196. Programmare giochi in 3D

Rosa Marincola
rosamarincola@virgilio.it

Premessa

In questo contributo presenterò delle attività realizzate nel mondo virtuale 3D Edmondo con la classe III A Sistemi Informativi Aziendali dell'I.I.S. "A. Guarasci" sez. Tecnico Economico di Rogliano (Cs) nell'ambito della sperimentazione d'informatica sulla land Scriptlandia [1, 2].
Si tratta di alcuni giochi classici di cui sono stati studiati gli algoritmi risolutivi e di cui è stata fatta la codifica in LSL (Linden Scripting Language), il linguaggio di programmazione simile a C, C#, Java, utilizzato nei mondi virtuali.
Il percorso didattico si colloca nell'ambito della Teoria dei Giochi di cui sono stati trattati alcuni elementi durante le ore curricolari d'informatica, con collegamenti interdisciplinari in economia aziendale e matematica.
La moderna teoria dei giochi può essere fatta coincidere con l'uscita del libro "Theory of Games and Economic Behavior" di John von Neumann e Oskar Morgenstern nel 1944 e questa disciplina rappresenta un buon modello per descrivere le interazioni strategiche tra agenti economici.
Molti risultati economici coinvolgono l'interazione strategica come ad esempio l'andamento di mercati non perfettamente competitivi, l'andamento nelle aste, l'andamento nelle negoziazioni economiche. La teoria dei giochi [3] ha applicazioni nel campo strategico-militare, nella politica, nella sociologia, nella psicologia, nell'informatica, nella biologia, nello sport. Lo studio e la costruzione di giochi si rivela un'attività didattica significativa e motivante per gli studenti.
Di seguito saranno presentati un gioco di strategia e due giochi di sorte.

Figura 1:Foto su Scriptlandia con nuove costruzioni tra cui il paraboloide iperbolico.

Il gioco dell'undici

Caratteristiche [4]:
1. Il gioco è condotto da due giocatori che eseguono alternativamente una mossa (la mossa è obbligatoria) e ne conoscono le regole.
2. Il gioco termina con esattamente uno di due possibili risultati: o vince chi gioca per primo, o vince l'avversario, ma esiste una strategia vincente per il giocatore che gioca per primo.
3. Ogni mossa consiste in una scelta da parte del giocatore di una mossa tra un insieme di

mosse possibili.
4. Ad ogni istante del gioco, i giocatori sono informati completamente su tutte le mosse già compiute e su tutte quelle che potranno venir fatte (informazione perfetta).
5. Il numero mosse in una partita è finito.

Regole del gioco:
Ci sono undici oggetti. I due giocatori si alternano nel raccogliere 1, 2 o 3 oggetti finché non restano più oggetti sul tavolo. Il giocatore costretto a raccogliere l'ultimo oggetto perde.

Strategia vincente per il primo giocatore:
1. A raccoglie 2 oggetti.
2. B raccoglie K oggetti (k= 3)
3. A raccoglie 4–K oggetti

Figura 2: Sperimentazione del gioco dell'undici, visibile il nautilus realizzato per l'esposizione dei giochi.

In Edmondo è stato costruito un oggetto (nel nostro caso un ipercubo, ma si può utilizzare una primitiva più semplice come un cubo a forma di pulsante o altro) nel cui contenuto sono stati inseriti:

- una notecard (file di testo) con le regole del gioco in modo che chiunque clicchi sull'oggetto le possa leggere;
- un tetraedro di colore rosso, in modo che ad ogni mossa vengano rezzati (costruiti), cioè appaiano tanti tetraedri quanti sono gli elementi ancora in gioco (di seguito è riportato lo script contenuto nei tetraedri perché si auto-cancellino ai clic successivi);
- lo script del gioco, in cui il PC gioca per primo, utilizza la strategia vincente e batte sempre l'avatar.

