CS1103 Workshop 5

FUNCTIONS

Functions, Built-in functions, Writing our own functions, Functions with parameters, Call-back functions

FUNCTIONS

What are they?

Actually, we've been USING (or 'calling') functions all along.

p5.js functions:

- **rect()**
- **fill()**
- **loadFont()**

One way to think of a function is simply as a command

FUNCTIONS

What are they?

We've also been adding our own code to functions called *automatically* by **p5.js**

- **setup() { … }**
- **draw() { … }**
- **preload() { … }**

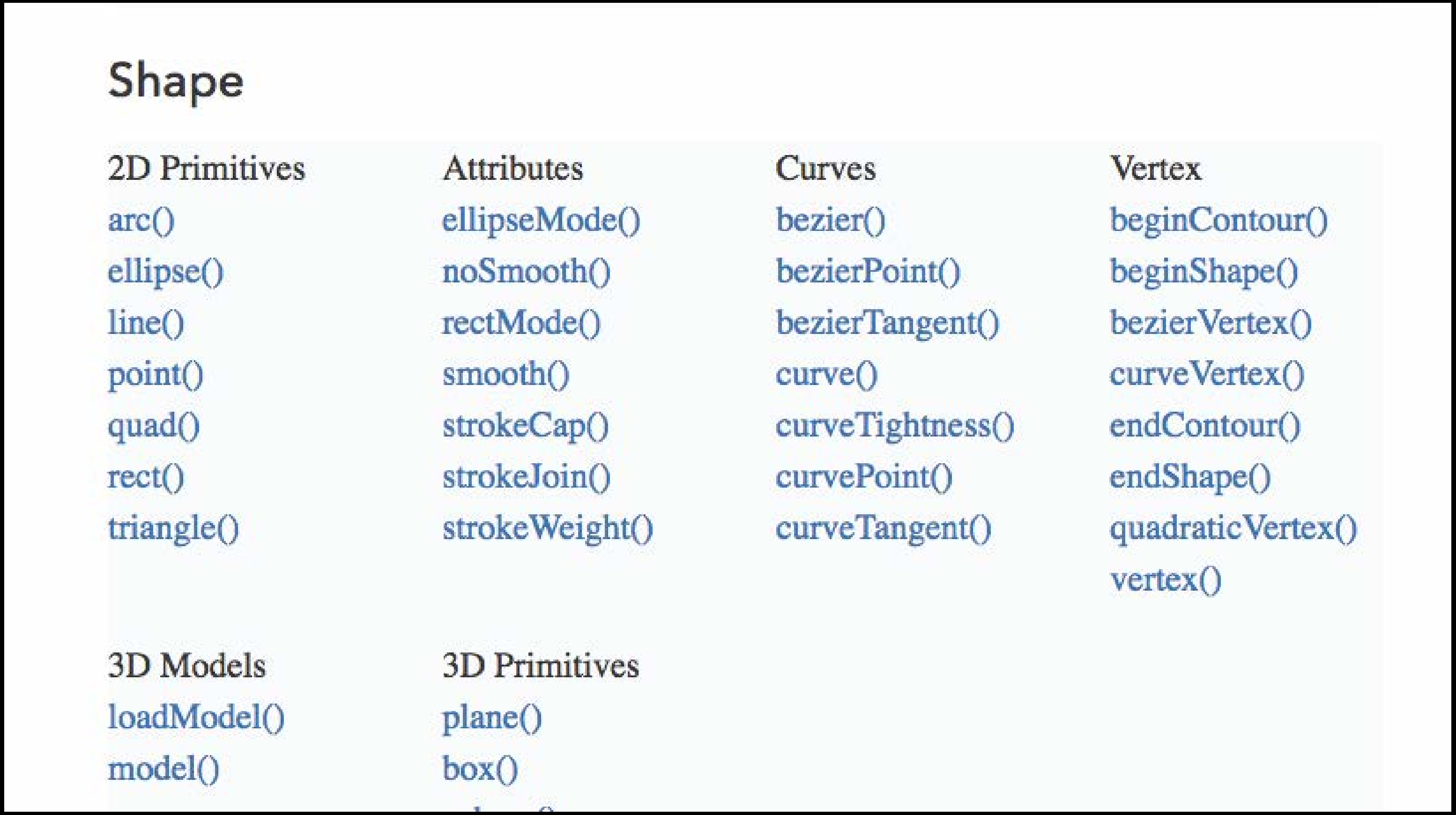
func·tion /**fuhnk**-shu*h*n/

n. a function is just a **block of code**, which we give a **name**, so that it may be **called** when needed

