Gráficos 3D Interactivos con Mathematica y LiveGraphics3D[1]

Walter Mora F
Escuela de Matemática
Instituto Tecnológico de Costa Rica

Resumen:

Se presentan algunos principios de programación de gráficos 3D usando Mathematica y LiveGraphics3D. Se muestra como hacer depender uno o varios objetos de un gráfico de uno o varios parámetros, llamados variables independientes, de tal manera que al variar estos parámetros, a través de puntos de arrastre, varían los objetos gráficos.

Palabras claves: Mathematica, GraphicsLive3D, gráficos parametrizados, parametrización, programación de gráficos.

Introducción

En esta comunicación se establece la manera de interactuar con los objetos geométricos 3D a través de la definición de variables independientes y dependientes (gráficos con parámetros), las cuales se pueden hacer variar arrastrando puntos, como se haría en Cabri o en Sketchpad, usando Mathematica y LiveGraphivs3D (versión 1.83). Primero se establecen los conceptos de variables dependientes e independientes, luego se estudia la manera de restringir el recorrido de un punto de arrastre y la manera de hacer depender un objeto geométrico de uno o varios parámetros y finalmente, se hacen aplicaciones relacionadas con la interpretación geométrica de conceptos relativamente complejos en cálculo de varias variables. Los ejemplos incluyen el código para generar de manera automática la página web en la que se levanta el gráfico.

En los ejemplos se usa LiveGraphics3D V 1.83 y DrawGraphics, por lo que se supone que estos componentes están bien instalados en el computador (más que instalar, se trata de pegar archivos en ciertas carpetas). Una introducción ala graficación 3D con Mathematica, LiveGraphics3D y JavaView se puede ver en el número anterior de la revista digital. Aquí mismo se pueden ver ejemplos de aplicación de LG3D en vectores, rectas y planos en el espacio, funciones de varias variables, cálculo y geometría. Otras apolicaciones se pueden encontrar en la página de ejemplos que mantiene el creador de LG3D, Martin Kraus.

Variables independientes y variables dependientes

Un gráfico 'parametrizado' tiene objetos tales como puntos

```
Graphics3D[{ Point[{ x, y , x^2 + y^2 } ] }] }
```

con coordenadas (x, y) que pueden variar (con el arrastre del mouse) .

Las coordenadas (x, y) se deben inicializar para tener una imagen inicial. Luego estas coordenadas se pueden "seleccionar con el mouse", *a través de un punto,* de tal manera que al arrastrar el mouse, sus valores cambian.

Si una variable u es *variable independiente*, su valor se puede modificar seleccionando y arrastrando el mouse sobre un punto que tiene al menos una vez la coordenada u (limpia).

Las variables independientes se declaran con una lista de reglas que inicializan las variables

```
independentVariables = {x -> 2, y -> 2}
```

En este caso, el punto `Point[{x,y,0}]` se puede seleccionar y "arrastrar" .

El punto `Point[{1.*x,1.*y, 2}]` no se puede seleccionar ni arrastrar pues las variables independientes no aparecen "limpias". Sin embargo al arrastrar `Point[{x,y,2}]`, el punto `Point[{1.*x,1.*y, 2}]` se mueve en función del primer punto. En realidad, las expresiones `'1.*x'` , `'1.*y'` son *variables dependientes.*

Las *variables dependientes* son variables que cambian en función de los valores de las variables independientes o son constantes.

Las variables dependientes se declaran con una lista de reglas. Por ejemplo las variables dependientes `xi` y `yi` se pueden declarar en función de las variables independientes `x` y `y`.

```
dependentVariables = {xi -> x/2, yi -> Sqrt[y]}
```

También una variable dependiente puede depender de otra variable dependiente anterior, en la lista de reglas de las variables dependientes

También se usan reglas para definir el comportamiento de las variables independientes pero en la lista de reglas de las variables dependientes. Estas reglas usan expresiones (de *Mathematica*) simples.

Por ejemplo, si queremos que la variable x solo se desplace entre 0 y 1, lo que hacemos es asignarle valor cero si estamos en un valor negativo y asignarle el valor 1 si estamos en un valor mayor que uno y en otro caso, el valor actual:

```
dependentVariables = {x -> If[ x < 0 , 0, If[ x > 1, 1, x] ]}
```

Este conjunto de reglas se pueden usar para implementar el gráfico en *Mathematica* pero se deben indicar y se pueden variar en los parámetros del applet que levanta el gráfico

Por ejemplo, si queremos que un punto $(x, y, 0)$ se mueva sobre un rectángulo $[0,4] \times [0,4]$, declaramos las variables independientes x e y, luego las inicializamos y después definimos su comportamiento con reglas

```
independentVariables = {x -> 2, y -> 2};

dependentVariables = {x -> If[x < 0, 0, If[x > 4, 4, x]],
                      y -> If[y < 0, 0, If[y > 4, 4, y]]};

punto = { AbsolutePointSize[6],
          RGBColor[1, 0, 0],
          Point[{x, y, 0}], (*---> punto de arrastre, se puede seleccionar*)
          Point[{1.*x, 1.*y, 1}],(*---> punto dependiente,No seleccionable*)
      };

g = Graphics3D[{punto}];

Show[N[g] //. Join[independentVariables, dependentVariables]];
```

Esto debe quedar plasmado en los parámetros del applet

```
<PARAM NAME=INDEPENDENT_VARIABLES VALUE='{x->2,y->2}'>
<PARAM NAME=DEPENDENT_VARIABLES VALUE='{x->If[x<0,0,If[x>4,4,x]],y->If[y<0,0,If[y>4,4,y]]}'>
```

Veamos el ejemplo completo

Ejemplo 1

Arrastrar un punto $(x, y, 0)$, inicializado con $x = 2, y = 2$, sobre el rectángulo $[0,4] \times [0,4]$

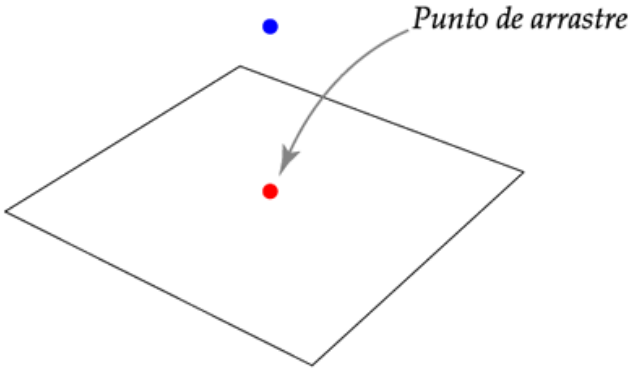

Figura 1.
[Ver con LiveGraphics3D]

El código completo incluye la implementación del gráfico y generar la página Web. Vamos a suponer que el archivo live.jar está en la carpeta `C:\MisLG3D` y que ahí quedará también la página Web

