState select:goto picked state");

                        //node.dot.mouseEnabled = false;
                        //stopFocus();

                        node.dot.visible = false;

                        //position icon at dot and make visible
                        node.icon.x = node.dot.x;
                        node.icon.y = node.dot.y;
                        node.icon.setTargetWidth(6);
                        node.icon.setTarget(node.dot.x, node.dot.y);
                        node.icon.visible = true;

                        //position tag at dot and make visible
                        node.tag.x = node.dot.x;
                        node.tag.y = node.dot.y;
                        //node.tag.visible = true;

                        //goto picked state
                        node.nodeState = node.picked;
                }

                public function deSelect():void{}

                public function toggleSelect():void{
                        select();
                }

                //public function filter():void{}

                public function drag():void{}

                public function drop():void{}

                //public function dropThumb():void{}

                public function popout():void{}

                public function popin():void{}

                public function close():void{}

                public function setTarget(tgx:Number,
tgy:Number):void{
```

```
                //trace("IdleState setTarget: " + tgx + ", " +
tgy);
                node.dot.setTarget(tgx, tgy);
        }

        public function rePosition():void{
                //trace("IdleState rePosition");
                node.dot.goToTarget();

                node.tag.x = node.dot.x;
                node.tag.y = node.dot.y;
        }

        public function reSize():void{}

        public function startFocus():void{
                //trace("startFocus");
                //node.dot.transform.colorTransform = redTrans-
form;

                /*private function doDrawCircle():void {
        var bounds:Shape = new Shape();
                var r:Rectangle = getRect(node.dot);
        //bounds.graphics.beginFill(bgColor);
                bounds.graphics.lineStyle(0, 0xFF3333); //red
                bounds.graphics.drawRect(r.x, r.y, r.width,
r.height);
                //bounds.graphics.endFill();
                node.addChild(bounds);
        }*/

                node.dot.dotButton.forceOver();
                node.tag.t.borderColor = 0xFF0000;

                node.tag.x = node.dot.x;
                node.tag.y = node.dot.y;
                //node.tag.visible = true;
        }
        public function stopFocus():void{
                //trace("stopFocus");
                //node.dot.transform.colorTransform = blkTrans-
form;
                //node.tag.visible = false;

                node.dot.dotButton.restoreOver();
                node.tag.t.borderColor = 0xCCCCCC;
        }

        public function startHover():void{}

        public function stopHover():void{}

    }
}
package metaswarm.node.nodestates   {
    import flash.events.TimerEvent;
    import flash.geom.ColorTransform;
    import flash.utils.Timer;
    import metaswarm.node.NodeState;

    public class PickedState implements NodeState{

        private var node:Node;

        private static var redTransform:ColorTransform;
        private static var bluTransform:ColorTransform;
        private static var blkTransform:ColorTransform;
        private static var whtTransform:ColorTransform;

        public function PickedState(node:Node)
        {
            //trace("construct PickedState");
            this.node = node;

            redTransform = new ColorTransform( );
            redTransform.color = 0xFF0000;
            bluTransform = new ColorTransform( );
            bluTransform.color = 0x0000ff;
            blkTransform = new ColorTransform( );
            blkTransform.color = 0x231F20;
            whtTransform = new ColorTransform( );
            whtTransform.color = 0xFFFFFF;
        }
        public function select():void{}

        public function deSelect():void{
                //trace("PickedState deselect");
                stopFocus();
                node.icon.visible = false;
                node.tag.visible = false; //not working

                //restore dot
                node.dot.x = node.icon.x;
                node.dot.y = node.icon.y;
                node.dot.setTarget(node.icon.x, node.icon.y);
                node.dot.visible = true;

                node.nodeState = node.idle;
        }

        public function toggleSelect():void{
                deSelect();
        }

        public function drag():void{
                //trace("PickedState drag: goto dragging
state");
                stopFocus();

                //position dot at icon so icon can find its
                //way back if thumb is closed
                node.dot.x = node.icon.x;
                node.dot.y = node.icon.y;

                //node.dataBox.alpha = 50;
                node.icon.startDrag();
                //node.dot.visible = false;
                //node.icon.visible = false;
                //node.thumb.visible = false;

                //remove tag
                node.tag.visible = false;

                //show meta & img
                //node.dataBox.visible = true;
                //node.thumb.visible = true;

                //load thumb
                node.thumb.visible = false;
                node.thumb.load("imgs/" + node.dat[0] + node.
dat[1]);

                node.thumb.make();
                node.thumb.x = node.icon.x;
                node.thumb.y = node.icon.y;
                node.addChild(node.thumb);

                //transition to drag state
                node.nodeState = node.dragging;
        }

        public function drop():void{
                //trace("PickedState drag: goto dragging
state");
                stopFocus();

                //position dot at icon so icon can find its
                //way back if thumb is closed
                node.dot.x = node.icon.x;
                node.dot.y = node.icon.y;

                //node.dataBox.alpha = 50;
                //node.icon.startDrag();
                //node.dot.visible = false;
                //node.icon.visible = false;
                //node.thumb.visible = false;

                //remove tag
                node.tag.visible = false;

                //show meta & img
                //node.dataBox.visible = true;
                //node.thumb.visible = true;

                //load thumb
                node.thumb.visible = true;
                node.thumb.load("imgs/" + node.dat[0] + node.
dat[1]);

                node.thumb.make();
                node.thumb.x = node.icon.x;
                node.thumb.y = node.icon.y;
                node.addChild(node.thumb);
```



```
                //---

                //trace("PickedState drag: goto dragging
state");
                //node.icon.visible = false;

                //node.icon.stopDrag();
                //node.thumb.stopDrag();

                //node.thumb.x = node.icon.x;
                //node.thumb.y = node.icon.y;
                node.thumb.setTarget(node.icon.x, node.icon.y);
                //node.thumb.visible = true;

                //transition to drag state
                node.nodeState = node.dropped;
        }

        //public function dropThumb():void{}

        public function popout():void{}

        public function popin():void{}

        public function close():void{}

        public function setTarget(tgx:Number,
tgy:Number):void{
                //trace("PickedState switch");

                //set new icon target only
                node.icon.setTarget(tgx, tgy);

        }

        public function rePosition():void{
                //trace("PickedState rePosition");

                //update dot position
                node.icon.goToTarget();

                //snap tag to new dot position
                node.tag.x = node.icon.x;
                node.tag.y = node.icon.y;

                //snap dot to new icon position
                //node.dot.x = node.icon.x;
                //node.dot.y = node.icon.y;

                //snap icon to new dot position
                /*node.icon.x = node.dot.x;
                node.icon.y = node.dot.y;*/
        }

        public function reSize():void{
                //trace("PickedState reSize");
                node.icon.scaleToTarget();
        }

        public function startFocus():void{
                //trace("startFocus");
                node.icon.iconButton.forceOver();
                node.tag.x = node.icon.x;
                node.tag.y = node.icon.y;
                //node.tag.visible = true;
                node.tag.t.borderColor = 0xFF0000;
        }
        public function stopFocus():void{
                //trace("stopFocus");
                node.icon.iconButton.restoreOver();

                //node.tag.visible = false;
                node.tag.t.borderColor = 0xCCCCCC;
        }

        public function startHover():void{
                //trace("");
                //node.tag.transform.colorTransform = whtTrans-
form;

                //zoom in icon
                node.icon.setTargetWidth(12);
                node.tag.x = node.icon.x;
                node.tag.y = node.icon.y;
                node.tag.visible = true;
        }

        public function stopHover():void{
                //trace("");
                //node.tag.transform.colorTransform = blkTrans-
form;

                //zoom back out
                node.icon.setTargetWidth(6);
                node.tag.visible = false;
        }
}
package metaswarm.node.nodestates   {
    import metaswarm.node.NodeState;

    public class PoppedState implements NodeState{

        private var node:Node;

        public function PoppedState(node:Node)
        {
            //trace("construct DroppedState");
            this.node = node;
        }

        public function select():void{}
        public function deSelect():void{}
        public function toggleSelect():void{}

        public function drag():void{
            //trace("DroppedState drag: goto dragging
state");

            //start dragging
            node.thumb.startDrag();

            //goto to dragging state
            node.nodeState = node.tdragging;
            node.prevState = node.popped;
        }

        public function drop():void{}
        //public function dropThumb():void{}
        public function popout():void{}

        public function popin():void{
            //trace("DroppedState popout");

            //position dataBox next to thumb and show
            node.dataBox.visible = false;

            //goto popped state
            node.nodeState = node.dropped;
        }

        public function close():void{
            //trace("DroppedState close");

            //position dot at target
            node.dot.snapToTarget();

            //position icon at thumb position
            node.icon.x = node.thumb.x;
            node.icon.y = node.thumb.y;
            node.icon.setTarget(node.dot.x, node.dot.y);
            node.icon.visible = true;

            node.thumb.visible = false; //hmm should i dis-
pose of bitmap???
            node.dataBox.visible = false;

            node.nodeState = node.picked;
        }

        public function setTarget(tgx:Number,
tgy:Number):void{
            //trace("DroppedState switch");

            //preview mainstate needs this to tile
            //node.thumb.setTarget(tgx, tgy);

            //set dot target incase thumb is closed,
            //dot it will get back in place in swarm
            node.dot.setTarget(tgx, tgy);
```

```
		}

		public function rePosition():void{
			//trace("DroppedState rePosition");

			//preview mainstate need this to tile
			node.thumb.goToTarget();

			//snap databox to thumb location
			node.dataBox.x = node.thumb.x + 125;
			node.dataBox.y = node.thumb.y;



		}

		public function startFocus():void{}
		public function stopFocus():void{}
		public function reSize():void{} //icon is not visible
so don't bother
		public function startHover():void{} //change these
names to hover or smth
		public function stopHover():void{}

	}
}
package metaswarm.node.nodestates   {
	import metaswarm.node.NodeState;

	public class ThumbDraggingState implements NodeState{

		private var node:Node;

		function ThumbDraggingState(node:Node)
		{
			//trace("construct DraggingState");
			this.node = node;
		}
		public function select():void{}

		public function deSelect():void{}

		public function toggleSelect():void{}

		public function drag():void{}

		public function drop():void{
			//trace("PickedState drag: goto dragging
state");
			//node.icon.visible = false;

			//node.icon.stopDrag();
			node.thumb.stopDrag();

			//node.thumb.x = node.icon.x;
			//node.thumb.y = node.icon.y;
			node.thumb.setTarget(node.thumb.x, node.
thumb.y);
			//node.thumb.visible = true;

			//goto dropped state
			node.nodeState = node.prevState;
		}

		//public function dropThumb():void{}

		public function popout():void{}

		public function popin():void{}

		public function close():void{}

		public function setTarget(tgx:Number,

tgy:Number):void{
			//trace("DraggingState switch");
			//node.dot.setTarget(tgx, tgy);
		}

		public function rePosition():void{
			//trace("DraggingState rePosition");
			//node.dot.goToTarget();
			//node.thumb.x = node.icon.x;
			//node.thumb.y = node.icon.y;

