CC-52B Computación Gráfica

Patricio.Inostroza@dcc.uchile.cl
Departamento de Ciencias de la Computación
Universidad de Chile

VRML

- 1994: Primera conferencia de World Wide Web
  - Nace la idea de una interfaz 3D para Internet
  - Inicialmente un concepto denominado *Virtual Reality Markup Language* o simplemente VRML
- Silicon Graphics pone a disposición su lenguaje de representación de objetos 3D: Open Inventor
- VRML 1.0 se apoya en Open Inventor, con las diferencia que nacen de una conexión Web

VRML

- VRML 1.0 se apoya en Open Inventor, con las diferencia que nacen de una conexión Web
- 1995:
  - Son publicadas las especificaciones definitivas de VRML 1.0
  - Pasa a denominarse *Virtual Reality Modeling Language*
    - Realza su carácter gráfico

VRML

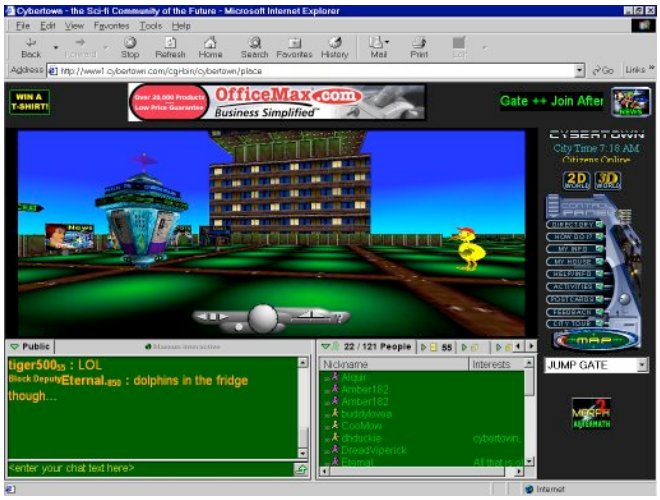
- VRML 1.0
  - Permite definir la apariencia de un mundo 3D
- Pero...
  - No es posible definir comportamiento entre los objetos
  - No hay niebla
  - ...
- Distintas empresas comienzan a usar sus propios formatos
  - Es necesario estandarizar!

VRML

- 1996 se realiza un llamado y seis sociedades presentan sus formatos
  - Se opta por el de Silicon graphics
  - VRML 2.0 es presentado en Siggraph'96
- 1997
  - VRML Architecture Group pasa a ser VRML Consortium que luego deriva a Web3D Consortium
  - www.web3d.org
- Diciembre 1997
  - VRML 2.0 es norma ISO
  - También se conoce como VRML'97

VRML

- Cybertown



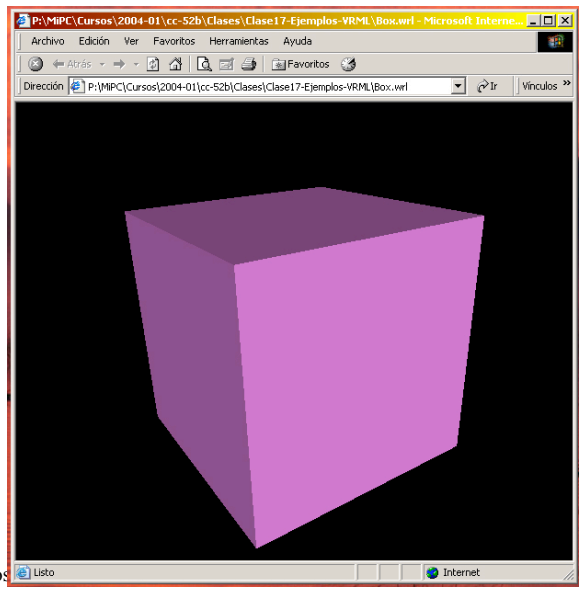
VRML: Browser

- Browser VRML:
  - Navegador que interpreta el contenido de un archivo VRML
- Visualiza el contenido
- Permite la interacción del usuario con el contenido.
- Puede ser una aplicación o un *plugin* para un navegador Web
- Algunos browser:
  - contact (www.blaxxun.com)
  - cosmo (www.cai.com/cosmo/)
  - cortona (www.parallelgraphics.com)
- Extensión de los archivos: .wrl ➜ world

