Σημειώσεις μαθήματος:

Τεχνολογικές και Διδακτικές Καινοτομίες με τη Χρήση της Πληροφορικής

Εικονική Πραγματικότητα

Μέρος 2ο

Σεπτέμβριος 2015

Στις σελίδες που ακολουθούν παρατίθενται ορισμένα πολύ βασικά scripts, ο τρόπος λειτουργίας και παραμετροποίησής τους. Περισσότερα μπορείτε να βρείτε στο:

http://www.free-lsl-scripts.com/cgi/freescripts.plx

ή κάνοντας αναζήτηση στο Google, βάζοντας στους όρους της αναζήτησης οπωσδήποτε και τις λέξεις lsl script (lsl είναι η γλώσσα προγραμματισμού του Second Life αλλά και του Opensim).

Γενικά πρέπει να προσέχετε τη σύνταξη των εντολών. Ότι βρίσκεται μέσα σε εισαγωγικά (" ") είναι κείμενο το οποίο και μπορείτε να αλλάξετε. Επίσης, όποια γραμμή του script έχει μπροστά της δύο πλάγιες γραμμές (//), είναι σχόλιο και δεν εκτελείται.

Προσοχή!
Η αντιγραφή και η επικόλληση στο χώρο που γράφουμε το script στο Opensim αρκετές φορές δεν δουλεύει καλά. Το πρόβλημα παρουσιάζεται συνήθως στην ή στις αγκύλες (}) που κλείνουν το script. Καλό είναι να τις διαγράφετε και να τις ξαναγράφετε. Επίσης, όλα τα scripts υπάρχουν και ως αντικείμενα έτοιμα προς χρήση στο Inventory σας στο φάκελο Lessons.

Script περιστροφής

1. Απλό script περιστροφής με την τοποθέτηση του αντικειμένου

```
default
{
   state_entry()
   {
llTargetOmega(<0.0, 0.0, 1.0>, 0.2, 1);
   }
}
```

2. Απλό script περιστροφής όταν κάνουμε κλικ στο αντικείμενο

```
default
{
   touch_start(integer num_detected)
   {
llTargetOmega(<0.0, 0.0, 1.0>, 0.2, 1);
   }
}
```

3.Script περιστροφής και σταματήματος με κλικ

```
default
{
      touch_start(integer num_detected)
      {
       llTargetOmega(<0.0, 0.0, 1.0>, 0.2, 1);
       state loop;
      }
}
state loop
{
      touch_start(integer num_detected)
      {
       llTargetOmega(<0.0, 0.0, 0.0>, 0, 0);
       state default;
      }
}
```

4. Script περιστροφής γύρω από απομακρυσμένο σημείο

Χρησιμοποιούμε 2 και περισσότερα αντικείμενα. Στο αντικείμενο που θα είναι το κέντρο της περιστροφής τοποθετούμε ένα από τα προηγούμενα scripts ανάλογα με το τι θέλουμε να επιτύχουμε. Επιλέγουμε τα αντικείμενα, με τελευταίο αυτό που περιέχει το script και τα κάνουμε link (Tools, Link)

Scripts ήχου

1. Απλό script ήχου, όταν ακουμπάμε ένα αντικείμενο.

Προϋποθέτει την τοποθέτηση του αρχείου ήχου στο αντικείμενο αυτό, στην καρτέλα Content. Στο συγκεκριμένο script πρέπει να αλλάξουμε το όνομα του αρχείου ήχου από sample_small σε ότι όνομα αρχείου έχουμε. Υπάρχει περιορισμός στη διάρκεια του αρχείου (10 δευτερόλεπτα).

```
default
{
   state_entry()
   {

   }

   touch_start(integer total_number)
   {
      llPlaySound("sample_small",1);
   }
}
```

2. Απλό script διαρκούς επανάληψης ήχου.

Προϋποθέτει την τοποθέτηση του αρχείου ήχου στο αντικείμενο αυτό, στην καρτέλα Content. Στο συγκεκριμένο script, πρέπει να αλλάξουμε το όνομα του αρχείου ήχου από sample_small σε ότι όνομα αρχείου έχουμε. Υπάρχει περιορισμός στη διάρκεια του αρχείου (10 δευτερόλεπτα).

```
default
{
   state_entry()
   {
         llLoopSound ("sample_small",1);

   }
}
```

3. Απλό script διαρκούς επανάληψης και σταματήματος ήχου, όταν ακουμπάμε ένα αντικείμενο.

Κάνοντας κλικ στο αντικείμενο ξεκινά ο ήχος. Με δεύτερο κλικ στο αντικείμενο, ο ήχος σταματάει. Προϋποθέτει την τοποθέτηση του αρχείου ήχου στο αντικείμενο αυτό, στην καρτέλα Content. Στο συγκεκριμένο script, πρέπει να αλλάξουμε το όνομα του αρχείου ήχου από sample_small σε ότι όνομα αρχείου έχουμε. Υπάρχει περιορισμός στη διάρκεια του αρχείου (10 δευτερόλεπτα).

```
default
{
      touch_start(integer num_detected)
      {
       llLoopSound("sample_small",1); //Put the sound file's name thats inside the object or the UUID in between the ""
       state loop;
      }
}
state loop
{
      touch_start(integer num_detected)
      {
       llStopSound();//this will stop the loop sound
       state default;
      }
}
```

4. Εκτέλεση αρχείου ήχου mp3

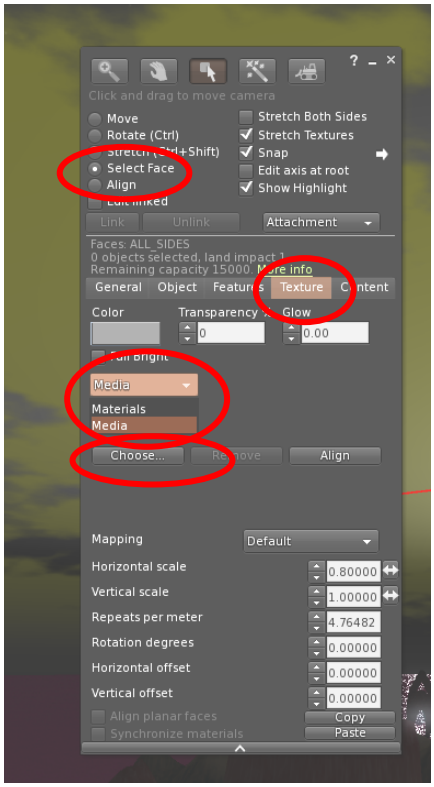
Δεν απαιτείται script, αλλά ακολουθούμε τα παρακάτω βήματα:

1. Τοποθετούμε το αρχείο ήχου .mp3 στο φάκελο Simonastick\www. Το αρχείο ήχου θα πρέπει να έχει bitrate μέχρι 192Kbps.
2. Στο φάκελο WWW, υπάρχουν 2 αρχεία με κατάληξη .html (video.html και sound_looping.html). Επιλέγουμε να επεξεργαστούμε το αρχείο sound_looping.html (ανοίγει και με Notepad-Σημειωματάριο).
3. Μεταβαίνουμε στο σημείο που παρακάτω είναι υπογραμμισμένο και απλά αλλάζουμε το όνομα του αρχείου σε αυτό που θα χρησιμοποιήσουμε:

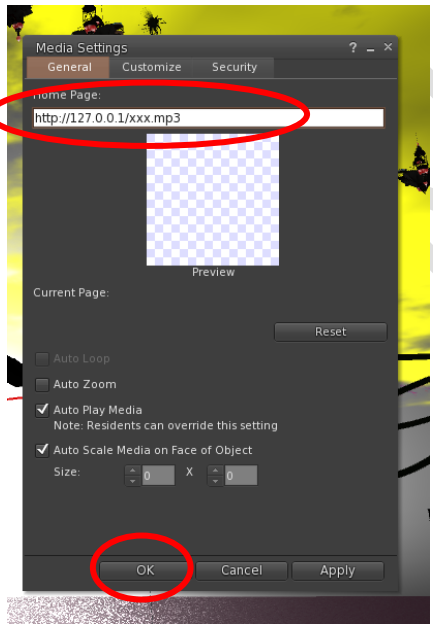
```
<script type="text/javascript">
   CodoPlayer("http://127.0.0.1/sound.mp3",

   {
      width: 1024,
      height: 768,
      autoplay: true,
      loop: true
   })
</script>
```

4. Αν το επιθυμούμε, αλλάζουμε τις δηλώσεις autoplay και loop, από true σε false. Με το autoplay ο ήχος παίζει αυτόματα και με το Loop ο ήχος παίζει αέναα.
5. Αποθηκεύουμε το αρχείο με το όνομα που επιθυμούμε
6. Τοποθετούμε στον εικονικό κόσμο ένα αντικείμενο (μπορεί να είναι και ένας απλός κύβος) στο οποίο κάνοντας κλικ θα παίζει ο ήχος.
7. Κάνουμε δεξί κλικ στο αντικείμενο και επιλέγουμε Edit.
8. Μεταβαίνουμε στην καρτέλα Texture.
9. Κάνουμε κλικ στο τετραγωνάκι Select Face που μας επιτρέπει να επιλέξουμε μόνο μια έδρα από το αντικείμενό μας. Παρατηρούμε ότι η συγκεκριμένη έδρα έχει αποκτήσει ένα κύκλο με έναν σταυρό στη μέση, που είναι η ένδειξη ότι η συγκεκριμένη έδρα είναι επιλεγμένη.

10. Επιλέγουμε Media από το πλήκτρο επιλογής Materials/Media και κάνουμε κλικ στο πλήκτρο Choose όπως φαίνεται στην παρακάτω εικόνα:



11. Στην καρτέλα που εμφανίζεται, πληκτρολογούμε: http://127.0.0.1/XXX.html (όπου XXX το όνομα του αρχείου που επεξεργαστήκαμε στα προηγούμενα βήματα) και πατάμε OK.



Παρατηρούμε ότι στο αντικείμενό μας εμφανίζονται χειριστήρια που μας επιτρέπουν να παίξουμε ή να σταματήσουμε τον ήχο.

5. Script που παίζει mp3 αρχεία με επιλογή

Πρέπει να βάλουμε τα αρχεία mp3 στο φάκελο simonastick\www και πρέπει να ονομάζονται 1.mp3, 2.mp3, κτλ

```
list music = [
   "Πρώτο κομμάτι",
   "Δεύτερο κομμάτι" ];

string url = "http://127.0.0.1/";
default
{
   touch_start(integer num)
   {
      // show dialog with available musics
      llDialog(llGetOwner(), "", [
         "Πρώτο κομμάτι",
         "Δεύτερο κομμάτι",
         "stop music"],
         -45);
   }
      state_entry()
   {
      llTargetOmega(<0.0, 0.0, 1.0>, 0.2, 1);
      llListen(-45, "", NULL_KEY, "");
   }
      listen(integer channel,string name,key id,string msg)
   {
      if (msg == "stop music") {
         // setting URL to empty string stops music playback
         llSetParcelMusicURL("");
      } else {
         // search position in music list
         integer len = llGetListLength(music);
         integer i;
         for (i = 0; i < len; i++) {
            string info = llList2String(music, i);
            if (llGetSubString(info, 0, llStringLength(msg) - 1) == msg) {
               // show info
               llWhisper(0, "loading music: " + info + "...");
               llWhisper(0, "click the play button at screen bottom");

               // stop previous music
               llSetParcelMusicURL("");

               // musics are named 1.ogg, 2.ogg etc.
               llSetParcelMusicURL(url + (string) (i+1) + ".mp3");
            }
         }
      }
   }
}
```

Άνοιγμα web σελίδων

Δεν απαιτείται script, αλλά ακολουθούμε τα βήματα τοποθέτησης αρχείου ήχου .mp3. Η μόνη διαφορά είναι ότι στο βήμα 7 γράφουμε τη διεύθυνση της ιστοσελίδας που θέλουμε να ανοίγει, πάντα όμως με το πρόθεμα http://.

Παρατήρηση: Το στερεό πρέπει να έχει διαστάσεις τέτοιες που να επιτρέπουν τη σωστή εμφάνιση της ιστοσελίδας.

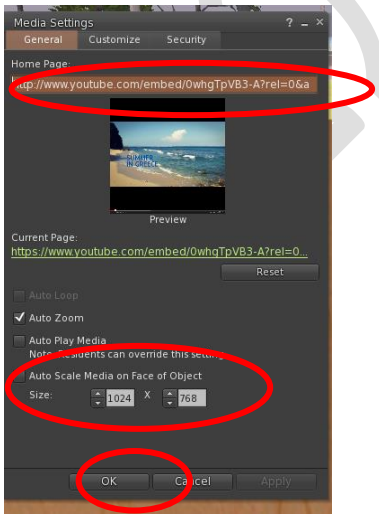
Παρατήρηση: Ο ενσωματωμένος φυλλομετρητής δεν είναι σε θέση να προβάλει σελίδες με περιεχόμενο activeX ή και Java. Όταν η σελίδα δεν προβάλλεται σωστά, θα πρέπει από τις ρυθμίσεις να ορίσετε οι σελίδες να ανοίγουν με εξωτερικό φυλλομετρητή (Preferences, Network)

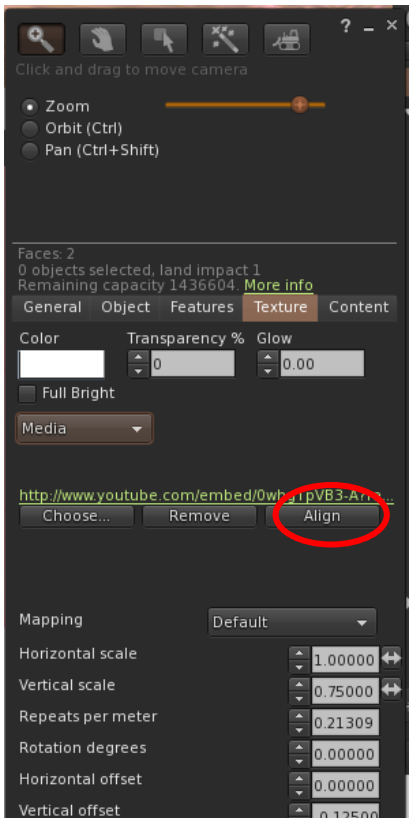
Προβολή video

Δεν απαιτείται script, αλλά ακολουθούμε τα βήματα τοποθέτησης αρχείου ήχου που παρουσιάστηκαν πιο πάνω. Το αρχείο που χρειάζεται να επεξεργαστείτε είναι το video.html. Τα αρχεία video καλό είναι να είναι ψηφιοποιημένα κατά mpeg (.mp4). Με κάποιο πρόγραμμα αποθήκευσης αρχείων από το Youtude, μπορούμε να "κατεβάσουμε" και να αποθηκεύσουμε στο φάκελο www τέτοια αρχεία.

Εναλλακτικά, μπορούμε να χρησιμοποιήσουμε υπερσύνδεσμο στο Youtube, με την εξής διαδικασία:

1. Αντιγράφουμε από τον υπερσύνδεσμο του Youtube μόνο το μέρος εκείνο που αφορά το ποιο βίντεο θέλουμε να παίξει, για παράδειγμα, από τον υπερσύνδεσμο:
   https://www.youtube.com/watch?v=0whgTpVB3-A χρειαζόμαστε μόνο το κομμάτι 0whgTpVB3-A
2. Στην επιφάνεια του αντικειμένου που θα παίξει το βίντεο, στην επιλογή Media, προσθέτουμε τον εξής σύνδεσμο, ως έχει αλλάζοντας μόνο το κομμάτι που αφορά το βίντεο:
   http://www.youtube.com/embed/0whgTpVB3-A?rel=0&autoplay=1;fs=0;autohide=0;hd=0;
   (παρατηρείστε το σημείο στο οποίο προσθέσαμε το βίντεο)
3. Αποεπιλέγουμε το Autoscale media on the face of the object και δίνουμε ως συγκεκριμένο μέγεθος το 1024X768 και κάνουμε κλικ στο OK για να επανέλθουμε στην αρχική καρτέλα
4. Κάνουμε κλικ στο πλήκτρο Align





Scripts εναλλαγής εικόνων

1. Slide show αυτόματο. Οι εικόνες πρέπει να είναι μέσα στην καρτέλα Content.

```
integer CrTXUR;
integer TotTXUR;
float wait = 5; //Number of seconds each picture is shown

default
{
   state_entry()
   {
llSetTimerEvent(wait);
   TotTXUR=llGetInventoryNumber(INVENTORY_TEXTURE);
}

   timer()
   {if(0 < TotTXUR)
      {if(CrTXUR < TotTXUR - 1)
         {   CrTXUR++; }

         else
         {  CrTXUR=0;
         TotTXUR=llGetInventoryNumber(INVENTORY_TEXTURE);}
          llSetTexture(llGetInventoryName(INVENTORY_TEXTURE, CrTXUR), ALL_SIDES); }
      }
}
```

2. Slide show με τις εικόνες να αλλάζουν με κλικ, σειριακά. Οι εικόνες πρέπει να είναι μέσα στην καρτέλα Content.

```
default
{

    touch_start(integer total_number)
   {
      integer number = llGetInventoryNumber(INVENTORY_TEXTURE);
      float rand = llFrand(number);
      integer choice = (integer)rand;
      string name = llGetInventoryName(INVENTORY_TEXTURE, choice);
      if (name != "")
         llSetTexture(name, 2);
   }
}
```

3. Slide show με μενού επιλογής εικόνας, οι εικόνες πρέπει να είναι μέσα στην καρτέλα Content.

```
integer i = 0;
integer currentPos = 0;

integer listener;
integer MENU_CHANNEL ;

// opens menu channel and displays dialog
Dialog(key id)
{
   list MENU1 = [];

   // count the textures in the prim to see if we need pages
   integer c = llGetInventoryNumber(INVENTORY_TEXTURE);
   if (c <= 12)
   {
      for (i = 0; i < c; i++ ) {
         MENU1 += llGetInventoryName(INVENTORY_TEXTURE, i);
      }
   }
   else
   {
      for (i = 10 * currentPos; i < (10 + (10 * currentPos)) ; i++) {

         // check to make sure name <= 24, or else the menu will not work.
         string aname = llGetInventoryName(INVENTORY_TEXTURE, i);
         if ( llStringLength(aname) >24)
         {
            llOwnerSay("The texture  named " + aname + " has too long of a name to be used, please give it a
shorter name <= 24 characters ");
         }
         else
         {
          if (i < c ) {
               MENU1 += aname;
           }
         }
      }
      MENU1 += ">>";
      if (currentPos != 0)
         MENU1 += "<<";
      else
         MENU1 += "-";
   }


   MENU_CHANNEL = (integer) (llFrand(10000) + 10000);
   listener = llListen(MENU_CHANNEL, "", NULL_KEY, "");

   llDialog(id, "Select one object below: ", MENU1, MENU_CHANNEL);
}
```

```
default
{
    on_rez(integer num)
    {
        // reset scripts on rez
        llResetScript();
    }

    touch_start(integer total_number)
    {
        // display the dialog
        Dialog(llDetectedKey(0));
    }

    listen(integer channel, string name, key id, string message)
    {
        if (channel == MENU_CHANNEL)
        {
            llListenRemove(listener);
            if (message == ">>")
            {
                currentPos ++;
                Dialog(id);
            }
            else if (message == "<<")
            {
                currentPos--;
                if (currentPos < 0)
                    currentPos = 0;
                Dialog(id);
            }
            else
            {
                // display the texture from menu selection
                llSetTexture(message, ALL_SIDES);

            }
        }
    }
}
```

Chat bot script

Απλό script διαφορετικών απαντήσεων κάθε φορά που κάνουμε κλικ στο αντικείμενο.