			//dragging icon so snap dot to icon
			/*node.dot.x = node.dataBox.x;
			node.dot.y = node.dataBox.y;*/

			//snap data box to thumb while dragging
			node.dataBox.x = node.thumb.x + 125;
			node.dataBox.y = node.thumb.y;
		}

		public function reSize():void{
			//trace("DraggingState reSize");
			node.icon.scaleToTarget(); //???
		}

		public function startFocus():void{}

		public function stopFocus():void{}

		public function startHover():void{}

		public function stopHover():void{}
	}
}
package metaswarm.scrap   {
	import flash.display.DisplayObject;
    import flash.display.Graphics;
    import flash.display.Shape;
	import flash.display.Sprite;
	import metaswarm.node.components.Particle;


	public class Box extends Particle{

		private static const Z_EASE:Number    = 1;
		private static const Z_SNAP:Number    = 1;

		//Target zoom
		private var tgw:Number;

		//radial distance to zoom target
		private var dw:Number;

		//zoom velocity
		private var vw:Number;

		public function Box(tgw:Number=0,
                                                dw:Number=0,
                                                vw:Number=0):void
		{
			this.x = 0;
			this.y = 0;
			this.tgw = tgw;
			this.dw = dw;
			this.vw = vw;
			init();
		}
		public function init():void {
			visible = false;
			focusRect = true;
			tabEnabled = true;
			doubleClickEnabled = true;
			//mouseEnabled = false;
		}

		public function load():void{
			//
		}
		//might depend on load later, so keep here
		//rather than put in init
		public function make():void{
			this.graphics.beginFill(0x0000ff);
			this.graphics.lineStyle(0);
			this.graphics.drawRect(0, 0, 10, 10);
			this.graphics.endFill();

		}

		public function scaleToTarget():void {
			dw = tgw - this.width;
			if(Math.abs(dw - Z_SNAP)<1){
				this.width = tgw;
				this.height = tgw;
				//removeEventListener(Event.Enter_FRAME,
onEnterFrame);
			}else{
				vw = dw * Z_EASE;
				this.width += vw;
				this.height += vw;
```



```
			}
		}

		public function setTargetWidth(a:Number) {
			tgw = a;
		}
	}
}

package metaswarm.scrap   {
	import flash.display.DisplayObject;
    import flash.display.Graphics;
    import flash.display.Shape;
	import flash.display.Sprite;
	import metaswarm.node.components.Particle;

	public class Cross extends Particle{

		public function Cross():void{
			init();
		}
		public function init():void {
			//visible = true;
			//focusRect = true;
			//tabEnabled = true;
		}

		public function load():void{
			//
		}
		//might dep on load later, so leave here!
		public function make():void{
			//trace("cross.make()");
			this.graphics.lineStyle(0);
			//this.graphics.lineStyle(1, 0xFF0000);
			this.graphics.moveTo(x, y-3);
			this.graphics.lineTo(x, y+3);
			this.graphics.moveTo(x-3, y);
			this.graphics.lineTo(x+3, y);
			/*trace("visible: " + this.visible);
			trace("x: " + this.x);
			trace("y: " + this.y);*/
		}
	}
}

package metaswarm.scrap   {
	import flash.display.*;
	import flash.net.URLRequest;
    import metaswarm.node.components.thumb.BitmapLoadr;
    import metaswarm.node.components.Particle;

	public class Image extends Particle{

		var path:String;
		var ldr:BitmapLoadr;
		var bmp:Bitmap;
		var bmpLoaded:Boolean;

		public function Image():void {
			ldr = new BitmapLoadr(this);
			bmp = new Bitmap();
			bmpLoaded = false;
			init();
		}
		public function init():void {
			addChild(ldr);
			//l.loadImg("fake string");

			visible = false;
			//focusRect = true;
			//tabEnabled = true;
		}

		public function load(path:String):void{
			this.path = path;
		}
		public function make():void{
			if(!bmpLoaded){
				ldr.loadBmp(path);
				bmpLoaded = true;
			}
		}
```

```
	}
}
package metaswarm.scrap   {
  import flash.display.*;
  import flash.text.*;
  import flash.events.*
  import flash.net.*;

  public class Loadr extends Sprite {

	var loadProg:TextField;
	var gnurbwurk:Metaswarm;

    public function Loadr(gnurbwurk:Metaswarm)
	{
		this.gnurbwurk = gnurbwurk;
		this.loadProg = new TextField();

		init();
    }

    private function init():void
	{
	//loadProg.width = stage.stageWidth;
	//loadProg.height = stage.stageHeight;
	addChild(loadProg);
    }

	public function preload():void {

		var loader:URLLoader = new URLLoader();

		// Instruct the loader to read the file as plain text -
This line is not
		// necessary because the dataFormat is DataFormat.TEXT
by default.
		loader.dataFormat = URLLoaderDataFormat.TEXT;

		// Register an event handler for when the data is fin-
ished downloading
		loader.addEventListener(Event.COMPLETE, loadComplete);

		// Listen for the progress event to check download
progress
		loader.addEventListener(ProgressEvent.PROGRESS, han-
dleProgress);

		// Load the HTML text from the example.html file
		loader.load(new URLRequest("../assets/index4.txt"));
    }

	private function handleProgress(event:ProgressEvent):void
	{
	// Calculate the percentage by multiplying the loaded-to-total
	// ratio by 100
	var percent:Number = Math.round(event.bytesLoaded
                                    / event.bytesTotal * 100 );

	loadProg.text = " Loaded: " + event.bytesLoaded + "\n"
                  + "  Total: " + event.bytesTotal + "\n"
                  + "Percent: " + percent;

		//trace(loadProg.text);
    }


    private function loadComplete(event:Event):void
	{
		var loader:URLLoader = URLLoader(event.target);

		// assign to output. The data property of the URLLoader
is the file contents.
		parseData(loader.data);
    }

	private function parseData(loadedData:String):void
	{
		var parsedData:Array = new Array();
		//var parsedMenu:Array = new Array();

		// Split the string into an array of words using a
space as the delimiter.
		var lines:Array = loadedData.split("\n");
		var words:Array = new Array();

		// Loop through the array and do something with each
```

```
word.
                // In this example, just output the values.
                for ( var i:int = 0; i < lines.length; i++ ) {
                        words = lines[i].split("\t");
                        parsedData.push(words);
                         //trace(parsedData[i][1]);
                }
                //gnurbwurk.parsedData = parsedData;
                //gnurbwurk.preloadComplete(parsedData);
                gnurbwurk.load(parsedData);
                //gnurbwurk.main.makePatterns();
        }

  }//end class
}//end pkg
package metaswarm.subset    {
        import flash.display.DisplayObject;
        import flash.display.Sprite;
        import flash.display.Graphics;
    import flash.display.Shape;

        public class Subset extends Sprite{

                public var nodes:Array;
                public var tiler:SubsetTiler;
                //var count:uint;

                //constructor:
                public function Subset(){
                        nodes = new Array();
                        tiler = new SubsetTiler(this);
                }

                public function tile():void {
                        //trace("tile");
                        tiler.grid();
                }

                public function restore():void {
                        //trace("tile");
                        tiler.restore();
                }

                public function disable():void {
                        trace("disable");
                        for(var i:uint=0; i<nodes.length; i++){
                                nodes[i].visible = false;
                        }
                }

        }//end class
}//end pkg
package metaswarm.subset    {
        import flash.display.DisplayObject;
        import flash.display.Sprite;
        import flash.display.Graphics;
    import flash.display.Shape;

        public class SubsetTiler extends Sprite{

                public var subset:Subset;
                //var tileSize:uint;

                //constructor:
                public function SubsetTiler(subset:Subset){
                        this.subset = subset;
                        //this.tileSize = 100; //later define dynamically
                }

                //this is beastly for now. it sets target of both
databox
                //and tag, regardless of whether dropped or picked
called it.
                //and s should not be hard coded!!
                public function grid():void{
                        //trace("tiler.grid");
                        var s:uint = 200;
                        var cnt:uint = subset.nodes.length;
                        //trace("cnt: " + cnt);
                        //var xMargin:uint = s/4;
                //var yMargin:uint = s/4;
                        var xsp:uint = s;
                        var ysp:uint = s;
                        var cols:uint = Math.ceil(((s+ysp) * cnt) /
1080);
                        //trace("cols: " + cols);
                        var xoff:uint = s/4;
                        var yoff:uint = s/4;

                        for(var i:uint=0; i<cnt; i++){
                                //trace("subset.nodes[i].thumb.x: " +
subset.nodes[i].meta.x);
                                //trace("subset.nodes[i].thumb.y: " +
subset.nodes[i].meta.y);
                                var px:Number = subset.nodes[i].thumb.x;
                                var py:Number = subset.nodes[i].thumb.y;
                                subset.nodes[i].thumb.setPrev(px, py);
                                //
                                var tgx:Number = xoff + xsp*(i%cols);
                                var tgy:Number = yoff + ysp*Math.floor(i/
cols);
                                subset.nodes[i].popin();
                                subset.nodes[i].thumb.setTarget(tgx,
tgy);
                                //subset.nodes[i].tag.setTarget(tgx,
tgy);
                        }
                }

                public function restore():void{
                        //trace("tiler.restore");
                        var cnt:uint = subset.nodes.length;
                        for(var i:uint=0; i<cnt; i++){
                                //trace("subset.nodes[i].thumb.x: " +
subset.nodes[i].meta.x);
                                //trace("subset.nodes[i].thumb.y: " +
subset.nodes[i].meta.y);
                                var tgx:Number = subset.nodes[i].thumb.
px;
                                var tgy:Number = subset.nodes[i].thumb.
py;
                                subset.nodes[i].popin();
                                subset.nodes[i].thumb.setTarget(tgx,
tgy);
                                //subset.nodes[i].tag.setTarget(tgx,
tgy);
                        }
                }

        }//end class
}//end pkg
package metaswarm.swarm    {
        import flash.display.DisplayObject;
        import flash.display.Sprite;
        import flash.display.Graphics;
    import flash.display.Shape;
    import metaswarm.swarm.swarmstates.BrowningState;
    import metaswarm.swarm.swarmstates.GridState;
    import metaswarm.swarm.swarmstates.LorenzeState;
    import metaswarm.swarm.swarmstates.ScatterState;
    import metaswarm.swarm.swarmstates.TypeState;
    import metaswarm.swarm.swarmstates.MapState;
    import metaswarm.subset.Subset;

        public class Swarm extends Sprite{

                //omit constants for now

                //Except for TextField and Video objects,
                //a display object with no content has a
                //width of 0, even if you try to set width
                //to a different value, so need these
                public var w:int;
                public var h:int;

                //internal composite objects
                public var nodes:Array;
                public var nodeCnt:uint;
                public var propCnt:uint;
                public var sortOrder:uint;

                //internal states
                public var lorenze:SwarmState;
                public var browning:SwarmState;
                public var scatter:SwarmState;
                public var grid:SwarmState;
                public var type:SwarmState;
                public var map:SwarmState;

