Lecture note

Lecture one and two are more the introductory part so let’s jump to another lecture

Lecture 3

To make any shape like rectangle or different primitive

Example:- (0,0(position),1,1(size)) the first 2 numbers are for position and 2nd numbers are for size.

Coloring shapes

* Every shape has two elements that can be colored stroke and fill.
* stroke is the border of the shape, and fill is the color inside the shape.
* Example :- Fill(“Green”);

Stroke(“Green”)

If you don’t want your shapes to have a stroke or fill you can use no stroke() or no fill() commands.

* Order of commands matters a lot you need to change the color of your stroke or fill before you draw the shape.

Other commands to fill

->no fill ();

->stroke weight(size);

->no stroke ();

->ellipse (x,y,z,w,h);

->fill(“color”);

->text (text,200,200,);

->stroke(“color”)

->rect(0,0,200,200);

->ellipse(x,y,w,h)

 **Bug**: A flaw or error in a program that causes it to behave unexpectedly or incorrectly.

 **Debugging**: The process of identifying, isolating, and fixing bugs within a program to ensure it runs smoothly.

 **Program**: A set of instructions or an algorithm that has been written in a programming language and can be executed by a computer

In Lesson 4, I learned about shapes and parameters in Game Lab, focusing on how to customize rectangles and ellipses beyond their default settings. The rect block, which initially creates a square with sides of length 50, has two required inputs for x and y coordinates to define the position of the square's top-left corner. However, it also includes two optional parameters for width and height, which can be revealed or hidden using the arrow on the right-hand side of the block. This flexibility allows for precise control over the size and dimensions of rectangles.

Similarly, the ellipse block has optional width and height parameters in addition to its required x and y inputs, which specify the center of the ellipse. These optional parameters can also be toggled using the arrow on the block, making it easy to create circles or ellipses with customized dimensions. This lesson enhanced my understanding of using parameters to create shapes with more versatility and control.

In Lesson 5, I learned about variables and their importance in programming. A variable acts as a label for storing information, such as numbers, text, or positions of shapes, that can be reused or modified throughout a program. Variables are fundamental for making programs dynamic and efficient.

The key benefits of using variables include reusability, as the same variable can be used in multiple places without repeating code; flexibility, since updating the value of a variable automatically applies changes wherever it is referenced; and simplified code, which becomes easier to read, understand, and debug. This lesson highlighted how variables are essential tools for organizing and streamlining code in any program.

In this lesson, I learned about using random numbers in programming to introduce variability and unpredictability. The randomNumber() block allows the generation of random values within a specified range, making it useful for creating dynamic and ever-changing outputs. By incorporating random numbers as parameters in drawing commands, I discovered how to make graphics vary with each program run. Additionally, I learned that using multiple instances of randomNumber() generates independent random values, enabling the creation of more complex and randomized designs. This concept enhances creativity and flexibility in coding, allowing for more engaging and interactive projects.