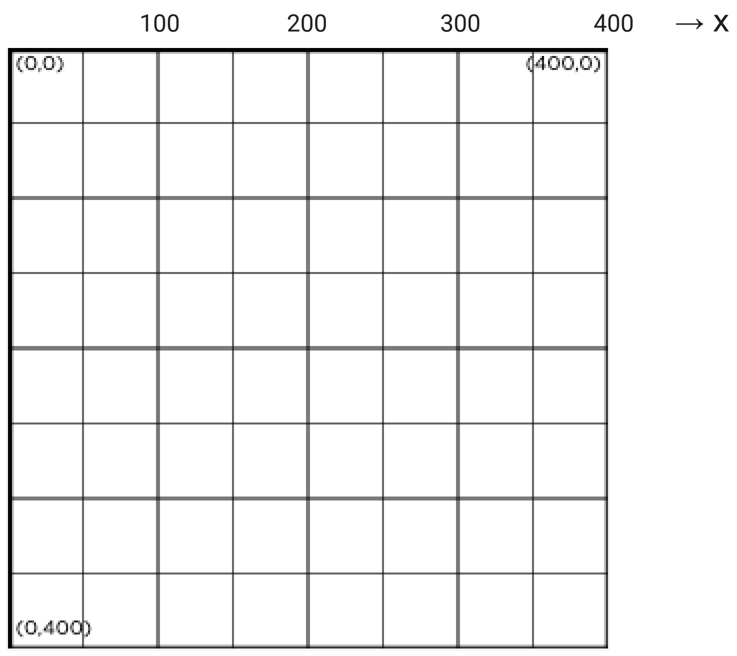
Lesson 2 - Plotting Shapes (computer grid)



(x, y)
0 ⇨ 400 on x-axis
0 ⇩ 400 on y-axis

Each square is 50x50 pixels

⊙ Circles (ellipse block) are drawn from the center out

□ Rectangles are drawn from the upper left

Lesson 3 - Drawing in Game Lab

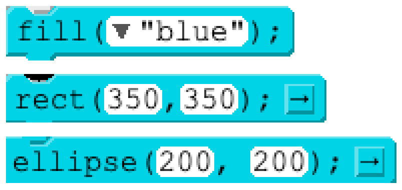
Vocabulary:

- Bug - Part of a program that does not work correctly.
- Debugging - Finding and fixing problems in an algorithm or program.
- Program - An algorithm that has been coded into something that can be run by a machine.

Introduced Code:

- `ellipse(x, y, w, h)`
- `fill(color)`
- `rect(x, y, w, h)`

Examples:



```
fill(▼ "blue");
rect(350, 350); →
ellipse(200, 200); →
```

Lesson 4 - Shapes and Randomization

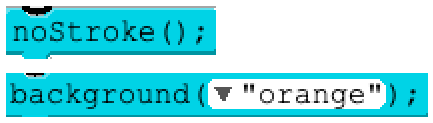
Vocabulary:

- Parameter - Additional information provided as input to a block to customize its functionality

Introduced Code:

- `background(color)` → put this at beginning (order matters)
- `ellipse(x, y, w, h)`
- `rect(x, y, w, h)`

Examples:



```
noStroke();
background(▼ "orange");
```

Lesson 5 - Variables

Vocabulary:

- Variable - A label for a piece of information used in a program.
- camelCase - the first letter of the variable is usually lower case, and each new word starts with a capital letter. This helps you see the words without spaces (spaces are not allowed in variable names)

New Code:

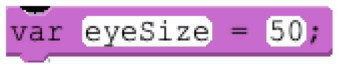
- `var x = ___;`
- `var x;`

Naming Rules:

- No spaces
- Can't begin with a number
- Spelling counts
- Case-sensitive

Example:

```
Equals sign = “gets” the value
eyeSize=50;
eyeSize “gets” the value of 50
```



```
var eyeSize = 50;
```

```
        ↖label  ↖value
```

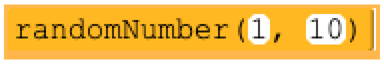
Lesson 6: Random Numbers

Vocabulary:
randomNumber() - used to generate random numbers in your programs. The parameters set the minimum and maximum value that could be generated. You can use this block anywhere that you could write a number.