                //internal state holder
                public var swarmState:SwarmState;
```



```
                //var swarmLoader:SwarmLoader;
                //var patternMaker:PatternMaker;

                //constructor:
                public function Swarm(){

                        //instantiate composite objs
                        nodes = new Array();

                        lorenze = new LorenzeState(this);
                        browning = new BrowningState(this);
                        scatter = new ScatterState(this);
                        grid = new GridState(this);
                        type = new TypeState(this);
                        map = new MapState(this);

                        init();
                }
                private function init():void{
                        //set intial state and show unique
                        //tags for that state
                        swarmState = grid;
                        sortOrder = 2;
                        showUniqueTags();
                }


                //these need to be changed
                public function preview():void {
                        var dropped:Subset = new Subset();
                        //var undropped:Subset = new Subset();
                        for(var i:uint=0; i<nodeCnt; i++){
                                if(nodes[i].isDropped() || nodes[i].
isPopped()){
                                        dropped.nodes.push(nodes[i]);
                                }else{
                                        //undropped.push(nodes[i]);
                                        nodes[i].visible = false;
                                }
                        }
                        dropped.tile();
                        //undropped.disable();
                }

                public function explore():void {
                        var dropped:Subset = new Subset();
                        //var undropped:Subset = new Subset();
                        for(var i:uint=0; i<nodeCnt; i++){
                                if(nodes[i].isDropped() || nodes[i].
isPopped()){
                                        dropped.nodes.push(nodes[i]);
                                }else{
                                        //undropped.push(nodes[i]);
                                        nodes[i].visible = true;
                                }
                        }
                        dropped.restore();
                        //undropped.disable();
                }

                //methods
                public function select(node:Node):void {
                        clearSelection();
                        node.select();
                }
                public function add(node:Node):void {
                        node.select();
                }
                public function remove(node:Node):void {
                        node.deSelect();
                }
                public function toggleSelect(node:Node):void {
                        node.toggleSelect();
                }
                public function clearSelection():void {
                        for(var i:uint=0; i<nodeCnt; i++){
                                nodes[i].deSelect();
                        }
                }
                public function selectBy(t:String):void {
                        clearSelection();
                        for(var i:uint=0; i<nodeCnt; i++){
                                if(nodes[i].tag.t.text == t){
                                        nodes[i].select();
                                }
                        }
                }
                public function addBy(t:String):void {
                        for(var i:uint=0; i<nodeCnt; i++){
                                if(nodes[i].tag.t.text == t){
                                        nodes[i].select();
                                }
                        }
                }
                public function filterBy(t:String):void {
                        for(var i:uint=0; i<nodeCnt; i++){
                                if(nodes[i].tag.t.text != t){
                                        //trace("deSelect");
                                        nodes[i].deSelect();
                                }
                        }
                }
                public function popin(node:Node):void{
                        node.popin();
                }
                public function popout(node:Node):void{
                        node.popout();
                }

                public function close(node:Node):void{
                        node.close();
                }

                public function drag(node:Node):void{
                        node.drag();
                }

                public function drop(node:Node):void{
                        node.drop();
                }
                public function dropSelected():void{
                        for(var i:uint=0; i<nodeCnt; i++){
                                if(nodes[i].isPicked()){
                                        nodes[i].drop();
                                }
                        }
                }
                public function startHover(node:Node):void{
                        node.startHover();
                }

                public function stopHover(node:Node):void{
                        //trace("swarm.stopHover");
                        node.stopHover();
                }

                public function startFocusBy(t:String):void {
                        for(var i:uint=0; i<nodeCnt; i++){
                                if(nodes[i].tag.t.text == t){
                                        nodes[i].startFocus();
                                }
                        }
                }

                public function startSelectByFocus(t:String):void {
                        for(var i:uint=0; i<nodeCnt; i++){
                                if(nodes[i].tag.t.text == t){
                                        nodes[i].startFocus();
                                }
                        }
                }

                public function startFilterByFocus(t:String):void {
                        for(var i:uint=0; i<nodeCnt; i++){
                                if(nodes[i].tag.t.text == t && nodes[i].
isPicked()){
                                        nodes[i].startFocus();
                                }
                        }
                }

                public function stopFocusBy():void {
                        for(var i:uint=0; i<nodeCnt; i++){
                                nodes[i].stopFocus();
                        }
                }

                public function printOutNodes():void{
                        trace("printOutCoordinates: ********************
********************************");
                        for(var i:uint=0; i<nodeCnt; i++){
                                trace("node " + i + ": ");
```

```
                        for(var j:uint=0; j<propCnt; j++){
                                trace("xpos,ypos " + j + ": " +
nodes[i].xpos[0][j] + ", " + nodes[i].ypos[0][j]);
                        }
                }
        }

        public function rePosition():void{
                //trace("swarm rePosition");
                for(var i:uint=0; i<nodeCnt; i++){
                        nodes[i].rePosition();
                }
        }

        public function reSize():void{
                //trace("swarm reSize");
                for(var i:uint=0; i<nodeCnt; i++){
                        nodes[i].reSize();
                }
        }

        public function incrSort():void{
                //trace("incrSort");
                if(sortOrder < propCnt-1){
                        sortOrder++;
                }else{
                        sortOrder = 0;
                }
                for(var i:uint=0; i<nodeCnt; i++){
                        //swarm.nodes[i].tag.setTxt(1);
                        nodes[i].tag.incr();
                }
                setPattern();
                showUniqueTags();
        }
        public function decrSort():void{
                //trace("decrSort");
                if(sortOrder > 0){
                        sortOrder--;
                }else{
                        sortOrder = propCnt-1;
                }
                for(var i:uint=0; i<nodeCnt; i++){
                        //swarm.nodes[i].tag.setTxt(1);
                        nodes[i].tag.decr();
                }
                setPattern();
                showUniqueTags();
        }

        public function showUniqueTags():void{
                for(var i:uint=0; i<nodeCnt; i++){
                        nodes[i].tag.visible = nodes[i].tag.isU-
niq;
                }
        }


        //delegated tas
ks****************************************
        public function incrState():void{
                //trace("delegate incrState");
                swarmState.incrState();
        }
        public function decrState():void{
                //trace("delegate decrState");
                swarmState.decrState();
        }
        public function setPattern():void{
                //trace("delegate swarm switch");
                swarmState.setPattern();
        }

    }//end swarm
}//end pkg


package metaswarm.swarm    {
  import flash.display.*;
  import flash.text.*;
  import flash.events.*
  import flash.net.*;
  import metaswarm.node.NodeLoader;
  import metaswarm.*;
```



```
public class SwarmLoader extends Sprite {

    private var _swarm:Swarm;

  public function SwarmLoader(swarm:Swarm)
    {
            this._swarm = swarm;
            //this.loadProg = new TextField();

            init();
  }

  private function init():void
    {
    //loadProg.width = stage.stageWidth;
    //loadProg.height = stage.stageHeight;
    //addChild(loadProg);
  }

    public function load(dat:Array):void{
            //trace("swarmloader load");
            _swarm.nodeCnt = dat.length;
            _swarm.propCnt = dat[0].length;
            //create nodes and add each to display list
            for(var i:uint=0; i<_swarm.nodeCnt; i++){
                    var tmp:Node = new Node();
                    var nodeLoader:NodeLoader = new NodeLoader(tmp);


                    nodeLoader.load(dat[i]); //maybe call after push
                    _swarm.nodes.push(tmp);
            }
    }
    public function make():void{
            //swarm.patternMaker.make(); //decouple from swarm
            for(var i:uint=0; i<_swarm.nodeCnt; i++){
                    var nodeLoader:NodeLoader = new NodeLoader(_
swarm.nodes[i]);
                    nodeLoader.make();
            }
    }
    public function go():void{
            //swarm.swarmState = swarm.grid;
            //swarm.sortOrder = 0;
            for(var i:uint=0; i<_swarm.nodeCnt; i++){
                    var nodeLoader:NodeLoader = new NodeLoader(_
swarm.nodes[i]);
                    nodeLoader.go();
                    _swarm.addChild(_swarm.nodes[i]);
            }
            //swarm.setPattern();
    }

  }//end class
}//end pkg
package metaswarm.swarm    {

    //fsm interface
    public interface SwarmState{

            //internal interface methods
            //function incrSort():void;
            //function decrSort():void;
            function incrState():void;
            function decrState():void;
            function setPattern():void;
    }
}
package metaswarm.swarm.patternmaker   {
    import flash.display.*;
    import flash.utils.*;
    import flash.events.*;
    import flash.geom.*;
    import flash.text.*;
    import metaswarm.swarm.patterns.GeoMapper;
    import metaswarm.swarm.patterns.LorenzeMaker;
    import metaswarm.swarm.patterns.TypeMaker;
    import metaswarm.swarm.Swarm;
    import metaswarm.swarm.patterns.GridMaker;

    //class needs to be broken up

    public class PatternMaker extends Sprite{

            private var _swarm:Swarm;
            //var typeMaker:TypeMaker; //don't know why i don't lo-
calize to function
```

```
            public function PatternMaker(swarm:Swarm):void{
                    this._swarm = swarm;
                    //this.typeMaker = new TypeMaker();
            }

            public function make():void
            {
                    //property count is hardcoded for now so can't
                    //add/remove patterns w/out changing
                    for(var j:uint=0; j<_swarm.propCnt; j++){
                      sortBy(j);
                      setUniqueTags(j);
                      grid(0, j);
                      scatter(1, j);
                      browning(2, j);
                      lorenze(3, j);
                      type(4,j);
                      map(5,j);
                    }
            }

            public function setUniqueTags(tid:uint):void{
                    var prevTag:String = new String();
                    var curTag:String = new String();
                    prevTag = "";
                    for(var i:uint=0; i<_swarm.nodeCnt; i++){
                            curTag = _swarm.nodes[i].dat[tid];
                            if(curTag == prevTag){
                                    _swarm.nodes[i].tag.uniqArr[tid]
= false;
                            }else{
                                    _swarm.nodes[i].tag.uniqArr[tid]
= true;
                            }
                            prevTag = _swarm.nodes[i].dat[tid];
                    }
            }

            public function type(pid:uint, sid:uint):void
            {
                    //trace("PatternMaker->type()");

                    //generate coordinates
                    var typeMaker:TypeMaker = new TypeMaker();
                    var points:Array = typeMaker.make("me++", _
swarm.nodeCnt, _swarm.w);

                    //map coordinates to nodes positions
                    for(var i:uint=0; i<_swarm.nodeCnt; i++){

                            _swarm.nodes[i].xpos[pid][sid] =
points[i][0];
                            _swarm.nodes[i].ypos[pid][sid] =
points[i][1];
                    }
            }

            //this is beastly for now
            public function grid(pid:uint, sid:uint):void
            {
                    //generate coordinates
                    var gridMaker:GridMaker = new GridMaker();
                    var points:Array = gridMaker.make(_swarm.no-
deCnt, _swarm.w, _swarm.h);

                    //map coordinates to nodes positions
                    for(var i:uint=0; i<_swarm.nodeCnt; i++){

