CMSC427
Interactive programs in Processing:
Polyline editor

Interactive programming

- Example: PaperSnowFlake
  - http://rectangleworld.com/PaperSnowflake/
- Big ideas today
  - Event driven programming
  - Object list
  - Model View Controller (MVC) architecture
- Polyline editor in Processing

Processing.org – generative model

Circle | Processing 3.3.5

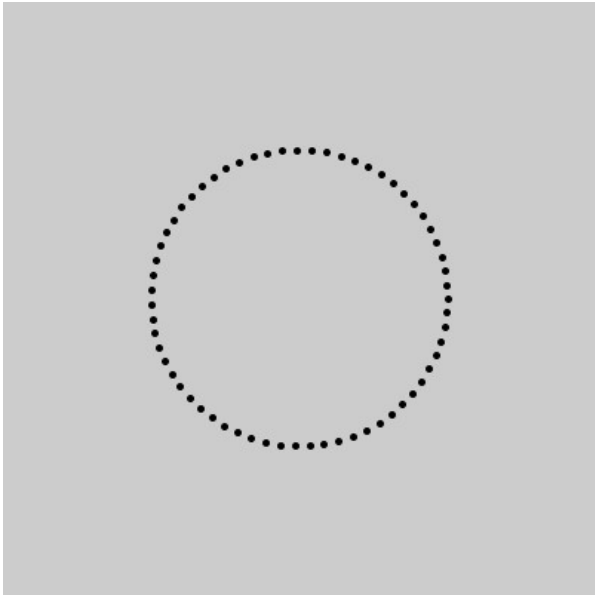
Java

Circle

```
1  // Parametric circle
2
3
4  size(400,400);
5  float r = 100;
6  strokeWeight(5);
7
8  for (float t = 0; t < 2*PI; t += 0.1) {
9    float x = width/2  + r*cos(t);
10   float y = height/2 + r*sin(t);
11   point(x,y);
12 }
13
14 save("curve.jpg");
```

Done saving.

Console Errors



Event driven program in Processing

Static sketch (runs once)

```
size(200,200);  // width, height

background(100,100,255);

fill(255,0,0);
stroke(0,0,255);

ellipse(width/2, height/2, 100, 100);

save("pic.jpg");
```

Event driven program in Processing

Static sketch (runs once)

```
size(200,200);

background(100,100,255);

fill(255,0,0);
stroke(0,0,255);

ellipse(width/2, height/2, 100, 100);

save("pic.jpg");
```

Dynamic sketch (runs forever)

```
void setup() {
  size(200,200);
}

void draw() {

  background(100,100,255);
  fill(255,0,0);
  stroke(0,0,255);

  ellipse(mouseX,mouseY, 100, 100);
 }
```

Details of dynamic sketch

```
void setup() {
  size(200,200);
}

void draw() {

  background(100,100,255);
  fill(255,0,0);
  stroke(0,0,255);

  ellipse(mouseX,mouseY, 100, 100);
}
```

On start event
Once when program starts

Details of dynamic sketch

```
void setup() {
  size(200,200);
}
```

On draw event
Every 1/30 second

Timing set with frameRate

```
void draw() {

  background(100,100,255);
  fill(255,0,0);
  stroke(0,0,255);

  ellipse(mouseX,mouseY, 100, 100);
 }
```

mouse Events

```
void setup() {
  size(200,200);
}

void draw() { }

void mousePressed() {
  rect(mouseX,mouseY,20,20);
}

void mouseDragged() {
  ellipse(mouseX,mouseY,
10,10);
}

void mouseReleased() {
  rect(mouseX,mouseY,20,20);
}
```

On mousePressed
Once when mouse button
is first pressed

On mouseDragged
As long as mouse button is pressed
And whenever mouse moves

On mouseReleased
Once when mouse button is released

Keyboard Events

```
void setup() {
  size(200,200);
}

void draw() { }
```

```
void keyPressed() {
  rect(mouseX,mouseY,20,20);
}
```

On keyPressed
Once when key is pressed

Putting it together: drawing program (BasicDraw)

```
color c;

void setup() {
  size(400,400);
  noStroke();
}

void draw() { }

void mouseDragged() {
   fill(c);
   ellipse(mouseX,mouseY,10,10);
}

void keyPressed() {
        if (key == 'r')  c = color(255,0,0);   // pen red
  else if (key == 'b') c = color(0,0,255);   // pen blue
  else if (key == 'b')  background(255,255,255);   // clear
  else if (key == 's')   save("pic.jpg");  //  save
}
```

Summary of basic Processing events and handlers

- On program start — *setup()*
- On frame timer — *draw()*
- On mousePressed — *mousePressed()*
- On mouseDragged — *mouseDragged()*
- On mouseReleased — *mouseReleased()*
- On keyPressed — *keyPressed()*

