Introdução à Box2D

Rossana Baptista Queiroz

Mini-Seminário

- Apresentação de ~15min (**01/09**)
  - Entregar a apresentação (.ppt, ou .pdf)
  - Opcional: materiais complementares (.zip)
- Investigar:
  - Propósito da Engine
  - Principais elementos
  - Linguagem
  - Free/Open/Proprietary
  - Quem está usando?
  - Exemplos (videozinhos, demos...)

Vai ser disponibilizado no Moodle
Caprichem!!! ☺

O que é a Box2D?

- Física de corpos rígidos no espaço 2D
  - Objetos não deformáveis
- Desenvolvida por Erin Catto
  - @erin_catto
- Open Source, licença zlib
- C++, independente de SO
- Módulos
  - Common
  - Collison
  - Dynamics
  - Rope
- Não faz *rendering*

Começando...

- Download da última versão
  - Fresquinha (03/11/2013) – v **2.3.0**
- Compilação com o Cmake (para VS2010) ou direto no VS2012 (*solution* vem junto)
- O que precisa configurar no VS2010/2012?
  - Properties → C/C++ → Additional Include Directories
    - Diretório com os cabeçalhos (.h)
  - Properties → Linker → Additional Include Libraries
    - Diretório com a biblioteca estática (.lib)
- Documentação oficial:
  - http://www.box2d.org/manual.pdf - Baixar!

Começando...

- Projeto Iniciando Box2D
  - Box2D + OpenGL para visualização
  - Tratamento de eventos com a GLUT
  - Rotinas de *callback* (deste projeto)

```
glutDisplayFunc(SimulationLoop);
glutReshapeFunc(Resize);
glutKeyboardFunc(Keyboard);
glutTimerFunc(framePeriod, Timer, 0);
```

Mundo

- Ambiente onde ocorre a simulação

```
// O objeto world serve para armazenar os dados da simulação
b2World *world;
```

- Inicializando o mundo

```
// Define the gravity vector.
b2Vec2 gravity(0.0f, -9.8f);

// Inicializa a biblioteca Box2D
world = new b2World(gravity);
```

Passo da simulação

- Parâmetros da simulação

```
// Define os parâmetro para a simulação
// Quanto maior, mais preciso, porém, mais lento
velocityIterations = 6;
positionIterations = 2;
timeStep = 1.0f / 60.0f; //60 Hz
```

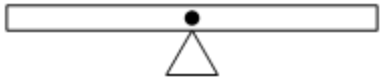
- Passo da simulação (a cada ciclo do programa)

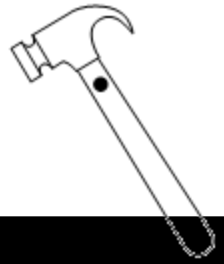
```
world->Step(timeStep, velocityIterations, positionIterations);
world->ClearForces();
```

CORPOS RÍGIDOS

Física de Corpos Rígidos

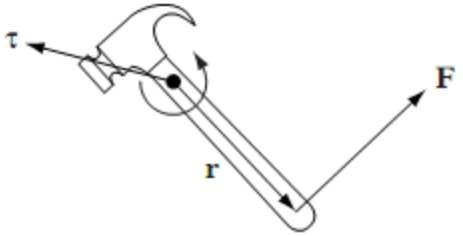
- Corpo Rígido
  - Conjunto finito de partículas cujas distâncias não variam com o tempo
- Centro de massa (ou “centro de gravidade”)
  - Ponto de balanço de um objeto: se você dividir o objeto em dois, passando qualquer linha fina sobre este ponto, você terá dois objetos que possuem exatamente o mesmo peso
  - Em objetos simétricos e com distribuição de massa igual nas partículas, esse ponto fica exatamente no seu centro





Física de Corpos Rígidos

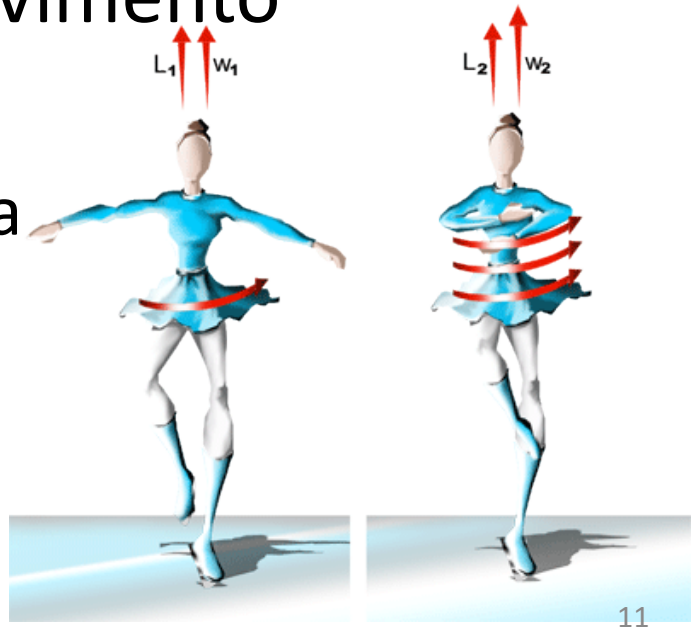
- Movimento de Translação: quando uma Força é aplicada direto no centro de massa do objeto
- Movimento de Rotação: quando a força é aplicada em outro ponto
  - gera um **Torque**



