UNIVERSIDAD TECNOLÓGICA NACIONAL

Facultad Regional Buenos Aires

Ingeniería en Sistemas de Información

Técnicas graficas por computador

2010

PROFESORES: **Matias Leone,**
Leandro Barbagallo

TRABAJO PRÁCTICO

TEMA: Motor de Física.

Objetivo del TP: Realizar una herramienta extensible que realice los cálculos de física.

GRUPO: **Piguyis**

Integrantes del equipo			
Legajo	Apellido y Nombre		
115227-0	García, Rodrigo Nicolas		
Entrega / Revisión	1	2	3
A entregar el	9-11-2010		
Entregado el			
Calificación			
OBSERVACIONES:			

OBJETIVOS

- Crear un motor de física aplicado a gráficos por computador.
- Mostrar lo desarrollado con escenarios de prueba.

ALCANCE

- Realizar los cálculos de colisiones entre esferas.
- Preformar las colisiones con octree.
- Realizar los cálculos físicos (derivadas) con una solución discreta, Aproximación de Euler.
- Dejar el código lo suficientemente extensible para que en próximos años se pueda continuar con la idea.

EXPLICACION DE IMPLEMENTACION

- La implementación vino dada por el estudio de diferentes técnicas que utilizan los motores actuales de física para los cálculos matemáticos necesarios.
- En base a los motores físicos Box2DLite, Box2D 2.0.1 (http://www.gphysics.com) (C++) y Box2DX (C#) se realizo un estudio de cómo implementan esto la gran variedad de soluciones.
- La implementación tiene cuatro etapas donde se realizan los cálculos necesarios para que la interacción sea lo mas real posible.
    - Inicialización:
    - Primera etapa: Se inicializa el mundo se agregan los objetos dinámicos (con diferentes masas y los estáticos con masa infinita), para optimización se agregan en un octree.
    - Rendering:
    - Segunda etapa: Se realizan los testeos de colisión, realizando un cálculo del punto de colisión que luego es utilizado para hacer el cálculo físico.
    - Tercera etapa: Se realizan los cálculos físicos (impulso elástico), utilizando la aproximación de euler, a partir del tiempo trascurrido en rendering.
    - Cuarta etapa: Luego de realizar las operaciones físicas, se dispone de las nuevas velocidades, lo que nos queda es aplicar dichas velocidades y calcular la nueva ubicación del objeto rígido.
- Explicación de clases importantes:
    - `World`: Es quien realiza la interacción con todas las etapas.
        - `public void Step(float timeStep):` metodo donde son llamadas todas las etapas de rendering.
        - `public void CollidePhase():` Testeo de colisiones (Contact) y inicializacion de Arbiters.
    - `Arbiter`: Contiene la lógica de la etapa física.
        - `public void PreStep(float inverseTimeStep):` Actualiza el Contact realizando los cálculos matemáticos necesarios para luego poder aplicar los impulsos a los cuerpos rígidos.
        - `public void ApplyImpulse():` Realiza el calculo de cómo van a terminar los impulsos, o sea las velocidades finales de los objetos intervinientes.
    - `RigidBody`: Es la representacion de un cuerpo rigido en el sistema.
        - Contiene los siguentes atributos importantes:
            - _mass: La masa del objeto.
            - _location: Es la posicion del cuerpo.
            - _velocity: Es la velocidad que tiene el objeto en este momento
            - _fuersasInternas: Son las unicas aplicadas apartir de estas se calcula la aceleracion del cuerpo, esta fuerza puede ser la gravedad.
            - _boundingVolume: Es el volumen que contiene al cuerpo. Actualmente solo existen volumenes esféricos.

- `BoundingVolume:` Es el volumen que contiene los cuerpos, esta es una clase abstracta.
  - Toda implementacion tiene que implementar los siguentes metodos:
    - `public abstract void SetPosition(Vector3 position);`
    - `public abstract Vector3 GetPosition();`
    - `public abstract float GetRadius();`
    - `public abstract void render();`
    - `public abstract void dispose();`
  - La única implementación en este momento es: `BoundingSphere`
- `IEscena`: Es una escena de test para mostrar el motor, de esta manera se implementa la SolucionAlumno:
  - `Se tiene que implementar`
    - `String GetTitle();`
    - `String GetDescription();`
    - `void InitEscena();`
    - `void Render(float elapsedTime);`
    - `void CloseEscena();`
  - Existe una implementación básica que es:
    - `EscenaBase`: de esta manera en todas las escenas se evita realizar todos los cambios necesarios para constrir una esena, sino que solo implementa la creacion de los cuerpos.