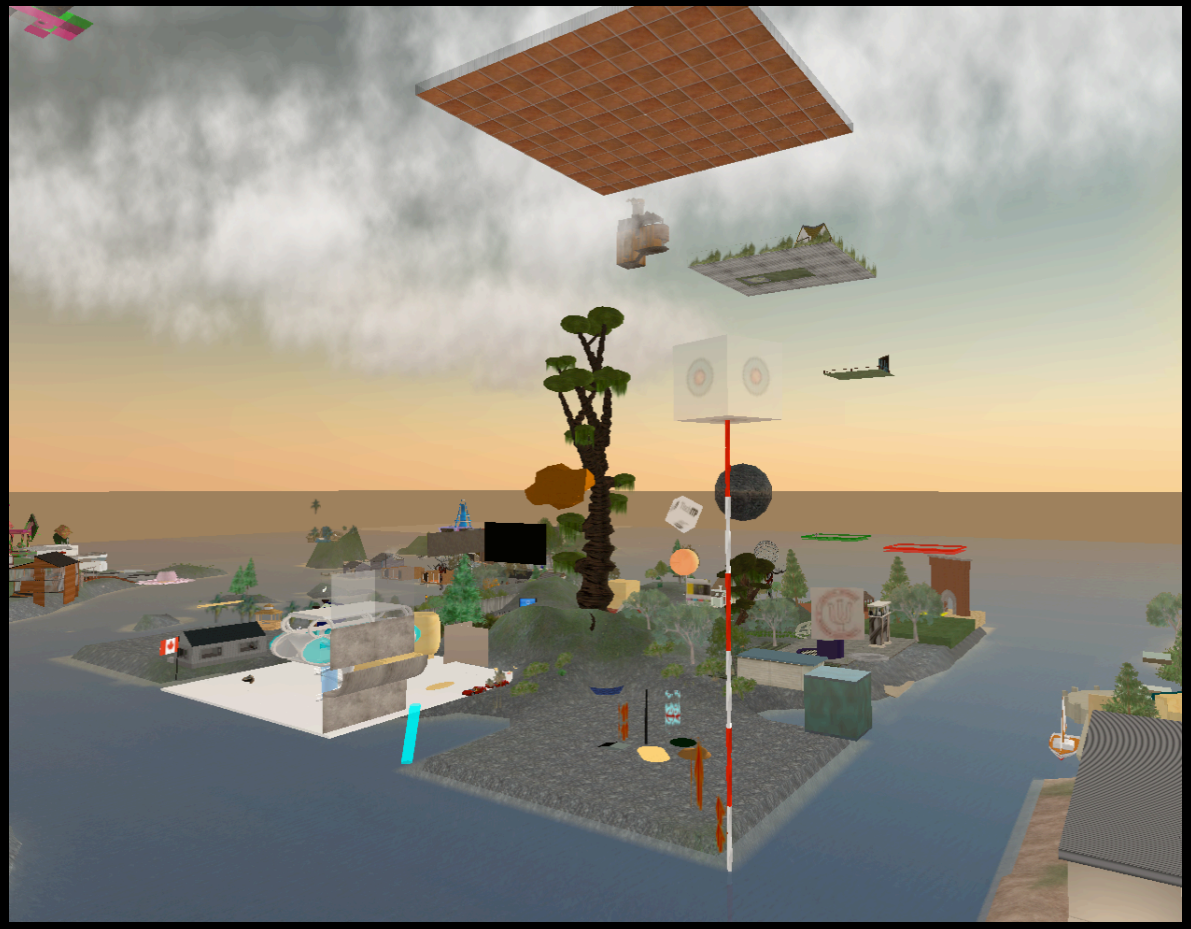
a tool for experiments: Second Life



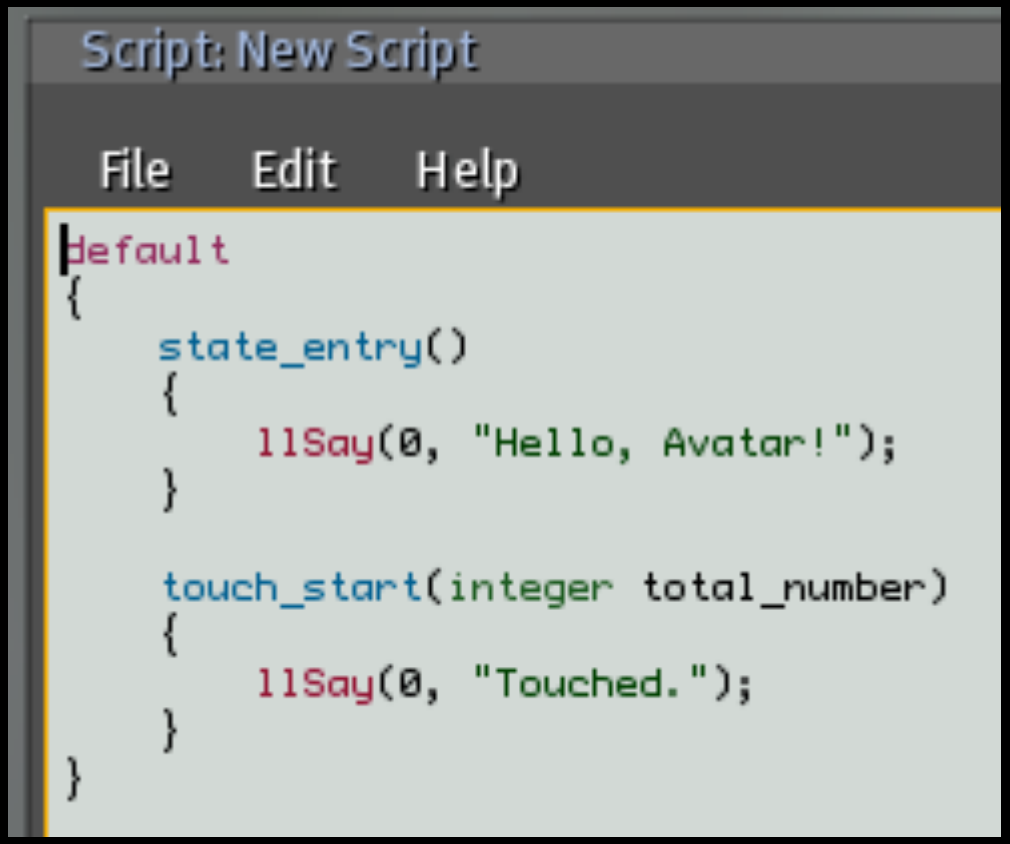
or *How I Learned to Stop Worrying and Love LSL*

(part 2, *varium et mutabile semper...*)

