Intro to Computer Science

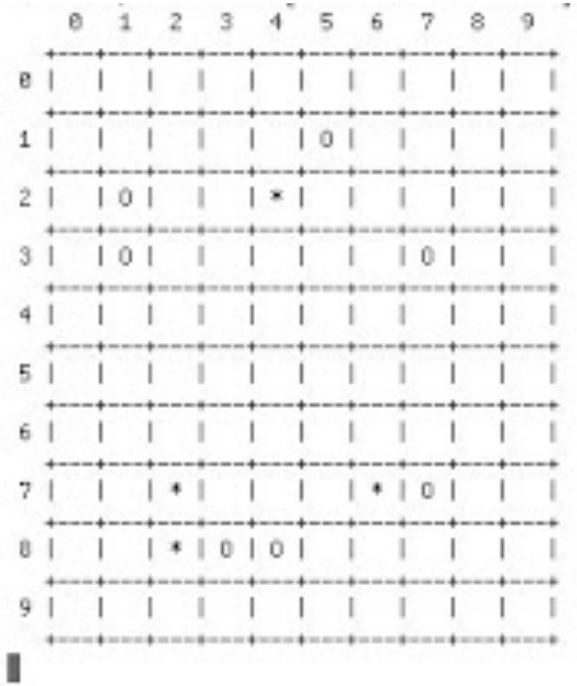
CS-UH 1001

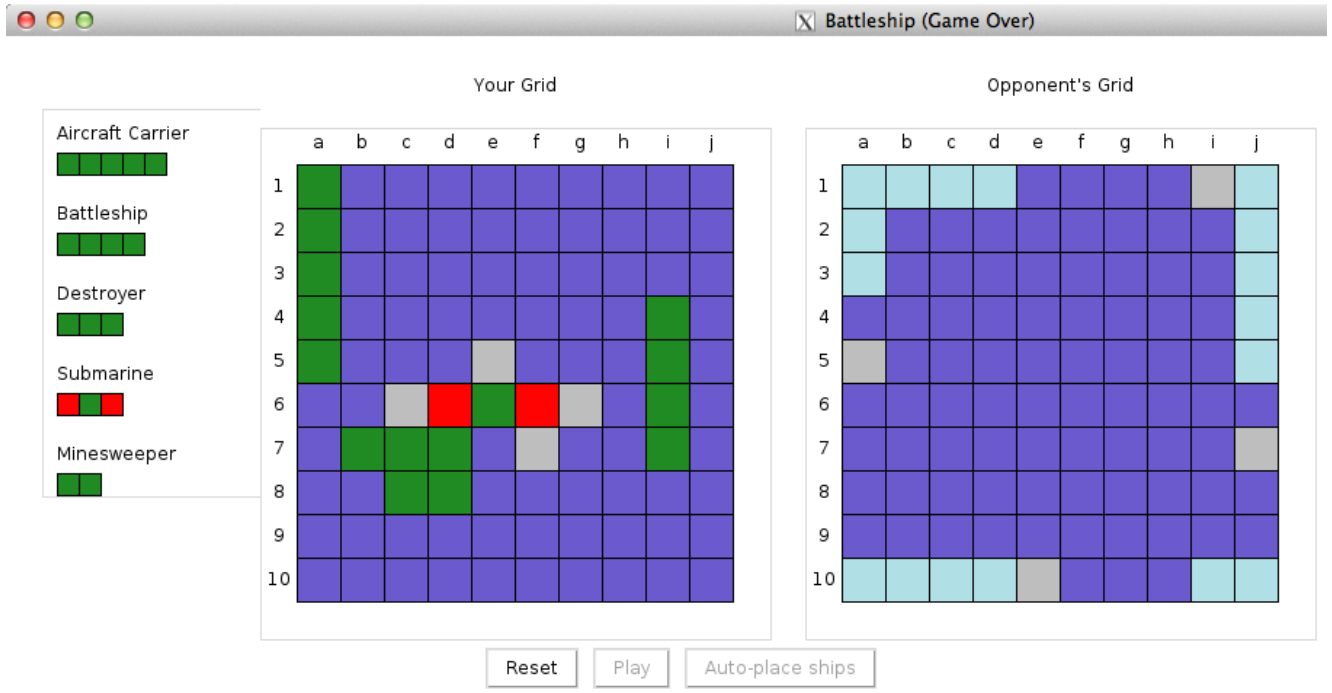
Lecture 15 – Graphical User Interface (GUI) using Processing

