

Informática Gráfica Animación y simulación

Juan Carlos Torres
Grupos C y D

Dpt. Lenguajes y Sistemas Informáticos
ETSI Informática y de Telecomunicación
Universidad de Granada

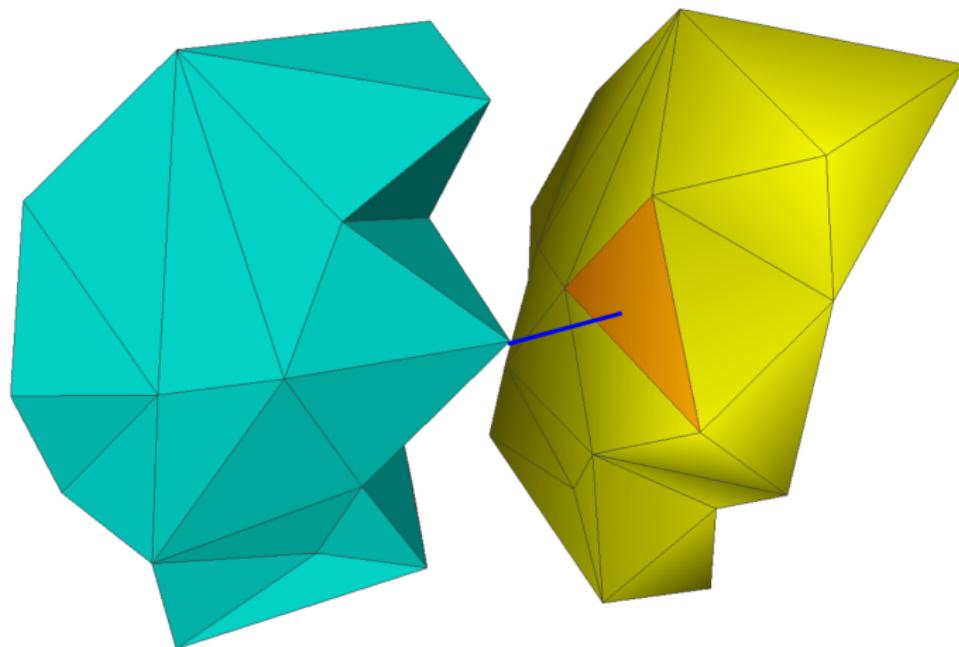
Curso 2024-25

Detección de Colisiones

Detección de colisiones

La detección de colisiones es un problema geométrico:

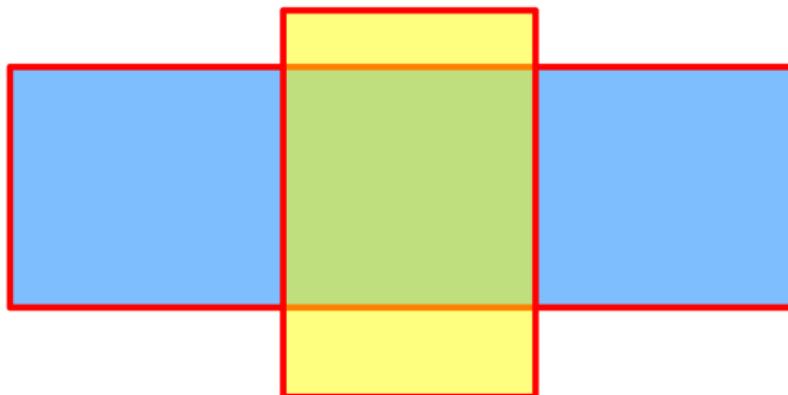
Dados dos objetos determinar si hay contacto (o solapamiento) entre ellos.
Si hay colisión es necesario determinar el punto de contacto.



Detección de colisiones

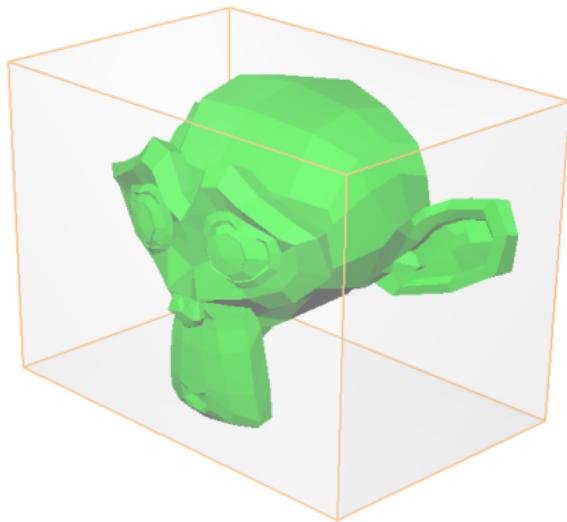
Algoritmos

- Se debe calcular la intersección entre cada par de objetos.
- El problema tiene complejidad $O(n^2)$.
- La complejidad del cálculo de las intersecciones depende de la geometría de los objetos.
- Es conveniente utilizar técnicas para descartar pares que no se intersecan (índices espaciales y volúmenes envolventes).



Volúmenes envolventes

Un volumen envolvente es un elemento geométrico simple que contiene al objeto.

