

SIMULAÇÃO DE FÍSICA CLIENT-SIDE APLICADA A SIMULAÇÃO DE PROJÉTEIS

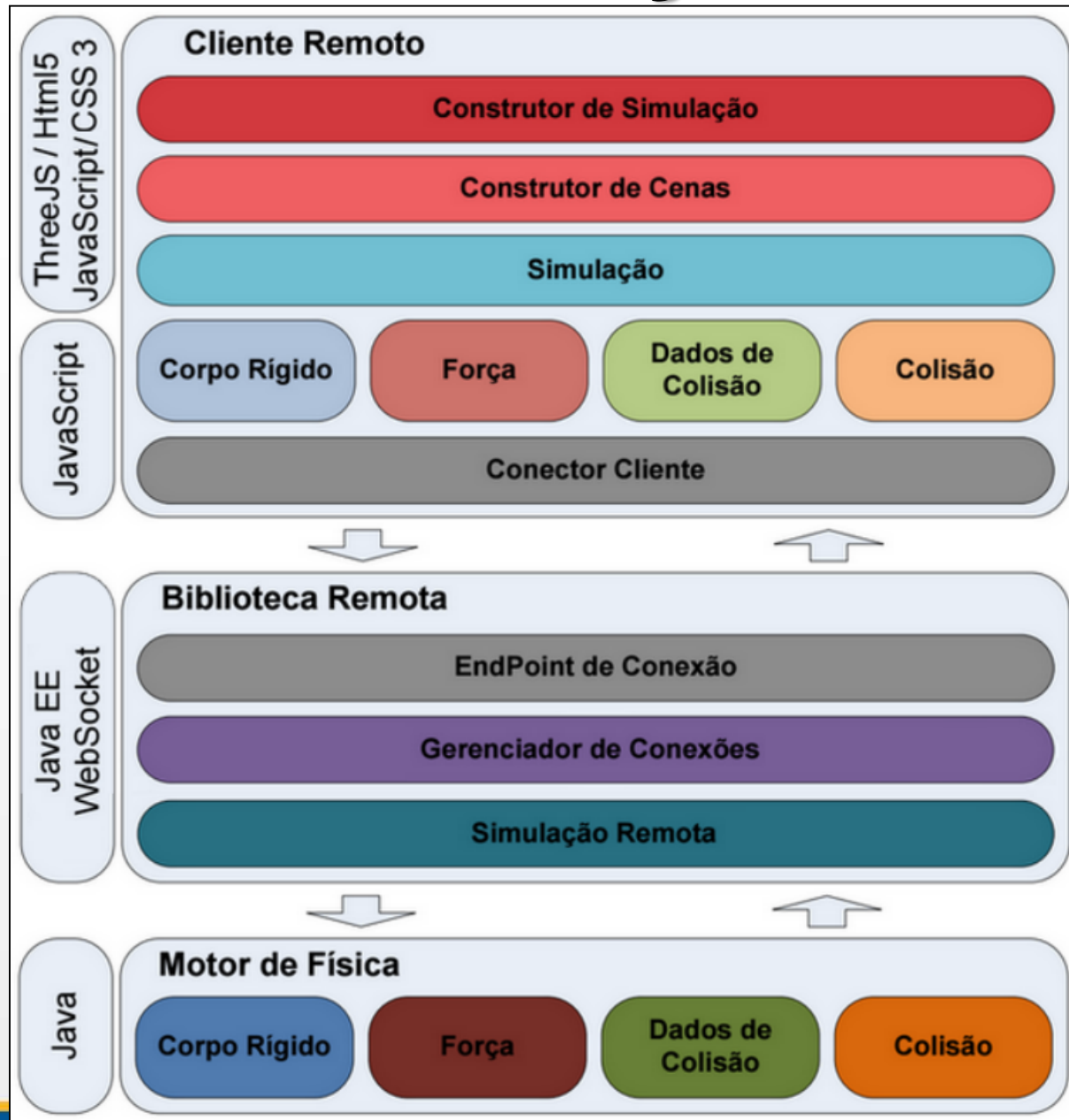
Aluno(a): Sérgio Luiz Tomio Junior

Orientador: Dalton Solano dos Reis

Roteiro

- Introdução
- Objetivos
- Fundamentação teórica
- Trabalhos correlatos
- Requisitos
- Especificação
- Implementação
- Operacionalidade da implementação
- Resultados e conclusões
- Discussões e sugestões