Nota: i commenti in LSL si scrivono preceduti da //

```
//Script da inserire nel tetraedro che utilizza il canale 9999 per dialogare con
//l'oggetto che lo rezza
default {
      state_entry()
      {
            llListen(9999,"",NULL_KEY,"DELETE");
      }

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

//script da inserire nell'oggetto che contiene il gioco, vi sono diversi stati
string stato="iniziale";
integer n=11;
integer index;
integer handle;
integer qhandle=0;
integer i;
vector vel=ZERO_VECTOR;
rotation rot=ZERO_ROTATION;

//questa funzione rezza i tetraedri
rezzer (integer n)
{
 for(i=0;i<n;i++)
        {
 //I tetraedri vengono disposti intorno all'oggetto
           vector pos=llGetPos()+<(llFrand(1)-0.5)*3, 1, 1>;
            llRezAtRoot("tetraedro",pos,vel,rot,5555);
        }
        llSleep(5);
        llShout(9999,"DELETE");
        if(qhandle!=0) llListenRemove(qhandle);
        return;}

default
{
    state_entry()
    {
     llSetText("Clicca per giocare \n al gioco dell'11 contro il PC",<1,1,1>,1);
     llShout(9999,"DELETE");
     stato="primo";
   }

touch_start(integer total_number)
{
    llShout(9999,"DELETE");
    if(qhandle!=0) llListenRemove(qhandle);
      key id=llDetectedKey(0);
      if(stato=="primo")
            {
            rezzer (n);
//al clic viene rilasciata la notecard con le regole, contenuta nell'inventario
dell'oggetto
            llGiveInventory(id, llGetInventoryName(INVENTORY_NOTECARD, 0));
```

```
            handle=llListen(-1,"",id,"");
            llSay(0, "il PC fa la prima mossa");
            n=n-2;
      llSay(0, "PC: ho tolto 2, ci sono " +(string)n +" elementi, clicca per
continuare!");
            rezzer (n);
            stato="secondo";
            return;
            }

    if (stato=="secondo")
            {
            handle=llListen(-1,"",id,"");
            llSetTimerEvent(10);
//Si apre una finestra di dialogo con l'utente per consentirgli di fare la sua
mossa
            llDialog(id,"Quanti elementi vuoi togliere?",[ "1","2","3" ],-1);
            stato ="terzo";
      return;
            }
}

listen(integer channel, string name, key id,string str)
    {
    llListenRemove(qhandle); qhandle=0;
    llShout(9999,"DELETE");
      llListenRemove(handle);
    llSetTimerEvent(0);
      if(stato=="terzo")
        {
            while (n>1)
            {
            index=(integer)(str);
            llSay(0,"Hai tolto "+(string)index);
            n=n-index;
            llSay(0,"Ora gli elementi sono "+(string)n);
            rezzer (n);
            n=n-(4-index);
            llSay(0,"PC: ho mosso, ora gli elementi sono "+(string)n);
            rezzer (n);
            if (n==1)
            {llSay(0,  "Puoi  togliere  solo  l'ultimo  elemento,  quindi  hai
perso");}

            else
                  {
                  stato="secondo";
                  return;
                  }
            }
            }

        llResetScript();
        }
    timer()
    {
            llListenRemove(handle);
            llSetTimerEvent(0);
            llResetScript();
    }
}
```

Il gioco dei dadi craps

Si tratta di un gioco di sorte basato sul lancio di due dadi cubici.
Regole del gioco:
Ogni volta che i dadi vengono gettati, ne vengono sommati i punteggi ottenuti.
Il giocatore vince immediatamente se ottiene come punteggio 7 oppure 11;
perde se ottiene 2, 3 oppure 12.
Se ottiene 4, 5, 6, 8, 9, 10 deve ricordare questo punteggio P
e ripetere il lancio dei dadi finché vince ottenendo di nuovo il punteggio P,
oppure perde ottenendo 7.

Per la realizzazione del gioco è stato costruito: un oggetto (nel nostro caso un altro ipercubo) nel cui contenuto o stati inseriti:

- un oggetto (nel nostro caso un altro ipercubo) nel cui contenuto è stata inserita una notecard con le regole del gioco in modo che chi clicca sull'oggetto le può leggere;
- due dadi cubici cui sono state applicate le texture delle facce dei dadi (realiazzate singolarmente con un programma di grafica), nel cui contenuto è stato inserito lo script per farli ruotare nello spazio (simulazione del lancio);
- lo script dell'oggetto che esegue il programma del gioco e dialoga con i cubi;
- i dadi e l'oggetto costituiscono un link-set di cui l'oggetto ipercubo è la radice.

