Sketchpad Assignment for HCI

By: Joshua Bainbridge

SHAPES

Assignment Notes: After constructing the state tables the first shape implemented was the line. Once the ability to draw a line was finished all the functionality (change colour, select, move, undo, cut and paste) was injected. The assignment was constructed in this order because of the nature of the sketchpad, all drawing items are derivates of the line item. Meaning, when all functionality is available for the line its can easily be implemented for other drawing items.

Lines

Description: The line component draws a straight line on the canvas. Assuming the user has selected the drawing mode via the button at the top, once a down click occurs point (xi,xj) is set. While holding down the left mouse button point(xj,yj) is continually set and the line is repainted on the canvas. Additionally, with every repaint the old line is removed from the object array list and the updated one is added.

The sketch component is free hand drawing of a line on the canvas. Implemented as an array of line objects with a length of 0. {xi,yi=xj,yj}

Polygon is implemented by connecting creating a new line object with starting and ending points being equal.

Mealy State:	Moveto(x,y) {output}	Dragto(x,y)	Downclick (x,y)	Upclick(x,y)
S:	{ } Next state: S	{drawline (x0.y0,x,y)} Next state: S	{(x0,y0)} ← (x,y)} Next state: S	{ } Next state: S

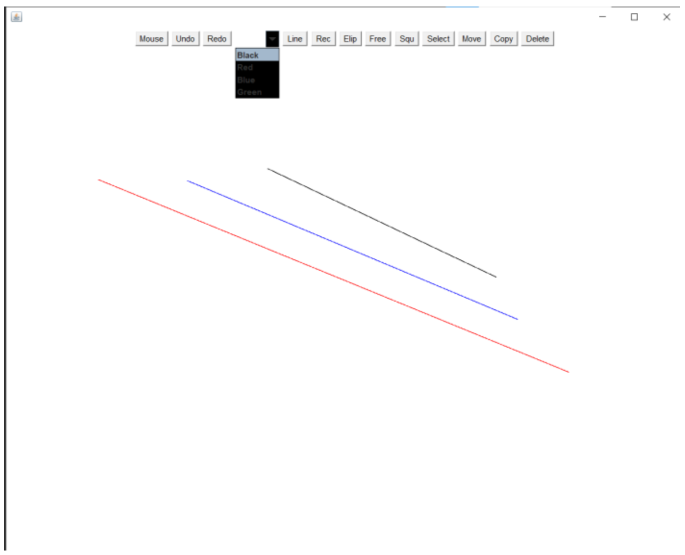
Sketch

Mealy State:	Moveto(x,y) {output}	Dragto(x,y)	Downclick (x,y)	Upclick(x,y)
S:	{ } Next state: S	{drawline (x0.y0,x0,y0)} Next state: S	{(x0,y0)} ← (x,y)} Next state: S	{ } Next state: S

Polygon

Mealy State:	Moveto(x,y) {output}	Dragto(x,y)	Downclick (x,y)	Upclick(x,y)
S1:	{ } Next state: S1	{drawline (x0.y0,x,y)} Next state: S	{(x0,y0)} ← (x,y)} Next state: S	{sx , sy} ← (x,y) Next state: S2
S2:		{drawline (sx.sy,x,y)} Next state: S	{(x0,y0)} ← (x,y)} Next state: S2	{sx , sy} ← (x,y) Next state: S2

Demo Example:



Rectangle

Description: The line component draws a rectangle on the canvas. Assuming the user has selected the drawing mode via the button at the top, once a down click occurs point (xi,xj) is set. While holding down the left mouse button point(xj,yj) is continually set. As point(xj,yj) changes the rectangle is drawn by adding and painting four lines objects: point(xi,yi,xi,yj), point(xi,yi,xj,yi), point(xi,yj,xj,yj), point(xj,yi,xj,yj).