Graphical User Interface (GUI)

- A GUI allows the user to interact with a program/electronic device through icons and other visual indicators rather than text via command line
  - Your OS uses GUIs (Windows, MacOS, Linux, Android, etc.)
  - Your programs use GUIs (MS Office, Calculator, etc.)

Example: Graphical User Interface (GUI)





Processing

- Processing is an open source programming language and integrated development environment (IDE)
- It was built for electronic arts and new media art
- The project started in 2001 by Casey Reas and Benjamin Fry (formerly of the Aesthetics and Computation group at MIT Media Lab)

Installing Processing

- Please install Processing from: processing.org/releases

- Make sure you install version 3.5.4!!!
  - Version 4.X does not support Python!

Stable Releases

Read about changes between Processing 3.0 and Processing 4.0.
The list of revisions covers the differences between releases in detail.

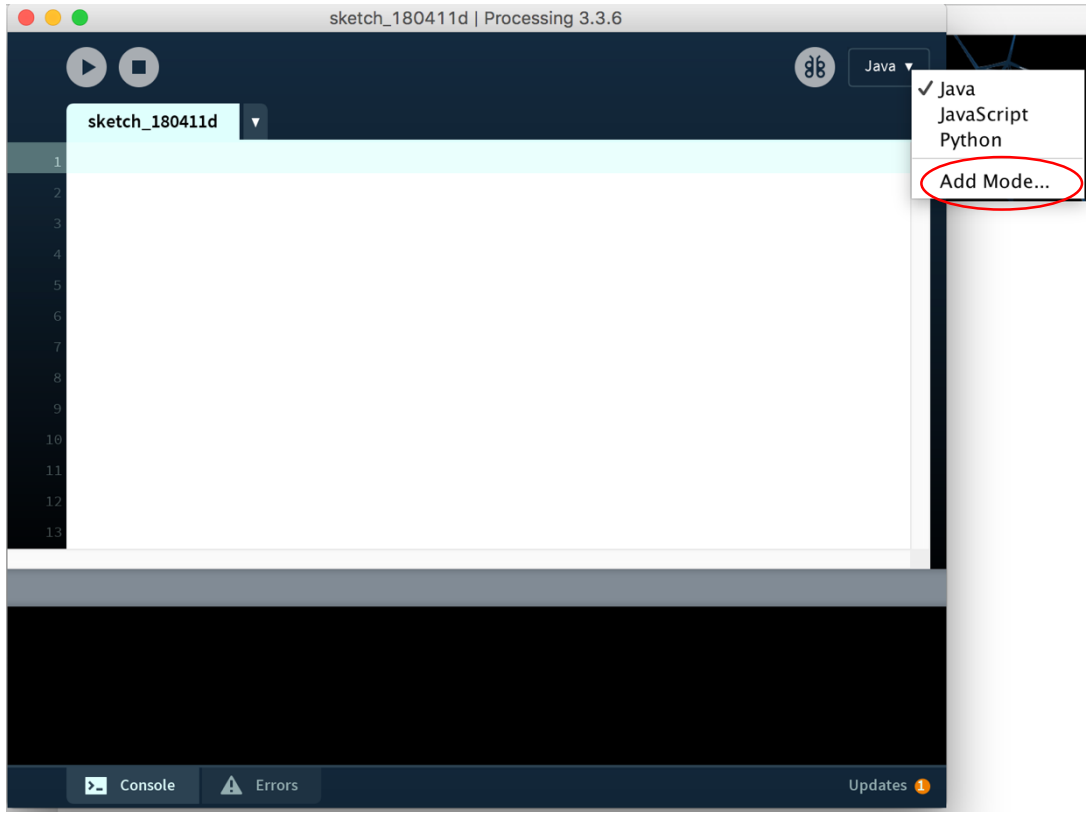
Version 4.2 - (February 20, 2023)