Σημείωση: Οι απαντήσεις επιλέγονται τυχαία.

```
default
{

state_entry()
      {
         llSetText("Κάντε κλικ για να σας δώσω απαντήσεις ",<1,1,1>,1.0);
      }


   touch_start(integer total_number)
   {
      list words = [
      "Ωραίος καιρός",
      "Θα βρέξει",
      "Θα χιονίσει",
      "Θα έχει ήλιο",
      "Θα έχει κρύο",
      "Θα έχει πολύ ζέστη",
      "Θα φυσάει",
      "Βροχερή μέρα σήμερα"];

      integer i = (integer)llFrand(llGetListLength(words));
      llSay(0,llList2String(words,i));

   }
}
```

Script παροχής πληροφοριών

Απαιτείται και μια notecard με το όνομα WelcomeInfo

Το script περιέχει:

```
// A message to say when clicked on.  Set to "" for no message
string message = "";

// A notecard to pass, when clicked on.  Set to "" for no notecard
// Could be any item in the object's inventory (e.g. a landmark).
string notecard = "WelcomeInfo";


default
{
```

```
    state_entry()
    {
        llSetText("Κάντε κλικ για πληροφορίες",<1,1,1>,1.0);
    }

    touch_start(integer num_detected)
    {
        // Say message, if set
        if (message != "") llSay(0, message);

        // Give notecard, if set
        if (notecard != "") llGiveInventory(llDetectedKey(0), notecard);

    }
}
```

Η notecard περιέχει:

Αυτές είναι οι πληροφορίες ...

Script quiz πολλαπλών επιλογών σε μενού

Απαιτείται και μια notecard με το όνομα quiz

Το script περιέχει:

```
string question;       // Text for the current question
string options;        // String of options - to be broken into a list
string posFeedback;    // Positive feedback text for current question
string negFeedback;    // Negative feedback for current question
string correct;        // Expected correct answer for current question

list questionSet;      // Full set of question text from notecard
list optionsSet;       // All option sets from notecard - one string of options per question
list posFeedbackSet;   // All positive feedback
list negFeedbackSet;   // All negative feedback
list correctSet;       // All correct answers

integer active;        // True if currently being used
integer currentLine;   // For extracting data from notecard
integer currentQ;      // For tracking number of current question
integer totalQs;       // Stores total number of questions extracted
integer score;         // Users score

integer liHandle;      // Handle to remove listener once quiz is complete

key quizUser;          // key of the avatar taking the quiz


// Function to reset variables between quiz uses
resetVars()
{
   question = "";
   options = "";
   posFeedback = "";
   negFeedback = "";
   correct = "";

   questionSet = [];
   optionsSet = [];
   posFeedbackSet = [];
   negFeedbackSet = [];
   correctSet = [];

   active = FALSE;
   currentLine = 0;
   currentQ = 0;
   totalQs = 0;
   score = 0;
}
```

```
// Function ro present feedback
presentFinal()
{
   string finalText = "\n";
   finalText += "Quiz complete \n";
   finalText += "You scored " + (string)score + " out of " + (string)totalQs;
   llDialog(quizUser, finalText, ["Close"], 12);
   active = FALSE;
   llListenRemove(liHandle);
}

// Function to present a question
presentQuestion()
{
   // Obtain list of options - seperated in string by vertical bars
   list theseOptions = llParseString2List(llList2String(optionsSet, currentQ), ["|"], []);

   // Output the question
   string dialogString = "\n" + (string)(currentQ+1) + ") " + llList2String(questionSet, currentQ) + "\n"; //
Question text

   integer i;
   list dialogOptions = [];
   for (i=0; i<llGetListLength(theseOptions); i++) // Options
   {
      //llSay(0, (string)(i+1) + "." + llList2String(theseOptions, i));  // puts numbers in front of options
      dialogOptions += [llList2String(theseOptions, i)];
   }

   llDialog(quizUser, dialogString, dialogOptions, 12);
}

default
{
   state_entry()
   {
      resetVars();
   }

   touch_start(integer num)
   {
      // If active
      if (active)
      {
         // Check if questions remain
         if (currentQ < totalQs)
         {
            // Trigger next question
            presentQuestion();
         }
         else
         {
            presentFinal();
```

```
        }
    }
    else
    {
        // Load questions from notecard
        resetVars();
        active = TRUE;
        quizUser = llDetectedKey(0);
        llGetNotecardLine("quiz", currentLine);
    }
}

// Rep[eatedly called when questions are being loaded
dataserver(key query_id, string data)
{
    // If not end of file
    if (data != EOF)
    {
        // Identiy type
        string type = llGetSubString(data, 0, 0);
        string value = llGetSubString(data, 1, -1);

        if (type != "#")
        {
            if (type == "?") question = value;
            else if (type =="*") options += value + "|";
            else if (type =="!") correct = value;
            else if (type =="+") posFeedback = value;
            else if (type =="-") negFeedback = value;

            currentLine++;
            llGetNotecardLine("quiz", currentLine);
        }
        else
        {
            //end of question block

            // Store options
            questionSet += [question];
            optionsSet += [llGetSubString(options, 0, -1)];
            posFeedbackSet += [posFeedback];
            negFeedbackSet += [negFeedback];
            correctSet += [correct];

            options = "";

            totalQs++;

            // Start loading next
            currentLine++;
            llGetNotecardLine("quiz", currentLine);
        }
    }
```

```
        else
        {
            // Present first question
            currentQ = 0;
            presentQuestion();

            // Start listeningj to input from person taking the quiz
            liHandle = llListen(12, "", quizUser, "");
        }
    }

    // Only listens to person that started the quiz
    listen(integer channel, string name, key id, string message)
    {
        // If matches correct answer
        if (message == "Next->")
        {
            if (currentQ < totalQs)
            {
                // Trigger next question
                presentQuestion();
            }
            else
            {
                presentFinal();
            }
        }
        else if (message == llList2String(correctSet, currentQ))
        {
            // Show positive feedback
            llDialog(quizUser, "\n" + llList2String(posFeedbackSet, currentQ), ["Next->"], 12);
            score++;
            currentQ++;
        }
        else
        {
            // Show negative feedback
            llDialog(quizUser, "\n" + llList2String(negFeedbackSet, currentQ), ["Next->"], 12);
            currentQ++;
        }
    }
}
```

Η notecard περιέχει (προσέξτε τη δομή των ερωτήσεων και απαντήσεων):

```
?Multiple choice: What colour is grass?
*Red
*Green
*Blue
!Green
+Correct
-The correct answer was B. Green
```

#
?What is the maximum number of letters for an option?
*7
*8
*9
!9
+Correct
-Wrong. The maximum is 9
#

Script κίνησης υφής

Η σύνταξη της εντολής είναι:
llSetTextureAnim(integer mode, integer face, integer sizex, integer sizey, float start, float length, float rate);

- integer mode – mask of Mode flags
- integer face – face number or ALL_SIDES
- integer sizex – horizontal frames (ignored for ROTATE and SCALE)
- integer sizey – vertical frames (ignored for ROTATE and SCALE)
- float start – Start position/frame number (or radians for ROTATE)
- float length – number of frames to display (or radians for ROTATE)
- float rate – frames per second (must not be zero)

ANIM_ON	0x01	Texture animation is on. This must be set to start the animation, cleared to stop it.
LOOP	0x02	Loop the texture animation.
REVERSE	0x04	Play animation in reverse direction.
PING_PONG	0x08	Play animation going forwards, then backwards.
SMOOTH	0x10	Slide in the X direction, instead of playing separate frames. In both SCALE and ROTATE modes, causes smooth transitions.
ROTATE	0x20	Animate texture rotation. Does not work with SCALE
SCALE	0x40	Animate the texture scale. Does not work with ROTATE

This slides a texture smoothly and loops it when it gets to the end.
llSetTextureAnim(ANIM_ON | SMOOTH | LOOP , ALL_SIDES, 1, 1, 1.0, 1.0, 1.0);
This slides a texture smoothly in the opposite direction
llSetTextureAnim(ANIM_ON | SMOOTH | LOOP , ALL_SIDES, 1, 1, 1.0, 1.0, -1.0);
This divides a texture into 64 "cells", 8 across, and 8 down, and flips through them, left to right, top to bottom. This is useful for cell animation.
llSetTextureAnim(ANIM_ON | LOOP, ALL_SIDES, 8, 8, 0.0, 64.0, 6.4);
This rotates a texture counter-clockwise at 2 revolutions per second. Change the last value to -2*TWO_PI to rotate clockwise.
llSetTextureAnim(ANIM_ON | SMOOTH | ROTATE | LOOP, ALL_SIDES,1,1,0, **TWO_PI**, 2***TWO_PI**);
This scales a texture larger and smaller.
llSetTextureAnim(ANIM_ON | SMOOTH | SCALE | PING_PONG | LOOP, ALL_SIDES, 1, 1, 1.0, 3.0, 2.0);
This turns off all texture animations
llSetTextureAnim(FALSE, ALL_SIDES, 0, 0, 0.0, 0.0, 1.0);

1. Απλό animation υφής

```
default {
   //this state will allow colors to be set
   state_entry() {
```

```
    llSetTextureAnim(ANIM_ON | LOOP, ALL_SIDES, 4, 4, 0, 0, 3);
  }
}
```

2. Αργή περιστροφή υφής

```
default
{
  state_entry()
  {
    llSetTextureAnim(ANIM_ON | SMOOTH | ROTATE | LOOP, ALL_SIDES,1,1,0, TWO_PI, 1);
  }
}// END //
```

3. Script αλλαγής διαφάνειας αντικειμένου, μετατροπής του σε phantom με βάση τις απαντήσεις του χρήστη

```
default

{

  touch_start(integer total_number)

  {

    llListen( 0, "", NULL_KEY, "" );

     llWhisper ( 0, "άσπρο ή κάτι άλλο;" ); // η ερώτηση

  }

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

  {

    if ( message == "άσπρο" ) // αν είναι η σωστή απάντηση

    {

         llWhisper( 0, "Σωστά!" ); // πες κάτι

        llPlaySound("ELSHOT",1);  //παίξε κάποιον ήχο που είναι μέσα στο content

        llSetAlpha(0.0, ALL_SIDES); // γίνε διάφανο

         llSetStatus(STATUS_PHANTOM, TRUE); // γίνε φάντασμα
```

```
            llSetTimerEvent(20); // σε 20 δευτερόλεπτα θα γίνει κάτι
        }
        else
        {
           llWhisper( 0, "Λάθος!" ); //πες κάτι για τη λάθος απάντηση
        }
    }
    timer()
{
            llSetAlpha(1.0, ALL_SIDES); // γίνε αδιαφανές
             llSetStatus(STATUS_PHANTOM, FALSE); // πάψε να είσαι φάντασμα
}

}
```

Scripts σωματιδίων

1. Καπνός (χρειάζεται υφή στο content, η υφή πρέπει να είναι ελαφρώς διάφανη)

```
generalParticleEmitterOn()
{
   llParticleSystem([
      PSYS_PART_FLAGS , 0
   //| PSYS_PART_BOUNCE_MASK      //Bounce on object's z-axis
   | PSYS_PART_WIND_MASK         //Particles are moved by wind
   | PSYS_PART_INTERP_COLOR_MASK   //Colors fade from start to end
   | PSYS_PART_INTERP_SCALE_MASK   //Scale fades from beginning to end
   | PSYS_PART_FOLLOW_SRC_MASK     //Particles follow the emitter
   | PSYS_PART_FOLLOW_VELOCITY_MASK//Particles are created at the velocity of the emitter
   //| PSYS_PART_TARGET_POS_MASK   //Particles follow the target
   | PSYS_PART_EMISSIVE_MASK      //Particles are self-lit (glow)
   //| PSYS_PART_TARGET_LINEAR_MASK//Undocumented--Sends particles in straight line?
   ,

   //PSYS_SRC_TARGET_KEY , NULL_KEY,//The particles will head towards the specified key
   //Select one of the following for a pattern:
   //PSYS_SRC_PATTERN_DROP              Particles start at emitter with no velocity
   //PSYS_SRC_PATTERN_EXPLODE           Particles explode from the emitter
   //PSYS_SRC_PATTERN_ANGLE             Particles are emitted in a 2-D angle
   //PSYS_SRC_PATTERN_ANGLE_CONE        Particles are emitted in a 3-D cone
   //PSYS_SRC_PATTERN_ANGLE_CONE_EMPTY    Particles are emitted everywhere except for a 3-D cone

   PSYS_SRC_PATTERN,         PSYS_SRC_PATTERN_ANGLE_CONE

   ,PSYS_SRC_TEXTURE,        "smoke3"      //UUID of the desired particle texture, or inventory name
   ,PSYS_SRC_MAX_AGE,         0          //Time, in seconds, for particles to be emitted. 0 = forever
   ,PSYS_PART_MAX_AGE,        50         //Lifetime, in seconds, that a particle lasts
   ,PSYS_SRC_BURST_RATE,      10         //How long, in seconds, between each emission
   ,PSYS_SRC_BURST_PART_COUNT,  50           //Number of particles per emission
   ,PSYS_SRC_BURST_RADIUS,     10.0         //Radius of emission
   ,PSYS_SRC_BURST_SPEED_MIN,   3           //Minimum speed of an emitted particle
   ,PSYS_SRC_BURST_SPEED_MAX,   8           //Maximum speed of an emitted particle
   ,PSYS_SRC_ACCEL,          <0.0,0,.05>   //Acceleration of particles each second
   ,PSYS_PART_START_COLOR,     <1.0,1.0,1.0>  //Starting RGB color
   ,PSYS_PART_END_COLOR,       <1.0,1.0,1.0>  //Ending RGB color, if INTERP_COLOR_MASK is on
   ,PSYS_PART_START_ALPHA,     0.6          //Starting transparency, 1 is opaque, 0 is transparent.
   ,PSYS_PART_END_ALPHA,       0.0          //Ending transparency
   ,PSYS_PART_START_SCALE,     <7,7,7>  //Starting particle size
   ,PSYS_PART_END_SCALE,       <7,7,7>  //Ending particle size, if INTERP_SCALE_MASK is on
   ,PSYS_SRC_ANGLE_BEGIN,      10 * DEG_TO_RAD //Inner angle for ANGLE patterns
   ,PSYS_SRC_ANGLE_END,        20 * DEG_TO_RAD//Outer angle for ANGLE patterns
   ,PSYS_SRC_OMEGA,           <0.0,0.0,0.0>  //Rotation of ANGLE patterns, similar to llTargetOmega()
        ]);
}

generalParticleEmitterOff()
```

```
{
   llParticleSystem([]);
}

default
{
   state_entry()
   {
      generalParticleEmitterOn();
   }

   touch_start( integer num )
   {
      // Uncomment the following line to allow this to be turned off
      //state off;
   }
}

state off
{
   state_entry()
   {
      generalParticleEmitterOff();
   }

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

2. Χιόνι (χρειάζεται υφή στο content, η υφή πρέπει να είναι ελαφρώς διάφανη)

```
default
{
   state_entry()
   {
       llParticleSystem([
      PSYS_PART_FLAGS , 0
   //| PSYS_PART_BOUNCE_MASK      //Bounce on object's z-axis
   | PSYS_PART_WIND_MASK          //Particles are moved by wind
   | PSYS_PART_INTERP_COLOR_MASK   //Colors fade from start to end
   | PSYS_PART_INTERP_SCALE_MASK   //Scale fades from beginning to end
   | PSYS_PART_FOLLOW_SRC_MASK     //Particles follow the emitter
   | PSYS_PART_FOLLOW_VELOCITY_MASK//Particles are created at the velocity of the emitter
   //| PSYS_PART_TARGET_POS_MASK   //Particles follow the target
   | PSYS_PART_EMISSIVE_MASK      //Particles are self-lit (glow)
   //| PSYS_PART_TARGET_LINEAR_MASK//Undocumented--Sends particles in straight line?
   ,
```

```
   //PSYS_SRC_TARGET_KEY , NULL_KEY,//The particles will head towards the specified key
   //Select one of the following for a pattern:
   //PSYS_SRC_PATTERN_DROP             Particles start at emitter with no velocity
   //PSYS_SRC_PATTERN_EXPLODE          Particles explode from the emitter
   //PSYS_SRC_PATTERN_ANGLE            Particles are emitted in a 2-D angle
   //PSYS_SRC_PATTERN_ANGLE_CONE       Particles are emitted in a 3-D cone
   //PSYS_SRC_PATTERN_ANGLE_CONE_EMPTY Particles are emitted everywhere except for a 3-D cone