2007.02 Mitja Hmeljak mitja at indiana edu

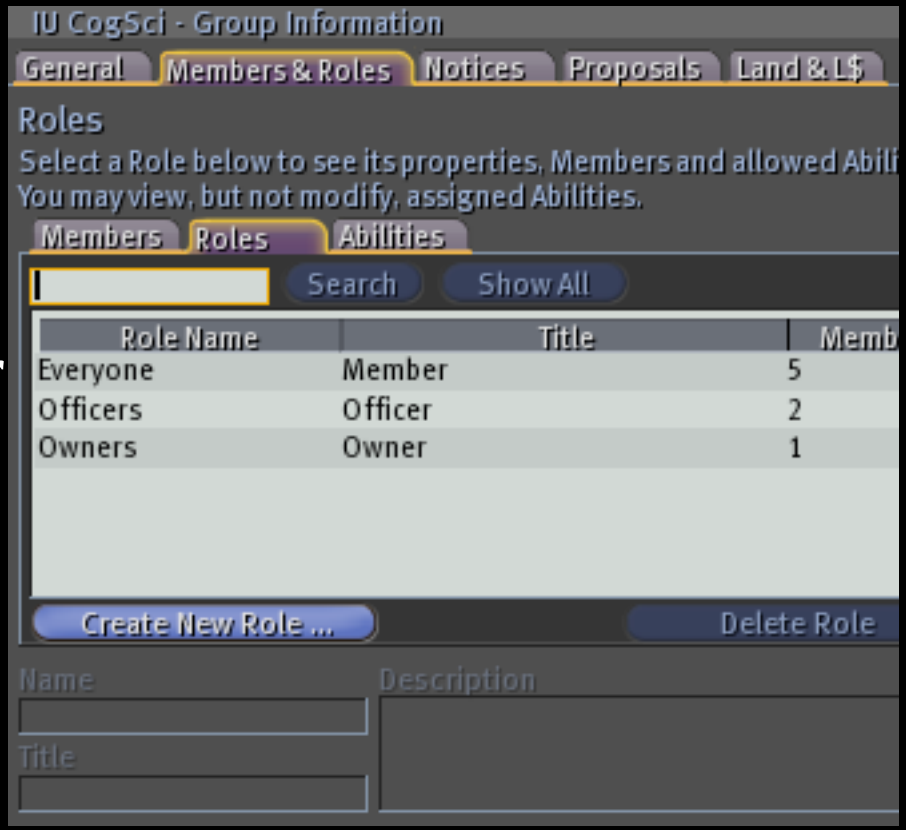
LSL semantics

- starting concepts:
  - events and
    event handlers
  - states
  - message-passing
  - library of functions