---

```
(*----------------------------  Gráfico  ----------------------------*)
    independentVariables = {x -> 2, y -> 2};
      dependentVariables = {x -> If[x < 0, 0, If[x > 4, 4, x]],
                            y -> If[y < 0, 0, If[y > 4, 4, y]]};

    puntos = {AbsolutePointSize[6],
              RGBColor[1, 0, 0], Point[{x, y, 0}],
              RGBColor[0, 0, 1], Point[{1.*x, 1.*y, 2}]};

    rectangulo = {Line[{{0, 0, 0}, {4, 0, 0}, {4, 4, 0}, {0, 4, 0}, {0, 0,0}}]};

    g = Graphics3D[{rectangulo, puntos},
                   Boxed -> False,
                   ViewPoint -> {1.761, -2.313, 1.732}
                   ];

(*ver el gráfico en Mathematica*)
    Show[N[g] //. Join[independentVariables, dependentVariables]];


(*----------------------------  página Web  ----------------------------*)
(* Generar la página Web 'ejemplo1.html'.                               *)
(* El archivo ejemplo1.m  es el archivo de coordenadas                  *)

    WriteLiveForm["ejemplo1.m", g, Dir -> "C:\\MisLG3D\\"];
(* Crea la página Web *)
    SetDirectory["C:\\MisLG3D\\"];
    strm = OpenWrite["ejemplo1.html"];

(* código de la página Web*)
    pagina = "<HTML><HEAD></HEAD> <BODY>
    <APPLET height=150 width=200  archive=live.jar   code=Live.class >
    <PARAM NAME=INPUT_FILE VALUE = ejemplo1.m >
    <PARAM NAME=INDEPENDENT_VARIABLES VALUE='{x->2,y->2}'>
    <PARAM NAME=DEPENDENT_VARIABLES \
    VALUE='{x->If[x<0,0,If[x>4,4,x]],y->If[y<0,0,If[y>4,4,y]]}'>
    </APPLET>
    </BODY></HTML>";
(* escribir en el archivo*)
    WriteString[strm, pagina];
    Close[strm];
```

Restringiendo el arrastre sobre una curva o sobre una superficie

Podemos usar la lista de reglas de las variables dependientes para restringir el recorrido de un punto. Lo que hacemos es redefinir las variables independientes de tal manera que cumplan las ecuaciones de la curva o la superficie.

Cuando seleccionamos un punto con variables independientes (x, y) y arrastramos el mouse, sobre cualquier sector de la pantalla, el punto de pantalla es proyectado sobre el plano XY (o un plano paralelo al plano XY). Así, si rotamos el gráfico y ponemos el plano XY de frente, entonces el punto (x, y) puede ser aproximadamente igual al punto de proyección (x_p, y_p). En todo caso, la idea es usar reglas para usar el punto (x_p, y_p) para redefinir el punto (x, y) de tal manera que esté en la curva o en la superficie que deseamos.

En el primer ejemplo vamos a usar coordenadas polares para restringir el recorrido de un punto a una circunferencia. El el segundo ejemplo vamos a usar directamente la ecuación de la superficie, en coordenadas rectangulares, para lograr que un punto recorra una superficie.

Ejemplo 2
Arrastrar un punto sobre la circunferencia $x^2 + y^2 = 4$.

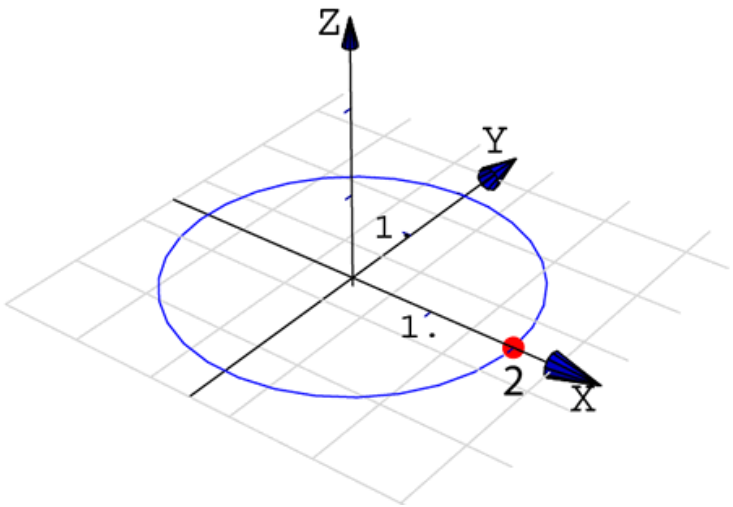

Figura 2.
[Ver con LiveGraphics3D]

Si C es la circunferencia en $\mathbb{R}^3$, en *Mathematica* C se parametriza así

$$C: \quad r[t\_] = \{2 * \text{Cos}[t],\ 2 * \text{Sin}[t],\ 0,\}$$

es decir, $x = 2 * \text{Cos}[t]$, $y = 2 * \text{Sin}[t]$ y $t = \text{ArcTan}[x, y]$.

Dichosamente `ArcTan` devuelve el ángulo de acuerdo al cuadrante en el que se encuentra el punto (x, y).

Estas ecuaciones las usaremos para restringir el recorrido del punto $(x, y, 0)$ de la siguiente manera: cuando el mouse selecciona el punto $(x, y, 0)$ en la pantalla y se produce el arrastre, el punto actual *en pantalla* es proyectado, sobre el plano XY, digamos en el punto (x_p, y_p). Calculamos el ángulo $t = \text{ArcTan}[x_p, y_p]$ y redefinimos (x, y) usando las ecuaciones $x = 2 * \text{Cos}[t]$, $y = 2 * \text{Sin}[t]$ y $t = \text{ArcTan}[x, y]$. Es claro que el punto $(x, y, 0)$ estará en la circunferencia.

En código, esto se vería así

```
independentVariables = {x -> 2, y -> 0};
```

```
dependentVariables = {theta -> ArcTan[x, y],
                      x -> 2*Cos[theta],
                      y -> 2*Sin[theta]
                      };
```

Observe que si ponemos la circunferencia de frente, el arrastre del punto con el mouse parecerá muy natural pues la proyección del punto de pantalla sobre el plano XY puede coincidir con el punto, es decir $(x_p, y_p) \approx (x, y)$.

Si rotamos la figura, el arrastre del punto puede presentar alguna dificultad debido a las restricciones a las que se ve sometida la proyección sobre el plano XY.

El código completo es el siguiente

```
independentVariables = {x -> 2, y -> 0};

dependentVariables = {
                      theta -> ArcTan[x, y],
                      x -> 2*Cos[theta],
                      y -> 2*Sin[theta]
                      };
circulo = Line[Table[{0 + 2*Cos[t], 0 + 2*Sin[t], 0} , {t, 0, 2Pi, 0.2}]];

punto = {
          AbsolutePointSize[6],
          RGBColor[1, 0, 0],
          Point[{x, y, 0}]
        };


g = Graphics3D[{Ejes[-02.5, 3, -2.5, 3, -0.1, 3], punto,
                RGBColor[0, 0, 1], circulo},
                Boxed -> False, ViewPoint -> {1.761, -2.313, 1.732}];

Show[N[g] //. Join[independentVariables,dependentVariables]];

(*página Web*)

WriteLiveForm["ejemplo3.m", g, Dir -> "C:\\MisLG3D\\"];
(* Crea la página Web *)
SetDirectory["C:\\MisLG3D\\"];
strm = OpenWrite["ejemplo3.html"];

(*código de la página Web*)
pagina = "<HTML><HEAD></HEAD> <BODY>
<APPLET height=400 width=400 archive=live.jar code=Live.class >
<PARAM NAME=INPUT_FILE VALUE = ejemplo3.m >
<PARAM NAME=INDEPENDENT_VARIABLES VALUE='{x->2,y->0}'>
<PARAM NAME=DEPENDENT_VARIABLES VALUE='{
                                       theta ->ArcTan[x,y],
                                       x-> 2*Cos[theta],
                                       y->2*Sin[theta]
                                      }'>
</APPLET>
</BODY></HTML>";

WriteString[strm, pagina];
Close[strm];
```

Ejemplo 3

Arrastrar un punto sobre el trozo de superficie

$$S: \quad (x-2)^2 + (y-2)^2 + z - 3 = 0, \quad (x,y) \in [1.25, 2.75] \times [1.25, 2.75].$$

El intervalo escogido es solo para ver mejor la superficie.