- Massa e Momento de Inércia:
  - No movimento de translação, quando a mesma força é aplicada a objetos de **massas** diferentes, observam-se **acelerações** diferentes.
  - No movimento de rotação, quando o mesmo **torque** é aplicado em objetos idênticos com **distribuição diferente de massa**, observam-se **acelerações angulares** diferentes.

Exemplo da Patinadora

- Momento de inércia $I_1 \neq I_2$
  - distribuição da massa do corpo
- Aceleração angular $w_1 \neq w_2$
- Porém, a quantidade de movimento (momentum) angular $L_1 = L_2$
  - Lei da conservação da energia



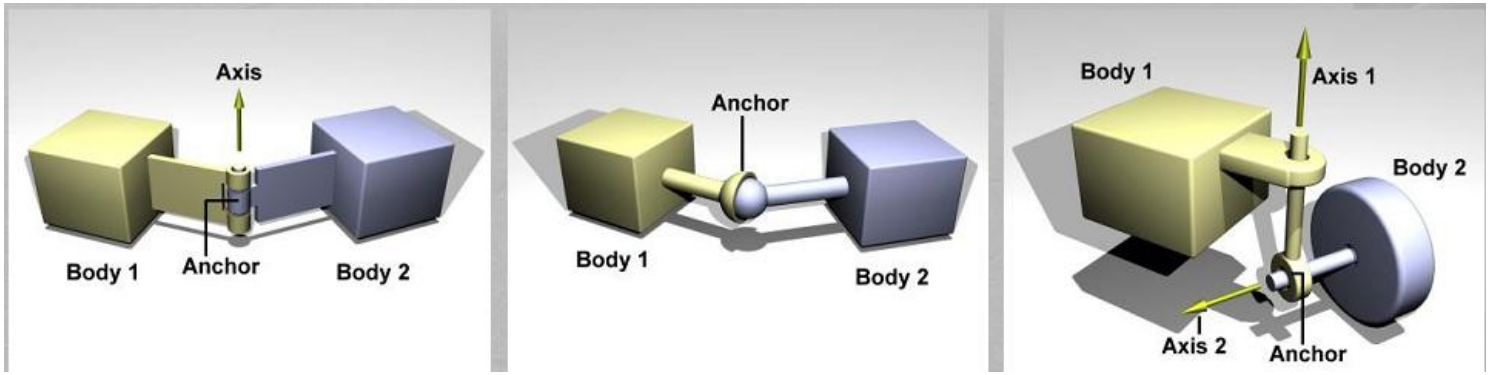
Posição ⇔ Ângulo
Velocidade ⇔ Velocidade angular
Aceleração ⇔ Aceleração angular
Força ⇔ Torque

Física de Corpos Rígidos

- Restrições
- Há situações em que um corpo não pode se mover livremente
  - Ele está colidindo com outro
  - Ele está ligado a outro por uma articulação
    - juntas

Física de Corpos Rígidos

- Juntas
  - Ponto de conexão entre 2 corpos rígidos
    - Esse ponto define restrições entre os movimentos dos 2 atores
    - Graus de liberdade
      - Na rotação
      - Na translação



Física de Corpos Rígidos

- Força de atrito: contrário a tendência de deslizamento
  - Coeficiente atrito dinâmico
  - Coeficiente de atrito estático (sempre maior que o dinâmico)

- Colisões
  - Quando dois objetos colidem, eles (normalmente) perdem parte de sua energia cinética
    - Coeficiente de restituição

Corpos Rígidos na Box2D

- O mundo é composto por corpos rígidos
- Possuem dois componentes
  - Atributos
  - Fixtures

