UNIVERSIDAD NACIONAL DEL LITORAL
FACULTAD DE INGENIERÍA Y CIENCIAS HÍDRICAS

Tecnicatura en diseño y programación de videojuegos

Modelos y Algoritmos para Videojuegos II

Unidad 4
Los Joint que unen al mundo

Docente
Lucas Sebastián Villa

Los Joint que unen al mundo

1. Pulley Joint (Junta de Polea)

Teoría:

- El Pulley Joint simula el comportamiento de una polea en un sistema físico, donde dos cuerpos están conectados por una cuerda que pasa por una serie de poleas.
- Este Joint aplica fuerzas a los cuerpos conectados para mantener la longitud de la cuerda y las velocidades de los cuerpos de acuerdo con la configuración establecida.

Aplicación en Videojuegos:

- En juegos de plataformas, el Pulley Joint se puede usar para crear sistemas de elevación, como ascensores o plataformas móviles que se mueven verticalmente.
- También es útil para simular la física de puentes colgantes o sistemas de poleas que requieren que el jugador ajuste las longitudes de las cuerdas para superar obstáculos.

Ejemplo:

```
// Crear una Pulley Joint entre dos cuerpos
b2PulleyJointDef pulleyJointDef;
pulleyJointDef.Initialize(body1, body2, groundAnchor1, groundAnchor2,
body1Anchor, body2Anchor, ratio);
b2PulleyJoint* pulleyJoint = (b2PulleyJoint*)world-
>CreateJoint(&pulleyJointDef);
```

2. Gear Joint (Junta de Engranaje)

Teoría:

- El Gear Joint simula la interacción entre dos cuerpos rígidos como si estuvieran conectados por un conjunto de engranajes, transfiriendo el movimiento rotacional de uno al otro.
- Este Joint mantiene una relación constante de velocidad angular entre los dos cuerpos, lo que garantiza un movimiento sincronizado de los engranajes.

Aplicación en Videojuegos:

- En juegos de puzzles o estrategia, el Gear Joint se puede utilizar para crear mecanismos de transmisión complejos que requieren que el jugador sincronice el movimiento de los engranajes para resolver desafíos.
- También es útil para simular sistemas de puertas controladas por engranajes o dispositivos de trampas accionados por movimiento en juegos de aventuras.

Ejemplo:

```
// Crear una Gear Joint entre dos cuerpos
b2GearJointDef gearJointDef;
gearJointDef.bodyA = body1;
```

```
gearJointDef.bodyB = body2;
gearJointDef.joint1 = joint1;
gearJointDef.joint2 = joint2;
gearJointDef.ratio = ratio;
b2GearJoint* gearJoint = (b2GearJoint*)world->CreateJoint(&gearJointDef);
```

3. Prismatic Joint (Junta Prismática)

Teoría:

- El Prismatic Joint restringe el movimiento relativo entre dos cuerpos a lo largo de una línea definida en un solo eje, permitiendo el movimiento lineal de un cuerpo con respecto al otro.
- Este Joint puede tener límites de translación para restringir el rango de movimiento lineal de los cuerpos conectados.

Aplicación en Videojuegos:

- En juegos de plataformas o acción, el Prismatic Joint se puede utilizar para crear plataformas móviles o puertas deslizantes que se mueven horizontal o verticalmente en respuesta a las acciones del jugador.
- También es útil para simular la física de brazos articulados en robots o vehículos controlados por el jugador en juegos de simulación o estrategia.

Ejemplo:

```
// Crear una Prismatic Joint entre dos cuerpos
b2PrismaticJointDef prismaticJointDef;
prismaticJointDef.Initialize(body1, body2, axis, anchor, lowerTranslation,
upperTranslation, enableLimit);
b2PrismaticJoint* prismaticJoint = (b2PrismaticJoint*)world-
>CreateJoint(&prismaticJointDef);
```

4. Revolute Joint (Junta de Revolución)

Teoría:

- El Revolute Joint permite el movimiento rotacional de un cuerpo alrededor de un punto fijo, proporcionando libertad angular para giros ilimitados entre los cuerpos conectados.
- Este Joint puede tener límites de ángulo para restringir el rango de rotación de los cuerpos.

Aplicación en Videojuegos:

- En juegos de plataformas o aventuras, el Revolute Joint se puede utilizar para simular articulaciones en personajes animados, permitiendo que los brazos, piernas o cabezas giren en respuesta a la física del juego o las acciones del jugador.
- También es útil para crear bisagras en puertas, conexiones en ruedas de vehículos o articulaciones en armas en juegos de acción o shooter.

Ejemplo:

```
// Crear una Revolute Joint entre dos cuerpos
b2RevoluteJointDef revoluteJointDef;
revoluteJointDef.Initialize(body1, body2, anchor);
b2RevoluteJoint* revoluteJoint = (b2RevoluteJoint*)world-
>CreateJoint(&revoluteJointDef);
```

5. Mouse Joint (Junta de Ratón)

Teoría:

- El Mouse Joint permite al usuario controlar un cuerpo en el mundo físico como si estuviera "arrastrando" un objeto con un ratón, proporcionando una forma interactiva de interactuar con objetos en el juego.
- Este Joint aplica fuerzas al cuerpo conectado para seguir la posición del cursor del ratón y permite al jugador interactuar directamente con objetos físicos en pantalla.

Aplicación en Videojuegos:

- En juegos de puzzles o aventuras, el Mouse Joint se puede utilizar para permitir que el jugador arrastre objetos en el mundo del juego, como bloques o llaves, para resolver acertijos o desafíos.
- También es útil para implementar controles de apuntar y disparar en juegos de disparos en primera persona o para controlar la dirección de movimiento de vehículos en juegos de carreras.

Ejemplo:

```
// Crear una Mouse Joint para interactuar con un cuerpo
b2MouseJointDef mouseJointDef;
mouseJointDef.bodyA = groundBody;
mouseJointDef.bodyB = body;
mouseJointDef.target = mousePosition;
mouseJointDef.maxForce = maxForce;
b2MouseJoint* mouseJoint = (b2MouseJoint*)world-
>CreateJoint(&mouseJointDef);
```