Notes for New Pong

Yanuar Heru Prakosa

30-05-2021

1 What Are We Doing

This project is basically to learn the basic of game development. I think this is good way on tracking the development step by step. Since the original course uses different folder each session it is kind of hassle. I want to make the whole development step into one project or at least one folder.

2 Class in Lua

By default Lua as programming language does not have class. I need to find another alternative if I want to utilize the object oriented programming concept to the coding. Here is the beginning of the learning link. This wiki page of lua-users talk about Simple Lua classes concept.

From the official Lua website documentation there are also some papers that talk about simulating class behavior. We can start from the content of the Object Oriented Programming docs. From there we can follow the link that specific talking about classes. For simplicity sake I will put the link the Classes official documentation page.

There is also one more source in GitHub which looks like making a construct of a Lua class. Here is the link to the repo. I think this might be a good chance to learn more about how the data structure of Lua and in this case used by Love2D really works.

2.1 Learning Classes from Lua Official Docs

Lua does not have a concept of class as part of its programming language. However, like JavaScript, Lua is a prototype based language. Meaning the mold (prototype) is just an object like any other object in the code. Class is separate entity hence to use it must be instantiated. Prototype on the other hand is just an object like any other which also subject of changes and implementation.

The class emulator is basically a table. Reminder: table is the only data structure available in Lua. Table is more like a Python dictionary but in table all components are in index. Thus you can call an object in the table using its key and/or its index.

```

        Account = {}
        Account.__index = Account

        function Account:create(balance)
        local acnt = {}
        setmetatable(acnt, Account)
```

```lua
acnt.balance = balance
return acnt
end

function Account:deposit(amount)
self.balance = self.balance + amount
end

function Account:getBalance()
return self.balance
end

--test create Account
bill = Account:create(1000)
print (bill:getBalance())
bill:deposit(5000)
print (bill.balance)
-- test consistency
sec = Account:create(55)
print (sec)
print (bill.balance)

```

WARNING: the variable inside the class object is all public. I can just access the balance of the bill and sec account just by calling the direct variable name in the table. Moreover I can just change the value of the variable without ever using setter method!