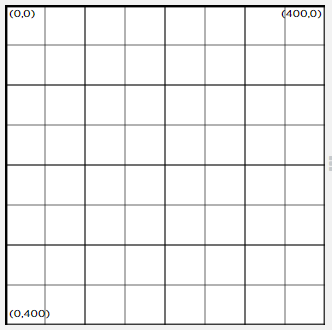
**Unit 3 Lesson 2- Plotting Shapes (computer grid)**



**Unit 3 Lesson 3- Drawing in Game Lab**

**Vocabulary:**

* But- 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.

**New Code:**

* fill(color)
* ellipse(x, y, w, h)
* rect(x, y, w, h)

**RGB- Red, Green, Blue**

**New Code**

* rgb(red, green, blue)

How to make any color:

1. Google ‘rgb’ calculator
2. rgb(0,0,0) = black
3. rgb(255, 255, 255) = white (255 is the largest number)

|  |  |
| --- | --- |
| **Colors I like** | **rgb(x,x,x)** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Unit 3 Lesson 4 – Shapes and Randomization**

**Vocabulary:**

* Parameter- an extra piece of information passed to a function to customize it

**New Code:**

* Background(color)
* randomNumber()

**Unit 3 Lesson 5- Variables**

**Vocabulary:**

* Variable- a placeholder for a piece of information that can change.
* 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:**

* Declare variable- var counter;
* Assign value to variable counter=0;

**Naming Rules:**

* No spaces
* Can’t begin with number
* Spelling counts
* Case-sensitive (Size does not equal size)

EXAMPLE

Var eyeSize;

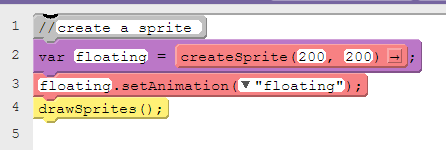
eyeSize = 20

**Unit 3 Lesson 6- Sprites**

|  |
| --- |
| Shapes: |
| background(color)  can use rgb values |
| rect(x, y, width, height) |
| ellipse(x, y, width, height) |
| line(x1, y1, x2, y2) |
|  |
| text(string, x, y, width, height)  string=text |
| textSize(pixels)  font size |
| Color and Style: |
| fill(‘color’) |
| noFill() |
| stroke(‘color’)  border color |
| noStroke() |
| strokeWeight()  thickness |

**Unit 3 Lesson 6 – Creating Sprites**

Creating Sprites



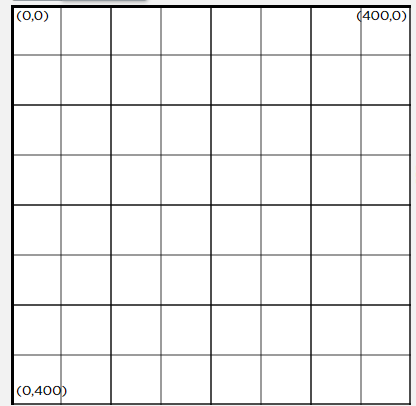
Line:

1. Is a note to the reader starts with //

2. The createSprite() block, which creates a new sprite at (200, 200) and assigns it to the variable label floating.

3. The sprite.setAnimation() block assigns an animation (or image) to the sprite.

4. Because sprites are just values stored as variables, they don't automatically get drawn on the screen. The drawSprites() block tells Game Lab to draw all of the sprites.



**Mini Project: Create a Gratitude Card**

Required elements: [Grab your reader’s attention with a great quote from the document or use this space to emphasize a key point. To place this text box anywhere on the page, just drag it.]

* Background
* At least 2 sprites
* Rectangle
* Ellipse
* Text

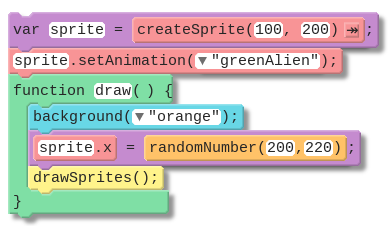
You will share your project with them.

**Unit 3 Lesson 7- The Draw Loop**

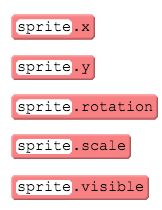
**Vocabulary:**

* 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.

