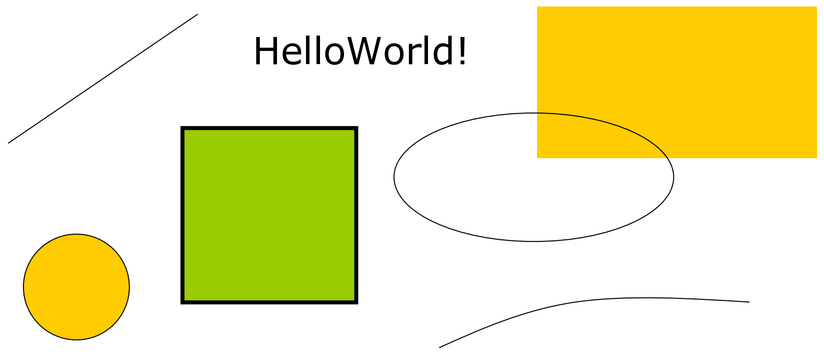
Graphics in Java

hussein suleman
uct cs 116 2005

What are graphics?

- Graphic primitives: lines, squares, rectangles, circles, ellipses, arcs, polygons, text, etc.

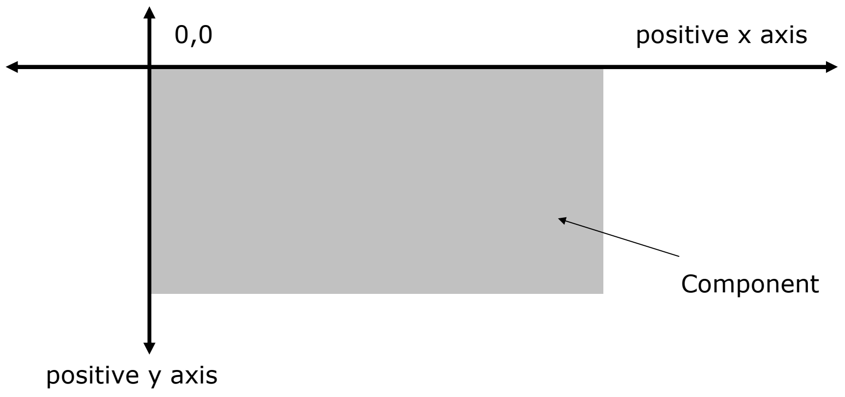


Built-in 2D Graphics Frameworks

- AWT Graphics (Slack)
  - drawRect (40, 40, 100, 30)
- Swing Graphics
  - Method-oriented
    - drawRect (40, 40, 100, 30)
  - "Object"-oriented
    - draw (new Rectangle2D.Float (40, 40, 100, 30))
      - draws a rectangle
    - fill (new Rectangle2D.Float (40, 40, 100, 30))
      - draws a filled rectangle

Component Coordinate System

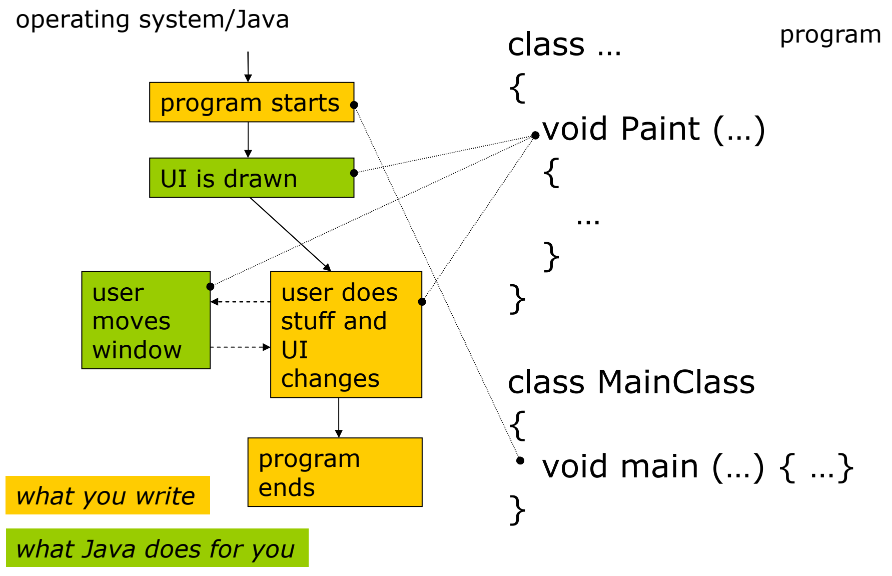
- All graphics must be drawn as part of or on a Swing/AWT component.
- All coordinates are then relative to that component.



