179. Curve algebriche: gioielli virtuali

Rosa Marincola
rosamarincola@virgilio.it

Premessa

Con EdMondo[1], l'INDIRE offre uno spazio gratuito (land) in un mondo virtuale, simile a Second Life e altri mondi virtuali 3D OpenSim, ma protetto, perché riservato a docenti e studenti italiani (anche minorenni), per sperimentare percorsi didattici innovativi, in una realtà "immersiva", dove è possibile collaborare in presenza e a distanza. L'ambiente consente la comunicazione sincrona sia in chat sia in voice, la presenza è simulata da avatar come nei videogames 3D abitualmente utilizzati dai ragazzi, con la differenza che i soggetti coinvolti sono risolutori e poi esecutori dei "videogiochi", non semplici fruitori. È possibile, infatti, esplorare, costruire (rezzare), ma soprattutto "animare" gli oggetti ossia simulare movimenti, comportamenti attraverso dei programmi: gli script, in LSL (Linden Scripting Language), un linguaggio simile a C, C #, Java, ma fortemente tipizzato.
A differenza di altri linguaggi con interfaccia grafica si possono utilizzare non solo le funzioni standard di libreria, gli operatori logici e aritmetici o le consuete tecniche di programmazione (metodologia top-down e altre), ma ad esempio per produrre animazioni o rezzare gli oggetti in particolari posizioni occorre predisporre dei modelli matematici in cui si utilizzano vettori e trasformazioni geometriche (in particolare rotazioni e traslazioni in R3), equazioni parametriche, diversi sistemi di riferimento, ecc. Tutto ciò crea un'interessantissima commistione per l'apprendimento integrato dell'informatica e della matematica in uno scenario motivante per gli adolescenti.
In questo contributo, in prosecuzione col "Lezioni di scripting in LSL a Scriptlandia"[2], presenterò degli script (codici scritti in LSL), ma non per generare animazioni di oggetti, dialoghi o cambiamenti di texture, bensì per applicazioni matematiche: generare (plottare) curve algebriche da un prim. Tali costruzioni, possono costituire un nuovo modo per affrontare lo studio delle curve e delle loro proprietà, in un contesto, che ne esalta anche gli aspetti estetici.
Nota di building: per ciascuna delle seguenti curve occorre rezzare un cubo e una sferetta di piccole dimensioni, rinominata "punto" fare il take, cioè "prenderla" nell'inventario e trascinarla nel contenuto del cubo, in cui inserire anche lo script di ciascuna curva..

Figura 1: suggestivo fascio di semicirconferenze

[1] http://www.scuola-digitale.it/ed-mondo/progetto/info/
[2] Matematicamente.it Magazine N.18 Dicembre 2012 http://www.matematicamente.it/magazine/18dic2012/177marincola-scriptlandia.pdf

La catenaria

Ho trattato l'argomento nell'articolo "Elementi di geometria descrittiva in natura e in architettura" (http://www.matematicamente.it/magazine/maggio2008/7-M.Marincola-Geometria_descrittiva.pdf).
Per prendere visione della tipologia di ambiente e per favorire quanti non abbiano molta familiarità con tali piattaforme, suggerisco di seguire il video You Tube: Lezioni di scripting - Edmondo Rappresentazione tridimensionale di curve parametriche dell'esperto programmatore Salahzar Stenvaag: https://www.youtube.com/watch?v=xk5uloijNxs .
Lo script di base utilizzato per plottare le curve è stato distribuito dal medesimo esperto durante la lezione di scripting in Edmondo del 16/01/2013 a cui ho partecipato insieme ad altri docenti.