Introdução

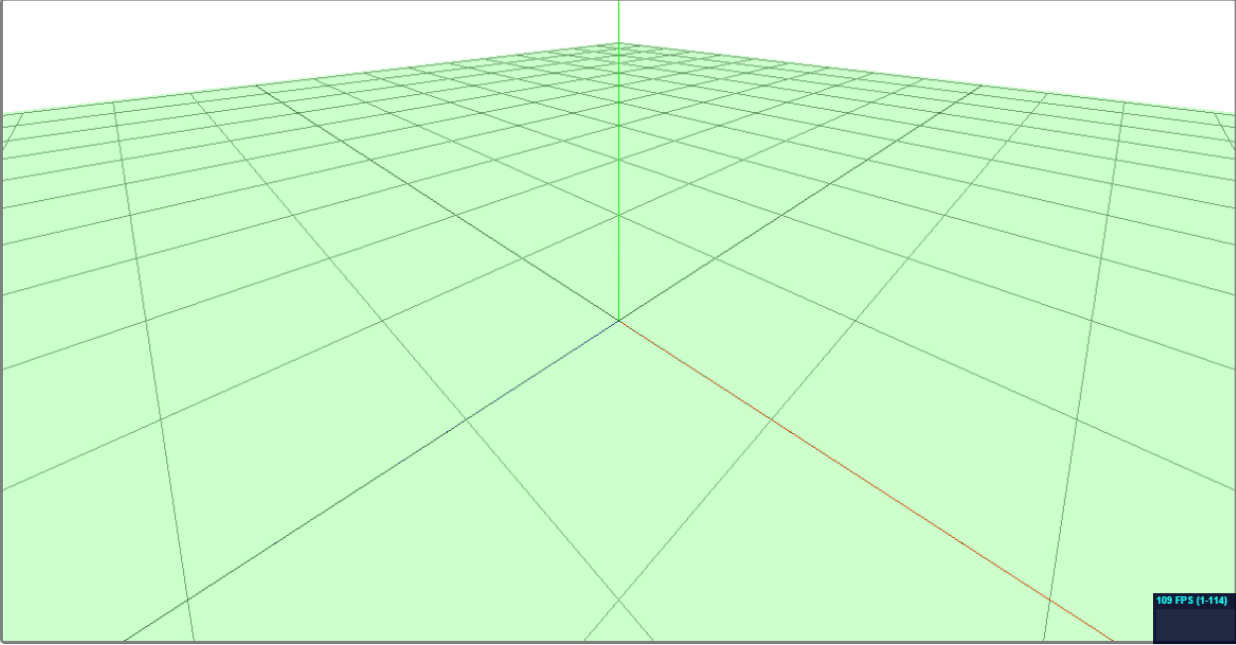


Introdução

VisEdu

Arquivo ▾ Criar Objetos ▾ Força Canhão Bolas do Canhão || ▶ Teste

Objetos



Propriedades:

Propriedades Cena:

Forças

Canhão

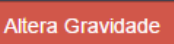
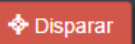
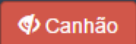
- Velocidade Inicial: 100m/s
- Posições: x:10 y:20 z:40

109 FPS (1-114)

X: X Y: Y Z: Z Mudar Posição

X: 20.00 Y: 23.00 Z: 20.00 Mudar Câmera ↻

The image shows a 3D visualization interface for a physics simulation. The main area is a green grid representing a 3D space. A red line is drawn across the grid, likely representing a trajectory or a force vector. The interface includes a top menu bar with buttons for 'Arquivo', 'Criar Objetos', 'Força', 'Canhão', 'Bolas do Canhão', and 'Teste'. On the left, there is a panel labeled 'Objetos'. On the right, there are panels for 'Propriedades' and 'Propriedades Cena'. The 'Propriedades Cena' panel lists 'Forças' and 'Canhão' with specific parameters: 'Velocidade Inicial: 100m/s' and 'Posições: x:10 y:20 z:40'. A FPS counter in the bottom right of the grid shows '109 FPS (1-114)'. At the bottom, there are two control panels: one for changing position (X, Y, Z) and another for changing camera (X, Y, Z).



Força elástica (N)
 $F = K \cdot x$ — Deformação da mola (M)
Constante Elástica (N/M)