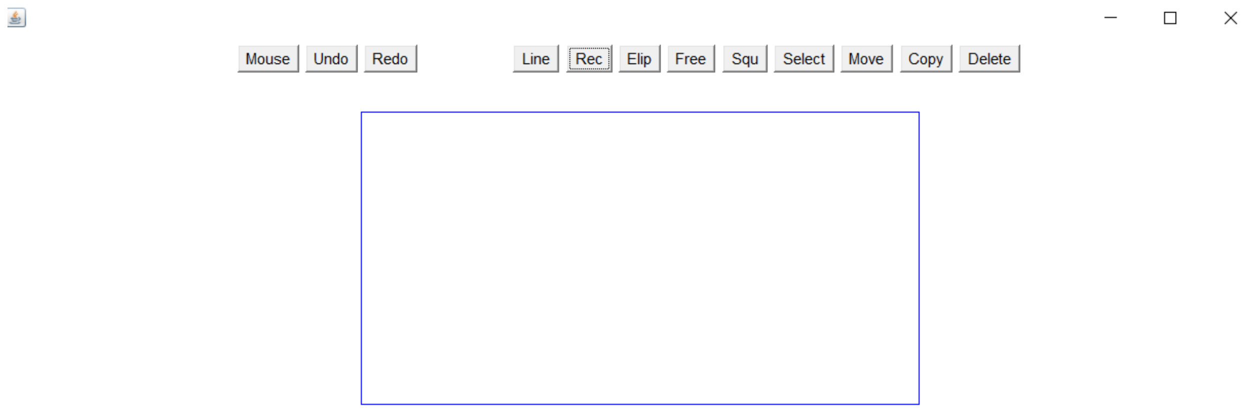
Square used the same process while ensuring side lengths are the same to create a square.

Mealy State:	Moveto(x,y) {output}	Dragto(x,y)	Downclick (x,y)	Upclick(x,y)
S:	{ } Next state: S	{drawline (x0.y0, x0,y), drawline (x0.y0, x,y0) drawline (x.y0, x,y) drawline (x0.y, x,y)} Next state: S	{(x0,y0)} ← (x,y)} Next state: S	{ } Next state: S

Square

Mealy State:	Moveto(x,y) {output}	Dragto(x,y)	Downclick (x,y)	Upclick(x,y)
S:	{ } Next state: S	int avg = (abs(x0-x))+ (abs(y0-y))/2 {drawline (x0.y0, x0,y0+avg), drawline (x0.y0, x0+avg,y0) drawline (x.y0, x0+avg,y+avg) drawline (x0.y0+avg, x0+avg,y0+avg)} Next state: S	{(x0,y0)} ← (x,y)} Next state: S	{ } Next state: S

Demo Example:



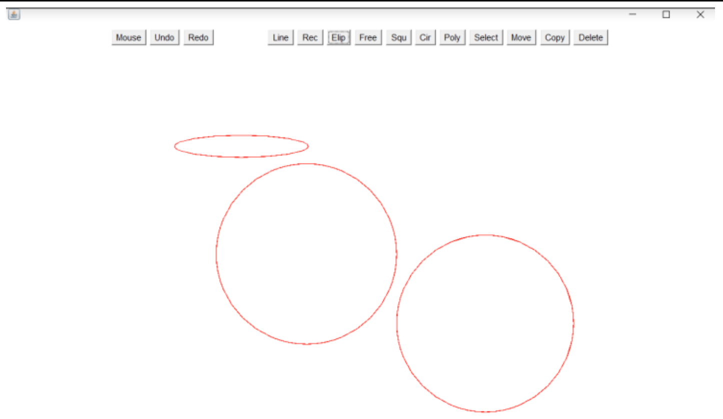
Description: The line component draws a ellipse on the canvas. Assuming the user has selected the drawing mode via the button at the top, once a down click occurs point (xi,xj) is set. While holding down the left mouse button point(xj,yj) is continually set. As point(xj,yj) changes the ellipse is drawn. {drawArc(xi,yi,xi-xj,yi-yj,30)}

Circle used the same process while ensuring the radius is even all the way around using draw oval.