Figura 2: catenaria

```
float e=2.718281;
integer angle=-180;
vector u;
vector v;
vector center;

plot(vector x)
{
    string modo="rez";
    llRezObject("punto",x,ZERO_VECTOR,ZERO_ROTATION,0);

}
//funzione per calcolare il coseno iperbolico
float cosh(float rad)
{ return
(llPow(e, rad)+(1/llPow(e, rad)))/2;
}

vector calcolapos(integer angle)
{
       float rad=DEG_TO_RAD*angle;
//equazione parametrica della catenaria
            return rad*u+cosh(rad)*(v)+center;

}

default
{
    touch_start(integer total_number)
    {
     llSetText("catenaria", <0,0,0>, 1);
        v=llVecNorm(llRot2Left(llGetRot()));
        u=llVecNorm(llRot2Fwd(llGetRot()));

        center=llGetPos();
        // metodo con il timer per generare meno lag

        llSetTimerEvent(0.01);
```

```
    }
    timer()
    {
        vector pos=calcolapos(angle);
        plot(pos);
        angle+=5;
        if(angle>180) llSetTimerEvent(0);

    }
}
```

Il quadrifoglio

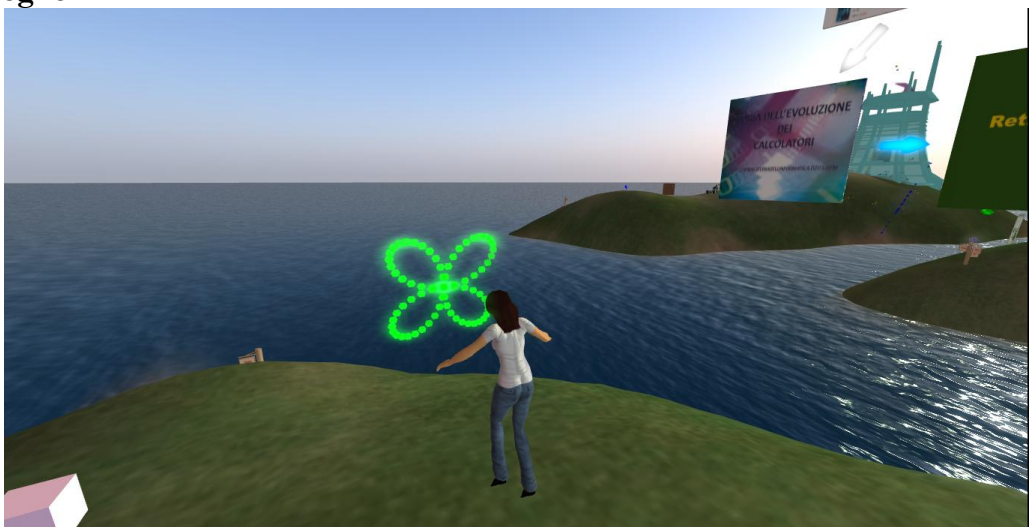

Figura 3: quadrifoglio

```
plot(vector x)
{
    string modo="rez";
    llRezObject("punto",x,ZERO_VECTOR,ZERO_ROTATION,0);

}
vector calcolapos(integer angle)
{
    float rad=DEG_TO_RAD*angle;
        return

//Equazione parametrica del quadrifoglio

(2*llCos(rad)*llPow(llSin(rad),2))*u+(2*llSin(rad)*llPow(llCos(rad),2))*(v)+center;

}

vector u; vector v;
vector center;
integer angle;

default
{
    touch_start(integer total_number)
    {

        v=llVecNorm(llRot2Left(llGetRot()));
        u=llVecNorm(llRot2Fwd(llGetRot()));

        center=llGetPos();
        angle=0;
        llSetTimerEvent(0.01);

    }
    timer()
    {
        vector pos=calcolapos(angle);
        plot(pos);
        angle+=5;
```

```
        if(angle>360) llSetTimerEvent(0);

    }
}
```