   PSYS_SRC_PATTERN,        PSYS_SRC_PATTERN_ANGLE_CONE

   ,PSYS_SRC_TEXTURE,           "snow_flake_particle_01"          //UUID of the desired particle texture, or
inventory name
   ,PSYS_SRC_MAX_AGE,        0.0         //Time, in seconds, for particles to be emitted. 0 = forever
   ,PSYS_PART_MAX_AGE,       30.0        //Lifetime, in seconds, that a particle lasts
   ,PSYS_SRC_BURST_RATE,     .01         //How long, in seconds, between each emission
   ,PSYS_SRC_BURST_PART_COUNT, 24        //Number of particles per emission
   ,PSYS_SRC_BURST_RADIUS,   10.0        //Radius of emission
   ,PSYS_SRC_BURST_SPEED_MIN, 0.1        //Minimum speed of an emitted particle
   ,PSYS_SRC_BURST_SPEED_MAX, 0.5        //Maximum speed of an emitted particle
   ,PSYS_SRC_ACCEL,          <0,0,-.20>  //Acceleration of particles each second
   ,PSYS_PART_START_COLOR,   <1,1,1>  //Starting RGB color
   ,PSYS_PART_END_COLOR,     <1,1,1>  //Ending RGB color, if INTERP_COLOR_MASK is on
   ,PSYS_PART_START_ALPHA,   1.0       //Starting transparency, 1 is opaque, 0 is transparent.
   ,PSYS_PART_END_ALPHA,     1.0       //Ending transparency
   ,PSYS_PART_START_SCALE,   <.05,.05,.05>  //Starting particle size
   ,PSYS_PART_END_SCALE,     <.05,.05,.05>  //Ending particle size, if INTERP_SCALE_MASK is on
   ,PSYS_SRC_ANGLE_BEGIN,    90 * DEG_TO_RAD //Inner angle for ANGLE patterns
   ,PSYS_SRC_ANGLE_END,      90 * DEG_TO_RAD//Outer angle for ANGLE patterns
   ,PSYS_SRC_OMEGA,          <0.0,0.0,0.0>  //Rotation of ANGLE patterns, similar to llTargetOmega()
        ]);
   }
}
```

3. Κι άλλο χιόνι

```
integer flag = FALSE;

on()
{
   llParticleSystem( [
PSYS_PART_FLAGS, 0,
PSYS_PART_START_COLOR, <3.10000, 2.30000, 1.70000>,
PSYS_PART_END_COLOR, <-0.10000, -0.60000, -1.70000>,
PSYS_PART_START_SCALE, <0.050000, 0.05000, 0.00000>,
PSYS_PART_END_SCALE, <0.00000, 0.00000, 0.00000>,
PSYS_SRC_PATTERN, 8,
PSYS_SRC_BURST_RATE, 0.000000,
PSYS_SRC_ACCEL, <0.00000, 0.00000, -0.40000>,
PSYS_SRC_BURST_PART_COUNT, 50,              // increase for more snow
PSYS_SRC_BURST_RADIUS, 0.000000,
```

```
PSYS_SRC_BURST_SPEED_MIN, 0.000000,
PSYS_SRC_BURST_SPEED_MAX, 1,          // increase for longer distance
PSYS_SRC_TARGET_KEY, (key)"",
PSYS_SRC_ANGLE_BEGIN, 3.141593,
PSYS_SRC_ANGLE_END, 6.283185,
PSYS_SRC_OMEGA, <0.00000, 0.10000, 0.00000>,
PSYS_SRC_MAX_AGE, 0.00000,
PSYS_SRC_TEXTURE, "snow_flake_particle_01",  // change this to any texture name that is also in the prim, or use the UUID. If you use the UUID and want to gice it to others, make sure the texture in your inventory has copy on it.
PSYS_PART_START_ALPHA, 0.300000,      // increase for less transparent snow
PSYS_PART_END_ALPHA, 0.00000
                    ] );

}

default
{
   state_entry()
   {
      on();
   }


   on_rez(integer prim)
   {
      llResetScript();
   }
}
```

4. Σωματίδια προς όλες τις κατευθύνσεις

```
default
 {
    state_entry()
    {
       llParticleSystem([
                    PSYS_SRC_PATTERN, PSYS_SRC_PATTERN_EXPLODE,

                    PSYS_SRC_MAX_AGE, 0.,
                    PSYS_PART_MAX_AGE, 9.,

                    PSYS_SRC_BURST_RATE, 5.,
                    PSYS_SRC_BURST_PART_COUNT, 500,

                    PSYS_SRC_BURST_RADIUS, .1,
                    PSYS_SRC_BURST_SPEED_MIN, 3.,
                    PSYS_SRC_BURST_SPEED_MAX, 3.,
                    PSYS_SRC_ACCEL, <0.0,0.0,-0.8>,
```

```
                PSYS_PART_START_COLOR, <246,38,9>/255.,
                PSYS_PART_END_COLOR, <246,38,9>/255,

                PSYS_PART_START_ALPHA, 0.9,
                PSYS_PART_END_ALPHA, 0.0,

                PSYS_PART_START_SCALE, <.3,.3,0>,
                PSYS_PART_END_SCALE, <.1,.1,0>,

                PSYS_PART_FLAGS
                , 0
                | PSYS_PART_EMISSIVE_MASK
                | PSYS_PART_INTERP_COLOR_MASK
                | PSYS_PART_INTERP_SCALE_MASK
                | PSYS_PART_FOLLOW_VELOCITY_MASK
                | PSYS_PART_WIND_MASK
                ]);
    }
}
```

5. Φωτιά

Δείτε το αντικείμενο flaming barrel από το φάκελο lights and fire

6. Δείτε το φάκελο particles

Scripts για φώτα

1. Απλό φως.

Το απλό φως δεν χρειάζεται script, παρά μόνο στην καρτέλα feature να δηλώσουμε light

2. Δέσμη φωτός

Μπορούμε να πετύχουμε το εφέ της δέσμης φωτός χωρίς τη χρήση script, απλά κάνοντας το στερεό ημιδιαφανές και glow.

3. Script για στερεό που ανάβει ως φως με το άγγιγμα

```
// Touch the object to light it up.
// Lighting is configurable.

integer light_s = TRUE;
vector lightcolor = <1.0, 0.75, 0.5>;
float intensity = 1.0; // 0-1
float radius = 10.0;   // 0-10
float falloff = 0.0;   // 0-1
float glow = 1.0;

switchit()
{
   float thisglow = 0.0;
   light_s = !light_s;
   if (light_s)
   {
      thisglow = glow;
   }
   llSetPrimitiveParams([
      PRIM_POINT_LIGHT, light_s, lightcolor, intensity, radius, falloff,
      PRIM_FULLBRIGHT, ALL_SIDES, light_s,
      PRIM_GLOW, ALL_SIDES, thisglow
   ]);
    llSetColor(lightcolor, ALL_SIDES);
}

default
{
   state_entry()
   {
    llSetText("Touch for light",<0,0,0>,1);
    switchit();
   }
```

```
    touch_start(integer total_number)
    {
        switchit();
    }
}
```

4. Script για φως που ανάβει όταν βραδιάζει

```
// The lights fade on and off according to the suns position above the horizon

vector lightColour = < 1.0, 1.0, 1.0> ;
float lightIntensity = 1.0 ;
float lightRadius = 8.0 ;
float lightFalloff = 0.75 ;


default
{
state_entry()
{
// Function entry
llSetTimerEvent(10); // Check every X seconds
}

timer()
{
integer light1 ;
vector sun = llGetSunDirection();
float intensity = (0.25 - sun.z)*4;
if ( intensity < 0.0 ) {
light1 = FALSE ;
intensity = 0.0;
} else {
light1 = TRUE ;
if ( intensity > 1.0 ) intensity = 1.0 ;
}
// Turn light on or off depending on intensity
llSetPrimitiveParams([PRIM_POINT_LIGHT, light1, lightColour , intensity, lightRadius, lightFalloff ]);
}
}
```

Scripts για κίνηση

1. Απλή κίνηση προς μια κατεύθυνση, όταν αγγίζουμε το αντικείμενο

```
default
{
        touch_start(integer total_number) {
                llSetPos(llGetPos()+<0,0,2>);
        }
}
```

Δοκιμάστε το script κάνοντας το αντικείμενο physical και δίνοντας κίνηση προς τα πάνω

2. Απλή παλινδρομική κίνηση, όταν αγγίζουμε το αντικείμενο

```
default
{
   touch_start(integer total_number)
   {
      llSetPos(llGetPos()+<0,0,2>);
      llSetPos(llGetPos()-<0,0,2>);
   }

}
```

3. Απλή κίνηση όταν αγγίζουμε το αντικείμενο και επαναφορά όταν το ξαναγγίζουμε

```
integer open = FALSE;

default
{
   touch_start(integer total_number) {
      if (open == FALSE) {
```

```
        open = TRUE;
        llSetPos(llGetPos()+<0,0,2>);
      } else {
        open = FALSE;
        llSetPos(llGetPos()-<0,0,2>);
      }

   }

}
```

4. Απλή περιστροφική κίνηση (σαν πόρτα που ανοιγοκλείνει)

```
integer open = FALSE;


default
{

   touch_start(integer total_number) {

      if (open == FALSE) {
        open = TRUE;
        llSetRot(llEuler2Rot(<0, 0, PI_BY_TWO>) * llGetRot());
      } else {
        open = FALSE;
        llSetRot(llEuler2Rot(<0, 0, -PI_BY_TWO>) * llGetRot());
      }

   }

}
```

5. Απλή κίνηση προς μια κατεύθυνση ανά Χ δευτερόλεπτα

Προσοχή το αντικείμενο θα κινείται για πάντα και τελικά θα χαθεί αν το βάλετε να κινείται προς τα πάνω ή θα φτάσει στα όρια του κόσμου αν κινείται προς άλλη κατεύθυνση.

```
default
{

   state_entry() {
      llSetTimerEvent(5);
   }
   timer() {
```

```
    llSetPos(llGetPos()+<0,0,0.5>);
  }

}
```

6. Απλή κίνηση προς μια κατεύθυνση ανά Χ δευτερόλεπτα μόλις ακουμπήσουμε το αντικείμενο

Προσοχή το αντικείμενο θα κινείται για πάντα και τελικά θα χαθεί αν το βάλετε να κινείται προς τα πάνω ή θα φτάσει στα όρια του κόσμου αν κινείται προς άλλη κατεύθυνση.

```
default
{

touch_start(integer total_number)
{
    llSetTimerEvent(5);
  }
  timer() {

    llSetPos(llGetPos()+<0,0,0.5>);
  }

}
```

7. Κίνηση αντικειμένου σε τυχαία θέση εντός ορισμένων ορίων ανά Χ δευτερόλεπτα

```
float maxX = 150;    // 255 max
float minXY = 10;    // get no closer than 10 meters to any edge
float maxY= 180;     // 255 max
float minH = 35;   // go no lower than this
float maxH = 100;    // go no higher than this

default
{
  touch_start(integer total_number)
  {
    llSetTimerEvent(5.0);          // every 5 seconds
  }

  timer()
  {
    vector C;

    C= llGetPos();

    if (C.y > maxY) // N
      C.y -= 20;
```

```
      if (C.x > maxX)  // E
         C.x -= 20;

      if (C.x < minXY)   // W
         C.x += 20;

      if (C.y  < minXY)    // S
         C.y += 20;

      if (C.z > maxH)   // U
         C.z -= 20;

      if (C.z < minH)    // D
         C.z += 20;


      float sign = llFrand(2.0);
      if (sign > 1.0)
         C.x += llFrand(10);
      else
         C.x -= llFrand(10);

      sign = llFrand(2.0);
      if (sign > 1.0)
         C.y += llFrand(10);
      else
         C.y -= llFrand(10);

      sign = llFrand(2.0);
      if (sign > 1.0)
         C.z += llFrand(10);
      else
         C.z -= llFrand(10);


      llSetPos(C);

   }
}
```


8. Κίνηση σε συγκεκριμένες θέσεις

```
integer current_movement = 0;
list coords= [<0,0,0>,<1,0,0>,<1,1,0>,<0,1,0>];    // make a 1 meter square
vector home;

default
{
```

```
    touch_start(integer total_number)
   {
     home = llGetPos();    // Remember where we parked!
    llSetTimerEvent(1.0);  //get it started
   }
    timer()
   {
      vector delta = llList2Vector(coords,current_movement);

      llSetPos(home + delta);  // move relative to home

       current_movement++;
      if (current_movement > llGetListLength(coords))
         current_movement = 0;

       llSetTimerEvent( llFrand(5.0) +5 );  //loop every 5 to 10 seconds
   }
   on_rez(integer start_param)
   {
      llResetScript();
   }

}
```

9. Script καταγραφής διαδοχικών θέσεων ενός αντικειμένου και στη συνέχεια παιξίματος της κίνησης που προκύπτει από αυτές.

Θα χρειαστούμε ένα prim το οποίο θα κινήσουμε στις διάφορες θέσεις και πρέπει να περιέχει το παρακάτω script. Το ακουμπάμε στην κάθε θέση, ώστε να καταγραφεί η θέση του, το μετακινούμε σε νέα θέση το ακουμπάμε πάλι, κοκ. Αν στο chat πληκτρολογήσουμε get positions, θα εμφανιστεί μια σειρά από συντεταγμένες τις οποίες πρέπει να αντιγράψουμε και να επικολλήσουμε σε συγκεκριμένη θέση μέσα στο script του αντικειμένου που θα κινείται.

```
string positions = "";
string quote = "\"";
string linebreak = "\n";

default
{
state_entry()
{
llListen( 0, "", NULL_KEY, "get positions");
}

touch_start(integer a)
{
```

```
if(positions != "")
positions += "," + linebreak;
positions += (quote + (string)llGetPos() + quote);
}

listen( integer channel, string name, key id, string message )
{
if ( id == llGetOwner() )
{
llSay(0, positions);
}
}
}
```

Το αντικείμενο που θα κινείται καλό θα είναι να το κάνουμε phantom ώστε να διαπερνά άλλα αντικείμενα που πιθανώς να εμποδίζουν την κίνησή του. Στο αντικείμενο που θα κινείται, προσθέστε το παρακάτω script (προσοχή η λίστα με τα waypoints είναι ενδεικτική)

```
// Script Name Train Engine
// Version 1.3
// Original posted by Nebadon
// Modified by Slow Putzo May 9, 2011

// Configuration/Settings
   // Sounds
      // Άφιξη σε στάση
         string stop = ""; //στα αυτάκια μπορείτε να δηλώσετε όνομα αρχείου ήχου (να περιέχεται στο
αντικείμενο)
         float stopvolume = 1.0;
      // Ενώ κινείται
         string running = ""; //στα αυτάκια μπορείτε να δηλώσετε όνομα αρχείου ήχου (να περιέχεται στο
αντικείμενο)
         float runningvolume = 1.0;
      // Σε ακινησία
         string idle = ""; //στα αυτάκια μπορείτε να δηλώσετε όνομα αρχείου ήχου (να περιέχεται στο
αντικείμενο)
         float idlevolume = 1.0;
      // Ενώ ξεκινά από στάση
         string start = ""; //στα αυτάκια μπορείτε να δηλώσετε όνομα αρχείου ήχου (να περιέχεται στο
αντικείμενο)
         float startvolume = 1.0;
         float startsoundlength = 0.5;
   // Multiple Trains
      integer enginechannel = -549816546;
   // Speed
      float speed = 20.0;  //μεγαλύτερη τιμή = μικρότερη ταχύτητα!!!
   // Train Offset
      vector cpoffset = <0.0, 0.0, 0.0>;
```

// Train waypoints, εδώ θα βάλετε τις συντεταγμένες, μετά από το > μπορείτε να βάλετε μια τιμή σε δευτερόλεπτα που θα σταματήσει πχ :10 Μετά μπορείτε να βάλετε το μήνυμα που θα εμφανίσει πχ :Μήνυμα: (προσοχή στη σύνταξη...

