Acadêmico: Victor Arndt Mueller
Orientador: Dalton Solano dos Reis

SIMULAÇÃO FÍSICA DE CORPOS RÍGIDOS EM 3D

Roteiro

- Introdução
- Objetivos do trabalho
- Fundamentação teórica
- Desenvolvimento do trabalho
- Conclusão
- Extensões

Introdução

- Importância da simulação de física
  - Jogos
  - Filmes
  - Simulações científicas
- Problemas
  - Complexidade
  - Desempenho
- Solução
  - Motores de física

Objetivos do trabalho

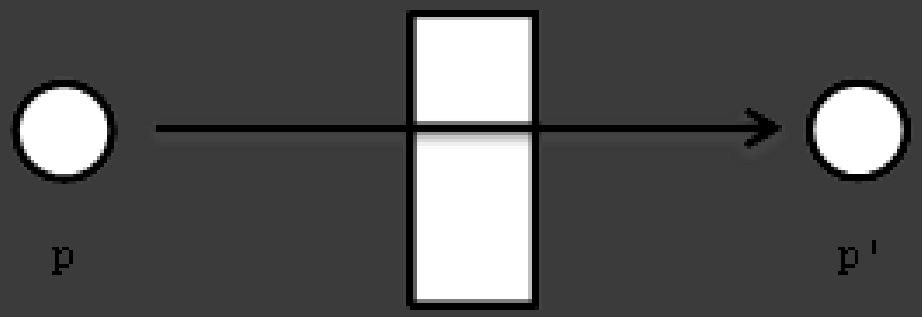
- Detectar colisões entre objetos convexos
- Tratar as colisões detectadas
- Simular o movimento dos objetos
- Disponibilizar uma aplicação exemplo

Trabalho correlatos

- Havok Physics (2010)
- PhysX (2010)
- Newton Game Dynamics (2007)
- Mobile 3D Game Engine (2005)

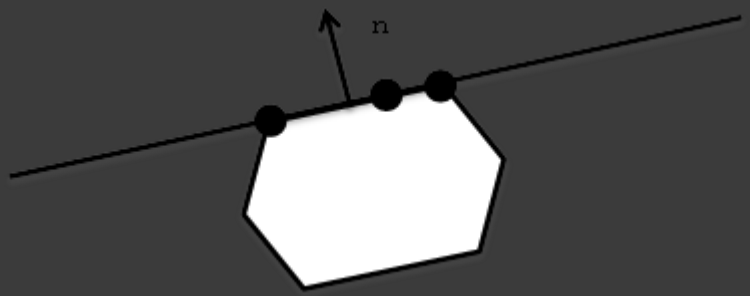
Fundamentação teórica

- Detecção de colisão
  - Pontos de contato, profundidade de penetração e normal da colisão