                            _swarm.nodes[i].xpos[pid][sid] =
points[i][0];
                            _swarm.nodes[i].ypos[pid][sid] =
points[i][1];
                    }
            }

            public function scatter(pid:uint, sid:uint):void
            {
                    for(var i:uint=0; i<_swarm.nodeCnt; i++){
                            _swarm.nodes[i].xpos[pid][sid] = Math.
random()*_swarm.w;
                            _swarm.nodes[i].ypos[pid][sid] = Math.
random()*_swarm.h;
                    }
            }
```

```
            //Lorenze Attractor Pattern-just guessed on random
range
            public function lorenze(pid:uint, sid:uint):void
            {
                    //generate coordinates
                    var lorenzeMaker:LorenzeMaker = new LorenzeMak-
er();
                    var points:Array = lorenzeMaker.make(_swarm.no-
deCnt, _swarm.w, _swarm.h);

                    //map coordinates to nodes positions
                    for(var i:uint=0; i<_swarm.nodeCnt; i++){

                            _swarm.nodes[i].xpos[pid][sid] =
points[i][0];
                            _swarm.nodes[i].ypos[pid][sid] =
points[i][1];
                    }
            }

            /*//genBrowning
            public function browning(pid:uint, sid:uint):void{
                    var range:int = 20;
                    swarm.nodes[0].xpos[pid][sid] = Math.
random()*swarm.w;
                    swarm.nodes[0].ypos[pid][sid] = Math.
random()*swarm.h;
                    for(var i:uint=1; i<swarm.nodeCnt; i++){
                            if(swarm.nodes[i].tag.dat[sid] == swarm.
nodes[i-1].tag.dat[sid]){
                                    swarm.nodes[i].xpos[pid][sid] =
swarm.nodes[i-1].xpos[pid][sid] + (Math.random()-.5)*range;
                                    swarm.nodes[i].ypos[pid][sid] =
swarm.nodes[i-1].ypos[pid][sid] + (Math.random()-.5)*range;
                            }else{
                                    swarm.nodes[i].xpos[pid][sid] =
Math.random()*swarm.w;
                                    swarm.nodes[i].ypos[pid][sid] =
Math.random()*swarm.h;
                            }
                    }
            }*/

            //genBrowning
            public function browning(pid:uint, sid:uint):void
            {
                    var range:int = 25;
                    _swarm.nodes[0].xpos[pid][sid] = _swarm.w/2;
                    _swarm.nodes[0].ypos[pid][sid] = _swarm.h/2;
                    for(var i:uint=1; i<_swarm.nodeCnt; i++){
                            _swarm.nodes[i].xpos[pid][sid] = _swarm.
nodes[i-1].xpos[pid][sid] + (Math.random()-.5)*range;
                            _swarm.nodes[i].ypos[pid][sid] = _swarm.
nodes[i-1].ypos[pid][sid] + (Math.random()-.5)*range;
                    }
            }

            //genBrowning
            public function map(pid:uint, sid:uint):void
            {
                    var geoMapper:GeoMapper = new GeoMapper();

                    //var coordinates:Array = new Array();
                    var latIndex:Number = 6;
                    var longIndex:Number = 5;

                    //for each node
                    for(var i:uint=1; i<_swarm.nodeCnt; i++){
                            var geoPoint:Array = [_swarm.nodes[i].
dat[longIndex], _swarm.nodes[i].dat[latIndex]];
                            //coordinates.push(geoPoint);

                            //convert node.dat to screen coordinates
                            var point:Array = geoMapper.
geoToScreen(geoPoint, _swarm.h/2);

                            //assign to node pos
                            _swarm.nodes[i].xpos[pid][sid] =
point[0];
                            _swarm.nodes[i].ypos[pid][sid] =
point[1];
                    }
            }

            //bubble sort. put in util class later
            public function sortBy(tid:uint):void
```

```
		{
			var a:Node = new Node();
			var b:Node = new Node();
			for(var i:uint=0; i<_swarm.nodeCnt; i++){
				for(var j:uint=0; j<_swarm.nodeCnt; j++)
{
					//assign neighbooring Nodes to
tmp nodes
					a = _swarm.nodes[i];
				b = _swarm.nodes[j];
					//if two strings not alphabetized
then swap them
				if(a.tag.dat[tid] > b.tag.dat[tid]){
						//swap
						//trace("swap: " + a.tag.
dat[sid] + " with " + b.tag.dat[sid]);
						_swarm.nodes[i] = b;
						_swarm.nodes[j] = a;
					}
				}
			}
		}
	}//end class
}//end pkg

package metaswarm.swarm.patterns   {
import flash.display.*;
import flash.utils.*;
import flash.events.*;
import flash.geom.*;
import flash.text.*;

	public class BrowningMaker extends Sprite {

		//lorenze constants - do not change
		//private static const XXX:Number = xxx;

		public function BrowningMaker():void{
		}

		//returns a matrix of 2d coordinates
		public function make(numOfPoints:uint, w:uint,
h:uint):Array
		{
			var points:Array = new Array();

			return points;
		}

	}//end class
}//end pkg
package metaswarm.swarm.patterns   {
import flash.display.*;
import flash.events.*;
import flash.geom.*;
import flash.text.*;
import flash.utils.*;

	public class GeoMapper extends Sprite {

		// map dependant
		private static const LONG_INIT:Number = -180;
		private static const LAT_INIT:Number = 90;
		private static const LONG_FINAL:Number = 180;
		private static const LAT_FINAL:Number = -90;
		private static const LONG_DELTA:Number = 360;
		private static const LAT_DELTA:Number = 180;

		public function GeoMapper():void{
		}

		//returns a matrix of 2d coordinates
		public function map(geoPoints:Array,
mapHeight:uint):Array
		{
			var points:Array = new Array();

			//lorenze iterative functions
			for(var i:uint=0; i<geoPoints.length; i++){
				//set next point coordinate
				var nextPoint:Array =
geoToScreen(geoPoints[i], mapHeight);
				points.push(nextPoint);
			}
			return points;
		}

		//returns a matrix of 2d coordinates
		public function geoToScreen(geoPoint:Array,
mapHeight:uint):Array
		{
			var long:Number = geoPoint[0];
			var lat:Number = geoPoint[1];
			var mapWidth:uint = mapHeight*2;

			var px:Number = (long + 180) * (mapWidth/360);
			var py:Number = (lat - 90) * (mapHeight/180) *
-1;


			var point:Array = [px, py];

			return point;
		}

	}//end class
}//end pkg
package metaswarm.swarm.patterns   {
import flash.display.*;
import flash.utils.*;
import flash.events.*;
import flash.geom.*;
import flash.text.*;

	public class GridMaker extends Sprite {

		//minimum size that text can be rastered at to provide
		//adequate sampling for large display
		//private static const FONTSIZE:Number = 42;

		public function GridMaker():void{
		}

		//returns a matrix of 2d coordinates
		public function make(numOfPoints:uint, w:uint,
h:uint):Array
		{
			var points:Array = new Array();

			//trace("for pattern: " + pid + ", and sor-
tOrder: " + sid + ": ");
			var xMargin:uint = 0;
		var yMargin:uint = 0;
			var cols:uint = Math.round(Math.
sqrt(numOfPoints));
			var xsp:uint = w/(cols+xMargin);
			var ysp:uint = h/(cols+yMargin);
			var xoff:uint = (w - xsp*cols)/2;
			var yoff:uint = (h - ysp*cols)/2;

			for(var i:uint=0; i<numOfPoints; i++){

				var px:Number = xoff + xsp*(i%cols);
				var py:Number = yoff + ysp*Math.floor(i/
cols);
				var nextPoint:Array = [px, py]
				points.push(nextPoint);
			}

			return points;
		}

	}//end class
}//end pkg
package metaswarm.swarm.patterns   {
import flash.display.*;
import flash.utils.*;
import flash.events.*;
import flash.geom.*;
import flash.text.*;

	public class LorenzeMaker extends Sprite {

		//lorenze constants - do not change
		private static const H:Number = 0.01;
		private static const A:Number = 10.0;
		private static const B:Number = 28.0;
		private static const C:Number = 8.0/3.0;

		public function LorenzeMaker():void{
		}
```



```
		//returns a matrix of 2d coordinates
		public function make(numOfPoints:uint, w:uint,
h:uint):Array
		{
			var points:Array = new Array();

			var sf:Number = h/300; //scale factor. adjust by
steping divisor by 100
			var xoff:uint = w/2;
			var yoff:uint = h/2;
			var x0:Number = (Math.random() * 10) / 10;
			var y0:Number = Math.random() * 20;
			var z0:Number = Math.random() * 10;

			//lorenze iterative functions
			for(var i:uint=0; i<numOfPoints; i++){
				var x1:Number = x0 + H * A * (y0-x0);
				var y1:Number = y0 + H * (x0 * (B - z0)
- y0);
				var z1:Number = z0 + H * (x0 * y0 - C *
z0);

				//set next point coordinate
				var px:Number = (x0 * 10) * sf + xoff;
				var py:Number = (y0 * 10) * sf + yoff;
				var nextPoint:Array = [px, py]
				points.push(nextPoint);

				//feedback
				x0 = x1;
				y0 = y1;
				z0 = z1;
			}
			return points;
		}

	}//end class
}//end pkg
// map dependant
var LONG_INIT = -180;
var LAT_INIT = 90;
var LONG_FINAL = 180;
var LAT_FINAL = -90;
var LONG_DELTA = 360;
var LAT_DELTA = 180;

// zoom/position dependant
var mapX = 0;
var mapY = 0;
var mapWidth = 800; //Stage.width;
var mapHeight = mapWidth/2; //lock aspect


	this._parent.map._x = mapX;
	this._parent.map._y = mapY;
	this._parent.map._width = mapWidth;
	this._parent.map._height = mapHeight;
	this._parent.map.setMask(msk);


// plot dots when button is pressed
this._parent.generateButton.onRelease = function() {

	//trace(rs.getLength());
	fields = rs.getColumnNames();
	for(var i=0; i<rs.getLength(); i++){
		//trace(rs.getItemAt(i)[fields[0]] + " , " +
rs.getItemAt(i)[fields[1]]);
		//imageId = rs.getItemAt(i)[fields[0]];
		imageLat = rs.getItemAt(i)[fields[0]];
		imageLong = rs.getItemAt(i)[fields[1]];
		//this._parent.traceTxt.text = Stage.width;
		//mapWidth = 600;
		//mapHeight = 298;

		//convert longitude & latitude to screen cordinates
		//for max precision latitude/longitude incr < lat/long
delta/stage hight/width respectively
		//4 sigfigs ain't bad then for my purposes, 5 would be
better though
		imageX = mapX + Math.round(((number(imageLong) + Math.
abs(LONG_INIT)) * (mapWidth / LONG_DELTA)));
		imageY = mapY + Math.round((((number(imageLat) * -1) +
Math.abs(LAT_INIT)) * (mapHeight / LAT_DELTA)));

