Virtual Worlds Report

Group 5

Interactive

Table of Contents

Group Members

13165852	Niall Houlihan
13146041	Liam Hearne
13132334	Cathal Lenihan
13123653	Evan O Hara

Intro

As we stated in the original report we continued on with the creation of a house and I feel our piece shows our hard work. As a group we decided on a simple cottage style bungalow but with a modern show house feel within the interior. As part of the interior we used multiple scripts which we will have below and stuck to the same plan we set out to use in the original report.

Group Member Statements

It is fair to say that all members of the group put in equal effort into this Project. Before we started we set each person specific roles. These differed to the ones set out in the report and often members interacted in different roles to make the house more interactive but also give it a more luxurious feel.

Cathal was the main scripter in the project. He was involved in the majority of the script creations or modification of scripts made by other members or found form an external place to make it work for the purpose we needed or to make it work more efficiently, Cathal's skills at scripting made it possible to give the project a more realistic feel.

Liam was one of the main builders within the house. Liam built the main exterior of the house and applied the textures. He also created a lot of the objects in the sitting room and helped cathal with the scripting. He was also the man who imported the objects and put them together for the final delivery and the presentation.

Niall was put in charge of the bathroom. He designed the taps, shower toilet and other minor objects within the bathroom to give it a realistic feel. He also used Photoshop to reduce the amount of prims as possible. He used it to create the fence and windows so that they would transparent and realistic with as little prims as possible.

Evan was in charge of the creation of the bedroom and made the bed cabinet and the clock of the room to a high standard. He was also in charge of importing objects from LSL such as the bird table cake and shopping bag which were too complicated for us to build but gave the house a nice finish.

Script Modification Modifications - Doors

We modified the door script so that it would open Horizontally Vertically and also sliding doors so that we could use them in the house for the purpose they would be in a normal house.

Vertically

The below script was the base for which we modified our scripts to open Vertically, We used this on the windows, fridge door, bedside locker, cabinet, interior doors and front door.

```
\\Front door
vector center_of_rotation = <0.0, 0.63, 0.0>;
vector rot = <0.0, 0.0, 90.0>;

rotate(vector rot)
{
   rotation vRotArc = llEuler2Rot(rot * DEG_TO_RAD);

   vector local_center_of_rotation = center_of_rotation * llGetLocalRot();

   vector vPosRotOffset = local_center_of_rotation * vRotArc;

   vector vPosOffsetDiff = local_center_of_rotation - vPosRotOffset;

   vector vPosNew = llGetLocalPos() + vPosOffsetDiff;

   rotation vRotNew = llGetLocalRot() * vRotArc;

   llSetPrimitiveParams( [PRIM_POSITION, vPosNew, PRIM_ROTATION,
vRotNew/llGetRootRotation()] );
}

default
{
   touch_start(integer num_detected)
   {
      if (llDetectedKey(0) == llGetOwner()) state open;
   }
}

state open
```

```
{
   state_entry()
   {
      rotate(rot);
   }

   touch_start(integer num_detected)
   {
         state closed;
   }
}

state closed
{
   state_entry()
   {
      rotate(-rot);
   }
   touch_start(integer num_detected)
   {
         state open;
   }
}
```

Horizontally

We also modified the script so that it would open towards the user. We felt this was important as it made the house feel more realistic. This was used in the cooker door.

```
\\Cooker Door
vector center_of_rotation = <0.0, 0.40, 0.0>;

vector rot = <0.0, 0.0, 90.0>;

rotate(vector rot)
{

   rotation vRotArc = llEuler2Rot(rot * DEG_TO_RAD);


   vector local_center_of_rotation = center_of_rotation * llGetLocalRot();


   vector vPosRotOffset = local_center_of_rotation * vRotArc;


   vector vPosOffsetDiff = local_center_of_rotation - vPosRotOffset;


   vector vPosNew = llGetLocalPos() + vPosOffsetDiff;


   rotation vRotNew = llGetLocalRot() * vRotArc;

   llSetPrimitiveParams( [PRIM_POSITION, vPosNew, PRIM_ROTATION,
vRotNew/llGetRootRotation()] );
}

default
{
   touch_start(integer num_detected)
   {
      if (llDetectedKey(0) == llGetOwner()) state open;
   }
}

state open
{
```

```
    state_entry()
    {
        rotate(rot);
    }

    touch_start(integer num_detected)
    {

            state closed;

    }
}

state closed
{
    state_entry()
    {
        rotate(-rot);
    }

    touch_start(integer num_detected)
    {

            state open;

    }
```