La rodonea

Figura 4: rodonea

```
//http://www.mi.imati.cnr.it/~alberto/mnG70cURPISP.pdf

plot(vector x)
{
    string modo="rez";
    llRezObject("punto",x,ZERO_VECTOR,ZERO_ROTATION,0);

}
vector calcolapos(integer angle)
{
        float rad=DEG_TO_RAD*angle;

//equazione parametrica della rodonea
    return (llCos(rad)*llSin(5*rad/7))*u+(llSin(rad)*llSin(5*rad/7))*(v)+center;

}

vector u; vector v;
vector center;
integer angle;

default
{
    touch_start(integer total_number)
    {

        v=llVecNorm(llRot2Left(llGetRot()));
        u=llVecNorm(llRot2Fwd(llGetRot()));

        center=llGetPos();
        angle=-500;
        llSetTimerEvent(0.01);

    }
    timer()
    {
        vector pos=calcolapos(angle);
        plot(pos);
        angle+=8;
        if(angle>740) llSetTimerEvent(0);

    }
}
```

Lacrima

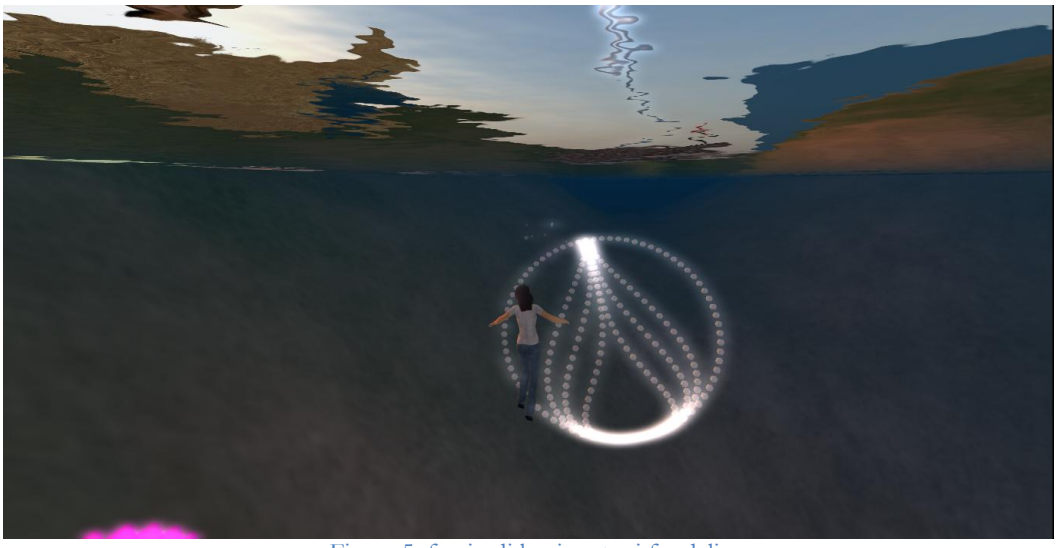

Figura 5: fascio di lacrime tra i fondali.

```
plot(vector x)
{
    string modo="rez";
    llRezObject("punto",x,ZERO_VECTOR,ZERO_ROTATION,0);

}
vector calcolapos(integer angle)
{
    float rad=DEG_TO_RAD*angle;
//equazione parametrica di una curva lacrima
        return 2*llCos(rad)*u+(2*llSin(rad)*llPow(llSin(rad/2),4))*(v)+center;

//per ottenere un fascio di curve, dopo averne rezzato una, lasciare il prim generatore nella stessa
posizione e variare l'esponente della potenza llPow e sostituire il 4 con 2 e con 8

//per rezzare la circonferenza circoscritta al fascio di curve, sostituire l'equazione parametrica
con la seguente (lasciare il prim generatore nella stessa posizione):
2*llCos(rad)*u+(2*llSin(rad))*(v)+center;
//per ottenere le semicirconferenze dell'arcobaleno, occorre far variare l'angolo da 0 a 180°

}

vector u; vector v;
vector center;
integer angle;

default
{
    touch_start(integer total_number)
    {

        v=llVecNorm(llRot2Left(llGetRot()));
        u=llVecNorm(llRot2Fwd(llGetRot()));

        center=llGetPos();


        // metodo con il timer
        angle=0;
        llSetTimerEvent(0.01);
       }
    timer()
    {
        vector pos=calcolapos(angle);
        plot(pos);
        angle+=5;
        if(angle>360) llSetTimerEvent(0);

     }
}
```