		//duplicate movieclip for each image location
		//duplicateMovieClip(this._parent.dot, imageId, 1);
		this._parent.attachMovie("dot", i, i);
		//trace(mc);
		this._parent[i]._x = imageX;
		this._parent[i]._y = imageY;
		//setProperty(this._parent.imageId, _x, imageX);
		//setProperty(this._parent.imageId, _y, imageY);
	}

}
package metaswarm.swarm.patterns   {
import flash.display.*;
import flash.utils.*;
import flash.events.*;
import flash.geom.*;
import flash.text.*;

	public class TypeMaker extends Sprite {

		//minimum size that text can be rastered at to provide
		//adequate sampling for large display
		private static const FONTSIZE:Number = 42;

		public function TypeMaker():void{
		}

		//given a text string, number of times to sample it,
and desired width of output
		//returns a matrix of 2d coordinates
		public function make(strToSample:String,
numOfSamples:uint, outputWidth:Number):Array
		{
			var coords:Array = new Array();
			var bitmap:Bitmap = new Bitmap();

			bitmap = raster(strToSample, FONTSIZE);
			coords = sample(bitmap, numOfSamples, output-
Width);

			return coords;
		}

		private function raster(strToRaster:String,
sizeToRasterAt:uint):Bitmap
		{
			//bitmap obj to draw text onto
			var bitmapData:BitmapData;

			// Create the text
			var txt:TextField = new TextField( );
			txt.text = strToRaster;
			txt.autoSize = TextFieldAutoSize.LEFT;

			var fmt:TextFormat = new TextFormat( );
			fmt.size = sizeToRasterAt;
			fmt.font = "Arial Black";
			fmt.color = 0x000000;

			txt.setTextFormat(fmt);

			// Make the BitmapData object, sized to accommo-
date the text. transp is false.
			bitmapData = new BitmapData(txt.textWidth, txt.
textHeight, false, 0xffffff);

			// Draw the text into the BitmapData object
			bitmapData.draw(txt);

			// Associate the BitmapData object with a Bitmap
object
			var bitmap:Bitmap = new Bitmap(bitmapData);

			return bitmap;
		}

		private function sample(bitmapToSample:Bitmap,
numOfSamples:uint, outputWidth:uint):Array
		{
			//matrix of 2d coordinate arrays to return
			var points:Array = new Array();

			//bitmapData to sample
			var bitmapData:BitmapData = bitmapToSample.bit-
mapData;

			//brute force determination number of black pix-
```

```
els
                        var numOfBlkPixels:uint = 0;
                        for (var i:uint=0; i<bitmapData.width; i++){
                                for (var j:uint=0; j<bitmapData.height;
j++){
                                        //trace(bitmapData.
getPixel(i,j));
                                        if(bitmapData.getPixel(i,j) == 0)
{
                                                numOfBlkPixels ++;
                                        }
                                }
                        }

                        //calc sample step and scale factor to scale
output to desired size
                        var step:Number = Math.sqrt(numOfBlkPixels / nu-
mOfSamples);
                        var sf:Number = outputWidth / bitmapData.width;

                        //resample img every step pixels
                        var count:uint = 0;
                        for (var m:Number=0; m<bitmapData.width;
m+=step){
                                for (var n:Number=0; n<bitmapData.
height; n+=step){
                                        if(bitmapData.getPixel(m,n) == 0)
{
                                                var nextPoint:Array =
[m*sf, n*sf];
                                                //trace("next point: " +
m*sf + ", " + n*sf);
                                                points.push(nextPoint);
                                                count++
                                        }
                                }
                        }

                        //check variables
                        /*trace("numOfBlkPixels: " + numOfBlkPixels);
                        trace("numOfSamples: " + numOfSamples);
                        trace("step: " + step);
                        trace("sf: " + sf);
                        trace("count: " + count);*/

                        //return coordinates of new blk pixels
                        return points;
                }

        }//end class
}//end pkg
package metaswarm.swarm.swarmstates {
        import metaswarm.swarm.Swarm;
        import metaswarm.swarm.SwarmState;

        public class BrowningState implements SwarmState{

                private static const BROWNING:Number = 2;

                private var swarm:Swarm;

                public function BrowningState(swarm:Swarm)
                {
                        //trace("construct GridState");
                        this.swarm = swarm;
                        init();
                }
                public function init():void{
                        //genPattern();
                }
                public function incrState():void{
                        //trace("ScatterState setPattern");
                        swarm.swarmState = swarm.lorenze;
                        swarm.setPattern();
                }
                public function decrState():void{
                        //trace("ScatterState setPattern");
                        swarm.swarmState = swarm.scatter;
                        swarm.setPattern();
                }
                //make this private
                public function setPattern():void{
                        //trace("GridState setPattern");
                        for(var i:uint=0; i<swarm.nodeCnt; i++){
                                swarm.nodes[i].setTarget(BROWNING,
swarm.sortOrder);
                        }
                }

        }//end class
}//end pkg
package metaswarm.swarm.swarmstates   {
        import metaswarm.swarm.Swarm;
        import metaswarm.swarm.SwarmState;

        public class GridState implements SwarmState{

                private static const GRID:Number = 0;

                private var swarm:Swarm;

                public function GridState(swarm:Swarm)
                {
                        //trace("construct GridState");
                        this.swarm = swarm;
                        init();
                }
                public function init():void{
                        //genPattern();
                }
                public function incrState():void{
                        //trace("ScatterState setPattern");
                        swarm.swarmState = swarm.scatter;
                        swarm.setPattern();
                }
                public function decrState():void{
                        //trace("ScatterState setPattern");
                        swarm.swarmState = swarm.map;
                        swarm.setPattern();
                }
                //make this private
                public function setPattern():void{
                        //trace("GridState setPattern");
                        /*var prev:String = new String();
                        prev = "";*/
                        for(var i:uint=0; i<swarm.nodeCnt; i++){
                                swarm.nodes[i].setTarget(GRID, swarm.
sortOrder);

                                /*if(swarm.nodes[i].tag.t.text == prev){
                                        swarm.nodes[i].tag.visible =
false;
                                }else{
                                        swarm.nodes[i].tag.visible =
false;
                                }
                                prev = swarm.nodes[i].tag.t.text;*/
                        }
                }

        }//end class
}//end pkg
package metaswarm.swarm.swarmstates   {
        import metaswarm.swarm.Swarm;
        import metaswarm.swarm.SwarmState;

        public class LorenzeState implements SwarmState{

                private static const LORENZE:int = 3;

                private var swarm:Swarm;

                public function LorenzeState(swarm:Swarm)
                {
                        //trace("construct GridState");
                        this.swarm = swarm;
                        init();
                }
                public function init():void{
                        //genPattern();
                }
                public function incrState():void{
                        //trace("ScatterState setPattern");
                        swarm.swarmState = swarm.type;
                        swarm.setPattern();
                }
                public function decrState():void{
                        //trace("ScatterState setPattern");
                        swarm.swarmState = swarm.browning;
                        swarm.setPattern();
                }
                //make this private
                public function setPattern():void{
                        //trace("GridState setPattern");
                        for(var i:uint=0; i<swarm.nodeCnt; i++){
                                swarm.nodes[i].setTarget(LORENZE, swarm.
sortOrder);
                        }
                }

        }//end class
}//end pkg
package metaswarm.swarm.swarmstates   {
        import metaswarm.swarm.Swarm;
        import metaswarm.swarm.SwarmState;

        public class MapState implements SwarmState{

                private static const MAP:Number = 5;

                private var _swarm:Swarm;

                public function MapState(swarm:Swarm)
                {
                        //trace("construct ScatterState");
                        this._swarm = swarm;
                        init();
                }
                public function init():void{
                        //do nothing
                }
                public function incrState():void{
                        //trace("ScatterState setPattern");
                        _swarm.swarmState = _swarm.grid;
                        _swarm.setPattern();
                }
                public function decrState():void{
                        //trace("ScatterState setPattern");
                        _swarm.swarmState = _swarm.type;
                        _swarm.setPattern();
                }
                public function setPattern():void{
                        //trace("ScatterState setPattern");
                        for(var i:uint=0; i<_swarm.nodeCnt; i++){
                                _swarm.nodes[i].setTarget(MAP, _swarm.
sortOrder);
                        }
                }

        }//end class
}//end pkg
package metaswarm.swarm.swarmstates   {
        import metaswarm.swarm.Swarm;
        import metaswarm.swarm.SwarmState;

        public class ScatterState implements SwarmState{

                private static const SCATTER:Number = 1;

                private var swarm:Swarm;

                public function ScatterState(swarm:Swarm)
                {
                        //trace("construct ScatterState");
                        this.swarm = swarm;
                        init();
                }
                public function init():void{
                        //do nothing
                }
                public function incrState():void{
                        //trace("ScatterState setPattern");
                        swarm.swarmState = swarm.browning;
                        swarm.setPattern();
                }
                public function decrState():void{
                        //trace("ScatterState setPattern");
                        swarm.swarmState = swarm.grid;
                        swarm.setPattern();
                }
                public function setPattern():void{
                        //trace("ScatterState setPattern");
                        for(var i:uint=0; i<swarm.nodeCnt; i++){
                                swarm.nodes[i].setTarget(SCATTER, swarm.
sortOrder);
                        }
                }

        }//end class
}//end pkg
package metaswarm.swarm.swarmstates   {
        import metaswarm.swarm.Swarm;
        import metaswarm.swarm.SwarmState;

        public class TypeState implements SwarmState{

                private static const TYPE:Number = 4;

                private var swarm:Swarm;

                function TypeState(swarm:Swarm)
                {
                        //trace("construct GridState");
                        this.swarm = swarm;
                        init();
                }
                public function init():void{
                        //genPattern();
                }
                public function incrState():void{
                        //trace("ScatterState setPattern");
                        swarm.swarmState = swarm.map;
                        swarm.setPattern();
                }
                public function decrState():void{
                        //trace("ScatterState setPattern");
                        swarm.swarmState = swarm.lorenze;
                        swarm.setPattern();
                }
                //make this private
                public function setPattern():void{
                        //trace("GridState setPattern");
                        for(var i:uint=0; i<swarm.nodeCnt; i++){
                                swarm.nodes[i].setTarget(TYPE, swarm.
sortOrder);
                        }

        }//end class
}//end pkg
package metaswarm.ui
{
    import flash.display.*;
        import flash.events.*;
        import flash.ui.Keyboard;
        import flash.net.*;
        import flash.utils.*;
        import flash.text.*;
        import metaswarm.ui.hoverlabel.HoverLabel;
        import metaswarm.ui.log.Log;
        import metaswarm.ui.sidebar.Sidebar;
        import metaswarm.swarm.Swarm;
        import metaswarm.ui.uistates.AboutState;
        import metaswarm.ui.uistates.ContactState;
        import metaswarm.ui.uistates.ExploreState;
        import metaswarm.ui.uistates.PreviewState;

        //import fl.controls.Label;

        //UI
    public class Ui extends Sprite
    {
                public var w:int;
                public var h:int;

                public var swarm:Swarm;
                public var sidebar:Sidebar;
                public var hoverLabel:HoverLabel;
                public var cmdLog:Log;
                //public var lastCmd:Label;