- System variables
  - Position position — mouseX,mouseY
  - Last key pressed — key

Event driven programming: general concepts

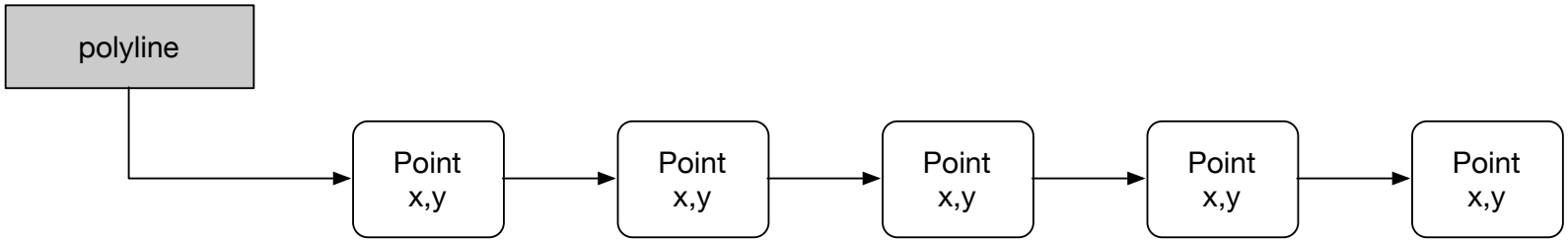
- Event
  - *Input action to program from user, or from operation system*
- Event loop
  - *while(true) process Event*
  - *Hidden in Processing*
- Event handlers (or callbacks)
  - *Method called when an event happens*
- Event queue
  - *Events in order of occurrence waiting for handling*
  - *Filled by OS window manager, emptied by program*

- *More in Java later*

Polyline editor

```
Polyline polyline;

void setup() {
  size(400,400);
  polyline = new Polyline();
}

void draw() {
  background(255);
  noFill();
  polyline.draw();
}

void keyPressed() {
  if (key == )
    polyline.close();
  else if (key == 'o')
    polyline.open();
}

void mousePressed() {
  if (mouseButton == LEFT)
      polyline.add(mouseX,mouseY);
  else if (mouseButton == RIGHT)
      polyline.pick(mouseX,mouseY);
  }

void mouseDragged() {
  polyline.pickUpdate(mouseX,mouseY);
}

void mouseReleased() {
  polyline.pickRelease();
}
```

Polyline as object list

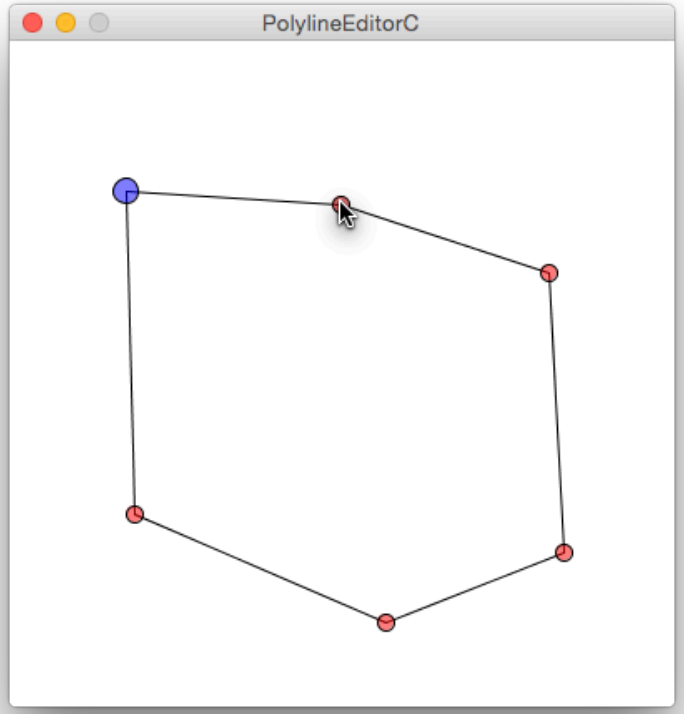


- *Operations*
  - Add object (point)
  - Change list property (open/close)
  - Display
    - From first to last in list. Later objects display in front.
  - Pick item from list