Figura 3: Sperimentazione del gioco dei dadi craps

Per realizzare la costruzione dei dadi è stato necessario individuare il numero che individua ciascuna faccia di un cubo (non immediatamente individuabile), per applicarle la texture con il numero punti giusti in modo che l'animazione ponesse, guardando il dado dall'alto (come avviene nella realtà), la faccia col numero casuale corrispondente, generato dallo script. Abbiamo utilizzato il seguente script per colorare e dunque individuare il numero di ogni singola faccia per applicare l'immagine corretta.

```
default
{
       touch_start(integer total_number)
{
//dopo aver individuato la faccia numero 4, ad essa viene applicata la texture
con 4 punti
```

```
            llSetColor(<llFrand(1),llFrand(1),llFrand(1)>, 4);

    }
}
```

Nei due dadi è stato inserito lo stesso script (l'unica differenza è che la scritta sul prim indica la dicitura "dado 2"). Esso consente di eseguire una lista di rotazioni su comando dell'oggetto radice. Il link-set crea cliccando nell'ordine sugli oggetti: dado 1, dado 2, poi sulla radice e poi sono stati si clicca contemporaneamente sui tasti Ctrl+L..

```
list rotations=[
   <270,0,0>,
   <0, 270, 0>,
   <270,0, 180>,
   <180,270,0>,
   <180,0,0>,
   <0,0,0>
    ];

setFrame(integer i)
{
        vector euler=llList2Vector(rotations,i-1);
//è stato inserito i-1 perché il primo elemento delle liste in LSL occupano il
posto zero

        rotation rot=llEuler2Rot(DEG_TO_RAD*euler);
        llSetLocalRot(rot);
}

default
{
    state_entry()
    {
        llSetText("Dado 2",<1,0,0>,1 );
        setFrame(0);
    }
    link_message(integer sender,integer channel,string str,key id)
    {
        //llSay(0,"received "+str);
        integer tick=(integer)str;
        setFrame(tick);
    }
}
```

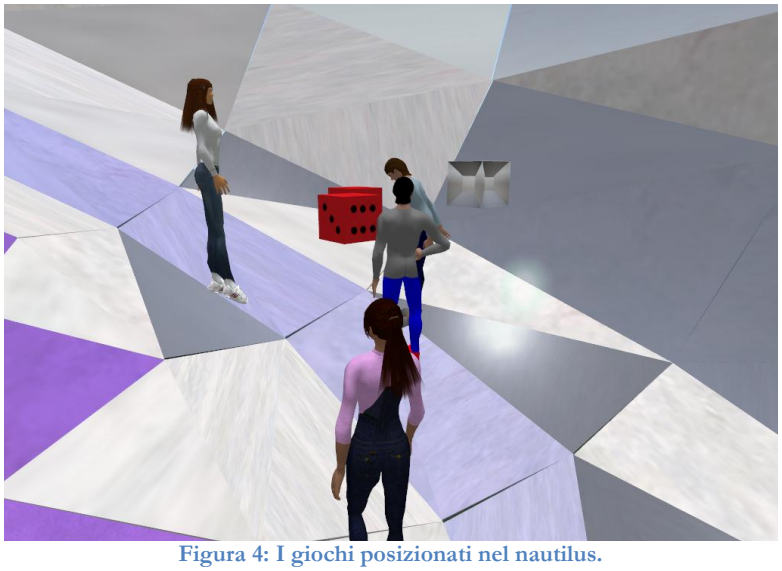

Figura 4: I giochi posizionati nel nautilus.

Il codice che esegue il gioco, dialoga con l'utente e mette in movimento i dadi è il seguente:

```
// nella radice del gioco dei dadi craps
integer dado1;
integer dado2;
integer totale;
integer tot;
integer t;

//questa funzione genera due numeri casuali da 1 a 6 che comunica ai due dadi
rispettivamente sul canale 90, calcola il punteggio totale e lo mostra in chat

integer lancia()
      {dado1=(integer)(llFrand(6)+1);
            llSay(0, "Dado 1: "+ (string)dado1);
            llMessageLinked(3,90,(string)dado1, NULL_KEY);
      dado2=(integer)(llFrand(6)+1);
            llSay(0, "Dado 2: "+ (string)dado2);
            llMessageLinked(2,90,(string)dado2, NULL_KEY);
      totale = dado1 + dado2;
            llSay (0, "Totale: "+ (string)totale);
            return totale;}

//questa funzione ritenta il lancio e secondo le regole del gioco fornisce gli
esiti
integer ritenta(integer input)
      {llSay(0, "Ritenta!");
      do
      { t=lancia();
      llSay(0, "Al rilancio hai totalizzato " + (string)t);
      if(t==7)
            {llSay(0, "Hai perso! Al rilancio hai riottenuto 7 "); return
t;}else {
             if (t==input) {llSay(0, "Hai vinto! Al rilancio riottenuto lo
stesso punteggio iniziale "+(string)tot); return t;}
                  }

      }
      while (t!=input && t!=7);
return t;}
```

```
default
{
    touch_start(integer total_number)
    {
       key id=llDetectedKey(0);
//viene rilasciata la notecard con le istruzioni
       llGiveInventory(id, llGetInventoryName(INVENTORY_NOTECARD, 0));
        llSetTimerEvent(15);
    //istruzione di chiamata della funzione che simula il lancio
       tot=lancia();

        if (tot==7 || tot==11)       {llSay(0, "Nuova partita. Hai vinto, hai
totalizzato "+ (string)totale);}
        else {
            if (tot==2 || tot==3 ||tot==12)   { llSay(0, "Nuova partita. Hai
perso, hai totalizzato "+ (string)totale);}
            else {ritenta(tot);
                        }
        }
     }
}
```