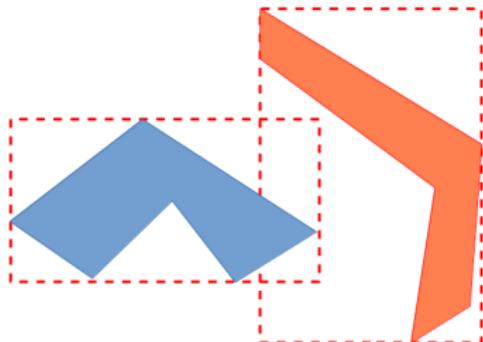
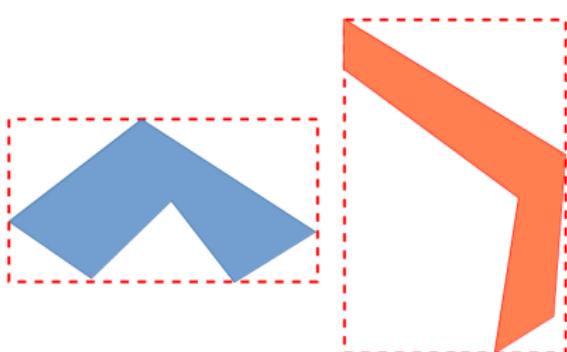


Volúmenes envolventes para detección de colisiones

Para realizar el test de forma mas eficiente se asocia a cada objeto un volumen envolvente (p.e. un paralelepípedo).

Se realiza un test de intersección entre volúmenes envolventes (cajas). Si las cajas no se intersectan los objetos tampoco.

Si los volúmenes se intersectan puede que los objetos no lo hagan.



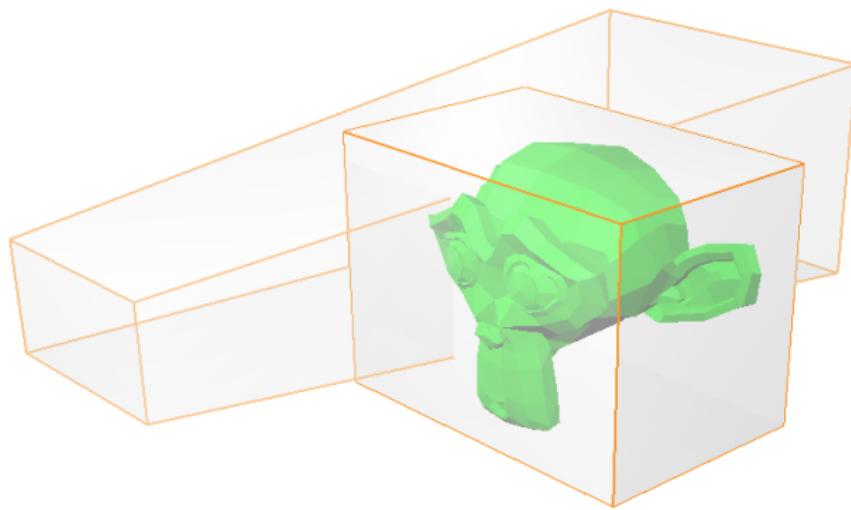
Volúmenes envolventes

Otras aplicaciones: Eliminación de partes no visibles

Determinar si la envolvente es visible

Si la envolvente es visible

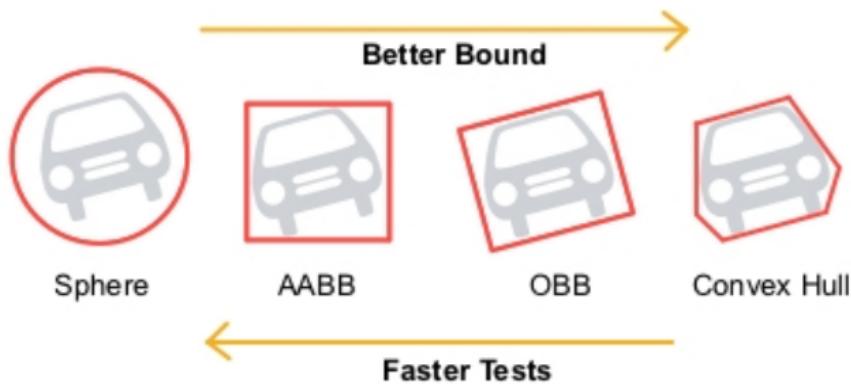
Visualizar componente



Volúmenes envolventes

La efectividad dependerá de lo ajustado que sea el volumen
El coste de cálculo, almacenamiento y filtrado también.

Bounding Volumes

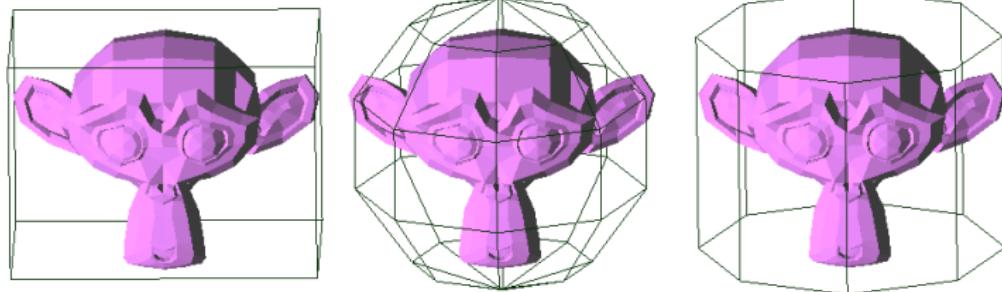


copyright haroldsserrano.com

<https://www.haroldsserrano.com/blog/tips-for-developing-a-collision-detection-system>

Volúmenes envolventes

Como volúmenes envolventes se suelen usar paralelepípedos (con orientación arbitraria o alineados con los ejes), esferas, cápsulas o poliedros convexos.



