# Text Based Coding (GameLab)

## Code.org:

- Code.org is an online platform for coding, especially for beginners in coding
- Since, it is an open source it can be accessed by anyone by just signing in
- Programs can be coded on this platform with the help of different labs it provides
- The labs are Sprite Lab, App Lab, Game Lab, Web Lab etc.
- These labs are based on various difficulty levels
- For our use we will be using the Game Lab

## Game Lab:

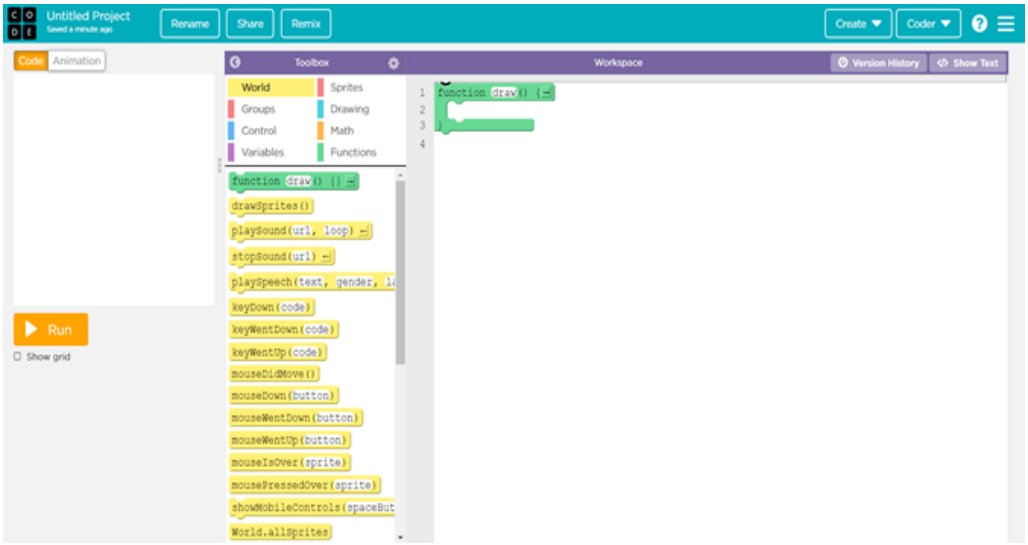
- Game Lab is a programming environment found in code.org
- In this lab, simple animations and games with objects and characters can be created that interact with each other
- The use of background, sounds, movements etc can be given to the game to make it interesting. The basic language used here is javascript
- The games created can later be published on the internet or played on a mobile phone

## Accessing Game Lab :

- Start code.org
- Login using your Gmail account
- Click on the create option
- Choose the “Game Lab” option

## Game Lab Interface:

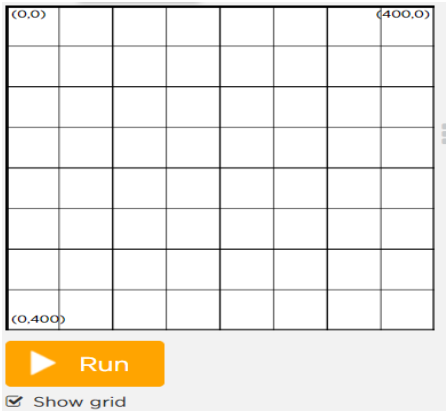
- Workspace : It is the main portion of the editor where the codes are written. It covers most of the screen.
- Output : It is located on the left side of the screen. The size of the output box is 400x400 units.
- Toolbox : It is located in between the workspace and the output. All the functions and commands are arranged here based on the property of the function. For example, inside ‘Drawing’ there will be background(), fill(), line() and other drawing related tools.