Example:



**Sprite Properties:**

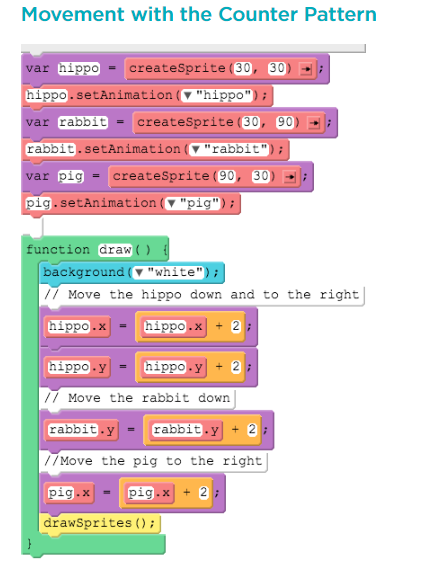


**Unit 3 Lesson 8- Counter Pattern**

**Vocabulary:**

* Expression- any valid unit of code that resolves to a value.
* Variable- a placeholder for a piece of information that can change.

**Unit 3 Lesson 9 – Sprite Movement**



**Unit 3 Lesson 10- Boolean Operators**

**Vocabulary:**

* Boolean- a single value of either TRUE or FALSE
* Conditionals- Statements that only run under certain conditions
* Expression – any valid unit of code that resolves to a value

**Unit 3 Lesson 11- Conditionals** **Boolean Operators**

|  |  |
| --- | --- |
|  |  |

Vocabulary:

* Boolean Expression- in programming, an expression that

evaluates to TRUE or FALSE

* If-Statement- The common programming structure that

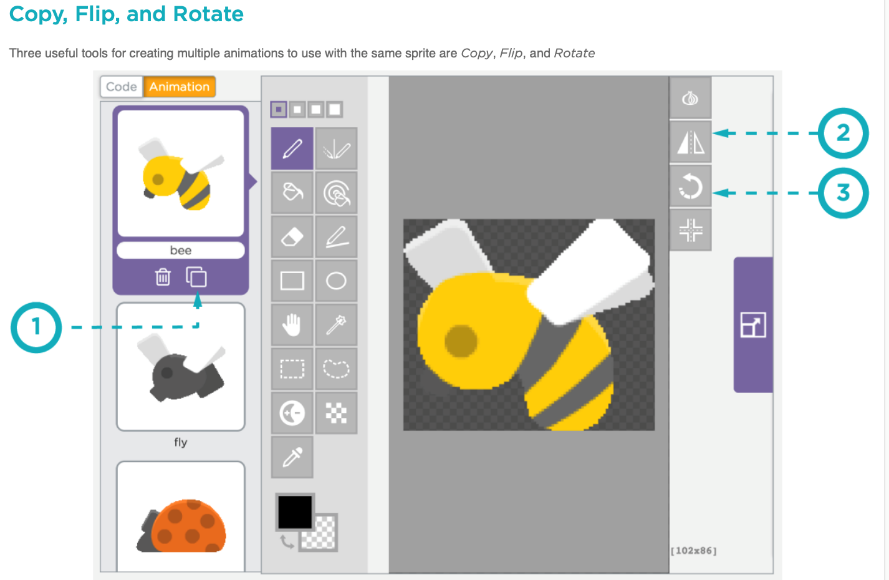
implements ‘conditional statements’.