Event-driven Graphics 1/2

- Graphics should NOT be drawn in the main program.
  - When a program is minimised and maximised, the graphics will not be redrawn.
- Instead, override the pre-defined `paint` or `paintComponent` method to specify how Java/OS should redraw the component whenever necessary.
  - The OS will then redraw the component whenever it is moved, resized, maximised, uncovered, etc.

Event-driven Graphics 2/2



```
class …
{
    void Paint (…)
    {
        …
    }
}

class MainClass
{
    void main (…) { …}
}
```

Example 1: Painting

```
class DrawPanel extends JPanel
{
   // override the painting routine of the component
   protected void paintComponent ( Graphics gr )
   {
      // first call the superclass's method
      super.paintComponent (gr);

      // then get a "handle" to the window for drawing
      Graphics2D canvas = (Graphics2D)gr;

      // issue a series of drawing commands
      canvas.draw (new Rectangle2D.Float (100, 100, 400, 400));
      canvas.draw (new Line2D.Float (159, 159, 441, 441));
      canvas.draw (new Line2D.Float (159, 441, 441, 159));
      canvas.draw (new Ellipse2D.Float (100, 100, 400, 400));
      canvas.drawString ("Hello World", 280, 520);
   }
}
```

Painting Example Swing 1/2

```
class TestFrame extends JFrame implements ActionListener
{
  public DrawPanel dp;

   public TestFrame ()
   {
      super ("Example One");
      setSize (600, 600);
      setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);

      JPanel pane = new JPanel();
      pane.setLayout (new BorderLayout ());
      JPanel pane2 = new JPanel();
      pane2.setLayout (new FlowLayout ());

      JButton exit = new JButton ("Exit");
      exit.addActionListener (this);
```

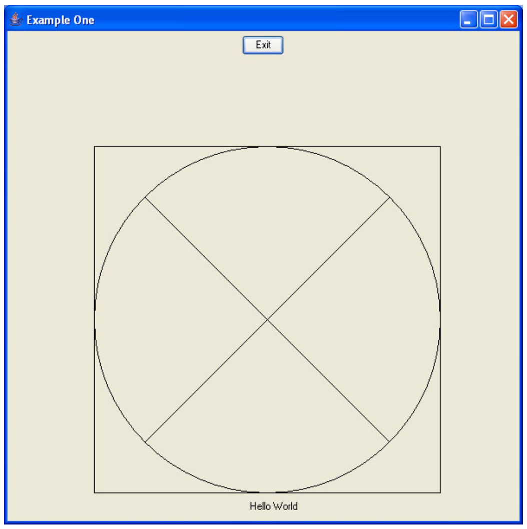
Painting Example Swing 1/2

```
      dp = new DrawPanel ();

      pane2.add (exit);
      pane.add ("North", pane2);
      pane.add ("Center", dp);

      setContentPane (pane);
      setVisible (true);
   }

   public void actionPerformed ( ActionEvent e )
   {
       System.exit(0);
   }
}
```

Painting Example Main Class

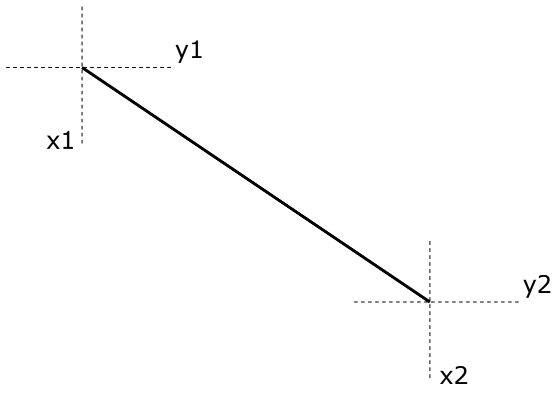
```
public class example1
{
   public static void main ( String [] arguments )
   {
      // set user interface style
      try {
         UIManager.setLookAndFeel
   (UIManager.getSystemLookAndFeelClassName ());
      } catch (Exception e) {};

      new TestFrame();
   }
}
```