**Reference link:-** [https://www.youtube.com/watch?v=PXn9gKiKKFo](https://www.youtube.com/watch?v=PXn9gKiKKFo)

## Canvas and Coordinates: :

- The output screen is also called the canvas
- The size of the canvas is 400x400 units
- The default color of the canvas is white, but it can be changed to a different color using a function called background(“”)
- The canvas is a 2D coordinate system. So every point inside the canvas can be represented by x- axis position and y-axis position
- The exact coordinates can be found out by just moving the mouse over the canvas




## Coding styles :

In Gamelab, the codes can be written on the workspace in two formats :

1. Block- based : The codes appear to be inside colorful blocks in this format. It provides a definite structure to the code blocks so that the codes are easily readable by the developer. It is mainly for beginner level coders.
2. Text- based: It is a more developed style of writing codes. Here the codes are in simple text format.

It is possible to easily switch between the two styles. This is done by clicking on a button called show text/ show block on the top right corner. The codes can be directly dragged and dropped from the toolbox to the workspace.

## Inbuilt functions in Game Lab:

When a game is created, multiple tasks take place in it such as interaction between characters, between a user and the characters, some background environment, movements of the characters etc. All these events take place whenever the game is played. The commands or the functions required to give this effect must be written inside a function called- 'function draw()'.

**function draw(){} :**

- It is an execution function
- Executes the code inside the block until the program is stopped
- It is a predefined function

In simple words, the codes that are supposed to run throughout the game have to be written inside the

function draw(). Creating objects or setting animation are done only once hence it could be written outside the function draw().

**Reference link:-** https://www.youtube.com/watch?v=G6QJeuHhqCM

## Sprites :

**Definition:**

- Sprites are objects, which is available only in code.org
- Sprites are used to make complex and interesting animations and games
- A sprite is able to store images or animations with a set of properties such as position and visibility

**Create Sprites:**

- To create a sprite, first a variable should be declared by using the command 'createSprite()'
- Example: var object = createSprite();
- The parameters of this function are:
- var sprite = createSprite(x, y, width, height);
- If nothing is mentioned inside the function, then by default, the x-position and y-position are set to 0 and width and height are 100 each
- The default shape of a sprite is always a rectangle
- To make the sprite visible, drawSprites() function must be used inside the 'function draw()'.

**Reference link:-** https://studio.code.org/docs/gamelab/createSprite/

**Properties :**

- Properties are the values associated with the objects in Javascript
- Sprites are a collection of different properties like the position, velocity, scale, etc.
- The syntax is :

  Sprite.property
- The details along with the examples are given in the following reference :

  https://studio.code.org/docs/gamelab/

- Example 1 -

```
var object = createSprite();
function draw() {
background('white');
object.x = 200;
object.y = 200;
 drawSprites();
}
```

In the above example, when the sprite is created, no parameters are given inside the createSprite() function. Inside function draw(), the property of the x - position and y - position is given by the user.

- Example 2 -

```
var object = createSprite();
function draw() {
  background('white');
  object.velocityX = 2;
  object.velocityY = 2;
  drawSprites();
}
```

In the above example, the property of velocity is given to the sprite. The sprite starts moving diagonally since both x and y velocities are given. It is also observed that the sprite goes out of the canvas with that velocity.

**Edges :**

- Edges are used to give a boundary on the canvas
- The edges are not visible on the canvas
- The syntax to create edges is given as-
  createEdgeSprites()
- Edges are given to prevent the sprites from going out of the canvas
  For example:

```
var object = createSprite(200,200,20,20);
object.velocityY = 4;
createEdgeSprites();
function draw() {
  background("white");
  object.bounceOff(edges);
  drawSprites();
}
```

In the above example, an object is created, which is moving in horizontal direction with velocity 4. The bounceOff() property makes the object bounce off from the edges as createEdgeSprite() has been created.

## Other options in Toolbox:

**World :**

- The world toolbox consists of the commands that can be used independent of the sprites playSound(), keyDown(), mouseDown() etc. are some of the functions in the “World” toolbox
- Example -

```
var object = createSprite();
function draw() {
  background('white');
  mouse.x = World.mouseX;
  mouse.y = World.mouseY;
  drawSprites();
}
```

In the above example, a sprite named object is created, whose x and y positions are set according to the position of the mouse.