Sliding Door

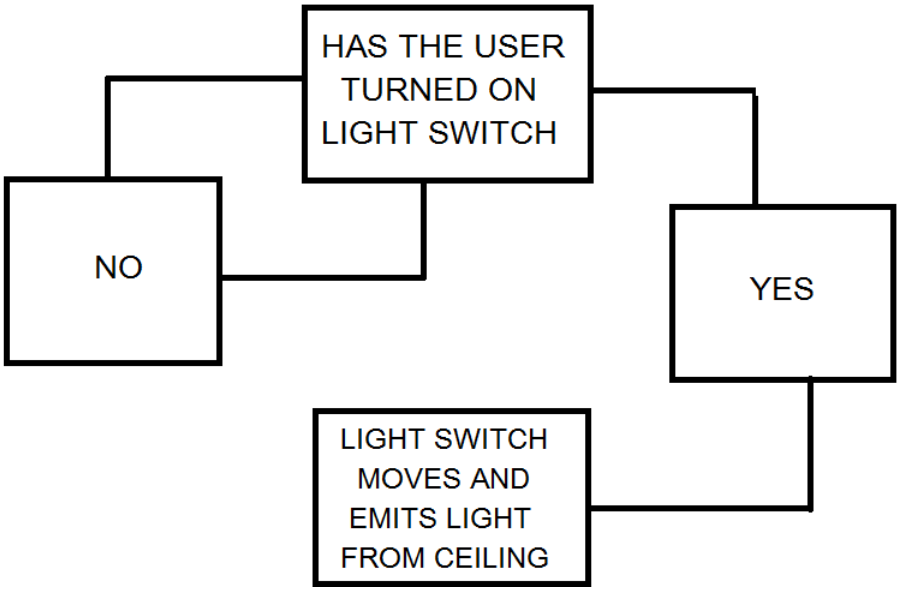
We decided to make a new script for a sliding door as we felt it would suit the shower in the bathroom. The scripts slides behind a panel allowing access to the shower and closes when touched again.

```
\\Sliding shower door
default
{
   touch_start(integer num_detected) { state up; }
}

state up
{
   state_entry() {
      llSetPos(llGetPos() + <-1.0,0.0,0.0>);
   }
   touch_start(integer num_detected) { state down; }
}

state down
{
   state_entry() {
      llSetPos(llGetPos() - <-1.0,0.0,0.0>);
   }
   touch_start(integer num_detected) { state up; }
}
```

State Transition Diagram & Interaction Between Scripts

Lights

This is the scripts and the diagram for the lighting system in the house. As soon as the user touches the light the bulb in the ceiling emits light and turns off when touched again.



