TriUlti
iFlowEdit
< (20/12/2011 - 29/12/2011) >

< 29/12/2011>

Weekly Report#7

KARAOĞUZ, Mehmet Ozan

Activities of iFlowEdit

Syntax of Processing Language is extremely similar to the syntax of Java Programming Language. On the account of this fact, I will use Java terms while explaining my part of the report.

Our approach to create activities is Object-Oriented approach, so the first job is to create a public class which is named as Activity. Just like most of the classes, Activity Class has a number of fields, methods and a constructor. Basic information about each of them is below.

Fields:
po: (type: PVector) This is the top left coordinate of the Activity.
si: (type: PVector) This is the size of the Activity.
ce: (type: PVector) This is the center position of the Activity.
name: (type: String) This is the name of the Activity.
locked: (type: boolean) This is true if the Activity is locked.
bover: (type: boolean) This is true if mouse over the Activity.
zindex: (type: int) This is the z index of the Activity.
sp: (type: StickyPoint) This is the sticky points of the Activity.

Methods:
testOver: This method tests whether the mouse is over the Activity.
mouseOver: This method makes mouse is over the Activity.
mouseOut: This method makes mouse is out the Activity.
lock: This method locks the Activity.
move: This method moves the Activity.
display: This method displays the Activity on the screen.

Drawing to the Screen

Creating classes, initializing fields, processing methods and adjusting settings are half part of the job. The main issue is to show these hard work smoothly on the screen. To do this so, we implement a method which is called draw. It is a reserved

function for Processing to draw on the screen. In this method, we simply called the display functions of the activities and connections. We also included a red strange horizontal line which moves from bottom to top in order to trace the process.

KAYRAK, Alaattin

Mouse Events

Mouse events are triggered by processing itself. Mouse events on this PoC is mainly working with the activity class. Mouse clicks on activities are decision point of the process for the events.

MousePressed() : When mouse clicked the function is invoked. If the clicked item is an activity and mouse cursor is on that activity, then locks the activity, brings it to the front and redraws the scene; if the clicked item is just an activity, then unlocks the activity and redraws the scene.

MouseDragged() : If the is on the mouse, it calls the function for moving the activity with new locations and then calls the funtion of redrawing.

MouseReleased() : If the mouse click released and the current clicked item is an activity, unlocks the activity and redraws the scene.

MouseMoved() : When mouse moves the function is invoked. Whenever mouse moves, all activities are scanned. If mouse is over on an activity on the time, it decides whether the activitiy's overing will be made true or current activitiy's overing will be made false. If the decides is made on the new activity, the scene will be redrawn.

mouseOut() : If mouse is out of the canvas, it calls mouseReleased function.

mouseClicked() :If the clicked item is not activity, the activity is brought to the front.

KORKMAZ, Ozan

Connections of iFlowEdit

This week connection class is written to connect activity objects. Connection class holds three names, which are name, name1 and name2 to keep string values of connection name, starting activity and ending activity. Recheck

function is to initiate the starting and ending activity. In display function of connection class is to draw the connection between two activities. There are four sticky points in each activity therefore, there is created a basic calculation to determine which sticky point will be connected. Then, drawing part is handled between the beginShape and endShape functions. Between these functions, a line is drawn starting from the selected sticky point of activity and ending with the selected sticky point of other activity. To show the direction of the connection, there is also created a direction triangle.

To connect the written activity and connection classes, there is also created StickyPoint that creates a connection point on activities. It holds pair vectors po and no, which are to keep the position and normal values of each sticky point. Normal values provide connection to which way it will be directed and they also provide connection from which way it will be connected to the ending activity.

Connection Class:

name(type:string): Holds connection name.

name1(type:string): Holds starting activity name.

name2(type:string): Holds ending activity name.

zindex(type:int): Holds the index of the created connection in the connectionList.

a1(type: Activity): Holds the starting activity object.

a2(type: Activity): Holds the ending activity object.

Connection(name:String, name1:String, name2:String): Constructor method of the
Connetion class.

recheck(): Initialize the a1 and a2 activities by taking the reference from activities list.

display(): Calculates the best connection points of activities and draw the line between them.

Connection Points of iFlowEdit

StickyPoint Class:

po(type:PVector): Holds position values of a stick point on an activity.

no(type:PVector): Holds normal values of a stick point on an activity.

StickyPoint(nx:float, ny:float): Constructor method of the StickPoint class.

StickyPoint(x:float, y:float, nx:float, ny:float): Constructor method of the StickPoint class.

move(x:float, y:float): Sets the position of a sticky point when the position of an activity changes.

display(): Draws the rectangle representation of a sticky point.