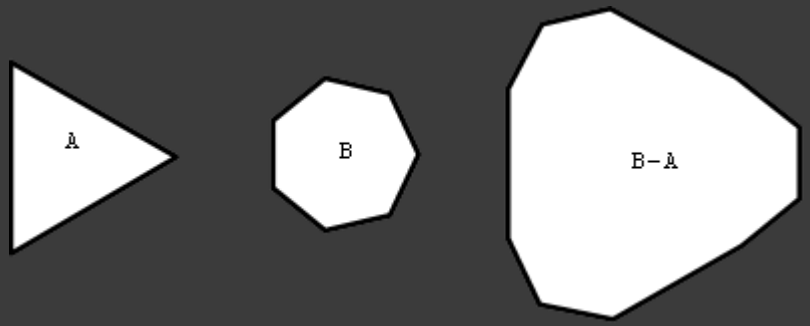
Fundamentação teórica

- Algoritmo XenoCollide
  - Funções de mapeamento de suporte



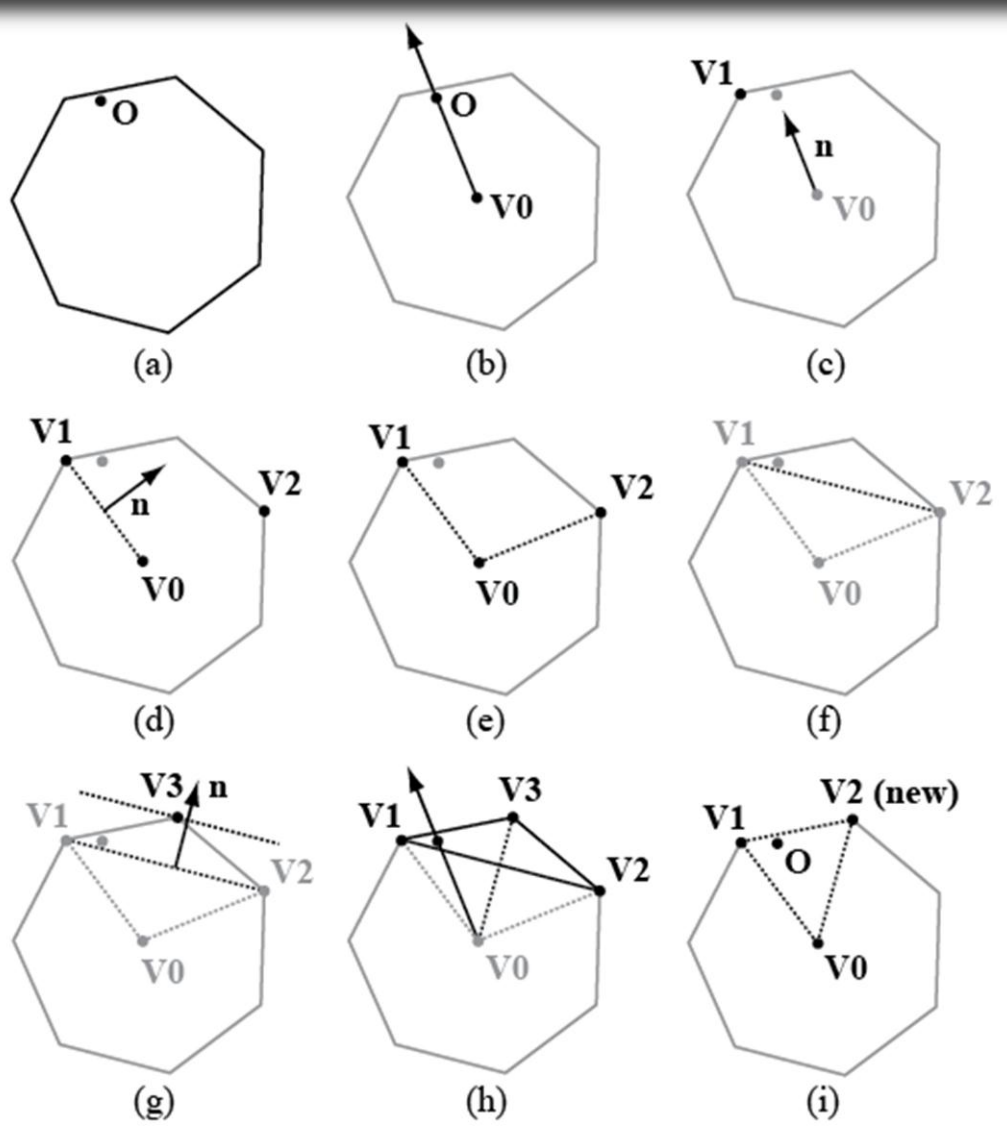
Fundamentação teórica

- Algoritmo XenoCollide
  - Diferença de Minkowski de dois objetos



Fundamentação teórica

- Algoritmo XenoCollide
  - Refinamento de portais de Minkowski


O
(a)
O
V0
(b)
V1
n
V0
(c)
V1
n
V2
V0
(d)
V1
V2
V0
(e)
V1
V2
V0
(f)
V3
n
V1
V2
V0
(g)
V1
V3
V2
V0
(h)
V1
V2 (new)
O
V0
(i)

Fundamentação teórica

- Tratamento da colisão
  - Separar os corpos em colisão
  - Força e torque levam tempo
  - Impulso tem efeito imediato

Fundamentação teórica

- Simulação
  - Calcular a nova posição após um *timestep*
  - Integração numérica
  - *Timestep* fixo e variável
    - Determinístico

Fundamentação teórica

- Orientação em 3D
  - *Quaternion*
    - Eixo $\boldsymbol{v}$ e ângulo θ
    - Comprimento unitário
    - Não é intuitivo
    - *Spherical linear interpolation*
  - Matriz
  - Ângulos de Euler

$$[\cos(\theta/2) \quad (\sin(\theta/2)\ \boldsymbol{v}_x \quad \sin(\theta/2)\,\boldsymbol{v}_y \quad \sin(\theta/2)\,\boldsymbol{v}_z)]$$

Requisitos

- Requisitos funcionais
  - Detectar colisões
  - Tratar as colisões
  - Simular movimento
  - Aplicação exemplo

Requisitos

- Requisitos não-funcionais
  - Visual Studio 2010 e C++
  - 30 quadros por segundo
  - Robustez numérica

