OGRE 3D

Particle systems

http://www.ogre3d.org

Ana Gil Luezas
Departamento de Sistemas Informáticos y Computación
Universidad Complutense de Madrid

Particle systems

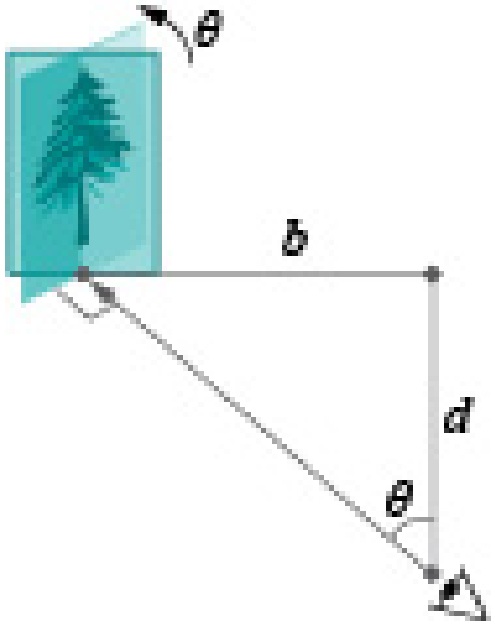
- Son sistemas dinámicos basados en una colección de elementos móviles (**particles**) generados por un **emitter**. Su comportamiento físico (SIM: Simulación física para videojuegos) lo determinan parámetros como la velocidad, la dirección, el tiempo de vida
- Durante el tiempo de vida pueden (es opcional) sufrir cambios mediante **affectors** (viento, gravedad, ...)
- Representación gráfica (renderización). En Ogre, por defecto, con **billboards** (carteles o vallas publicitarias)
- Se pueden crear y configurar completamente en código, pero es habitual tenerlos en scripts (archivos de extensión **.particle**) que se pueden generar con un editor gráfico

Billboards

- Billboard: carteles o vallas publicitarias

Panel (rectángulo) que se orienta hacia la cámara (cada frame).

En Ogre, por motivos de eficiencia, se gestionan en grupos: **BillboardSet**



Permiten simular varios efectos: texto, botones, vegetales (impostores), sistemas de partículas (humo, estelas, ...)

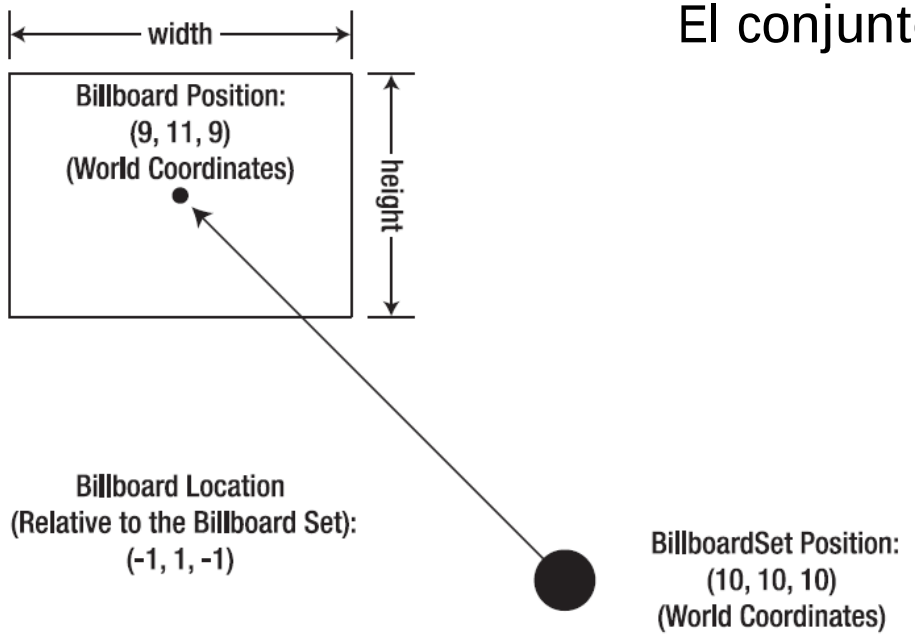
Billboard sets

- La clase **BillboardSet** hereda de **MovableObject** y **Renderable**

  Todos los billboards de un grupo (BillboardSet) tienen que tener el mismo tamaño y material.