- Programs written with functions are more structure, elegant and easier to read.
- Functions are ***reusable***, specifically, a function is written once and can be invoked whenever necessary.

Built-in Functions

P5.JS HAS HUNDREDS OF FUNCTIONS THAT ARE *BUILT- IN*



Shape

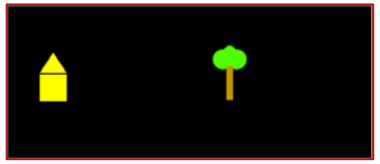
2D Primitives	Attributes	Curves	Vertex
arc()	ellipseMode()	bezier()	beginContour()
ellipse()	noSmooth()	bezierPoint()	beginShape()
line()	rectMode()	bezierTangent()	bezierVertex()
point()	smooth()	curve()	curveVertex()
quad()	strokeCap()	curveTightness()	endContour()
rect()	strokeJoin()	curvePoint()	endShape()
triangle()	strokeWeight()	curveTangent()	quadraticVertex()
			vertex()

3D Models	3D Primitives
loadModel()	plane()
model()	box()

We can also create our own!

Example: Draw house and tree *at random locations*

Like before, we can put drawing code in setup.

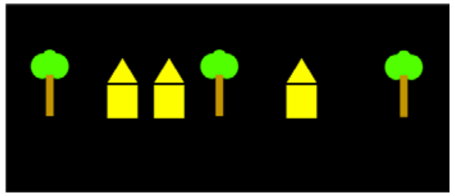


```
function setup() {
   createCanvas(600, 250);
   background(0);
   noStroke();

   //draw a house at a random location
   var xh = random(width);
   fill(255, 255, 0);
   triangle(xh, 80, xh - 20, 110, xh + 20, 110);
   rect(xh - 20, 112, 40, 40);

   //draw a tree at a random location
   var xt = random(width);
   fill(0, 255, 0);
   ellipse(xt, 80, 20, 20);
   ellipse(xt - 10, 90, 30, 30);
   ellipse(xt + 10, 90, 30, 30);
   fill(200, 150, 0);
   rect(xt - 5, 100, 10, 50);
}
```

Example: Draw house and tree *at random locations*

But, if we put the code in drawHouse() and drawTree()
Then we can call from setup() multiple times!

```
function setup() {
  createCanvas(600, 250);
  background(0);
  noStroke();

  drawHouse(); drawHouse(); drawHouse();
  drawTree(); drawTree(); drawTree();
}

function drawHouse() {
  var x = random(width);
  fill(255, 255, 0);
  triangle(x, 80, x - 20, 110, x + 20, 110);
  rect(x - 20, 112, 40, 40);
}

function drawTree() {
  var x = random(width);
  fill(0, 255, 0);
  ellipse(x, 80, 20, 20);
  ellipse(x - 10, 90, 30, 30);
  ellipse(x + 10, 90, 30, 30);
  fill(200, 150, 0);
  rect(x - 5, 100, 10, 50);
}
```

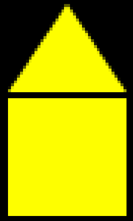


We say:
we create our own functions: drawHouse and drawTree.

To conclude:

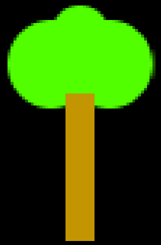
- We have put some lines of code to draw a house.
- We think of these lines as to do one single task : *to draw a house*
- We give this command a name: let's call it drawHouse()
- And write its **definition** as a *function*:

```
function drawHouse() {
   .. your code ..
}
```



- Similar for drawTree()

```
function drawTree() {
   .. your code ..
}
```



- Discussion: What are the advantages of writing functions?