La eficacia del volumen envolvente depende de tres factores:

- Facilidad de construcción.
- Simplicidad del cálculo de intersecciones.
- Ajuste a la forma del objeto.

Cálculo de caja envolvente alineada (AABB)

El cálculo de la caja envolvente alineada con los ejes (Axis Aligned Bounding Box) se realiza obteniendo el máximo y mínimo de cada coordenada. La caja tiene vértices extremos en $(X_{min}, Y_{min}, Z_{min})$ y $(X_{max}, Y_{max}, Z_{max})$.

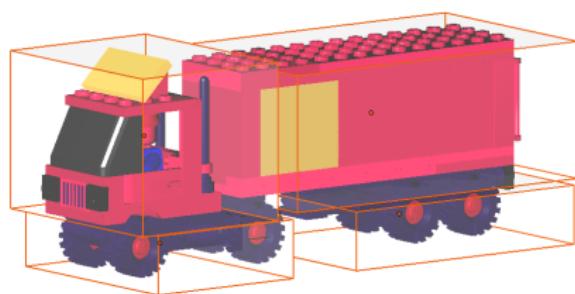
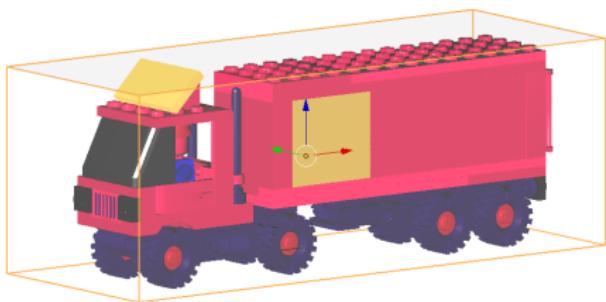
```
...
Bounding.Xmax= v[0]->x;
Bounding.Xmin= v[0]->x;
Bounding.Ymax= v[0]->y;
Bounding.Ymin= v[0]->y;
Bounding.Zmax= v[0]->z;
Bounding.Zmin= v[0]->z;
for (j = 1; j < num_elems; j++) {
    if(v[j]->x > Bounding.Xmax) Bounding.Xmax=v[j]->x;
    else if(v[j]->x < Bounding.Xmin) Bounding.Xmin=v[j]->x;

    if(v[j]->y > Bounding.Ymax) Bounding.Ymax=v[j]->y;
    else if(v[j]->y < Bounding.Ymin) Bounding.Ymin=v[j]->y;

    if(v[j]->z > Bounding.Zmax) Bounding.Zmax=v[j]->z;
    else if(v[j]->z < Bounding.Zmin) Bounding.Zmin=v[j]->z;
}
```

Jerarquía de volúmenes envolventes

- En un modelo jerárquico cada nodo de la jerarquía puede tener su volumen envolvente.
- Cuando el test con el último nivel sea positivo será necesario realizar el test de colisión con las primitivas geométricas para determinar la colisión con precisión.

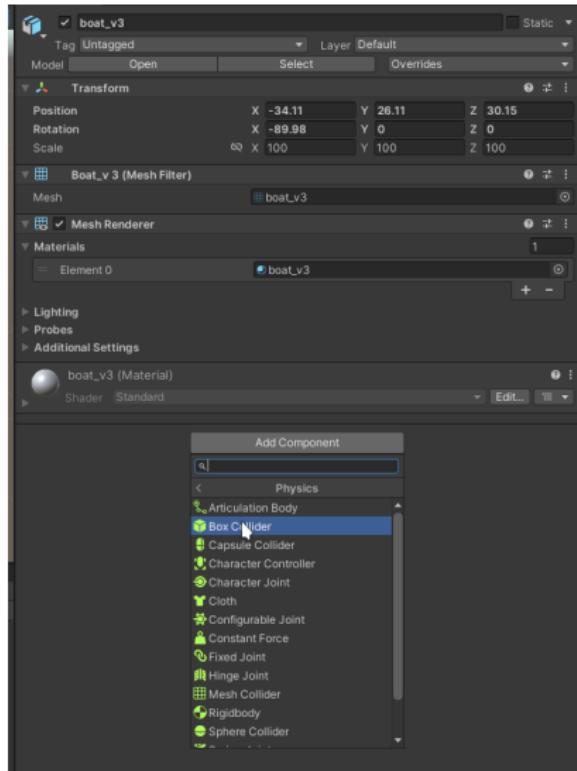


Volúmenes envolventes en Unity

Los volúmenes envolventes están representados por los componentes *Collider* (en *Addcomponet > Physics*):