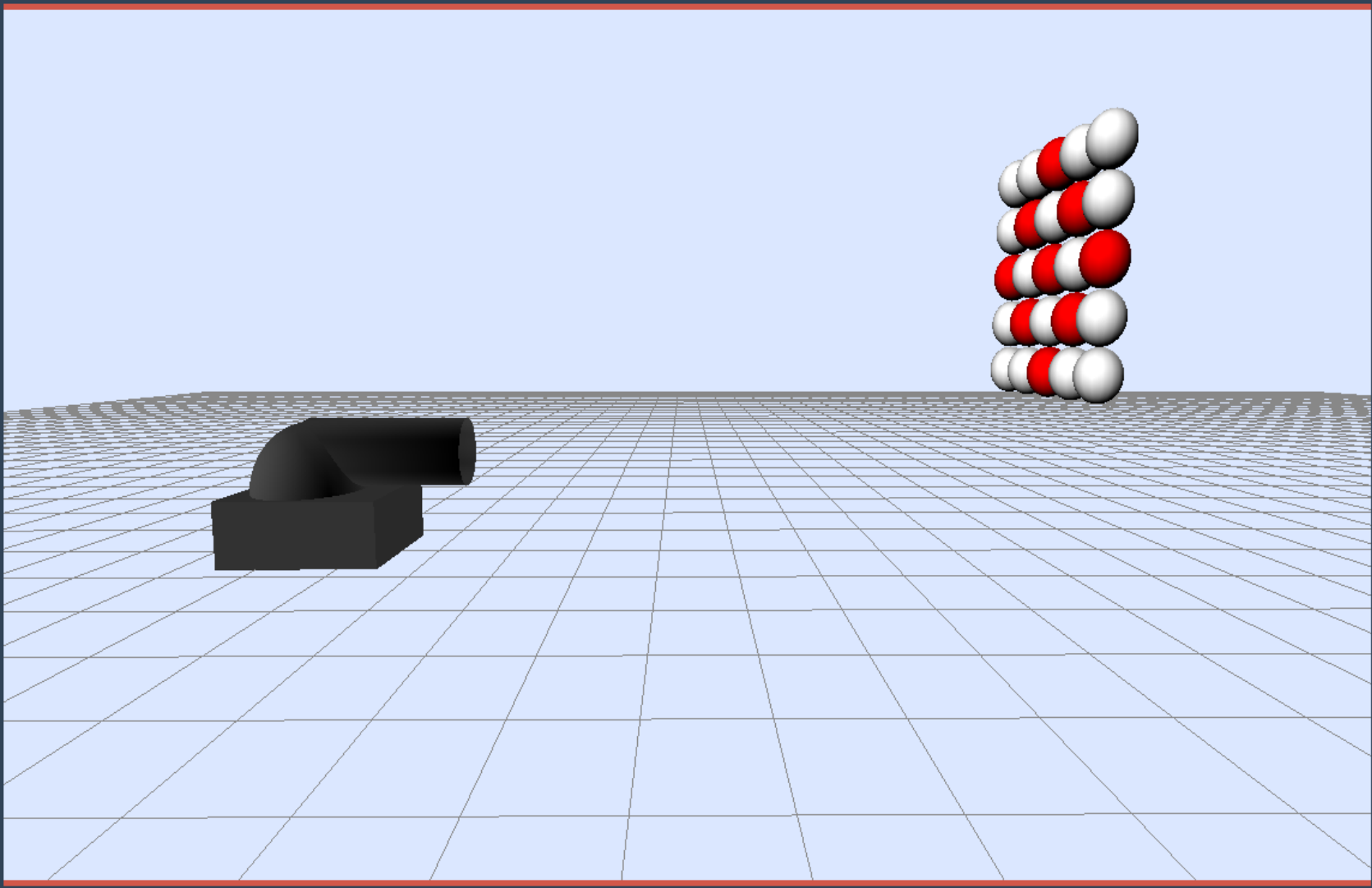
K: x:

Material do projétil:

Atual

Ângulo do canhão:

↑
↓

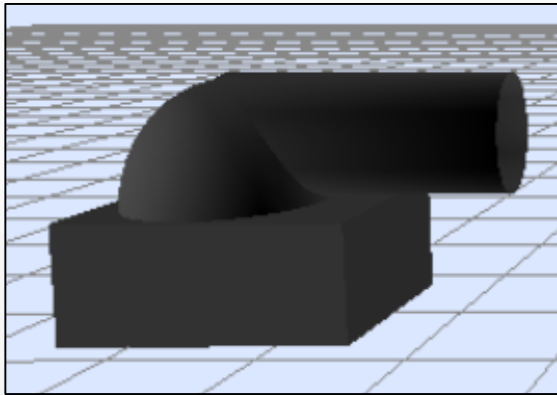


Objetivos

- O objetivo deste trabalho é construir um adaptador entre a interface do motor de física CannonJS e a interface do motor de física Hefesto.
- Os objetivos específicos são:
 - integrar o motor adaptado com a aplicação web Ballistic (ZANLUCA, 2015),
 - comparar a performance do motor adaptado com o motor Hefesto, desenvolvido em Java.