About () :

- Currently, in drawHouse(), the brackets () is empty.

  Even so, the () are needed.

- This is similar to noFill(), noStroke() etc..
- **xxx()** or **xxx(..)** means a function

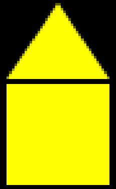
  Without (), it is not a function.

- Examples:

  These are NOT functions: `mouseX, width, frameCount`

  These are functions: `fill(r,g,b), noFill(), random(v1, v2)`

```
function drawHouse() {
  .. your code ..
}
```



They are variables that store values

They are functions that we can run.

Functions with Parameters

- Some functions will **accept parameters**, for example, rect() requires parameters to specify the location, the width, and height.
- Some functions **require no parameter**, for example, noStroke(), noFill()

We can add parameters in our functions!

```
createCanvas(600, 250);
background(0);
noStroke();

//draw a house
fill(255, 255, 0);
triangle(100, 80, 80, 110, 120, 110);
rect(80, 112, 40, 40);
```

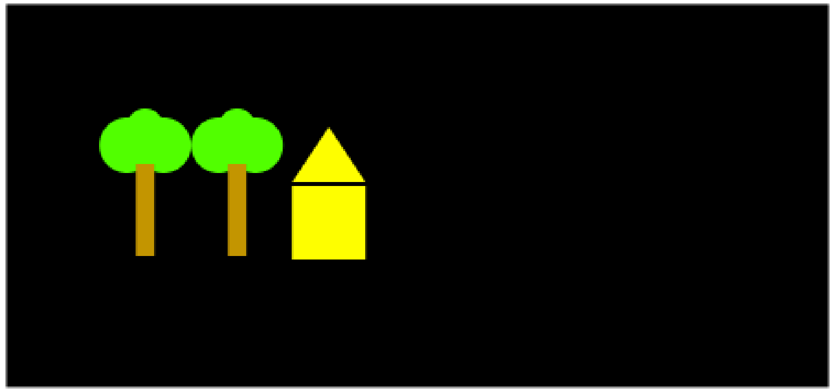
Example: Draw house and tree *at given locations*

```
function setup() {
  createCanvas(600, 250);
  background(0);
  noStroke();

  var xh = random(width);
  drawHouse(xh);
  drawTree(xh-50);
  drawTree(xh-100);
}

function drawHouse(x) {
  ~~var x = random(width);~~
  fill(255, 255, 0);
  triangle(x, 80, x - 20, 110, x + 20, 110);
  rect(x - 20, 112, 40, 40);
}

function drawTree(x) {
  ~~var x = random(width);~~
  fill(0, 255, 0);
  ellipse(x, 80, 20, 20);
  ellipse(x - 10, 90, 30, 30);
  ellipse(x + 10, 90, 30, 30);
  fill(200, 150, 0);
  rect(x - 5, 100, 10, 50);
}
```

We now have the x-coordinate obtained from the parameter **x**. Therefore the variable is not needed.

Put the code in drawHouse() and drawTree()
Then call them with parameter values for x

