

Source Code for Troy McConaghy's Holo-Emitter System

by Troy McConaghy

Introduction

This document contains the source code for a holo-emitter system that I made for Second Life. For more information about the system and a *User's Guide*, see my blog at:

<http://www.troymcconaghy.com/blog/2007/9/12/open-source-holo-emitter-for-sl.html>

The Holo-Emitter System

There are four scripts that work together to make the holo-emitter system work. They are named:

Holo-Emitter Script
Scene-Container Script
Object Script A
Object Script B

Please make sure you have all four scripts or the system won't work. The source code for those scripts (version 1.0) is below. Note that they are all open source (GPL 2.0).

Holo-Emitter Script

```
// Holo-Emitter Script
// Part of Troy McConaghy's Holo-Emitter System
//
// Copyright (C) 2007 by Troy McConaghy (Troy McLuhan in Second Life)
//
// This program is free software; you can redistribute it and/or
// modify it under the terms of the GNU General Public License
// as published by the Free Software Foundation; either version 2
// of the License, or (at your option) any later version.
//
// This program is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
// GNU General Public License for more details.
//
// You can get a copy of the GNU General Public License at
// http://www.gnu.org/licenses/gpl.txt
// or by writing to the Free Software Foundation, Inc.,
```

```

// 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.

// Global constants - should be changed if there are any other holo-emitters in the
sim, both here and in all other scripts
integer delete_channel = -3423489;
integer menu_channel   = -1847395;
integer HE_channel     = -9385735;

// Global variables
list options;
integer menu_listen;

clear_scene()
{
    llRegionSay(delete_channel, "delete"); // Sends a "delete" message to the
entire sim on the delete_channel
}

default
{
    state_entry()
    {
        llSay(0, "Ready");
        llListen(HE_channel, "", NULL_KEY, "ping");
    }

    touch_start(integer total_number)
    {
        // Generate the menu of scenes by scanning the holo-emitter's inventory
        options = ["Clear"]; // even if there are no scenes in the holo-emitter,
one option is always to clear the scene
        integer num_scenes = llGetInventoryNumber(INVENTORY_OBJECT); // = number of
objects in the holo-emitter's inventory
        if (num_scenes>0)
        {
            integer scene_nbr;
            for (scene_nbr=0; scene_nbr<num_scenes; scene_nbr++)
            {
                options = options + [llGetInventoryName(INVENTORY_OBJECT,
scene_nbr)]; // appends another scene to the list of options
            }
        }

        llDialog(llDetectedKey(0), "Select the scene you would like to see:",
options, menu_channel);
        menu_listen = llListen(menu_channel, "", NULL_KEY, "");
    }

    listen(integer channel, string name, key id, string message)
    {
        if ((channel==HE_channel) && (message=="ping"))
        {
            //llSay(0, "I heard a ping, so I'm replying with my position now (on the
HE_channel).");
            vector HE_pos = llGetPos(); // = holo-emitter position <x,y,z>
            string str = (string)HE_pos.x+" "+(string)HE_pos.y+" "+
(string)HE_pos.z;

```

```

        llRegionSay(HE_channel, str);
        //llSay(0, "My position string is "+str);
    }
    else if (channel==menu_channel)
    {
        if (message=="Clear")
        {
            clear_scene();
        }
        else // message = the name of the scene-container to rez
        {
            clear_scene();
            llRezAtRoot(message, <0.0,0.0,9.0>+llGetPos(), ZERO_VECTOR,
ZERO_ROTATION, 1);
        }

        llListenRemove(menu_listen);
    }
}
}

```

Scene-Container Script

```

// Scene-Container Script
// Part of Troy McConaghy's Holo-Emitter System
//
// Copyright (C) 2007 by Troy McConaghy (Troy McLuhan in Second Life)
//
// This program is free software; you can redistribute it and/or
// modify it under the terms of the GNU General Public License
// as published by the Free Software Foundation; either version 2
// of the License, or (at your option) any later version.
//
// This program is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
// GNU General Public License for more details.
//
// You can get a copy of the GNU General Public License at
// http://www.gnu.org/licenses/gpl.txt
// or by writing to the Free Software Foundation, Inc.,
// 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.

default
{
    on_rez(integer start_param)
    {
        string obj_desc = llToLower(llGetObjectDesc()); // = object description
string, made lowercase
        //llSay(0, "obj_desc = "+obj_desc);

        if (obj_desc == "live") // then the scene-container is live, so rez all of
its contents then delete the scene-container
        {
            integer num_obj = llGetInventoryNumber(INVENTORY_OBJECT); // = number
of objects in the scene-container's inventory
            //llSay(0, "Number of objects in my inventory = "+(string)num_obj);

```

```

        if (num_obj>0)
        {
            integer obj;
            for (obj=0; obj<num_obj; obj++)
            {
                //llSay(0, "Rezzing obj with inventory index = "+(string)obj);
                llRezAtRoot(llGetInventoryName(INVENTORY_OBJECT, obj),
<0.0,0.0,1.0>+llGetPos(), ZERO_VECTOR, ZERO_ROTATION, 0);
                llSleep(3.0);
            }
            //llSay(0, "Done rezzing all objects in inventory");
        }
        llSleep(10.0);
        llDie();
    }

    // If the scene-container is not live, then control now passes to the
state_entry() event handler
}

state_entry()
{
    llSay(0,"You can now add and remove objects from my inventory.");
}

}