VRML: Ejemplo básico

- Una caja centrada en (0, 0, 0)

```
#VRML V2.0 utf8
Box {}
```



VRML: Nodos geométricos básicos

- Nodos geométricos predefinidos:

```
Box { size 2 2 2}

Cone {
    bottomRadius 1
    height 2
}
```

```
Sphere { radius 1}

Cylinder {
    height 2
    radius 1
}
```


VRML: Nodos geométricos básicos

- El tamaño de estos nodos puede ser modificado:

```
Cylinder {
    height 3.1
    radius 0.2
}

Cylinder {
    height 2.4
}
```

VRML: Campos de una variable

- La declaración de las variables de un nodo contienen cuatro campos:
  - Ejemplo: Box { field SFVec3f size 2 2 2}

- El primer campo indica el alcance de la variable
- El segundo campo indica el tipo de la variable
- El tercer campo es el nombre de la variable
- El cuarto campo contiene el valor de la variable

VRML: Alcance de la variable

Ejemplo: Box { field SFVec3f size 2 2 2}

- Una variable esta definida con uno de los cuatro tipos de alcances:
  - field : la variable no puede ser modificada
  - eventIn: la variable puede ser modificada, pero no se puede conocer el valor de esta
  - eventOut: se puede conocer el valor de la variable, pero esta no puede ser modificada
  - exposeField: el valor puede ser modificado y se puede conocer su valor

- Observación: Las modificaciones se realizan mediante el envió/recepción de eventos

VRML: Alcance de la variable

- Ejemplos:

  Box { field SFVec3f size 2 2 2}

  ➜ una vez definido el tamaño, no es posible cambiarlo.

  ImageTexture {
  exposedField MFString url []
  field SBool repeatS TRUE
  field SFBool repeatT TRUE
  }

  ➜ La url con la dirección de la imagen puede ser conocida y modificada.

VRML: Tipo de la variable

Ejemplo: Box { field SFVec3f size 2 2 2}

- Cada variable es de tipo simple o bien múltiple
- Ejemplo:
  SFFloat y MFFloat

- SF : single field, la variable es simple
- MF: multiple field, la variable es un arreglo
- Float: el tipo

VRML: Tipo de la variable

- SFBool: un valor 'booleano' (TRUE o FALSE)
- MFColor: una lista de SFColor
- SFColor: contiene tres float (en el rango [0, 1]) que representan el rojo, el verde y el azul

- Ejemplo:

```
Material {
    diffuseColor 0.5 0.5 0.5
...}
```


VRML: Tipo de la variable

- SFImage:
  - Define una imagen 2D en pixeles
  - Es usada como textura.
  - Formato: <width> <height> <num components> <pixels values>

- Por ejemplo:

  2 2 1 0xFF 0x00 0xFF 0xFF
- corresponde a una imagen de 2x2 con una componente por color (tonalidades de gris)

VRML: Tipo de la variable

- SFInt32: corresponde a un valor entero de 32 bits.
- MFInt32: arreglo de ‘n’ enteros (n=0,...)

- Ejemplos:

```
Switch {
    whichChoice 2
...}
IndexedLineSet {
    coordIndex [ 0 1 3 –1 3 4 5 7 -1]
...}
```


VRML: Tipo de la variable

- MFRotation: una lista de SFRotation
- SFRotation: un campo con cuatro componentes
  - los 3 ejes de rotación y el ángulo en radianes

- Ejemplo: una rotación en el eje X en 180^{o}

```
Transform {
    rotation 1 0 0 3.1415
...}
```

VRML: Tipo de la variable

- SFString: un string
- MFString: una lista de SFString

- Ejemplo:

```
WorldInfo {
    title “my first world”
    info [“DCC”, “U.Chile”]
}
```


VRML: Tipo de la variable

- SFTime: float que contiene el número de segundos
- que han pasado desde el 1 enero de 1970.
- MFTime: una lista de SFTime

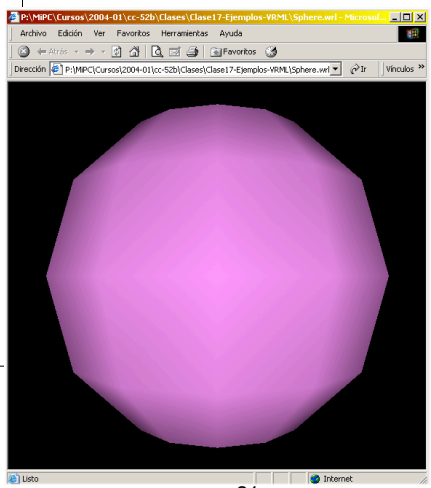
- SFVec3f: un vector con tres float
- MFVec3f: una lista de SFVec3f