La farfalla

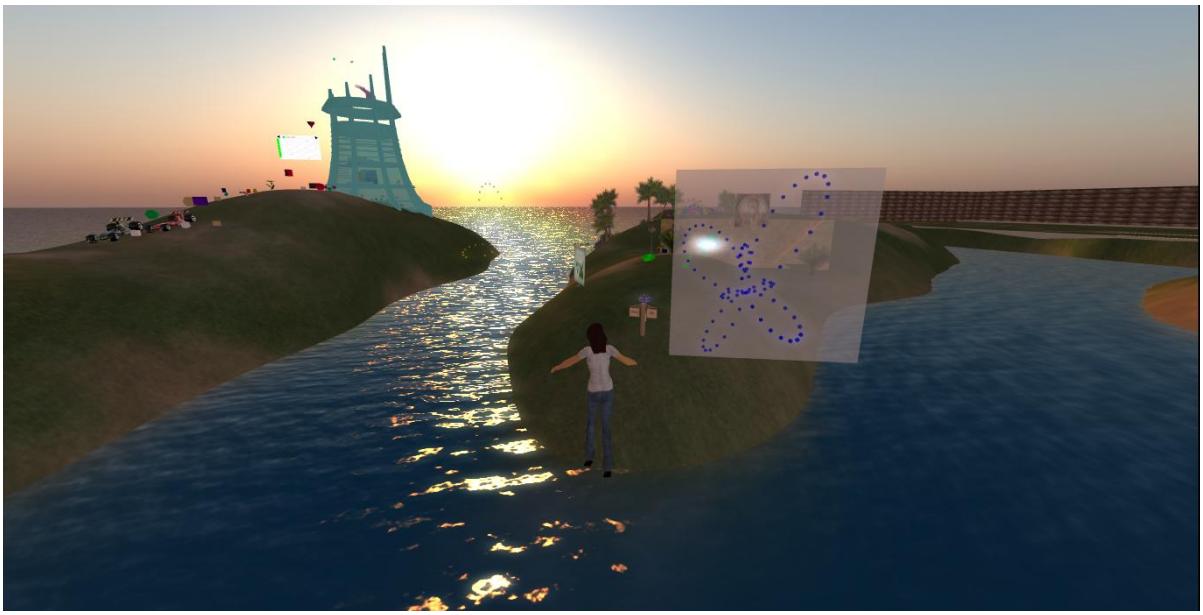

Figura 6: curva farfalla

http://www.mi.imati.cnr.it/~alberto/mnG70cURPISP.pdf

```
float e=2.7;
      integer angle=0;
   vector u;
   vector v;
vector center;

plot(vector x)
{
    string modo="rez";
    llRezObject("punto",x,ZERO_VECTOR,ZERO_ROTATION,0);

}

vector calcolapos(integer angle)
{
    float rad=DEG_TO_RAD*angle;
          return
  llSin(rad)*(llPow(e,     llCos(rad))-     2*llCos(4*rad)+llPow(llSin(rad/12-PI/24),     5))*u     +
llCos(rad)*(llPow(e, llCos(rad))-2*llCos(4*rad)+
llPow(llSin(rad/12-PI/24),5))*(v)+center;
 }

default
{
    touch_start(integer total_number)
    {
     llSetText("Farfalla", <0,0,0>, 1);
        v=llVecNorm(llRot2Left(llGetRot()));
        u=llVecNorm(llRot2Fwd(llGetRot()));

        center=llGetPos();
        // metodo con il timer

        llSetTimerEvent(0.01);

    }
    timer()
    {
        vector pos=calcolapos(angle);
        plot(pos);
        angle+=2;
        if(angle>360) llSetTimerEvent(0);

    }
}
```

Figura 7: curva farfalla con effetti speciali.