Mealy State:	Moveto(x,y) {output}	Dragto(x,y)	Downclick (x,y)	Upclick(x,y)
S:	{ } Next state: S	{ drawArc (x0.y0, |x-x0|,|y-y0|,angle), drawArc (x0.y0, |x-x0|,(-1)*|y-y0|,angle }Next state: S	{(x0,y0)} 🡐 (x,y)} Next state: S	{ } Next state: S

Circle

Mealy State:	Moveto(x,y) {output}	Dragto(x,y)	Downclick (x,y)	Upclick(x,y)
S:	{ } Next state: S	int avg = (abs(x0-x))+ (abs(y0-y))/2 { drawArc (x0.y0, |x-x0|,|y-y0|,angle), drawArc (x0.y0, |x-x0|,(-1)*|y-y0|,angle }Next state: S	{(x0,y0)} 🡐 (x,y)} Next state: S	{ } Next state: S



Demo example:

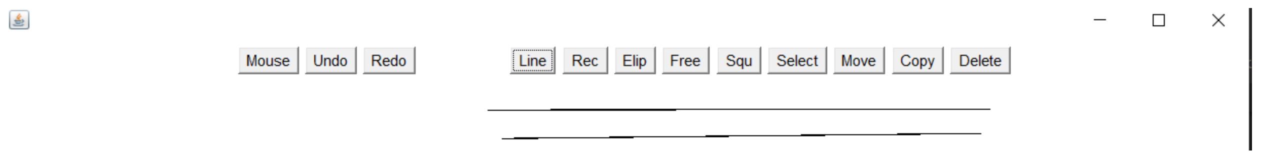
SELECTION FUNCTIONS

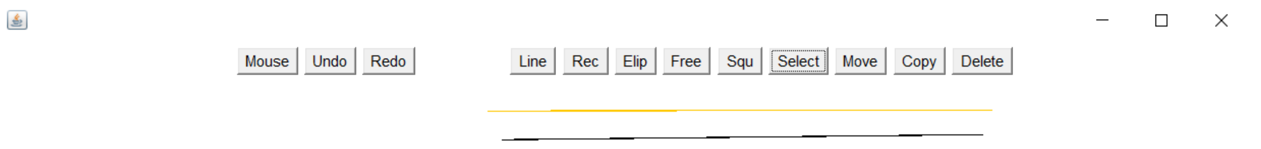
<u>Select</u>

Mealy State:	Moveto(x,y) {output}	Dragto(x,y)	Downclick (x,y)	Upclick(x,y)

S1:	{ }
	Next state: S1
		{}
		Next state: S1
			{(x0,y0)} 🡐 (x,y)}
			Ipair(x0,y0)
			If d <= dim {ob.st toggle}
			repaint()
			Next state: S1
				{ }
				Next state: S1

Demo Example:

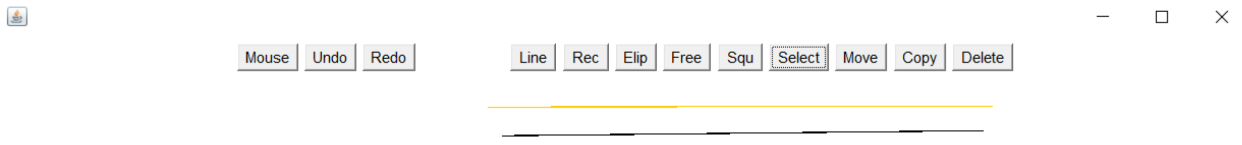


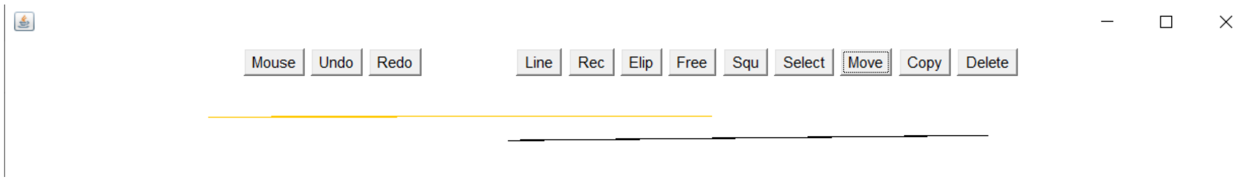


Move

Mealy State:	Moveto(x,y) {output}	Dragto(x,y)	Downclick (x,y)	Upclick(x,y)
S1:	{ } Next state: S1	Int movx = x0-ob.xi Int movy = y0-ob.yi {ob.xi,ob.yi} 🡐 (x0,y0) {ob.xj,ob.yj} 🡐 (x0+movx,y0+movy) repaint() Next state: S1	{(x0,y0)} 🡐 (x,y)} Next state: S1	{ } Next state: S1

Demo example:

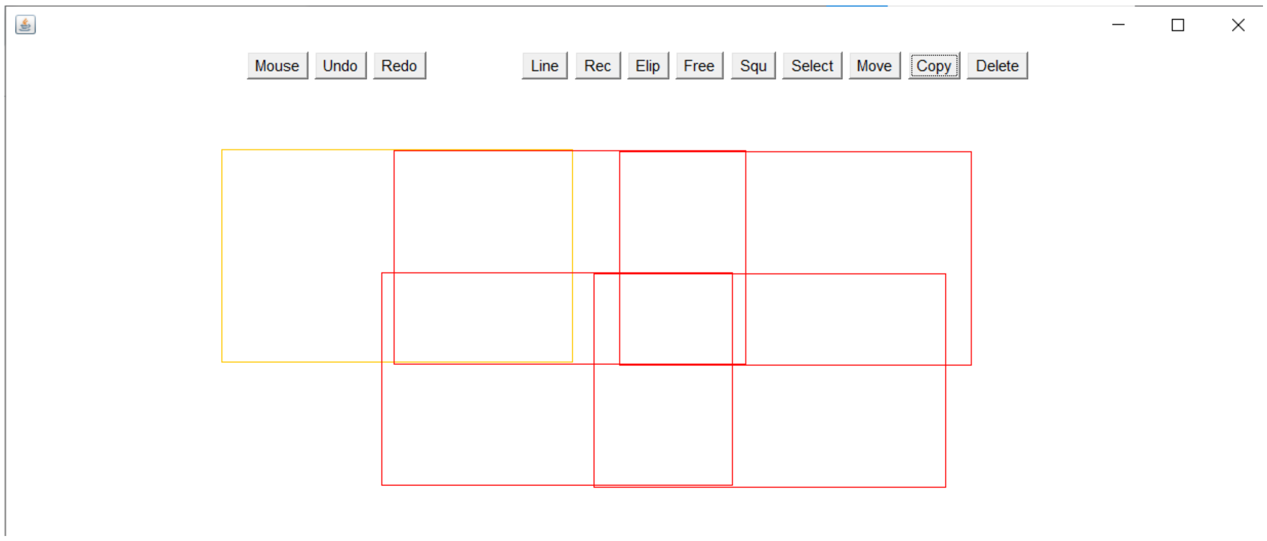




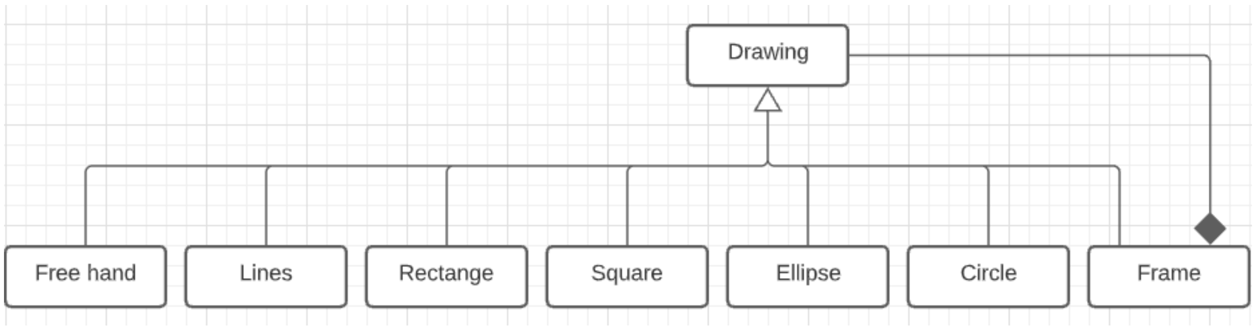
Paste

Mealy State:	Moveto(x,y) {output}	Dragto(x,y)	Downclick (x,y)	Upclick(x,y)
S:	{ } Next state: S	{} Next state: S	{(x0,y0)} 🡐 (x,y)} new paste = ob.selected { paste.xi, paste.yi} 🡐 (x0,y0) Next state: S	{ } Next state: S

Demo example



Object Diagram



Design Patterns

While constructing the Sketchpad programs one a design pattern was noticed. This pattern occurred when adding new drawing items to the program. Each was a simple permutation of the line item. Every drawing object is treated in the same way. This design pattern is known as a composite design pattern.