Ho scelto il gioco dei dadi craps perché di facile comprensione, molto presente nei casinò online e in alcuni siti è presentato come un gioco in cui si vince facilmente.
Questo lavoro ha fornito l'occasione non solo per trattare di programmazione in modo ludico, ma per discutere di probabilità e sfatare l'illusione di poter realizzare facili guadagni col gioco d'azzardo. Abbiamo definito insieme lo spazio degli eventi e avvalendoci di una ricerca su Internet è stato possibile trovare un'immagine con le coppie di valori che si possono presentare sulle facce di due dadi cubici perfettamente equilibrati. Per ciascuno dei punteggi totali abbiamo calcolato la probabilità secondo l'impostazione classica ed è stato subito evidente che non tutti i punteggi totali presentano lo stesso numero di casi elementari favorevoli (in figura è visibile un'immagine con lo spazio degli eventi e la distribuzione di probabilità).

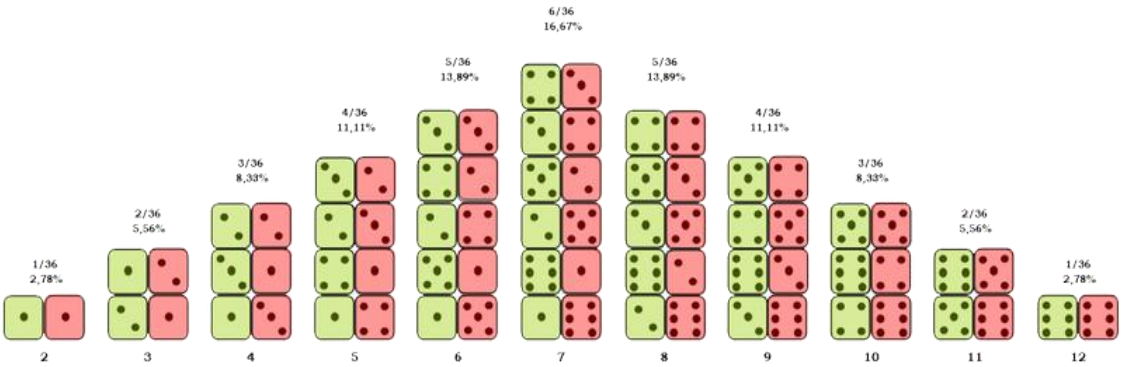

Figura 5 immagine tratta da: http://www.optionclub.it/probabilita_2_.html

Il gioco della morra cinese

Questo è un classico gioco di mano molto popolare in cui il numero delle situazioni possibili è finito. La strategia tra giocatori umani riguarda solo l'uso della psicologia per predire o influenzare le scelte dell'avversario [cfr. 7].
I due giocatori tengono la mano chiusa a pugno e la fanno dondolare, al "Via" ogni giocatore cambia immediatamente il pugno in una delle tre possibili "mani" (o armi):
-Sasso: la mano chiusa a pugno.
-Carta: la mano aperta con tutte le dita stese.
-Forbice: mano chiusa con indice e medio estesi a formare una V.

Lo scopo è sconfiggere l'avversario scegliendo un segno in grado di battere quella dell'altro, secondo le seguenti regole:

1. Il sasso spezza le forbici (vince il sasso)
2. Le forbici tagliano la carta (vincono le forbici)
3. La carta avvolge il sasso (vince la carta)

Se i due giocatori scelgono la stessa arma, il gioco è pari e si gioca di nuovo.