                //internal states
                public var explore:UiState;
                public var preview:UiState;
                //public var about:UiState;
                //public var contact:UiState;

                //internal state holder
                public var uiState:UiState;

        public function Ui():void
                {
                        swarm = new Swarm();
                        sidebar = new Sidebar();
                        hoverLabel = new HoverLabel();
```

```
		//lastCmd = new Label();
		cmdLog = new Log();

		explore = new ExploreState(this);
		preview = new PreviewState(this);
		//about = new AboutState(this);
		//contact = new ContactState(this);

		uiState = explore;

		init();
	}
	public function init():void{
		//can't access this.stage until main is added to
stage
		//so listen for event then use stage
		addEventListener(Event.ADDED_TO_STAGE, onAdded-
ToStage);
	}

	public function onAddedToStage(event:Event):void
	{
		swarm.x = stage.stageWidth/6;
		swarm.y = stage.stageHeight/6;
		swarm.w = stage.stageWidth*(2/3);
		swarm.h = stage.stageHeight*(2/3);
	}

	//delegate
	public function onOpenState():void{
		//
		uiState.onOpenState();
	}
	public function onEnterFrame():void{
		//
		uiState.onEnterFrame();
	}
	public function onRightKey():void{
		//
		uiState.onRightKey();
	}
	public function onLeftKey():void{
		//
		uiState.onLeftKey();
	}
	public function onUpKey():void{
		//
		uiState.onUpKey();
	}
	public function onDownKey():void{
		//
		uiState.onDownKey();
	}
	public function onSpaceKey():void{
		//
		uiState.onSpaceKey();
	}
	public function onKey1():void{
		//
		uiState.onKey1();
	}
	public function onKey2():void{
		//
		uiState.onKey2();
	}
	public function onKey3():void{
		//
		uiState.onKey3();
	}
	public function onKey4():void{
		//
		uiState.onKey4();
	}
	public function onMouseClick(node:Node,
button:Object):void{
		//
		uiState.onMouseClick(node, button);
	}
	public function onCtrlClick(node:Node,
button:Object):void{
		//
		uiState.onCtrlClick(node, button);
	}
	public function onShftClick(node:Node,
button:Object):void{
		//
		uiState.onShftClick(node, button);
	}
	public function onAltClick(node:Node,
button:Object):void{
		//
		uiState.onAltClick(node, button);
	}
	public function onCtrlShftClick(node:Node,
button:Object):void{
		//
		uiState.onCtrlShftClick(node, button);
	}
	/*function onDataBoxClick(node:Node,
button:Object):void{
		uiState.onDataBoxClick(node, button);
	}*/
	public function onMouseDown(node:Node,
button:Object):void{
		//
		uiState.onMouseDown(node, button);
	}
	public function onMouseUp(target:Node):void{
		//
		uiState.onMouseUp(target);
	}
	public function onMouseOver(target:Node,
button:Object):void{
		//
		uiState.onMouseOver(target, button);
	}
	public function onCtrlOver(target:Node,
button:Object):void{
		//
		uiState.onCtrlOver(target, button);
	}
	public function onShftOver(target:Node,
button:Object):void{
		//
		uiState.onShftOver(target, button);
	}
	public function onCtrlShftOver(node:Node,
button:Object):void{
		//
		uiState.onCtrlShftOver(node, button);
	}
	public function onCtrlOut():void{
		//
		uiState.onCtrlOut();
	}
	public function onShftOut():void{
		//
		uiState.onShftOut();
	}
	public function onCtrlShftOut():void{
		//
		uiState.onCtrlShftOut();
	}
	public function onMouseOut(target:Node):void{
		//
		uiState.onMouseOut(target);
	}

    }//end class
}//end pkg
package metaswarm.ui
{
	import metaswarm.*;
	import metaswarm.swarm.SwarmLoader;
	import metaswarm.swarm.patternmaker.PatternMaker;
	import metaswarm.ui.hoverlabel.LabelMaker;
	import metaswarm.ui.log.LogMaker;
	import metaswarm.ui.sidebar.SidebarMaker;

	public class UiLoader
	{
		private var _ui:Ui;
		private var _labelMaker:LabelMaker;
		private var _sidebarMaker:SidebarMaker;
		private var _patternMaker:PatternMaker;
		private var _logMaker:LogMaker;

		public function UiLoader(main:Ui) {
			this._ui = main;
			this._patternMaker = new PatternMaker(_
ui.swarm);
			this._labelMaker = new LabelMaker(_
ui.hoverLabel);
			this._sidebarMaker = new SidebarMaker(_
ui.sidebar);
			this._logMaker = new LogMaker(_ui.cmdLog);
			init();
		}

		private function init():void {
			//
		}

		public function load(dat:Array):void{
			//trace("main.load()");
			var swarmLoader:SwarmLoader = new SwarmLoader(_
ui.swarm); //<-------------------------------------not sure about this
			swarmLoader.load(dat);
		}
		public function make():void{
			//trace("main.make()");
			var swarmLoader:SwarmLoader = new SwarmLoader(_
ui.swarm); //<-------------------------------------not sure about this
			swarmLoader.make();
			_labelMaker.make();
			_patternMaker.make();
			_sidebarMaker.make();
			_logMaker.make();
		}
		public function go():void{
			//trace("main.go()");
			var swarmLoader:SwarmLoader = new SwarmLoader(_
ui.swarm); //<-------------------------------------not sure about this
			swarmLoader.go();
			_labelMaker.go();
			_sidebarMaker.go();
			_logMaker.go();


			_ui.addChild(_ui.swarm);
			_ui.addChild(_ui.hoverLabel);
			_ui.addChild(_ui.sidebar);
			_ui.addChild(_ui.cmdLog);

			//set initial pattern
			_ui.swarm.setPattern();
		}
	}
}
package metaswarm.ui {

	//fsm interface
	public interface UiState{

		//internal interface methods
		function onOpenState():void;
		function onEnterFrame():void;
		function onRightKey():void;
		function onLeftKey():void;
		function onUpKey():void;
		function onDownKey():void;
		function onSpaceKey():void;
		function onKey1():void;
		function onKey2():void;
		function onKey3():void;
		function onKey4():void;
		function onMouseClick(node:Node, button:Object):void;
		function onCtrlClick(node:Node, button:Object):void;
		function onShftClick(node:Node, button:Object):void;
		function onAltClick(node:Node, button:Object):void;
		function onCtrlShftClick(node:Node, button:Object):void;
		//function onDataBoxClick(node:Node,
button:Object):void;
		function onMouseDown(node:Node, button:Object):void;
		function onMouseUp(target:Node):void;
		function onMouseOver(target:Node, button:Object):void;
		function onCtrlOver(target:Node, button:Object):void;
		function onShftOver(node:Node, button:Object):void;
		function onCtrlShftOver(node:Node, button:Object):void;
		function onCtrlOut():void;
		function onShftOut():void;
		function onCtrlShftOut():void;
		function onMouseOut(target:Node):void;

	}//end interface
}//end pkg

package metaswarm.ui.hoverlabel {
	import flash.display.*;
	import flash.text.*;

	public class HoverLabel extends Sprite{

		//add formatting constants here
		public var labelMaker:LabelMaker;
		public var _output:TextField;
		public var _text:String;

		public function HoverLabel():void
		{
			_output = new TextField();
			_text = new String();
			labelMaker = new LabelMaker(this);

			init();
		}
		public function init():void {
			visible = true;
			focusRect = false;
			tabEnabled = false;
			mouseEnabled = false;
		}

		/*public function load():void{
			//
		}
		public function make():void{
			labelMaker.make();
		}
		public function go():void{
			labelMaker.go();
		}*/

		public function get text():String {
	    return _text;
	}
	public function set text(newText:String):void {
		_text = newText;
		}

		public function update():void{
			_output.x = mouseX + 15;
			_output.y = mouseY + 5;
			_output.text = _text;
		}

		public function show():void{
			_output.visible = true;
		}

		public function hide():void{
			_output.visible = false;
		}


	}//end class
}//end pkg
package metaswarm.ui.hoverlabel {
	import flash.display.*;
	import flash.text.*;

	public class LabelMaker extends Sprite{

		private var hl:HoverLabel;

		public function LabelMaker(hoverLabel:HoverLabel):void{
			this.hl = hoverLabel;
			init();
		}
		public function init():void {
			//
		}
		public function make():void {
			//var fmt:TextFormat = new TextFormat( );
			var fmt:TextFormat = new TextFormat("Technic",
10, 0xfff);

			//tag.t.embedFonts = true;
			//tag.t.antiAliasType = AntiAliasType.ADVANCED;
			hl._output.text = "";
			hl._output.setTextFormat(fmt);
			hl._output.selectable = false;
			hl._output.mouseEnabled = false;
			//log.id = 0;
```

```
                //t.width =
                //t.height =
                hl._output.background = true
                hl._output.backgroundColor = 0xf0ff64; //black
                //t.border = true;
                //t.borderColor = 0x333333; //dark gray
                hl._output.autoSize = TextFieldAutoSize.LEFT;
                //addChild(this);

                //tag.t.text = tag.dat[tag.id];


        }

        public function go():void {
                hl.addChild(hl._output);
        }
    }
}
package metaswarm.ui.log {
    import flash.display.*;
    import flash.text.*;

    public class Log extends Sprite{

        //add formatting constants here
        //public var logMaker:LogMaker;
        public var output:TextField;
        public var entries:Array;

        public function Log():void
        {
                output = new TextField();
                //logMaker = new LogMaker(this);
                entries = new Array();

                init();
        }
        public function init():void {
                visible = true;
                focusRect = false;
                tabEnabled = false;
                //tag.mouseEnabled = false;
        }

        public function load():void{
                //
        }
        public function make():void{
                //logMaker.make();
        }

        public function append(txt:String):void{
                trace("txt: " + txt);
                trace("entries[" + entries.length + "]: " +
entries[entries.length-1]);

                trace("_");

                entries.push(txt);
                //output.unshift(txt + '\n');
                output.replaceText(0, 0, txt + '\n');

                /*
                if(txt != entries[entries.length-1]){
                        trace("log it");
                        entries.push(txt);
                        //output.appendText(txt + '\n');
                        //output.unshift(txt + '\n');
                        output.replaceText(0, 0, txt + '\n');
                }else{
                        //entries.push(txt);
                        //output.appendText(entries.pop() +
'\n');

                        //output.text = entries.pop();
                }
                */
        }


    }//end class
}//end pkg
package metaswarm.ui.log {
    import flash.display.*;
    import flash.text.*;

public class LogMaker extends Sprite{

        private var _log:Log;

        public function LogMaker(log:Log):void{
                this._log = log;
                init();
        }
        public function init():void {
                //
        }
        public function make():void {
                //var fmt:TextFormat = new TextFormat( );

                var fmt:TextFormat = new TextFormat("Technic",
10, 0xfff);
                //fmt.color = 0xfff;
                //tag.t.embedFonts = true;
                //tag.t.antiAliasType = AntiAliasType.ADVANCED;
                _log.output.text = "";
                _log.output.setTextFormat(fmt);
                _log.output.selectable = false;
                _log.output.mouseEnabled = false;
                //log.id = 0;

