

Tipos recursivos y E/S: Superficies

Curso 2002/03, Fecha:22/11/2002

Enunciado 1 (ElevationGrid) *El lenguaje VRML (y X3D) incluye una primitiva que permite construir superficies bidimensionales indicando los puntos de elevación. Almacenar el siguiente código VRML en un fichero y visualizarlo.*

```
#VRML V2.0 utf8

Shape {
  geometry ElevationGrid {
    creaseAngle 2.5
    height [ 0 0 0 0 0 0
            0 0 1 1 0 0
            0 1 2 2 1 0
            0 1 2 2 1 0
            0 0 1 1 0 0
            0 0 0 0 0 0 ]
    solid FALSE
    xDimension 6
    zDimension 6
  }
  appearance Appearance {
    material Material { diffuseColor 0 1 0 }
  }
}
```

Enunciado 2 (Plot2D) *El siguiente programa permite representar superficies bidimensionales. Las superficies se representan como funciones de tipo **Float** → **Float** → **Float***

```
> module Pf602 where
> type Superficie = Float → Float → Float
> w sup = writeFile "fun.wrl"
>         (cabecera ++
>          viewPoint (40,30,95) ++
>          background ++
```

```

> plotS (0,0,0) (0,1,0) sup)
> cabecera = "#VRML_V2.0_utf8\n\n"
> sx, sy :: Float
> sx = 64; sy = 64
> plotS p c = translate p . color c . verS
> viewPoint p = "Viewpoint_{_position_}" ++ sh3 p ++ "}\n"
> background =
> "Background_{_groundAngle_[_1.309,_1.571_]}\n" ++
> "groundColor_[_0.1_0.1_0,_0.4_0.2_0.2,_0.6_0.6_0.6_]}\n" ++
> "skyAngle_[_1.309,_1.571_]}\n" ++
> "skyColor_[_0_0.2_0.7,_0_0.5_1,_1_1_1_]_}"
> verS f =
> "_geometry_ElevationGrid_{_}\n" ++
> "creaseAngle_1.57\n" ++
> "height_{_}\n" ++ puntos f ++ "]\n" ++
> "solid_FALSE\n" ++
> "xDimension_" ++ show (round sx) ++
> "zDimension_" ++ show (round sy) ++ "}"
> puntos f = concat [linea y f ++ "\n" | y <- [1..sy]]
> linea y f = concat [show (f x y) ++ "_" | x <- [1..sx]]
> translate (x,y,z) s =
> "Transform_{_translation_}" ++ sh3 (x,y,z) ++
> "\n_children_[_]" ++ s ++ "}"
> color (r,g,b) s = "Shape_{_}\n" ++ s ++
> "\n_appearance_Appearance_{_material_Material_{_}" ++
> "\n_diffuseColor_" ++ sh3 (r,g,b) ++ "}_}\n"
> sh3 (x,y,z) = show x ++ "_" ++ show y ++ "_" ++ show z ++ "_"
> s1 x y = sin (x + y)

```

Enunciado 3 (circulo) Construir una función *circulo* tal que al evaluar *circulo* (x,y) r h s genera una superficie insertando un círculo en la posición (x,y) de radio r y altura h a partir de la superficie s

En <http://www.di.uniovi.es/labra/PLF/prac/worlds/circulo.wrl> puede observarse el resultado de evaluar w (*circulo* (32,32) 10 3 s1) Tipo:

```
> circulo :: (Float, Float) -> Float -> Float ->
```

> *Superficie* → *Superficie*

Enunciado 4 (Texto) *Añadir el siguiente fragmento que permite generar un mensaje de texto*

```
> wt txt = writeFile "text.wrl"
>         (cabecera ++
>         viewPoint (0,0,20) ++
>         background ++
>         putText (0,0,0) (0,0.1,0.1) txt ++
>         putBox (4,-2,-0.2) (0.8,1,1) (10,10,0.1))
>
> putText p c = translate p . color c . verText
> putBox p c = translate p . color c . verBox
>
> verText t = "geometry_Text_{_string_}" ++ t ++ "\}"
>   where l = length t
> verBox (sx, sy, sz) =
>   "geometry_Box_{_size_}" ++ sh3 (sx, sy, sz) ++ "\}"
```

Enunciado 5 (leerTxt) *El siguiente programa lee un mensaje de un fichero y lo muestra por pantalla al en mayúsculas*

```
> leeTxt = do
>         cs ← readFile "f.txt"
>         putStrLn ("Mensaje=" ++ map toUpper cs)
```

Modificar el programa anterior para que escriba el mensaje leído en un fichero VRML

Enunciado 6 (w2) *Construir una función w2 que represente 2 superficies utilizando colores diferentes.*

*En <http://www.di.uniovi.es/labra/PLF/prac/worlds/sup2.wrl> puede observarse el resultado de evaluar w2 s1 ($\backslash x y \rightarrow \sin x * \cos y$)*

Enunciado 7 (QuadTrees) *El siguiente fragmento define un tipo de datos que representa quadrees con elevaciones. Estos quadrees representan en cada cuadrante la elevación correspondiente. La función q2s convierte un quadtree en una superficie y puede utilizarse para visualizarlos.*

```
> data QTE = Plano Float
>         | Div QTE QTE QTE QTE
>
> q2s :: QTE → (Float → Float → Float)
> q2s qt = q2s' qt ((0,0), d) (\x y → 0)
>
> q2s' (Plano h) ((x,y), d) f =
```

```

> cuadro ((x,y),(x+d,y+d)) h f
> q2s' (Div q1 q2 q3 q4) ((x,y),d) f =
>   if d <= 0 then f
>   else
>     let d2 = d `div` 2
>     in (q2s' q1 ((x,y),d2) .
>         q2s' q2 ((x+d2,y),d2) .
>         q2s' q3 ((x,y+d2),d2) .
>         q2s' q4 ((x+d2,y+d2),d2)) f

> cuadro (p1,p2) h f =
>   \x y → if dentro (x,y) p1 p2
>           then h
>           else f x y

> dentro (x,y) (x1,y1) (x2,y2) =
>   x >= fromInt x1 && x <= fromInt x2 &&
>   y >= fromInt y1 && y <= fromInt y2

> d :: Int
> d = round sx

> e1 = Div (Plano 0) (Plano 5) (Plano (-5)) (Plano 10)

```

Enunciado 8 (Galeria) Construir una galería de mundos virtuales. La galería puede escribirse directamente en HTML utilizando la siguiente plantilla o generarse automáticamente mediante un programa Haskell. Esta última opción sería preferible. Todos los mundos deben incluir una sencilla explicación de cómo se han generado.

Una posible galería podría se incluye en <http://www.di.uniovi.es/labra/-PLF/prac/worlds/galeria1.html>

```

<html><body>
<ol>
<li><p>Superficie (sin(x+y))</p>
  <embed src="superficie.wrl"
        height="500" width="500" />
</li>
<li><p>Usando función circulo</p>
  <embed src="circulo.wrl"
        height="500" width="500" />
</li>
</ol>
</body>
</html>

```

Enunciado 9 (MasQuadtrees(Opcional)) *Definir otros tipos de superficies mediante el método anterior*

- *Estudiar la generación de superficies infinitas*
- *Desarrollar una librería con combinadores básicos como los desarrollados en la práctica 3 (rotaciones, repeticiones, esquinas, etc.)*

Enunciado 10 (MasVRML(Opcional)) *Definir una librería con funciones VRML de propósito general. Aumentar las posibilidades de visualización (incluir ejes de coordenadas, enlaces a páginas Web, rotaciones, texturas, etc.*