- Later in class: Scene graph with 3D objects

Pick operation

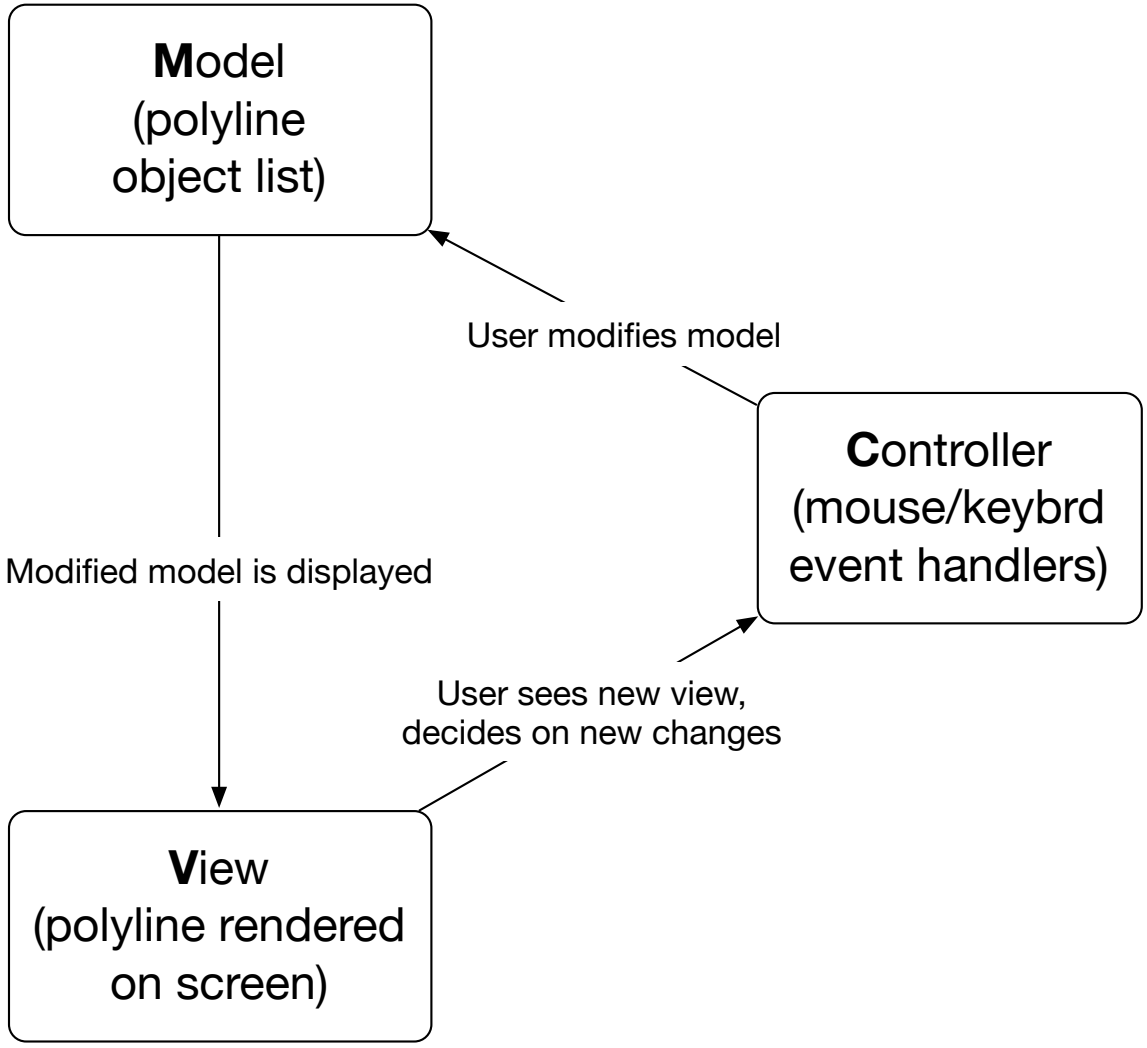
- Pick object on list for individual manipulation



- Search list for closest object to mouse position, return ptr to object

- Sequential search:
  should pick first object matched, or last?

Model View Controller (MVC) software architecture



https://en.wikipedia.org/wiki/Model–view–controller

What you should know after today

1. Mechanisms and terminology of event driven programming (event, event loop, event handler, event queue)
2. Basic events and handlers in Processing.
3. How to look up Processing commands used in class.
4. How to run and modify the PolylineEditor program.
5. Concept of object list and basic operations (add, display, pick)
6. Concepts of Model-View-Controller software architecture

Today's resources

- PaperSnowFlake
  - http://rectangleworld.com/PaperSnowflake/
- Processing
  - https://processing.org
  - Resource for quick program "sketches", concepts
  - Sketches: StaticSketch.pde, DynamicSketch.pde, BasicDraw.pde, ProcessingEvents.pde, PolylineEditor.pde+Polyline.pde

Additional notes on physical and logical input devices

- Multiple types of real physical input devices
  - Mouse, keyboard, gamepad, mocap, tablet pen, spaceball, touch screen, more
- Can generalize with logical input devices
  - *Locator* produces (x,y) position on the screen
  - *Valuator* produces range of values x
  - *Stroke* produces polyline as sequence p1, p2, p3, …, pn
  - *Camera* produces 2d image
  - *Keyboard* produces character or string
- Mouse can be used for many logical devices