Light Switch

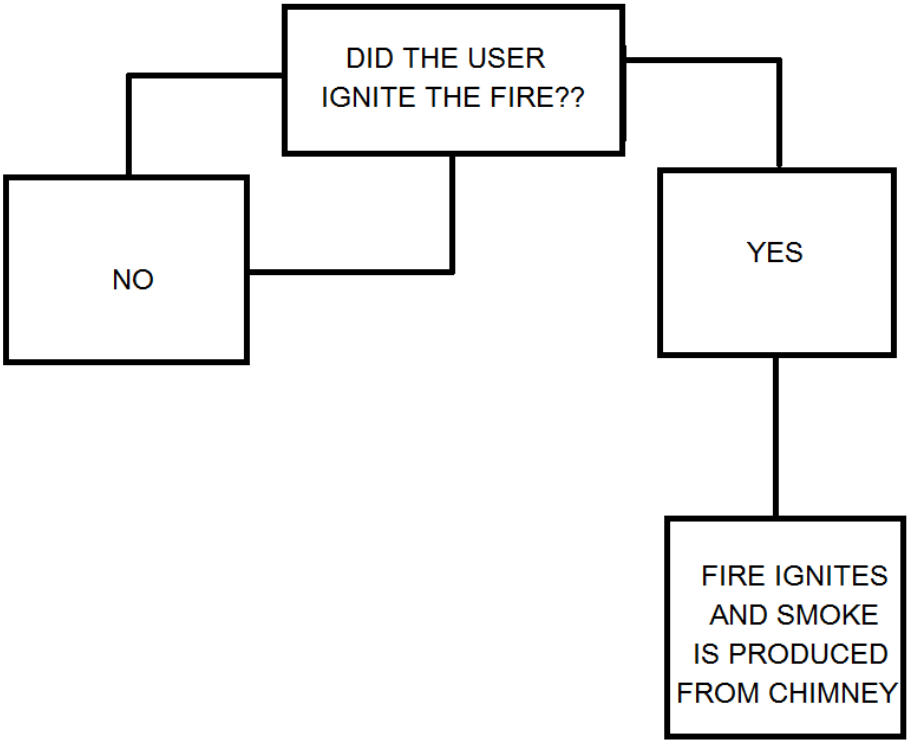
```
//for the panel of the lightswitch
integer myswitch;

default
{


   state_entry()
   {

      myswitch=FALSE;
   }

   touch_start(integer total_number)
   {

     if(myswitch==FALSE)
     {

     //Turn Light Bulb ON

      llMessageLinked(LINK_ALL_CHILDREN, 0, "start", NULL_KEY);
     myswitch=TRUE;


     }
     else
    {

    //Turn Light Bulb Off

    llMessageLinked(LINK_ALL_CHILDREN, 0, "stop", NULL_KEY);
    myswitch=FALSE;
     }

   }
}
```

Light

```
//put in the light bulbs
default
{
   state_entry()
   {
      llSetPrimitiveParams([PRIM_FULLBRIGHT,ALL_SIDES,FALSE]);

   }

   link_message(integer sender_num, integer num, string str, key id)
   {
      if(str=="stop")
      {
        llSetPrimitiveParams([PRIM_POINT_LIGHT, FALSE, <1, 1, 0.5>, 1.0, 10.0, 0.75,PRIM_GLOW
,ALL_SIDES,0]);
        llSetPrimitiveParams([PRIM_FULLBRIGHT,ALL_SIDES,FALSE]);
      }
      if(str=="start")
      {
        llSetPrimitiveParams([PRIM_POINT_LIGHT, TRUE, <1, 1, 0.5>, 1.0, 10.0, 0.75,PRIM_GLOW
,ALL_SIDES,0.2]);
        llSetPrimitiveParams([PRIM_FULLBRIGHT,ALL_SIDES,TRUE]);
      }
   }
}
```

Fire - Chimney

The fire and chimney are two linked scripts. When the fire is ignited by the user the smoke particles are released via the chimney however when the user turns off the fire the smoke will also be turned off.