Draw two trees to the left of the house

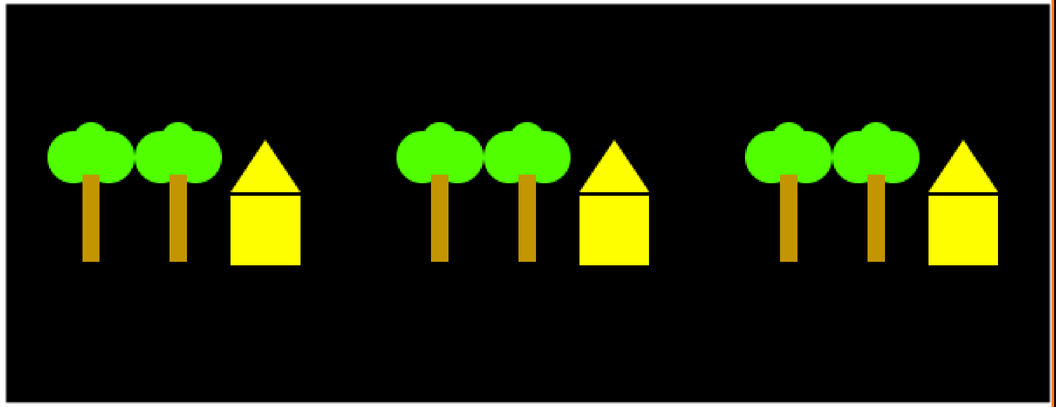


Example: Draw house and tree *at calculated locations*

```
function setup() {
  createCanvas(600, 300);
  background(0);
  noStroke();

  for (var i = 0; i < 3; i++) {
     var xh = 150 + i * 200;
     drawHouse(xh);
     drawTree(xh - 50);
     drawTree(xh - 100);
  }
}

function drawHouse(x) {
  fill(255, 255, 0);
  triangle(x, 80, x - 20, 110, x + 20, 110);
  rect(x - 20, 112, 40, 40);
}

function drawTree(x) {
  fill(0, 255, 0);
  ellipse(x, 80, 20, 20);
  ellipse(x - 10, 90, 30, 30);
  ellipse(x + 10, 90, 30, 30);
  fill(200, 150, 0);
  rect(x - 5, 100, 10, 50);
}
```

Put the code in drawHouse() and drawTree()
Then call them with parameter values for x

Repeat 3 times!!
Draw two trees and a house



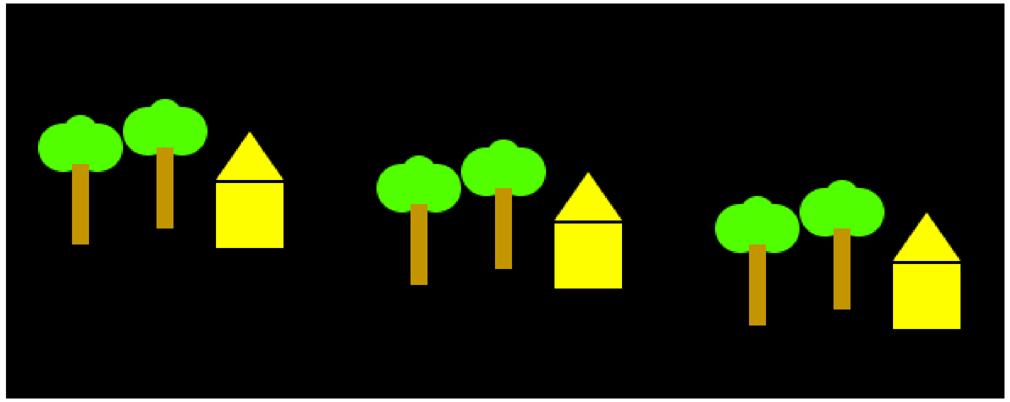
Example: Draw house and tree *at calculated locations (x, y)*

```
function setup() {
  createCanvas(600, 300);
  background(0);
  noStroke();

  for (var i = 0; i < 3; i++) {
    var xh = 150 + i * 200;
    var yh = 80 + i * 25;
    drawHouse(xh, yh);
    drawTree(xh - 50, yh - 10);
    drawTree(xh - 100, yh);
  }
}

function drawHouse(x, y) {
  fill(255, 255, 0);
  triangle(x, y, x - 20, y + 30, x + 20, y + 30);
  rect(x - 20, y + 32, 40, 40);
}

function drawTree(x, y) {
  fill(0, 255, 0);
  ellipse(x, y, 20, 20);
  ellipse(x - 10, y + 10, 30, 30);
  ellipse(x + 10, y + 10, 30, 30);
  fill(200, 150, 0);
  rect(x - 5, y + 20, 10, 50);
}
```

Put the code in drawHouse() and drawTree()
Then call them with parameter values for x and y

Repeat 3 times!!