Fundamentação Teórica

- Mecânica básica – Lei de Hooke



$$F = K \cdot x$$

Força elástica (N) K: x:
Deformação da mola (M) 1 1
Constante Elástica (N/M)

Fundamentação Teórica

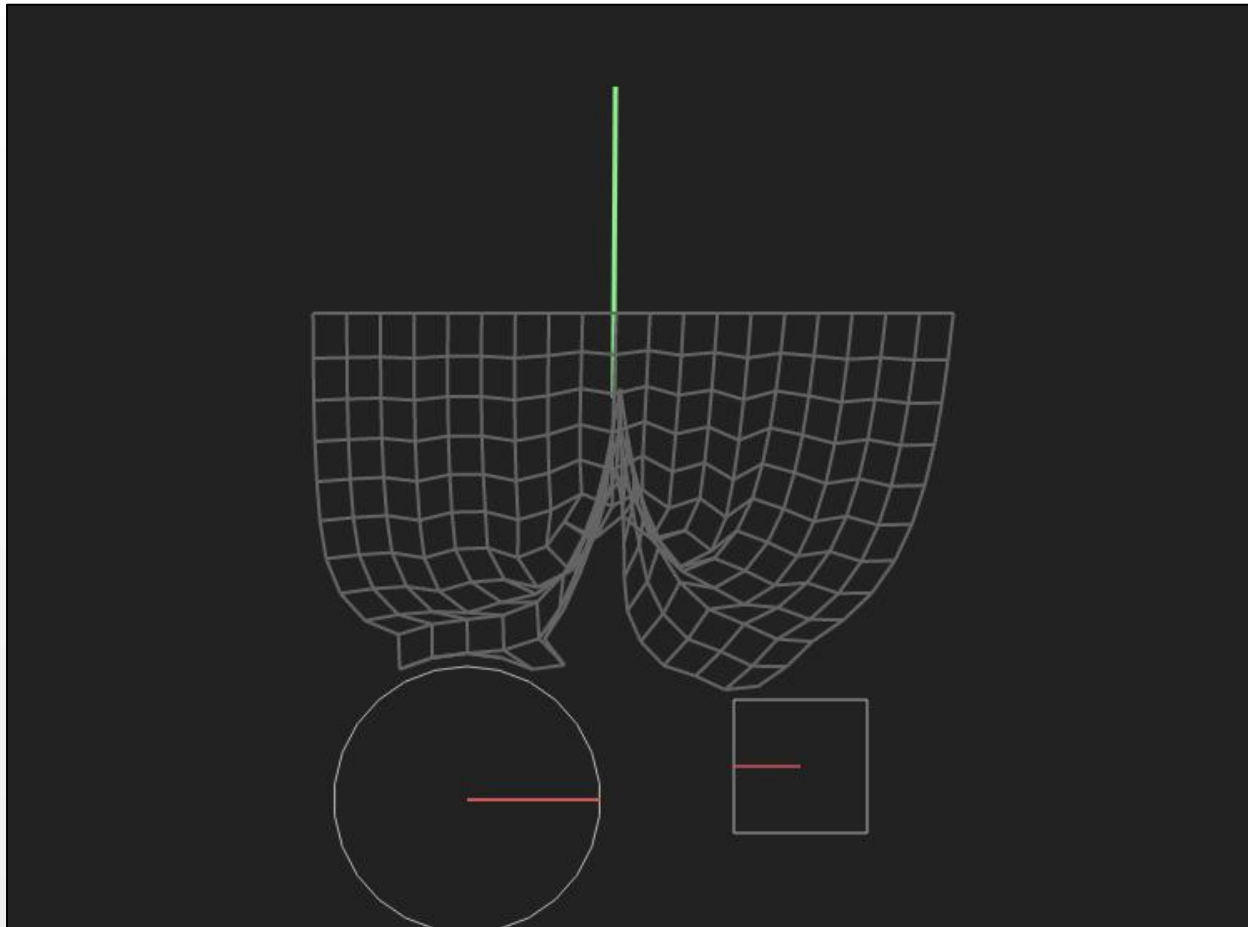
- Simulação de física
 - Motores de física
 - Aplicações
- Desenvolvimento Web
 - HTML, CSS e Javascript
 - Javascript – scripting e herança prototipal sem um sistema rígido de tipos