Paint Example Output



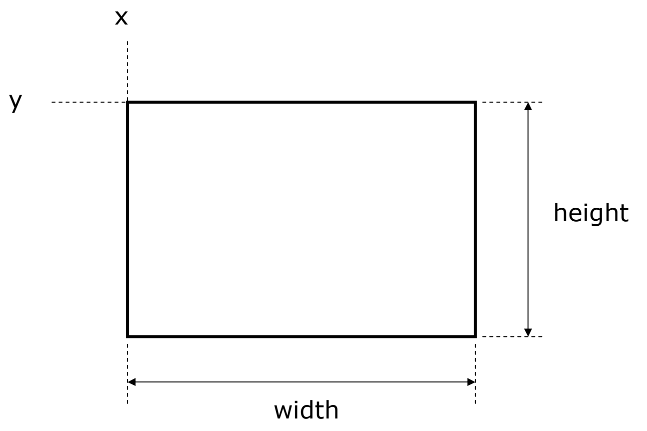
Graphics Primitives: Line

- `new Line2D.Float (x1, y1, x2, y2)`



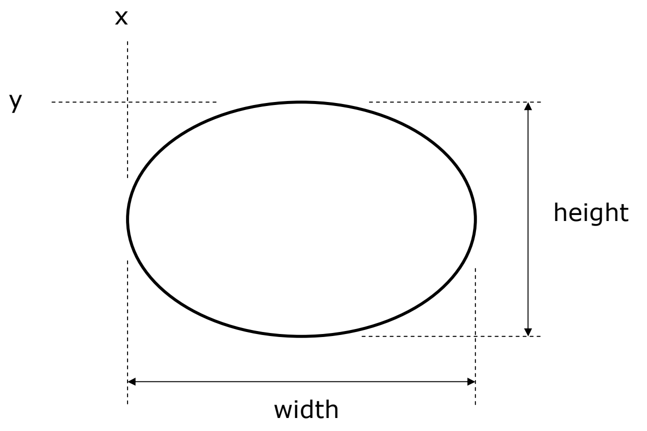
Graphics Primitives: Rectangle

- `new Rectangle2D.Float (x, y, width, height)`



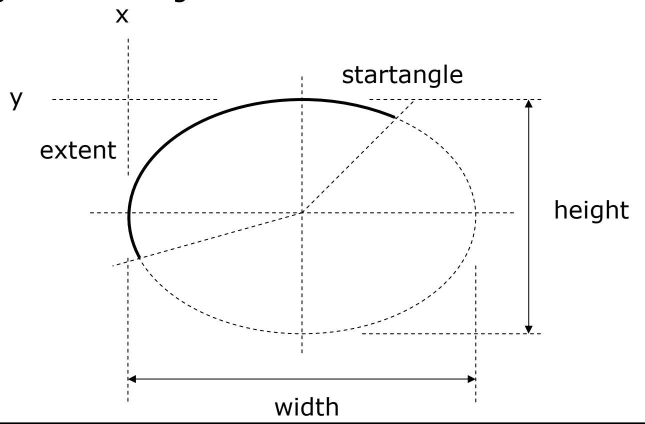
Graphics Primitives: Ellipse

- `new Ellipse2D.Float (x, y, width, height)`



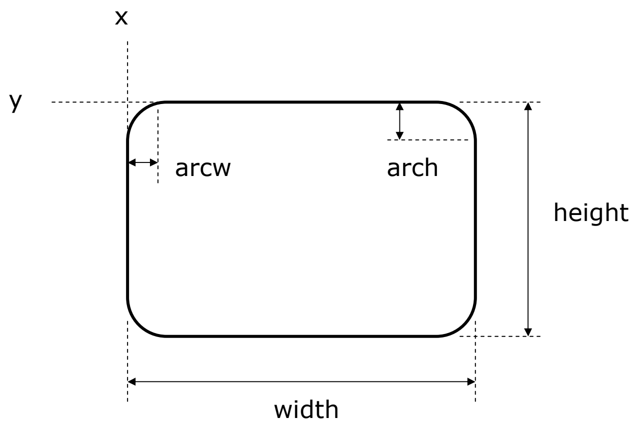
Graphics Primitives: Arc

- `new Arc2D.Float (x, y, width, height, startangle, extent, type)`
  - type is in {Arc2D.PIE, Arc2D.CHORD, Arc2D.OPEN}
  - angles are in degrees