Flames

```
make_particles()
{
   llParticleSystem([
        PSYS_PART_FLAGS ,
         PSYS_PART_INTERP_COLOR_MASK
         | PSYS_PART_INTERP_SCALE_MASK
         | PSYS_PART_EMISSIVE_MASK
         | PSYS_PART_FOLLOW_VELOCITY_MASK
         | PSYS_PART_WIND_MASK
         | PSYS_PART_BOUNCE_MASK
         ,
        PSYS_SRC_PATTERN, (integer)2,
        PSYS_SRC_TEXTURE, llGetInventoryName(INVENTORY_TEXTURE, 0),
        PSYS_PART_MAX_AGE, (float)1.25,
        PSYS_SRC_MAX_AGE, (float) 0.0,
        PSYS_SRC_BURST_RATE, (float) 0.05,
```

```
        PSYS_SRC_BURST_PART_COUNT, (integer)7,
        PSYS_SRC_BURST_SPEED_MIN, (float).1,
        PSYS_SRC_BURST_SPEED_MAX, (float).3,
        PSYS_SRC_ACCEL, <0.0,0.0,1.5>,
        PSYS_PART_START_COLOR, <1,1,0>,
        PSYS_PART_END_COLOR, <.4,0,0>,
        PSYS_PART_START_ALPHA, (float).8,
        PSYS_PART_END_ALPHA, (float).0,
        PSYS_PART_START_SCALE, <.25,.25,FALSE>,
        PSYS_PART_END_SCALE, <.5,.5, FALSE>,
        PSYS_SRC_ANGLE_BEGIN, (float) .03*PI,
        PSYS_SRC_ANGLE_END, (float)0.00*PI,
        PSYS_SRC_OMEGA,  <0.0,0.0,0.0>
         ]);
}

integer myswitch;

default
{
   state_entry()
   {
      llParticleSystem([]);
      llMessageLinked(LINK_ALL_CHILDREN, 0, "stop", NULL_KEY);
     myswitch=FALSE;
   }

   touch_start(integer total_number)
   {
      state on;
   }
}

state on
{
   state_entry()
   {
      llMessageLinked(LINK_ALL_CHILDREN, 0, "start", NULL_KEY);
      myswitch=TRUE;
      make_particles();
   }

   touch_start(integer total_number)
   {
      state default
```

Chimney

```
\\Chimney
make_particles()
{
   llParticleSystem([
      PSYS_PART_FLAGS ,
          PSYS_PART_WIND_MASK
         | PSYS_PART_INTERP_COLOR_MASK
         | PSYS_PART_INTERP_SCALE_MASK
         | PSYS_PART_FOLLOW_SRC_MASK
         | PSYS_PART_FOLLOW_VELOCITY_MASK
         | PSYS_PART_EMISSIVE_MASK
        ,
   PSYS_SRC_PATTERN,         PSYS_SRC_PATTERN_ANGLE_CONE,
   PSYS_SRC_TEXTURE,         "sprite-particle-cloud"
   ,PSYS_SRC_MAX_AGE,          0.0
   ,PSYS_PART_MAX_AGE,         10.0
   ,PSYS_SRC_BURST_RATE,       0.5
   ,PSYS_SRC_BURST_PART_COUNT,  3
   ,PSYS_SRC_BURST_RADIUS,     10.0
   ,PSYS_SRC_BURST_SPEED_MIN,   .4
   ,PSYS_SRC_BURST_SPEED_MAX,   .5
   ,PSYS_SRC_ACCEL,           <0.0,0,.05>
   ,PSYS_PART_START_COLOR,     <1.0,1.0,1.0>
   ,PSYS_PART_END_COLOR,       <1.0,1.0,1.0>
   ,PSYS_PART_START_ALPHA,     0.9
   ,PSYS_PART_END_ALPHA,       0.0
   ,PSYS_PART_START_SCALE,     <.25,.25,.25>
   ,PSYS_PART_END_SCALE,       <.75,.75,.75>
   ,PSYS_SRC_ANGLE_BEGIN,      0 * DEG_TO_RAD
   ,PSYS_SRC_ANGLE_END,        45 * DEG_TO_RAD
   ,PSYS_SRC_OMEGA,           <0.0,0.0,0.0>
         ]);
}

myon_state()
{
   make_particles();

}


myoff_state()
{
```

```
  llParticleSystem([]);
}


default
{
   state_entry()
   {
        myoff_state();

   }

link_message(integer sender_num, integer num, string str, key id)
   {
      if(str=="stop")
      {
         myoff_state();
      }
      if(str=="start")
      {
        myon_state();
      }
   }
}
```

Other Scripts

Clock

The clock is broken into 3 separate scripts. The Seconds script moves the hand around the clock every second. The Minute script rotates the hand around the clock every 60 seconds and the Hour hand moves the hand around the clock every 60 minutes.