```
list waypoints =
[

// Εδώ επικολλάτε τις συντεταγμένες που θα προκύψουν από το get positions

 "<86.750000,40.737026,22.090572>:3:Αρχή διαδρομής παρακαλώ περιμένετε (3 δευτερόλεπτα):",
"<81.750000,18.343945,22.090572>:10:Ένα ωραίο πιάνο (10 δευτερόλεπτα):",
"<96.580795,21.413113,22.090572>:5:Να και ένα ψηλό δένδρο (5 δευτερόλεπτα):",
"<98.381363,34.500000,22.090572>:7:Άλογα! (7 δευτερόλεπτα):",
"<97.294624,40.015301,22.090572>:5:Δεξαμενές με χημικά (5 δευτερόλεπτα):",
"<93.928665,40.015301,22.090572>"

];


// Initialize global variables

integer waypoint = 0;
integer numwaypoints = 0;
vector next_pos = ZERO_VECTOR;
string cpdata = "";
string spause = "";
float pause = 0.0;
string csname = "";
string nsname = "";
integer cpdataloc;
rotation rot;
vector increment;
integer i;
integer soundstart = 0;

// code starts here

default
{
   on_rez(integer start_param)
   {
      llResetScript();
   }

   state_entry()
   {
      numwaypoints = llGetListLength(waypoints);
      waypoint = 0;
         }

   touch_start(integer total_number)
   {
```

```
      waypoint = 0;
      state run_route;
   }
}

state run_route
{
   state_entry()
   {
      if (waypoint < numwaypoints)
      {

       state next_waypoint;

      }
      else
      {
       llSay(0, "Τέλος διαδρομής");
       state default;
       // Αν βάλουμε // στις 2 πάνω γραμμές και βγάλουμε τα // από τις 2 κάτω γραμμές, η κίνηση θα είναι
αέναη

       //waypoint = 0;
       //state next_waypoint;
      }
   }
}

state next_waypoint
{
   state_entry()
   {
      next_pos = ZERO_VECTOR;
      cpdata = llList2String(waypoints, waypoint);
      spause = "";
      pause = 0.0;
      csname = "";
      nsname = "";
      cpdataloc = llSubStringIndex(cpdata, ":");
      if (cpdataloc == -1)
      {
         next_pos = ((vector)cpdata + cpoffset);
      }
      else
      {
         next_pos = ((vector)llGetSubString(cpdata, 0, cpdataloc - 1) + cpoffset);
         cpdata = llDeleteSubString(cpdata, 0, cpdataloc);
         cpdataloc = llSubStringIndex(cpdata, ":");
         spause = llGetSubString(cpdata, 0, cpdataloc - 1);
         if (llGetSubString(spause, -1, -1) == "s" && stop != "")
          {
            llStopSound();
            llPlaySound(stop, stopvolume);
```

```
            pause = (float)llGetSubString(cpdata, 0, cpdataloc - 2);
          }
        else
          {
            pause = (float)llGetSubString(cpdata, 0, cpdataloc - 1);
          }
        cpdata = llDeleteSubString(cpdata, 0, cpdataloc);
        cpdataloc = llSubStringIndex(cpdata, ":");
        if (cpdataloc != -1)
          {
            if (idle != "")
              {
                llLoopSound(idle, idlevolume);
              }
            csname = llGetSubString(cpdata, 0, cpdataloc - 1);
            cpdata = llDeleteSubString(cpdata, 0, cpdataloc);
            nsname = llGetSubString(cpdata, 0, -1);
             if (llStringLength(csname) > 0) llSay(0, "" + csname + ".");
          }
        if (pause > 0.0)
          {
            llSleep(pause);
            if (start != "")
              {
                llStopSound();
                llPlaySound(start, startvolume);
                llResetTime();
                soundstart = 1;
              }
          }
        //if (llStringLength(nsname) > 0) llWhisper(0, "" + "" + "");
      }
      if (soundstart == 1)
       {
         if (llGetTime() >= startsoundlength)
          {
            soundstart = 0;
            llLoopSound(running, runningvolume);
          }
       }
      llShout(enginechannel, "nextpos");
      rot = llGetRot() * llRotBetween(<0,0,1> * llGetRot(), next_pos - llGetPos());
      increment = (next_pos - llGetPos()) / speed;
      for(i=0;i<speed;++i)
       {
         llSetPrimitiveParams([PRIM_POSITION, (llGetPos() + increment), PRIM_ROTATION, rot]);
       }
      ++waypoint;
      state run_route;
    }
}
```

Σημαντικό!

Βάλτε ως αντικείμενο κίνησης έναν απλό κύβο (καλό θα είναι να αλλοιώσετε λίγο το σχήμα του ώστε να δείχνει προς την κατεύθυνση της 1ης κίνησης). Όταν τρέξετε το script θα παρατηρήσετε ότι αυτός περιστρέφεται κάπως. Αφήστε το script να ολοκληρώσει τη διαδρομή και περιστρέψτε τον κύβο ώστε να έχει σωστή θέση. Ξανατρέξτε το script και θα δείτε ότι τώρα κινείται κανονικά. Τώρα μπορείτε να συνδέσετε σε αυτόν (link) οποιοδήποτε άλλο περίπλοκο αντικείμενο, προσέχοντας βέβαια ο κύβος να είναι parent prim και όχι child (τον επιλέγουμε τελευταίο πριν κάνουμε link).

Script χρήσης της εντολής llMoveToTarget (Προσοχή! Τα παρακάτω script από την έκδοση 0.8 και μετά δεν μπορούν να εκτελεστούν!)

Η εντολή llMoveToTarget επιτρέπει την ομαλή κίνηση απλών prim αλλά και περίπλοκων αντικειμένων, σε αντίθεση με την εντολή llSetPos.

Σύνταξη: llMoveToTarget(vector target, float tau);
Όπου: vector target=συντεταγμένες του κόσμου και float tau=επιβράδυνση (καλή τιμή η 0.2 ή η 0.4)

Προσοχή! Απαραίτητη προϋπόθεση είναι το αντικείμενο που πρόκειται να κινηθεί να είναι **physical**.

Προσοχή! Η κίνηση είναι επιβραδυνόμενη. Στο χρόνο (t) που θα έχει οριστεί ως μεταβλητή σε άλλο σημείο του script, το αντικείμενο θα εκτελέσει περίπου το 64% της διαδρομής του, σε χρόνο 2t το 87%, σε χρόνο 4t το 98% και σε χρόνο 6t το 99%. Αυτό δημιουργεί την ανάγκη προσεκτικού συνδυασμού χρόνου και απόστασης, έτσι ώστε το τελικό αποτέλεσμα να ικανοποιεί τις σχεδιαστικές ανάγκες.

Σημείωση: Στο Opensim η κίνηση που προκύπτει μπορεί να είναι εντελώς διαφορετική από την επίσημη τεκμηρίωση της εντολής. Το αντικείμενο μπορεί να φτάνει στον προορισμό του στον μισό από τον οριζόμενο χρόνο.

Σημείωση: Η εντολή llMoveToTarget μπορεί να συνδυαστεί με την εντολή llStopMoveToTarget( ) η οποία και σταματά κάθε κίνηση

Σημείωση: Αντικείμενα με βάρος μεγαλύτερο των 800 κιλών θα κινηθούν μέχρι ένα σημείο, αλλά μετά θα τα υπερνικήσει η βαρύτητα.

1. Απλό script κίνησης κάθε φορά που ακουμπάμε το αντικείμενο
Παρατηρείστε τι θα συμβεί σε 25 δευτερόλεπτα

```
vector startPos;
float gTimetostop = 25.0;

default
    {
    touch_start( integer num )
    {

 llSetTimerEvent(gTimetostop);
llSetStatus(STATUS_PHYSICS, TRUE);
```

```
        startPos = llGetPos();


llMoveToTarget(startPos + <0,0,10>, 0.4);
   }

   timer()
{
llStopMoveToTarget( );
}
}
```

2. Script Avatar follower

```
vector vpos;
default
{
touch_start(integer total_number)
{
vpos = llGetPos();
llSetStatus(STATUS_PHYSICS,TRUE);
llSensorRepeat("", llDetectedKey(0), AGENT, 30, TWO_PI, 3.0);
}
sensor(integer total_number)
{
llMoveToTarget(llDetectedPos(0) + <-0.5,-0.5,0.0>, 0.2);
llRotLookAt(llDetectedRot(0), 0.5, 0.5);
}
no_sensor()
{
llMoveToTarget(vpos, 0.2);
llSensorRemove();
}

}
```

3. Script μετακίνησης σε συγκεκριμένες θέσεις

```
vector StartPos;
default
{
touch_start(integer total_number)
{
llSetStatus(STATUS_PHYSICS, TRUE);
StartPos = llGetPos();
//η κάτω γραμμή είναι ίδια με το llSetStatus(STATUS_PHYSICS, TRUE);
llSetStatus(1, 1);
llLookAt(<5.0, 0.0, 0.0 > +llGetPos(), 2, 0.5);
llMoveToTarget(<5.0, 0.0, 0.0 > +llGetPos(), 3);
llSleep(3);
llStopMoveToTarget();
llStopLookAt();
```

```
llLookAt(<0.0, 5.0, 0.0 > + llGetPos(), 2, 0.5);
llMoveToTarget(<0.0, 5.0, 0.0 > +llGetPos(), 3);
llSleep(3);
llStopMoveToTarget();
llStopLookAt();
llLookAt(<-5.0, 0.0, 0.0 > + llGetPos(), 2, 0.5);
llMoveToTarget(<-5.0, 0.0, 0.0 > +llGetPos(), 3);
llSleep(3);
llStopMoveToTarget();
llStopLookAt();
llLookAt(StartPos, 2, 0.5);
llMoveToTarget(StartPos, 3);
llSleep(3);
llStopMoveToTarget();
llStopLookAt();
llSetStatus(1, 0);
// η πάνω γραμμή είναι ίδια με το llSetStatus(STATUS_PHYSICS, FALSE);
}
}
```

4. Απλό script οδήγησης σκαφών
(δείτε και το αντικείμενο cabin cruiser+motor στο φάκελο Oblects\Vehicles\Boats)

```
// Very simple vehicle script, mod for OpenSim & ODE & VEHICLE code
// By Kitto Flora  September 2009

// remodified by Hiro Protagonist for basic sailing aquatic motorcraft
// September 2009

// grafted in from Owen Oyen's work implementing sail force simulations

// hacked on by christy lock



float speed;
float speedAdjust = 0.05;
key agent;

vector angular_motor;
vector Sitpos = <0.3,0.35,3.0>; // εδώ βάζετε τη θέση που θα καθήσει το avatar

rotation Sitrot;
integer tt;


string sit_message = "drive";
```

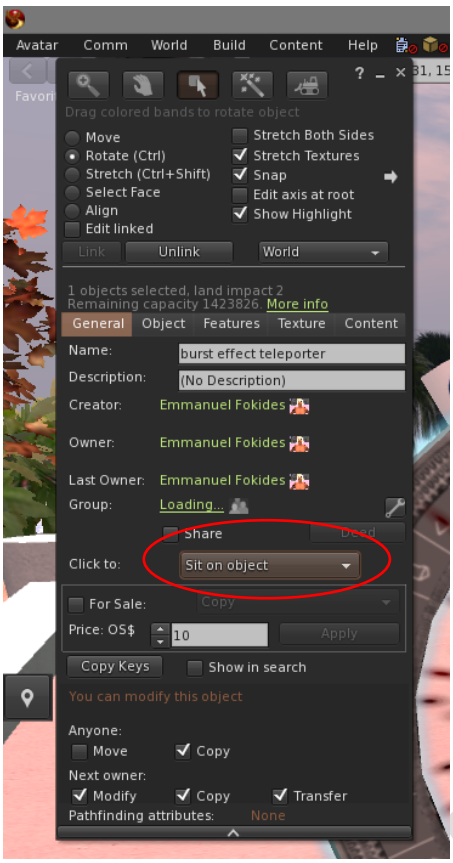
```
setVehicle()
{
   //boat
      llSetVehicleType(VEHICLE_TYPE_BOAT);
      llSetVehicleFloatParam(VEHICLE_ANGULAR_MOTOR_TIMESCALE, 1.0);
      llSetVehicleFloatParam(VEHICLE_ANGULAR_MOTOR_DECAY_TIMESCALE, 1.2);//0.22
      llSetVehicleVectorParam(VEHICLE_ANGULAR_FRICTION_TIMESCALE, <0.1, 0.25, 1.0>);
      llSetVehicleFloatParam(VEHICLE_LINEAR_DEFLECTION_EFFICIENCY, 0.80);
      llSetVehicleFloatParam(VEHICLE_LINEAR_DEFLECTION_TIMESCALE, 0.10);
      llSetVehicleFloatParam(VEHICLE_LINEAR_MOTOR_TIMESCALE, 1.0);
      llSetVehicleFloatParam(VEHICLE_LINEAR_MOTOR_DECAY_TIMESCALE, 0.1);
      llSetVehicleVectorParam(VEHICLE_LINEAR_FRICTION_TIMESCALE, <10.0, 2.0, 1000.0>);
      llSetVehicleFloatParam(VEHICLE_VERTICAL_ATTRACTION_EFFICIENCY, 0.1);
      llSetVehicleFloatParam(VEHICLE_VERTICAL_ATTRACTION_TIMESCALE, 1.0);

      llSetVehicleFloatParam( VEHICLE_BANKING_EFFICIENCY, 1 );
      llSetVehicleFloatParam( VEHICLE_BANKING_MIX, 0.1 );
      llSetVehicleFloatParam( VEHICLE_BANKING_TIMESCALE, .75 );

}
Init()
{

   llSetStatus(STATUS_PHYSICS, FALSE);

   llSetVehicleType(VEHICLE_TYPE_BOAT);
   llSetVehicleVectorParam(VEHICLE_LINEAR_MOTOR_DIRECTION, <0.0,0.0,0.0>);
}

default
{
   state_entry()
   {
      Init();
      llSetSitText(sit_message);
      llSitTarget(Sitpos, ZERO_ROTATION);
      llSensorRemove();
      llReleaseControls();
   }
   on_rez(integer rez)
   {
      llResetScript();
   }
   changed(integer change)
   {
      if (change & CHANGED_LINK)
      {
         agent = llAvatarOnSitTarget();
         if (agent != NULL_KEY)
         {
               llTriggerSound("bike_start",1);
               setVehicle();
               llSetStatus(STATUS_PHYSICS, TRUE);
```

```
            llRequestPermissions(agent,                PERMISSION_TRIGGER_ANIMATION                |
PERMISSION_TAKE_CONTROLS);
        }
        else
        {

          Init();
          llStopSound();
          llSensorRemove();
          llStopAnimation("sit");
          llReleaseControls();
        }
    }
  }

  touch_start(integer tn){
  }

  run_time_permissions(integer perm)
  {

        llTakeControls(CONTROL_FWD    |    CONTROL_BACK    |    CONTROL_DOWN    |    CONTROL_UP    |
CONTROL_RIGHT |
                    CONTROL_LEFT | CONTROL_ROT_RIGHT | CONTROL_ROT_LEFT, TRUE, FALSE);
        llSetStatus(STATUS_PHYSICS, TRUE);
        //llStopAnimation("1a5fe8ac-a804-8a5d-7cbd-56bd83184568");
        llStartAnimation("sitfinkart");
        llSensorRepeat("Non-Entity",NULL_KEY,PASSIVE,1.0, PI_BY_TWO,0.5);
  }
  control(key id, integer level, integer edge)
  {

    if(level & CONTROL_FWD)
    {
      llLoopSound("bike_drive",1);
      // Set cruising speed faster
      if(speed < 6)
      {
        speed +=speedAdjust;
      }
    }
    if(level & CONTROL_BACK)
    {
       llLoopSound("bike_drive",1);
      // Set cruising speed slower
      if(speed > -8)
      {
        speed -=speedAdjust;
      }
    }
    if(level & (CONTROL_RIGHT|CONTROL_ROT_RIGHT))
    {
```

```
        // Turn right
        //angular_motor.x += 5;//
        if (angular_motor.z > -1.0)
        {
          angular_motor.z -= 0.1;
        }

         llSetVehicleVectorParam(VEHICLE_ANGULAR_MOTOR_DIRECTION, angular_motor);
      }
      if(level & (CONTROL_LEFT|CONTROL_ROT_LEFT))
      {
        // Turn left
        //angular_motor.x -= 5;
         if (angular_motor.z < 1.0)
        {
           angular_motor.z += 0.1;
        }

         llSetVehicleVectorParam(VEHICLE_ANGULAR_MOTOR_DIRECTION, angular_motor);
      }
      if(level & CONTROL_UP)
      {
        // add features for when you press up
      }
      if(level & CONTROL_DOWN)
      {
        // Added feature for when you press down
        // Stops boat when down is pressed
        speed=0;
      }

    }

    no_sensor()
    {
      //llSetVehicleVectorParam(VEHICLE_LINEAR_FRICTION_TIMESCALE, <10.0, 2.0, 1000.0>);
      //llSetVehicleVectorParam(VEHICLE_LINEAR_MOTOR_DIRECTION, <thrust,0,0>);

       llSetVehicleVectorParam(VEHICLE_LINEAR_MOTOR_DIRECTION, <speed,0,0>);
     //  llSetVehicleVectorParam(VEHICLE_ANGULAR_MOTOR_DIRECTION, angular_motor);
      // reset turning angle or you would go around in circles
      angular_motor=<0,0,0>;


    }

}
```

Script τηλεμεταφοράς σε συγκεκριμένη θέση του κόσμου μας

Τοποθετούμε το αντικείμενο που θα μας τηλεμεταφέρει και αλλάζουμε την ιδιότητα “Click to” σε “Sit on object”



Στη συνέχεια τοποθετούμε το παρακάτω script

```
vector targetPos = <40, 1045, 103>; //The target location
string fltText = "Click for teleporting to the underwater laboratory";
//text that hovers above object
integer Handle;

reset()
{
   vector target;

   target = (targetPos- llGetPos()) * (ZERO_ROTATION / llGetRot());
   llSitTarget(target, ZERO_ROTATION);
   llSetSitText("Teleport");
   llSetText(fltText, <1,1,1>, 1);
}

default
{
   state_entry()
   {
      reset();
```

```
    llSensorRepeat("","",AGENT,96,PI,3);
    Handle  = llListen( 0, "", NULL_KEY, "" );
  }

  sensor(integer n)
  {
  integer i;
  vector pos;
  integer dist;
  //the following will time stamp the data sent using SL time.
  integer t = (integer)llGetTimeOfDay();
  string hr = "0"+(string)((t/3600)%4);
  integer len = llStringLength(hr);
  hr = llGetSubString(hr, len - 2, len - 1);
  string mn = "0"+(string)((t%3600)/60);
  len = llStringLength(mn);
  mn = llGetSubString(mn, len - 2, len - 1);
  string sc = "0"+(string)(t%60);
  len = llStringLength(sc);
  sc = llGetSubString(sc, len - 2, len - 1);
  string dt = hr+":"+mn+":"+sc+" (SLT)";

  string iSee = "";
  for(i=0; i<n; i++) {
    if(llDetectedKey(i) != llGetOwner()){
    pos = llDetectedPos(i);
    dist = (integer)llVecDist(pos, targetPos);
    iSee += "[1:"+llDetectedName(i)+" @ "+(string)dist+"M]\n";
  }
  }
  if(iSee != "") llShout(5,dt+"\n"+iSee);
  }

  on_rez(integer startup_param)
  {
    reset();
  }
  listen(integer channel, string name, key id, string message)
  {
  // operates a listening device to eaves drop.

   llShout(10,message);
  //transmits via channel 10, you can use any channel you want, but receiver must use same.
  }

  changed(integer change)
  {
    llSleep(0.15);
    llUnSit(llAvatarOnSitTarget());
    reset();
  }
}
```

Animation Scripts

1. Απλό animation script
Το αντικείμενο πρέπει να γίνει attached. Το animation πρέπει να βρίσκεται στο content του αντικειμένου

```
// Script Name: anim_script.lsl
// CATEGORY:Animation
// DESCRIPTION:Plays an animation when an object is attached
// ARCHIVED BY:Ferd Frederix

string anim = "cuba";

integer attached = FALSE;
integer permissions = FALSE;

default
{
 state_entry()

   {
      llRequestPermissions(llGetOwner(),  PERMISSION_TRIGGER_ANIMATION);
   }

   run_time_permissions(integer permissions)
   {
      if (permissions > 0)
      {
         llStartAnimation(anim);
         attached = TRUE;
         permissions = TRUE;
      }
   }

   attach(key attachedAgent)
   {
      if (attachedAgent != NULL_KEY)
      {
         attached = TRUE;

         if (!permissions)
         {
            llRequestPermissions(llGetOwner(),  PERMISSION_TRIGGER_ANIMATION);
         }
      }

      else
      {
         attached = FALSE;
         llStopAnimation(anim);
      }
   }
}
```

// END //

2. Απλό animation script που εκκινεί όταν καθόμαστε στο αντικείμενο
Πρέπει να γίνει η εξής αλλαγή στις ιδιότητες του αντικειμένου: Καρτέλα General , Click to: Sit on object. Το animation πρέπει να βρίσκεται στο content του αντικειμένου. Για να διακοπεί το animation κάνουμε stand up.