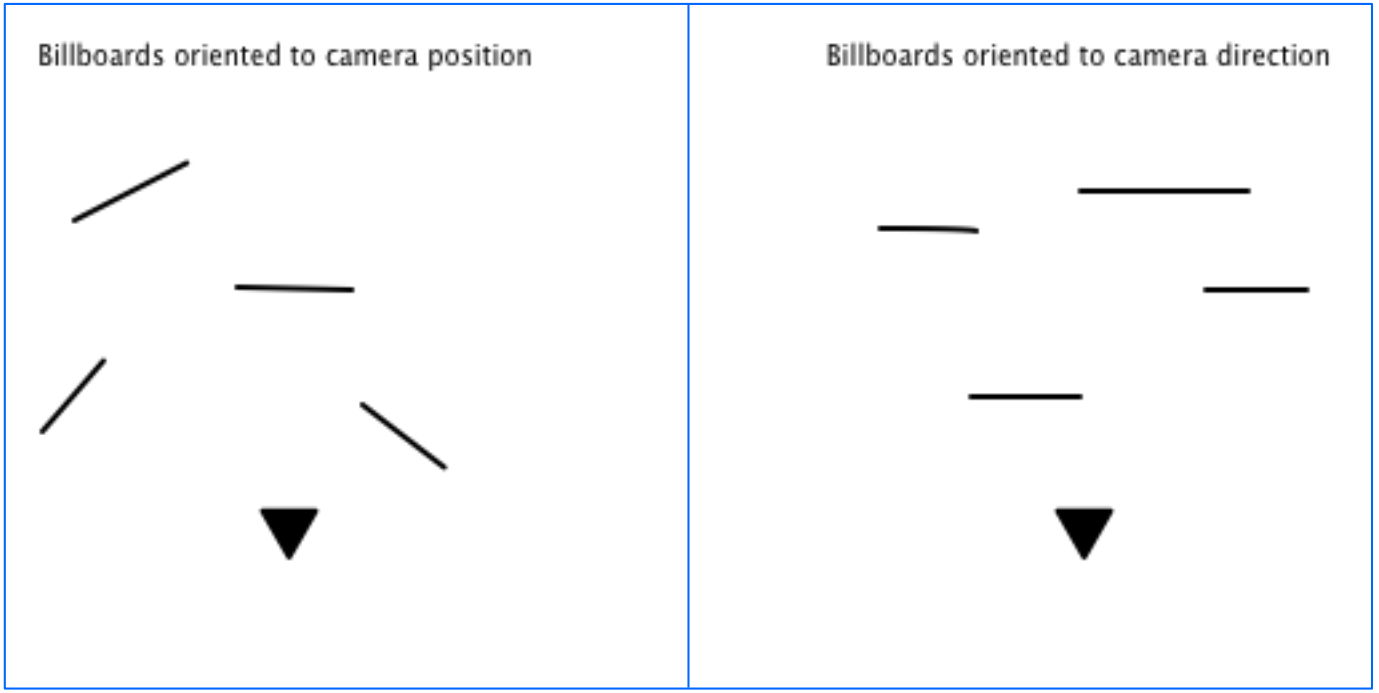
  Las posiciones de cada billboard son relativas a la posición del conjunto.

  El conjunto se trata como un único objeto.