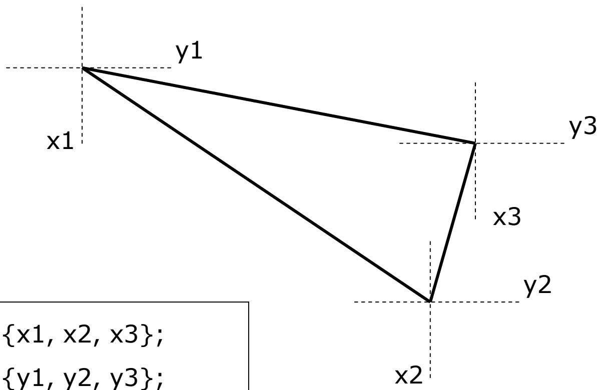
Graphics Primitives: RoundedRect

- `new RoundRectangle2D.Float (x, y, width, height, arcw, arch)`



Graphics Primitives: Polygon

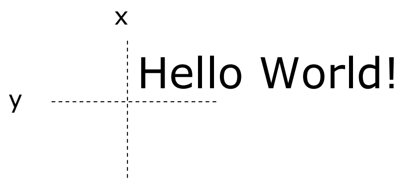
- `new Polygon ( int [] xpoints, int [] ypoints, int npoints)`



```
int x [] = {x1, x2, x3};
int y [] = {y1, y2, y3};
p = new Polygon (x, y, 3);
```

Graphics Primitives: Text

- `drawString (text_string, x, y)`



Line Attributes

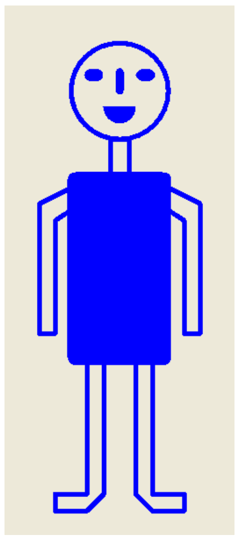
- `setColor ( Color c )`
  - e.g., Color.blue, Color.red, Color.green
  - sets the colour to be used for all subsequent graphics.
- `setStroke (new BasicStroke (weight, cap, join))`
  - cap is in {BasicStroke.CAP_ROUND/CAP_BUTT/CAP-SQUARE}
  - join is in {BasicStroke.JOIN_ROUND/JOIN_MITER/JOIN_BEVEL}
  - sets the type of line and the way one line joins another at corners.

Text Attributes

- `setFont ( Font f )`
  - sets the font to be used for all subsequent text drawn.
  - `new Font ( name, style, weight )`
    - e.g., `new Font ("TimesRoman", Font.PLAIN, 12)`
- Font names also include `"Serif"` and `"SansSerif"`.
- Font styles also include `BOLD` and `ITALIC`.

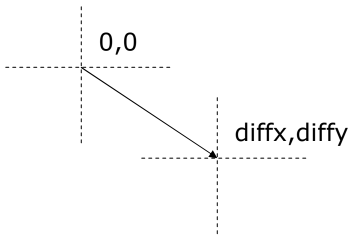
Problem

- Draw the following figure using Java's graphics primitives:



Coordinate Transformations

- `scale (scalex, scalex)`
  - scales all subsequent coordinates in graphics primitive operations by scalex in x direction and scaley in y direction.
- `translate (diffx, diffy)`
  - moves the origin of the axes to the location specified.



Example 2: Mouse Interaction

```
class DrawPanel extends JPanel
   implements ActionListener, MouseMotionListener, MouseListener
{
   JButton zoomin, zoomout;
   float shiftx, shifty, scale;
   float startx, starty;
   int boy;

   // set up panel
   DrawPanel ( JButton zin, JButton zout )
   {
        zoomin = zin;
        zoomout = zout;
        reset();
   }

    // set default values in variables
   public void reset ()
   {
        scale = 1.0f;
        shiftx = 0;
        shifty = 0;
        boy = 100;
   }
```

