

Second Life - Beginner Scripting

Overview

Participants will add a Script to a Homework Drop Box to Collect Documents from Students, Send an Email, or Teleport Students to a Classroom Location.

Add an 'Email Me' script in a Mailbox and Create their own Pose Ball. Prerequisite: Avatars will NOT be created in this class. Please see me before class, create your own or use an ATS Avatar. Supplies: Microphone, speakers, headset and software provided in class. Optional: Thirty minute lab time available immediately following 60 minute workshop.

Objectives

Participants will:

- Create a simple touch script
- Identify the scripting properties of a pose ball
- Receive and Add an "Email Me" script to an object

Table of Contents

Contents

Second Life - Beginner Scripting	1
Table of Contents	2
Contents	2
LSL Linden Scripting Language	3
Default State	6
Ann Enigma's Autoscript (Enigma)	7
Particles-LSL-Generator	7
Scratch for Second Life	7
Example Scripts	7
Using the Linden Script Language	8

LSL Linden Scripting Language

Here is the default script. This is auto generated by Second Life when you Open the Inventory and Select Create Script.

```
Default
{
    State-entry()
    {
        11Say(0, "Hello, Avatar!");
    }
    Touch_start (integer total_number)
    {
        11Say(0, "Touched.");
    }
}
```

This script starts with the word default. The word default specifies the name or state of or you might say the condition of the object. For this script there is only one state. This state, which is named default, is the starting state for any script in Second Life.

If the script is to cause the object to perform two actions – a script can be created by using one state and an ‘if’ or ‘else’ statement or by using two states. Here is an example of both....It is

```
Default
{
    State_entry()
    {
        Value = TRUE;
    }
    Touch_state (integer total_number)
    {
        If ( value==TRUE )
        {
            11Say (0, "On");
            Value = FALSE;
        }
        Else
        {
            11Say (0, "Off");
            Value = TRUE;
        }
    }
}
```



preferred that two states are used rather than 'if' or 'else'....either works but coders say that state engines should be used when possible.

Script constructed entirely within its default state:

A variable, named value, is set to either TRUE or FALSE. As the user touches the object, the object will say either "On" or "Off". As the object is touched these values alternate.

This is how coders prefer to write this action.

```
Default
{
    touch_state (integer total_number)
    {
        11Say (), "On");
        State off;
    }
}

state off
{
    touch_start (integer total_number)
    {
        11Say(0, "Off");
        State default;
    }
}
```

On/Off Example:

```
default //default state is mandatory
{
    state entry() // runs each time the state is entered
    {
        llSay(0, "turning on!"); //object speaks!
        llSetColor(<1.0, 1.0, 1.0>, ALL SIDES); // sets all sides
to most bright
        // note the semicolons at the end of each instruction (do
not put them at the end of if statements)
    }

    touch start(integer total_number) // another event with only
one function inside
    {
        state off; // sets the script to a new "state" and starts
running "state off"
    }
} // this curly bracket ends the body of the default state.

state off // a second state besides "default"
{
    state entry() // this is run as soon as the state is entered
    {
        llSay(0, "turning off!");
        llSetColor(<0.0, 0.0, 0.0>, ALL SIDES); // sets all sides
as dark as possible
    }

    touch start(integer total_number)
    {
        state default;
    }
}
```



Default State

Let us examine the default state.

First we see the "[state_entry](#)" event, which gets triggered each time the default state is entered. Which in this case is entered the first time the script is run.

SPEAK TO ME!

The first line in the event `state_entry` is...

```
llSay(0, "turning on!");
```

This makes the object speak "turning on!" on channel zero. What is channel zero? It is the same channel you see all public chat on.

A semicolon ends the line and yet another instruction follows.

```
llSetColor(<1.0, 1.0, 1.0>, ALL_SIDES);
```

This turns the prim to its brightest tint. If you take the texture off the prim, you would see it as bright white but with a texture, it looks "normal." The three ones stand for the Red, Green, and Blue values of the tint.

At this point the event is finished with the two lines of commands. Then the script waits idle in the default state for more events to happen.

TOUCHED BY AN AVATAR

While idle in the default state a touch will trigger the "[touch_start](#)" event.

Inside of the "touch_start" event is only one command:

```
state off;
```

This is a command to move immediately to a new state named "off".

This state is defined after the default state and nearly mirrors the default state, except that it turns the prim dark and, when touched, will put the script back into default mode, thus creating a loop.

Enters default state



Runs code in "state entry"
Waits to be touched
When touched enters "state off"
Enters "state off"
Runs code in "state entry" (note in the "off" state's body)
Waits to be touched
When touched enters "default" state
Then the whole thing starts over.

Second Life scripting goes from elementary to extremely advanced. I know firsthand that many of us are curious, but fear getting started.

Ann Enigma's Autoscript (Enigma)

Try <http://www.3greeneggs.com/autoscript/> - Ann Enigma's Autoscript, a wonderful tool which has made a difference for many Residents. With a few clicks, you can make a script, then examine and edit it further. Only problem is: it hasn't been getting enough exposure, so a lot of people who could've been helped by it... didn't know it existed!

Particles-LSL-Generator

This formatted webpage may be easier to use than SL's own script editor. Change your parameters and try it out inworld.

<http://particles-lsl-generator.bashora.com/index.php>

Scratch for Second Life

Eric Rosenbaums's exciting app (for Mac and Windows) in development which lets you arrange scripts visually as if they were Legos.

http://web.mit.edu/~eric_r/Public/S4SL/

Example Scripts

Here are examples of LSL scripts you may find useful. For more, try the ScriptLibrary page or the Script Library forum on the Second Life website.

You may use these scripts freely as they stand, or as a base to build your own. Taking them and selling them would be considered impolite, however. If you want to use them in a "commercial" script, talk to the original author first.



<http://lslwiki.net/lslwiki/wakka.php?wakka=examples>

Using the Linden Script Language

This page is a short tutorial on using the Linden Script Language (LSL). It includes a collection of examples that illustrate basic LSL capabilities in graphics, "physics," communication between users and scripts, and object creation. Examples are chosen to present fairly isolated "building blocks" that may be combined later to create more complex scripts, and to better understand the [Examples](#) scripts and [Script Library](#) presented on the Wiki.

<http://people.cc.ku.edu/~grobe/intro-to-LSL/index.html#texture>