Diagrama de classes

Diagrama de classes

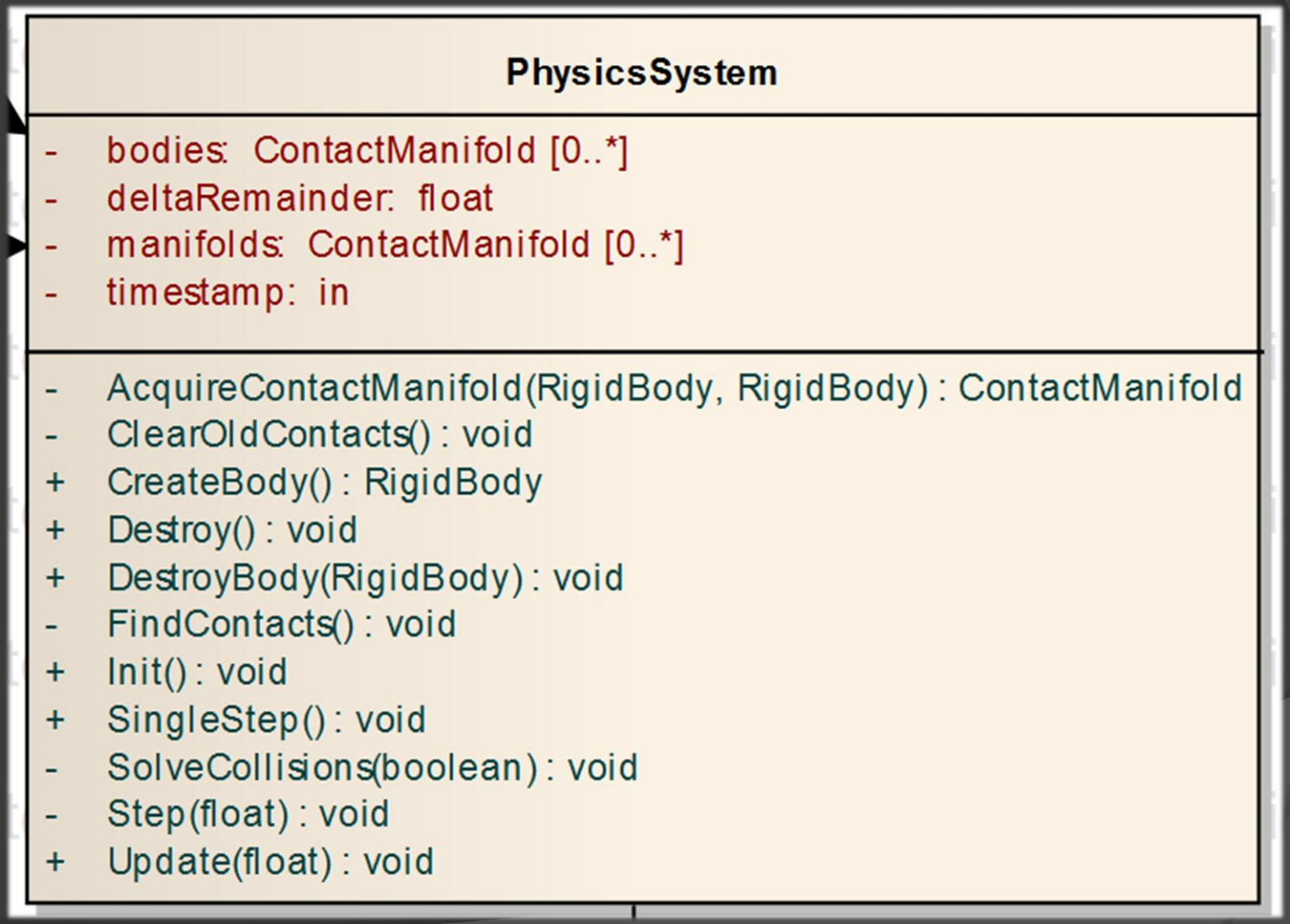


Diagrama de classes