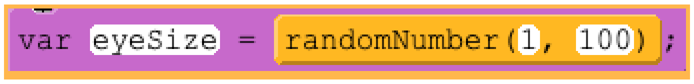
Introduced Code:

`randomNumber(min, max);` →the bigger the range, the bigger the movement

Example:



```
randomNumber(1, 10)
```



```
var eyeSize = randomNumber(1, 100);
```

Lesson 7: Sprites

Vocabulary:

- Dot notation - the way that sprites' properties are used in Game Lab, by connecting the sprite and property with a dot.
- Property - A label for a characteristic of a sprite, such as its location and appearance
- Sprite - A character on the screen with properties that describe its location, movement, and look.

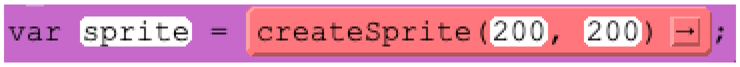
Introduced Code:

`drawSprites()`

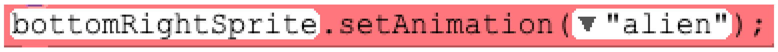
`var sprite = createSprite(x, y)`

`sprite.setAnimation (label)`

Examples: ↙location on the grid (x, y)

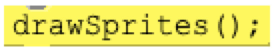


```
var sprite = createSprite(200, 200);
```



```
bottomRightSprite.setAnimation("alien");
```

↖image located in animation tab



```
drawSprites();
```

Lesson 8: Sprite Properties

Vocabulary:

- Property - A label for a characteristic of a sprite, such as its location and appearance

Introduced Code:

`sprite.rotation` – changes across

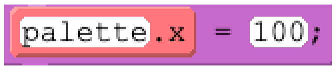
`sprite.scale` – changes up and down

`sprite.x` – rotate/spin

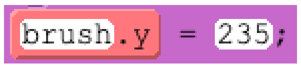
`sprite.y` - size {less than 1 is smaller, more than 1 is bigger}

`sprite.visible` - can see/not see

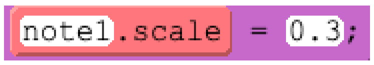
Examples:



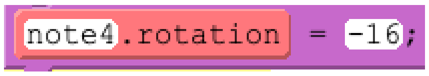
```
palette.x = 100;
```



```
brush.y = 235;
```



```
note1.scale = 0.3;
```



```
note4.rotation = -16;
```

Lesson 9: Text

Vocabulary:

- textFont - changes the default font Arial
- textSize - changes the default size, 12 pixels
- textAlign - change where the text is displayed relative to the (x,y) position specified. The default is that (x,y) is the top left corner of the text.

Note:
Text that does not fit completely within the display area will not be drawn or seen. Use the fourth and fifth parameters to create a text box to display the text in with automatic line wrapping.

Introduced Code:

`textFont()` `textSize()` `textAlign()`

Examples:

```
text("Hello World", 50, 200);
textSize(20);
text("Four score and seven years ago...", 30, 200);
```

Lesson 10: Mini-Project - Captioned Scenes

Criteria:
- ☐ **background(color)**
- ☐ **At least 2 sprites**
- ☐ **Must use the rect block**
- ☐ **Must use the ellipse block**
- ☐ **Needs text - should tell a joke**
- ☐ **Use textSize**

Shapes:	**Colo**
background(color)	fill('c
rect(x, y, width, height)	noFi
ellipse(x, y, width, height)	stroke('color') border color
text (string, x, y, width, height)	noStroke()
textSize(pixels) font size	strokeWeight() thickness

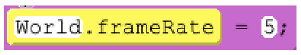
Lesson 11: The Draw Loop

Vocabulary:
- Animation - a series of images that create the illusion of motion by being shown rapidly one after the other
- Frame - a single image within an animation
- Frame Rate - the rate at which frames in an animation are shown, typically measured in frames per second

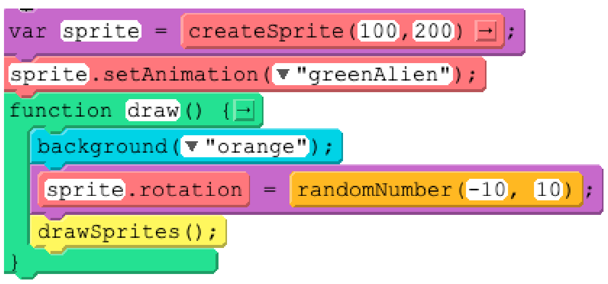
New Code:

`World.frameRate` `function draw() {}`

Examples:



```
World.frameRate = 5;
```



```
var sprite = createSprite(100, 200);
sprite.setAnimation("greenAlien");
function draw() {
  background("orange");
  sprite.rotation = randomNumber(-10, 10);
  drawSprites();
}
```

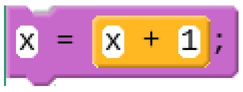
Lesson 12: Sprite Movement

Vocabulary:

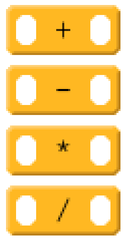
Counter pattern - used to make an image fly across the screen, to count down a timer, or to keep track of clicks. It is used with a variable `x` to count up by one.