Fundamentação Teórica

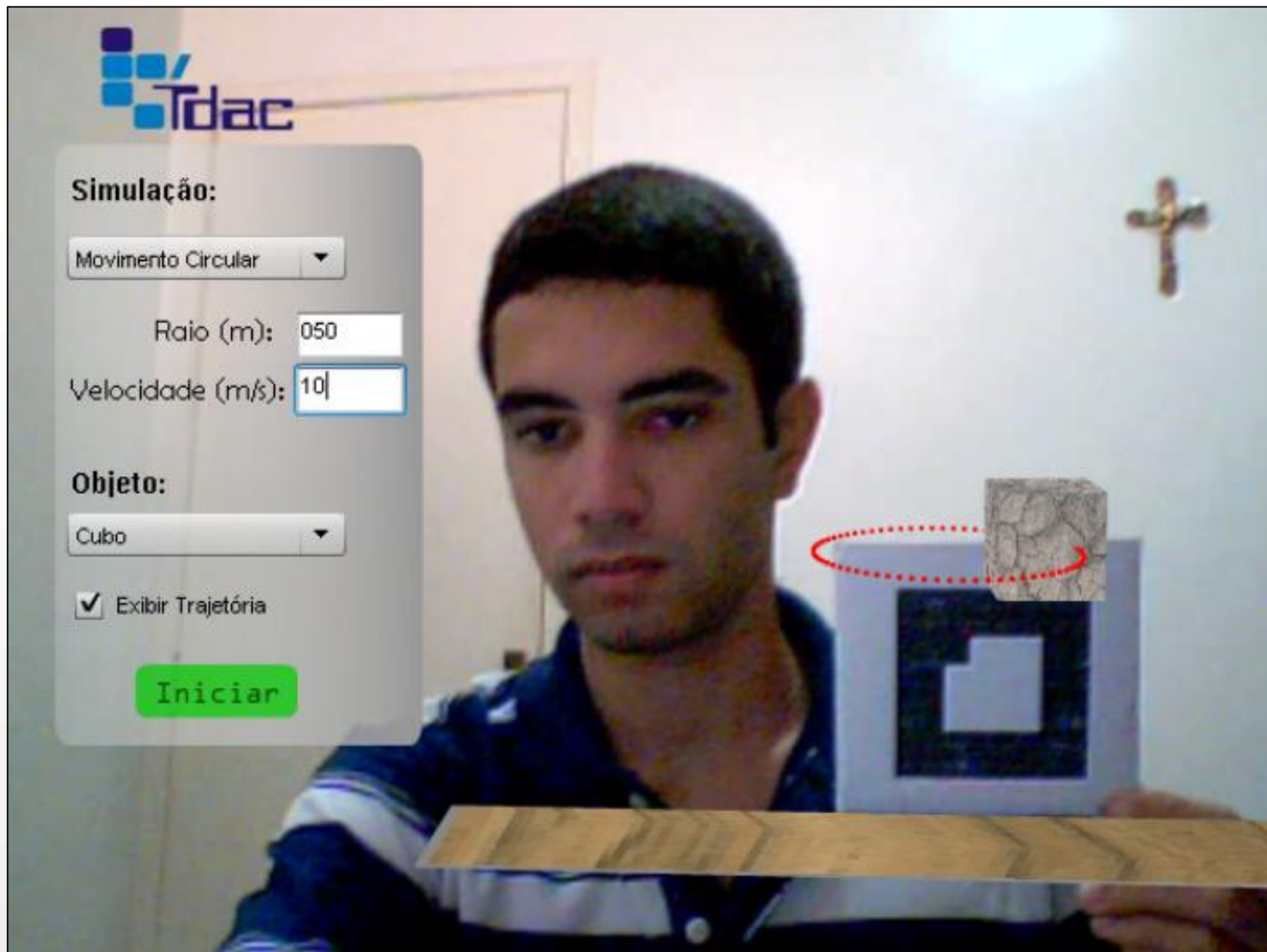
- Padrões de projeto
 - Adapter
 - Abstract factory
- CannonJS
 - Semelhanças com o Hefesto
 - Diferenças no tratamento de colisões
 - Time step
 - Constraints
 - Constraint solver