                //t.width =
                //t.height =
                //_log.output.background = true
                //_log.output.backgroundColor = 0x999999; //
light gray
                //t.border = true;
                //t.borderColor = 0x333333; //dark gray
                _log.output.autoSize = TextFieldAutoSize.LEFT;
                //addChild(this);

                //tag.t.text = tag.dat[tag.id];

                _log.output.x=1100;
                _log.output.y=0;
        }
        public function go():void {
                _log.addChild(_log.output);
        }
    }
}
package metaswarm.ui.sidebar {
    import flash.display.*;
    import flash.text.*;

    public class Sidebar extends Sprite{

        public var sidebarMaker:SidebarMaker;
        //
        public var header:TextField;
        public var shortcuts:TextField;
        public var states:TextField;
        public var nodeList:TextField;
        public var footer:TextField;

        public function Sidebar():void
        {
                header = new TextField();
                shortcuts = new TextField();
                states = new TextField();
                nodeList = new TextField();
                footer = new TextField();

                sidebarMaker = new SidebarMaker(this);

                init();
        }
        public function init():void {
                //
        }
        public function load():void{
                //
        }
        public function make():void{
                sidebarMaker.make();
        }


    }//end class
}//end pkg
package metaswarm.ui.sidebar {
```



```
    import flash.display.*;
    import flash.geom.ColorTransform;
    import flash.text.*;

    public class SidebarMaker extends Sprite{

        public var sb:Sidebar;

        public function SidebarMaker(sidebar:Sidebar):void{
                this.sb = sidebar;
                init();
        }
        public function init():void {
                //
        }
        public function make():void {

                var h1fmt:TextFormat = new TextFormat( );
                h1fmt.font = "technic";
                h1fmt.size = 16;
                h1fmt.leftMargin = 10;

                var fmt:TextFormat = new TextFormat( );
                fmt.font = "technic";
                fmt.size = 16;
                fmt.leftMargin = 10;

                // ************************************ shared
properties
                /*
                var bg:ColorTransform;
                bg.color = 0xFAFAFA;
                var border:ColorTransform;
                border.color = 0xE6E6E6;
                */
                var sbWidth:int = 180;
                var xOffset:int = 20;
                var yOffset:int = 50;

                // ************************************
                sb.header.text = '\n' + "meta.swarm" + '\n\n';
                sb.header.setTextFormat(h1fmt);
                sb.header.selectable = false;
                sb.header.mouseEnabled = false;
                sb.header.x=xOffset;
                //sb.header.y=0;
                sb.header.width = sbWidth;
                sb.header.height = 40;
                //sb.header.autoSize = TextFieldAutoSize.CENTER;
                sb.header.background = true;
                sb.header.backgroundColor = 0xffffff;
                sb.header.border = false;
                sb.header.borderColor = 0xE6E6E6;

                // ************************************
                sb.shortcuts.text = " " + '\n' +
                                                        "naviga-
tion" + '\n\n' +
                                                        "^ - in-
crement sort" + '\n' +
                                                        "v - dec-
rement sort" + '\n' +
                                                        "> - in-
crement pattern" + '\n' +
                                                        "< - dec-
rement pattern" + '\n\n';
                sb.shortcuts.setTextFormat(fmt);
                sb.shortcuts.selectable = false;
                sb.shortcuts.mouseEnabled = false;
                sb.shortcuts.x = xOffset;
                //sb.shortcuts.y = 0;
                sb.shortcuts.width = sbWidth;
                sb.shortcuts.height = 140;
                //sb.shortcuts.autoSize = TextFieldAutoSize.CEN-
TER;
                sb.shortcuts.background = true;
                sb.shortcuts.backgroundColor = 0xffffff;
                sb.shortcuts.border = false;
                sb.shortcuts.borderColor = 0xE6E6E6;

                //  ************************************
                sb.states.text = " " + '\n' +
                                                        "selection" +
'\n\n' +
                                                        "r - select / de-
select" + '\n' +
                                                        "ctrl + r - add /
remove" + '\n' +
                                                        "shft + r -  se-
lect by" + '\n' +
                                                        "ctrl&shft + r -
filter by" + '\n\n';
                sb.states.setTextFormat(fmt);
                sb.states.selectable = false;
                sb.states.mouseEnabled = false;
                sb.states.x = xOffset;
                sb.states.width = sbWidth;
                sb.states.height = 130;
                //sb.states.autoSize = TextFieldAutoSize.LEFT;
                sb.states.background = true;
                sb.states.backgroundColor = 0xffffff;
                sb.states.border = false;
                sb.states.borderColor = 0xE6E6E6;

                // ************************************
                sb.nodeList.text = '\n' + "nodes" + '\n\n';
                sb.nodeList.setTextFormat(fmt);
                sb.nodeList.selectable = false;
                sb.nodeList.mouseEnabled = false;
                //sb.nodeList.scrollV = 1;
                sb.nodeList.x = xOffset;
                sb.nodeList.width = sbWidth;
                sb.nodeList.height = 900 - (sb.header.textHeight
+
sb.shortcuts.textHeight +
sb.states.textHeight +
sb.footer.textHeight);
                //sb.nodeList.autoSize = TextFieldAutoSize.LEFT;
                sb.nodeList.background = true;
                sb.nodeList.backgroundColor = 0xffffff;
                sb.nodeList.border = false;
                sb.nodeList.borderColor = 0xE6E6E6;

                // ************************************
                sb.footer.text = '\n' + "by boot" + '\n\n';
                sb.footer.setTextFormat(fmt);
                sb.footer.selectable = false;
                sb.footer.mouseEnabled = false;
                sb.footer.x = xOffset;
                sb.footer.width = sbWidth;
                sb.footer.height = 40;
                //sb.footer.autoSize = TextFieldAutoSize.LEFT;
                sb.footer.background = true;
                sb.footer.backgroundColor = 0xffffff;
                sb.footer.border = false;
                sb.footer.borderColor = 0xE6E6E6;

                // ************************************
                sb.header.y = yOffset;
                sb.shortcuts.y = sb.header.height;
                sb.states.y = sb.shortcuts.y + sb.shortcuts.
height;
                sb.nodeList.y = sb.states.y + sb.states.height;
                sb.footer.y = 900 - sb.footer.height;

        }

        public function go():void {
                //sb.addChild(sb.header);
                sb.addChild(sb.shortcuts);
                sb.addChild(sb.states);
                //sb.addChild(sb.nodeList);
                //sb.addChild(sb.footer);
        }
    }
}
package metaswarm.ui.uistates   {
    import metaswarm.ui.Ui;
    import metaswarm.ui.UiState;

    public class AboutState implements UiState{

        private var main:Ui;

        public function AboutState(main:Ui)
        {
                //trace("construct ExploreState");
                this.main = main;
                init();
```

```
		}
		private function init():void{
		}

		//delegated meth-
ods********************************************
		public function onEnterFrame():void{
		}
		public function onRightKey():void{
		}
		public function onLeftKey():void{
		}
		public function onUpKey():void{
		}
		public function onDownKey():void{
		}
		public function onSpaceKey():void{
			//main.mainState = main.xxx;
		}
		public function onKey1():void{
			//main.picked.restore();
			//main.mainState = main.explore;
		}
		public function onKey2():void{
			//main.picked.restore();
			//main.dropped.tile();
			//main.mainState = main.preview;
		}
		public function onKey3():void{
			//main.mainState = main.about;
		}
		public function onKey4():void{
			//main.mainState = main.contact;
		}
		public function onMouseClick(node:Node,
button:Object):void{
		}
		public function onCtrlClick(node:Node,
button:Object):void{
		}
		public function onShftClick(node:Node,
button:Object):void{
		}
		public function onAltClick(node:Node,
button:Object):void{
		}
		public function onCtrlShftClick(node:Node,
button:Object):void{
		}
		public function onCtrlShftOver(node:Node,
button:Object):void{
		}
		public function onCtrlShftOut():void{
		}
		public function onMouseDown(node:Node,
button:Object):void{
			//target.drag();
		}
		public function onMouseUp(target:Node):void{
			//target.drop();
		}
		public function onMouseOver(target:Node,
button:Object):void{
			//target.startPrev();
		}
		public function onCtrlOver(target:Node,
button:Object):void{
			//target.startPrev();
		}
		public function onShftOver(target:Node,
button:Object):void{
			//target.startPrev();
		}
		public function onCtrlOut():void{
			//target.startPrev();
		}
		public function onShftOut():void{
			//target.startPrev();
		}
		public function onMouseOut(target:Node):void{
			//target.stopPrev();
		}


	}//end class
}//end pkg
```

```
package metaswarm.ui.uistates   {
	import metaswarm.ui.Ui;
	import metaswarm.ui.UiState;

	class ContactState implements UiState{

		var main:Ui;

		function ContactState(main:Ui)
		{
			//trace("construct ExploreState");
			this.main = main;
			init();
		}
		public function init():void{
		}

		//delegated meth-
ods********************************************
		public function onEnterFrame():void{
		}
		public function onRightKey():void{
		}
		public function onLeftKey():void{
		}
		public function onUpKey():void{
		}
		public function onDownKey():void{
		}
		public function onSpaceKey():void{
			//main.mainState = main.xxx;
		}
		public function onKey1():void{
			//main.mainState = main.explore;
		}
		public function onKey2():void{
			//main.mainState = main.preview;
		}
		public function onKey3():void{
			//main.mainState = main.about;
		}
		public function onKey4():void{
			//main.mainState = main.contact;
		}
		public function onMouseClick(node:Node,
button:Object):void{
		}
		public function onCtrlClick(node:Node,
button:Object):void{
		}
		public function onShftClick(node:Node,
button:Object):void{
		}
		public function onAltClick(node:Node,
button:Object):void{
		}
		public function onCtrlShftClick(node:Node,
button:Object):void{
		}
		public function onCtrlShftOver(node:Node,
button:Object):void{
		}
		public function onCtrlShftOut():void{
		}
		public function onMouseDown(node:Node,
button:Object):void{
			//target.drag();
		}
		public function onMouseUp(target:Node):void{
			//target.drop();
		}
		public function onMouseOver(target:Node,
button:Object):void{
			//target.startPrev();
		}
		public function onCtrlOver(target:Node,
button:Object):void{
			//target.startPrev();
		}
		public function onShftOver(target:Node,
button:Object):void{
			//target.startPrev();
		}
		public function onCtrlOut():void{
			//target.startPrev();
		}
		public function onShftOut():void{
			//target.startPrev();
		}
		public function onMouseOut(target:Node):void{
			//target.stopPrev();
		}


	}//end class
}//end pkg
package metaswarm.ui.uistates {
	import metaswarm.*;
	import metaswarm.ui.Ui;
	import metaswarm.swarm.Swarm;
	import metaswarm.ui.UiState;
	import com.greensock.TweenLite;

	public class ExploreState implements UiState{

		private var main:Ui;

		public function ExploreState(main:Ui)
		{
			//trace("construct ExploreState");
			this.main = main;
			init();
		}
		public function init():void{
		}