Note:
Every time this code is run, it will take the current value of x, add 1, and save that as the new value of x. While this particular instance of the Counter Pattern uses addition, you could also use subtraction to count down.

Introduced Code:

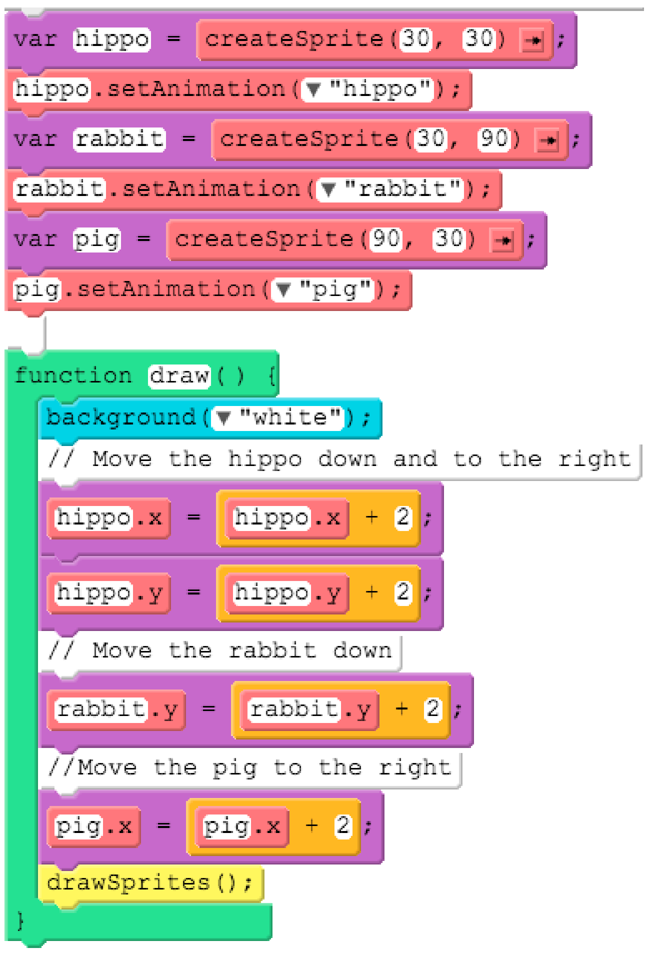


```
x = x + 1;
```

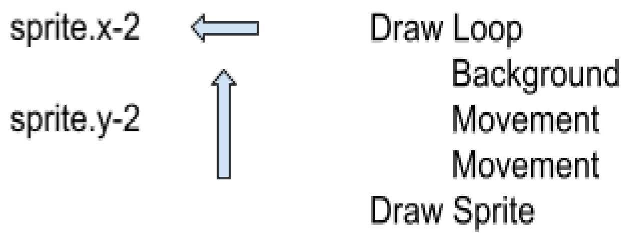


```
_ + _
_ - _
_ * _
_ / _
```

Example:



```
var hippo = createSprite(30, 30);
hippo.setAnimation("hippo");
var rabbit = createSprite(30, 90);
rabbit.setAnimation("rabbit");
var pig = createSprite(90, 30);
pig.setAnimation("pig");

function draw() {
  background("white");
  // Move the hippo down and to the right
  hippo.x = hippo.x + 2;
  hippo.y = hippo.y + 2;
  // Move the rabbit down
  rabbit.y = rabbit.y + 2;
  //Move the pig to the right
  pig.x = pig.x + 2;
  drawSprites();
}
```



Lesson 13: Mini-Project - Animation

Criteria:

- ☐ Background
- ☐ Multiple Sprites with multiple properties in the draw loop
- ☐ Text
- ☐ Sprite Movement
- ☐ Multiple variables and values are updated during the program.
- ☐ At least one variable or property uses the counter pattern

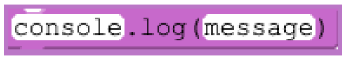
What kind of music do planets sing?
Neptunes!

Lesson 14: Conditionals

Vocabulary:

- Boolean Expression - in programming, an expression that evaluates to True or False.
- Condition - Something a program checks to see whether it is true before deciding to take an action.
- Conditionals - Statements that only run when certain conditions are true.