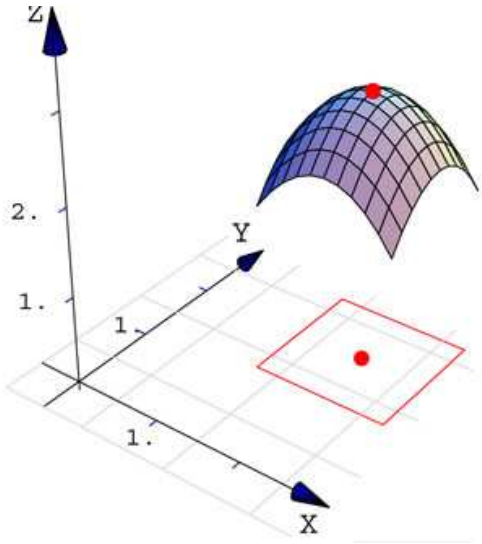

Figura 3.
[Ver con LiveGraphics3D]

En este ejemplo hay dos elementos:

1. tomar un punto de la superficie S y 'pasearlo' por la superficie
2. definir un dominio de acción: solo se consideran puntos (x, y, z) de la superficie S con pre-imagen (x, y) en el cuadrado $[1.25, 2.75] \times [1.25, 2.75]$

Debemos hacer lo siguiente:

1. definir las variables independientes: inicializarlas de tal manera que coorespondan a punto sobre S
2. definir su comportamiento para que, al arrastrar el punto, no nos salgamos de la superficie S
3. definir su comportamiento para que las pre-imagen (x, y) no se salga del dominio $[1.25, 2.75] \times [1.25, 2.75]$
4. Implementar un sistema de ejes 3D
5. implementar la superficie S

- Observemos que $P \in S$ si $P = (x, y, -(x-2)^2 - (y-2)^2 + 3)$, así que para restringir el punto al rectángulo y a la superficie se usarán las reglas

```
independentVariables = {x -> 2, y -> 2};

   dependentVariables = {
      x -> If[x < 1.25, 1.25, If[x > 2.75, 2.75, x]],
      y -> If[y < 1.25, 1.25, If[y > 2.75, 2.75, y]],
      z -> -(x - 2)^2 - (y - 2)^2 + 3
      };
```

- En el código que sigue aparece el siguiente código

```
puntos =
{AbsolutePointSize[6], RGBColor[1, 0, 0],
  Point[{x, y, z}], (*----> punto de arrastre sobre S*)
  Point[{1.*x, 1.*y, z + 0.01}], Point[{1.*x, 1.*y, z - 0.01}],
  Point[{x, y, 0}]  (*----> pre-imagen es también punto de arrastre*)
      };
```

Los puntos `Point[{1.*x, 1.*y, z + 0.01}]` y `Point[{1.*x, 1.*y, z - 0.01}]`, son dos puntos que se

usan previendo el ocultamiento del punto `Point[{x, y, z}]` por los polígonos de la superficie.

- El ejemplo usa la biblioteca 'DrawGraphics' para los ejes 3D. Puede consultar el número anterior de la revista para ver su uso.

El código completo es el siguiente

```
(*------------------------------ ejes 3D ------------------------------*)
Needs["DrawGraphics`DrawingMaster`"];

(* Ejes 3D *)
Ejes[xmin_, xmax_, ymin_, ymax_, zmin_, zmax_] := {
(*Flechas*)
SurfaceColor[RGBColor[0, 0, 1]],
ArrowLine3D[{{xmin, 0, 0}, {0, 0, 0}, {xmax, 0, 0}}],
ArrowLine3D[{{0, ymin, 0}, {0, 0, 0}, {0, ymax, 0}}],
ArrowLine3D[{{0, 0, zmin}, {0, 0, 0}, {0, 0, zmax}}],
Text[
StyleForm["X", FontSize -> 12, FontWeight -> "Bold"], {xmax, -0.3,
0}],
Text[
StyleForm["Y", FontSize -> 12, FontWeight -> "Bold"], {-0.3, ymax,
0.1}],
Text[
StyleForm["Z", FontSize -> 12, FontWeight -> "Bold"], {0, -0.2,
zmax}],
(* numeros *)
Table[Text[i, {i, -0.3, 0}], {i, 1, xmax - 2, 1}],
Table[Text[i, {-0.3, i, 0}], {i, 1, ymax - 2, 1}],
Table[Text[i, {-0.3, -0.3, i}], {i, 1, zmax - 2, 1}],

RGBColor[0, 0, 0.627451],
(* rayitas *)
Table[Line[{{-0.1, i, 0}, {0, i, 0}}], {i, 1, ymax - 1, 1}],
Table[Line[{{i, -0.1, 0}, {i, 0, 0}}], {i, 1, xmax - 1, 1}],
Table[Line[{{0, -0.1, i}, {0, 0, i}}], {i, 1, zmax - 1, 1}],

(*grid*)
GrayLevel[0.843137],
Table[Line[{{xmin, i, 0}, {xmax, i, 0}}], {i, ymin, ymax, 1}],
Table[Line[{{i, ymin, 0}, {i, ymax, 0}}], {i, xmin, xmax, 1}]
};

(*------------------------------ Gráfico ------------------------------*)

independentVariables = {x -> 2, y -> 2};

dependentVariables = { x -> If[x < 1.25, 1.25, If[x > 2.75, 2.75, x]],
y -> If[y < 1.25, 1.25, If[y > 2.75, 2.75, y]],
z -> -(x - 2)^2 - (y - 2)^2 + 3
};
(*Parabolide*)
minx = 1.25; miny = 1.25; maxx = 2.75; maxy = 2.75;
n = 10;
dx = (maxx - minx)/n;
dy = (maxy - miny)/n;
f[vx_, vy_] = -(vx - 2)^2 - (vy - 2)^2 + 3;


paraboloid = Table[Polygon[{{i , j , f[i, j]},
{i + dx , j , f[(i + dx), j]},
{i + dx , j + dy ,f[(i + dx), (j + dy)]},
{i ,j + dy ,f[i, (j + dy)]}}],
{i, minx, maxx - dx/2, dx},
{j, miny, maxy - dy/2, dy}];

(*Punto(s) *)
puntos = {AbsolutePointSize[6], RGBColor[1, 0, 0],
Point[{x, y, z}], (*----> punto de arrastre *)
Point[{1.*x, 1.*y, z + 0.01}], Point[{1.*x, 1.*y, z - 0.01}],
```

```
Point[{x, y, 0}] (*----> pre-imagen*)
};
rectangulo = {Line[{{1.25, 1.25, 0}, {2.75, 1.25, 0}, {2.75, 2.75, 0},
{1.25,2.75, 0}, {1.25, 1.25, 0}}]
};

g = Graphics3D[{Ejes[-0.5, 3, -0.5, 3, -0.1, 4], paraboloid, puntos,
RGBColor[1, 0, 0], rectangulo},
Boxed -> False,
ViewPoint -> {1.761, -2.313, 1.732},
PlotRange -> All,
AspectRatio -> Automatic];

(*ver el gráfico*)
Show[N[g] //. Join[independentVariables, dependentVariables]];


(*----------------------------- página Web -----------------------------*)

WriteLiveForm["ejemplo2.m", g, Dir -> "C:\\MisLG3D\\"];
(* Crea la página Web *)
SetDirectory["C:\\MisLG3D\\"];
strm = OpenWrite["ejemplo2.html"];

(*código de la página Web*)
pagina = "<HTML><HEAD></HEAD> <BODY>
<APPLET height=400 width=400 archive=live.jar code=Live.class >
<PARAM NAME=INPUT_FILE VALUE = ejemplo2.m >
<PARAM NAME=INDEPENDENT_VARIABLES VALUE='{x->2,y->2}'>
<PARAM NAME=DEPENDENT_VARIABLES VALUE='{
x->If[x<1.25,1.25,If[x>2.75,2.75,x]],
y->If[y<1.25,1.25,If[y>2.75,2.75,y]],
z->-(x-2)^2 -(y-2)^2+3}'>
</APPLET>
</BODY></HTML>";

WriteString[strm, pagina];
Close[strm];
```

Escalando una superficie

A veces necesitamos escalar una superficie cuando, por ejemplo, manipulamos la altura o el radio de la base en un cono o cuando hacemos una simulación de la proyección de un sólido sobre un plano. En estas y otras situaciones, tenemos uno o varios puntos de arrastre cuyas variables independientes modifican las otras partes de la superficie.