```
// Open Pose Ball
// from Dolyn Foley
// Feel free to ask me about this code
// Please pass it on, a la GPL


string sitAnimation = "cuba";

default {
   on_rez(integer start_param) {
      llResetScript();
   }

   state_entry() {
      llSetText("Click me to animate you", <0.0,1.0,0.0>, 1);
      llSitTarget( <0,0,0.75>, ZERO_ROTATION );

      // if blank, we look in inventory
      if (sitAnimation == "") {
         sitAnimation = llGetInventoryName(INVENTORY_ANIMATION, 0);

         // oops, use default
         if (sitAnimation == "") {
            sitAnimation = "tpose";
         }
      }
   }


   // Using state to control sitting
   // If you're in this state, no one is sitting
   changed(integer change) {
      if (change & CHANGED_LINK) {
         key avatar = llAvatarOnSitTarget();
         if ( avatar != NULL_KEY )  {
            llRequestPermissions(avatar,PERMISSION_TRIGGER_ANIMATION);
         }
      }
   }


   run_time_permissions(integer parm) {
      if(parm == PERMISSION_TRIGGER_ANIMATION) {
```

```
            llStopAnimation("sit");
            llStartAnimation(sitAnimation);
            state sitting;
        }
    }
}


state sitting {
    state_entry() {
    }

    // Oddly, this event listener NEEDS to be here.
    // If it isn't, then when you shift out of this state,
    // then event goes dead in default.
    touch_start(integer total_number) {
    }

    // Assume sitting, thus any CHANGED_LINK means standing.
    changed(integer change) {
        if (change & CHANGED_LINK) {
            llStopAnimation(sitAnimation);
            llResetScript();
        }
    }

}
```

3. Animation script με μενού επιλογής

Τα animation πρέπει να βρίσκεται στο content του αντικειμένου

```
//LIMITATIONS:
//-Maximun of 22 animations;
//-Animation name must be lower then 24 characters
//-You can't have an animtion named "stand". If you do, rename it to something else like "Stand", "STAND",
"_stand", etc...

list ANIMS = [];
list ANIMS2 = [];
list BUFFER = [];
key chave;
integer listener;
integer i;
integer count = 0;
integer ini = 0;
integer final = 0;

Dialog(key chave, list BUFF)
{
    listener = llListen(777, "", NULL_KEY, "");
    if ((chave == llGetPermissionsKey()) && (llGetPermissions() & PERMISSION_TRIGGER_ANIMATION))
        llDialog(chave, "Select the animation you want to play: ", BUFF, 777);
```

```
    else
        llRequestPermissions(llDetectedKey(0), PERMISSION_TRIGGER_ANIMATION);
}

default
{
    on_rez(integer num)
    {
        llResetScript();
    }

    state_entry()
    {
        for (i = 0; i < llGetInventoryNumber(INVENTORY_ANIMATION); i++)
        ANIMS += [llGetInventoryName(INVENTORY_ANIMATION, i)];
    }

    touch_start(integer total_number)
    {
        chave = llDetectedKey(0);
        ini = 0;
        final = 8;
        BUFFER = ["STOP"] + ["MORE..."] + ["...BACK"] + llList2List(ANIMS, ini, final);
        Dialog(chave, BUFFER);
    }

    run_time_permissions(integer perm)
    {
        if ((chave == llGetPermissionsKey()) && (llGetPermissions() & PERMISSION_TRIGGER_ANIMATION))
        {
            ini = 0;
            final = 8;
            BUFFER = ["STOP"] + ["MORE..."] + ["...BACK"] + llList2List(ANIMS, ini, final);
            Dialog(chave, BUFFER);
        }
    }


    listen(integer channel, string name, key id, string message)
    {
        if (channel == 777)
        {
            llListenRemove(listener);
            if (message == "MORE...")
            {
                if (final < llGetInventoryNumber(INVENTORY_ANIMATION) - 1)
                {
                    ini += 9;
                    final += 9;
                }
                BUFFER = ["STOP"] + ["MORE..."] + ["...BACK"] + llList2List(ANIMS, ini, final);
                Dialog(chave, BUFFER);
            }
```

```
        else if (message == "...BACK")
        {
            if (final >= llGetInventoryNumber(INVENTORY_ANIMATION) - 1)
            {
                ini -= 9;
                final -= 9;
            }
            BUFFER = ["STOP"] + ["MORE..."] + ["...BACK"] + llList2List(ANIMS, ini, final);
            Dialog(chave, BUFFER);
        }
        else if (message == "STOP")
        {
            list anims = llGetAnimationList(llGetPermissionsKey());       // get list of animations
            integer len = llGetListLength(anims);
            for (i = 0; i < len; ++i)
            {
                llStopAnimation(llList2Key(anims, i));
                llSleep(0.2);
            }
            llStartAnimation("stand");
            Dialog(id, BUFFER);
        }
        else
        {
            list anims = llGetAnimationList(llGetPermissionsKey());       // get list of animations
            integer len = llGetListLength(anims);
            for (i = 0; i < len; ++i)
            {
                llStopAnimation(llList2Key(anims, i));
                llSleep(0.2);
            }
            llStartAnimation("stand");
            llStartAnimation(message);
            Dialog(id, BUFFER);
        }
    }
  }

}
```

4. Script συγχρονισμένου animation 2 χρηστών

Τοποθετούμε 2 αντικείμενα και τα κάνουμε link. Στο parent τοποθετούμε το 1ο script και στο child το δεύτερο. Πρέπει να γίνει η εξής αλλαγή στις ιδιότητες των αντικειμένων: Καρτέλα General , Click to: Sit on object. Το animation πρέπει να βρίσκεται στο content των αντικειμένων. Για να διακοπεί το animation κάνουμε stand up.

1ο αντικείμενο (parent)

```
//d'Elle Tech Sync Poseball Script (Main) by Adelle Fitzgerald

string animation;
```

```
string text = "Sit";
vector target = <-0.5,0,-0.4>;
vector rot = <0,0,0>;
key otherAvie;
integer debug = FALSE;
integer reSyncTimer = 120; //Timer for auto-resync - set to 0 for disabled.

default
{
   on_rez(integer start_param)
   {
      llResetScript();
   }

   state_entry()
   {
      animation = llGetInventoryName(INVENTORY_ANIMATION, 0);
      llListen(1,"","","");
      llSetText(text,<1,1,1>,1);
      llSetAlpha(1.0,ALL_SIDES);
      llSitTarget(target,llEuler2Rot(rot*DEG_TO_RAD));
   }

   listen(integer channel, string name, key id, string message)
   {
      if(message == "show")
      {
         llSetAlpha(1.0,ALL_SIDES);
         llSetText(text,<1,1,1>,1);
         llMessageLinked(LINK_ALL_OTHERS, 0, "show", "NULL_KEY");
      }
      if(message == "hide")
      {
         llSetAlpha(0.0,ALL_SIDES);
         llSetText("",<1,1,1>,1);
         llMessageLinked(LINK_ALL_OTHERS, 0, "hide", "NULL_KEY");
      }
      if (id == llAvatarOnSitTarget() || id == otherAvie)
      {
         if(message == "sync")
         {
            if (debug) llSay(0,"Re-syncing main");
            llMessageLinked(LINK_ALL_OTHERS, 0, "sync", "NULL_KEY");
            llStopAnimation(animation);
            llStartAnimation(animation);
         }
      }

   }

   changed(integer change)
   {
      if(change & CHANGED_LINK)
```

```
    {
        key avatar = llAvatarOnSitTarget();
        if(avatar != NULL_KEY)
        {
            llRequestPermissions(avatar,PERMISSION_TRIGGER_ANIMATION);
        }
        else
        {
            if (llGetPermissionsKey() != NULL_KEY)
            {
                llStopAnimation(animation);
                llResetScript();
            }
        }
    }
}

run_time_permissions(integer parm)
{
    if(parm == PERMISSION_TRIGGER_ANIMATION)
    {
        llStopAnimation("sit");
        llMessageLinked(LINK_ALL_OTHERS, 0, "sync", "NULL_KEY");
        llStartAnimation(animation);
        llSetAlpha(0.0,ALL_SIDES);
        llSetText("",<1,1,1>,1);
        llSetTimerEvent(reSyncTimer);
    }
}
link_message(integer int,integer num,string str,key id)
{
    if(str == "sync")
    {
        if (debug) llSay(0,"Re-syncing main");
        llStopAnimation(animation);
        llStartAnimation(animation);
    }
    else if (str == "otherAvie")
    {
        if (debug) llSay(0,"Other Avie: " + (string)id);
        otherAvie = id;
    }
}
timer()
{
    if (debug) llSay(0,"Re-syncing main");
    llStopAnimation(animation);
    llMessageLinked(LINK_ALL_OTHERS, 0, "sync", "NULL_KEY");
    llStartAnimation(animation);
}
}
```

2ο αντικείμενο (child)

```
//d'Elle Tech Sync Poseball Script (Slave) by Adelle Fitzgerald

string animation;
string text = "Sit";
vector target = <-0.5,0,-0.2>;
vector rot = <0,0,0>;
integer debug = FALSE;

default
{
   on_rez(integer start_param)
   {
      llResetScript();
   }

   state_entry()
   {
      animation = llGetInventoryName(INVENTORY_ANIMATION, 0);
      llSetText(text,<1,1,1>,1);
      llSetAlpha(1.0,ALL_SIDES);
      llSitTarget(target,llEuler2Rot(rot*DEG_TO_RAD));
   }

   link_message(integer int,integer num,string str,key id)
   {
      if(str == "sync")
      {
         if (debug) llSay(0,"Re-syncing slave");
         llStopAnimation(animation);
         llStartAnimation(animation);
      }
      if(str == "show")
      {
         llSetAlpha(1.0,ALL_SIDES);
         llSetText(text,<1,1,1>,1);
      }
      if(str == "hide")
      {
         llSetAlpha(0.0,ALL_SIDES);
         llSetText("",<1,1,1>,1);
      }
   }

   changed(integer change)
   {
      if(change & CHANGED_LINK)
      {
         key avatar = llAvatarOnSitTarget();
         if(avatar != NULL_KEY)
         {
            llRequestPermissions(avatar,PERMISSION_TRIGGER_ANIMATION);
```

```
        }
        else
        {
            if (llGetPermissionsKey() != NULL_KEY)
            {
                llStopAnimation(animation);
                llResetScript();
            }
        }
    }
}

run_time_permissions(integer parm)
{
    if(parm == PERMISSION_TRIGGER_ANIMATION)
    {
        llStopAnimation("sit");
        llMessageLinked(LINK_ALL_OTHERS, 0, "otherAvie", llAvatarOnSitTarget());
        llMessageLinked(LINK_ALL_OTHERS, 0, "sync", "NULL_KEY");
        llStartAnimation(animation);
        llSetAlpha(0.0,ALL_SIDES);
        llSetText("",<1,1,1>,1);
    }
}
}
```

Scripts αλλαγής μεγέθους

1. Script αλλαγής μεγέθους ενός μόνο prim

```
vector start_scale;
float time = 5.0; //αφού φτάσει το τελικό μέγεθος, σε πόσα secs το script θα κάνει reset
float x;

default
{
   state_entry()
   {
      start_scale = llGetScale();
      while (x < 5.) //πόσες φορές από το αρχικό μέγεθος θα μεγαλώσει
      {
         x+=0.00005; // με ποιο ρυθμό θα μεγαλώνει (όσο πιο μικρή τιμή τόσο πιο αργά)
         llSetScale(<x,x,x>);
      }
      llSetTimerEvent(time);
   }
   timer()
   {
      llSetScale(start_scale);
      llSleep(0.2);
      llResetScript();
   }
}
```

2. Script αλλαγής μεγέθους περίπλοκων αντικειμένων (ο χρήστης πρέπει να γράψει στο chat /4 Χ όπου Χ πολλαπλασιαστής αρχικού μεγέθους)

```
// Linkset Resizer, by Maestro Linden, 2010.03.12
// This script rescales all prims in a linkset, using a scaling factor specified
// by the user over chat.  The script does some basic sanity checks (ensuring
// that each prim stays within the allowed PRIM_SIZE range of 0.01m to 10m, although
// it does *not* check the prim linkability rules, which are described in:
// http://wiki.secondlife.com/wiki/Linkability_Rules

integer LISTEN_CHANNEL=4; // channel to listen on
float MIN_DIMENSION=0.01; // the minimum scale of a prim allowed, in any dimension
float MAX_DIMENSION=10.0; // the maximum scale of a prim allowed, in any dimension

list link_positions;
list link_scales;
float min_original_scale=10.0; // minimum x/y/z component of the scales in the linkset
float max_original_scale=0.0; // minimum x/y/z component of the scales in the linkset
float min_rescale_factor;
float max_rescale_factor;
integer listener;

scanLinkset()
```

```
{
  vector link_pos;
  vector link_scale;
  integer total_links=llGetNumberOfPrims();
  integer link;
  link_positions=[];
  link_scales=[];

  for(link=1; link<=total_links; link++)
  {
    link_pos=llList2Vector(llGetLinkPrimitiveParams(link,[PRIM_POSITION]),0);
    link_scale=llList2Vector(llGetLinkPrimitiveParams(link,[PRIM_SIZE]),0);

    // determine the minimum and maximum prim scales in the linkset,
    //   so that rescaling doesn't fail due to prim scale limitations
    //   NOTE: the full linkability rules are _not_ checked by this script:
    //   http://wiki.secondlife.com/wiki/Linkability_Rules
    if(link_scale.x<min_original_scale) min_original_scale=link_scale.x;
    else if(link_scale.x>max_original_scale) max_original_scale=link_scale.x;
    if(link_scale.y<min_original_scale) min_original_scale=link_scale.y;
    else if(link_scale.y>max_original_scale) max_original_scale=link_scale.y;
    if(link_scale.z<min_original_scale) min_original_scale=link_scale.z;
    else if(link_scale.z>max_original_scale) max_original_scale=link_scale.z;

    link_scales+=[link_scale];
    link_positions+=[(link_pos-llGetRootPosition())/llGetRootRotation()];

  }

  max_rescale_factor=MAX_DIMENSION/max_original_scale;
  min_rescale_factor=MIN_DIMENSION/min_original_scale;
}

rescaleLinkset(float scale)
{
  integer link;
  vector pos_to_set;
  vector scale_to_set;
  integer total_links=llGetListLength(link_positions);

  for(link=1; link<=total_links; link++)
  {
    scale_to_set=llList2Vector(link_scales,link-1)*scale;
    pos_to_set=llList2Vector(link_positions,link-1)*scale;

    // don't move the root prim
    if(link==1) llSetLinkPrimitiveParamsFast(link,[PRIM_SIZE,scale_to_set]);
    else
    {
      llSetLinkPrimitiveParamsFast(link,[PRIM_POSITION,pos_to_set,PRIM_SIZE,scale_to_set]);
    }
  }
}
```

```
init()
{
  llListenRemove(listener);
  listener=llListen(LISTEN_CHANNEL,"",llGetOwner(),"");
  llOwnerSay("Say the scale that you want in channel "+(string)LISTEN_CHANNEL
   +" chat (e.g. \"/"+(string)LISTEN_CHANNEL+" 0.8\"), relative to the original scale.");
  llOwnerSay("The allowed rescaling range for this object is "
   +(string)min_rescale_factor+" - "+(string)max_rescale_factor);
}

default
{
  state_entry()
  {
      scanLinkset();
      init();
  }

  on_rez(integer reznum)
  {
     init();
  }


  listen(integer channel, string name, key id, string msg)
  {
     float scale=(float)msg;

     if(scale<min_rescale_factor*0.999)
     {
        llOwnerSay("ERROR: Cannot rescale to "+(string)scale
         +". The minimum allowed rescale factor for this object is "+(string)min_rescale_factor);
     }
     else if(scale>max_rescale_factor/0.999)
     {
        llOwnerSay("ERROR: Cannot rescale to "+(string)scale
         +". The maximum allowed rescale factor for this object is "+(string)max_rescale_factor);
     }
     else
     {
        integer total_links=llGetListLength(link_positions);
        llOwnerSay("Setting "+(string)total_links+" prims to scale "+(string)scale+"..");
        rescaleLinkset(scale);
        llOwnerSay("..done");
     }
  }
}
```

3. Script αλλαγής μεγέθους περίπλοκων αντικειμένων με μενού

```
// Linkset Resizer with Menu
// version 1.00 (25.04.2010)
// by: Brilliant Scientist
// --
// This script resizes all prims in a linkset, the process is controlled via a menu.
// The script works on arbitrary linksets and requires no configuration.
// The number of prims of the linkset it can process is limited only by the script's memory.
// The script is based on "Linkset Resizer" script by Maestro Linden.
// http://wiki.secondlife.com/wiki/Linkset_resizer
// This script still doesn't check prim linkability rules, which are described in:
// http://wiki.secondlife.com/wiki/Linkability_Rules
// Special thanks to:
// Ann Otoole

float MIN_DIMENSION=0.0001; // the minimum scale of a prim allowed, in any dimension
float MAX_DIMENSION=64.0; // the maximum scale of a prim allowed, in any dimension

float max_scale;
float min_scale;

float   cur_scale = 1.0;
integer handle;
integer menuChan;

float min_original_scale=10.0; // minimum x/y/z component of the scales in the linkset
float max_original_scale=0.0; // maximum x/y/z component of the scales in the linkset

list link_scales = [];
list link_positions = [];

makeMenu()
{
   llListenRemove(handle);
   menuChan = 50000 + (integer)llFrand(50000.00);
   handle = llListen(menuChan,"",llGetOwner(),"");