Criação de um corpo rígido

- Definições dos atributos do corpo

```
// Cria as definicoes do CORPO rígido.
b2BodyDef bodyDef;
bodyDef.position.Set(0,10);
bodyDef.type = b2_dynamicBody;

// Cria um novo corpo a partir das informações da bodyDef
// Nunca use NEW ou MALLOC para criar um corpo
world->CreateBody(&bodyDef);
```

Corpos Rígidos

Atributos - type

- struct b2BodyDef::type
  - Define o tipo do corpo.
    - *b2_staticBody*
    - *b2_kinematicBody*
    - *b2_dynamicBody*

Corpos Rígidos

Atributos - type

- *b2_staticBody*
  - Não se move durante a simulação
  - Tem massa infinita e velocidade zero
  - Pode ser movido manualmente pelo programa
  - Usado para criar obstáculos

Corpos Rígidos

Atributos - type

- *b2_kinematicBody*
  - Move-se de acordo com sua velocidade.
  - Não responde a forças.
  - Pode ser movido manualmente pelo programa, mas em geral deve ser movido por sua velocidade.

Corpos Rígidos

Atributos - type

- *b2_dynamicBody*
  - tem seu comportamento totalmente simulado
  - Pode ser movido manualmente pelo programa, mas em geral deve ser movido pelas forças a ele aplicadas.
  - Tem massa finita e diferente de zero.
  - Se for tentado colocar massa zero, ela é convertida para 1kg.

Corpos Rígidos

Atributos

- struct b2BodyDef::Position
  - Define a posição do corpo.
  - Deve ser usado antes de criar o objeto
- struct b2BodyDef::angle
  - Define a orientação do corpo.
  - Em radianos

Corpos Rígidos

Atributos

- struct b2BodyDef::linearDamping
  - Reduz a velocidade do objetos durante uma translação.
  - Não depende de fricção.
  - Usar entre 0 e 0.1.
- struct b2BodyDef::angularDamping
  - Reduz a velocidade do objetos durante uma rotação.
  - Não depende de fricção.
  - Usar entre 0 e 0.1.

Corpos Rígidos

Atributos

- struct b2BodyDef::awake
  - true/false
  - Define manualmente como o objeto está
  - Um objeto com awake == false não é simulado
  - Permite melhorar o desempenho da simulação
  - Se estiver dormindo e outro objeto colidir nele, ele acorda.

Corpos Rígidos

Atributos

- struct b2BodyDef::allowSleep
  - true/false
  - Permite que um objeto deixe de ser simulado.
  - A detecção é feita pela própria BOX
  - Melhora a performance

Corpos Rígidos

Atributos

- struct b2BodyDef::fixedRotation
  - true/false
  - Evita que o corpo sofra rotações
  - Mesmo que alguma força seja aplicada sobre ele.

Corpos Rígidos

Atributos

- struct b2BodyDef::bullet
  - true/false
  - Afeta somente corpos dinâmicos.
  - Usado para forçar o uso de CCD(continuous collision detection)
  - Reduz o desempenho
  - Usa-se em objetos que se movem muito rápido no cenário

Corpos Rígidos

Atributos

- struct b2BodyDef::active
  - true/false
  - Define se um objeto participa ou não dos cálculos da física.

Corpos Rígidos

- Definem as propriedades únicas do objeto
- A estrutura de dados para isto é a **b2FixtureDef**

- Atributos
  - Forma
  - Densidade
  - Atrito
  - Restuição

Corpos Rígidos

- Criação de uma fixture

```
b2FixtureDef fixture;
// Cria uma nova Fixture a partir dos parâmetros
da física do objeto a partir dos dados da b2FixtureDef
// Nunca use malloc ou New
body->CreateFixture(&fixture);
.....
body->DestroyFixture(&fixture);
```

Corpos Rígidos

Fixtures

- struct b2FixtureDef::shape
  - Define a forma do objeto
  - Pode ser uma AABB ou uma OOBB. Isto deve ser definido com o método SetAsBox

Corpos Rígidos

Fixtures

- struct b2FixtureDef::shape
- AABB
  - void SetAsBox(float32 hx, float32 hy);
  - hx: metade da altura da AABB
  - hy: metade da largura da AABB

Corpos Rígidos

Fixtures

- struct b2FixtureDef::shape
- OOBB
  - void SetAsBox(float32 hx, float32 hy, const b2Vec2& center, float32 angle);
  - center : centro da box em coordenadas locais
  - angle: angulo de rotação da box