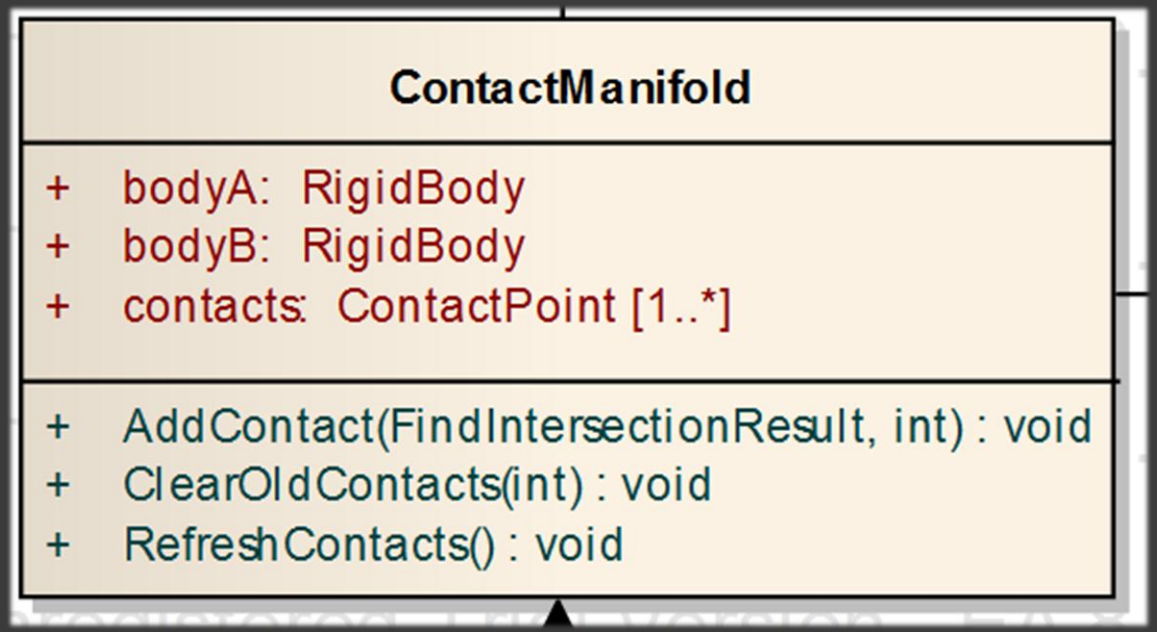


Diagrama de classes

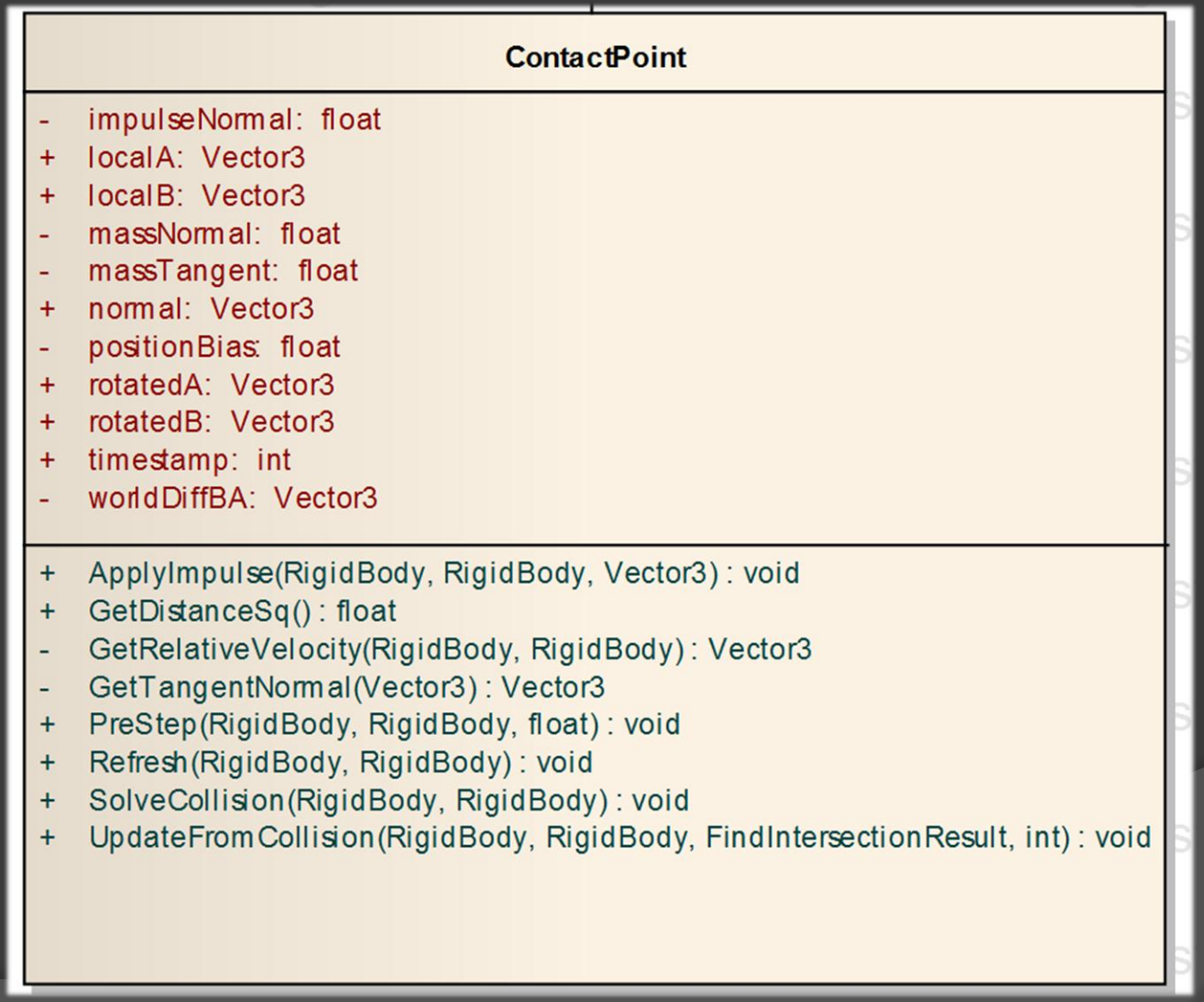