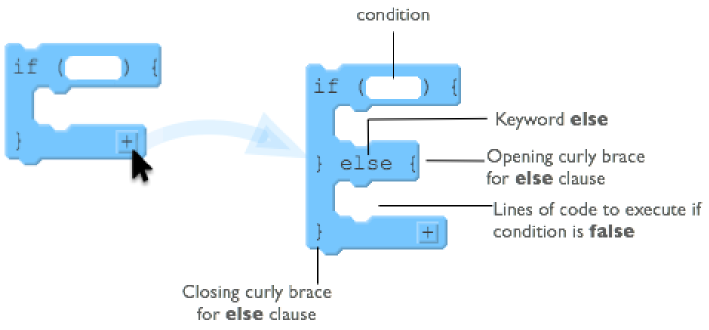
Introduced Code:



```
console.log(message)
```

___ != ___ ___ < ___ ___ <= ___ ___ == ___ ___ > ___ ___ >= ___

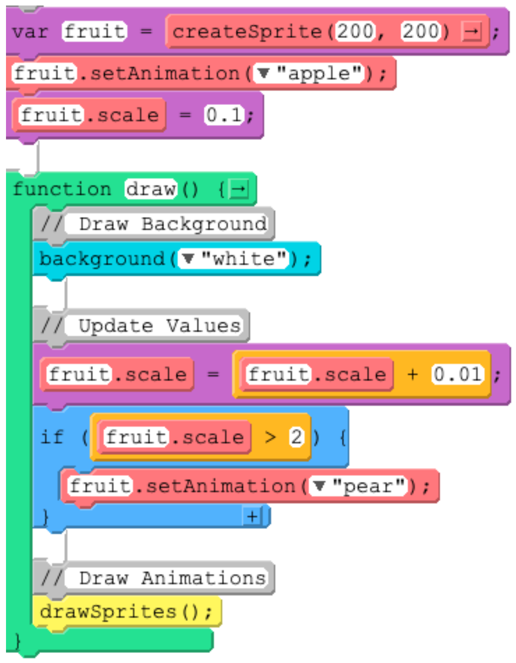
`if (condition) { statement }`



condition
Keyword **else**
Opening curly brace for **else** clause
Lines of code to execute if condition is **false**
Closing curly brace for **else** clause

Example:

↙creates sprite



```
var fruit = createSprite(200, 200);
fruit.setAnimation("apple");
fruit.scale = 0.1;

function draw() {
  // Draw Background
  background("white");

  // Update Values
  fruit.scale = fruit.scale + 0.01;
  if (fruit.scale > 2) {
    fruit.setAnimation("pear");
  }

  // Draw Animations
  drawSprites();
}
```

Lesson 15: Keyboard Input

Vocabulary:

- keyDown - detect whether a specific keys are being pressed down.

New Code:

`if (condition) { statement1 } else { statement2 }`

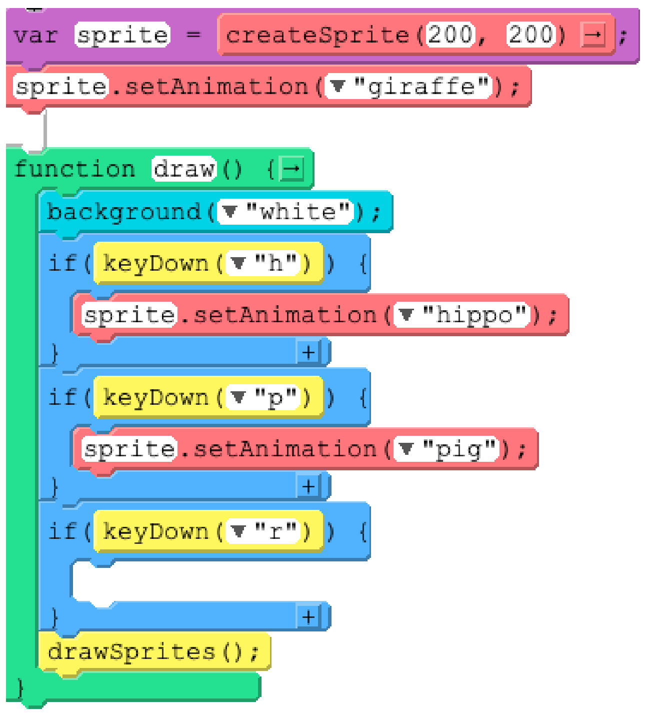
`keyDown(code)` – checks if the key specified is pressed.

`keyWentDown(code)` – generates a single true value when the key is pressed down, no matter how long a key is pressed.

`keyWentUp(code)` – checks if the key specified was released.