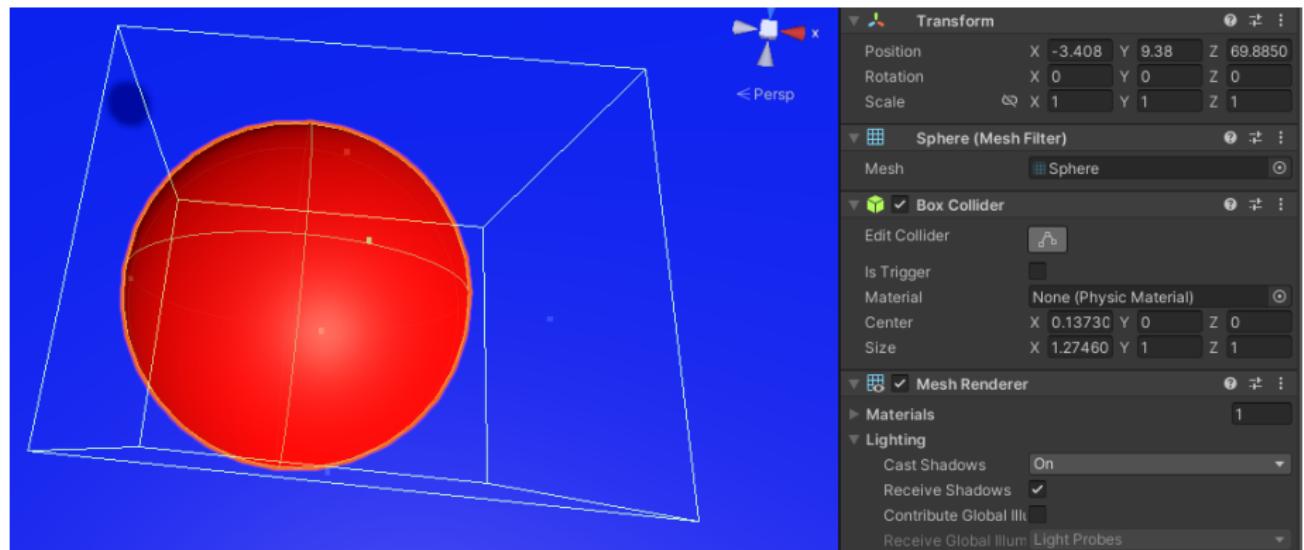
- Box Collider
- Capsule Collider
- Mesh Collider
- Sphere Collider
- Terrain Collider

Unity solo calcula colisiones con el *Collider*. Solamente se calcula colisión con la geometría si se usa un *MeshCollider*.



Collider

Podemos cambiar el tipo de envolvente y editar la envolvente.

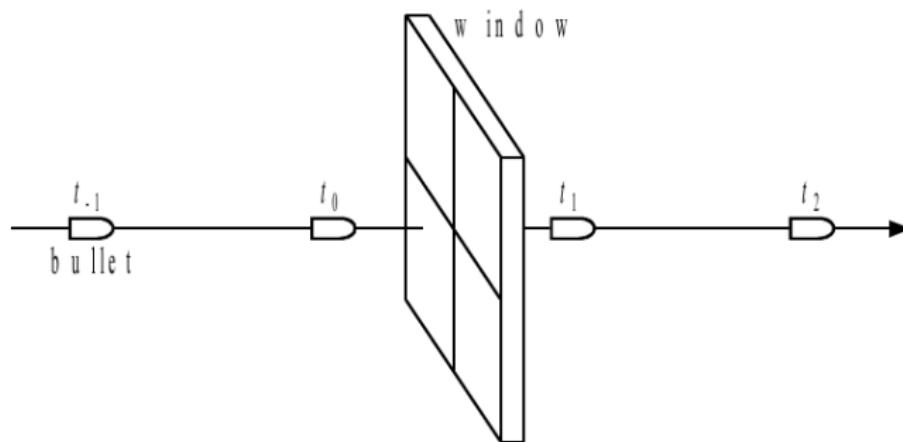


Colisiones en sistemas dinámicos

Escenas dinámicas

Si al menos uno de los objetos está en movimiento, no es suficiente con calcular las colisiones al final de cada paso de iteración.

Determinar las colisiones al final de cada paso puede hacer que no se detecten colisiones si el tiempo de integración es alto.

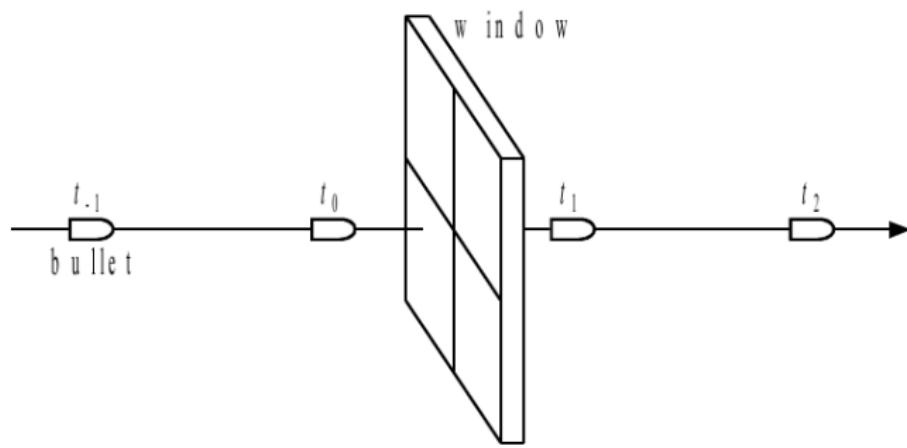


Colisiones en sistemas dinámicos

Es necesario calcular intersecciones entre las trayectorias de los objetos.