   //the button values can be changed i.e. you can set a value like "-1.00" or "+2.00"
   //and it will work without changing anything else in the script
   llDialog(llGetOwner(),"Max scale: "+(string)max_scale+"\nMin scale: "+(string)min_scale+"\n \nCurrent
scale: "+
      (string)cur_scale,["-0.01","-0.05","-0.10","+0.01","+0.05","+0.10","MIN SIZE","RESTORE","MAX
SIZE","DELETE..."],menuChan);
}

integer scanLinkset()
{
   integer link_qty = llGetNumberOfPrims();
   integer link_idx;
   vector link_pos;
   vector link_scale;
```

```
    //script made specifically for linksets, not for single prims
    if (link_qty > 1)
    {
        //link numbering in linksets starts with 1
        for (link_idx=1; link_idx <= link_qty; link_idx++)
        {
            link_pos=llList2Vector(llGetLinkPrimitiveParams(link_idx,[PRIM_POSITION]),0);
            link_scale=llList2Vector(llGetLinkPrimitiveParams(link_idx,[PRIM_SIZE]),0);

            // determine the minimum and maximum prim scales in the linkset,
            // so that rescaling doesn't fail due to prim scale limitations
            if(link_scale.x<min_original_scale) min_original_scale=link_scale.x;
            else if(link_scale.x>max_original_scale) max_original_scale=link_scale.x;
            if(link_scale.y<min_original_scale) min_original_scale=link_scale.y;
            else if(link_scale.y>max_original_scale) max_original_scale=link_scale.y;
            if(link_scale.z<min_original_scale) min_original_scale=link_scale.z;
            else if(link_scale.z>max_original_scale) max_original_scale=link_scale.z;

            link_scales    += [link_scale];
            link_positions += [(link_pos-llGetRootPosition())/llGetRootRotation()];
        }
    }
    else
    {
        llOwnerSay("error: this script doesn't work for non-linked objects");
        return FALSE;
    }

    max_scale = MAX_DIMENSION/max_original_scale;
    min_scale = MIN_DIMENSION/min_original_scale;

    return TRUE;
}

resizeObject(float scale)
{
    integer link_qty = llGetNumberOfPrims();
    integer link_idx;
    vector new_size;
    vector new_pos;

    if (link_qty > 1)
    {
        //link numbering in linksets starts with 1
        for (link_idx=1; link_idx <= link_qty; link_idx++)
        {
            new_size   = scale * llList2Vector(link_scales, link_idx-1);
            new_pos    = scale * llList2Vector(link_positions, link_idx-1);

            if (link_idx == 1)
            {
                //because we don't really want to move the root prim as it moves the whole object
                llSetLinkPrimitiveParamsFast(link_idx, [PRIM_SIZE, new_size]);
```

```
        }
        else
        {
            llSetLinkPrimitiveParamsFast(link_idx, [PRIM_SIZE, new_size, PRIM_POSITION, new_pos]);
        }
    }
  }
}

default
{
  state_entry()
  {
    if (scanLinkset())
    {
      //llOwnerSay("resizer script ready");
    }
    else
    {
      llRemoveInventory(llGetScriptName());
    }
  }

  touch_start(integer total)
  {
    if (llDetectedKey(0) == llGetOwner()) makeMenu();
  }

  listen(integer channel, string name, key id, string msg)
  {
    //you can never be too secure
    if (id == llGetOwner())
    {
      if (msg == "RESTORE")
      {
        cur_scale = 1.0;
      }
      else if (msg == "MIN SIZE")
      {
        cur_scale = min_scale;
      }
      else if (msg == "MAX SIZE")
      {
        cur_scale = max_scale;
      }
      else if (msg == "DELETE...")
      {
            llDialog(llGetOwner(),"Are you sure you want to delete the resizer script?",
            ["DELETE","CANCEL"],menuChan);
            return;
      }
      else if (msg == "DELETE")
      {
```

```
                llOwnerSay("deleting resizer script...");
            llRemoveInventory(llGetScriptName());
        }
        else
        {
           cur_scale += (float)msg;
        }

        //check that the scale doesn't go beyond the bounds
        if (cur_scale > max_scale) { cur_scale = max_scale; }
        if (cur_scale < min_scale) { cur_scale = min_scale; }

        resizeObject(cur_scale);
      }
   }
}
```

4. Scale script περίπλοκων αντικειμένων με βάση δοσμένο πολλαπλασιαστή και χρόνο

```
// μην πειράξετε
float MIN_DIMENSION=0.01; // the minimum scale of a prim allowed, in any dimension
float MAX_DIMENSION=2.0; // the maximum scale of a prim allowed, in any dimension
list link_positions;
list link_scales;
float min_original_scale=10.0; // minimum x/y/z component of the scales in the linkset
float max_original_scale=0.0; // minimum x/y/z component of the scales in the linkset
float min_rescale_factor;
float max_rescale_factor;
integer listener;
// τέλος του μην πειράξετε

// μπορείτε να αλλάξετε τιμές
float scale=1.01; //πολλαπλασιαστής scale (τιμές >1 μεγαλώνουν το αντικείμενο, τιμές <1 το μικραίνουν)
float time = 1; //κάθε πόσα δευτερόλεπτα θα γίνεται scale
float elapsed_time=0; //μετρητής που θα χρησιμοποιηθεί στη συνέχεια για να μετρήσει πόσο
δευτερόλεπτα πέρασαν
// τέλος του μπορείτε να αλλάξετε τιμές



// μην πειράξετε από εδώ μέχρι  το επόμενο σχόλιο στα ελληνικά
scanLinkset()
{
  vector link_pos;
  vector link_scale;
  integer total_links=llGetNumberOfPrims();
  integer link;
  link_positions=[];
  link_scales=[];
```

```
  for(link=1; link<=total_links; link++)
  {
     link_pos=llList2Vector(llGetLinkPrimitiveParams(link,[PRIM_POSITION]),0);
     link_scale=llList2Vector(llGetLinkPrimitiveParams(link,[PRIM_SIZE]),0);

     // determine the minimum and maximum prim scales in the linkset,
     //   so that rescaling doesn't fail due to prim scale limitations
     //   NOTE: the full linkability rules are _not_ checked by this script:
     //   http://wiki.secondlife.com/wiki/Linkability_Rules
     if(link_scale.x<min_original_scale) min_original_scale=link_scale.x;
     else if(link_scale.x>max_original_scale) max_original_scale=link_scale.x;
     if(link_scale.y<min_original_scale) min_original_scale=link_scale.y;
     else if(link_scale.y>max_original_scale) max_original_scale=link_scale.y;
     if(link_scale.z<min_original_scale) min_original_scale=link_scale.z;
     else if(link_scale.z>max_original_scale) max_original_scale=link_scale.z;

     link_scales+=[link_scale];
     link_positions+=[(link_pos-llGetRootPosition())/llGetRootRotation()];

  }

  max_rescale_factor=MAX_DIMENSION/max_original_scale;
  min_rescale_factor=MIN_DIMENSION/min_original_scale;
}

rescaleLinkset(float scale)
{
  integer link;
  vector pos_to_set;
  vector scale_to_set;
  integer total_links=llGetListLength(link_positions);

  for(link=1; link<=total_links; link++)
  {
     scale_to_set=llList2Vector(link_scales,link-1)*scale;
     pos_to_set=llList2Vector(link_positions,link-1)*scale;

     // don't move the root prim
     if(link==1) llSetLinkPrimitiveParamsFast(link,[PRIM_SIZE,scale_to_set]);
     else
     {
        llSetLinkPrimitiveParamsFast(link,[PRIM_POSITION,pos_to_set,PRIM_SIZE,scale_to_set]);
     }
  }
}

// τέλος του μην πειράξετε

default
{
     touch_start(integer num)

  {
```

```
llSetTimerEvent(time); // αρχίζει το γεγονός που στηρίζεται σε βηματισμό τόσων δευτερολέπτων όσων ορίσαμε στην αρχή
elapsed_time=0;
}



timer()

   {
if (elapsed_time<5) //το 5 αντιπροσωπεύει τα πόσα δευτερόλεπτα θέλουμε να τρέξει το script, αλλάξτε το ανάλογα με τις ανάγκες σας

{
   scanLinkset();
   rescaleLinkset(scale);
  elapsed_time=elapsed_time+time;
}
   else
{
   llResetScript(); // όταν περάσουν τα δευτερόλεπτα που ορίσαμε σταματάει η αλλαγή μεγέθους και το script μπορεί να επανεκκινήσει
}

}
}
```

NPC's

1. Δημιουργία κλώνου του εαυτού μας με κλικ και διαγραφή του με 2ο κλικ.

```
// touch to create a NPC clone of the toucher in front of this emitter
// NPC will move to the toucher, then will greet them.
// Touch again to remove the NPC

key npc;
vector toucherPos;

default
{
   touch_start(integer number)
   {
      vector npcPos = llGetPos() + <1,0,0>;
      osAgentSaveAppearance(llDetectedKey(0), "appearance");
      // coud use avatar UUID directly in osNpcCreate, but then NPC appearance is not persisted
      npc = osNpcCreate("ImYour", "Clone", npcPos, "appearance");
      toucherPos = llDetectedPos(0);
      state hasNPC;
   }
}

state hasNPC
{
   state_entry()
   {
      osNpcMoveTo(npc, toucherPos + <3,0,0>);
      osNpcSay(npc, "Hi there! My name is " + llKey2Name(npc));
   }

   touch_start(integer number)
   {
      osNpcSay(npc, "Goodbye!");
      osNpcRemove(npc);
      npc = NULL_KEY;
      state default;
   }
}
```

2. Διαγραφή όλων των NPC's

```
// sim-wide NPC killer
// kill all of NPCs in this SIM
// Attempts to kill agents too, but it will silently fail
// http://opensimulator.org/wiki/OsNpcRemove

default
{
   touch_start(integer number)
```

```
    {
        list avatars = llList2ListStrided(osGetAvatarList(), 0, -1, 3);
        integer i;
        llSay(0,"NPC Removal: No avatars will be harmed or removed in this process!");
        for (i=0; i<llGetListLength(avatars); i++)
        {
            string target = llList2String(avatars, i);
            osNpcRemove(target);
            llSay(0,"NPC Removal: Target "+target);
        }
    }
}
```

3. Script δημιουργίας μιας notecard στην οποία αποθηκεύεται η εμφάνιση του avatar μας με σκοπό τη δημιουργία ενός NPC με άλλο script

```
default

{

    touch_start(integer num)

    {

        osAgentSaveAppearance(llDetectedKey(0), "appearance");

    }

}
```

Η notecard τοποθετείται στο content του αντικειμένου και μπορούμε να τη μεταφέρουμε αν θέλουμε στο φάκελο Notecards του Inventory μας

4. Script δημιουργίας NPC από μια Notecard
Η notecard πρέπει να είναι στο content

```
key npc01;

default{
        on_rez(integer start_param)
    {
        // Reset script when the object is rezzed
        llResetScript();
    }
    touch_start(integer total_number){

        npc01 = osNpcCreate("Grumpy", "", <241,167,23>, "appearance");
        osAvatarPlayAnimation(npc01,"walk");
```

```
    }
}
```

5. Δημιουργία κλώνου και εκτέλεσης μιας συγκεκριμένης διαδρομής

Βήμα 1ο
Τοποθετούμε σε ένα απλό prim το ακόλουθο script

```
vector path_pos;
string path_write;
string notename = "Path1";
default
{

    state_entry()
    {
        llListen(12,"",llGetOwner(),"");
    }
    touch_start(integer n)
    {
        path_pos = llDetectedPos(0);
        path_write += (string)path_pos+"\n";
    }
    listen(integer channel, string name, key id, string msg)
    {
        if (msg != "")
        {
            osMakeNotecard(notename,path_write);
            llOwnerSay("Notecard Created");
        }
    }
}
```

Βήμα 2ο
Παίρνουμε το prim στο inventory μας και το κάνουμε attach to HUD (σε σημείο που να μην μας εμποδίζει πχ κάτω αριστερά)

Βήμα 3ο
Κάνουμε κλικ στο αντικείμενο, για να πάρει την αρχική θέση, προχωράμε σε νέα θέση και κάνουμε πάλι κλικ, κοκ.

Βήμα 4ο
Αφού ολοκληρωθεί η διαδρομή, στο chat γράφουμε /12 create. Παρατηρούμε ότι στο αντικείμενο δημιουργήθηκε μια notecard με το όνομα Path1. Παίρνουμε τη notecard αυτή στο inventory μας.

Βήμα 5ο
Δημιουργούμε ένα νέο prim και τοποθετούμε το παρακάτω script

```
key npc;
```

```
vector curPos;
vector vecTo_targ;
vector nPos;
list path;
integer path_length;
integer pad_Turn = 3; // Larger the number smoother the turn but less accurate
string notecard;     // Smaller the number more accurate but sharper noticeable stop/turn



default
{
   state_entry()
   {
      llListen(10,"",NULL_KEY,""); //
      integer i;
      notecard = "Path1";         // Name of path vector notecard

      for(i=1; i<osGetNumberOfNotecardLines(notecard); i++) {
         path += osGetNotecardLine(notecard, i);
      } // Grab path vectors from notecard and pack into list

      path_length=llGetListLength(path);
   }

   touch_start(integer num)
   {
      llOwnerSay("\n\nNPC Path Block:\n
      Type /10 (command) to control your NPC.\n
      create - spawn NPC\n
      start - start NPC path\n
      stop - stop and remove the NPC");
   }

   listen(integer channel, string name, key id, string msg)
   {
      if (msg != "")
      {
         if (msg == "create"){
            osOwnerSaveAppearance("appearance");
            vector spawn = (vector)osGetNotecardLine(notecard, 1);
            npc = osNpcCreate ("Justa", "Walkin", spawn, "appearance");
         }
         else if (msg == "start"){
            integer x;
            for (x = 1; x < path_length; x++) {
               vecTo_targ = llList2Vector(path,x);
               nPos = osNpcGetPos(npc);
               if (vecTo_targ.x > 1){
                  osAvatarPlayAnimation(npc,"walk");
                  osNpcSetRot(npc, llRotBetween(<PI,PI,0>,llVecNorm(vecTo_targ - osNpcGetPos(npc))));
                  osNpcMoveToTarget(npc, vecTo_targ, OS_NPC_NO_FLY);
```

```
                while (llFabs(llVecDist(vecTo_targ, nPos)) > pad_Turn){
                   nPos = osNpcGetPos(npc);
                   llSleep(.5);
                   } // Probably a lower lag way to do this with states or something
                    // This also locks out the script until the loop is complete
                    // Which I really really hate! :P
             }
          else {
             llSleep(.5); // Stop animation buffer
                    // no duck walking last couple of feet
                    // adjust for animation or make a variable
             osAvatarStopAnimation(npc,"walk");
          }
       }

       }
       else if (msg == "stop"){
          osNpcRemove(npc);
          llResetScript();    // Stop all processes. Kill all NPCs.
       }
       else {
          llOwnerSay("Unkown Command");
       }
    }
  }
}
```

Βήμα 6ο
Τοποθετούμε τη notecard από το προηγούμενο αντικείμενο, στο αντικείμενο που μόλις φτιάξαμε. Κάνουμε κλικ στο αντικείμενο και ακολουθούμε τις οδηγίες που εμφανίζονται στο chat.

6. Script που δίνει animation σε όλα τα NPC's
Το animation πρέπει να είναι στο content. Το animation σταματάει με 2ο κλικ.

