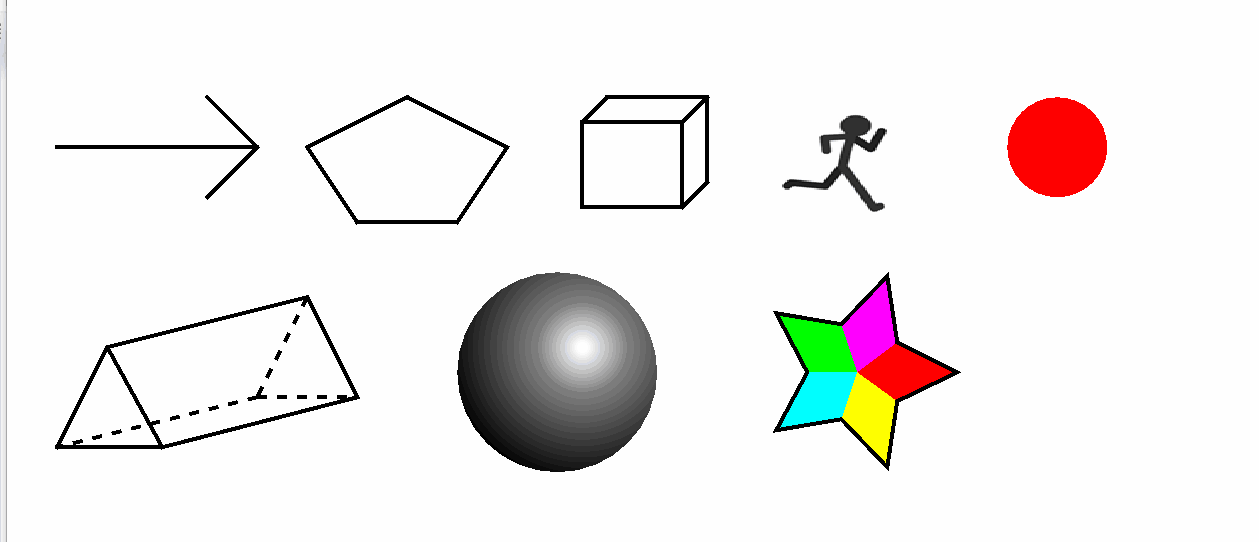
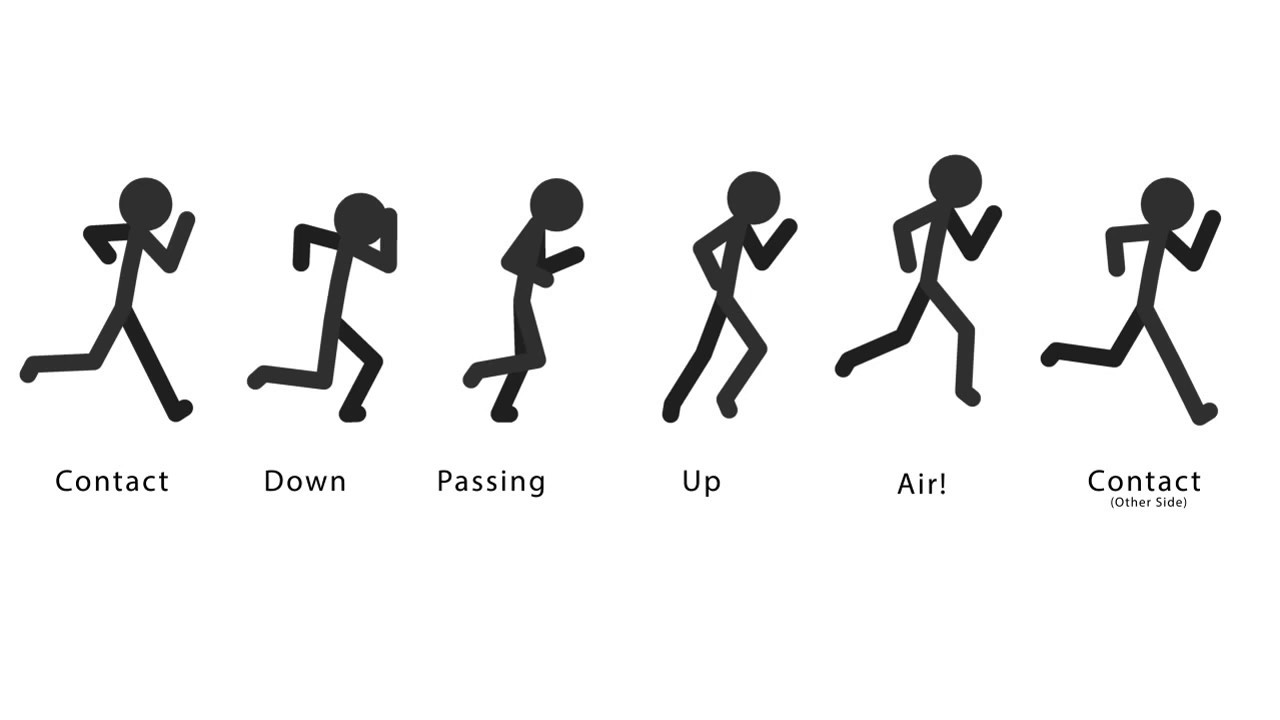
# Graphics Lab: Drawing and Animation

  
(forgive some artifacting on the color gradients due to the gif format)

