Erasmus+
2020-1-UK01-KA201-079177

7. Scripting Introduction

INTRODUCTION

- ✔ Scripts can make an object move, listen, talk, operate as a vehicle or weapon, change color, size or shape.
- ✔ A script can make an object listen to your words as well as talk back to you, scripts even let objects talk to each other.
- ✔ If you have built in Second Life/ OpenSimulator, everything you can define in the edit window can be defined in a script. All interaction you see between objects or between avatars and objects is via scripts.

INTRODUCTION

- ✔ Scripting is harder to learn than basic object manipulation, but is very rewarding once you make progress. For creating scripts it can be helpful to have a local copy of them on your local hard-drive as they're basically just text and edit them with one of the alternate editors.

LSL LANGUAGE

- ✔ LSL is the Linden Scripting Language. This is the language all scripts in OpenSimulator are written in.
- ✔ The structure of LSL is largely based on Java and C, both of which are widely used programming languages in the real world.
- ✔ A script in Second Life/ OpenSimulator is a set of instructions that can be placed inside any primitive object in the world, but not inside an avatar.
- ✔ Avatars, however, can wear scripted objects. LSL scripts are written with a built-in editor/compiler.

SCRIPTS

- ✔ To create a script, select an object and go to the Contents tab of the edit menu. Select the ‘create script’ option and open it.
- ✔ An object can have more than one script files that control it. If you have a linked set, you can have scripts in the ROOT prim, but also in the individual parts of it.
- ✔ In general, LSL commands you run in the ROOT prim, will affect the whole object, while scripts in the individual/child prim only affect that parts.

STATES

- ✔ A script file may contain one or more states, but only one of them will be active at a certain point.
- ✔ A state is a block of code that describes how the object behaves.
- ✔ More specifically, the state contains one or more event blocks that specify when something will happen (e.g. when someone clicks the object) and inside the event block there are the commands that specify the actions that will happen then (e.g. change the color of the object).

STATES

- If there are multiple states defined, you can use the state command to move from one state to another. Each state can have completely different event blocks, so having multiple states is useful when you want the behavior of the object to radically change at some points.

EXAMPLE

- So, here is a very simple example of script:

```
{
  state {
    touch_start(integer num) {
      llSay(0,”Hello! You clicked me!”);
    }
  }
}
```

EVENTS

- Lets start with some of the most common events:
  - **state_entry:** Event triggered when you enter the state, including when the script is reset/modified.
  - **touch_start:** Event triggered when a user clicks on the object
  - **collision:** Event triggered when an avatar collides with the object
  - **sensor:** Event triggered when an avatar gets near (specific radius) the object
  - **timer:** Event triggered periodically based on specified interval
  - **listen:** Event triggered when a message is sent in a channel the object waits for messages
  - **link_message:** Event triggered when a message is sent to the specific part of a linked set

FUNCTIONS

- ✔ Functions lay inside of events and are either defined by you or are built-in functions. Built in functions start with two lowercase L's such as: llSay(). Functions take "arguments" or values in the parentheses that follow it. LSL as a language uses pass-by-value for all types.

FUNCTIONS

- Here are some useful actions you can perform and the corresponding LSL functions to do it.
  - Send Chat Message (llSay)
  - Change color(llSetColor)
  - Change texture (llSetTexture)
  - Change Transparency (llSetAlpha)
  - Change Position (llSetPos)
  - Rotate Object (llTargetOmega)
  - An object is given to the avatar (llGiveInventory)
  - Object stops/sleeps X seconds between two actions (llSleep)

VARIABLES TYPES

- The LSL language supports the following variable types:
  - **Integer** (an integer number)
  - **String** (a string value)
  - **Float** (a floating point number)
  - **List** (set of values that can be of different types)
  - **Vector** (a group of three float values that is commonly used for Positions, Velocity and Colours)
- You can define global variables outside of states that will be accessible inside them.

FLOW CONTROL

- Flow control inside a function or event is similar to other programming languages. You can control the flow of the commands with:
  - if / else
  - for, while, do-while
  - jump, return
  - state

OPERATORS

- The operators used are also similar to other languages:
  - Arithmetic operators: + , - , * , / , %
  - Logical operators: && , || , == , !=