- 0 corrisponde al pugno chiuso, cioè "Sasso";
- 1 rappresenta la mano tesa, cioè "Carta";
- 2 rappresenta le due dita indice e medio, cioè "Forbici";

Le situazioni possibili sono:

Giocatore 1	Giocatore 2	Vincitore
0	0	Pari
0	1	Giocatore 2
1	0	Giocatore 1
1	1	Pari
0	2	Giocatore 1
2	0	Giocatore 2
2	2	Pari
2	1	Giocatore 1
1	2	Giocatore 2

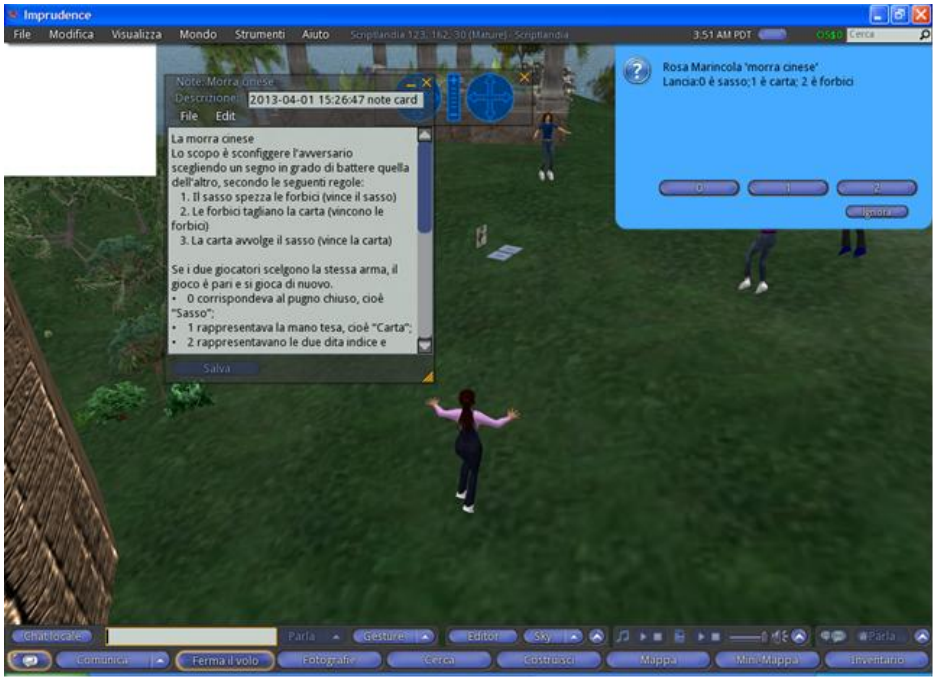

Figura 6: La realizzazione del gioco della morra cinese, visibili la notecard e la finestra di dialogo.

Lo script da inserire nella radice del link-set è il seguente:

```
integer giocatore1;
integer giocatore2;
integer index;
integer handle;
default
{
    state_entry()
    {
            llSetText("Clicca per giocare\n alla morra cinese \n contro il
PC",<1,1,1>,1);
            llMessageLinked(2,10,"giocatore","");
            llMessageLinked(3,10,"giocatore","");

      }

      touch_start(integer total_number)
      {
            llSay(0, "Lancio casuale, il PC è il giocatore 1");
//il PC gioca contro l'avatar e genera la sua arma in modo casuale
            giocatore1=(integer)(llFrand(3));
            llSay(0,"Il giocatore 1 ha lanciato "+(string)giocatore1);
            key id=llDetectedKey(0);
            llGiveInventory(id, llGetInventoryName(INVENTORY_NOTECARD, 0));
            handle=llListen(-1,"",id,"");
            llSetTimerEvent(10);
//la finestra di dialogo consente all'utente di scegliere cosa lanciare
      llDialog(id,"Lancia:0 è sasso;1 è carta; 2 è forbici",[ "0","1","2" ],-1);

}

listen(integer channel, string name, key id,string str)
    {
      llListenRemove(handle); llSetTimerEvent(0);
```

```
    index=(integer)(str);
    giocatore2=index;
    llSay(0,"Il giocatore 2 ha lanciato "+(string)giocatore2);

//in base al numero generato in modo casuale e alla scelta effettuata dal
giocatore sulle due carte appare l'immagine corrispondente, il dialogo avviene
sul canale 10
      if(giocatore1==0)
                              llMessageLinked(2,10,"sasso","");
      if(giocatore1==1)
                              llMessageLinked(2,10,"carta","");
      if(giocatore1==2)
                              llMessageLinked(2,10,"forbici","");
      if(index==0)
                              llMessageLinked(3,10,"sasso","");
      if(index==1)
                              llMessageLinked(3,10,"carta","");
      if(index==2)
                        llMessageLinked(3,10,"forbici","");
llSetTimerEvent(0);

        if(giocatore1==giocatore2) {llSay(0, "Pari");}
                  else
                              {
                        if ((giocatore1==0 && giocatore2==1)||(giocatore1==1 &&
                              giocatore2==2)       ||       (giocatore1==2       &&
giocatore2==0))
                              {llSay(0,llDetectedName(0)+"Hai vinto tu");}
                                    else
                                          {llSay(0, "Ha vinto il giocatore 1 cioè
il PC");}
                                    llSleep(8);
                                    llMessageLinked(2,10,"giocatore","");
                                    llMessageLinked(3,10,"giocatore","");}
      llResetScript();
      }
}
```