```

Object Script A

```

// Object Script A
// Part of Troy McConaghy's Holo-Emitter System
//
// Copyright (C) 2007 by Troy McConaghy (Troy McLuhan in Second Life)
//
// This program is free software; you can redistribute it and/or
// modify it under the terms of the GNU General Public License
// as published by the Free Software Foundation; either version 2
// of the License, or (at your option) any later version.
//
// This program is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
// GNU General Public License for more details.
//
// You can get a copy of the GNU General Public License at
// http://www.gnu.org/licenses/gpl.txt
// or by writing to the Free Software Foundation, Inc.,
// 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.

// Global constants - should be changed if there are any other holo-emitters in the
sim, both here and in all other scripts
integer HE_channel = -9385735;

// Global variables
vector rel_pos;
rotation rot;

```

```

goto( vector destpos )
{
    //Based on warpPos
    //R&D by Keknehv Psaltery, 05/25/2006
    //with a little pokeing by Strife, and a bit more
    //some more munging by Talarus Luan
    //Final cleanup by Keknehv Psaltery
    // Compute the number of jumps necessary
    integer jumps = (integer)(llVecDist(destpos, llGetPos()) / 10.0) + 1;
    // Try and avoid stack/heap collisions
    if (jumps > 100 )
        jumps = 100;    // 1km should be plenty
    list rules = [ PRIM_POSITION, destpos ]; //The start for the rules list
    integer count = 1;
    while ( ( count = count << 1 ) < jumps)
        rules = (rules=[]) + rules + rules; //should tighten memory use.
    llSetPrimitiveParams( rules + llList2List( rules, (count - jumps) << 1,
count) );
}

ping()
{
    llRegionSay(HE_channel, "ping"); // Sends out a ping on the HE_channel to
determine the holo-emitter's position
}

default
{
    on_rez(integer start_param)
    {
        state repos;
    }

    state_entry()
    {
        // This event is triggered if this script is dragged from inventory to the
contents of an object
        llListen(HE_channel, "", NULL_KEY, "");
        ping();
        llSetTimerEvent(2.0); // Just in case the first ping isn't heard, more are
sent out until one gets heard
    }

    timer()
    {
        ping();
    }

    listen(integer channel, string name, key id, string message)
    {
        if (channel==HE_channel)
        {
            llSetTimerEvent(0.0); // Turns off the timed pings, since a ping was
heard and a reply has been received

            vector pos = llGetPos();

```

```

        if (pos.z <= llGround(ZERO_VECTOR))
        {
            string obj_name = llGetObject_name();
            llSay(0,"WARNING! The center of the object "+obj_name+" is below
ground level.");
            llSay(0,"The holo-emitter system won't be able to put it back
there");
            llSay(0,"because it can only place objects whose center is above
ground level.");
        }

        list pos_list = llCSV2List(message);
        vector HE_pos = <llList2Float(pos_list,0), llList2Float(pos_list,1),
llList2Float(pos_list,2)>;
        rel_pos = pos - HE_pos;

        //llSay(0,"pos = "+(string)pos+", HE_pos = "+(string)HE_pos+", rel_pos
= "+(string)rel_pos);

        rot = llGetRot();
        //llSay(0, "I stored my position in rot, rot = "+(string)rot);

        state await_rerez;
    }
}

state await_rerez
{
    state_entry()
    {
        llSay(0,"You can now take me into your inventory and then put me in a
scene-container from there.");
    }

    on_rez(integer start_param)
    {
        state repos;
    }
}

state repos
{
    state_entry()
    {
        //llSay(0,"Now in state repos");
        llListen(HE_channel, "", NULL_KEY, "");
        ping();
        llSetTimerEvent(2.0); // Just in case the first ping isn't heard, more are
sent out until one gets heard
    }

    timer()
    {
        ping();
    }

    listen(integer channel, string name, key id, string message)

```

```

    {
        if (channel==HE_channel)
        {
            llSetTimerEvent(0.0); // Turns off the timed pings, since a ping was
heard and a reply has been received

            list pos_list = llCSV2List(message);
            vector HE_pos = <llList2Float(pos_list,0), llList2Float(pos_list,1),
llList2Float(pos_list,2)>;
            vector pos = HE_pos + rel_pos;

            llSetRot(rot);
            goto(pos);
            llSetRot(rot);
            goto(pos);
            llSetRot(rot);
            goto(pos);
            llSetRot(rot);

            // Now that the object is positioned, this script isn't needed anymore,
so remove this script from the object's inventory
            llRemoveInventory(llGetScriptName());
        }
    }
}

```

Object Script B

```

// Object Script B
// Part of Troy McConaghy's Holo-Emitter System
//
// Copyright (C) 2007 by Troy McConaghy (Troy McLuhan in Second Life)
//
// This program is free software; you can redistribute it and/or
// modify it under the terms of the GNU General Public License
// as published by the Free Software Foundation; either version 2
// of the License, or (at your option) any later version.
//
// This program is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
// GNU General Public License for more details.
//
// You can get a copy of the GNU General Public License at
// http://www.gnu.org/licenses/gpl.txt
// or by writing to the Free Software Foundation, Inc.,
// 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.

// Global constants - should be changed if there are any other holo-emitters in the
sim, both here and in all other scripts
integer delete_channel = -3423489;

default
{
    state_entry()
    {

```

```
    llListen(delete_channel, "", NULL_KEY, "delete");
}

listen(integer channel, string name, key id, string message)
{
    llDie();
}
}
```