Seconds

```
integer is_on;

default
{
   state_entry()
   {
      llSetTimerEvent(1.0);
   }

   timer()
   {
  llSetLocalRot(llGetLocalRot()*llEuler2Rot(<0,0,-6>*DEG_TO_RAD));
   }
}
```

Minutes

```
integer is_on;

default
{
   state_entry()
   {
      llSetTimerEvent(60.0);
   }


   timer()
   {
   llSetLocalRot(llGetLocalRot()*llEuler2Rot(<0,0,-6>*DEG_TO_RAD));
   }
}
```

Hours

```
integer is_on;

default
{
   state_entry()
   {
      llSetTimerEvent(3600.0);
   }


   timer()
   {
   llSetLocalRot(llGetLocalRot()*llEuler2Rot(<0,0,-6>*DEG_TO_RAD));
   }
}
```

Musical

Our project has multiple musical scripts. The doorbell makes a sound like a traditional doorbell when the user touches the switch .

```
default
{
   touch_start(integer total_num)
   {
      llTriggerSound("doorbell-6_converted", 1.0);
      llSetPos(llGetPos()+<0.025,0,0>);
      state off;
   }
}

state off
{
 state_entry() {

      llSetPos(llGetPos()-<0.025,0,0>);
      state default;
  }
}
```

It also has a radio function. As soon as the user turns on the radio it plays an interview we downloaded from YouTube will play until either the interview is over or until the user stops the interview.

```
//put into the radio
list songs = ["interview",9];
integer volume = 50;
integer lis_count;
integer playing;
integer busy;
integer part;
integer lis;
integer sl;
float delay;
list cancel = ["CANCEL"];
list playlist;
list waiting;
list song;
string vol_str = "Volume";
string song_str = "Songs";
```

```
string song_name;
list StrideOfList(list src, integer stride, integer start, integer end)
{
   list l = [];
   integer ll = llGetListLength(src);
   if(start < 0)start += ll;
   if(end < 0)end += ll;
   if(end < start) return llList2List(src, start, start);
   while(start <= end)
   {
      l += llList2List(src, start, start);
      start += stride;
   }
   return l;
}

list Volumes(integer vol)
{
   integer v = 0;
   list l = [];
   do
   {
      if(v != vol)
      l += [((string)v)];
   }
   while((++v) <= 10);
   return l;
}

PageOne(key k, integer c)
{
   llDialog(k, "\nAdjust the volume or select a song to play?", [vol_str, song_str] + cancel, c);
}

PlaySong(string n)
{
   song = [];
   integer c = -1;
   string name = "";
   do
   {
      if(llSubStringIndex((name = llGetInventoryName(INVENTORY_SOUND, (++c))), n) != -1)
      song += [name];
   }
   while(name);
```

```
    delay = llList2Float(songs, (llListFindList(songs, [n]) + 1));
    if((sl = llGetListLength(song)))
    {
        llPreloadSound(llList2String(song, (part = 0)));
        if(sl > 1)
        llPreloadSound(llList2String(song, 1));
        playing = FALSE;
        llSetTimerEvent(0.01);
    }
}

integer Chan()
{
    return llRound((llFrand(-5000000.0) + -500000.0));
}

float ScaleVol(integer v)
{
    return (v * 0.1);
}

Listen(integer c, key a)
{
    lis = llListen(c, "", a, "");
}

RemoveListen(integer b)
{
    llListenRemove(lis);
    lis_count = 0;
    if(b)
    busy = FALSE;
    lis = 0;
}

SetListenTimer(integer p)
{
    if(p) {
        while(((++lis_count) * llRound(delay)) < 30);
    }
    else
    {
        lis_count = 1;
        llSetTimerEvent(30.0);
    }
```

```
}

integer CheckWaitingRoom(integer c)
{
   if(llGetListLength(waiting) > 0)
   {
      key a = llList2Key(waiting, 0);