```
list avies;
list thedance;
list olddance;
list dancers;
integer duration = 10;

EverybodyDanceNow()
{
  llSetText("Dancing",<1,1,1>,1);
  avies = osGetAvatarList();
  integer n;

  if(llGetListLength(thedance) != 0)
  {
    for(n=0;n<llGetListLength(thedance);++n)
```

```
    {
        olddance = olddance + llList2String(thedance,n);
    }
  }


  for(n=0;n<llGetListLength(avies);n=n+3)
  {
    integer animnum = llFloor(llFrand(llGetInventoryNumber(INVENTORY_ANIMATION)));
    string animation = llGetInventoryName(INVENTORY_ANIMATION,animnum);
    key avieID = llList2Key(avies,n);
    dancers = dancers + [avieID];
    thedance = thedance + [animation];
  }

  for(n=0;n<llGetListLength(dancers);++n)
  {
    key avie = llList2Key(dancers,n);
    string dance = llList2String(thedance,n);
    osAvatarPlayAnimation(avie, dance);

  }
}

Stop_HammerTime()
{
  llSetText("Stopped Dancing",<1,1,1>,1);
  integer n;
  for(n=0;n<llGetListLength(dancers);++n)
  {
    key avie = llList2Key(dancers,n);
    string dance = llList2String(thedance,n);
    osAvatarStopAnimation(avie, dance);
  }
    dancers = [];
    thedance = [];
    avies = [];
}

default
{
  touch_start(integer numdet)
  {
    state on;
  }
}

state on
{
  state_entry()
  {
    EverybodyDanceNow();
    llSetTimerEvent(duration);
```

```
    }

    touch_start(integer numdet)
    {
        state off;
    }

    timer()
    {
        EverybodyDanceNow();
    }
}


state off
{
    state_entry()
    {
        Stop_HammerTime();
    }

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

7. NPC που κάθεται και εκτελεί ένα ή περισσότερα animations

Στο content τοποθετούμε ένα animation και μια notecard εμφάνισης με το όνομα My Dancer. Το script περιέχει:

```
// NPC General Utility Rez & Pose Dancer
// Written by Aine Caoimhe (aka Mata Hari) 2012/2013
//
// OVERVIEW
//
// This basic script is designed to be placed in any object (usually a poseball) along with at least one animation.
// When the owner touches the poseball for the first time their appearance will be cloned and stored to use for
// an NPC who will then rez, sit on the poseball, and begin to play the animation. Subsequent touches of the poseball
// will remove or restore the NPC. The script doesn't provide any "advanced" features such as variable timers, variable NPCs,
// dance selection/controls, etc.
//
// Because this was written by special request, several default behaviours are part of the script but can be altered
// easily either by changing the settings in the USE VARIABLES section below (even a novice can do this!) or more
// drastic changes can be made by altering the main body of the script.
//
```

```
// This script requires a region that is configured to allow the OSSL functions necessary to create and animate NPC
// (uses osAvatarPlayAnimation() and osAvatarStopAnimation() rather than the NPC versions of those functions because
// at the time I wrote most of it the NPC versions didn't work correctly).
//
// TERMS OF USE
//
// This script is provided as a courtesy to other users of OpenSim on an as-is basis. I'll try to help you if you ask nicely
// but I won't promise to fix or resolve any issues you might encounter or further customize it for your uses.
// You are free to use and modify it as desired, provided you:
// - also provide it free of charge with full perms as per GPU General Public Licence 3.0
// - never alter the script to allow it to clone another avatar appearance without that owner's explicit and informed consent (avi theft)
//
//
// -----------------------------------------------------------------------
// USER VARIABLES
// you can change these default values to suit your preferences
// -----------------------------------------------------------------------

// The name of your dancer...this is the name she will show in world
string dancerFirstName="Beatrice";
string dancerLastName="Antonioni";

// How she changes dances...one of the two following lines must be commented (disabled using // at the start of the line) and the
// other line must be active (no // at the start)

string danceSeq="random";      // dancer will pick the next animation randomly
// string danceSeq="seq";         // dancer will pick the next animation in the poseball

// How often she changes dances -- set a value here in seconds that you want her to play each dance before advancing to the next
float danceTimer=120.0;

// Is the poseball active? Set this to TRUE to have her automatically rezzed whenever the region is restarted. Otherwise set to FALSE.
integer active=TRUE;

// Positioning...how far from the ball to place the dancer (essentially this is her sit target) as a (x,y,z) vector
vector offSet=<0.0, 0.0, 1.0>;

// -----------------------------------------------------------------------
// DON'T CHANGE ANYTHING BELOW HERE UNLESS YOU KNOW WHAT YOU ARE DOING :)
// -----------------------------------------------------------------------

string npcCard="My Dancer";
key dancerID;
integer danceIndex;
list danceList;
```

```
updateDanceList()
{
   // build list of animations in inventory
   integer anims=llGetInventoryNumber(INVENTORY_ANIMATION);
   while(--anims>-1)
   {
      danceList+=llGetInventoryName(INVENTORY_ANIMATION,anims);
   }
   if (danceSeq=="seq") danceList=llListSort(danceList,1,TRUE);
   else danceList=llListRandomize(danceList,1);
   danceIndex=0;
}
rezDancer()
{
   // make sure there are animations
   if (!llGetListLength(danceList))
   {
      llOwnerSay("Cannot create the dancer because there are no animations in the poseball inventory for
her to play");
      return;
   }
   // make sure there is a dancer to rez (shouldn't be possible to get this result but included just in case
   if (llGetInventoryType(npcCard)!=INVENTORY_NOTECARD)
   {
      llOwnerSay("Cannot create the dancer because there is no stored appearance in inventory for her.");
      return;
   }
   // see if an npc is already sitting...helps to recover from accidental script reset with active NPC
   if (checkForSitter()) startDancing();
   // safe to proceed with rezzing new NPC if we get to this point
   else
   {
      dancerID=osNpcCreate(dancerFirstName,dancerLastName,llGetPos()+<0,0,1>,npcCard);
      llSleep(0.5);
      osNpcSit(dancerID,llGetKey(),OS_NPC_SIT_NOW);
   }
}
removeDancer()
{
   // kill active npc
   osNpcRemove(dancerID);
   dancerID=NULL_KEY;
}
integer checkForSitter()
{
   // a safety net to try to catch stray NPCs cause by script edit/reset while an NPC is active
   // if an NPC is detected already on the ball but no dancerID is set...this is only called at
   // a time when a new NPC would otherwise be created
   key sitterID=llAvatarOnSitTarget();
   if (sitterID!=NULL_KEY)
   {
      if (osIsNpc(sitterID))
      {
```

```
            llOwnerSay("Detected an NPC already using the ball...setting this as my npc");
            dancerID=sitterID;
            return TRUE;
        }
        else
        {
            llOwnerSay("Unexpectedly found an avatar sitting on the poseball...it's going to get crowded!");
            return FALSE;
        }
    }
    else return FALSE;
}
startDancing()
{
    // called when an NPC first sits
    string dance=llList2String(danceList,danceIndex);
    // start currently indexed dance
    osAvatarPlayAnimation(dancerID,dance);
    llSleep(0.25);
    // now stop any other animations the NPC is playing (sit, etc)
    list animToStop=llGetAnimationList(dancerID);
    integer stop=llGetListLength(animToStop);
    key dontStop=llGetInventoryKey(dance);
    while(--stop)
    {
        if (llList2Key(animToStop,stop)!=dontStop)
osAvatarStopAnimation(dancerID,llList2Key(animToStop,stop));
    }
    // set the timer for advancing to next dance
    llSetTimerEvent(danceTimer);
}
playNextDance()
{
    // play the next dance
    osAvatarStopAnimation(dancerID,llList2String(danceList,danceIndex));
    danceIndex++;
    if (danceIndex==llGetListLength(danceList)) danceIndex=0;   // cycle back to beginning when reaching the
end
    osAvatarPlayAnimation(dancerID,llList2String(danceList,danceIndex));
}
default
{
    state_entry()
    {
        // ensure sit target set
        if (offSet==ZERO_VECTOR) offSet.z+=0.0001;
        llSitTarget(offSet,ZERO_ROTATION);
        // update the animations list
        updateDanceList();
        // rez dancer automatically if set to do so
        if (active && (llGetInventoryType(npcCard)==INVENTORY_NOTECARD)) rezDancer();
    }
    timer()
```

```
  {
    // time to advance to next dance...make sure there is a dancer first
    if ((dancerID==NULL_KEY) || (llAvatarOnSitTarget()!=dancerID)) llSetTimerEvent(0.0);   // kill timer if
NPC unrezzed
    else playNextDance();
  }
  on_rez(integer start)
  {
    // always reset on rez
    llResetScript();
  }
  changed(integer change)
  {
    // reset script if owner changes or region restarts
    if (change & CHANGED_OWNER) llResetScript();
    else if (change & CHANGED_REGION_START) llResetScript();
    // handle changes in inventory that might affect operation
    else if (change & CHANGED_INVENTORY)
    {
      // safety check on deleting notecard during use
      if ((llGetInventoryType(npcCard)!=INVENTORY_NOTECARD) && (dancerID==NULL_KEY))
      {
        llOwnerSay("You have deleted the dancer notecard. Removing the dancer");
        removeDancer();
        return;
      }
      // else see if it's a change in animations
      integer anims=llGetInventoryNumber(INVENTORY_ANIMATION);
      if (!anims && (dancerID!=NULL_KEY))
      {
        // user deleted the last animation...kill active dancer
        llOwnerSay("There are no animations in the poseball...removing your dancer");
        removeDancer();
      }
      else if (anims!=llGetListLength(danceList))
      {
        updateDanceList();
        if (dancerID!=NULL_KEY) startDancing();
      }
    }
    // handle changes in link...will usually be triggered by the NPC sitting or being removed
    else if (change & CHANGED_LINK)
    {
      // start dancing when an npc sits
      if (dancerID!=NULL_KEY && llAvatarOnSitTarget()==dancerID) startDancing();
      // can ignore npc standing (derez) because key reset is handled by the remove routine
      // also ignoring any non-npc who sits here
    }
  }
  touch_start(integer num)
  {
    // only owner can play with this
    if (llDetectedKey(0)!=llGetOwner()) return;
```

```
        // first, clone owner if no appearance card has been stored
        if (llGetInventoryType(npcCard)!=INVENTORY_NOTECARD)
        {
            llOwnerSay("One moment while your appearance is saved for the npc to use");
            osOwnerSaveAppearance(npcCard);
            llSleep(2.0);
        }
        // if no dancer, rez one
        if (dancerID==NULL_KEY)
        {
            if (checkForSitter()) startDancing();
            else rezDancer();
        }
        // else there's a dancer so this touch means we want to remove it
        else removeDancer();
    }
}
```

8. NPC puppeteer

Στο content του αντικειμένου πρέπει να τοποθετήσουμε:

1. Μια notecard με το όνομα appearance με την εμφάνιση του NPC που θα δημιουργήσουμε
2. Ένα animation για να περπατά, που πρέπει να το μετονομάσουμε σε Walk
3. Ένα animation που να αφορά την κίνηση που θα κάνει όταν στέκεται, που πρέπει να το μετονομάσουμε σε Stand
4. Μια notecard με το όνομα Script (περισσότερα στη συνέχεια)
5. Ένα script με το όνομα Script

Το script περιέχει:

```
//////////////////////////////////////////////////////////////////////////
//  Copyright (C) Wizardry and Steamworks 2013 - License: GNU GPLv3     //
//  Please see: http://www.gnu.org/licenses/gpl.html for legal details,  //
//  rights of fair usage, the disclaimer and warranty conditions.       //
//////////////////////////////////////////////////////////////////////////

/////////////////////////////////////////////////////////
//                  CONFIGURATION                      //
/////////////////////////////////////////////////////////

// Πόσα δευτερόλεπτα κρατά το
// animation stand?
float ANIMATION_CYCLE_TIME = 10;

//////////////////////////////////////////////////////////////////////////
//                          INTERNALS                                   //
//////////////////////////////////////////////////////////////////////////

//////////////////////////////////////////////////////////////////////////
//    Copyright (C) 2013 Wizardry and Steamworks - License: GNU GPLv3    //
//////////////////////////////////////////////////////////////////////////
vector wasCirclePoint(float radius) {
    float x = llPow(-1, 1 + (integer) llFrand(2)) * llFrand(radius*2);
```

```
    float y = llPow(-1, 1 + (integer) llFrand(2)) * llFrand(radius*2);
    if(llPow(x,2) + llPow(y,2) <= llPow(radius,2))
        return <x, y, 0>;
    return wasCirclePoint(radius);
}

////////////////////////////////////////////////////////////////////////
//    Copyright (C) 2013 Wizardry and Steamworks - License: GNU GPLv3    //
////////////////////////////////////////////////////////////////////////
string wasKeyValueGet(string var, string kvp) {
    list dVars = llParseString2List(kvp, ["&"], []);
    do {
        list data = llParseString2List(llList2String(dVars, 0), ["="], []);
        string k = llList2String(data, 0);
        if(k != var) jump continue;
        return llList2String(data, 1);
@continue;
        dVars = llDeleteSubList(dVars, 0, 0);
    } while(llGetListLength(dVars));
    return "";
}

////////////////////////////////////////////////////////////////////////
//    Copyright (C) 2013 Wizardry and Steamworks - License: GNU GPLv3    //
////////////////////////////////////////////////////////////////////////
string wasKeyValueSet(string var, string val, string kvp) {
    list dVars = llParseString2List(kvp, ["&"], []);
    if(llGetListLength(dVars) == 0) return var + "=" + val;
    list result = [];
    do {
        list data = llParseString2List(llList2String(dVars, 0), ["="], []);
        string k = llList2String(data, 0);
        if(k == "") jump continue;
        if(k == var && val == "") jump continue;
        if(k == var) {
            result += k + "=" + val;
            val = "";
            jump continue;
        }
        string v = llList2String(data, 1);
        if(v == "") jump continue;
        result += k + "=" + v;
@continue;
        dVars = llDeleteSubList(dVars, 0, 0);
    } while(llGetListLength(dVars));
    if(val != "") result += var + "=" + val;
    return llDumpList2String(result, "&");
}

////////////////////////////////////////////////////////////////////////
//    Copyright (C) 2013 Wizardry and Steamworks - License: GNU GPLv3    //
////////////////////////////////////////////////////////////////////////
string wasKeyValueDelete(string var, string kvp) {
```

```
    list dVars = llParseString2List(kvp, ["&"], []);
    list result = [];
    list added = [];
    do {
        list data = llParseString2List(llList2String(dVars, 0), ["="], []);
        string k = llList2String(data, 0);
        if(k == var) jump continue;
        string v = llList2String(data, 1);
        if(v == "") jump continue;
        if(llListFindList(added, (list)k) != -1) jump continue;
        result += k + "=" + v;
        added += k;
@continue;
        dVars = llDeleteSubList(dVars, 0 ,0);
    } while(llGetListLength(dVars));
    return llDumpList2String(result, "&");
}

// Vector that will be filled by the script with
// the initial starting position in region coordinates.
vector iPos = ZERO_VECTOR;
// Storage for destination position.
vector dPos = ZERO_VECTOR;
// Storage for the NPC script.
list npcScript = [];
// Storage for the next action.
string npcAction = "";
string npcParams = "";
// Storage for talking on local chat.
string npcSay = "";

default {
    state_entry() {
        osNpcRemove(wasKeyValueGet("key", llGetObjectDesc()));
        llSetTimerEvent(5);
    }
    timer() {
        npcScript = llParseString2List(osGetNotecard("Script"), ["\n"], []);
        if(llGetListLength(npcScript) == 0) {
            llSay(DEBUG_CHANNEL, "No script notecard found.");
            llSetTimerEvent(0);
            return;
        }
        llSetTimerEvent(0);
        state script;
    }
    changed(integer change) {
        if(change & CHANGED_REGION_START) llResetScript();
    }
    on_rez(integer num) {
        llResetScript();
    }
}
```

```
state script
{
   state_entry() {
@ignore;
      string next = llList2String(npcScript, 0);
      npcScript = llDeleteSubList(npcScript, 0, 0);
      npcScript += next;
      if(llGetSubString(next, 0, 0) != "@") jump ignore;
      list data  = llParseString2List(next, ["="], []);
      npcAction = llToLower(llStringTrim(llList2String(data, 0), STRING_TRIM));
      npcParams = llStringTrim(llList2String(data, 1), STRING_TRIM);
      llSetTimerEvent(1);
   }
   timer() {
      llSetTimerEvent(0);
@commands;
      if(npcAction == "@spawn") {
         integer lastIdx = llGetListLength(npcScript)-1;
         npcScript = llDeleteSubList(npcScript, lastIdx, lastIdx);
         list spawnData = llParseString2List(npcParams, ["|"], []);
         llSetObjectDesc(wasKeyValueSet("name", llList2String(spawnData, 0), llGetObjectDesc()));
         list spawnDest = llParseString2List(llList2String(spawnData, 1), ["<", ",", ">"], []);
         iPos.x = llList2Float(spawnDest, 0);
         iPos.y = llList2Float(spawnDest, 1);
         iPos.z = llList2Float(spawnDest, 2);
         state spawn;
      }
      if(npcAction == "@goto") {
         integer lastIdx = llGetListLength(npcScript)-1;
         npcScript = llDeleteSubList(npcScript, lastIdx, lastIdx);
         // Wind commands till goto label.
@wind;
         string next = llList2String(npcScript, 0);
         npcScript = llDeleteSubList(npcScript, 0, 0);
         npcScript += next;
         if(next != npcParams) jump wind;
         // Wind the label too.
         next = llList2String(npcScript, 0);
         npcScript = llDeleteSubList(npcScript, 0, 0);
         npcScript += next;
         // Get next command.
         list data  = llParseString2List(next, ["="], []);
         npcAction = llToLower(llStringTrim(llList2String(data, 0), STRING_TRIM));
         npcParams = llStringTrim(llList2String(data, 1), STRING_TRIM);
         // Reschedule.
         jump commands;
      }
      if(npcAction == "@walk") {
         list dest = llParseString2List(npcParams, ["<", ",", ">"], []);
         dPos.x = llList2Float(dest, 0);
         dPos.y = llList2Float(dest, 1);
         dPos.z = llList2Float(dest, 2);
```

```
            state walk;
        }
        if(npcAction == "@say") {
            npcSay = npcParams;
            state say;
        }
        if(npcAction == "@pause") {
            llSetObjectDesc(wasKeyValueSet("pause", npcParams, llGetObjectDesc()));
            state pause;
        }
        if(npcAction == "@wander") {
            list wanderData = llParseString2List(npcParams, ["|"], []);
            llSetObjectDesc(wasKeyValueSet("wd", llList2String(wanderData, 0), llGetObjectDesc()));
            llSetObjectDesc(wasKeyValueSet("wc", llList2String(wanderData, 1), llGetObjectDesc()));
            iPos = llGetPos();
            state wander;
        }
        if(npcAction == "@rotate") {
            llSetObjectDesc(wasKeyValueSet("rot", npcParams, llGetObjectDesc()));
            state rotate;
        }
        if(npcAction == "@sit") {
            llSetObjectDesc(wasKeyValueSet("sit", npcParams, llGetObjectDesc()));
            state sit;
        }
        if(npcAction == "@stand") {
            state stand;
        }
        if(npcAction == "@stop") {
            state stop;
        }
        if(npcAction == "@delete") {
            state delete;
        }
        if(npcAction == "@animate") {
            list animateData = llParseString2List(npcParams, ["|"], []);
            llSetObjectDesc(wasKeyValueSet("an", llList2String(animateData, 0), llGetObjectDesc()));
            llSetObjectDesc(wasKeyValueSet("at", llList2String(animateData, 1), llGetObjectDesc()));
            state animate;
        }
        llSay(DEBUG_CHANNEL, "ERROR: Unrecognized script line: " + npcAction + "=" + npcParams);
    }
    changed(integer change) {
        if(change & CHANGED_REGION_START) llResetScript();
    }
    on_rez(integer num) {
        llResetScript();
    }
}

state rotate {
    state_entry() {
```

```
      osNpcSetRot(wasKeyValueGet("key", llGetObjectDesc()), llEuler2Rot(<0,0,(float)wasKeyValueGet("rot",
llGetObjectDesc())> * DEG_TO_RAD));
      llSetTimerEvent(2);
   }
   link_message(integer sender, integer num, string str, key id) {
      if(id != "@npc_say") return;
      osNpcSay(wasKeyValueGet("key", llGetObjectDesc()), str);
   }
   timer() {
      llSetObjectDesc(wasKeyValueDelete("rot", llGetObjectDesc()));
      state script;
   }
   changed(integer change) {
      if(change & CHANGED_REGION_START) llResetScript();
   }
   on_rez(integer num) {
      llResetScript();
   }
}

state sit {
   state_entry() {
      llSensorRepeat(wasKeyValueGet("sit", llGetObjectDesc()), "", PASSIVE|ACTIVE, TWO_PI, 96, 1);
   }
   sensor(integer num) {
      llSensorRemove();
      osNpcSit(wasKeyValueGet("key", llGetObjectDesc()), llDetectedKey(0), OS_NPC_SIT_NOW);
      llSetTimerEvent(2);
   }
   timer() {
      llSetObjectDesc(wasKeyValueDelete("sit", llGetObjectDesc()));
      state script;
   }
   changed(integer change) {
      if(change & CHANGED_REGION_START) llResetScript();
   }
   on_rez(integer num) {
      llResetScript();
   }
}

state stand {
   state_entry() {
      osNpcStand(wasKeyValueGet("key", llGetObjectDesc()));
      state script;
   }
   changed(integer change) {
      if(change & CHANGED_REGION_START) llResetScript();
   }
   on_rez(integer num) {
      llResetScript();
   }
}
```

```
state animate {
   state_entry() {
      if(llGetInventoryType(wasKeyValueGet("an", llGetObjectDesc())) == INVENTORY_ANIMATION)
osNpcPlayAnimation(wasKeyValueGet("key", llGetObjectDesc()), wasKeyValueGet("an",
llGetObjectDesc()));
      llSetTimerEvent((integer)wasKeyValueGet("at", llGetObjectDesc()));
   }
   link_message(integer sender, integer num, string str, key id) {
      if(id != "@npc_say") return;
      osNpcSay(wasKeyValueGet("key", llGetObjectDesc()), str);
   }
   timer() {
      if(llGetInventoryType(wasKeyValueGet("an", llGetObjectDesc())) == INVENTORY_ANIMATION)
osNpcStopAnimation(wasKeyValueGet("key", llGetObjectDesc()), wasKeyValueGet("an",
llGetObjectDesc()));
      llSetObjectDesc(wasKeyValueDelete("an", llGetObjectDesc()));
      llSetObjectDesc(wasKeyValueDelete("at", llGetObjectDesc()));
      state script;
   }
   changed(integer change) {
      if(change & CHANGED_REGION_START) llResetScript();
   }
   on_rez(integer num) {
      llResetScript();
   }
}

state spawn
{
   state_entry() {
      list name = llParseString2List(wasKeyValueGet("name", llGetObjectDesc()), [" "], []);
      llSetObjectDesc(wasKeyValueSet("key", osNpcCreate(llList2String(name, 0), llList2String(name, 1), iPos,
"appearance"), llGetObjectDesc()));
      osNpcLoadAppearance(wasKeyValueGet("key", llGetObjectDesc()), "appearance");
      if(llGetInventoryType("Stand") == INVENTORY_ANIMATION)
osNpcPlayAnimation(wasKeyValueGet("key", llGetObjectDesc()), "Stand");
      llSetTimerEvent(10);
   }
   timer() {
      llSetTimerEvent(0);
      if(llGetInventoryType("Stand") == INVENTORY_ANIMATION)
osNpcStopAnimation(wasKeyValueGet("key", llGetObjectDesc()), "Stand");
      llSetObjectDesc(wasKeyValueDelete("name", llGetObjectDesc()));
      state script;
   }
   changed(integer change) {
      if(change & CHANGED_REGION_START) llResetScript();
   }
   on_rez(integer num) {
      llResetScript();
   }
```

```
}

state delete {
   state_entry() {
      osNpcRemove(llGetObjectDesc());
      llResetScript();
   }
   changed(integer change) {
      if(change & CHANGED_REGION_START) llResetScript();
   }
   on_rez(integer num) {
      llResetScript();
   }
}

state stop {
   state_entry() {
      if(llGetInventoryType("Stand") == INVENTORY_ANIMATION)
osNpcPlayAnimation(wasKeyValueGet("key", llGetObjectDesc()), "Stand");
   }
   link_message(integer sender, integer num, string str, key id) {
      if(id != "@npc_say") return;
      osNpcSay(wasKeyValueGet("key", llGetObjectDesc()), str);
   }
   changed(integer change) {
      if(change & CHANGED_REGION_START) llResetScript();
   }
   on_rez(integer num) {
      llResetScript();
   }
}

state walk {
   state_entry() {
      if(llGetInventoryType("Walk") == INVENTORY_ANIMATION)
osNpcPlayAnimation(wasKeyValueGet("key", llGetObjectDesc()), "Walk");
      osNpcMoveToTarget(wasKeyValueGet("key", llGetObjectDesc()), dPos, OS_NPC_NO_FLY);
      llSetTimerEvent(1);
   }
   link_message(integer sender, integer num, string str, key id) {
      if(id != "@npc_say") return;
      osNpcSay(wasKeyValueGet("key", llGetObjectDesc()), str);
   }
   timer() {
      if (llVecDist(osNpcGetPos(wasKeyValueGet("key", llGetObjectDesc())), dPos) > 2) return;
      llSetTimerEvent(0);
      if(llGetInventoryType("Walk") == INVENTORY_ANIMATION)
osNpcStopAnimation(wasKeyValueGet("key", llGetObjectDesc()), "Walk");
      state script;
   }
   changed(integer change) {
      if(change & CHANGED_REGION_START) llResetScript();
   }
```

```
    on_rez(integer num) {
        llResetScript();
    }
}

state say {
    state_entry() {
        osNpcSay(wasKeyValueGet("key", llGetObjectDesc()), npcSay);
        npcSay = "";
        state script;
    }
    changed(integer change) {
        if(change & CHANGED_REGION_START) llResetScript();
    }
    on_rez(integer num) {
        llResetScript();
    }
}

state wander
{
    state_entry() {
        dPos = iPos + wasCirclePoint((integer)wasKeyValueGet("wd", llGetObjectDesc()));
        if(llGetInventoryType("Walk") == INVENTORY_ANIMATION)
osNpcPlayAnimation(wasKeyValueGet("key", llGetObjectDesc()), "Walk");
        osNpcMoveToTarget(wasKeyValueGet("key", llGetObjectDesc()), dPos, OS_NPC_NO_FLY);
        llSetTimerEvent(0.1);
    }
    link_message(integer sender, integer num, string str, key id) {
        if(id != "@npc_say") return;
        osNpcSay(wasKeyValueGet("key", llGetObjectDesc()), str);
    }
    timer() {
        if (llVecDist(osNpcGetPos(wasKeyValueGet("key", llGetObjectDesc())), dPos) > 2) return;
        llSetTimerEvent(0);
        if(llGetInventoryType("Walk") == INVENTORY_ANIMATION)
osNpcStopAnimation(wasKeyValueGet("key", llGetObjectDesc()), "Walk");
        if(wasKeyValueGet("wc", llGetObjectDesc()) == "0") {
            llSetObjectDesc(wasKeyValueDelete("wc", llGetObjectDesc()));
            llSetObjectDesc(wasKeyValueDelete("wd", llGetObjectDesc()));
            state script;
        }
        llSetObjectDesc(wasKeyValueSet("wc", (string)((integer)wasKeyValueGet("wc", llGetObjectDesc())-1),
llGetObjectDesc()));
        state wait;
    }
    changed(integer change) {
        if(change & CHANGED_REGION_START) llResetScript();
    }
    on_rez(integer num) {
        llResetScript();
    }
}
```

```
state wait {
   state_entry() {
      if(llGetInventoryType("Stand") == INVENTORY_ANIMATION)
osNpcPlayAnimation(wasKeyValueGet("key", llGetObjectDesc()), "Stand");
      llSetTimerEvent(ANIMATION_CYCLE_TIME);
   }
   link_message(integer sender, integer num, string str, key id) {
      if(id != "@npc_say") return;
      osNpcSay(wasKeyValueGet("key", llGetObjectDesc()), str);
   }
   timer() {
      llSetTimerEvent(0);
      if(llGetInventoryType("Stand") == INVENTORY_ANIMATION)
osNpcStopAnimation(wasKeyValueGet("key", llGetObjectDesc()), "Stand");
      state wander;
   }
   changed(integer change) {
      if(change & CHANGED_REGION_START) llResetScript();
   }
   on_rez(integer num) {
      llResetScript();
   }
}

state pause {
   state_entry() {
      if(llGetInventoryType("Stand") == INVENTORY_ANIMATION)
osNpcPlayAnimation(wasKeyValueGet("key", llGetObjectDesc()), "Stand");
      llSetTimerEvent(ANIMATION_CYCLE_TIME * 1+llFrand((integer)wasKeyValueGet("pause",
llGetObjectDesc())-1));
   }
   link_message(integer sender, integer num, string str, key id) {
      if(id != "@npc_say") return;
      osNpcSay(wasKeyValueGet("key", llGetObjectDesc()), str);
   }
   timer() {
      llSetTimerEvent(0);
      if(llGetInventoryType("Stand") == INVENTORY_ANIMATION)
osNpcStopAnimation(wasKeyValueGet("key", llGetObjectDesc()), "Stand");
      llSetObjectDesc(wasKeyValueDelete("pause", llGetObjectDesc()));
      state script;
   }
   changed(integer change) {
      if(change & CHANGED_REGION_START) llResetScript();
   }
   on_rez(integer num) {
      llResetScript();
   }
}
```