\*\* Boolean Operators ask a question\*\*

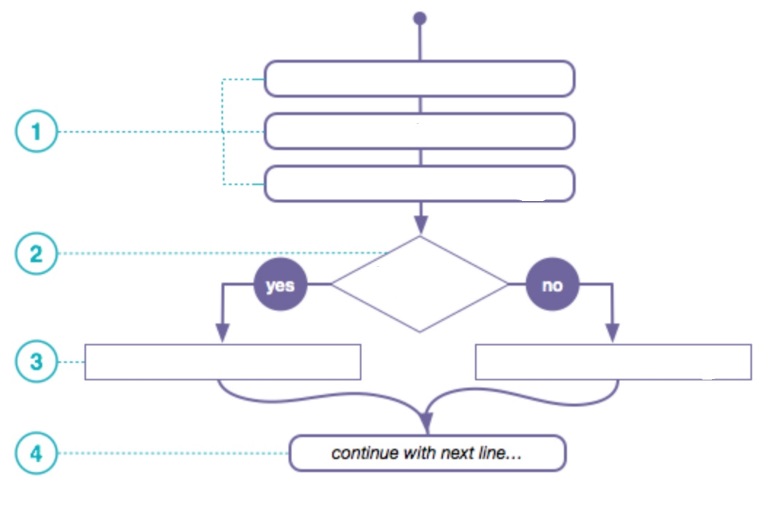
**Unit 3 Lesson 12- Keyboard Input (and Edit Sprites)**

**New Code:**

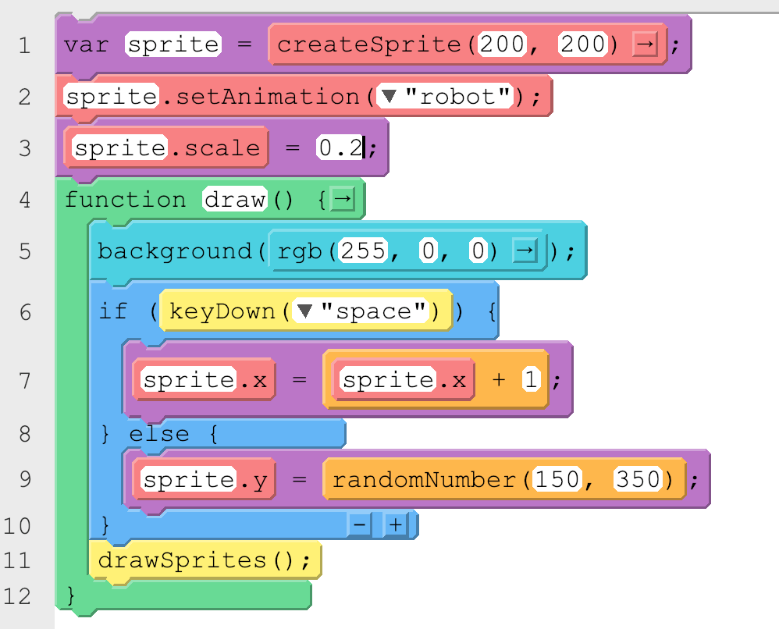
KeyDown(code) example: keyDown(“space”);



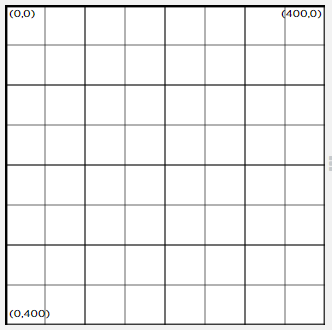
**Unit 3 Lesson 13- Other Forms of Input (mouse, If-Else block)**

  **Your example here ↑**

**Unit 3 Lesson 13- Other Forms of Input (mouse, If-Else block)**



**Unit 3 Lesson 14- Interactive Card Project**



|  |  |  |
| --- | --- | --- |
| Criteria | Yes/No | Description |
| Uses at least 3 sprites |  |  |
| At least one sprite responds to user input (eg. keyDown, mouseDidMove) |  |  |
| Updates at least 3 different sprite properties in the draw loop (eg. sprite.x, sprite.scale, sprite.visible) |  |  |
| Uses at least 1 conditional that is triggered by a variable or sprite property (eg. sprite.y > 300) |  |  |
| Increment or decrement a variable or sprite property (eg score = score + 1) |  |  |
| The card includes text that tells the user how to use the card |  |  |

**Unit 3 Lesson 15- Velocity**

**New Code:**

* sprite.rotationSpeed
* sprite.velocityX
* sprite.velocityY

**Helpful hints:**

* Rotation number is degrees- remember that there are 360 degrees in a full rotation
* Negative numbers – counter-clockwise
* Velocity will control the speed and direction
* Velocity blocks can be used inside and outside the draw loop