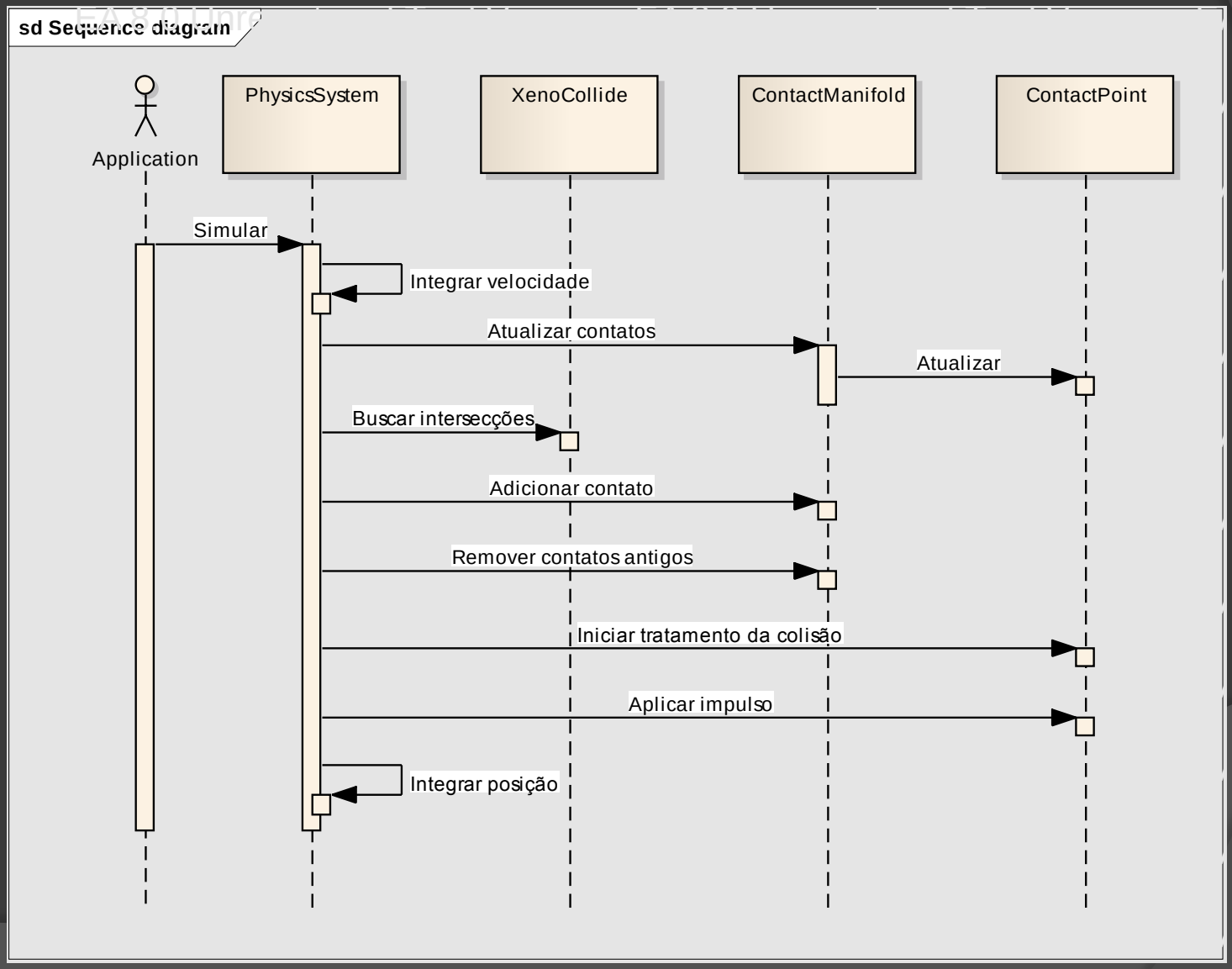
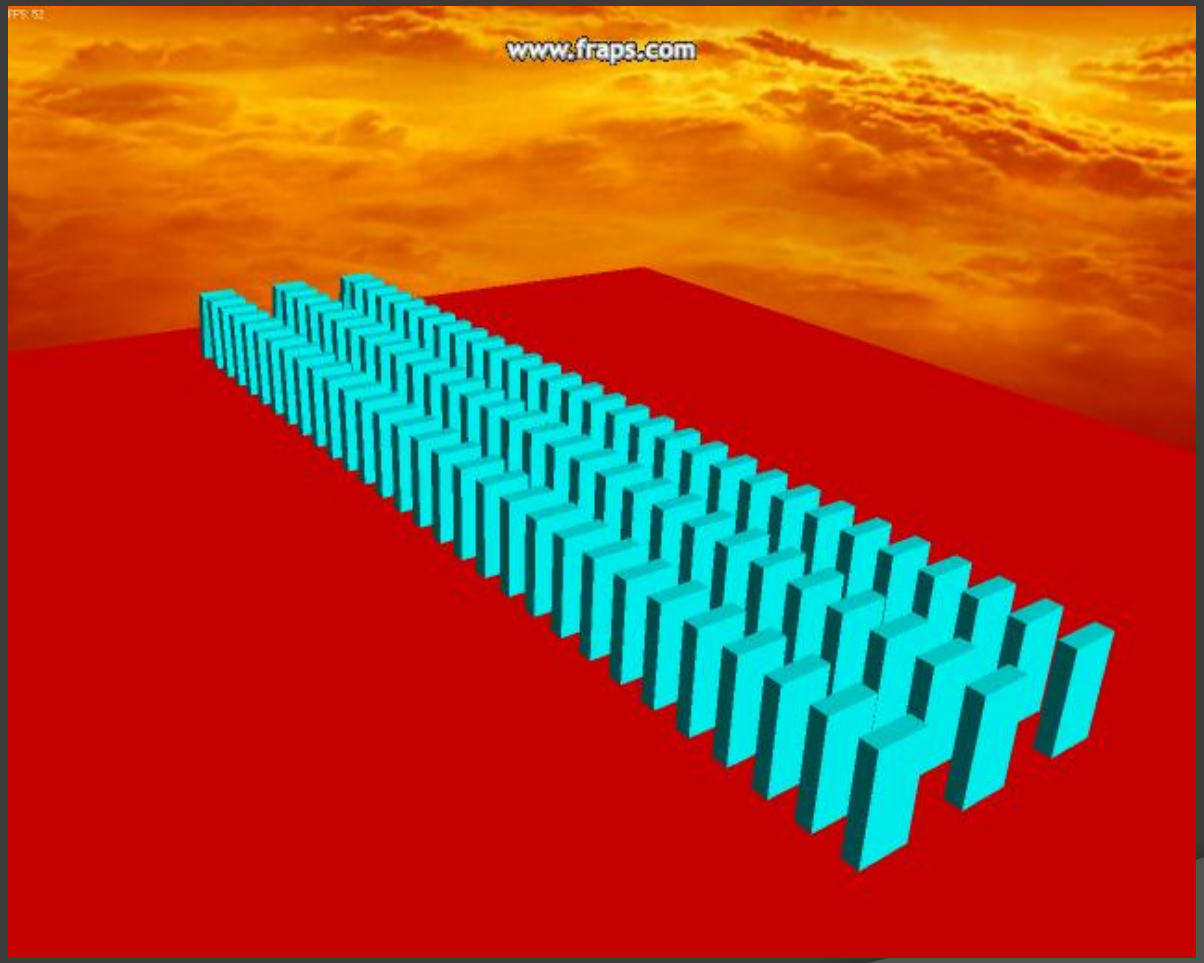