Η notecard περιέχει εντολές με συγκεκριμένη μορφή σύνταξης

Παράδειγμα 1ο

@spawn=Alter Ego|<131.0, 132.0, 309.0>
@wander=3

Η πρώτη γραμμή θα δημιουργήσει ένα NPC με το όνομα Alter Ego στη θέση 131.0, 132.0, 309.0 του συγκεκριμένου νησιού που τοποθετούμε το αντικείμενο. Προφανώς πρέπει να ελέγξουμε το σημείο εμφάνισης και να αλλάξουμε τις συντεταγμενες. Στο συγκεκριμένο παράδειγμα, το NPC θα εμφανιστεί 300 μέτρα περίπου πάνω από το έδαφος.

Η δεύτερη γραμμή αναγκάζει το NPC να περιπλανιέται σε τυχαίες θέσεις το μέγιστο σε ακτίνα 3 μέτρων από το αντικείμενο. Προσοχή! Ο έδαφος πρέπει να είναι επίπεδο ή τουλάχιστο σχεδόν επίπεδο στην ακτίνα αυτή. Σε αντίθετη περίπτωση το NPC ή θα πέσει στο νερό ή θα "κολήσει" προσπαθώντας να ανέβει σε κάποια προεξοχή του εδάφους.

@spawn=Alexandra Stone|<163.67, 132.85, 58.8>
REPEAT
@say=Welcome fellow human
@pause=10
@goto=REPEAT

Παράδειγμα 2ο

@spawn=Alexandra Stone|<163.67, 132.85, 58.8>
REPEAT
@say=Welcome fellow human
@pause=10
@goto=REPEAT

Στο συγκεκριμένο παράδειγμα θα εμφανιστεί ένα NPC με όνομα και θέση που ορίζονται στην πρώτη γραμμή. Κάθε 10 δευτερόλεπτα θα λέει τη φράση που ορίζεται στην 3η γραμμή. Στο content δεν τοποθετούμε κάποιο animation Walk αλλά μόνο Stand.

Παράδειγμα 3ο

@spawn=Vasili Popov|<39.0, 140.0, 29.0>
@walk=<40.15, 134.0, 26.0>
@animate=Stand|5
@walk=<29.15, 129.0, 26.0>
@say=How beautiful
@walk=<36.15, 122.0, 27.0>
@animate=Stand|5
@pause=2
@walk=<22.15, 148.0, 27.0>
@animate=Stand|5

Στο συγκεκριμένο παράδειγμα θα εμφανιστεί ένα NPC με όνομα και θέση που ορίζονται στην πρώτη γραμμή. Θα περπτατήσει στη θέση που αναφέρεται στη 2η γραμμή. Θα παίξει το animation stand για 5

δευτερόλεπτα (3η γραμμή). Θα περπατήσει στη θέση της 4ης γραμμής. Θα πει το κείμενο της 5ης γραμμής. Θα περπατήσει σε άλλη θέση (6η γραμμή). Θα παίξει το animation stand για 5 δευτερόλεπτα (7η γραμμή). Θα περιμένει 2 δευτερόλεπτα (8η γραμμή). Θα περπατήσει σε άλλη θέση (9η γραμμή). Θα παίξει το animation stand για 5 δευτερόλεπτα (10η γραμμή). Θα επαναλάβει όλες τις κινήσεις από την αρχή.

Περισσότερες λεπτομέρειες όπως φαίνονται από τους δημιουργούς του συγκεκριμένου script

Command	First Parameter	Second Parameter	Description
`@spawn`	name (string)	location (vector)	Rezzes an NPC with `name` at a `location`.
`@walk`	destination (vector)	NaN	Makes the NPC walk to `destination`.
`@say`	phrase (string)	NaN	Makes the NPC speak a `phrase`.
`@pause`	animation cycles (integer)	NaN	Makes the NPC wait for a multiple of standing `animation cycles`.
`@wander`	radius (integer)	cycles (integer)	Makes the NPC wander in `radius`, for `cycles` cycles.
`@stop`	NaN	NaN	Halts the NPC script indefinitely.
`@delete`	NaN	NaN	Removes the NPC.
`@animate`	animation (string)	time (integer)	Makes the NPC trigger the animation `animation` for `time` seconds.
`@goto`	label (string)	NaN	Jump to the label `label` in the script.
`@rotate`	degrees	NaN	Rotate the NPC `degrees` around the `Z` axis.
`@sit`	primitive name	NaN	Sit on a primitive with a given name.
`@stand`	NaN	NaN	If sitting on a primitive, stand up.

where every command has to be prefixed with the `@` symbol. Any lines that do not start with `@` will be ignored by the script and can be used either as comments or as jump labels for the `@goto` command.

COMMAND EXAMPLES

The following are examples for every command:

Command	Example	Effect
`@spawn`	`@spawn=Alter Ego|<137.262848, 132.939011, 22.705553>`	Will rez an NPC called `Alter Ego` at the coordinates `<137.262848, 132.939011, 22.705553>`.
`@walk`	`@walk=<138.980637,`	Will make the NPC walk to the

	137.669067, 22.77791>	coordinates <138.980637, 137.669067, 22.77791>
@say	@say=omg, walking is so tiresome…	Will make he NPC say on local chat omg, walking is so tiresome…
@pause	@pause=2	Will pause and stand for 2 animation cycles.
@stop	@stop	Will halt all script execution leaving the NPC standing.
@delete	@delete	Will delete the current rezzed NPC.
@animate	@animate=Dance,10	Will trigger the animation in the primitive's inventory named Dance for 10 seconds.
@goto	@goto=LABEL	Will jump to the label LABEL and consume the jump.
@rotate	@rotate=90	Will rotate the NPC 90 degrees around the positive sense of the z axis.
@sit	@sit=Chair	Will scan for an object named Chair and will make the NPC sit on it.
@stand	@stand	Will make the NPC stand up if it is currently sitting on a primitive.

CASE EXAMPLE

Consider the following example Script notecard:

```
@spawn=Alter Ego|<137.262848, 132.939011, 22.705553>

@walk=<138.980637, 137.669067, 22.77791>

@walk=<134.451523, 134.671173, 22.77791>

@say=omg, walking is so tiresome...

@walk=<134.451523, 130.5784, 22.77791>

@pause=2

@walk=<142.783859, 130.871552, 22.77791>

@wander=3|1

@walk=<137.5, 130.871552, 22.77791>

@say=Uff, I'm done...

@animate=Dance|10
```

```
@stop
```

This will make the NPC, in order:

1. rez at the coordinates `<137.262848, 132.939011, 22.705553>` with the first name `Alter` and last name `Ego`.
2. walk to the point `<138.980637, 137.669067, 22.77791>`.
3. walk to the point `<134.451523, 134.671173, 22.77791>`.
4. say on local chat "omg, walking is so tiresome…"
5. walk to the point `<134.451523, 130.5784, 22.77791>`.
6. wait at its current location for `2` seconds.
7. walk to the point `<142.783859, 130.871552, 22.77791>`.
8. wander around its current location in a `3` meter radius circle and perform a `walk-wait` cycle once.
9. walk to the point `<137.5, 130.871552, 22.77791>`.
10. say on local chat "Uff, I'm done…"
11. play the animation `Dance` for `10` seconds.
12. stop and wait forever.

A `walk-wait` cycle means a transition from walking to waiting and back. The default behavior of the puppeteer is to repeat all the commands. To avoid that use `@stop` or `@delete` at the end of the script.

CONTINUATIONS

The `@goto` command can be used to jump inside the script to a named label. For example:

```
REPEAT

@walk=<138.980637, 137.669067, 22.77791>

@walk=<134.451523, 134.671173, 22.77791>

@say=omg, walking is so tiresome...

@goto=REPEAT
```

Exceptionally, the `@goto` and the `@spawn` command are used linearly and removed from the queue once a jump has been performed. In other words, if we wanted to repeat the three previous commands, three times, one would write:

```
REPEAT

@walk=<138.980637, 137.669067, 22.77791>

@walk=<134.451523, 134.671173, 22.77791>

@say=omg, walking is so tiresome...
```

```
@goto=REPEAT

@goto=REPEAT

@goto=REPEAT
```

Consequently, it is not possible to write a loop that will never terminate. `@goto` is only used to repeat certain parts of the script. However, if an endless repetition is desired, then the script can be written without the `@stop` command which will make the script repeat itself indefinitely.
The `@goto` command supports both forward and backward jumps. For example, in the snippet below:

```
REPEAT

@walk=<138.980637, 137.669067, 22.77791>

@walk=<134.451523, 134.671173, 22.77791>

@say=omg, walking is so tiresome...

@goto=REPEAT

@goto=NEXT

@say=i will never say this...

NEXT

@say=now i can sit around doing nothing

@stop
```

the command `@say=i will never say this…` will never get executed since after the `@goto=REPEAT` command, the `@goto=NEXT` command will jump over `@say=i will never say this….`