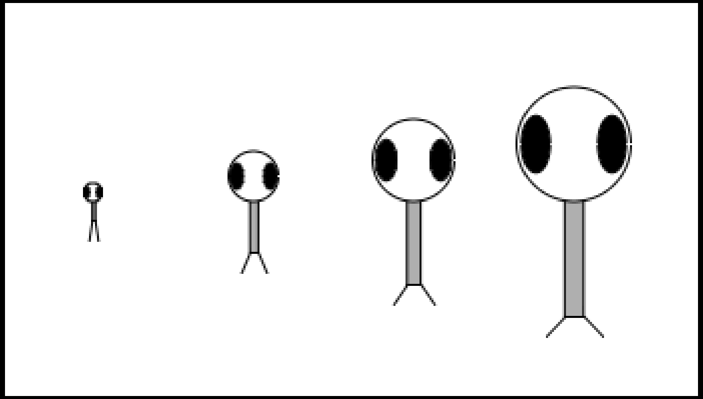
Draw two trees and a house



```
// x and y specify location
// w and h specify width and height of zoog
// eyeSize specify the size of zoog's eyes
function drawZoog(x, y, w, h, eyeSize) {
    // Draw Zoog's body
    stroke(0);
    fill(175);
    rectMode(CENTER);
    rect(x, y, w/6, h*2);

    // Draw Zoog's head
    fill(255);
    ellipse(x, y-h/2, w, h);

    // Draw Zoog's eyes
    fill(0);
    ellipse(x-w/3, y-h/2, eyeSize, eyeSize*2);
    ellipse(x+w/3, y-h/2, eyeSize, eyeSize*2);

    // Draw Zoog's legs
    stroke(0);
    line(x-w/12, y+h, x-w/4, y+h+10);
    line(x+w/12, y+h, x+w/4, y+h+10);
}
```

Job 1:

Given the function drawZoog() with 5 parameters, write a program that draws the picture below.



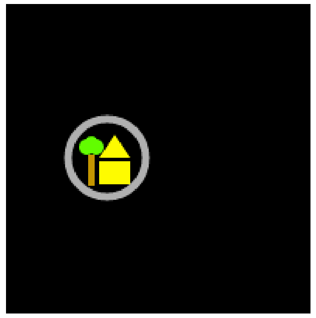
Another example: Division of labor – painting and moving

```
function draw() {
  paint(); //painting object as one task
  move(); //moving object location as another task
}

function paint() {
     .. Code to write ..
}

function move() {
     .. Code to write ..
}
```

Example: Division of labor – painting / moving

Original program:
Move ball to the right, then back again

```
var circleX = 0;
var circleY = 100;

function setup() {
  createCanvas(200, 200);
}

function draw() {

  //paint
  background(0);
  stroke(175);
  strokeWeight(5);
  fill(0);

  ellipse(circleX, circleY, 50, 50);
  .. Other painting ..

  //move
  if (circleX > width) {
    circleX = 0;
  }

  circleX = circleX + 1;
}
```



New program structure:
Divide to painting + moving tasks in 2 functions

```
var circleX = 0;
var circleY = 100;

function setup() {
  createCanvas(200, 200);
}

function draw() {
  paint();
  move();
}

function paint() {
  background(0);
  stroke(175);
  strokeWeight(5);
  fill(0);

  ellipse(circleX, circleY, 50, 50);
  .. Other painting ..
}

function move() {
  circleX = circleX + 1;
  if (circleX > width) {
    circleX = 0;
  }
}
```

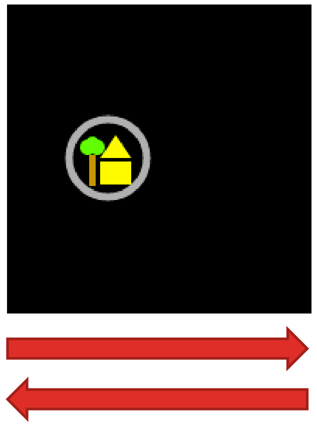
What are the benefits of writing the tasks in 2 functions?