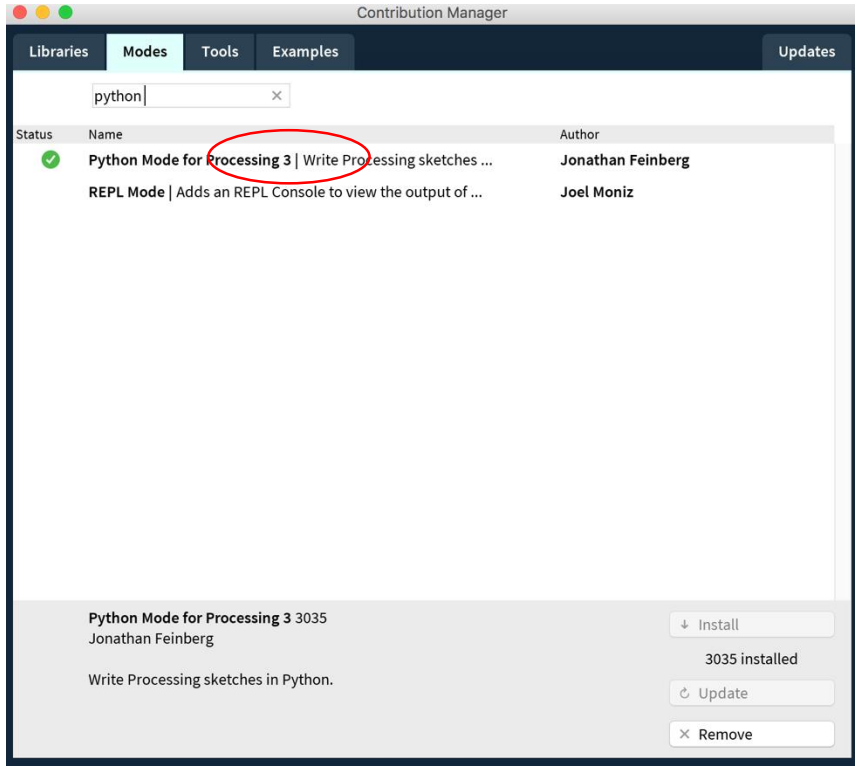
Version 3.5.4 - (January 17, 2020)

- macOS
- Linux Intel 64-bit
- Windows 64-bit
- Windows 32-bit

Version 2.2.1 - (July 31, 2014)

