Week 1 Reflection

This week, I began learning the fundamentals of ROBLOX development. I focused primarily on how variables are declared, the mathematical operators, as well as the syntax for the language in general. A few things stuck out immediately:

- There are no brackets in lua
- Functions and statement close with “end”
- There is no direct “typing” in lua for variables.
- Indentation doesn’t matter, but it sure helps, especially with there being no brackets
- Comments begin with “—” and span the entire line
- Using the correct way to “path” your references is important. Using :WaitForChild(””) is the best way to do it, otherwise the server won’t wait for the object. It’ll just pass through.
- task.wait() is more efficient than wait(), and it’s the new standard

The syntax is very different from any other language I’ve programmed in before as well. In order to iterate through a list of children within a for loop, you must declare:

```
for i,v in pairs(folderChildren) do
  -- i is the index of the child
  -- v is the child object itself
end
```

The lua-ROBLOX language also appears to be generally centered around attribute and property manipulation. Whether it’s manipulating the characteristics of the player, or the physics within the game, it all has to do with manipulation.

I must say, remembering the syntax hasn’t been easy for me so far, but I imagine that over time it’ll get a bit easier. The ROBLOX code editor itself is easy to understand and

provides ample amount of assistance if required. I’m not relying on autocomplete yet though, I want to get everything down first.

An important note

Using a while loop without a task.wait() function will crash your code editor as well as your computer. For the first few days of programming in LUA, I was doing this constantly.

For example this would crash my computer,

```
local function ahmanWrong()
  while true do
    print("ahman")
  end
end

-- the right way to do it
local function ahmanRight()
  while task.wait() do
    print("ahman")
  end
end
```

Important Note

ROBLOX is also unique within its niche of game development also because you’re able to design and build your own assets, (example: a house), directly from the same application in which you’re writing your code. This is an upside, but it’s also been a downside for me in regards to organization. Combining scripts with part objects isn’t always the most straightforward task, especially when the attributes or location of the part objects can change.

Something that I thought about was the idea of the “race condition.” I’m not sure if it directly applies here, but the philosophy behind it is that I don’t want my code to be reliant on a part attribute that may be changed by other scripts. Thus, modularity within the organization of code is important.