`mouseDown(button)` – checks if the mouse button specified is pressed.

Example:



```
var sprite = createSprite(200, 200);
sprite.setAnimation("giraffe");

function draw() {
  background("white");
  if (keyDown("h")) {
    sprite.setAnimation("hippo");
  }
  if (keyDown("p")) {
    sprite.setAnimation("pig");
  }
  if (keyDown("r")) {
  }
  drawSprites();
}
```

Lesson 16: Mouse Input

Vocabulary:

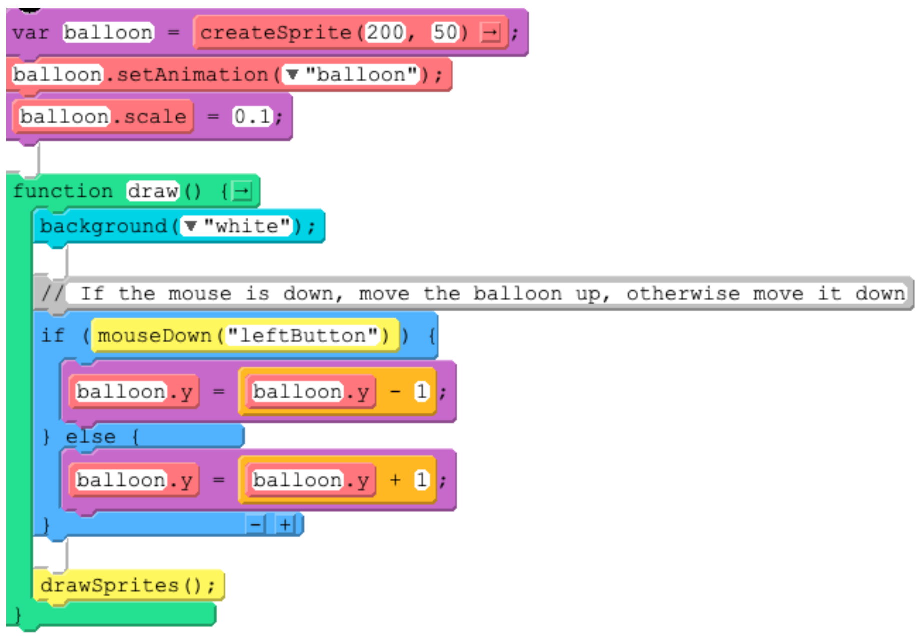
- mouseDown - checks if the mouse button specified is pressed.

New Code:

```
mouseDown(button)
```

Example:

↙sprite



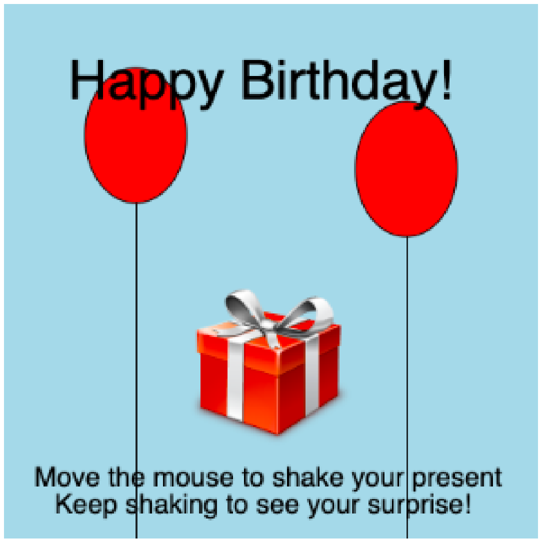
```
var balloon = createSprite(200, 50);
balloon.setAnimation("balloon");
balloon.scale = 0.1;

function draw() {
  background("white");
  // If the mouse is down, move the balloon up, otherwise move it down
  if (mouseDown("leftButton")) {
    balloon.y = balloon.y - 1;
  } else {
    balloon.y = balloon.y + 1;
  }
  drawSprites();
}
```

```
balloon.y = balloon.y - 1 → sprite moves up
balloon.y = balloon.y + 1 → sprite moves down
```

Lesson 17: Project - Interactive Card

Criteria:

- ☐ Background
- ☐ Draw Loop
- ☐ At least 1 random number
- ☐ Multiple sprites and set animation
- ☐ Multiple properties updated in the draw loop
- ☐ Multiple use input using key presses and mouse movements
- ☐ If block inside the draw loop
- ☐ Boolean comparison block (eg., <, >, ==)
- ☐ At least 1 variable or property uses the counter pattern


Happy Birthday!
Move the mouse to shake your present
Keep shaking to see your surprise!