Si hay colisión se debe determinar

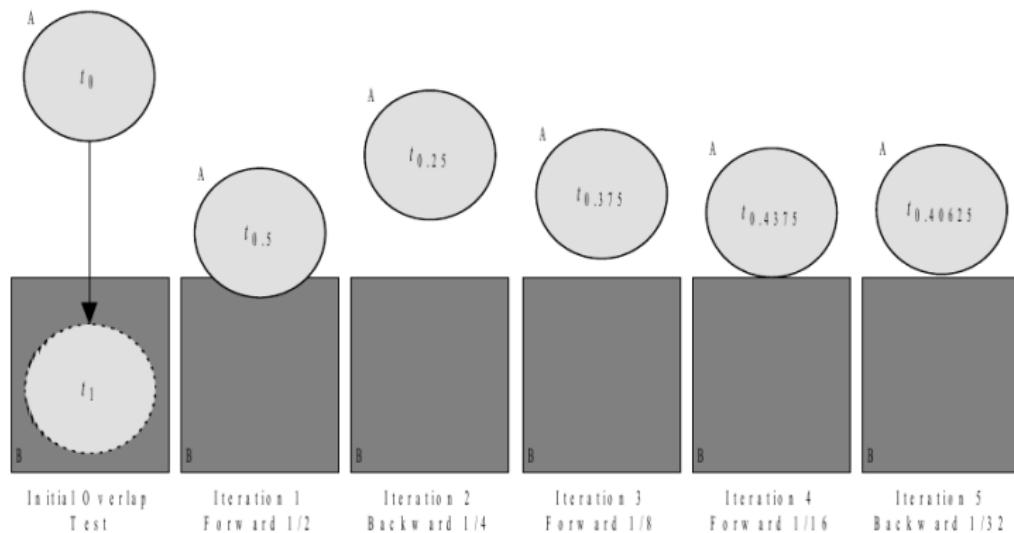
- Punto de contacto
- Tiempo de colisión



Colisiones en sistemas dinámicos

Cálculo del tiempo de colisión

Se puede usar biseción del intervalo de tiempo para aproximar el tiempo de colisión.



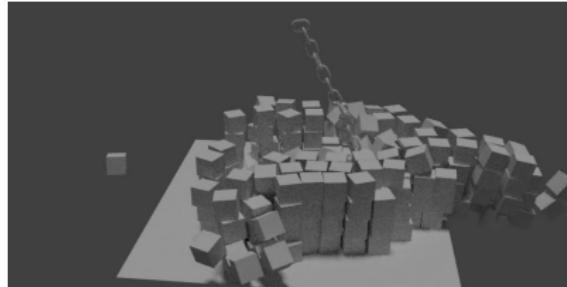
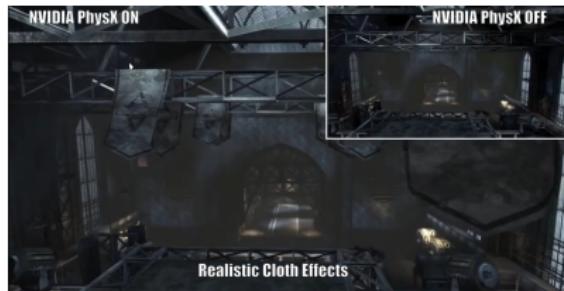
Simulación Física

Simulación física

La simulación física trata de reproducir el comportamiento dinámico y cinemático de los objetos de la escena.

Implica:

- Representar el estado de los objetos: posición, velocidad, aceleración y momento angular.
- Representar propiedades físicas de los objetos: densidad, elasticidad, coeficiente de fricción.
- Resolver las ecuaciones de la mecánica del sistema (Integración en el tiempo)



Simulación física

Componentes

- Detección de colisiones
- Cinemática
 - Cálculo de velocidades
- Dinámica
 - Cálculo de fuerzas
- Fractura

Tipos de objetos

- Puntual
- Rígido
- Deformable
- Fluido
- Tela

Simulación física

Para realizar la simulación física de la escena es necesario resolver las ecuaciones de la mecánica clásica:

Velocidad

$$\vec{V} = \frac{d\vec{S}}{dt}$$

Aceleración

$$\vec{a} = \frac{d\vec{V}}{dt}$$

Fuerza

$$\vec{F} = m\vec{a}$$

Fricción

$$f = \mu m \vec{n} \vec{g}$$

Velocidad angular

$$\vec{\omega} = \frac{d\vec{\theta}}{dt}$$

Aceleración angular

$$\vec{\alpha} = \frac{d\vec{\omega}}{dt}$$

Ley de Hooke (Muelle)

$$\vec{F} = -k\Delta\vec{X}$$

Simulación física

En la escena partimos del estado inicial de todos los objetos (posiciones, orientaciones, velocidades, aceleraciones, masas, velocidades y aceleraciones angulares), y debemos calcular la evolución de configuración de la escena a lo largo del tiempo.

El cálculo se realiza discretizando el tiempo, e integrando las ecuaciones en el tiempo: a partir de los valores en el instante t se calcula el estado en el instante $t + \Delta t$, discretizando las ecuaciones anteriores:

$$\vec{S}_{t+\Delta t} = \vec{S}_t + \vec{V}_t \Delta t$$

$$\vec{V}_{t+\Delta t} = \vec{V}_t + \vec{a}_t \Delta t$$

$$\vec{a}_{t+\Delta t} = \frac{\vec{F}_t}{m}$$

$$\vec{\theta}_{t+\Delta t} = \vec{\theta}_t + \vec{\omega}_t \Delta t$$

$$\vec{\omega}_{t+\Delta t} = \vec{\omega}_t + \vec{\alpha}_t \Delta t$$

$$\vec{F}_{t+\Delta t} = -k \vec{X}_t$$

Ejemplo de Simulación física: proyectil

Objeto de masa M , con velocidad inicial \vec{V}_0 que cae libremente sin rozamiento.