      if(!c)
      {
         RemoveListen(0);
         Listen((c = Chan()), a);
         SetListenTimer(playing);
      }
      PageOne(a, c);
      waiting = llDeleteSubList(waiting, 0, 0);
      return 1;
   }
   return 0;
}

default
{
   on_rez(integer param)
   {
      llStopSound();
      llResetScript();
   }
   changed(integer change)
   {
      if(change & CHANGED_INVENTORY)
      llResetScript();
   }
   touch_start(integer nd)
   {
      while(nd)
      {
         key agent = llDetectedKey(--nd);
         if(!busy)
         {
            busy = TRUE;
            integer channel = Chan();
            SetListenTimer(playing);
            Listen(channel, agent);
            PageOne(agent, channel);
         }
```

```
        else
        {
            list a = [agent];
            if(llListFindList(waiting, a) == -1)
            waiting += a;
        }
    }
}
listen(integer chan, string name, key id, string msg)
{
    if(msg != llList2String(cancel, 0))
    {
        SetListenTimer(playing);
        if(msg == vol_str)
        {
            llDialog(id, "\nChange the volume?\nThe current volume is set at \"" + ((string)volume) +
"\"", cancel + Volumes(volume), chan);
            return;
        }
        if(msg == song_str)
        {
            string current = "";
            if(llGetListLength(playlist) > 0)
            {
                current = "\n\nThe songs currently queued are\n\"" + llList2String(playlist, 0) + "\"
(currently playing)";
                if(llGetListLength(playlist) > 1)
                current += "\n\"" + llDumpList2String(llList2List(playlist, 1, -1), "\"\n\"") + "\"";
            }
            llDialog(id, llGetSubString(("\nSelect a song to play?" + current), 0, 500), cancel +
StrideOfList(songs, 2, 0, -1), chan);
            return;
        }
        if(llListFindList(Volumes(volume), [msg]) != -1)
        {
            llAdjustSoundVolume(ScaleVol((volume = ((integer)msg))));
            PageOne(id, chan);
            return;
        }
        if(llGetListLength((playlist += [msg])) == 1)
        PlaySong((song_name = msg));
    }
    if(CheckWaitingRoom(chan))
    return;
    RemoveListen(1);
```

```
    }
    timer()
    {
        if(llGetListLength(playlist) > 0)
        {
            if(!playing)
            {
                llSetTimerEvent(delay);
                playing = TRUE;
            }
            llPlaySound(llList2String(song, part), ScaleVol(volume));
            if((++part) == sl)
            {
                if(llGetListLength(playlist) > 1)
                {
                    song_name = llList2String((playlist = llDeleteSubList(playlist, 0, 0)), 0);
                    llSleep(delay);
                    PlaySong(song_name);
                }
                else
                {
                    llSetTimerEvent(0.0);
                    song_name = "";
                    playing = FALSE;
                    playlist = [];
                }
            }
            else if(part == (sl - 1))
            llPreloadSound(llList2String(song, 0));
            else
            llPreloadSound(llList2String(song, (part + 1)));
        }
        if(lis && (!(--lis_count)))
        {
            if(!(CheckWaitingRoom(0)))
            RemoveListen(1);
        }
    }
}
```