L'elica

Figura 8: elica

```
plot(vector x)
{
    string modo="rez";
    llRezObject("nuvola",x,ZERO_VECTOR,ZERO_ROTATION,0);
}
vector calcolapos(integer angle)
{
    float rad=DEG_TO_RAD*angle;
        return 2*llCos(2*PI*rad)*u+2*llSin(2*PI*rad)*(v)+center+4*rad*w ;

}

vector u;
vector v;
vector w;
vector center;
integer angle;

default
{
    touch_start(integer total_number)
    {

        v=llVecNorm(llRot2Left(llGetRot()));
        u=llVecNorm(llRot2Fwd(llGetRot()));
               w=llVecNorm(llRot2Up(llGetRot()));
        center=llGetPos();
```

```
        angle=0;
        llSetTimerEvent(0.01);

    }
    timer()
    {
        vector pos=calcolapos(angle);
        plot(pos);
        angle+=5;
        if(angle>360) llSetTimerEvent(0);

    }
}
```

Il Nastro di Möbius

Figura 9: nastro di Möbius

```
//Al posto della sferetta, per il rezzing, utilizzare un prim di nome "mattone"
//a forma di piccolo parallelepipedo, in questo modo si otterrà come effetto una superficie anziché
di una curva. Trascinare il "mattone" dal proprio inventario in quello del prim da cui verrà rezzata
la curva
plot(vector x)
{
    string modo="rez";
    llRezObject("mattone",x,ZERO_VECTOR,ZERO_ROTATION,0);

}
vector calcolapos(integer angle)
{
        float rad=DEG_TO_RAD*angle;
        //equazione parametrica
        return
(1+0.5*llCos(rad/2))*llCos(rad)*u+(1+0.5*llCos(rad/2))*llSin(rad)*(v)+center+0.5*llSin(rad/2)*w;

}

vector u;
vector v;
vector w;
vector center;
integer angle;

default
{
    touch_start(integer total_number)
    {
  llSetText("Nastro di Moebius", <1,1,1>,1);
        v=llVecNorm(llRot2Left(llGetRot()));
        u=llVecNorm(llRot2Fwd(llGetRot()));
w=llVecNorm(llRot2Up(llGetRot()));
        center=llGetPos();
```

```
        // metodo con il timer
        angle=0;
        llSetTimerEvent(0.01);

    }
    timer()
    {
        vector pos=calcolapos(angle);
        plot(pos);
        angle+=8;
        if(angle>720) llSetTimerEvent(0);

    }
}
```

Conclusioni

In questo contributo ho voluto mostrare la bellezza di alcuni modelli matematici in 2D e 3D realizzati con l'uso delle tecnologie in ambiente virtuale "giocando" in modo opportuno sulla variazione di angoli, vettori ed equazioni parametriche a partire da un unico script di base.

Sitografia essenziale

EdMondo:
http://www.scuola-digitale.it/ed-mondo/progetto/info/
LSL Portal:
http://wiki.secondlife.com/wiki/LSL_Portal
A quanti non conoscono il linguaggio LSL segnalo un mio breve tutorial con riferimenti anche ad altre risorse sull'argomento: Lezioni di scripting in LSL a Scriptlandia
http://www.matematicamente.it/il_magazine/numero_18%3a_dicembre_2012/177._lezioni_di_scripting_in_lsl_a_scriptlandia_201302017915/
Elementi di geometria descrittiva in natura e in architettura:
http://www.matematicamente.it/magazine/maggio2008/7-M.Marincola-Geometria_descrittiva.pdf
Repertorio di curve piane speciali:
http://www.mi.imati.cnr.it/~alberto/mnG70cURPISP.pdf
Encyclopédie des forms mathématiques remarquables:
http://www.mathcurve.com/index.htm
Il nastro di Möbius:
http://areeweb.polito.it/didattica/polymath/htmlS/argoment/Matematicae/Aprile_07/AnelliMobius.htm
Playlist dei video di scripting EdMondo realizzati da Salahzar Steenvag:
http://www.youtube.com/playlist?list=PLXTK688RnbsFbDvW-5l8u8vdAgPoC_y_E