$$\vec{S}_{k+1} = \vec{S}_k + \vec{V}_k \Delta t$$

$$\vec{V}_{k+1} = \vec{V}_k + \vec{a}_k \Delta t$$

$$\vec{a}_{k+1} = \frac{\vec{g}}{m}$$

```
float V[3], S[3]={0,2,0}, a[3]={0,-9.8,0};  
float Dt=0.01, V0[3]={1000,200,0};  
float M = 10;  
...  
void init() {  
    a[2]= a[2]/M;  
    for(int i=0;i<3;++i)  
        V[i] = V0[i];  
}  
...  
void idle(){  
    if(disparado) {  
        for(int i=0;i<3;++i) {  
            S[i] += V[i] * Dt;  
            V[i] += a[i] * Dt; }  
    }  
    glutPostRedisplay();  
}
```

Ejemplo de Simulación física: muelle

Objeto de masa M, unido a un muelle de constante de Hook k sin rozamiento.

$$\vec{S}_{k+1} = \vec{S}_k + \vec{V}_k \Delta t$$

$$\vec{v}_{k+1} = \vec{V}_k + \vec{a}_k \Delta t$$

$$\vec{a}_{k+1} = \frac{\vec{F}}{m}$$

$$\vec{F}_{k+1} = -k\vec{X}$$

```
float S[3]={0,2,0},a[3]={0,0,0},v[3]={0,0,0};  
float Dt=0.01;  
float M = 10, K = 2,F;  
...  
void init() {  
}  
...  
void idle(){  
    F = - K * S[1];  
    a[1] = F /M;  
    for(int i=0;i<3;++i) {  
        S[i] += V[i] * Dt;  
        V[i] += a[i] * Dt;  
    }  
    glutPostRedisplay();  
}
```

Simulación física

La simulación es costosa:

- En sistemas interactivos (Juegos, Entornos virtuales,etc.) se hace cálculo aproximado.
- En sistemas no interactivos se puede realizar un cálculo preciso, que es mas lento: Películas, Aplicaciones científicas y técnicas.

Motores de física

Open source

- ODE
- NEWTON
- Bullet

Comerciales

- Havok (Intel)
- Physx (nVidia)
- Vortex (Montreal)
- Realflow (Nextlimit, España)

Objetos

- Rigid bodies
- Colliders
- Physic Material
- Joints
- Physics articulations
- Character Controllers

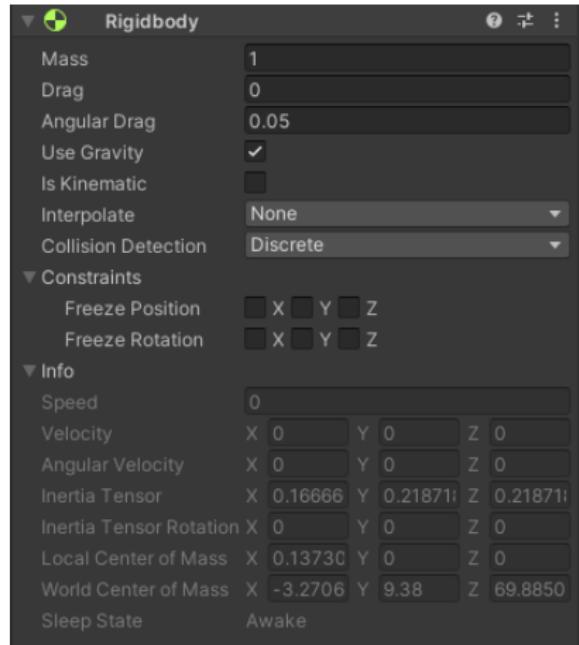
Las colisiones lanzan eventos *OnCollisionEnter*, *OnCollisionStay* y *OnCollisionExit*.

Simulación en Unity: Rigid bodies

Un Rigidbody (objeto rígido) es el componente que permite el comportamiento físico para un objeto.

Si un objeto tiene asociado un Rigidbody responderá a la gravedad. Además, si tiene Collider estará afectado por colisiones de otros objetos.

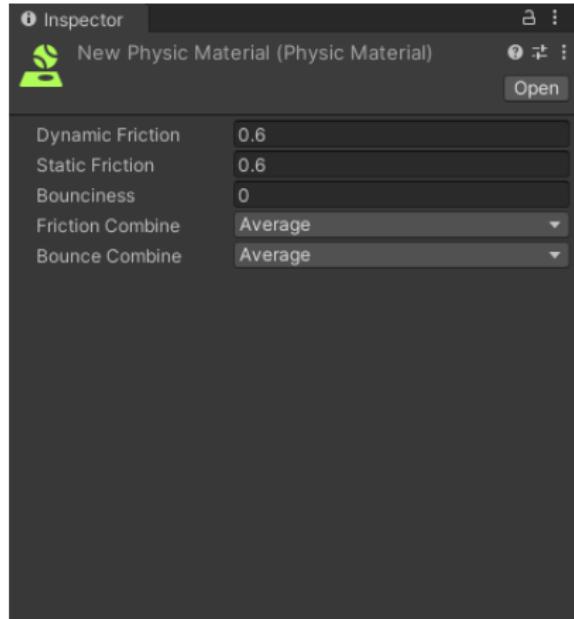
Se puede configurar la simulación para realizar *Continuous collision detection*



Simulación en Unity: Physics Material

Material físico es un componente que se utiliza para ajustar las propiedades físicas de los colliders en el motor de física.

Define atributos como fricción y restitución que afectan la interacción entre objetos, permitiendo un control más preciso sobre la simulación de colisiones y rebotes en un entorno 3D.