Billboards sets

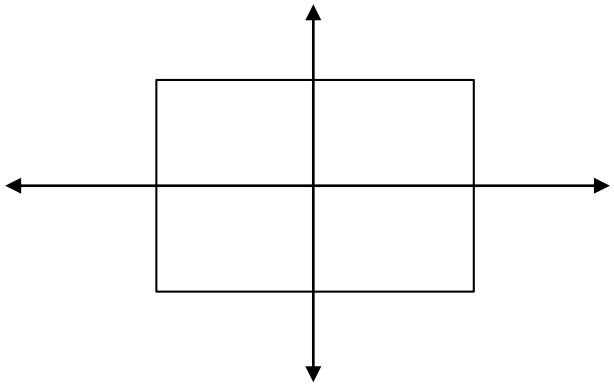
- Se pueden orientar de distintas formas:

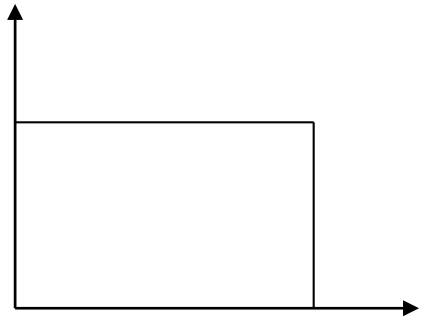


Billboard sets

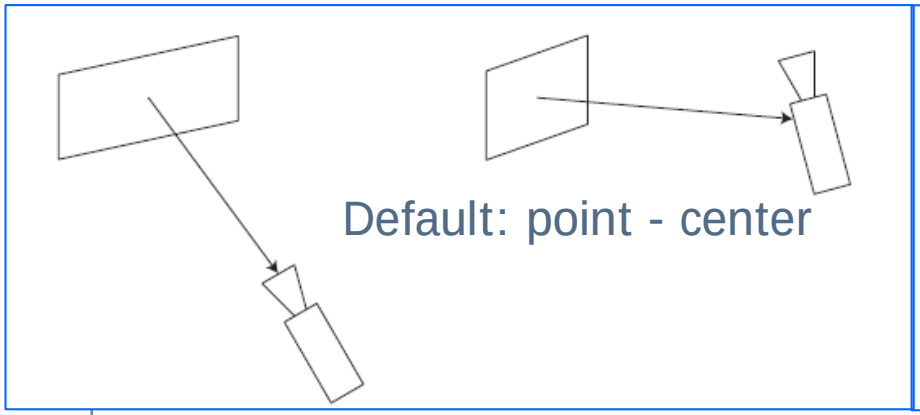
- Podemos configurar el sistema local, la forma de orientarse

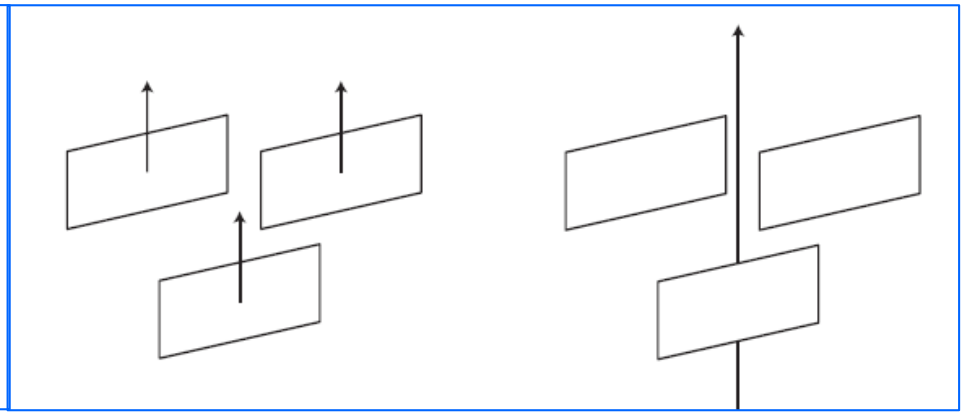
billboard_origin center | bottom_left | ...