**Groups :**

- This toolbox is used to create groups and then add sprites to that group.
- To create a new group, the syntax is:

```
var group_name  = createGroup();
```

- After multiple sprites are created it is added to group with the following command:

```
group_name= add(sprite);
```

- Details regarding the groups will be explained in the next classes
- The different functions of the group are present in this toolbox

**Control :**

- The control toolbox contains: for loop, while loop, if and if else conditions
- Example 1 -

```
var object = createSprite(200,200,20,20);
function draw() {
  background('white');
  if(keyDown('LEFT_ARROW')){
    object.x = object.x - 2;
  }
  if(keyDown('RIGHT_ARROW')){
    object.x = object.x + 2;
  }
  drawSprites();
}
```

In the above block of code, the condition is that if the left arrow on the keyboard is pressed, then the object will move left. Same goes for the movement towards the right.

- Example 2 -

```
for (var i = 20; i < 400; i=i + 50) {
  var object = createSprite(200,i,20,20);
  object.setAnimation('apple_1_1');
  object.scale = 0.1;
}
function draw() {
  background('white');
  drawSprites();
}
```

In the above example, the property of 'sprite.setAnimation()' is used to give an image to the sprite. Then, it is reduced in size using the 'sprite.scale' property. The 'for loop' is being used to generate multiple objects vertically after a fixed interval.

**Variables :**

- The variable toolbox contains the commands to create, declare, and assign variables
- It also contains the console.log() command, which displays the input message on the console
- Example -

```
var object = createSprite(200,200);
console.log(object.width);
function draw() {
  background('white');
  drawSprites();
}
```

- In the above example, a variable is declared and assigned which is named as an object. The width of that sprite is displayed in the console.

**Drawing :**

- It consists of all the drawing tools such as background(), fill(), stroke(), text() etc.
- Shapes can also be drawn with these functions like rect(), ellipse(), shape() etc.
- Example -

```
function draw() {
  background('white');
  fill("red");
  textSize(25);
  text("Hello", 200, 80);
  fill("yellow");
  rect(180, 150, 80, 80);
}
```

In the above example, only drawing tools are used. A rectangle is drawn which is filled with yellow color. Also a text is displayed using the 'text('string', x, y)' function.

**Math :**

- It consists of all the mathematical operators like addition, subtraction, multiplication, as well as to find the minimum, maximum, round etc values of numbers
- It also has logical operators such as AND, OR, NOT etc
- Example -

```
var a=3, b=10, c, d;
c = Math.random(a,b);
console.log(c);
d= Math.round(c);
console.log(d);
```

In the above example, a random number is generated using the 'Math.random(lower_limit, upper_limit)' function. The 'Math.round()' function rounds it off to the nearest integer.

**Functions :**

- This tool box contains the block of codes to create a function and call a function
- As mentioned earlier, the 'function draw()' is pre-defined
- User defined functions can also be created and called

**Reference link:- https://studio.code.org/docs/gamelab/**

## Game States:

**World :**

- Game State helps us create many instances in a game like GameStart, GameOver etc
- The objects in the game behave differently at different GameState
- We can store the information about the state of a game using variables
- After storing, conditional programming is used to instruct the computer to behave differently for different states
- For example :
    - Start state- When a player just opens a game, he is not playing but the interface of the game will show along with the option to start the game.
    - Play state- It will be the state/stage where the player will interact with the characters in the game. When the player is controlling the game and it is in play mode.

    - End state- If the Player loses the game, he can no longer play, the game will stop.
    - Restart state- After the end state the game may have the option to start from the beginning. This will be the restart state.
- The game state is optional. It is completely dependent on the coder how he wants the game to be

## Sharing the game:

- Click on the Share option on the top panel of the game lab interface
- A link is provided which will directly run the game

## How to run on mobile::

- Click on 'Send to phone'
- The game will directly open on phone when the QR is scanned with the mobile camera
- The users outside the US can run on the mobile using the QR
- For US students/teachers the phone numbers can be entered

**********