Para construir una superficie que se pueda escalar, usamos comandos de *Mathematica* que soporten expresiones simbólicas, usualmente primitivas (por ejemplo, Plot3D no las acepta), para construir la superficie de tal manera que los polígonos que la conforman estén en función de las variables independientes.

En nuestro primer ejemplo vamos a construir un círculo que se escala al manipular su radio a través de un punto. En el segundo ejemplo hacemos lo mismo para un cono, escalándolo al manipular el radio de la base y su altura, finalmente aplicamos esta idea para simular la proyección de una superficie sobre el plano YZ

Ejemplo 4

Implementar un círculo $x^2 + y^2 = r^2$, es decir un círculo de radio variable $r > 0$, que se escale con el arrastre de un punto.

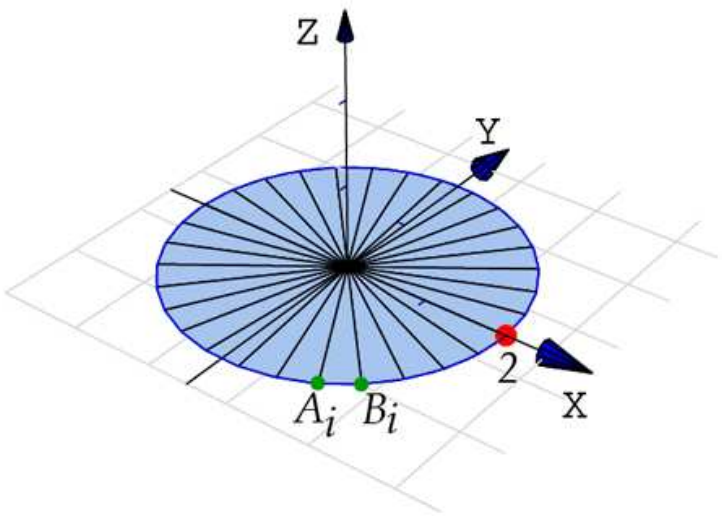

Figura 4.
[Ver con LiveGraphics3D]

Podemos implementar el círculo con n triángulos $\triangle A_i B_i C_i$ con el vértice B_i en el centro. Los otros dos puntos tienen un ángulo fijo y dependen del valor del radio. En efecto, las ecuaciones de los puntos serían

$$A_i = (r*\cos(t_i),\ r*\text{sen}(t_i),\ 0),\quad B_i = (0,0,0),\quad C_i = (r*\cos(t_{i+1}),\ r*\text{sen}(t_{i+1}),\ 0)$$

aquí $t_i = \dfrac{i\,2\pi}{n} \in [0,\ 2\pi],\ i = 0, 1, \cdots, n$

- Para hacer variar r, incluimos en el gráfico el punto de arrastre `Point[{r,0,0}]`. Puesto que el arrastre puede llevar a valores negativos, en vez del radio usamos `ra -> Abs[r]`
- En la implementación, como pasamos de t_i a t_{i+1}, entonces i varia de 1 hasta $n-1$.
- El cono se construye con 30 puntos. Una versión más 'suave' requiere más puntos, digamos 60 puntos. Se debe tener en cuenta que, en gráficos más complicados, el peso del archivo '.m' puede hacerse oneroso tanto al descargarlo como al manipular el gráfico en un navegador.

El código completo es

```
(*------------------------------  Gráfico  ------------------------------*)
  independentVariables = {r -> 2};
  dependentVariables   = {ra -> Abs[r]};

  circunferencia =
    Line[Table[{0 + ra*Cos[t], 0 + ra*Sin[t], 0} , {t, 0, 2Pi, 0.2}]];

    n = 30;
    paso = 2Pi/n;
 circulo =
    Table[Polygon[{ {ra*Cos[i*paso], ra*Sin[i*paso], 0}, {0, 0,0},
                    {ra*Cos[(i + 1)*paso], ra*Sin[(i + 1)*paso],0},
                    {ra*Cos[i*paso], ra*Sin[i*paso], 0}  }]
          , {i, 0, n - 1, 1}];

 punto = { AbsolutePointSize[6],
           RGBColor[1, 0, 0],
           Point[{r, 0, 0}]
         };


g = Graphics3D[{Ejes[-02.5, 3, -2.5, 3, -0.1, 3], punto,
                EdgeForm[], circulo,
                RGBColor[0, 0, 1], circunferencia},
```

```
                Boxed -> False,
                ViewPoint -> {1.761, -2.313, 1.732}];

Show[N[g] //. Join[independentVariables, dependentVariables]];


(*----------------------------  página Web  ----------------------------*)
(* Generar la página Web 'ejemplo4.html'.                               *)
(* El archivo ejemplo4.m  es el archivo de coordenadas                  *)

    WriteLiveForm["ejemplo4.m", g, Dir -> "C:\\MisLG3D\\"];
(* Crea la página Web *)
    SetDirectory["C:\\MisLG3D\\"];
    strm = OpenWrite["ejemplo4.html"];

(* código de la página Web*)
    pagina = "<HTML><HEAD></HEAD> <BODY>
    <APPLET height=150 width=200  archive=live.jar   code=Live.class >
    <PARAM NAME=INPUT_FILE VALUE = ejemplo4.m >
    <PARAM NAME=INPUT_FILE VALUE = ejemplo4.m >
    <PARAM NAME=INDEPENDENT_VARIABLES VALUE='{r->2}'>
    <PARAM NAME=DEPENDENT_VARIABLES VALUE='{ra->Abs[r]}'>
    </APPLET>
    </BODY></HTML>";
(* escribir en el archivo*)
    WriteString[strm, pagina];
    Close[strm];
```

Ejemplo 5

Implementar un cono al que se le pueda manipular la altura y el radio de la base

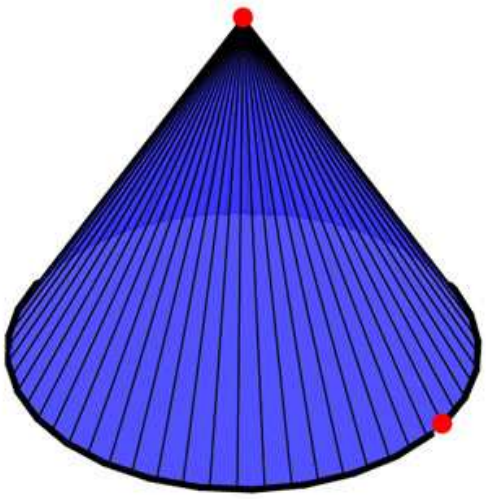

Figura 5.
[Ver con LiveGraphics3D]

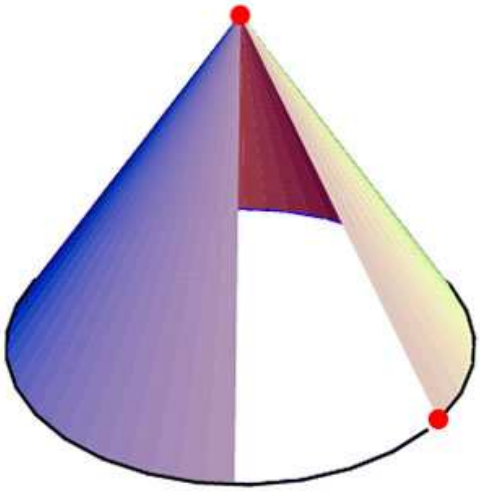

Figura 6.