Seating Function

The seating function allows the user to sit down on the chair couch and toilet in a way that looks natural. with the couch we designed two cushions on it which allowed us to apply two scripts and allow two people to sit on the couch.

Chair/Couch/Toilet

```
\\Chair
string animation = "ANIMATIONNAME";
vector sittarget = < -0.3, 0.0, 0.55>;
vector sitangle = < 0.0, 0.0, 180.0>;


integer only_owner = 0;

integer HOV_ON = 0;
integer OBJHIDE_ON = 0;

string HOVERTEXT = "";
vector COLOR = < 1.0, 1.0, 1.0>;


integer SITTXT_ON = 1;
string SITTEXT = "";


rotation sitrotation;
key owner;
key sitter = NULL_KEY;
integer SITTING;
integer LN;
integer LS;


integer test_sit() {
   if(LS == 1) {
      if(llAvatarOnLinkSitTarget(LN) != NULL_KEY) return 1;
      else return 0;
   }
   else if(LS == 0) {
      if(llAvatarOnSitTarget() != NULL_KEY) return 1;
      else return 0;
```

```
    }
    else return 0;
}


default {
    state_entry() {
        SITTING = 0;
        if(only_owner == 1) owner = llGetOwner();
        if(llGetNumberOfPrims() > 1) {
            LN = llGetLinkNumber();
            LS = 1;
        }
        else {
            LN = LINK_SET;
            LS = 0;
        }
        sitrotation = llEuler2Rot(sitangle * DEG_TO_RAD);
        llSitTarget(sittarget,sitrotation);
        llSetClickAction(CLICK_ACTION_SIT);
        if(SITTXT_ON) llSetSitText(SITTEXT);
        if(HOV_ON) llSetText(HOVERTEXT,COLOR,1.0);
        else llSetText("",<0.0,0.0,0.0>,1.0);
        if(OBJHIDE_ON) llSetLinkAlpha(LN,1.0,ALL_SIDES);
    }

    on_rez(integer num) {
        llResetScript();
    }

    changed(integer change) {
        if(change & CHANGED_LINK) {
            if(SITTING == 0 && LS == 1) sitter = llAvatarOnLinkSitTarget(LN);
            if(SITTING == 0 && LS == 0) sitter = llAvatarOnSitTarget();
            if(only_owner == 1 && sitter != owner && sitter != NULL_KEY) {
                llUnSit(sitter);
                llInstantMessage(sitter,"You are not permitted to sit here.");
                sitter = NULL_KEY;
            }
            else if(SITTING == 0 && sitter != NULL_KEY) {
                SITTING = 1;
                llRequestPermissions(sitter,PERMISSION_TRIGGER_ANIMATION);
            }
            else if(SITTING == 1 && test_sit() == 0) {
                if(llGetPermissions() & PERMISSION_TRIGGER_ANIMATION) llStopAnimation(animation);
```

```
            if(HOV_ON) llSetText(HOVERTEXT,COLOR,1.0);
            if(OBJHIDE_ON) llSetLinkAlpha(LN,1.0,ALL_SIDES);
            SITTING = 0;
            sitter = NULL_KEY;
          }
        }
    }

    run_time_permissions(integer perm) {
        if(perm & PERMISSION_TRIGGER_ANIMATION) {
            if(HOV_ON) llSetText("",COLOR,1.0);
            if(OBJHIDE_ON) llSetLinkAlpha(LN,0.0,ALL_SIDES);
            llStopAnimation("sit");
            llStartAnimation(animation);
        }
    }
}
```

Particles

In addition to using the smoke particle seen earlier on in the report we also used the water particle in the tap and the shower. The tap when touched sprayed water out of it into the sink. The shower when touch also sprayed water from the shower head.

Tap

```
//put into the tap
make_particles()
{
    llParticleSystem([
            PSYS_PART_FLAGS,
            PSYS_PART_INTERP_COLOR_MASK
          | PSYS_PART_INTERP_SCALE_MASK
          | PSYS_PART_FOLLOW_SRC_MASK
          | PSYS_PART_FOLLOW_VELOCITY_MASK
          | PSYS_PART_EMISSIVE_MASK,
    PSYS_SRC_PATTERN,          PSYS_SRC_PATTERN_ANGLE_CONE,
    PSYS_SRC_TEXTURE,          "Water",
    PSYS_SRC_MAX_AGE,          0.0,
    PSYS_PART_MAX_AGE,         1.5,
    PSYS_SRC_BURST_RATE,       0.0,
    PSYS_SRC_BURST_PART_COUNT,  9,
    PSYS_SRC_BURST_RADIUS,     1.0,
    PSYS_SRC_BURST_SPEED_MIN,   .3,
```

```
    PSYS_SRC_BURST_SPEED_MAX,   .9,
    PSYS_SRC_ACCEL,          <0.0,0,.05>,
    PSYS_PART_START_COLOR,     <1.0,1.0,1.0>,
    PSYS_PART_END_COLOR,       <1.0,1.0,1.0>,
    PSYS_PART_START_ALPHA,     0.7,
    PSYS_PART_END_ALPHA,       0.3,
    PSYS_PART_START_SCALE,     <.05,.05,.05>,
    PSYS_PART_END_SCALE,       <.05,.05,.05>,
    PSYS_SRC_ANGLE_BEGIN,      0 * DEG_TO_RAD,
    PSYS_SRC_ANGLE_END,        0 * DEG_TO_RAD,
    PSYS_SRC_OMEGA,          <0.0,0.0,0.0>
        ]);
}

integer myswitch;

default
{
    state_entry()
    {
        llParticleSystem([]);
        llMessageLinked(LINK_ALL_CHILDREN, 0, "stop", NULL_KEY);
       myswitch=FALSE;
    }

    touch_start(integer total_number)
    {
        state on;
    }
}

state on
{
    state_entry()
    {
        llMessageLinked(LINK_ALL_CHILDREN, 0, "start", NULL_KEY);
        myswitch=TRUE;
        make_particles();
    }

    touch_start(integer total_number)
    {
        state default;
    }
}
```