- Ejemplo:

```
Box { size 2.1 0.4 3.7}
```

VRML:Ejemplo

```
#VRML V2.0 utf8
Box {}
    Transform {
        translation 1 0 0
            children [
                Sphere {radius 1.2}
        ]
    }
```



VRML: Nodos geométricos

- Nodos de geometría
  - Describen la forma de un objeto
  - Primitivas:
    - Box{...} Cone{...} Cylinder{...} Sphere{...} Text{...}
  - Nodos creados en base a puntos:
    - ElevationGrid{...} Extrusion{...} IndexedFaceSet {...} IndexedLineSet{...} PointSet{...}

VRML: PointSet

PointSet: define un conjunto de puntos

```
PointSet {
    SFNode color # los colores de los puntos
    SFNode coord # las coordenadas de los puntos
}
```

Ejemplo:

```
Shape {
    geometry PointSet {
        coord Coordinate { point [ 0 0 0, 0 1 0, ...]
        }
    color Color { color [...] }
    }
}
```


VRML: IndexedLineSet

IndexedLineSet: nodo que define una lista de líneas

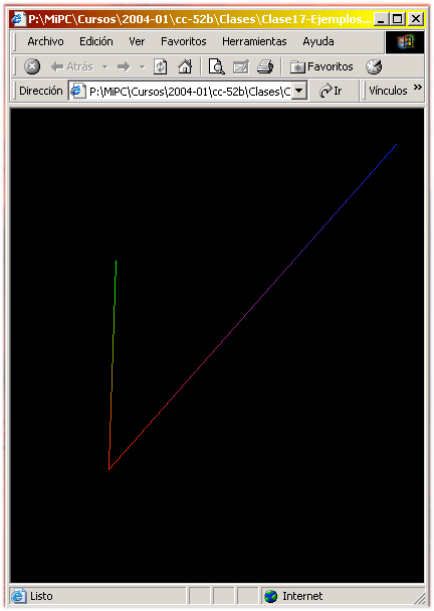
```
IndexedLineSet {
    SFNode color # los colores de los puntos
    SFNode coord # las coordenadas de los puntos
    MFInt32 coorIndex # las listas de puntos, separadas por -1
    ...
}
```

Ejemplo:

```
IndexedLineSet {
    coord Coordinate {
    point [ 0 0 0, 0 1 0, ...]
    }
    coordIndex [ 0 1 –1 1 2 –1 ... ]
    color Color { color [...] }
}
```

VRML: IndexedLineSet

```
IndexedLineSet {
      coord Coordinate {
      point [ 0 0 0, 0 1 0, ...]
      }
      coordIndex [ 0 1 –1 1 2 –
      1 ... ]
      color Color { color [...] }
}
```



VRML: IndexedFaceSet

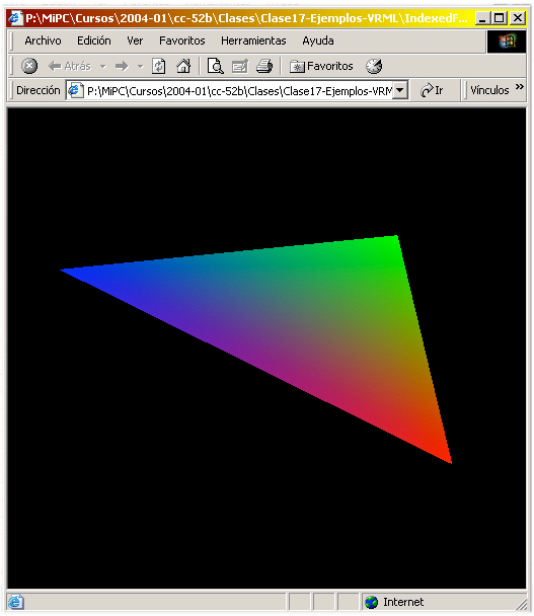
IndexedFaceSet: nodo que define una lista de caras

```
IndexedFaceSet {
    SFNode color # los colores de los puntos
    SFNode coord # las coordenadas de los
      puntos
    MFInt32 coorIndex # las listas de puntos,
      separadas por -1
    ...
}
```

```
IndexedFaceSet{
      coord Coordinate {
      point [ 0 0 0, 0 1 0, ...]
      }
      coordIndex [ 0 1 –1 1 2 –1 ... ]
      color Color { color [...] }
}
```