billboard_type: point | oriented_self | oriented_common

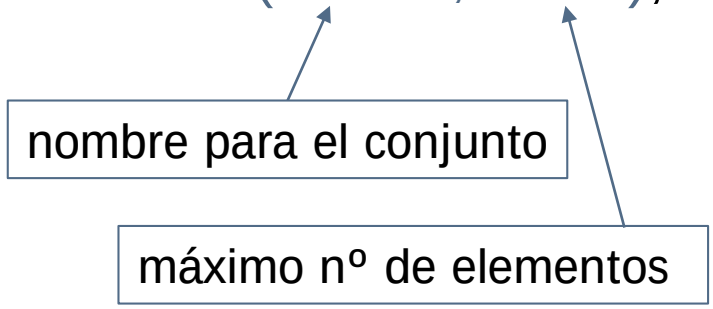




Billboard sets

- **Crear el conjunto**: establecer el nombre, el número de elementos, dimensiones, material, ...

```
BillboardSet* bbSet = mSM-> createBillboardSet(NameBS, MaxEls);
bbSet -> setDefaultDimensions(w, h);
bbSet -> setMaterialName(...);
...
// Colocar el BillboardSet en el grafo de la escena
node -> attachObject(bbSet);
...  // la posición del nodo será la del BillboardSet
// Crear los elementos del conjunto: establecer su posición, ...
Billboard* bb = bbSet -> createBillboard(Vector3(x, y, z));
                                    // posición relativa al grupo (nodo)
...
```



Script del material

```
material IG2App/Panel
{
    technique
    {
        pass
        {
            lighting off
            texture_unit
            {
                texture "..."
                tex_address_mode clamp
            }
        }
    }
}
```

Particle systems

- La clase **ParticleSystem** hereda de **MovableObject**

  Necesita un emisor y un renderizador

- Ejemplo. Crear un sistema a partir de un script (archivo .particle)

```
ParticleSystem* pSys = mSM ->
                        createParticleSystem("psSmoke", "IG2App/Smoke");

pSys -> setEmitting(false);

mPSNode -> attachObject(pSys);
```

nombre para el sistema

nombre de la entrada en el script .particle

La posición y la dirección de emisión son relativas al nodo.

Ejemplo: script de partículas (en un archivo .particle)

```
particle_system  IG2App/Smoke
{  // p. s. attributes
   renderer              billboard
   billboard_type        point
   particle_width        35
   particle_height       35
   quota                 500
   material              IG2App/Smoke

   emitter Point
   {  // e. attributes
      direction          0 1 0
      angle              35
      emission_rate      15
      time_to_live       4
      velocity_min       50
      velocity_max       80
   }
```

```
   affector ColourImage
   {  // a. attributes
      image  smokecolors.png
   }
   affector Rotator // the texture
   {  // a. attributes
      rotation_range_start          0
      rotation_range_end            360
      rotation_speed_range_start    -60
      rotation_speed_range_end      200
   }
   affector Scaler
   {  // a. attributes
      rate       50
   }
   affector DirectionRandomiser
   {  //  a. attributes
      randomness 5
   }
}
```

Ejemplo: script del material

```
material IG2App/Smoke
{
    technique
    {
        pass
        {
            lighting off
            scene_blend alpha_blend
            depth_write off
            diffuse vertexcolour

            texture_unit
            {
                texture smoke.png
                tex_address_mode clamp
            }
        }
    }
}
```

Particle systems

- **Atributos** o características (del renderizado) del sistema de partículas (**.particle**)

✓ renderer: sistema de renderizado (de la clase ParticleSystemRenderer)

Default: billboard (otros en plugins)

✓ Cada clase de renderizador puede tener sus propios atributos de configuración

✓ Atributos del renderizador basado en billboards:

- material entrada de un .material
  Default: none (blank material)
- particle_width, particle_height: del rectángulo
  Default: 100
- quota: máximo número de partículas que se muestran a la vez
  Default: 10

- **Atributos** o características (del renderizado) del sistema de partículas (continuación)
  - billboard_type <point|oriented_common|oriented_self |...>: indica cómo se orientan los billboards. El defecto es que apunten a la cámara

    Default: point
  - billboard_origin <top_left|top_center|top_right|center_left|center |...>: indica dónde se sitúa el origen del sistema local

    Default: center
  - billboard_rotation_type <vertex | texcoord>: indica si rotan los vértices o lo hacen las coordenadas de textura