Installing Python Mode for Processing



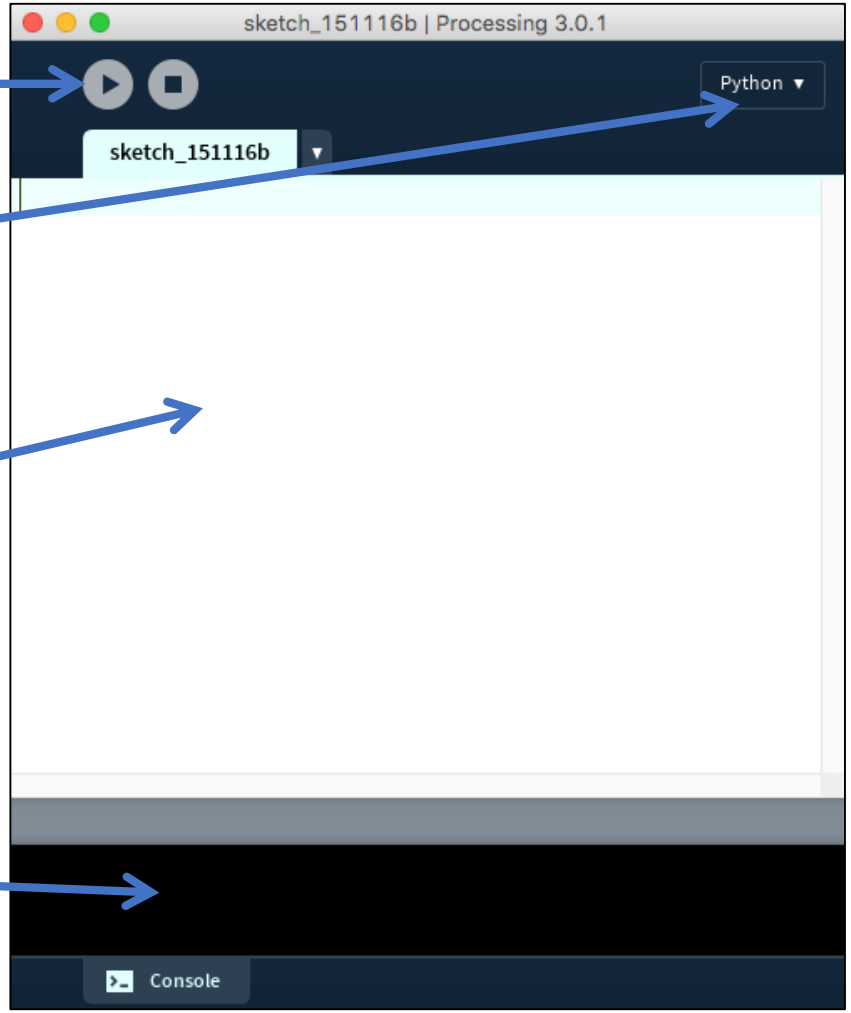


Install Python Mode for Processing 3
- Python Mode for Processing 4 does not work!

When the installation finish, restart processing
Switch the mode to python

Processing IDE

- Run/Stop buttons
- Mode
- Text editor
- Terminal output



Processing Basic Mandatory Functions

- For every Processing program, these two functions must be defined:
  - setup()
  - draw()

The setup() Function

- The setup() is the first function that is executed when the Processing program (sketch) starts
- The function is used to define the initial environment properties such as canvas size, background color, etc.
  - Canvas screen size:
    - size(width, height): both width and height are in pixels
  - Window background color:
    - background(color): color values (0-255) for grayscale, or R,G,B for other colors
- Important: any variable declared inside the setup() function will not be accessible anywhere else (local function variable!)

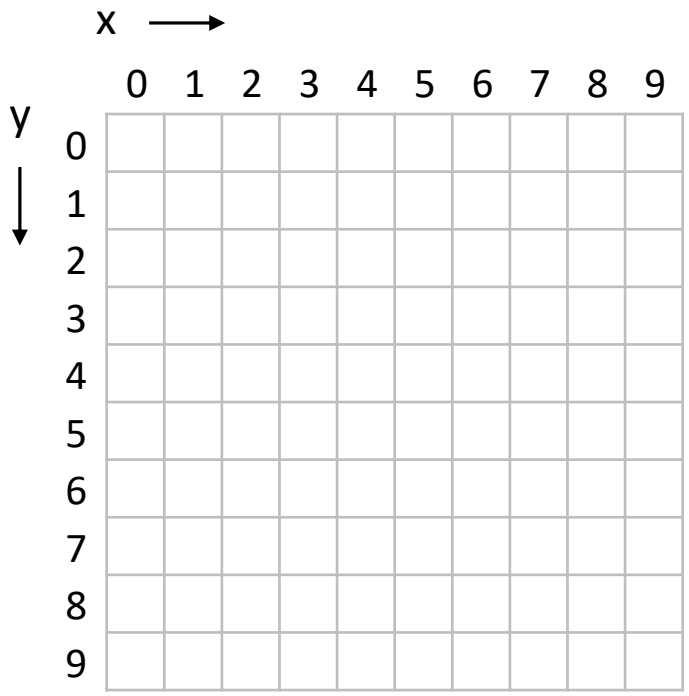
The draw() Function

- This function is called directly after the setup() function
- The function will continuously loop and executes the lines of code inside it
  - Processing calls this function from the background (callback)
- The number of times the draw() function is invoked per second can be defined by the frameRate() function
  - e.g., frameRate(30)
  - Default Processing frame rate is 60 frames per second