		//delegated meth-
ods************************************************
		public function onOpenState():void{
			main.cmdLog.append("in explore state");
		}
		public function onEnterFrame():void{
			main.swarm.rePosition();
			main.swarm.reSize();
			main.hoverLabel.update();
		}
		public function onRightKey():void{
			main.swarm.incrState();
			main.cmdLog.append("next pattern");
			//main.lastCmd.text = "next pattern";
		}
		public function onLeftKey():void{
			main.swarm.decrState();
			main.cmdLog.append("previous pattern");
		}
		public function onUpKey():void{
			main.swarm.incrSort();
			main.cmdLog.append("increment sort");
		}
		public function onDownKey():void{
			main.swarm.decrSort();
			main.cmdLog.append("decrement sort");
		}
		public function onSpaceKey():void{
			//main.dropped.tile(); //could move to preview
state onEnterFrame
			//main.mainState = main.preview;
			main.swarm.dropSelected();
			main.cmdLog.append("drop selected");
		}
		public function onKey1():void{
			//

		}
		public function onKey2():void{
			trace("ExploreState->onKey2");
			main.swarm.preview();
			main.uiState = main.preview;
			//main.uiState.onOpenState();
			main.cmdLog.append("preview mode");
		}
		public function onKey3():void{
			//main.mainState = main.about;

		}
		public function onKey4():void{
			//main.mainState = main.contact;
			TweenLite.to(main.swarm, .25, {scaleX:1, scal-
eY:1});
		}
		public function onCtrlClick(node:Node,
button:Object):void
		{
			switch (button.type) {
				case "dot":
					//dot clear and select
					//main.swarm.add(node);
					//main.cmdLog.append("add/remove
to selection");
					//tag clear and selectby
					main.swarm.add(node);
					//main.swarm.addBy(node.
tag.t.text);
					main.cmdLog.append("add node: " +
node.dat[0] + node.dat[1]);
					break;
				case "icon":
					//tag clear and selectby
					main.swarm.remove(node);
					main.cmdLog.append("remove node:
" + node.dat[0] + node.dat[1]);
					break;
				case "tag":
					//main.swarm.addBy(node.
tag.t.text);
					break;
				default:
					trace("oops, you ctrl clicked
button of type: " + button.type);
			}
		}
		public function onShftClick(node:Node,
button:Object):void
		{
			switch (button.type) {
				case "dot":
					//main.swarm.filterBy(node.
tag.t.text);
					//main.cmdLog.append("filter by: "
+ node.tag.t.text);
					main.swarm.selectBy(node.
tag.t.text);
					main.cmdLog.append("select by: "
+ node.tag.t.text);
					break;
				case "icon":
					//
					//main.swarm.xxx(node);
					main.swarm.selectBy(node.
tag.t.text);
					main.cmdLog.append("select by: "
+ node.tag.t.text);
					break;
				case "tag":
					/*main.swarm.filterBy(node.
tag.t.text);
					main.cmdLog.append("filter by: " +
node.tag.t.text);*/
					break;
				default:
					trace("damn it, you shft clicked
button of type: " + button.type);
			}
		}
		public function onAltClick(node:Node,
button:Object):void
		{
			//can't use alt b/c window uses it to focus
toolbar
		}
		public function onCtrlShftClick(node:Node,
button:Object):void
		{
			switch (button.type) {
				case "dot":
					//
					break;
				case "icon":
					//
					main.swarm.filterBy(node.
tag.t.text);
					main.cmdLog.append("filter by: " +
node.tag.t.text);
					break;
				case "tag":
					//
					break;
				default:
```

```
                                trace("shit, you shft clicked
button of type: " + button.type);
                        }
                }
                public function onMouseClick(node:Node,
button:Object):void
                {
                        switch (button.type) {
                                case "dot":
                                        //dot clear and select
                                        main.swarm.select(node);
                                        main.cmdLog.append("select node:
" + node.dat[0] + node.dat[1]);
                                        break;
                                case "icon":
                                        //tag clear and selectby
                                        //main.swarm.selectBy(node.
tag.t.text);
                                        //main.cmdLog.append("select by:
" + node.tag.t.text);
                                        //DO NOTHING B/C drag will handle
this case! <------------------------NOTE
                                        break;
                                case "tag":
                                        //
                                        //main.swarm.xxx(node);
                                        main.swarm.selectBy(node.
tag.t.text);
                                        //main.cmdLog.append("filter by: "
+ node.tag.t.text);
                                        break;
                                case "popout": //rename popoutData
                                        //trace("popout");
                                        main.swarm.popout(node);
                                        break;
                                case "popin":
                                        //trace("popin");
                                        main.swarm.popin(node);
                                        break;
                                case "closeThumb":
                                        //
                                        main.swarm.close(node);
                                        break;
                                default:
                                        //trace("Not a valid type of but-
ton");
                        }
                }

                //public function onDataBoxClick(node:Node,
button:Object):void{}

                public function onMouseDown(node:Node,
button:Object):void
                {
                        switch (button.type) {
                                case "icon":
                                        //
                                        main.swarm.drag(node); //should
be dragIcon!!
                                        main.cmdLog.append("drag " +
node.dat[0] + node.dat[1]);
                                        break;
                                case "bmp":
                                        main.swarm.drag(node); //should
be dragThumb!!
                                        main.cmdLog.append("drag " +
node.dat[0] + node.dat[1]);
                                        break;
                                default:
                                        //break
                        }
                }
                public function onMouseUp(node:Node):void{
                        main.swarm.drop(node);
                }

                //over
                public function onMouseOver(node:Node,
button:Object):void{
                        main.hoverLabel.show();
                        main.swarm.startHover(node);
                        //main.cmdLog.append("start hover");
                        //main.swarm.startFocus(target.tag.t.text);

                        switch (button.type) {
                                case "dot":
                                        main.hoverLabel.text = "select "
+ node.dat[0] + node.dat[1];
                                        break;
                                case "icon":
                                        main.hoverLabel.text = "drag to
pluck " + node.dat[0] + node.dat[1];
                                        break;
                                case "tag":
                                        main.hoverLabel.text = "Clear and
Select by: " + node.tag.t.text;
                                        break;
                                case "popout": //rename popoutData
                                        main.hoverLabel.text = "show data
for " + node.dat[0] + node.dat[1];
                                        break;
                                case "popin":
                                        main.hoverLabel.text = "hide data
for " + node.dat[0] + node.dat[1];
                                        break;
                                case "closeThumb":
                                        main.hoverLabel.text = "close
thumbnail and return " + node.dat[0] + node.dat[1] + " to the swarm";
                                        break;
                                case "bmp":
                                        main.hoverLabel.text = "drag " +
node.dat[0] + node.dat[1];
                                        break;
                                default:
                                        //trace("Not a valid type of but-
ton");
                        }
                }
                public function onCtrlOver(node:Node,
button:Object):void{
                        main.swarm.startHover(node);
                        //main.cmdLog.append("start hover");
                        //main.swarm.startFocusBy(node.tag.t.text);
                        main.hoverLabel.text = "Add/remove " + node.
dat[0] + node.dat[1];
                }
                public function onShftOver(node:Node,
button:Object):void{
                        main.swarm.startHover(node);
                        main.swarm.startSelectByFocus(node.tag.t.text);
//should be focusDotsBy
                        main.hoverLabel.text = "Select by: " + node.
tag.t.text;
                }
                public function onCtrlShftOver(node:Node,
button:Object):void{
                        main.swarm.startHover(node);
                        main.swarm.startFilterByFocus(node.tag.t.text);
//should be focusIconsBy
                        main.hoverLabel.text = "Filter by: " + node.
tag.t.text;
                }

                //out
                public function onMouseOut(node:Node):void{
                        main.hoverLabel.hide();
                        main.swarm.stopHover(node);
                        //main.cmdLog.append("stop hover");
                        //main.swarm.stopFocus(target.tag.t.text);
                }
                public function onCtrlOut():void{
                        //main.swarm.startHover(node);
                        main.swarm.stopFocusBy();
                        //main.cmdLog.append("stopFocusBy");
                }
                public function onShftOut():void{
                        //main.swarm.startHover(node);
                        main.swarm.stopFocusBy();
                        //main.cmdLog.append("stopFocusBy");
                }
                public function onCtrlShftOut():void{
                        //
                }


        }//end class
}//end pkg
package metaswarm.ui.uistates   {
        import metaswarm.*;
        import metaswarm.ui.Ui;
        import metaswarm.ui.UiState;
```



```
        import metaswarm.swarm.Swarm;
        import com.greensock.TweenLite;

        public class PreviewState implements UiState{

                private var main:Ui;

                public function PreviewState(main:Ui)
                {
                        //trace("construct ExploreState");
                        this.main = main;
                        init();
                }
                public function init():void{
                }

                //delegated meth-
ods********************************************
                public function onOpenState():void{
                        main.cmdLog.append("in preview state");
                }
                public function onEnterFrame():void{
                        main.swarm.rePosition();
                        main.swarm.reSize();
                        main.hoverLabel.update();
                }
                public function onRightKey():void{
                }
                public function onLeftKey():void{
                }
                public function onUpKey():void{
                }
                public function onDownKey():void{
                }
                public function onSpaceKey():void{
                        //main.mainState = main.xxx;
                        TweenLite.to(main.swarm, 1, {scaleX:3, scal-
eY:3});
                }
                public function onKey1():void{
                        trace("previewState->onKey1");
                        main.swarm.explore();
                        main.uiState = main.explore;
                        main.cmdLog.append("explore mode");
                }
                public function onKey2():void{
                        //main.mainState = main.preview;
                }
                public function onKey3():void{
                        //main.mainState = main.about;
                        TweenLite.to(main.swarm, .25, {scaleX:3, scal-
eY:3});
                }
                public function onKey4():void{
                        //main.mainState = main.contact;
                }
                public function onMouseClick(node:Node,
button:Object):void{
                }
                public function onCtrlClick(node:Node,
button:Object):void{
                }
                public function onShftClick(node:Node,
button:Object):void{
                }
                public function onAltClick(node:Node,
button:Object):void{
                }
                public function onCtrlShftClick(node:Node,
button:Object):void{
                }
                public function onCtrlShftOver(node:Node,
button:Object):void{
                }
                public function onCtrlShftOut():void{
                }
                public function onMouseDown(node:Node,
button:Object):void{
                        //target.drag();
                }
                public function onMouseUp(target:Node):void{
                        //target.drop();
                }
                public function onMouseOver(target:Node,
button:Object):void{
                        //target.startPrev();
                }
                public function onCtrlOver(target:Node,
button:Object):void{
                        //target.startPrev();
                }
                public function onShftOver(target:Node,
button:Object):void{
                        //target.startPrev();
                }
                public function onCtrlOut():void{
                        //target.startPrev();
                }
                public function onShftOut():void{
                        //target.startPrev();
                }
                public function onMouseOut(target:Node):void{
                        //target.stopPrev();
                }

        }//end class
}//end pkg
```