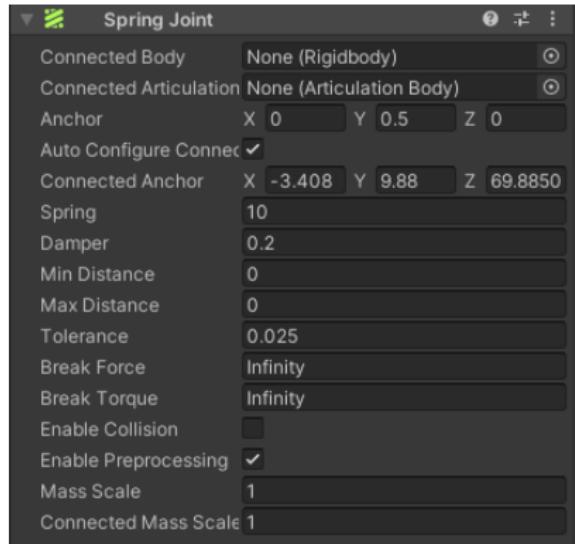


Simulación en Unity: Joints

Un *joint* es un componente que permite conectar y controlar la relación entre dos o más objetos en una simulación física.

Los joints se utilizan en el sistema de física de Unity para simular restricciones entre objetos. Ejemplos:

- Fixed Joint: mantiene dos objetos juntos sin permitir movimiento relativo.
- Hinge Joint: simula una conexión de bisagra permitiendo el movimiento rotacional alrededor de un eje específico.
- Spring Joint: mantiene unidos los objetos por un muelle.



Animación

Animación

La **animación por ordenador** es la técnica de generación de animaciones usando software para crear sensación de movimiento continuo.

- La escena y los actores están modelados en el ordenador.
- Se generan entre 24 y 30 imágenes por segundo.
- Para cada imagen se modifica el modelo para hacerlo evolucionar al siguiente fotograma.

Se basa en la persistencia de la visión

- Aristoteles observó que la imagen del sol permanecía después de mirarlo.
- Nos hace percibir como continuas una sucesión rápida de imágenes discretas.
- Es el mismo fundamento de la animación clásica, el cine, la televisión.

Técnicas

Fotogramas clave (keyframe)

Solo se editan algunos fotogramas, que se utilizan para *calcular* los fotogramas intermedios.

Esqueletos (Rigging)

Se añade un *esqueleto* al objeto. La edición de la pose de la malla se realiza editando el esqueleto, como se edita un modelo jerárquico.

Keyframe

El animador define configuraciones del modelo separadas varios fotogramas en la animación.

El sistema calcula los fotogramas intermedios (in-betweens) por interpolación o morphing.

Esta técnica se usaba ya en animación 2D clásica.

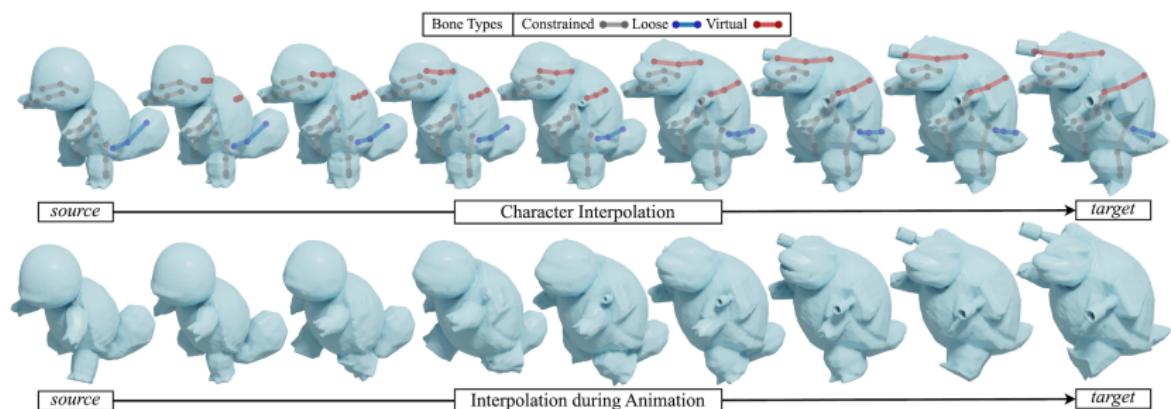


(c) Walt Disney Company, from "The Illusion of Life"

Interpolación entre keyframe

Se utilizan algoritmos que generan una malla $M(u)$ por interpolación entre dos mallas M_0 y M_1

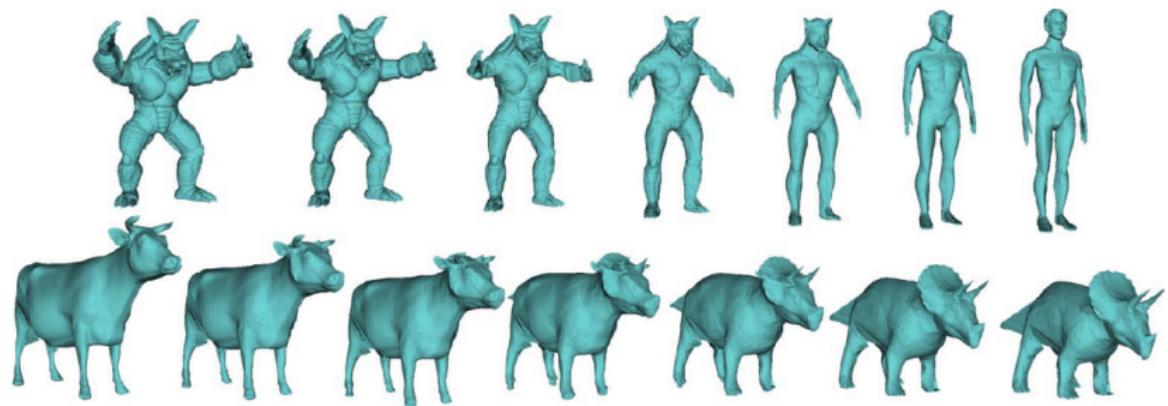
$$M(u) = F(M_0, M_1, u) \text{ t.q. } M(0) = M_0, M(1) = M_1 \quad (1)$$