Corpos Rígidos

Fixtures

- struct b2FixtureDef::density
  - Usada para calcular a massa do objeto
  - Toma como base a área
  - Pode ser positiva ou nula.

```
//Primeiro, criamos a definição do corpo
b2BodyDef bodyDef;
bodyDef.position.Set(0,10);
bodyDef.type = b2_dynamicBody;

//Estamos usando uma forma de poligono, que pode ter até 8 vértices
b2PolygonShape forma;
forma.SetAsBox(5,5);

//Depois, criamos uma fixture que vai conter a forma do corpo
b2FixtureDef fix;
fix.shape = &forma;
//Setamos outras propriedades da fixture
fix.density = 10.0;
fix.friction = 0.5;
fix.restitution = 0.5;

//Por fim, criamos o corpo...
object =  world->CreateBody(&bodyDef);
//... e criamos a fixture do corpo
object->CreateFixture(&fix);
```

Corpos Rígidos

- Fixtures
  - Outras formas:
    - b2CircleShape

```
b2CircleShape circle;
circle.m_radius = 1.0;

b2FixtureDef fixCircle;
fixCircle.shape = &circle;
fixCircle.density = 1.0;

object->CreateFixture(&fixCircle);
```

    - b2EdgeShape

```
b2EdgeShape shape;
shape.Set(b2Vec2(-20.0f, -20.0f),
b2Vec2(20.0f, -20.0f));
ground->CreateFixture(&shape,0.0);
```

Corpos Rígidos

Fixtures

- struct b2FixtureDef::friction
  - Define o coeficiente de atrito entre objetos
  - Usada para simular o deslocamento de um objeto sobre outro.
  - O atrito dinâmico e o estático são iguais
  - Usa-se entre 0 e 1, mas pode ser qualquer número não negativo.

Corpos Rígidos

Fixtures

- struct b2FixtureDef::friction
  - A fricção resultante entre dois objetos

$$FR = \sqrt{F1 * F2}$$

Corpos Rígidos

Fixtures

- struct b2FixtureDef::restitution
  - Usado para fazer objetos saltarem no momento de uma colisão.
  - Usa-se valores entre 0 e 1.
  - Um valor 0 define uma colisão inelástica
    - o objeto não salta
  - Um valor 1 define uma colisão perfeitamente elástica
    - A velocidade do objeto depois da colisão é a mesma de antes da colisão

Corpos Rígidos

- struct b2FixtureDef::restitution
  - A restituição resultante entre dois objetos é dada pela fórmula

$$RR = \max(R1, R2)$$

Simulação

- Executada em duas fases
  - Velocity phase
    - Determina os impulsos produzidos pelos objetos uns nos outros
  - Position Phase
    - Ajusta a posição dos objetos apra reduzir as colisões
- Processos iterativos
    - Geram aproximações cada vez melhores
    - Muitas iterações
      - Simulação mais realista
      - Maior tempo de simulação

Simulação

- Executa com base em *Time Steps*
- Define-se o tempo em que se quer saber o resultado da simulação

```
timeStep = 1/60
world->Step(timeStep, velocityIterations,
positionIterations);
```

Simulação

- Obter Informações de todos os objetos

```
void PrintBodies()
{
    b2Body *b;
    float ang;
    b2Vec2 pos;
    for(b = world->GetBodyList(); b; b=b->GetNext())
    {
      pos = b->GetPosition();
      ang = b->GetAngle();
      printf("%4.2f %4.2f %4.2f\n", pos.x, pos.y, ang);
    }
}
```

Exercícios

- Ver lista de exercícios!!!

Densidade

- Densidade = massa/volume
  - Considerar profundidade de 1m para todos os objetos
  - Ou ssar a área ao invés do volume, pois é 2D
  - Volume da caixa (cubo)
    - Volume do cubo: altura x largura x profundidade
  - Volume do círculo (esfera)

$$V = \frac{4}{3}\pi r^3$$

- Ou usar apenas a área πr^2

Passos para a criação do corpo rígido

```
//Novo objeto
b2Body *novoObjeto;        (...)
```

```
//1º passo: criação da definição do corpo (b2BodyDef)
b2BodyDef b;
b.position.Set(20,20);
b.type = b2_dynamicBody;
//2º passo: criação do corpo pelo mundo (mundo cria corpo)
novoObjeto = world->CreateBody(&b);
//3º passo: criação da definição da forma (b2PolygonShape, b2CircleShape ou b2EdgeShape)
b2PolygonShape caixa;
caixa.SetAsBox(2, 2);
//4º passo: criação da definição da fixture (b2FixtureDef)
//Não esquecer de associar a forma com a fixture!
b2FixtureDef f;
f.shape = &caixa;
f.density = 2.0/4*4; //2kg e área da caixa de lado 4m
//5º passo: criação da fixture pelo corpo (objeto cria fixture)
novoObjeto->CreateFixture(&f);
```