Canvas

- The canvas is represented as points (pixels)
  - The upper left corner is the origin (x=0, y=0)

- Example, canvas with 10x10 pixels:



Processing Basic Shapes

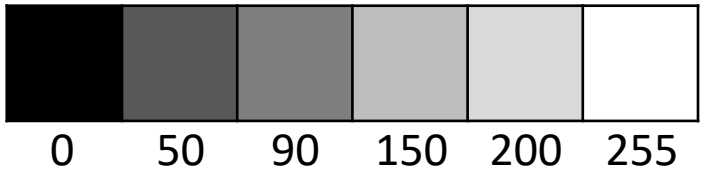
- Basic shapes:
  - point(x, y): Draws a point at x, y
  - line(x1, y1, x2, y2): draws a line from x1,y1 to x2,y2
  - rect(x, y, width, height): draws a rectangle
  - triangle(x1, y1, x2, y2, x3, y3): draws a triangle
  - ellipse(x, y, width, height): draws an ellipse

- Shapes are formatted using:
  - noFill(): transparent background
  - strokeWeight(width): sets the width of the border lines
  - fill(color): sets the color of the shape
  - stroke(color): sets the color of lines and border lines

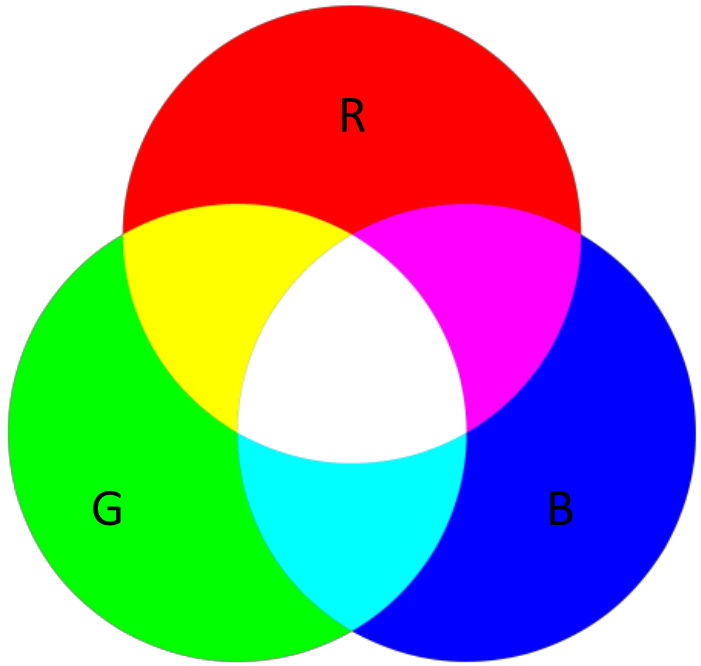
- Note: the formatting functions must be called before drawing a shape

Colors

- Colors are represented in
  - Grayscale: 0 – 255
    - Numbers closer to 0 will be darker, and closer to 255 will be lighter



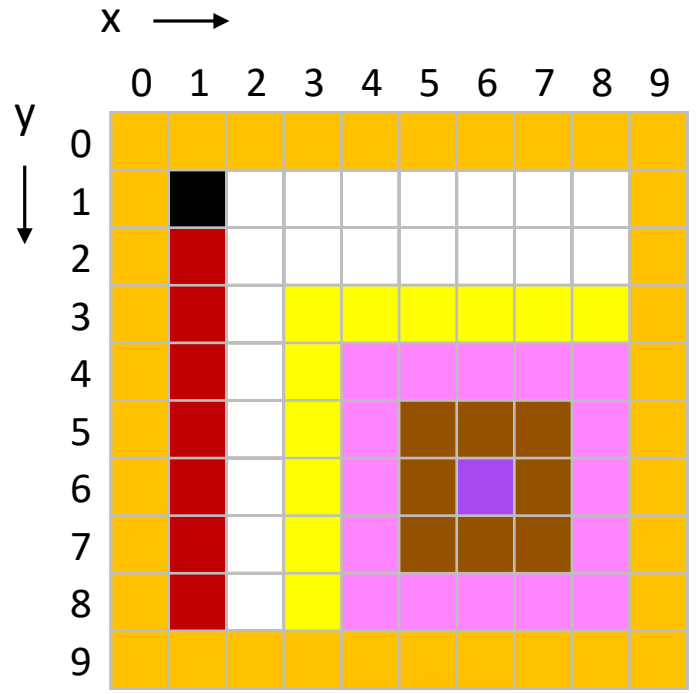
  - RGB (Red, Green, Blue):
    - Red: 255, 0, 0
    - Green: 0, 255, 0
    - Blue: 0, 0, 255
    - Black: 0, 0, 0
    - White: 255, 255, 255