Figura 7: il gioco ormai funzionante posto nel nautilus.

Le due carte della morra sono state costruite a partire da una primitiva a forma di cubo, poi deformata modificandone le dimensioni (è stata azzerata la quota e modificata l'ascissa) in modo da ottenere un rettangolo. Nel contenuto sono state inserite 4 immagini (texture): una con un disegno tipico delle corte da gioco (come in figura 7) e le immagini rappresentanti "Sasso" , "Carta", "Forbici". Per creare il link-set occorre cliccare nell'ordine sulle due carte e poi sulla radice e selezionare contemporaneamente i tasti Ctrl+L. Nelle carte è stato inserito anche il seguente script:

```
string texture;
 default
{

      link_message(integer sender_num, integer num, string mess, key id)
     {
          if(mess=="giocatore")
                    {texture ="giocatore";  llSetTexture(texture, ALL_SIDES);}
          if(mess=="sasso")
                    {texture ="sasso"; llSetTexture(texture, ALL_SIDES);}
          if(mess=="carta")
                    {texture ="carta"; llSetTexture(texture, ALL_SIDES);}
          if(mess=="forbici") {texture ="forbici";
                                        llSetTexture(texture, ALL_SIDES);}
     }
}
```

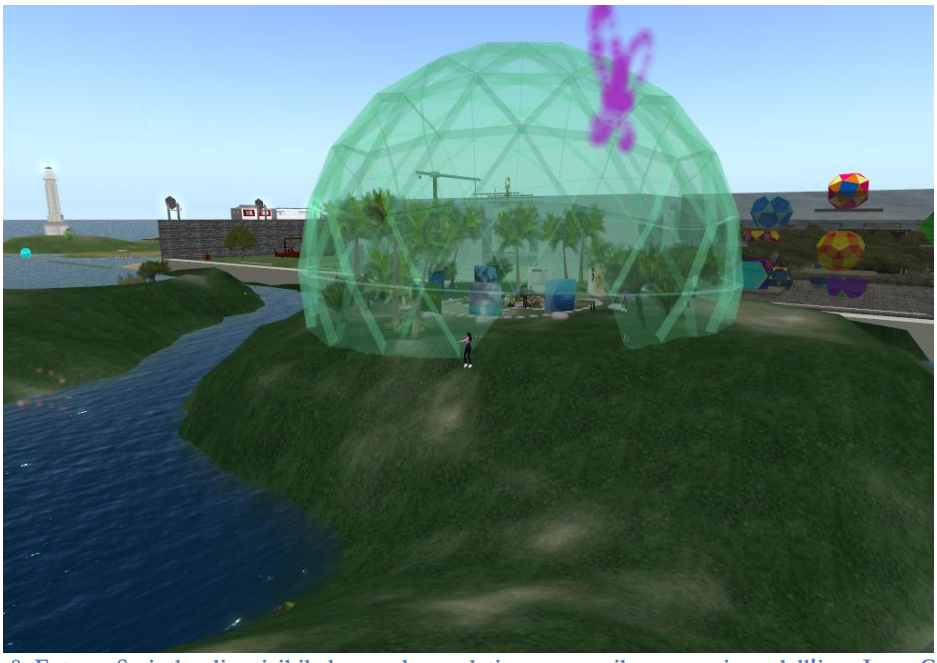

Figura 8: Foto su Scriptlandia, visibile la cupola geodetica per gentile concessione dell'ing. Luca Galletti.

