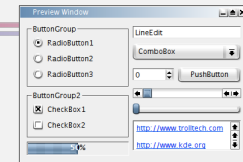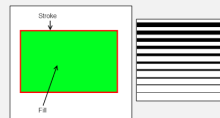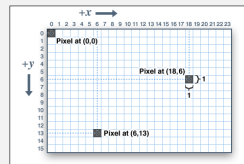# Drawing and Animation

---

GUI Programming in Java

- a *graphical user interface* (GUI) allows users to interact with a program through *direct manipulation* of graphical items
  - contrast with text-based UIs based on typing

- constructing a GUI involves creating and managing windows as well as the components within a window – menus, buttons, drop-down lists, canvases (for drawing), …

- Java provides three toolkits for GUI programming
  - AWT (now obsolete)
  - Swing (2nd generation, still supported)
  - JavaFX (the choice for new applications, a standalone component that isn't part of the standard JDK distribution)

Ingredients for Drawing in JavaFX

- a *graphics context*

- what *shape* to draw
  - line, oval, rectangle

- *where* to draw the shape
  - (x,y) coordinates
  - (0,0) is the upper left corner of the drawing window

- *how big* to make the shape
  - units are in pixels

- *appearance* of the shape
  - line thickness
  - stroke color
  - fill color

https://www.researchgate.net/figure/Common-graphical-primitives-used-in-ABM-visualization_fig1_256364822
https://learn.adafruit.com/adafruit-gfx-graphics-library/coordinate-system-and-units
https://stackoverflow.com/questions/4125152/difference-between-stroke-and-fill
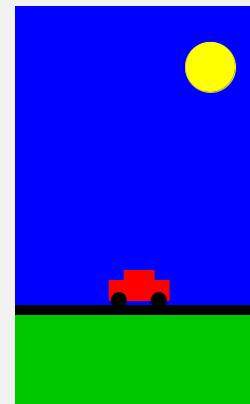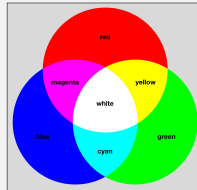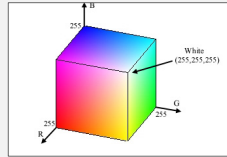

Graphics Context

- all drawing in JavaFX goes through a *graphics context*
  - each graphics context is associated with a different drawing surface
  –

- type is `GraphicsContext`

Position and Size

(x,y) coordinates
(0,0) is the upper left corner of the drawing window

What Shape

- `g.strokeLine(x1,y1,x2,y2)` draws a line from the point with coordinates (x1,y1) to the point with coordinates (x2,y2). The width of the line is 1, unless a different line width has been set by calling `g.setLineWidth()`, and the color is black unless a different color has been set by calling `g.setStroke()`.
- `g.strokeRect(x,y,w,h)` draws the outline of a rectangle with vertical and horizontal sides. This subroutine draws the outline of the rectangle whose top-left corner is x pixels from the left edge of the drawing area and y pixels down from the top. The horizontal width of the rectangle is w pixels, and the vertical height is h pixels. Color and line width are set by calling `g.setStroke()` and `g.setLineWidth()`.
- `g.fillRect(x,y,w,h)` is similar to `g.strokeRect()` except that it fills in the inside of the rectangle instead of drawing an outline, and it uses the color set by `g.setFill()`.
- `g.strokeOval(x,y,w,h)` draws the outline of an oval. The oval just fits inside the rectangle that would be drawn by `g.strokeRect(x,y,w,h)`. To get a circle, use the same values for w and for h.
- `g.fillOval(x,y,w,h)` is similar to `g.strokeOval()` except that it fills in the inside of the oval instead of drawing an outline.

- *g* is a variable with the graphics context

- `stroke` outlines the shape (no fill)
- `fill` draws a filled shape

Appearance

- `g.setFill(c)` is called to set the color to be used for filling shapes. The parameter, c is an object belonging to a class named *Color*. There are many constants representing standard colors that can be used as the parameter in this subroutine. The standard colors range from common colors such as `Color.BLACK`, `Color.WHITE`, `Color.RED`, `Color.GREEN`, `Color.BLUE`, and `Color.YELLOW`, to more exotic color names such as `Color.CORNFLOWERBLUE`. (Later, we will see that it is also possible to create new colors.) For example, if you want to fill shapes with red, you would say "`g.setFill(Color.RED);`". The specified color is used for all subsequent fill operations up until the next time `g.setFill()` is called. Note that previously drawn shapes are **not** affected!
- `g.setStroke(c)` is called to set the color to be used for stroking shapes. It works similarly to `g.setFill`.
- `g.setLineWidth(w)` sets the size of the pen that will be used for subsequent stroke operations, where w is measured in pixels.

- set fill, stroke, and line width *before* drawing shapes
  - think of it as "set the current fill/stroke/line width"

- it is not necessary to set for every shape drawn
  - the current setting stays in effect until changed

Color

- Java defines a large number of colors

  - standard colors
    - Color.RED
    - Color.YELLOW
    - Color.GREEN
    - Color.CYAN
    - Color.BLUE
    - Color.MAGENTA
    - Color.WHITE
    - Color.BLACK

  - other colors
    - https://htmlcolorcodes.com/color-names/
    - https://openjfx.io/javadoc/17/javafx.graphics/javafx/scene/paint/Color.html

https://www.pinterest.com/pin/426012445987636198/
https://www.allbusinesstemplates.com/template/WDC9R/rgb-color-chart/

RGB Color

- a color is defined using three color components – red, green, blue
  - values range from 0 (off) to 255 (on) for each component
  - grayscale values have the same red, green, blue components

- *additive* color mixing for light
  - red + blue = magenta (255,0,255)
  - red + green = yellow (255,255,0)
  - green + blue = cyan (0,255,255)
  - all colors combined makes white
    - white is (255,255,255)
    - black is (0,0,0)

- making your own color

```
Color c = Color.rgb(100,50,0);
```


https://www.britannica.com/science/RGB-colour-model
http://matlab.izmiran.ru/help/toolbox/images/color4.html

Animation

- an *animation* is a sequence of individual *frames*

- need *animation variables* for what changes from one frame to the next

---