El código sería similar al del ejemplo anterior. En vez de poner un vértice en el origen, lo ponemos auna altura adecuada. Vamos a mantener la base circular. Tendríamos dos variables independientes: la altura y el radio.

```
independentVariables = {r -> 2, h -> 3};
dependentVariables = {ra -> Abs[r], h -> If[h < 0, 0, h]};

circunferencia = Line[
Table[{0 + ra*Cos[t], 0 + ra*Sin[t], 0} , {t, 0, 2Pi, 0.2}]];
n = 60;
paso = 2Pi/n;
```

```
circulo =
Table[Polygon[{ {ra*Cos[i*paso], ra*Sin[i*paso], 0}, {0, 0, 0},
{ra*Cos[(i + 1)*paso], ra*Sin[(i + 1)*paso], 0}, {ra*Cos[i*paso], ra*Sin[i*paso], 0} }],
{i, 0, n - 1, 1}];


cva = Line[Table[{0 + 2.01*Cos[t], 0 + 2.01*Sin[t], 0} , {t, 0, 2Pi, 0.2}]];


cono = Table[Polygon[{ {ra*Cos[i*paso], ra*Sin[i*paso], 0}, {0, 0, h},
{ra*Cos[(i + 1)*paso], ra*Sin[(i + 1)*paso], 0},
{ra*Cos[i*paso], ra*Sin[i*paso], 0} }], {i, 0, n - 1, 1}];

puntos = {
AbsolutePointSize[6],
RGBColor[1, 0, 0],
Point[{r, 0, 0}],
Point[{0, 0, h}]
};


g = Graphics3D[{
puntos,
RGBColor[0, 0, 1], circunferencia,
cono,
GrayLevel[0],
AbsoluteThickness[2],
cva},
Boxed -> False, ViewPoint -> {1.761, -2.313,1.732},
PlotRange -> All
];

Show[N[g] //. Join[independentVariables, dependentVariables]];

(*Página Web*)

WriteLiveForm["ejemplo5.m", g, Dir -> "C:\\MisLG3D\\"];
(* Crea la página Web *)
SetDirectory["C:\\MisLG3D\\"];
strm = OpenWrite["ejemplo5.html"];

(*código de la página Web*)
pagina = "<HTML><HEAD></HEAD> <BODY>
<APPLET height=400 width=400 archive=live.jar code=Live.class >
<PARAM NAME=INPUT_FILE VALUE = ejemplo5.m >
<PARAM NAME=INDEPENDENT_VARIABLES VALUE='{r->2, h->3}'>
<PARAM NAME=DEPENDENT_VARIABLES VALUE='{ra ->Abs[r],h -> If[h<0,0,h]}'>
</APPLET>
</BODY></HTML>";

WriteString[strm, pagina];(*escribe en el archivo*)
Close[strm];
```

Ejemplo 6 (Proyección perpendicular de una superficie**)**

Sea S la superficie $x^2+y^2=4$ con $x \in [0,2]$, en el primer octante, limitada por el plano $x+y=5$. Para visualizar la proyección perpendicular de esta superficie sobre el plano YZ, vamos a implementar la superficie y luego vamos a declarar a x como variable independiente.

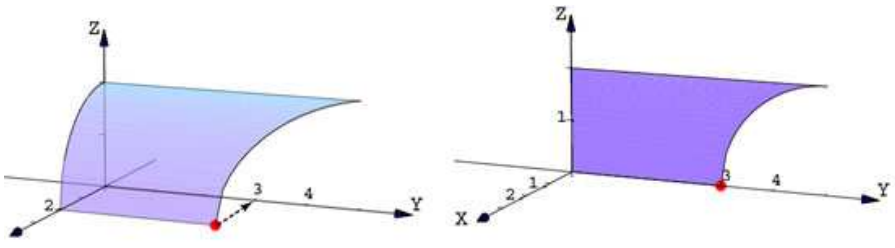

Figura 7.
[Ver con LiveGraphics3D]

Para proyectar la superficie, cada primera componente de cada objeto gráfico (excepto los Ejes) la vamos a multiplicar por la variable dependiente `ft -> x/2`, así conforme x varía de 2 a 0, la superficie va a sufrir un escalamiento que la va llevar desde su forma original (cuando `fx = 1`) hasta el plano YZ (cuando `fx = 0`).

El código es como sigue

```
Clear[x, y, z, n];
SetOptions[Arrow3D, HeadScaling3D -> 1, HeadLength3D -> 0.3];

independentVariables = {x -> 2};
dependentVariables = {fx -> x/2};


y[vx_] = 5 - vx;
z[vx_] = Sqrt[4 - vx^2];

vxmin = 0; vxmax = 2;
n = 70;
dvx = (vxmax - vxmin)/n;


Superficie = N[Table[
Polygon[{{ fx*i, 0, z[i]},
{fx* (i + dvx), 0, z[i + dvx]},
{fx* (i + dvx), y[i + dvx], z[i + dvx]},
{ fx*i, y[i], z[i]},
{ fx*i, 0, z[i]}
}],
{i, 0, 2 - dvx, dvx} ]];

contorno = {Line[Table[{fx*i, y[i], z[i]}, {i, 0, 2, 0.05}]],
Line[Table[{fx*i, 0, z[i]}, {i, 0, 2, 0.05}]],
Line[{{fx*2, 0, 0}, {fx*2, 3, 0}}],
Line[{{0, 0, 2}, {0, 5, 2}}]
};


punto = { AbsolutePointSize[6],
RGBColor[1, 0, 0],
Point[{x, 0, 0}],
Point[{x, 3, 0}]
};

g = Graphics3D[{
Ejes[-0.5, 3, -0.5, 6, -0.1, 3], punto,
EdgeForm[],
Superficie,
contorno
},
AmbientLight -> RGBColor[0, 0, 0.356863],
Boxed -> False, ViewPoint -> {2.281, 2.702, 0.572},
PlotRange -> All
];

Show[N[g] //. Join[independentVariables, dependentVariables]];

(* ----------------------Crear la página Web----------------------*)
WriteLiveForm["ejemplo11.m", g, Dir -> "C:\\MisLG3D\\"];
SetDirectory["C:\\MisLG3D\\"];
strm = OpenWrite["ejemplo11.html"];

(*código de la página Web*)
pagina = "<HTML><HEAD></HEAD> <BODY>
<APPLET height=400 width=400 archive=live.jar code=Live.class >
<PARAM NAME=INPUT_FILE VALUE = ejemplo11.m >
<PARAM NAME=INDEPENDENT_VARIABLES VALUE='{x -> 2}'>
<PARAM NAME=DEPENDENT_VARIABLES VALUE='{x -> If[x<0,0,If[x>2,2,x]],
fx->x/
2}'>
</APPLET>
</BODY></HTML>";

WriteString[strm, pagina];(*escribe en el archivo*)
Close[strm];
```

Imprimir el valor de una variable
Muchas veces, en el proceso de arrastre de un punto, es conveniente imprimir el valor de una variable o generar algún evento gráfico. Para imprimir el valor de *una variable independiente o dependiente* `var`, se usa `Text[var, a,b,c]`. Por ejemplo, para imprimir el valor de una variable y, con dos decimales, se usa

```
independentVariables = {y -> 1};
    dependentVariables = {valy -> Round[100*y]/100 };

  g = Graphics3D[{Text[valy, {0,y,0}]}];
```

Nota: Para poder imprimir el valor de una variable, ésta debe de estar en la lista de variables dependientes o independientes.

Eventos gráficos
En LG3D, algunos eventos gráficos se pueden implementar con un `If`. El uso es muy restringido, solamente se permiten expresiones como

```
If[ test , listas de primitivas y/o directivas]
```

Sin embargo, la lista ***no*** puede ser una lista invocada con `Table[]`.