Trabalhos Correlatos

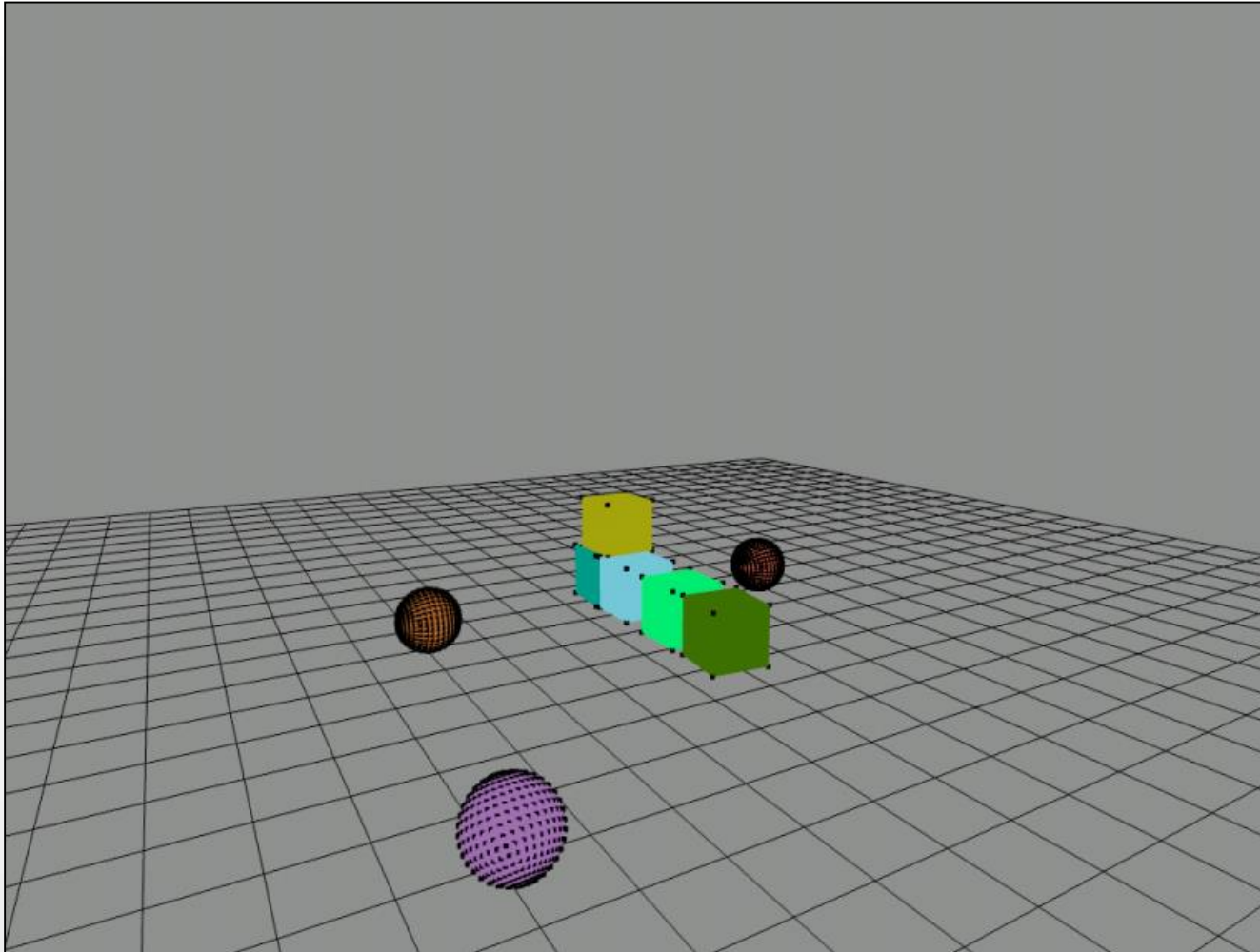
- MatterJS



- SimulAR



- Protótipo de uma ferramenta gráfica para criação de simulações na área da física em dispositivos móveis