In this lab, you’ll expand on the concepts demonstrated in the lecture\notes by drawing and animating basic shapes and figures. Your task is to create a graphical scene featuring the following items, all drawn on the same screen. You may use the base [javax.swing.JFrame](https://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html) to draw to, or you may place a [javax.swing.JPanel](https://docs.oracle.com/javase/8/docs/api/?javax/swing/JPanel.html) on the JFrame and draw to that. Refer to the examples in eCampus for setting up the scene. The figures you will be adding are:

1. **Arrow**
   1. Draw an arrow using basic line and polygon shapes. The exact style/shape is up to you but simple lines are just fine.
   2. Challenge: Make an Arrow Object that you can give a [java.awt.Point](https://docs.oracle.com/javase/8/docs/api/index.html?java/awt/Point.html) to and have the Arrow point to that point. Optionally add an additional Point or angle to give a direction and size or other characteristics of the arrow.
2. **Pentagon**
   1. Create a five-sided figure. It doesn’t need to have equal angles or side lengths, but aim for a clear pentagon shape.
   2. We don’t need a class/object here, since the [java.awt.Polygon](https://docs.oracle.com/javase/8/docs/api/index.html?java/awt/Polygon.html) class (which holds points and can draw a figure) works for this. You can use the Polygon class or draw simple lines to represent the pentagon. I would however separate this out into a static method called drawPentagon or something.
3. **Cube**
   1. Represent a 3D cube using lines and perspective. Experiment with depth perception by varying the line weights or colors.
   2. You can make a Cube class which defines the attributes like size (width, height, and depth) and colors, potentially of each face of the cube.
4. **Triangular Prism**
   1. Construct a 3D triangular prism (a triangle extended into three dimensions).
   2. Use line weight, stroke style, or different shades to distinguish the "back" edges from the "front" ones. You would need to use [java.awt.Graphics2D](https://docs.oracle.com/javase/8/docs/api/index.html?java/awt/Graphics2D.html) and set the stroke on the lines to be able to affect how they are drawn.
5. **Sphere**
   1. Go beyond drawing a simple circle. Represent a sphere by adding shading, gradients, or other techniques to create a sense of depth and roundness.
6. **5-pointed Star** 
   1. Each arm of the star should be a different color.   
      (Notice that the center, plus 2 inner points of the star and 1 outer point of the star crates a quadrilateral that you can save as a Polygon and fill with a color of your choosing).

Objectives 7 through 9 will require a loop to be constructed and the scene to be continually redrawn. It is okay if you redraw the entire scene or just the affected portion. You cannot use a loop inside of the paint method. Instead, you will have to create a loop elsewhere (in main after creating and showing the frame is fine). In the loop you will add a delay (though [Thread.sleep](https://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html#sleep(long))) and you will call repaint if using paint (JFrame) or paintComponent (JPanel).  
Timers are not necessary as everything can be done just fine with a loop that updates each figure once per cycle. Refer to the examples in eCampus that demonstrate how to loop the application.

1. **Progress Circle**
   1. The circle will have a red background and will show a green amount that increases (filling the red circle). We can do this by using the fillArc method in Graphics.
   2. Once the green has filled the circle it will clear (set the filled portion back to 0) and will restart.
2. **Sprite Animation**
   1. Display a simple sprite image that shows 6 frames of a stick figure running animation.  
      You can read the image using [ImageIO.read(File)](https://docs.oracle.com/javase/8/docs/api/index.html?javax/imageio/ImageIO.html)
   2. Break the sprite image into individual frames and draw them one at a time each loop. This should give the illusion of animation.  
      To separate the image into its individual frames you can use [BufferedImage.getSubimage()](https://docs.oracle.com/javase/8/docs/api/index.html?javax/imageio/ImageIO.html)
   3. Ensure the animation repeats indefinitely by cycling back the first image once you reach the end.  
        
      (The image is included as an attachment)
3. **Circle at clicks**
   1. We will add a [MouseListener](https://docs.oracle.com/javase/7/docs/api/java/awt/event/MouseListener.html) (or adaptor) which will listen for a click on our JFrame (and/or JPanel). When a click occurs an object should be created at the point. Note that the [MouseEvent](https://docs.oracle.com/javase/7/docs/api/java/awt/event/MouseEvent.html) given by the MouseListener has a [getPoint()](https://docs.oracle.com/javase/7/docs/api/java/awt/event/MouseEvent.html#getPoint()) method.
   2. At the click point our ClickObject will draw a circle that gets larger and fades every loop. The fading can be done by adding transparency or alpha to our color we choose:  
      new Color(1.0f,0.0f,0.0f,0.5f);  
      This version takes floats and the last value is a percentage of transparency. The first the are the same Red, Green, Blue values as a regular Color but again, they will now be in a decimal percent from 0 to 1.0. So our exampled color above is a half-faded red.
   3. Each of these ClickObjects should be added to a list and updated each loop.
   4. Once the figure fades completely it can be deleted from the list

## Submission Requirements

All the shapes/figures must be drawn on the same scene. Include all the java files necessary to run your code (as .java, no .class files). If you used any textfiles or additional resources be sure to clear those with the instructor beforehand.

## Grading

* The scene working and looping/updating the image correctly and the proper use of objects where appropriate is 10%
* Each of the 8 figures is 10%
* The mouse click circles is the final 10%
* Include anything you think warrants additional credit