Por ejemplo, podemos dibujar una línea con un color distinto dependiendo del valor de la variable

```
If[variable < 1, {RGBColor[1, 0, 0],Line[{{variable, -1, 0}, {0, 3, 0}}]},
       (*sino*) {RGBColor[0, 0, 1],Line[{{variable, -1, 0}, {0, 3, 0}}]}];
```

También, en vez de `Line[Table[{t, 0, 4 - t^2}, {t, 0, 1, 0.2}]]` (*que produce un error de sintaxis*), LG3D reconoce su equivalente extendido

```
independentVariables = {y -> 1}
ls = If[y < 3, Line[{{0, 0, 4}, {0.2, 0, 3.96}, {0.4, 0, 3.84},
                     {0.6, 0, 3.64}, {0.8, 0,3.36}, {1., 0, 3.}}]];
 g = Graphics3D[{ls}]
```

En el siguiente ejemplo hacemos una simulación un poco brusca, del corte de dos planos.

Ejemplo 7
Implementar un gráfico en el que se arrastra el plano $z = 2 - x$ hasta cortar el plano $y = 3$.

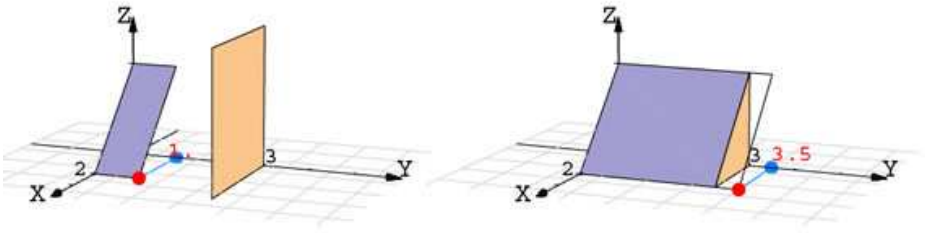

Figura 8.
[Ver con LiveGraphics3D]

Como el plano esta generado por la recta $z = 2 - x$ que esta en el plano XZ, la variable de arrastre debe ser y.

Una idea simple es implementar ambos gráficos con la coordenada y como variable de arrastre, de tal manera que si $y > 3$ se produzca el corte.

El plano $y = 3$ primero es un rectángulo, cuando el arrastre lleva la variable y hasta $y = 3$, se cambia a un triángulo.

```
independentVariables = {y -> 1};
planoY3 =
    If[y < 3,   {Polygon[{{3, 3, 0}, {0, 3, 0}, {0, 3, 3}, {3, 3, 3}, {3,3, 0}}]},
       (*sino*) {Polygon[{{2, 3, 0}, {0, 3, 0}, {0, 3,2}, {2, 3, 0}}]}
      ];
```

El plano $z = 2 - x$ primero es un rectángulo que depende del valor de y, cuando el arrastre lleva la variable y hasta $y = 3$, se fija el rectángulo y se sigue con unas líneas.

```
independentVariables = {y -> 1};
planoXZ =
    If[y <= 3, {Polygon[{{0, 0, 2}, {2, 0, 0}, {2, y, 0}, {0, y, 2}, {0,0, 2}}]},
      (*sino*) {Polygon[{{0, 0, 2}, {2, 0, 0}, {2, 3,0}, {0, 3, 2}, {0, 0, 2}}],
                Line[{{0, 3, 2}, {0, y, 2}, {2, y, 0}, {2, 3, 0}}]
               }
      ];
```

El código completo es

```
(*------------------------------  Gráfico  ------------------------------*)
(*Opciones para flechas 3D*)
  SetOptions[Arrow3D, HeadScaling3D -> 1, HeadLength3D -> 0.3];

  independentVariables = {y -> 1};
    dependentVariables = {valy -> Round[100*y]/100};

 planoXZ =
    If[y <= 3, {Polygon[{{0, 0, 2}, {2, 0, 0}, {2, y, 0}, {0, y, 2}, {0,0, 2}}]},
      (*sino*) {Polygon[{{0, 0, 2}, {2, 0, 0}, {2, 3,0}, {0, 3, 2}, {0, 0, 2}}],
                Line[{{0, 3, 2}, {0, y, 2}, {2, y, 0}, {2, 3, 0}}]
               }
       ];

 planoY3 =
    If[y < 3,  {Polygon[{{3, 3, 0}, {0, 3, 0}, {0, 3, 3}, {3, 3, 3}, {3,3, 0}}]},
      (*sino*) {Polygon[{{2, 3, 0}, {0, 3, 0}, {0, 3,2}, {2, 3, 0}}]}
      ];

 punto = { AbsolutePointSize[6],
           RGBColor[1, 0, 0],
           Point[{2, y, 0}],
           Text[ valy,  {0, y, 0}, {0, -1}],(*imprime valor de y*)
           RGBColor[0, 0.501961, 1],
           Point[{0, y, 0}],
           Line[{{2, y, 0}, {0, y, 0}}]
      };

 g = Graphics3D[{Ejes[-02.5, 4, -2.5, 6, -0.1, 3],
                 punto, planoXZ, planoY3},
                 Boxed -> False,
                 ViewPoint -> {3.640, 1.467, 0.793},
                 PlotRange -> All ];

Show[N[g] //. Join[independentVariables, dependentVariables]];


(*-----------------------------  página Web  -----------------------------*)
(* Generar la página Web 'ejemplo4.html'.                                 *)
(* El archivo ejemplo6.m  es el archivo de coordenadas                    *)

    WriteLiveForm["ejemplo6.m", g, Dir -> "C:\\MisLG3D\\"];
(* Crea la página Web *)
    SetDirectory["C:\\MisLG3D\\"];
    strm = OpenWrite["ejemplo6.html"];

(* código de la página Web*)
```

```
    pagina = "<HTML><HEAD></HEAD> <BODY>
    <APPLET height=150 width=200  archive=live.jar   code=Live.class >
    <PARAM NAME=INPUT_FILE VALUE = ejemplo6.m >
    <PARAM NAME=INDEPENDENT_VARIABLES VALUE='{y->1}'>
    <PARAM NAME=DEPENDENT_VARIABLES VALUE='{valy ->Round[100*y]/100}'>
    </APPLET>
    </BODY></HTML>";
(* escribir en el archivo*)
    WriteString[strm, pagina];
    Close[strm];
```

Ejemplo 8

Sea Q el sólido en el primer octante, limitado por la superficie $x^2+y^2=4$ y los planos $x+y=5$ y $z=2$.

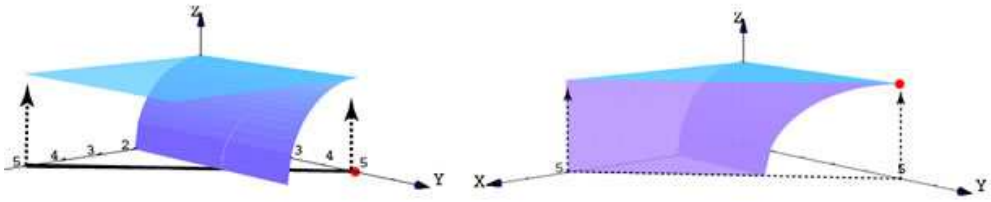

Figura 9.
[Ver con LiveGraphics3D]

Podemos hacer una pequeña animación, arrastrando un punto, simulando el corte (un poco brusco) del plano $x+y=5$ con la superficie y el otro plano. Para hacer tolerable la superposición inadecuada de superficies, todas las superficies las construimos con bastantes polígonos. Como estamos usando un 'If', todas las listas de polígonos y líneas se generaron aparte, con Table[], y luego se pegaron en el código (en el If).

Expresiones de Mathematica reconocidas por LG3D

Una expresión tal como

```
independentVariables = {y -> 1};
    dependentVariables = {valy -> Round[100*y]/100};
```

no nos va a causar un errorr de sintaxis, en cambio

```
independentVariables = {y -> 1};
    dependentVariables = {valy ->  GCD[36,45]};
```

si nos produce un error.
LG3D reconoce las operaciones básicas: +, -, /, *, >, <, <=, ==, ^, etc. y muchas otras funciones de *Mathematica*.