[Answer]
There are a lot:
e.g. to revise one task, we don't want to cause trouble to others;
debugging becomes easy;
the draw function can change to call other move functions (moveSlow(), moveIrregular(), ..)

Example: Division of labor – painting / moving / bouncing

Given program:
Bouncing the ball left and right

```
var circleX = 25;
var circleY = 100;
var direction = 1;

function setup() {
  createCanvas(200, 200);
}

function draw() {

  //paint
  background(0);
  stroke(175);
  strokeWeight(5);
  fill(0);

  ellipse(circleX, circleY, 50, 50);
  .. Other painting ..

  //move
  circleX = circleX + direction;

  //bounce
  if (circleX > width - 25 || circle < 25) {
    direction = direction * -1;
  }
}
```



Job 2: New program structure
Divide tasks in 3 functions: paint(), move(), bounce()

```
var circleX = 25;
var circleY = 100;
var direction = 1;

function setup() {
  createCanvas(200, 200);
}
```

Your code

Callback Functions

What are they?

Callback function is also a kind of built-in function. However, we do not need to explicitly invoke a callback function. Precisely, each callback function corresponds to an event (e.g., mouse clicked) and is invoked automatically when the event happens.

Hence, callback function is also referred to as "**event handler**".

Callback function is particularly useful for program with user interaction.

Callback Functions

What are they?

Examples of callback functions are:

mousePressed(): Invoke every time when a mouse button is pressed.

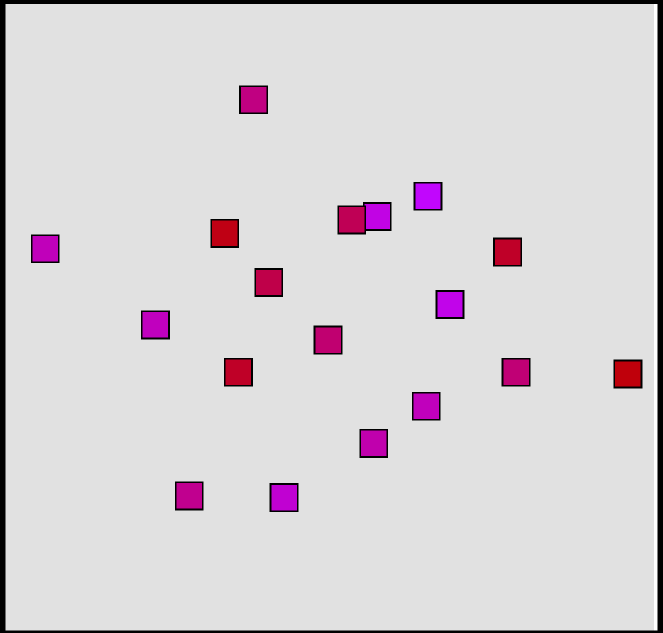
mouseReleased(): Invoke every time when a mouse button is released.

mouseDragged(): Invoke every time when a mouse button is dragged.

keyPressed(): Invoke every time when a key is pressed.

Example: Small rectangle is drawn every time a mouse button is clicked
Canvas is erased when a key is pressed

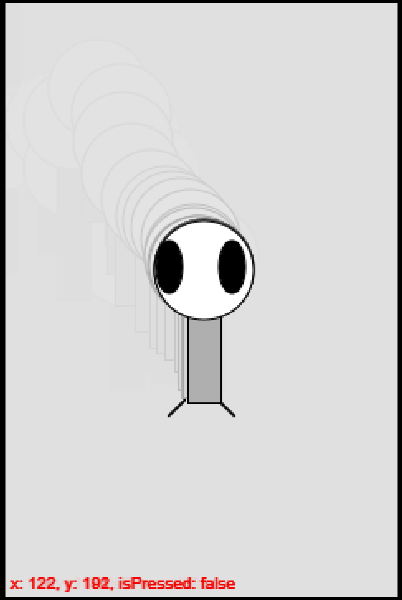
```
function setup() {
  createCanvas(400, 400);
  background(225);
}

// Whenever a user clicks the mouse, the code inside mousePressed() is executed.
function mousePressed() {
  stroke(0);
  fill(200, 0, random(255));
  rectMode(CENTER);
  rect(mouseX, mouseY, 16, 16);
}

// Whenever a user presses a key, the code inside keyPressed() is executed.
function keyPressed() {
  background(225);
}
```



Example 2:

Draw Zoog's eyes and legs at the center.