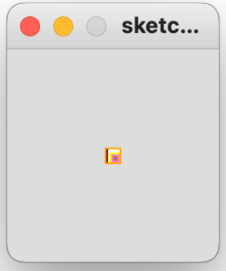


- Colors can be eliminated (transparent) with the functions:
  - noStroke()
  - noFill()

Using Multiple Shapes and Colors

```
point(1, 1)

stroke(127, 0, 0) #dark red
line(1, 2, 1, 8)

stroke(255, 255, 0) #yellow
fill(255, 140, 210) #pink
rect(3, 3, 6, 6)

stroke(190, 130, 90) #brown
fill(165, 75, 240) #purple
rect(5, 5, 2, 2)

fill(255, 170, 0) #orange
noFill() #transparent
rect(0, 0, 9, 9)
```





Let's Create our First Program

```
def setup():
    size(640, 320)
    stroke(0, 0, 0)
    background(255, 255, 255)

def draw():
    line(0, 0, 100, 50)
```

Sets the color of the of the lines and borders around shapes

Draws a line between two points, takes 4 arguments (x1, y1, x2, y2)

Mouse Functions

- In Processing there are a number of useful global variables and callback functions that are related to the mouse input:
  - mouseX: a variable that holds the current x-coordinate of the mouse position on the screen
  - mouseY: a variable that holds the current y-coordinate of the mouse position on the screen
  - mouseClicked(): a function definition that is called after a mouse button has been pressed and then released

Modify our First Program

```
def setup():
    size(640, 320)
    stroke(0, 0, 0)
    background(255, 255, 255)


def draw():
    line(0, 0, mouseX, mouseY)
```

Holds the x and y location of the mouse

Example: Drawing an Ellipse

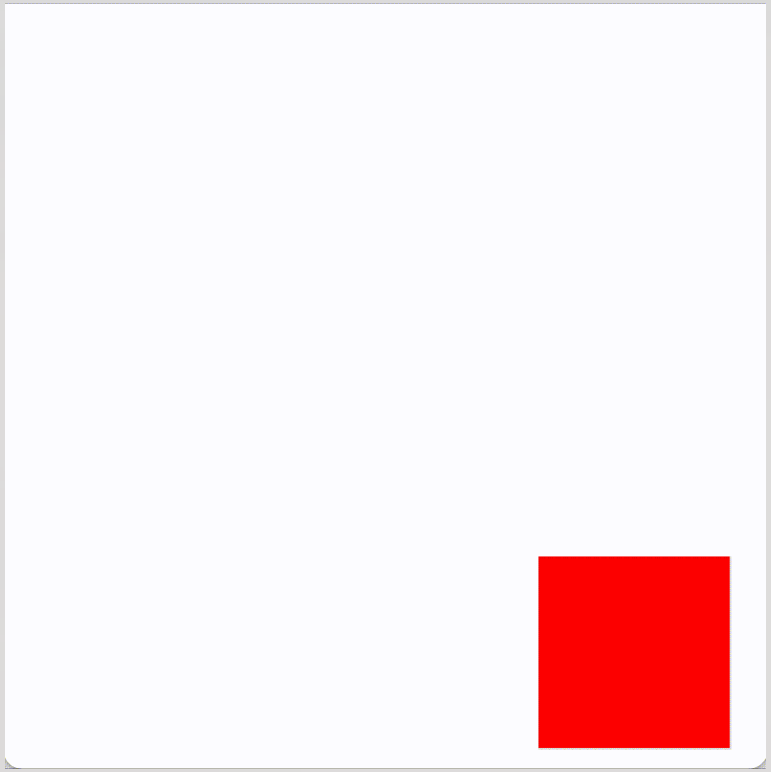
```
def setup():
    size(640, 320)
    stroke(0, 0, 0)
    background(255, 255, 255)

def draw():
    ellipse(150, 25, mouseX, mouseY)
```