La programación está restringida a

`If[`*expr1*`,` *expr2*`]`
`If[`*expr1*`,` *expr2*`,` *expr3*`]`
`If[`*expr1*`,` *expr2*`,` *expr3*`,` *expr4*`]`
`Which[`*expr1*`,` *expr2*`,` ...`]`
`Switch[`*expr1*`,` *expr2*`,` *expr3*`,` ...`]`

A continuación veremos una lista de expresiones reconocidas por LG3D

Aritmética

`Plus[`*expr1*`,` *expr2*`,` ...`]`	*expr1* + *expr2* + ...

`Subtract[`*expr1*`, `*expr2*`, ...]`	*expr1* - *expr2* - ...
`Minus[`*expr*`]`	*-expr*
`Times[`*expr1*`, `*expr2*`, ...]`	*expr1 expr2* ...
`Subtract[`*expr1*`, `*expr2*`]`	*expr1* - *expr2*
`Divide[`*expr1*`, `*expr2*`]`	*expr1* / *expr2*
`Power[`*expr1*`, `*expr2*`, ...]`	*expr1* ^ *expr2* ^ ...

Constantes

`Pi`	3.14159265358979323846264
`E`	2.71828182845904523536028
`Degree`	0.01745329251994329576923
`GoldenRatio`	1.61803398874989484820458
`EulerGamma`	0.57721566490153286060651
`Catalan`	0.91596559417721901505460
`Khinchin`	2.68545200106530644530971
`Glaisher`	1.28242712910062263687534
`I`, `Infinity`, `Indeterminate`, `ComplexInfinity`	in LiveGraphics3D: NaN (not a number)

Funciones

`Abs[`*expr*`]`	
`Sign[`*expr*`]`	
`Round[`*expr*`]`	
`IntegerPart[`*expr*`]`	
`FractionalPart[`*expr*`]`	
`Floor[`*expr*`]`	
`Ceiling[`*expr*`]`	
`Chop[`*expr*`]`	
`Max[`*expr1*`, `*expr2*`, ...]`	
`Min[`*expr1*`, `*expr2*`, ...]`	
`Re[`*expr*`]`	in LiveGraphics3D: *expr*
`Im[`*expr*`]`	in LiveGraphics3D: 0
`Conjugate[`*expr*`]`	in LiveGraphics3D: *expr*
`Arg[`*expr*`]`	in LiveGraphics3D: 0 o `Pi`
`Mod[`*expr1*`, `*expr2*`]`	
`Quotient[`*expr1*`, `*expr2*`]`	

Random

`Random[]`	
`SeedRandom[]`	
`SeedRandom[`*expr*`]`	

Funciones elementales

`Log[`*expr*`]`	
`Log[`*expr1*`, `*expr2*`]`	

`Exp[`*expr*`]`

`Power[`*expr1*`, `*expr2*`, ...]` *expr1* ^ *expr2* ^ ...

`Sqrt[`*expr*`]`

`Sin[`*expr*`]`

`Cos[`*expr*`]`

`Tan[`*expr*`]`

`Csc[`*expr*`]`

`Sec[`*expr*`]`

`Cot[`*expr*`]`

`ArcSin[`*expr*`]`

`ArcCos[`*expr*`]`

`ArcTan[`*expr*`]`

`ArcTan[`*expr1*`, `*expr2*`]`

`ArcCsc[`*expr*`]`

`ArcSec[`*expr*`]`

`ArcCot[`*expr*`]`

`Sinh[`*expr*`]`

`Cosh[`*expr*`]`

`Tanh[`*expr*`]`

`Csch[`*expr*`]`

`Sech[`*expr*`]`

`Coth[`*expr*`]`

`ArcSinh[`*expr*`]`

`ArcCosh[`*expr*`]`

`ArcTanh[`*expr*`]`

`ArcCsch[`*expr*`]`

`ArcSech[`*expr*`]`

`ArcCoth[`*expr*`]`

Factorial

`Factorial[`*expr*`]` *expr*`!`

`Factorial2[`*expr*`]` *expr*`!!`

`Binomial[`*expr1*`, `*expr2*`]`

`Multinomial[`*expr1*`, `*expr2*`, ...]`

`Pochhammer[`*expr1*`, `*expr2*`]`

`Gamma[`*expr*`]`

`Beta[`*expr1*`, `*expr2*`]`

`LogGamma[`*expr*`]`

Teoría de números

`Mod[`*expr1*`, `*expr2*`]`

`PowerMod[`*expr1*`, `*expr2*`, `*expr3*`]`

`Quotient[`*expr1*`, `*expr2*`]`

Hipergeométricas

`Erf[`*expr*`]`

`Erf[`*expr1*`, `*expr2*`]`

`Erfc[`*expr*`]`

`Erfi[`*expr*`]`

`Gamma[`*expr*`]`

`Beta[`*expr1*`, `*expr2*`]`

Distribuciones

`DiscreteDelta[`*expr1*`, `*expr2*`, ...]`

`KroneckerDelta[`*expr1*`, `*expr2*`, ...]`

`UnitStep[`*expr1*`, `*expr2*`, ...]`

Programación > Tests

`Equal[`*expr1*`, `*expr2*`, ...]`	*expr1* == *expr2* == ...
`Unequal[`*expr1*`, `*expr2*`, ...]`	*expr1* `!=` *expr2* `!=` ...
`Less[`*expr1*`, `*expr2*`, ...]`	*expr1* < *expr2* < ...
`Greater[`*expr1*`, `*expr2*`, ...]`	*expr1* > *expr2* > ...
`LessEqual[`*expr1*`, `*expr2*`, ...]`	*expr1* <= *expr2* <= ...
`GreaterEqual[`*expr1*`, `*expr2*`, ...]`	*expr1* >= *expr2* >= ...
`NumberQ[`*expr*`]`	in LiveGraphics3D: `True` (`1.0`)
`NumericQ[`*expr*`]`	in LiveGraphics3D: `True` (`1.0`)
`IntegerQ[`*expr*`]`	
`EvenQ[`*expr*`]`	
`OddQ[`*expr*`]`	
`Positive[`*expr*`]`	
`Negative[`*expr*`]`	
`NonPositive[`*expr*`]`	
`NonNegative[`*expr*`]`	
`TrueQ[`*expr*`]`	
`ValueQ[`*expr*`]`	in LiveGraphics3D: `True` (`1.0`)

Programación > Operadores lógicos

`Not[`*expr*`]`	`!`*expr*
`And[`*expr1*`, `*expr2*`, ...]`	*expr1* `&&` *expr2* `&&` ...
`Or[`*expr1*`, `*expr2*`, ...]`	*expr1* `||` *expr2* `||` ...
`Xor[`*expr1*`, `*expr2*`, ...]`	
`Implies[`*expr1*`, `*expr2*`]`	
`True`	in LiveGraphics3D: 1.0
`False`	in LiveGraphics3D: 0.0

Programación > Control de evaluación

`Evaluate[`*expr*`]`	in LiveGraphics3D: *expr*
`Hold[`*expr*`]`	in LiveGraphics3D: *expr*
`HoldComplete[`*expr*`]`	in LiveGraphics3D: *expr*
`HoldForm[`*expr*`]`	in LiveGraphics3D: *expr*

`ReleaseHold[`*expr*`]`	in LiveGraphics3D: *expr*

Input and Output

`StandardForm[`*expr*`]`	in LiveGraphics3D: *expr*
`TraditionalForm[`*expr*`]`	in LiveGraphics3D: *expr*
`InputForm[`*expr*`]`	in LiveGraphics3D: *expr*
`OutputForm[`*expr*`]`	in LiveGraphics3D: *expr*
`DisplayForm[`*expr*`]`	in LiveGraphics3D: *expr*
`FullForm[`*expr*`]`	in LiveGraphics3D: *expr*
`NumberForm[`*expr*`]`	in LiveGraphics3D: *expr*
`ScientificForm[`*expr*`]`	in LiveGraphics3D: *expr*
`EngineeringForm[`*expr*`]`	in LiveGraphics3D: *expr*
`PaddedForm[`*expr*`]`	in LiveGraphics3D: *expr*

Interpretación geométrica de algunos conceptos del cálculo en varias variables

Vector tangente

En esta sección vamos a implementar el vector tangente a una curva. De aquí se puede saltar de manera sencilla a otros conceptos como derivada direccional y plano tangente.