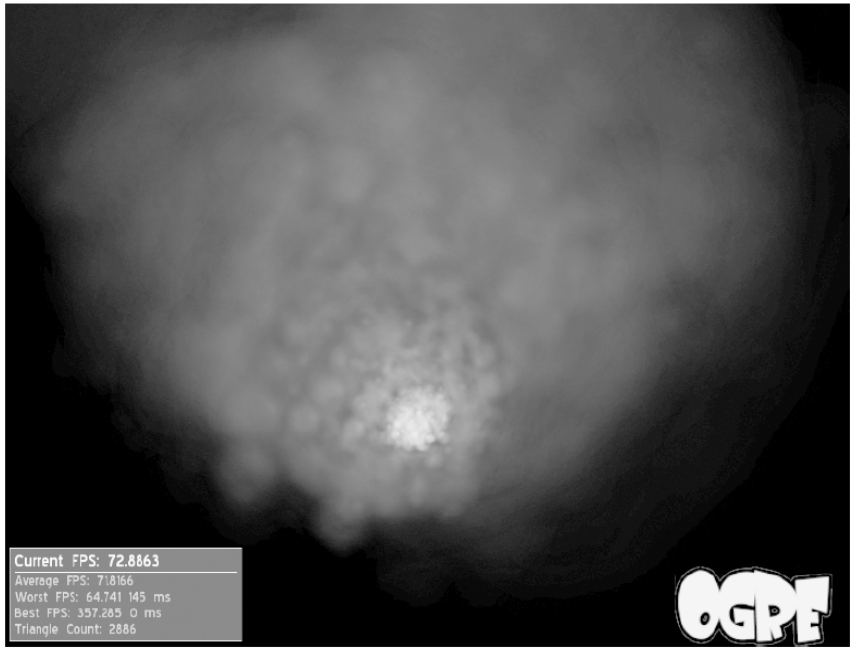
    Default: texcoord
  - cull_each <true|false>: comprueba si cada partícula de forma individual está en el frustum (simulación de lluvia, por ejemplo) y se hace el culling de cada partícula, o está en el frustum el conjunto (bounding box) de las partículas

    Default: false
  - Sorted <true|false>: las partículas se ordenan con respecto a la cámara

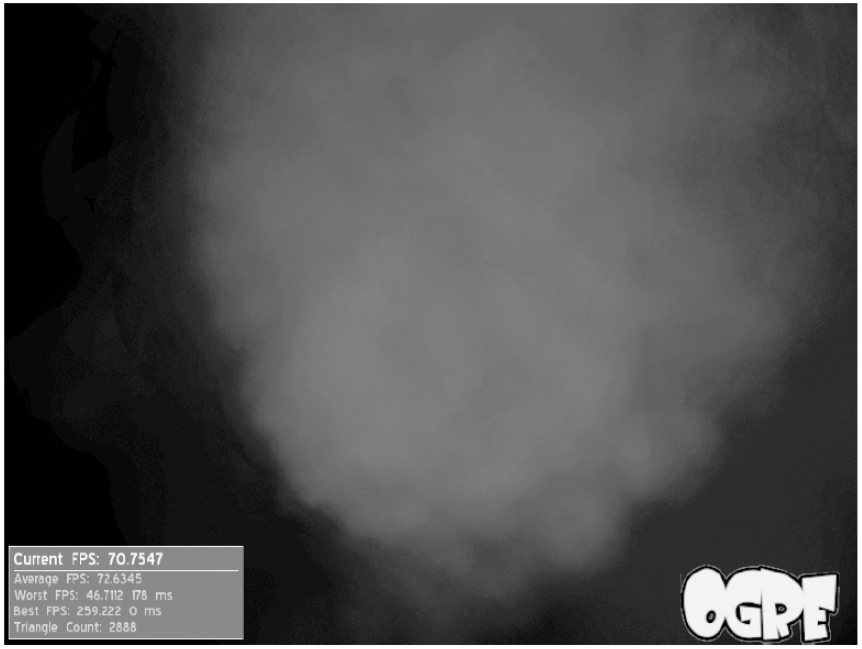
    Default: false

Particle systems

- Las partículas se pueden ordenar con respecto a la distancia a la cámara



Sorting disabled



Sorting enabled

- **Particle Emitters**: Point, Ring, Cylinder, Ellipsoid, Box, ...