Mouse Interaction: Painting

```
   public void drawBoy ( Graphics2D canvas, int x, int y )
   {
     canvas.translate (x, y);
     canvas.setColor (Color.blue);
     canvas.setStroke (new BasicStroke (3.0f, BasicStroke.CAP_ROUND, BasicStroke.JOIN_ROUND));
     canvas.draw (new Ellipse2D.Float (20, 0, 60, 60));
     canvas.fill (new Ellipse2D.Float (28, 16, 12, 8));
     canvas.fill (new Ellipse2D.Float (60, 16, 12, 8));
     canvas.setStroke (new BasicStroke (5.0f, BasicStroke.CAP_ROUND, BasicStroke.JOIN_ROUND));
     canvas.draw (new Line2D.Float (50, 18, 50, 30));
     canvas.setStroke (new BasicStroke (3.0f, BasicStroke.CAP_ROUND, BasicStroke.JOIN_ROUND));
     canvas.fill (new Arc2D.Float (40, 30, 20, 20, 0, -180, Arc2D.PIE));
     canvas.draw (new Rectangle2D.Float (44, 60, 12, 40));
     canvas.fill (new RoundRectangle2D.Float (18, 80, 64, 120, 10, 10));
     int hand1x[] = {20,0,0,10,10,20}; int hand1y[] = {90,100,180,180,110,104};
     canvas.draw (new Polygon (hand1x,hand1y,hand1x.length));
     int hand2x[] = {80,100,100,90,90,80}; int hand2y[] = {90,100,180,180,110,104};
     canvas.draw (new Polygon (hand2x,hand2y,hand2x.length));
     int leg1x[] = {40,40,30,10,10,30,30}; int leg1y[] = {198,280,290,290,280,280,198};
     canvas.draw (new Polygon (leg1x,leg1y,leg1x.length));
     int leg2x[] = {60,60,70,90,90,70,70}; int leg2y[] = {198,280,290,290,280,280,198};
     canvas.draw (new Polygon (leg2x,leg2y,leg2x.length));
     canvas.translate (-x, -y);
   }

  protected void paintComponent ( Graphics gr )
  {
     super.paintComponent (gr);
     Graphics2D canvas = (Graphics2D)gr;
     canvas.translate (shiftx, shifty);
     canvas.scale (scale, scale);
     drawBoy (canvas, boy, 100);
  }
```

Mouse Interaction: Actions

```
   public void actionPerformed ( ActionEvent e )
   {
      if (e.getSource() == zoomin)
         scale *= 1.20f;
      else if (e.getSource() == zoomout)
         scale /= 1.20f;
      else
         reset();
      repaint();
   }

   public void mouseDragged ( MouseEvent m )
   {
    shiftx += (m.getX() - startx);
    shifty += (m.getY() - starty);
    startx = m.getX();
    starty = m.getY();
    repaint();
   }
   public void mouseMoved ( MouseEvent m ) {}
   public void mouseClicked ( MouseEvent m ) {}
   public void mouseEntered ( MouseEvent m ) {}
   public void mouseExited ( MouseEvent m ) {}
   public void mousePressed ( MouseEvent m )
   {
    startx = m.getX();
    starty = m.getY();
   }
   public void mouseReleased ( MouseEvent m ) {}
}
```

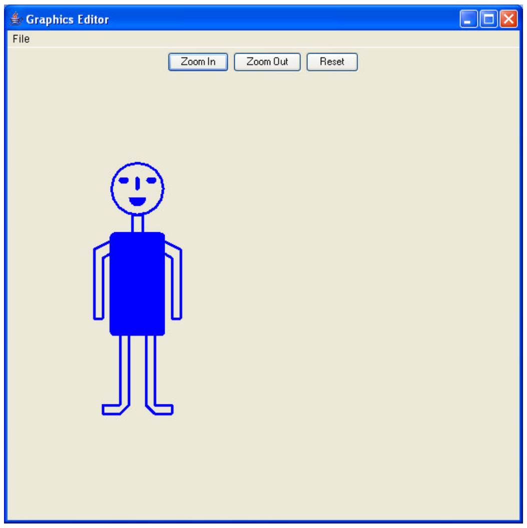
Mouse Interaction: Frame 1/2

```
class TestFrame extends JFrame implements ActionListener
{
   public DrawPanel dp;

   public TestFrame ()
   {
      super ("Graphics Editor");
      setSize (600, 600);
      setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);

      JPanel pane = new JPanel();
      pane.setLayout (new BorderLayout ());
      JPanel pane2 = new JPanel();
      pane2.setLayout (new FlowLayout ());

      JButton zin = new JButton ("Zoom In");
      JButton zout = new JButton ("Zoom Out");
      JButton reset = new JButton ("Reset");
      dp = new DrawPanel (zin, zout);

      pane2.add (zin);
      pane2.add (zout);
      pane2.add (reset);

      pane.add ("North", pane2);
      pane.add ("Center", dp);
```