Move Zoog's body following the mouse until the user clicks to confirm the placement.


```
x: 122, y: 192, isPressed: false
```

```
var x, y, isPressed;

function setup() {
  createCanvas(240, 360);
  ellipseMode(CENTER);
  rectMode(CENTER);

  isPressed = false;
}

function draw() {
  background(225);

  if (isPressed == false) {
    x = mouseX;
    y = mouseY;
  }

  drawEyes();
  drawLegs();

  placeLego();  //draw body and head (mouse location)
  drawEyes();  //redraw eyes on lego

  textSize(10); noStroke(); fill(255,0,0);
  text("x: "+x+", y: "+y+", isPressed: "+isPressed, 5, 355);
}
```

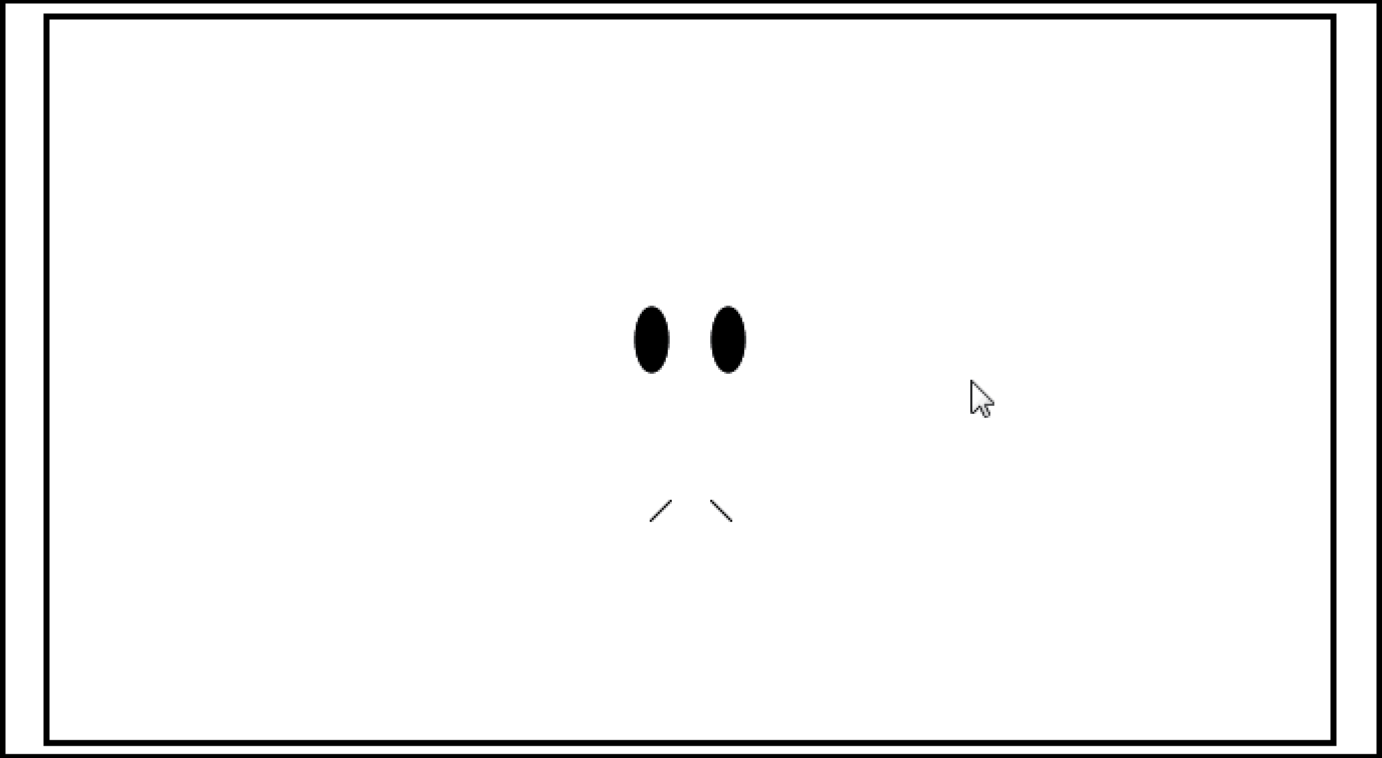
Functions: drawEyes, drawLegs, placeLego

```
function mousePressed() {
  isPressed = true;
}
```

```
function drawEyes () {

  fill(0);
  ellipse(101, 160, 16, 32);
  ellipse(139, 160, 16, 32);
}

function drawLegs() {

  stroke(0);
  line(110, 240, 100, 250);
  line(130, 240, 140, 250);
}

function placeLego() {

  // Draw body and head (mouse location)

  stroke(0);
  fill(175);
  rect(x, y, 20, 100);

  stroke(0);
  fill(255);
  ellipse(x, y - 30, 60, 60);
}
```