VRML: IndexedFaceSet

```
IndexedFaceSet{
      coord Coordinate {
      point [ 0 0 0, 0 1 0, ...]
      }
      coordIndex [ 0 1 –1 1 2 –1 ... ]
      color Color { color [...] }
}
```



VRML: Shape

- Shape: nodo contiene la apariencia y la geometría de un objeto:

```
Shape {
    geometry     # nodo que define una geometría
    appearance # nodo que define la apariencia de
                 # la geometría (debe ser de tipo
      Appearance)
}
```

#: hasta el fin de la línea es un comentario

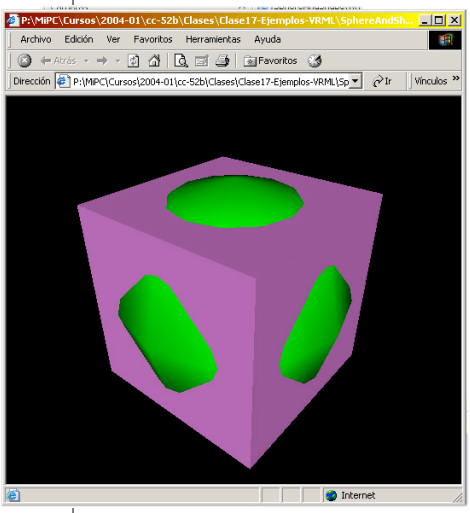
VRML: Apariencia

- Nodo Appearance
  - Este nodo describe el color o textura:

```
Appearance{
    SFNode material # un nodo de tipo Material
    SFNode texture # un nodo de tipo textura
    SFNode textureTransform # un nodo de tipo
      TextureTransform;
                          # que define la transformación
                          # aplicada a la textura
}
```


VRML: Ejemplo de Shape+Material

```
#VRML V2.0 utf8
Box {}
    Shape {
    geometry Sphere { radius 1.2}
    appearance Appearance {
        material Material {
            diffuseColor 0 1 0
        }
    }
}
```



VRML: DEF/USE

DEF: permite dar un nombre a un nodo para su posterior utilización mediante USE

```
#VRML V2.0 utf8
DEF MyBox Box { }
Transform {
      translation 1 0 0
      children [
            Shape {
                  geometry USE MyBox
                  appearance Appearance {
                        material Material { diffuseColor 0 1 0 }
                  }
            }
      ]
}
```

PROTO

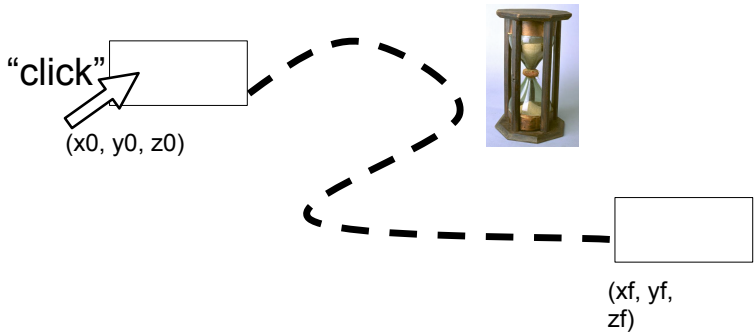
- PROTO: Interfaz que permite definir nuevos nodos creados a partir de nodos ya conocidos

```
PROTO myBox [
      field SFColor theColor 1 0 0
]
{
      Shape {
            Appearance Appearance {
                  material Material {
                        diffuseColor IS theColor
                  }
                  geometry Box {}
            }
      }
}
```

```
Transform {
      translation 1 1 0
      children [
            myBox{ theColor 1 0 1}
      ]
}
```

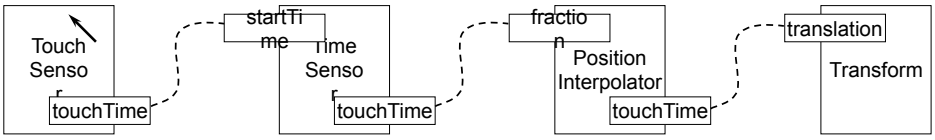

Comportamientos

- Cómo programar el desplazamiento de un objeto dentro de una trayectoria y en un tiempo dado ?