Diagrama de sequência



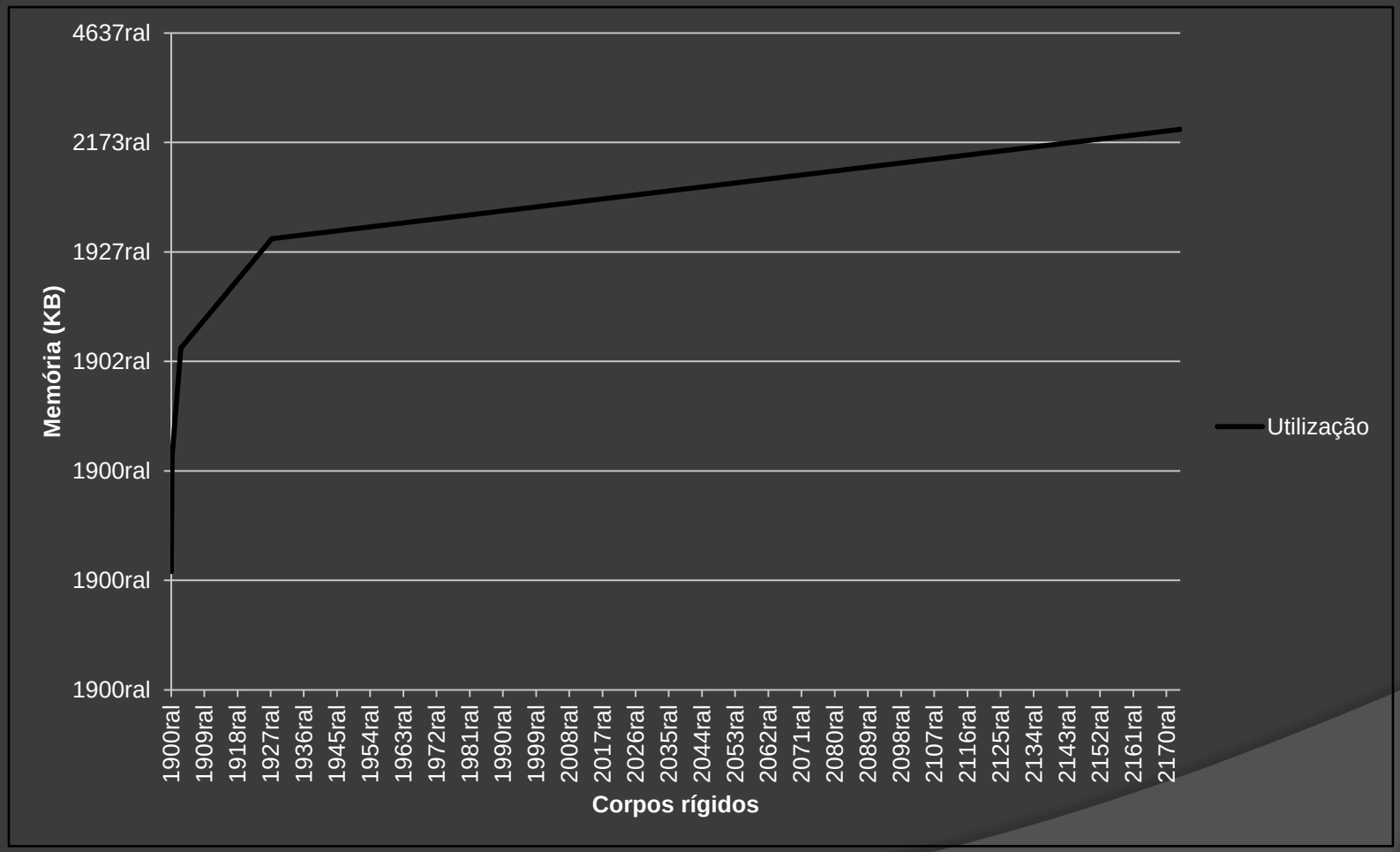
Operacionalidade



Resultados e discussões

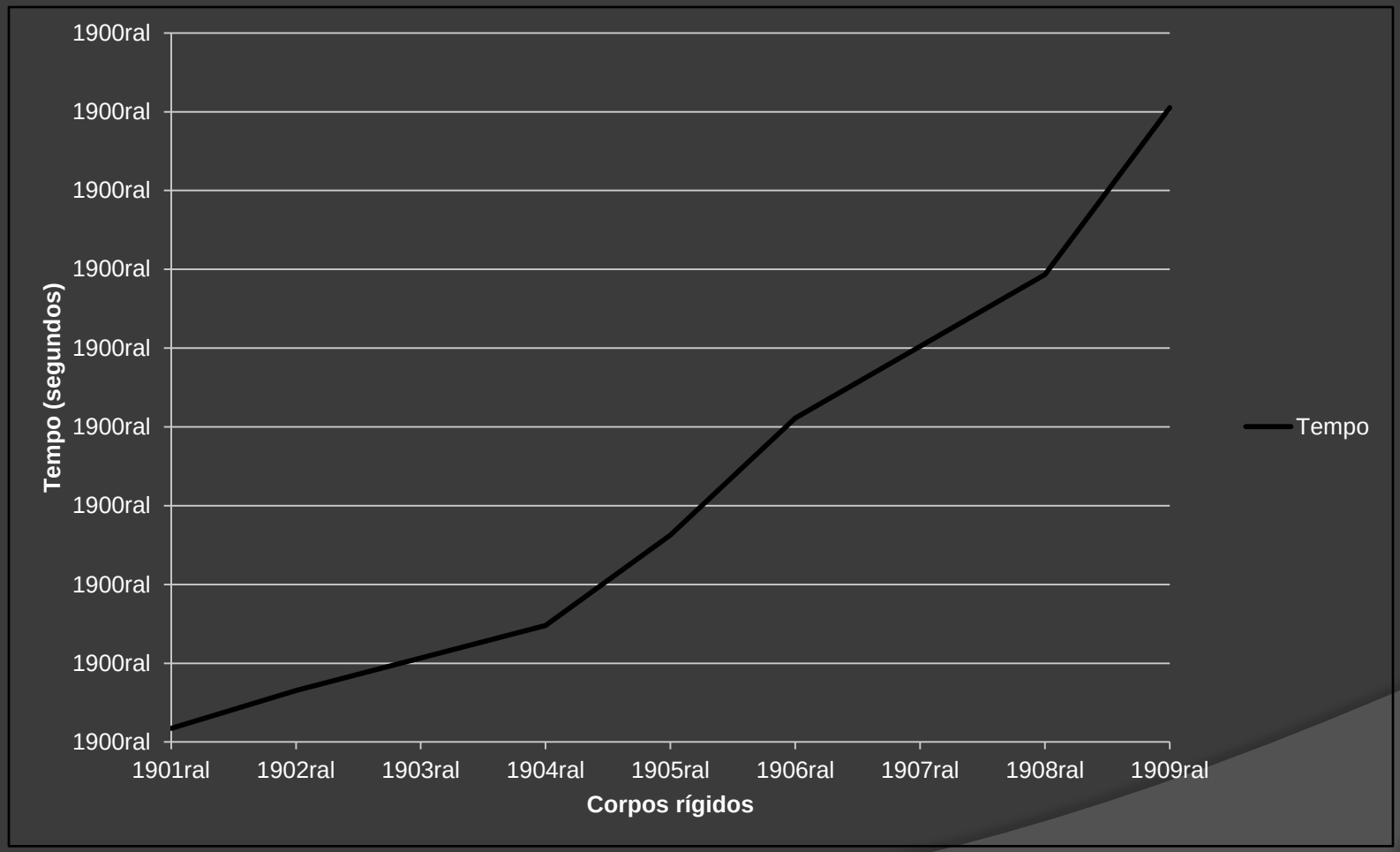
- Newton-Stormer-Verlet
- XenoCollide
- *Timestep* fixo
- Robustez

Resultados e discussões



CPU: Intel Celeron SU2300 Memória: 2GB GPU: Intel GMA 4500MHD

Resultados e discussões



Conclusão

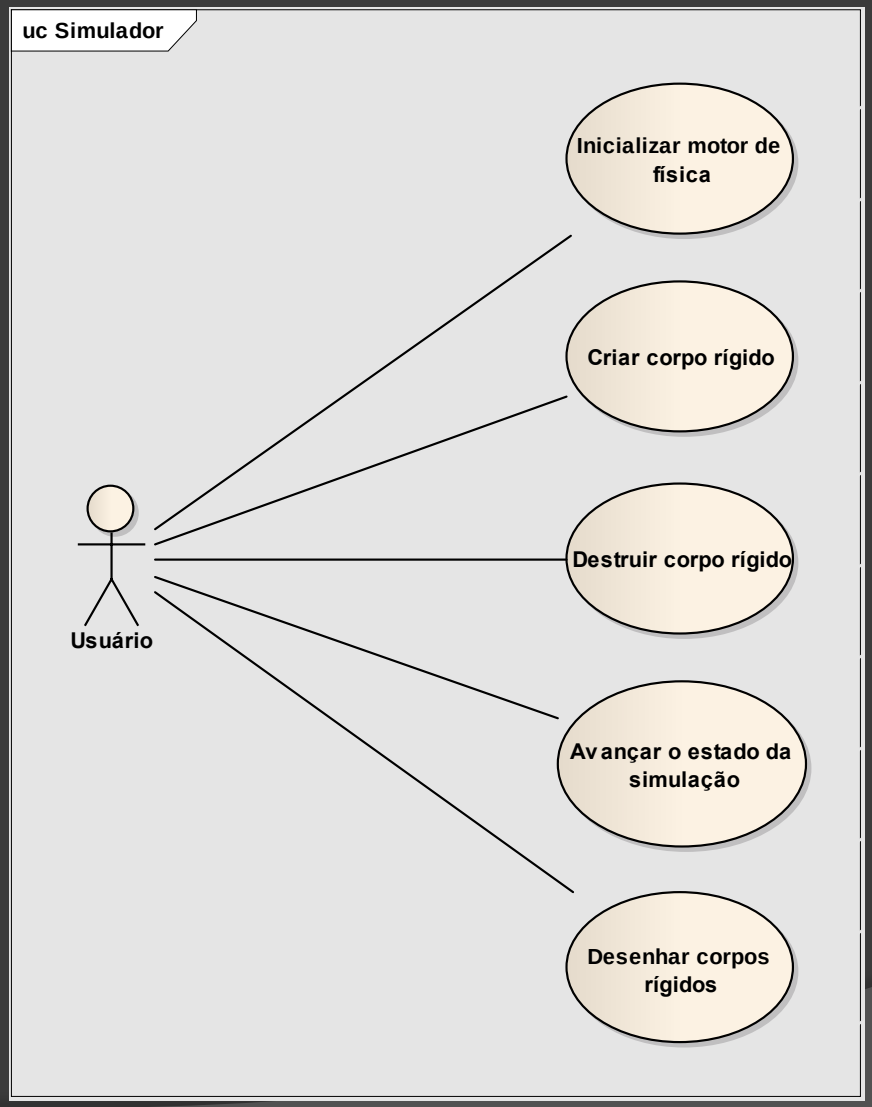
- Detecção da colisão
- Tratamento da colisão
- Simulação
- Aplicação exemplo