Job 3:

Modify the previous program so that the body of Zoog is moved only when the mouse button is held down and dragging over the canvas.



Assignment of this lesson:

Create a program according to the requirement below.

Submit it on Canvas.

Due: Please check the deadline on Canvas.

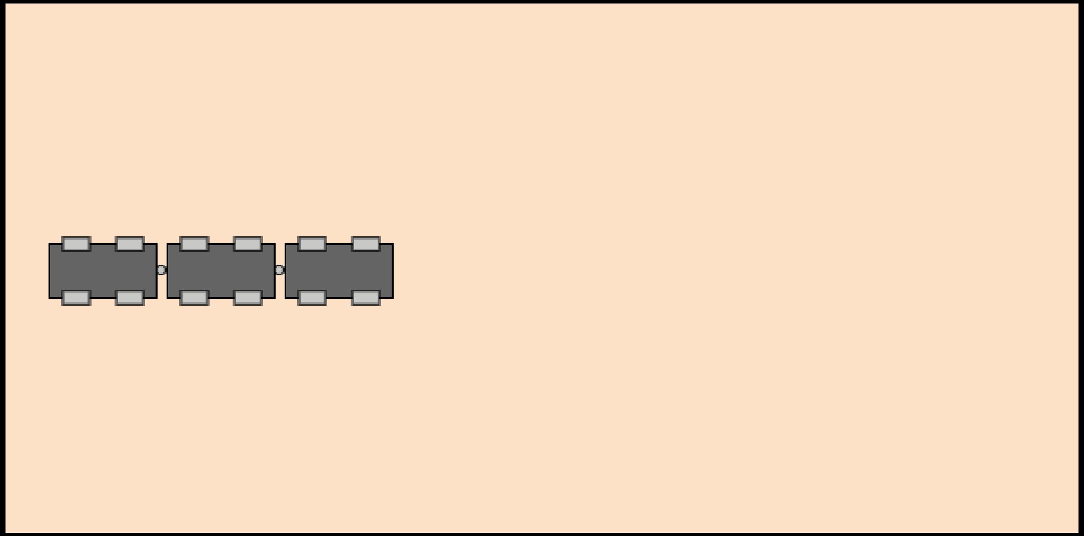
You are given a function for drawing car of different size at different location. Write a program, using the `drawCar()` function, such that a train of 3 cars will move. Furthermore, the size of cars is double when left mouse button is pressed, and reduced by half when the right mouse button is pressed. Interact with the animation below!

1. Achieve required effect as given in the animation. (4 marks)

   Hint: For checking left/right mouse buttons, use the system variable mouseButton (http://p5js.org/reference/)

   Note: Do not change the given `drawCar()` function

2. Implement creative features through writing and calling functions with parameters. (1 mark)



Animation: https://courses.cs.cityu.edu.hk/cs1103/public/Workshop05_AsgAnimation/