Shower

```
//put into the shower head
make_particles()
{
   llParticleSystem([
        PSYS_SRC_TEXTURE, llGetInventoryName(INVENTORY_TEXTURE, 0),
       PSYS_PART_START_SCALE, <0.04, .3, FALSE>, PSYS_PART_END_SCALE, <.2, 0.5, FALSE>,
       PSYS_PART_START_COLOR, <.6,.6,.6>,    PSYS_PART_END_COLOR, <0.3,0.4,.6>,
       PSYS_PART_START_ALPHA,  (float)0.75,          PSYS_PART_END_ALPHA, (float)0.50,
       PSYS_SRC_BURST_PART_COUNT, (integer)5,
       PSYS_SRC_BURST_RATE, (float) 0.01,
       PSYS_PART_MAX_AGE, (float)1.0,
       PSYS_SRC_MAX_AGE,(float) 0.0,
       PSYS_SRC_PATTERN, (integer)8,
       PSYS_SRC_BURST_SPEED_MIN, (float)1.3,   PSYS_SRC_BURST_SPEED_MAX, (float)1.9,
       PSYS_SRC_BURST_RADIUS, 0.1,
       PSYS_SRC_ANGLE_BEGIN, (float) 0.08*PI,    PSYS_SRC_ANGLE_END, (float)0.08*PI,
       PSYS_SRC_ACCEL, <0.0,0.0, - 2.0 >,
       PSYS_PART_FLAGS,
                     PSYS_PART_INTERP_COLOR_MASK
                   | PSYS_PART_INTERP_SCALE_MASK
                   | PSYS_PART_EMISSIVE_MASK
                   | PSYS_PART_FOLLOW_VELOCITY_MASK,
       PSYS_SRC_OMEGA,  <0.0,0.0,0.0>
        ]);
}

integer myswitch;

default
{
   state_entry()
   {
      llParticleSystem([]);
      llMessageLinked(LINK_ALL_CHILDREN, 0, "stop", NULL_KEY);
     myswitch=FALSE;
   }

   touch_start(integer total_number)
   {
      state on;
   }
}
```

```
state on
{
   state_entry()
   {
      llMessageLinked(LINK_ALL_CHILDREN, 0, "start", NULL_KEY);
      myswitch=TRUE;
      make_particles();
   }

   touch_start(integer total_number)
   {
      state default;
   }
}
```

Moving Object

We also decided to make a moving lawnmower in the garden although we felt it couldn't cut grass we made it move backwards and forwards as it gave the lawnmower a purpose and also made a nice feature to the house.

```
\\Lawnmower
default
{
   touch_start(integer num_detected) { state up; }
}

state up
{
   state_entry() {
      llSetPos(llGetPos() + <1.0,0.0,0.0>);
   }
   touch_start(integer num_detected) { state down; }
}

state down
{
   state_entry() {
      llSetPos(llGetPos() - <1.0,0.0,0.0>);
   }
   touch_start(integer num_detected) { state up; }
}
```

Images

Unfortunately my images would not work when exported into Microsoft word s I have sent them in the email in a folder.

Screenshots

These two pictures show that when the fire is on the chimney produce smoke and when the fire is off there is no more smoke.

Above is the plan of the house, different rooms from within the house and the view of the exterior of the house