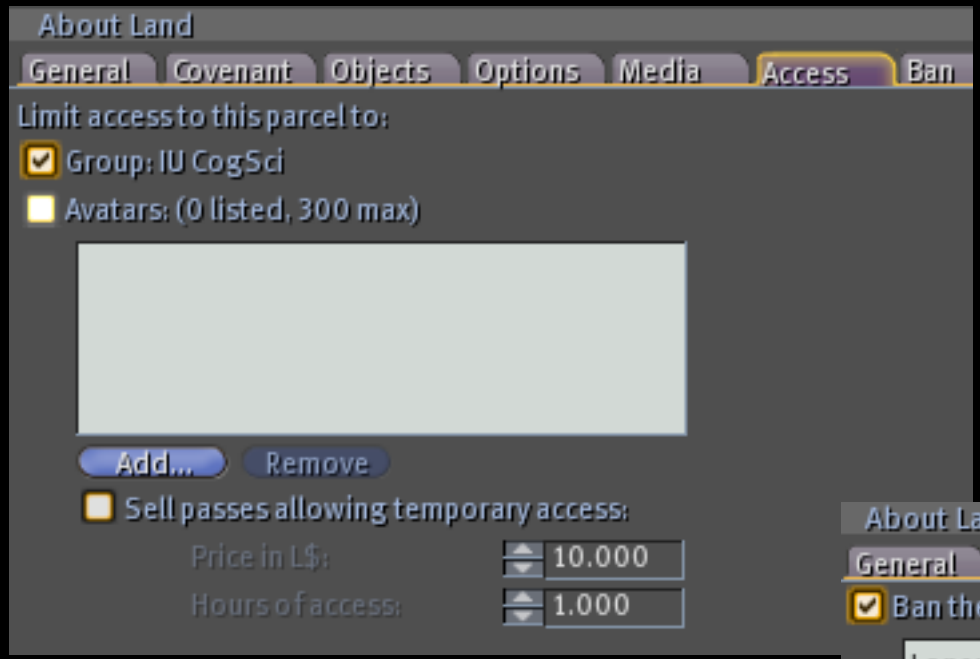
groups in SL

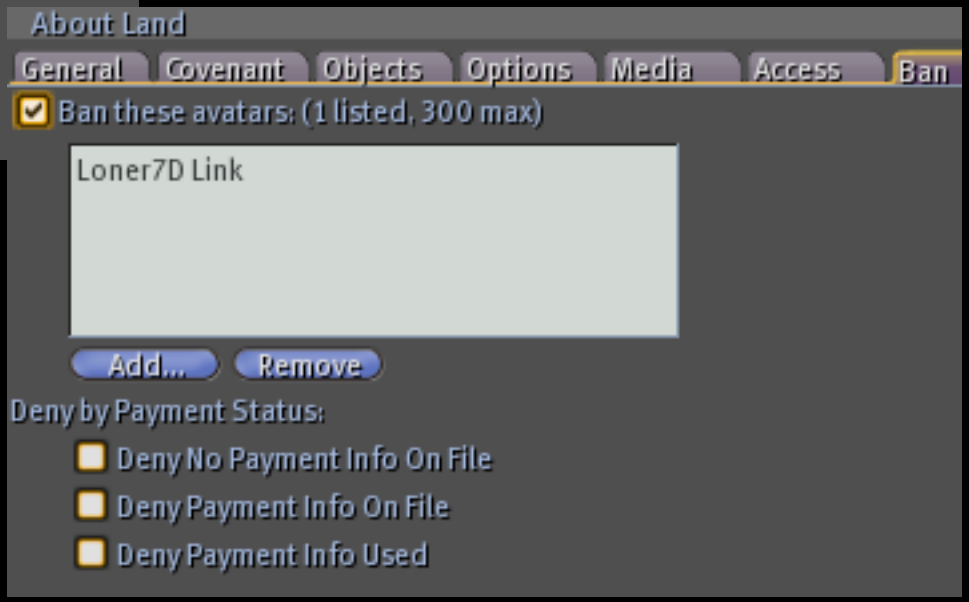
- create a group:
  - become group owner
- group officers
- everyone in group
  - abilities
  - land access