Comportamientos

- El usuario selecciona el objeto (“click”)
- El sensor (*touchTime*) activa el sensor de tiempo
- El reloj (*TimeSensor*) comienza a generar valores a intervalos regulares desde un T= t0 hasta un T= tf
- Por cada valor enviado al interpolador de posición, se genera una nueva posición
- Esta nuevo valor de posición modifica la variable *translation* de un nodo Transform, lo que permite realizar el movimiento



Comportamientos

- El paso de los valores entre los sensores se realiza mediante la creación de una ruta (*ROUTE*)
- Los tipos de datos de los eventos conectados con *ROUTE* deben coincidir

- Recordemos que un tipo de dato comienza con:
  - *SF*: se asume un sólo valor *(Single Field)*
  - *MF*: se asume una lista de valores *(Multiple Field)*
- Cada variable posee un alcance de tipo *field*, *eventIn*, *eventOut* o *exposedField*

Comportamientos: TimeSensor

- Una animación requiere un control del tiempo
- Un nodo *TimeSensor* es similar a un cronómetro

```
TimeSensor {
exposedField SFTime cycleInterval 1.0
exposedField SFTime startTime 0
exposedField SFTime stopTime 0
exposedField SFBool loop FALSE
exposedField SFBool enabled TRUE
}
```

- *startTime* y *stopTime* : cuando comenzar y terminar
- *enabled :* está habilitado o deshabilitado
- *cycleInterval* : segundos de cada ciclo

Comportamientos: PositionInterpolator

```
PositionInterpolator {
    eventIn SFFloat set_fraction
    exposedField MFFloat key
    exposedField MFVec3f keyValue
    eventOut SFVec3f value_changed
    }
```

- *set_fraction* (o simplemente *fraction*): la fracción del valor que debe generar
- *value_changed* (o simplemente *value*): el valor generado
- *key*: fracción de distribución de los rangos; valores entre 0.0 y 1.0
- *keyValue*: los rangos de valores a generar; uno por cada *key*

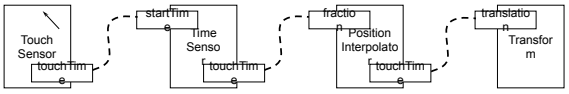
VRML: PositionInterpolator

- Ejemplo:

```
PositionInterpolator {
    key [0.0 0.2 1.0]
    keyValue [0 0 0, 1 1 1, 0 0 0]
}
```

- Genera los valores de posición desde (0, 0, 0) hasta (1, 1, 1), en sólo un 20% del tiempo total
- Luego, genera el desplazamiento de retorno desde (1, 1, 1) a (0, 0, 0), en el resto del tiempo
- Si *fraction* recibe un valor 0.6, *value* tomará el valor (0.5 0.5 0.5)

Comportamientos



Ejemplo:

```
Sphere {}
DEF myTransform Transform {
    translation1 0 0
    children [
        DEF myTouch TouchSensor {}
        Box {}
    ]
}
DEF myTime TimeSensor { cycleInterval 5 }
DEF myPosition PositionInterpolator {
    key [0 0.2 1]
    keyValue [0 0 0, 1 1 1, 0 0 0]
}
ROUTE myTouch.touchTime TO myTime.startTime
ROUTE myTime.fraction TO myPosition.fraction
ROUTE myPosition.value TO myTransform.translation
```

Comportamientos

- Diferentes tipos de nodos permiten llevar a cabo una animación
- Los nodos sensores permiten captar eventos del ambiente:
  - *Collision, ProximitySensor, TimeSensor, VisibilitySensor LOD, Billboard*
- del *pointing device*:
  - *Anchor, TouchSensor, CylinderSensor, PlaneSensor, SphereSensor*
- Los nodos interpoladores permiten interpolar características de un nodo:
  - *ColorInterpolator, CoordinateInterpolator, NormalInterpolator, OrientationInterpolator, PositionInterpolator, ScalarInterpolator*
- Los nodos dependientes del tiempo:
  - *TimeSensor, AudioClip, MovieTexture*

Propuestos

- Semáforo: dibuje tres esferas de colores distintos, seleccionando una (*click*) permita que el color resalte, dando un color gris a las otras dos esferas.

- Dibuje una esfera y permita que gire al momento de ser seleccionada
- Idem, pero además debe desplazarse a otro lugar pero sin resbalar
- Idem, pero debe ir y volver

- Dibuje una objeto. Modificando la transparencia permita que este desaparezca y aparezca.

- ¿Qué es EAI (External Authoring Interface) ?

Bibliografía

- Bibliografía
  - Web3D Consortium. Portal con las especificaciones de VRML. http://www.web3d.org/
  - Varios tutoriales en línea, por ejemplo:
    - http://dmi.uib.es/~abasolo/cursovrml/indice.htm
  - Late Night, VRML 2.0 with Java. Bernie Toehl al. Ziff-Davis Press, 1997.