Interpolación (*Morphing*)

Es posible transformar la malla utilizando algoritmos que generan una malla $M(u)$ por interpolación entre dos mallas M_0 y M_1

$$M(u) = F(M_1, M_2, u) \text{ t.q. } M(0) = M_0, M(1) = M_1 \quad (2)$$



B. Mocanu, T. Zaharia: A pseudo metamesh approach for 3D mesh morphing. 2013 IEEE International Conference on Consumer Electronics

Métodos de edición

- Simulación física.
- Proceduralmente.
- Edición del esqueleto (Rigging).
- Captura de movimientos.
- Cinemática inversa.

Estas técnicas se suelen combinar.

Simulación física

La configuración de los objetos que representan objetos físicos inanimados se puede calcular en cada fotograma usando simulación física.



[https:](https://80.lv/articles/creating-wave-simulations-with-blenders-flip-fluids/)

//80.lv/articles/creating-wave-simulations-with-blenders-flip-fluids/

Animación procedural

En algunos casos se puede calcular el comportamiento de un objeto de forma plausible(sin necesidad de que sea físicamente correcto) usando un algoritmo:

- Rotura de objetos.
- Andar.



<https://www.youtube.com/watch?v=U2JRjvxEn8>

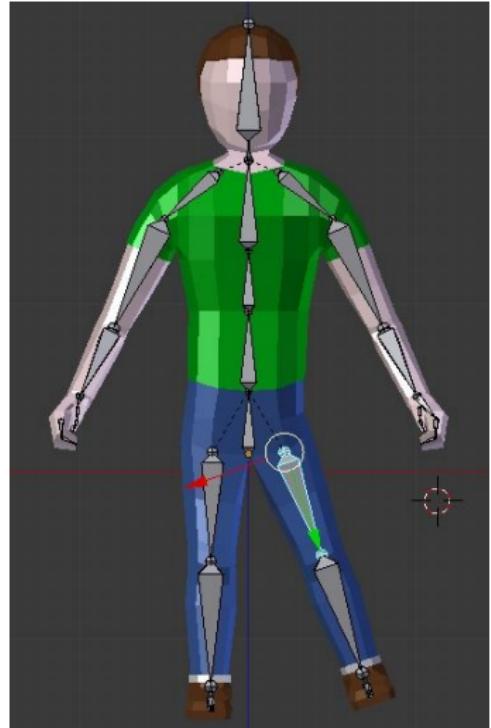
Rigging

La pose de los personajes se debe modificar en cada fotograma clave. Para facilitar la edición de la pose de los personajes se les añade un esqueleto (*rigging*).

El esqueleto es un modelo simplificado del personaje, formado por segmentos rígidos unidos por articulaciones, al que está vinculada la geometría del personaje.

El esqueleto se modifica en cada keyframe:

- Interactivamente por el animador
- Usando animación procedural
- Mediante captura de movimiento



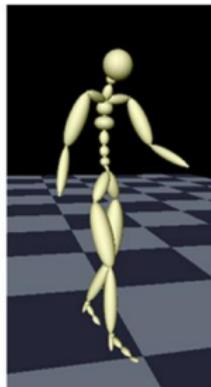
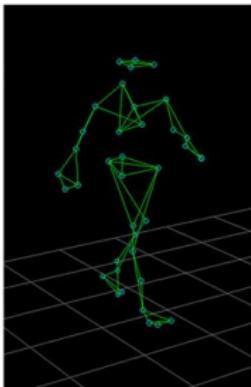
<https://www.graphicsandprogramming.net/fra/tutorial/blender/rigging/comment-rig-un-personnage-low-poly-avec-blender-3d>

Captura de movimiento (Mocap)

Transferencia de los movimientos realizados por un actor a un modelo 3D.

Proceso:

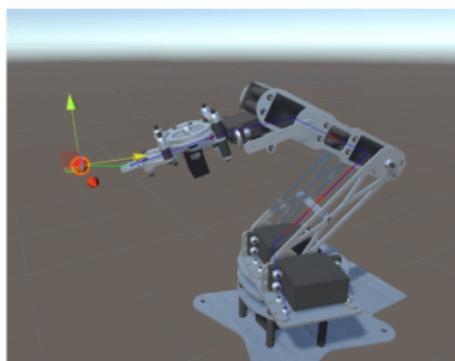
- El actor viste un traje con marcadores adosados.
- Se registra el movimiento del actor mediante sistemas de tracking (normalmente cámaras calibradas).
- A partir de las posiciones de los marcadores se calculan la configuración del esqueleto.



Cinemática inversa

Cálculo de la configuración de un modelo articulado para conseguir que alcance una determinada posición.

Utiliza métodos de optimización (descenso de gradiente).



<https://www.alanzucconi.com/2017/04/17/procedural-animations>