about SL land access

land access and ban lists





agent and object menus

- agent menu
  - agent related actions
  - avatar properties

- object menu
  - can be modified by script

a ticklish object

```
integer gNumOfTouches;

default {
  on_rez(integer pOnRez){  // parameter supplied by llRezObject()
    llSay(0, "Rezzed (" + (string)pOnRez + ") .");
    gNumOfTouches = 0;
  }

  state_entry(){
    llSay(0, "default state entered.");
    gNumOfTouches = 0;
  }

  touch_start(integer pTouchStart){
    gNumOfTouches = gNumOfTouches + 1;
    if (gNumOfTouches > 10) {
      llSay(0, "Why are you " + (string)pTouchStart + " touching me, " + llDetectedName(0) + " ?");
      //  llDetectedName(0) returns a meaningful value in events:
      //  collision(), collision_start(), collision_end()
      //  sensor(), touch(), touch_start(), touch_end()
      gNumOfTouches = 0;
    }
    // now change state to something else, since touched:
    state ticklish;
  }
}

state ticklish {
    // event handler for entry into this state:
    state_entry() {
      vector lColor = llGetColor(ALL_SIDES);
      llSay(0, "hi hi hi hi");
      llSetColor(<1,0,0>, ALL_SIDES); // set color to red temporarily
      llSleep(0.1);
      llSetColor(lColor, ALL_SIDES); // set color back to original color
      state default;
    }
}
```

a teleport?

```
// 2007.01.13 - Mitja Hmeljak
//
// this object will teleport the avatar to the platform upstairs

vector gDestinationCoords = <218, 194, 142>;

default {

    state_entry() {
        // llAvatarOnSitTarget() in the changed() function
        //  only detects avatars sitting on sit targets defined with llSitTarget(),
        //  so this needs to be done first:
        // llSitTarget(<0.0, 0.0, 0.01>, ZERO_ROTATION);

        // teleport by sitting and displacing the avatar:
        // the new position is computed relative to the object owning the script
        llSitTarget(gDestinationCoords - llGetPos(), ZERO_ROTATION);
        // then make the object spin about Z just for fun:
        llTargetOmega(<0,0,1>,2,1);
        // then change the label instead of "Sit Here":
        llSetSitText("platform");
        // turn on floating label telling us what the object does:
        llSetText("teleport\nto\nplatform", <0.0,0.5,0.5>, 1.0);
    }

    // event handler for property changes of the object:
    changed(integer change) {
        // when an avatar sits on an object, it causes a CHANGED_LINK event:
        if (change & CHANGED_LINK) {
           // wait half a second:
            llSleep(0.5);
            // let's check if there's an avatar sitting (i.e. key not NULL_KEY) :
            key avaKey = llAvatarOnSitTarget();
            if (avaKey != NULL_KEY) {
                // unsit the avatar:
                llUnSit(avaKey);
            }
        }
    }

} // end of default{}
```

communiqué

Method	Script Delay	Object owner	Other users	Other objects	Scripts in the same object	Send to computers outside SL	Receive from outside SL	Comment
Chat: Whisper, Say, Shout	No	Yes	Yes	Yes	No	No	No	Must be within chat distance to be able to receive.
`llOwnerSay`	No	Yes	No	No	No	No	No	Owner must be in the same sim.
`llDialog` Create	Yes	Yes	Yes	No	No	No	No	Only the directed user can receive and they must be in the sim.
`llDialog` Response	No	Yes	Yes	Yes	No	No	No	Receiver must be within chat distance of where the dialog box was created.
Instant Messages	Yes	Yes	Yes	No	No	No	No	
Link Messages	No	No	No	No	Yes	No	No	Only scripts contained within a given linked object may receive.
Email	Yes	No	No	Yes	Yes	Yes	Yes	link messages are better for intra-object communication.
XML-RPC	No	No	No	No	No	No	Yes	Only connections from an external computer to SL can be initiated.
HTTP	No	No	No	No	No	Yes	No	Only connections from SL to a non-Linden Lab server can be initiated.