Breakout Session I

Rectangle

Example: Using the Rectangle Class (ex_15.1)

- Use the Rectangle class from last class to draw a rectangle that follows the mouse
  - Add a method to it that displays the rectangle based on the attributes (x, y, w, h)
  - Hint: You will need to modify the move() method



Keyboard Functions

- In Processing there are a number of global variables and callback functions that are related to the keyboard input:
  - keyPressed(): a function that is called after a key has been pressed
    - key: a variable that holds the value of the most recent key press (a, b, c, etc.)
    - keyCode: a variable that holds the value of the most recent special key press (LEFT, RIGHT, DOWN, UP, etc.)
  - keyReleased(): a function that is called after a key has been released
  - More information can be found here: https://processing.org/reference/keyCode.html

Example: Using the Rectangle Class (ex_15.1)

- Use the Rectangle class from last class to draw a rectangle that follows the mouse
  - Hint: You will need to modify the move() method

- Add keyboard input so that the rectangle can be resized using the UP and DOWN arrow keys

Example: Creating Art – Particles (ex_15.2)

- Create a Particle class
  - with the following attributes:
    - x, y: Absolute position of the particle
    - d: Diameter of the particle
    - cx, cy: Center point (origin) of the circle
    - r: Magnitude of the circle
    - θ: Counterclockwise angle of the Particle
  - and the following methods:
    - update(): Modify θ, x and y
      - $\theta = \theta + 0.1$
      - $x = cx + r * \cos \theta$
      - $y = cy + r * \sin \theta$
      - $r = r - 0.2$
    - display(): Display the Particle

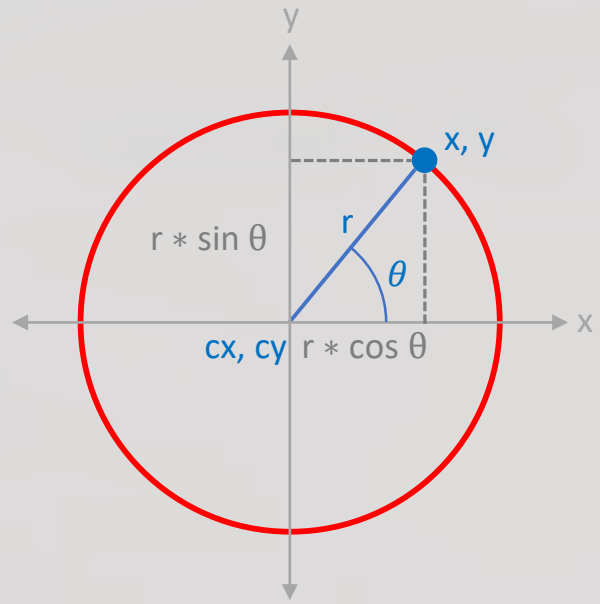


Image Handling

- img = loadImage(path to the image on disk): loads an image and supports four image types: .png, .jpg, .gif, and .tga.
  - Use loadImage() only once to load the image from memory!

- image(img, x, y, width, height, x1, y1, x2, y2): used to display the loaded image on the screen
  - img: is the name of the variable used to load the image
  - x and y: the location of the upper left corner of the image on the screen
  - width and height (optional): resize/shrink/expand the image when displayed on the screen
  - x1,y1,x2, and y2 (optional): crop the loaded image
    - x1,y1 represent the upper left corner
    - x2,y2 represent the lower right corner

Other Useful Processing Functions

- There are lots of other useful functions in Processing

- The link below provides a full list and description of each: https://py.processing.org/reference/