Recordemos que si un curva suave C está parametrizada por $r(t) = (x(t), y(t), z(t)) \text{ con } t \in [a, b]$, el vector tangente en $P = r(t_0)$ es $r'(t_0) = (x'(t_0), y'(t_0), z'(t_0))$.

La implementación de la curva C podría ser

```
curvaC = Line[Table[{x(t),y(t),z(t)}, {t, a, b, 0.1}]];
```

mientras que el vector tangente lo podemos implementar usando `Arrow3D[]` de la biblioteca `DrawGraphics`

```
independentVariables = {t -> t0};
P  = {x(t) , y(t)  , z(t) };
vT = {x'(t), y'(t) , z'(t)};
vTT=Arrow3D[P,P + vT];
g=Graphics3D[{Point[P], curvaC, vvT}];
```

Ejemplo 9

Graficar la curva $C: (x, 4 - x, 4 - (x - 1)^2)$ con $x \in [0, 3]$ y el vector tangente a C en un punto de arrastre $P \in C$

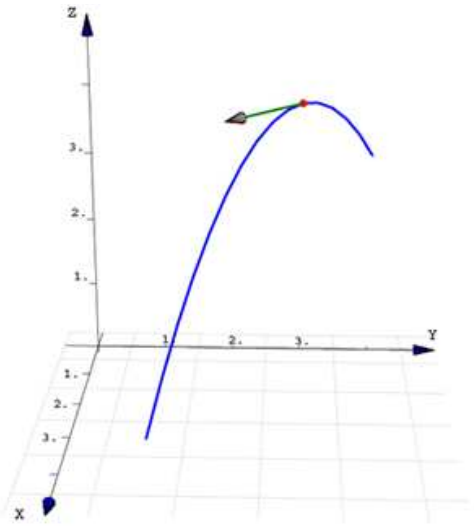

Figura 10.
[Ver con LIveGraphics3D]

```
(*------------------------------  Gráfico ------------------------------*)
  SetOptions[Arrow3D, HeadScaling3D-> 1, HeadLength3D -> 0.3];
  independentVariables = {x -> 1, y ->1};
  dependentVariables = {};

  P  = {x, 4 - x, 4 - (x - 1)^2};  (*puntos de la curva*)
  vT = {1,-1, -2(x - 1)};          (*vector tangente   *)
  curva =Line[Table[{t, 4 - t, 4 - (t - 1)^2}, {t, 0, 3, 0.2}]];

  vTT = Arrow3D[P, P + vT];
 punto = { AbsolutePointSize[6],
           RGBColor[1, 0, 0],
           Point[P]
      };

g = Graphics3D[{
                Ejes[-0.5, 5, -0.5, 5, -0.1, 5],
                punto, RGBColor[0, 0, 1],
                AbsoluteThickness[2],
                RGBColor[0.0588235, 0.52549, 0.0705882],
                vTT,
                RGBColor[0, 0, 1],
                curva},
                Boxed -> False,
                ViewPoint -> {3.096, 0.100, 1.095},
                PlotRange -> All
             ];

Show[N[g] //. Join[independentVariables, dependentVariables]];

(*-----------------------------  página Web  -----------------------------*)
(* Generar la página Web 'ejemplo7.html'.                                 *)
(* El archivo ejemplo7.m  es el archivo de coordenadas                    *)

  WriteLiveForm["ejemplo7.m",g,Dir\[Rule]"C:\\MisLG3D\\"];
  SetDirectory["C:\\MisLG3D\\"];
  strm=OpenWrite["ejemplo7.html"];

(*código de la página Web*) pagina="<HTML><HEAD></HEAD> <BODY>
    <APPLET height=400 width=400  archive=live.jar   code=Live.class >
    <PARAM NAME=INPUT_FILE VALUE = ejemplo7.m >
    <PARAM NAME=INDEPENDENT_VARIABLES VALUE='{x->1}'>
    <PARAM NAME=DEPENDENT_VARIABLES VALUE='{x->If[x<0,0,If[x>3,3,x]]}'>
    </APPLET>
    </BODY></HTML>";

WriteString[strm,pagina];
Close[strm];
```

Derivada direccional y plano tangente

Para la implementación de la interpretación geométrica de la derivada direccional, necesitamos una superficie suave S, un punto $P \in S$ y un vector $\vec{u}$. Usamos el hecho (geométrico) de que la recta tengente a S en P, en la dirección de $\vec{u} = (u_1, u_2) \in \mathbb{R}^2$, es la recta tangente en P, a la curva C de intersección entre el plano generado por la recta $L: (x, y, z) = P + t(u_1, u_2, 0)$ y la superficie S. Por tanto podemos usar el código del ejemplo anterior.

Para el caso del plano tangente a S en $P \in S$, observamos que este plano Π tiene ecuación vectorial $\Pi: (x, y, z) = P + t\vec{v} + s\vec{w}$. Podemos escoger $\vec{v}$ como el vector tangente a S en P, en la dirección del eje X, o sea en la dirección de $\vec{u} = (1, 0)$ y $\vec{w}$ como el vector tangente a S en P, en la dirección del eje Y, o sea en la dirección de $\vec{u} = (0, 1)$

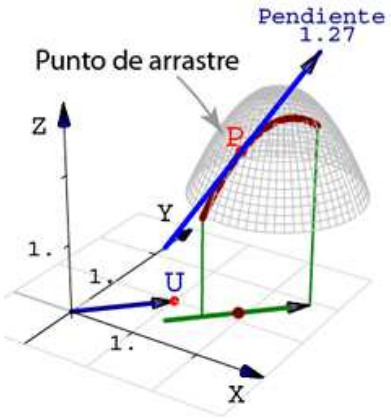

Figura 11.
[Ver con LiveGraphics3D]

Figura 12.
[Ver con LiveGraphics3D]

Conclusión

A través de los ejemplos se han presentado las ideas básicas de cómo usar LiveGraphics3D para generar gráficos parametrizados. Los parámetros se declaran como variables independientes y su valor cambia arrastrando puntos que los contienen en sus coordenadas (de manera limpia). Las posibilidades de programación están limitadas por las expresiones que reconoce LG3D que, en particular, no admite constructores de listas ni control de flujos. Aún así se pueden crear gráficos parametrizados de gran complejidad.

Bibliografía

1 Martin, Kraus. LiveGraphics3D. Consultada 24 mar. 2005. Disponible en http://wwwvis.informatik.uni-stuttgart.de/~k /LiveGraphics3D/index.html

2 Pita R. C. 1995. Cálculo Vectorial. Prentice-Hall. México, MX, 1078 p.

3 Davis, B.1994. Calculus and Mathematica. Addison-Wesley. New York, US, 330p.

4 Schneider, Ph. Eberly, D. 2003. Geometric Tools for Computers Graphics. Ed. Morgan Kaufmann. San Francisco California. 1007

5 Wolfram Research. Consultado 20 de mayo 2005. Disponible en http://www.wolfram.com

6 LiveGraphics3d Home. Consultado 20 de mayo de 2005. Disponible en http://www.vis.informatik.uni-stuttgart.de/ kraus/LiveGraphics3D/

7 Arrow3D. Consultado 20 de mayo 2005. Disponible en http://library.wolfram.com/infocenter/TechNotes/4117/

8 DrawGraphics. Consultado 20 de mayo 2005. Disponible en http://home.earthlink.net/ djmp/Mathematica.html