objects talk to you

```
default {
    state_entry() {
        llShout(0,"yo buddy, I'm listening to you " + llKey2Name(llGetOwner()) + " !");
        // llListen(on channel, from object, from agent, for certain messages)
        //
        // listen on channel zero for any chat spoken by the object owner:
        llListen(0,"","","");
    }

    listen(integer channel, string name, key id, string message) {
        if (llToLower(message) == "yo") {
            // if the message from the owner is "hello", respond with "Hello.".
            llSay(0,"Yo.");
        }
    }
}
```

objects listening to each other...

```
// 2007.02.09..12 Mitja -
//  object listening to another object's position

default {
    state_entry() {
        // listen on channel zero for any chat spoken by the object owner.
        llSay(0,"listener reset");
        llSetText("you are going to be tracked!", <0.0,0.1,0.8>, 1.0);
        llListen(12435,"experiHat5scripted5",NULL_KEY,"");
    }

    listen(integer pChannel, string pName, key pID, string pMessage) {
        vector lDisplayPositionV;
        vector lAgentPositionV;

        // check if the message corresponds to a predefined string.
        // llToLower converts the message to lowercase.
        // This way, "HELLO", "Hello" or "HeLLo" will all work the same way.

        // split the list in two parts:  XYZ vector and  timestamp:

        list lReceivedString = llParseString2List(pMessage, ["|"], []);

        lAgentPositionV = (vector)pMessage;
        lDisplayPositionV.x = 191.0 + (10 * ((lAgentPositionV.z -  22.0) / 768.0));
        lDisplayPositionV.y = 185.0 + (10 * ((lAgentPositionV.x - 188.0) /  64.0));
        lDisplayPositionV.z =  22.0 + (10 * ((lAgentPositionV.y - 164.0) /  64.0));

        // move us there!
        llSetPos(lDisplayPositionV);

//        llSay(0,(string)lDisplayPositionV);
//        llSay(0,"received [" + pMessage + "]");
//        llSay(0,"received [" + (string)lAgentPositionV + "]");

        llSetText("you are at " + pMessage, <0.0,0.1,0.8>, 1.0);

    }
}
```

… and objects talking to each other

```
// Mitja 2007.02.10..19
// v 0.3
// detecting an object's position, and communicating it to another object for display

integer gAttached = 0;

default {
  state_entry() {
    gAttached = 0;
    llSetTimerEvent(2.0);
    llSay(0, "state_entry() timer reset.");
  }

  on_rez(integer pStartParam) {
    // when rezzed by an agent, pStartParam is always 0
    llSetTimerEvent(2.0);
    llSay(0, "on_rez("+ (string)pStartParam +") completed.");
  }

  touch_start(integer pTotNum) {
    vector lLocalPositionTouched = llGetPos();
    // ask the agent permission to attach ourself to them:
    llRequestPermissions(llDetectedKey(0), PERMISSION_ATTACH);
    llSay(0, "Touched(" + (string)pTotNum + ") at " +
             (string)lLocalPositionTouched +
             " and requested permissions to attach.");
  }

  // "run_time_permissions" event is triggered:
  //   - either after the user has responded to a permission request
  //   - or if permissions granted to the script have changed
  run_time_permissions(integer pPermissions) {
    if (pPermissions & PERMISSION_ATTACH)  {
      llAttachToAvatar(ATTACH_HEAD);
    }
  }

  attach(key pAttachedKey) {
    if (pAttachedKey != NULL_KEY){
      llWhisper( 0, "object attached to " + llKey2Name(pAttachedKey));
      gAttached = 1;
    } else {
      llWhisper( 0, "object detached from " + llKey2Name(pAttachedKey));
      gAttached = 0;
    }
  } // attach()

  timer() {
    vector lLocalPositionTimer = llGetPos();
    // llSetTimerEvent(llFrand(4.0)); // to randomize next timer event if necessary
    if (gAttached==1) {
      llShout(12435, (string)lLocalPositionTimer);
    }
  } // timer()

} // default()
```

for more LSL, read:

[1] - *Linden Scripting Language Guide*, Aaron Brashears, Andrew Meadows, Cory Ondrejka, Doug Soo, Donald Kjer, Linden Lab 2003
http://secondlife.com/developers/resources/pdfs/LSLGuide.pdf

[2] - Linden Scripting Language wiki:
http://rpgstats.com/wiki/index.php?title=LSL101Chapter1

[3] - LSL tutorial list on secondlife.com:
http://wiki.secondlife.com/wiki/LSL_Tutorial

[4] - in-world tutorial from Bromley College island
http://slurl.com/secondlife/Daydream SE Islands/206/40

slide-by-slide references

0. view of Campus: Second Life island. Retrieved on 2007.02.07.
8. table from http://rpgstats.com/wiki/index.php?title=Communications LSL wiki. Retrieved on 2007.02.07.

for more information

- mitja at indiana edu
  - in-world as Mitja Omlet
- IU CogSci Q400 space on Campus island for Spring 2007:
  - http://slurl.com/secondlife/Campus/198/174/30/

end of *How I Learned to Stop Worrying and Love LSL* **(part 2, varium et mutabile semper...) - to be continued.**