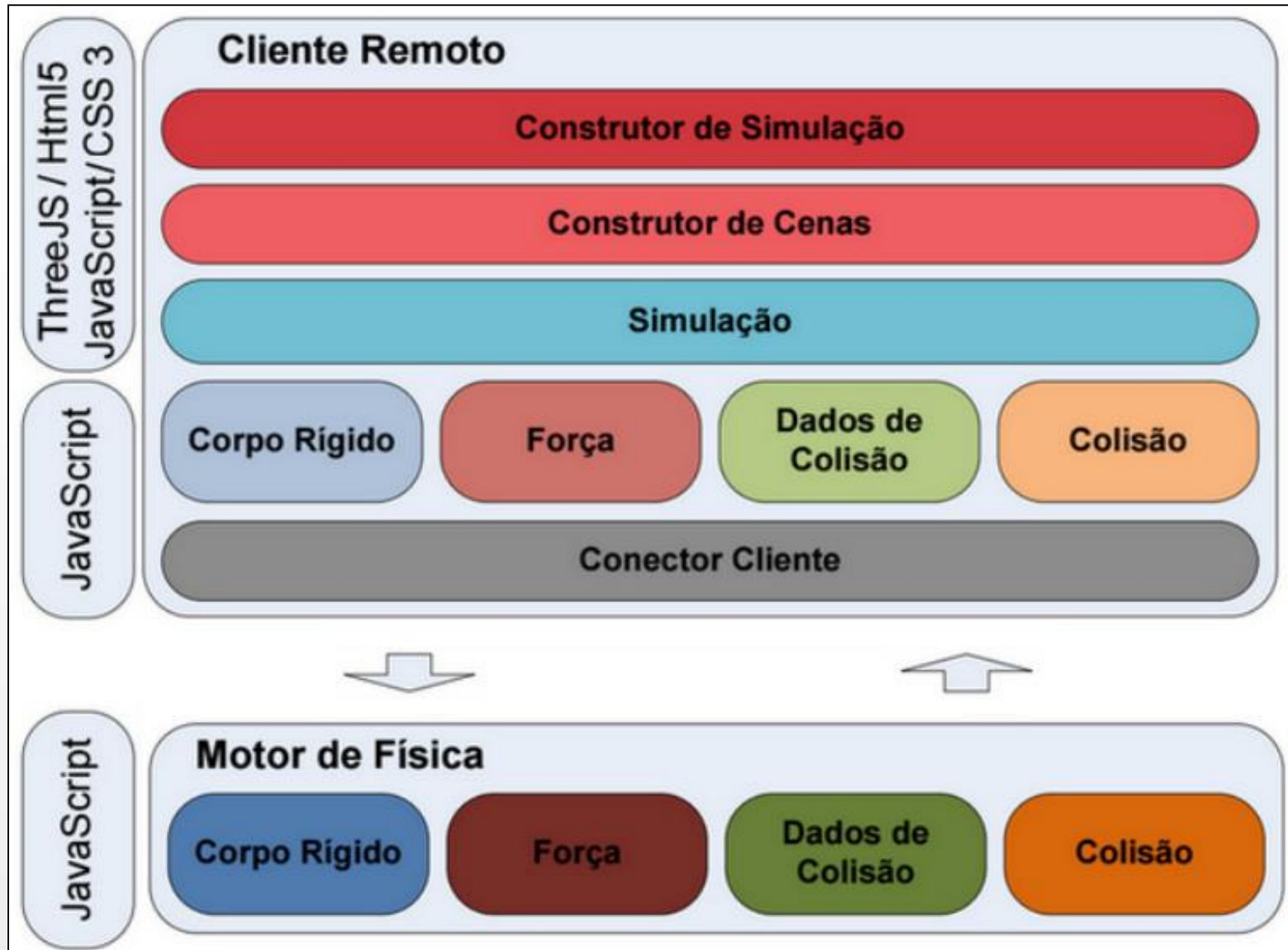
Requisitos

- O motor de física deverá:
 - criar corpo rígido e permitir configurar: aceleração, velocidade, rotação e massa (Requisito Funcional – RF),
 - realizar cálculos de integração dos corpos seguindo as leis da mecânica clássica (RF),
 - ser desenvolvido de forma que permita auto reuso e extensão (RNF),
 - utilizar a linguagem de programação Javascript (RNF),
 - funcionar independente de uma conexão com a internet (RNF).