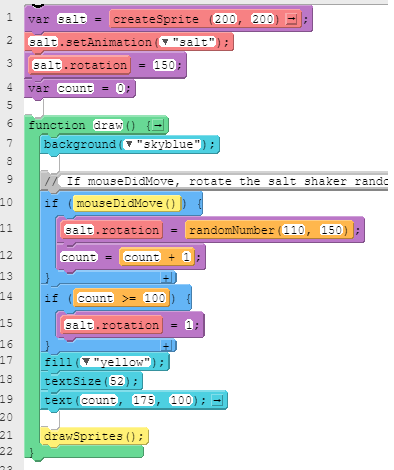
**Math Review:**

**=**

7 + -1

7 - 1

**Creating a Score**



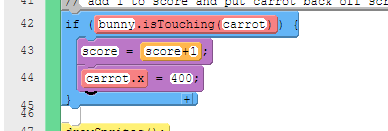
**Unit 3 Lesson 16- Collision Detection**

isTouching()

Writing out the math each time you want to check whether two sprites are touching can take a while, so a programmer created the isTouching block, which can check whether one sprite is touching another sprite (the target).

Scoreboard

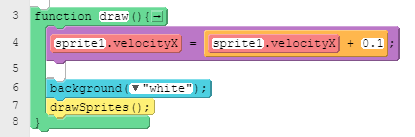
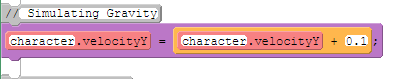
You can also use isTouching to decide whether you should increase the score. In this game, the score is stored inside the 'score' variable. It is displayed on the screen using the text block.



**Unit 3 Lesson 17- Complex Sprite Movement**

Velocity and the Counter Pattern

Using a sprite.velocityX property with the counter pattern will change a sprite's velocity during the program. This makes the sprite speed up.



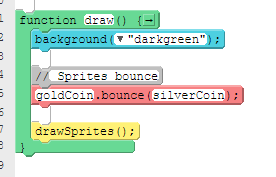
**Unit 3 Lesson 18- Collisions**

## displace

The displace block causes the sprite to **push** the target as long as they are touching each other. The sprite keeps moving normally.

## collide

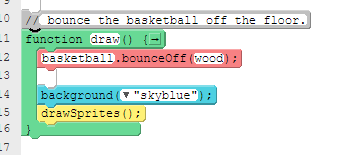
The collide block makes the **sprite stop** when it runs into the target. If the target is moving, it will push the sprite with it. The target keeps moving normally.



## bounce

The bounce block makes the **sprite and the target bounce** when they touch each other. Both the sprite and the target change how they are moving.

## bounceOff



The bounceOff block makes the sprite bounce off the target. The target keeps moving normally.

**Unit 3 Lesson 19 – Functions**

Write 5 step instructions for your morning routine.

1.\_

2.\_

3.\_

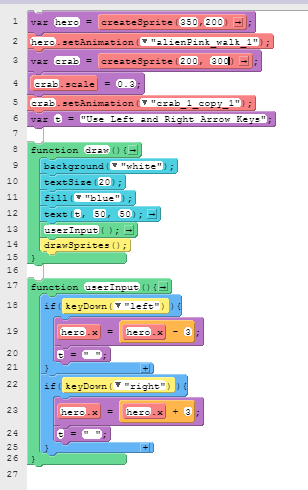
4.\_

5.\_

Take one of your steps above and break it down into 5 steps.

* 1. –
  2. \_
  3. \_
  4. \_
  5. \_

**Creating Text as a Variable**



**Unit 3 Lesson 22 – Final Project**

|  |  |  |
| --- | --- | --- |
| **Criteria** | **Yes/No** | **Comments** |
| At least three sprites |  |  |
| At least one animated sprite |  |  |
| At least one variable that is updated during the game (example: lives, score) |  |  |
| At least two backgrounds that are triggered to change by a conditional |  |  |
| The draw loop contains **functions** |  |  |
| The program uses whitespace, indentation, and comments |  |  |
| The game is playable and works as intentended |  |  |
| The game is creative and original |  |  |
| Directions are included that explain how to play the game. |  |  |

**Paragraph explain the theme of my game and how it will work:**

**Keyboard Short Cuts**