Extensões

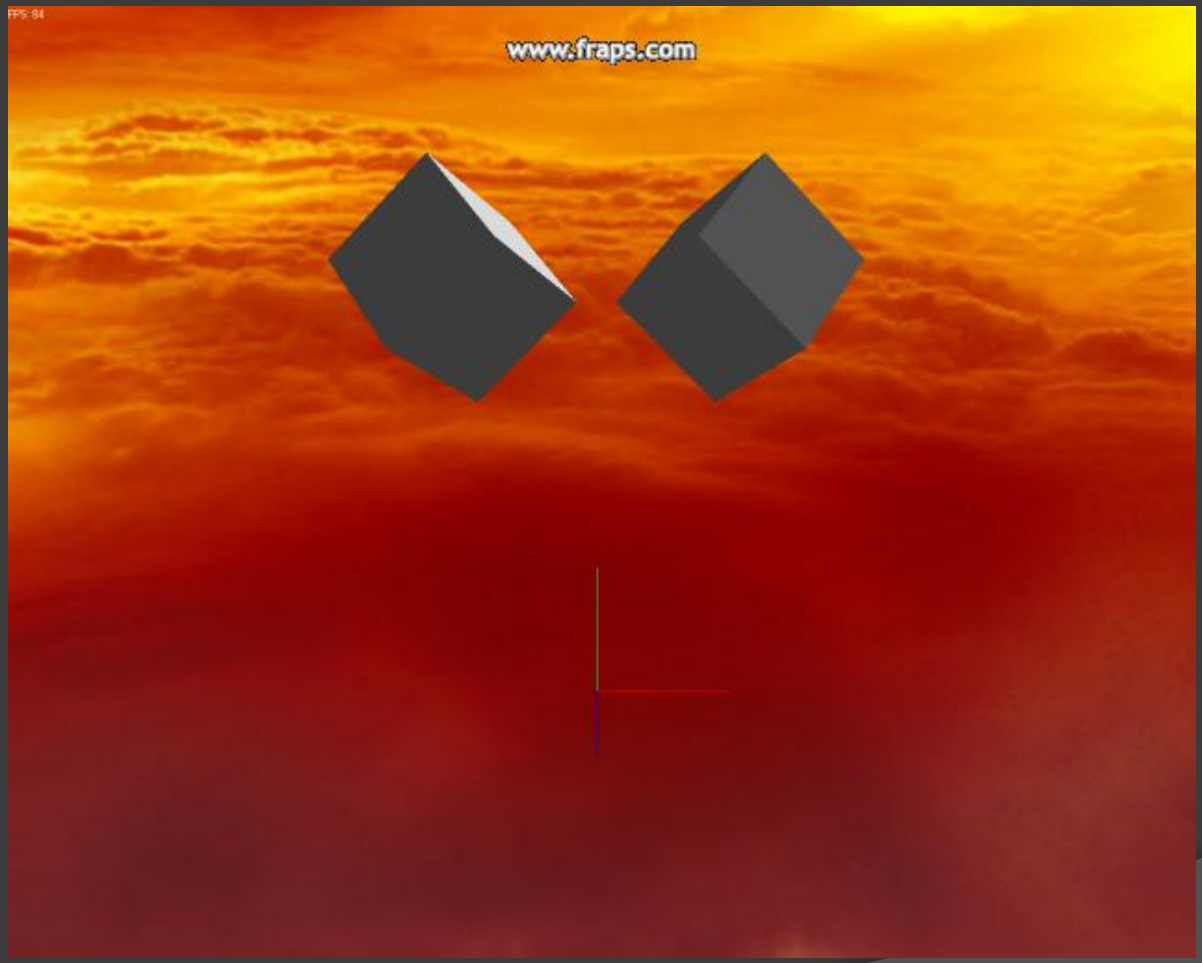
- Melhorar tratamento da colisão
- Melhorar desempenho
- Mais recursos