Conclusioni e ringraziamenti

Realizzare i giochi è stato un modo efficace per far riflettere gli studenti sui vari aspetti dei problemi legati alla stesura degli algoritmi risolutivi e alla codifica in LSL, ma, oltre a questi aspetti classici della programmazione, essi hanno dovuto anche costruire gli oggetti, curare le animazioni e gestire la corretta comunicazione tra prim mediante appositi canali. L'utilizzo dei mondi virtuali, quindi, consente un arricchimento e un ampliamento degli elementi da gestire rispetto a quanto avviene utilizzando altri ambienti di programmazione. Il contesto offre sempre nuovi spunti per attività con gli studenti, che, trovandosi in un ambiente a loro familiare, molto simile a quello di alcuni videogames che abitualmente

utilizzano, dimostrano una maggiore determinazione nel voler sperimentare e realizzare dei lavori significativi. I luoghi virtuali, consentono anche di realizzare costruzioni sempre nuove e rinnovare l'aspetto delle land con nuovi spazi espositivi.
Ringrazio l'ing. Luca Galletti per averci concesso la sua cupola geodetica (vedi figura 8) e per aver importato con mesh triangolare il nautilus e il paraboloide iperbolico da me modellati col software free Blender, poiché i file con esso generati, non sono visibili con alcuni viewer come Imprudence (software con cui si accede ai mondi virtuali).

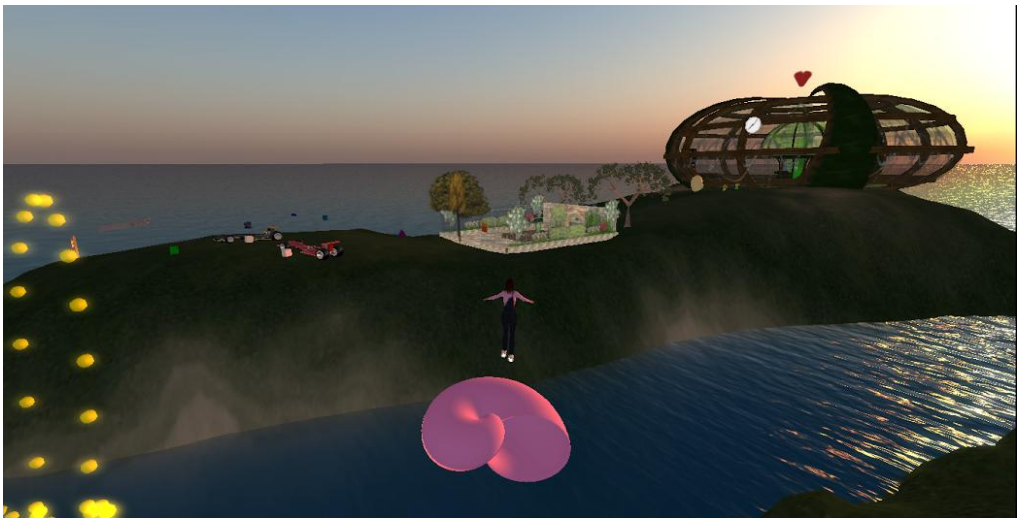

Figura 9: La mesh del nautilus da me realizzata con Blender

Sitografia essenziale

[1] Lezioni di scripting in LSL a Scriptlandia
http://www.matematicamente.it/magazine/18dic2012/177marincola-scriptlandia.pdf
[2] Curve algebriche: gioielli virtuali
http://www.matematicamente.it/magazine/19aprile2013/179-Maricola-Curve.pdf
[3] Teoria dei giochi
http://it.wikipedia.org/wiki/Teoria_dei_giochi
[4] Dispensa di Fondamenti di Informatica
http://www.di.uniba.it/~plantamura/DispenseFI0506/Algoritmi%20per%20giochi.pdf
[5] Procedure e funzioni
http://www.dsi.unive.it/~prog1/Esercizi_in_aula/Esercizi5.pdf
[6] Introduzione al metodo Monte Carlo
didattica.dma.unifi.it/WebWrite/pub/Fisica/.../Metodi_MonteCarlo.odt
[7] Morra cinese
http://it.wikipedia.org/wiki/Morra_cinese
[8] Blender, software free di modellazione 3D
http://www.blender.org/