Mouse Interaction: Frame 2/2

```
      zin.addActionListener (dp);
      zout.addActionListener (dp);
      reset.addActionListener (dp);
      dp.addMouseMotionListener (dp);
      dp.addMouseListener (dp);

      MenuBar mb = new MenuBar ();
      Menu file = new Menu ("File");
      MenuItem exit = new MenuItem ("Exit");
      exit.addActionListener (this);
      file.add (exit);
      mb.add (file);
      setMenuBar (mb);

      setContentPane (pane);
      setVisible (true);
   }

   public void actionPerformed ( ActionEvent e )
   {
   System.exit(0);
   }
}
```

Mouse Interaction: Output



MouseMotionListener Interface

- `mouseMoved` is invoked when the mouse is moved and no buttons are being pressed.
- `mouseDragged` is invoked when the mouse is moved while one or more buttons are held down.
  - mousePressed
    - ->mouseDragged
    - ->mouseReleased
- Parameter same as for MouseListener.

Scrolling the Canvas

- Store position of canvas as a set of offsets that must be added to all coordinates before drawing.
- When a mouse button is pressed, store the position of the mouse.
- When mouse is dragged, calculate difference between current position and stored position and add this to the offsets.
- Use `translate` to offset canvas prior to drawing.

Zooming In and Out

- Store zoom state as a set of scale multipliers in each direction.
- Before drawing any graphics, multiply the coordinates by the multipliers.
- When zooming in/out, multiply the multipliers by factors greater than or less than 1.
- Use `scale` to scale canvas coordinates prior to drawing.

The `repaint` method

- `repaint` can be called explicitly after any changes to the user interface.
- `repaint` causes Java to invalidate the region i.e., make it seem in need of repainting.
  - once a region has been invalidated, Java will call the `paint` function of the component, when it is safe to do so.

Example 3: Animation

- Create multiple images on a single canvas, with parameters to indicate relative position.
- Each time a button is clicked (or some trigger is activated), move the images to resemble animation by changing the parameters used by paintComponent to position graphics.
- Non-interactive animation typically uses a separate “thread” (like a program) to control the animation.

Animation: Painting

```
protected void paintComponent ( Graphics gr )
  {
     super.paintComponent (gr);
     Graphics2D canvas = (Graphics2D)gr;
     canvas.translate (shiftx, shifty);
     canvas.scale (scale, scale);
     drawBoy (canvas, boy, 100, walk);
     drawGirl (canvas, girl, 100, walk);
     if ((girl - boy) == 100)
     {
           drawHeart (canvas, girl, 50);
     }
  }
```

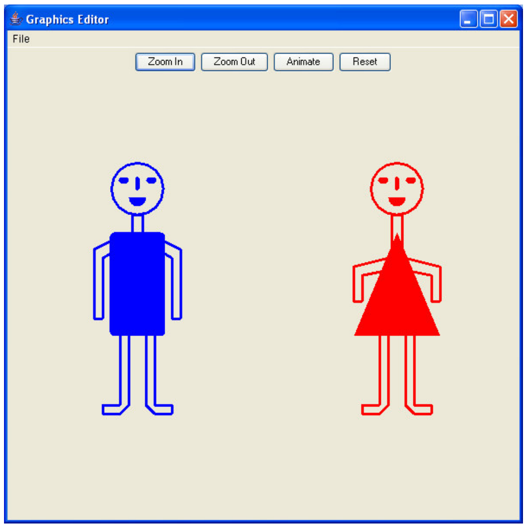
Animation: Button Processing

```
public void actionPerformed ( ActionEvent e )
  {
       if (e.getSource() == zoomin)
       scale *= 1.20f;
     else if (e.getSource() == zoomout)
        scale /= 1.20f;
     else if (e.getSource() == animate)
     {
       if (boy != 200)
       {
            boy += 5;
                girl -= 5;
                walk = 1-walk;
         }
     }
     else
        reset();
     repaint();
  }
```

Animation: Mouse Actions

```
public void mouseDragged ( MouseEvent m )
  {
    shiftx += (m.getX() - startx);
    shifty += (m.getY() - starty);
    startx = m.getX();
    starty = m.getY();
    repaint();
  }
  public void mousePressed ( MouseEvent m )
  {
    startx = m.getX();
    starty = m.getY();
  }
```

Animation: Output


Graphics Editor
File
Zoom In
Zoom Out
Animate
Reset