  Salvo para Point, hay que especificar las dimensiones

  Attributes:

  position: default 0 0 0. Relativa al nodo

  direction: default 1 0 0. Relativa al nodo

  angle (degrees): default 0. Indica un cono en la dirección

  emission_rate (particles/second): default 10

  time_to_live (seconds): default 10. Desde que es emitida

  duration (seconds): default 0 (infinito). El tiempo de actividad

  velocity (world_units_per_second): default 1

  ...

https://ogrecave.github.io/ogre/api/latest/_scripts.html

- **Particle Affectors**:

  Linear Force (gravedad, viento): modifica la trayectoria usando un vector de fuerza

  ColourFader (per second): modifica el color usando un color de incremento

  ColourImage, ColourInterpolator

  Scaler (per second)

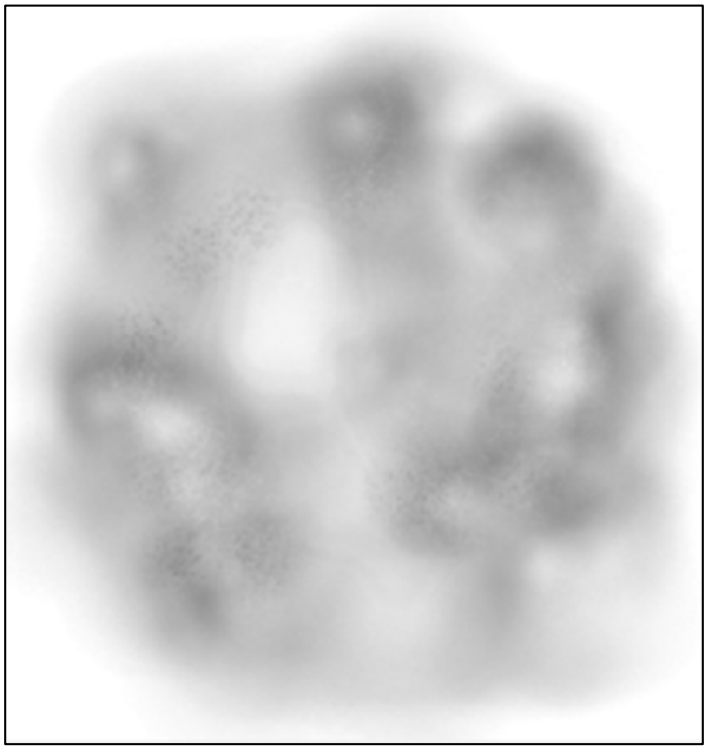
  Rotator (the texture)

  DirectionRandomiser

  ...

https://ogrecave.github.io/ogre/api/latest/_scripts.html

Affector ColourImage:



Textura 2d para las partículas (RGBA): smoke.png -> dada en el material

Se combinará el color del téxel de la textura 2d del material con un color de la textura 1d (array de colores dado en el affector), en función del tiempo de vida de la partícula.

Textura 1d con los colores: smokecolors.png

Ejemplo. Explosión de la boya

- Se quiere hacer explotar la boya poco después de que muera Sinbad y solo si esto ocurre
  - Crear un sistema de partículas en la constructora de la clase **Boya**

```
pSysBoya = mSM->createParticleSystem("parSysBombBoya",
                                     "IG2App/Smoke");
```

  - Crear un nodo y adjuntar el sistema a ese nodo

```
mPSNode = mNode->createChildSceneNode();
mPSNode->attachObject(pSysBoya);
```

  - Hacer que el sistema permanezca inactivo de momento

```
pSysBoya->setEmitting(false);
```

  - Definir las características del sistema de partículas de la boya en una nueva entrada del archivo **.particle**
  - Definir el método **receiveEvent()** de forma que el sistema de partículas explota si el evento lo envía Sinbad

Ribbon trails

- Permiten definir estelas que dejan los objetos al moverse.

  BillboardChain: Análogo a un BillboardSet, pero los elementos forman cadenas. Hay que actualizarlos manualmente.
  RibbonTrail subclase de BillboardChain: implementa el posicionamiento de los elementos de la cadena siguiendo a un objeto.