FIM

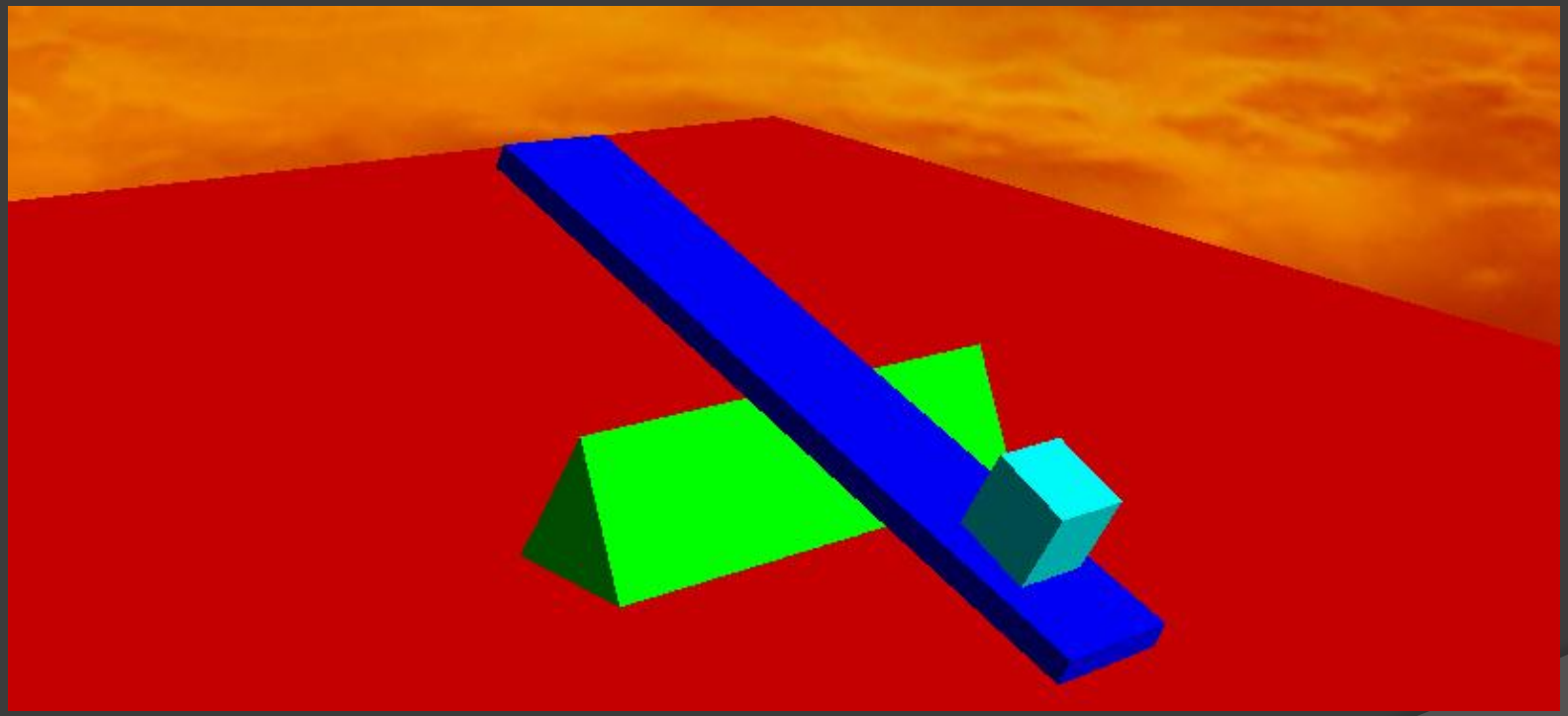
Diagrama de casos de uso



Operacionalidade



Operacionalidade



Operacionalidade

Código fonte

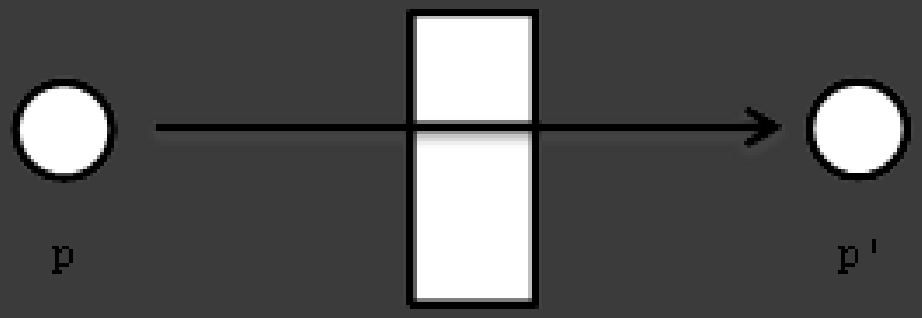
```
void PhysicsSystem::Step( float step )
{
    ++m_iTimestamp;

    // Integrate velocity and apply damping
    for each( RigidBody* pBody in m_Bodies )
    {
        // ...
        pBody->linearVelocity += pBody->invMass * pBody->force * step;
        pBody->angularVelocity += pBody->invInertiaTensor.TransposedTransform( pBody->torque ) * step;
        // ...
    }

    this->RefreshContacts();
    this->FindContacts();
    this->ClearOldContacts();
    this->SolveCollisions( step );

    for each( RigidBody* pBody in m_Bodies )
    {
        pBody->position += pBody->linearVelocity * step;

        pBody->orientation += (pBody->orientation * Quaternion( 0, pBody->angularVelocity )) * step * 0.5f;
        pBody->orientation.Normalize();
    }
}
```

Fundamentação teórica

- Robustez numérica
  - Inexatidão
    - Representação de ponto flutuante
    - Objetos de tamanho desproporcional
    - Entre outros
  - Casos degenerados
    - Vértices iguais
    - Arestas colineares
    - Entre outros

Fundamentação teórica

- Detecção de colisão
  - Pontos de contato, profundidade de penetração e normal da colisão
  - Discreta ou contínua
    - *Tunneling*



Fundamentação teórica

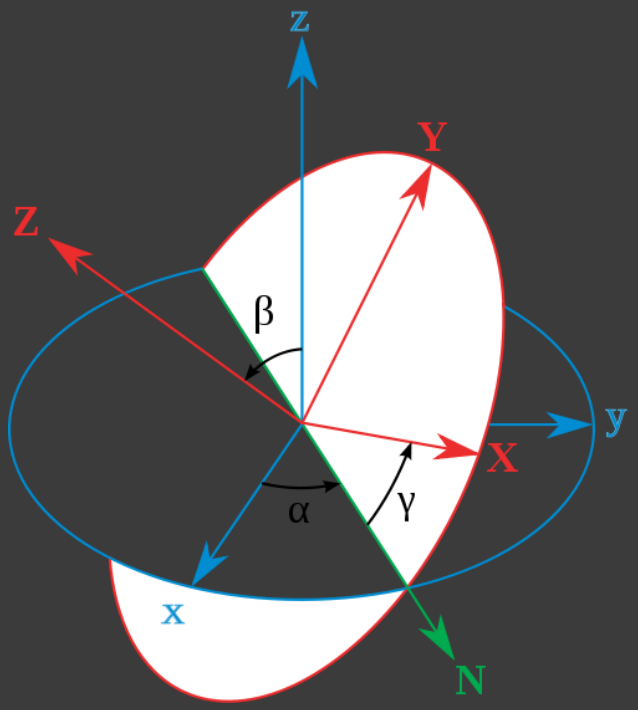
- Algoritmo XenoCollide
  - Resultado do teste
    - Resultado booleano
    - Normal de contato, ponto de contato e profundidade de penetração

Implementação

- Técnicas utilizadas
  - Visual Studio 2010
  - C++
  - OpenGL

Fundamentação teórica

- Orientação em 3D
  - Ângulos de Euler
    - Menor representação possível
    - Fácil entendimento
    - *Gimbal lock*



Fundamentação teórica

- Orientação em 3D
  - Matriz de rotação
    - Ortogonal
    - A inversa desfaz a rotação
    - *Matrix creep*
    - Não é intuitiva

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$