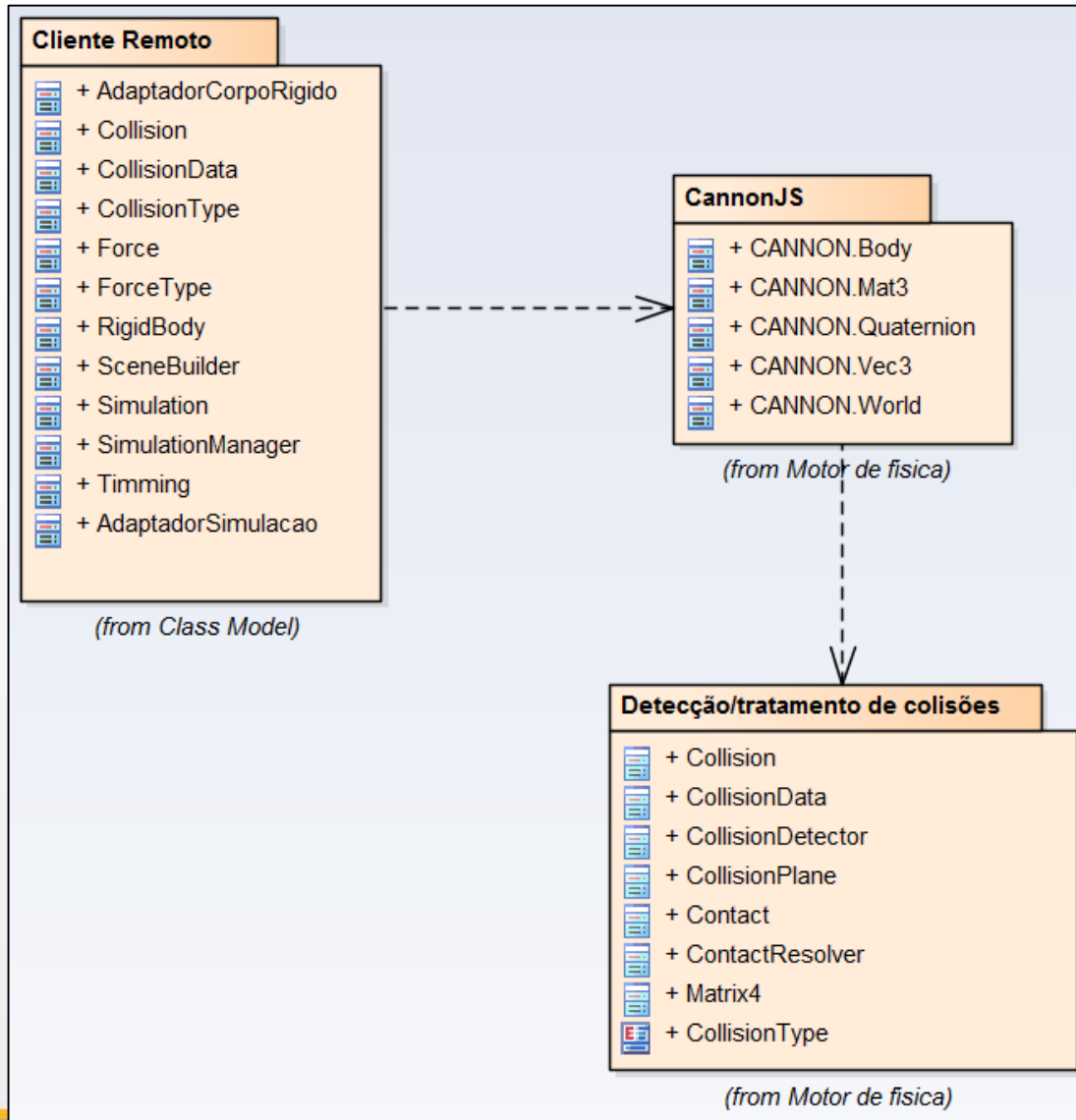
Requisitos

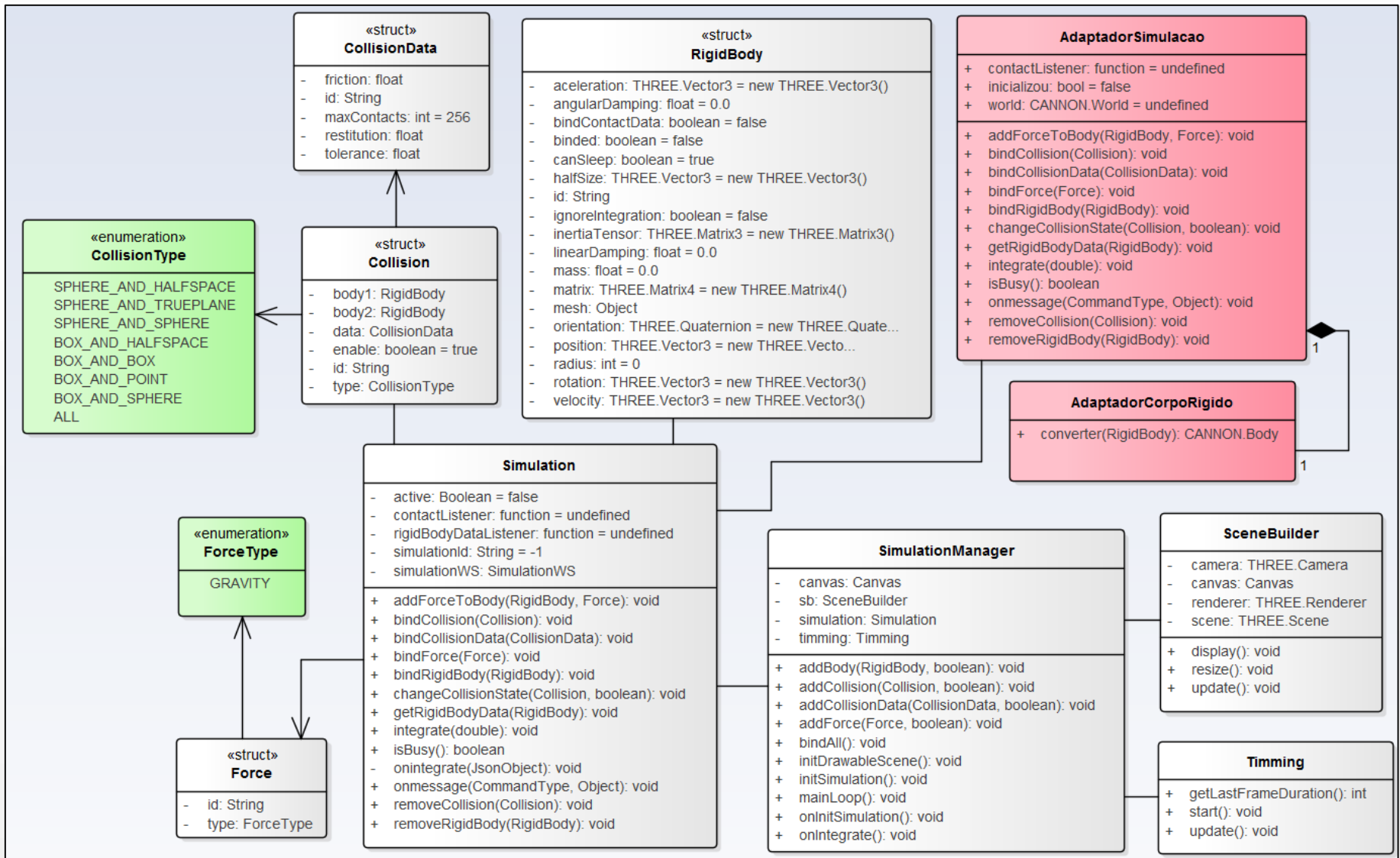
- O cliente remoto deverá:
 - utilizar o novo motor de física (RNF),
 - apresentar a mesma interface do cliente remoto desenvolvido por Teixeira (2015) (RNF),
 - ser desenvolvido na linguagem Javascript (RNF).

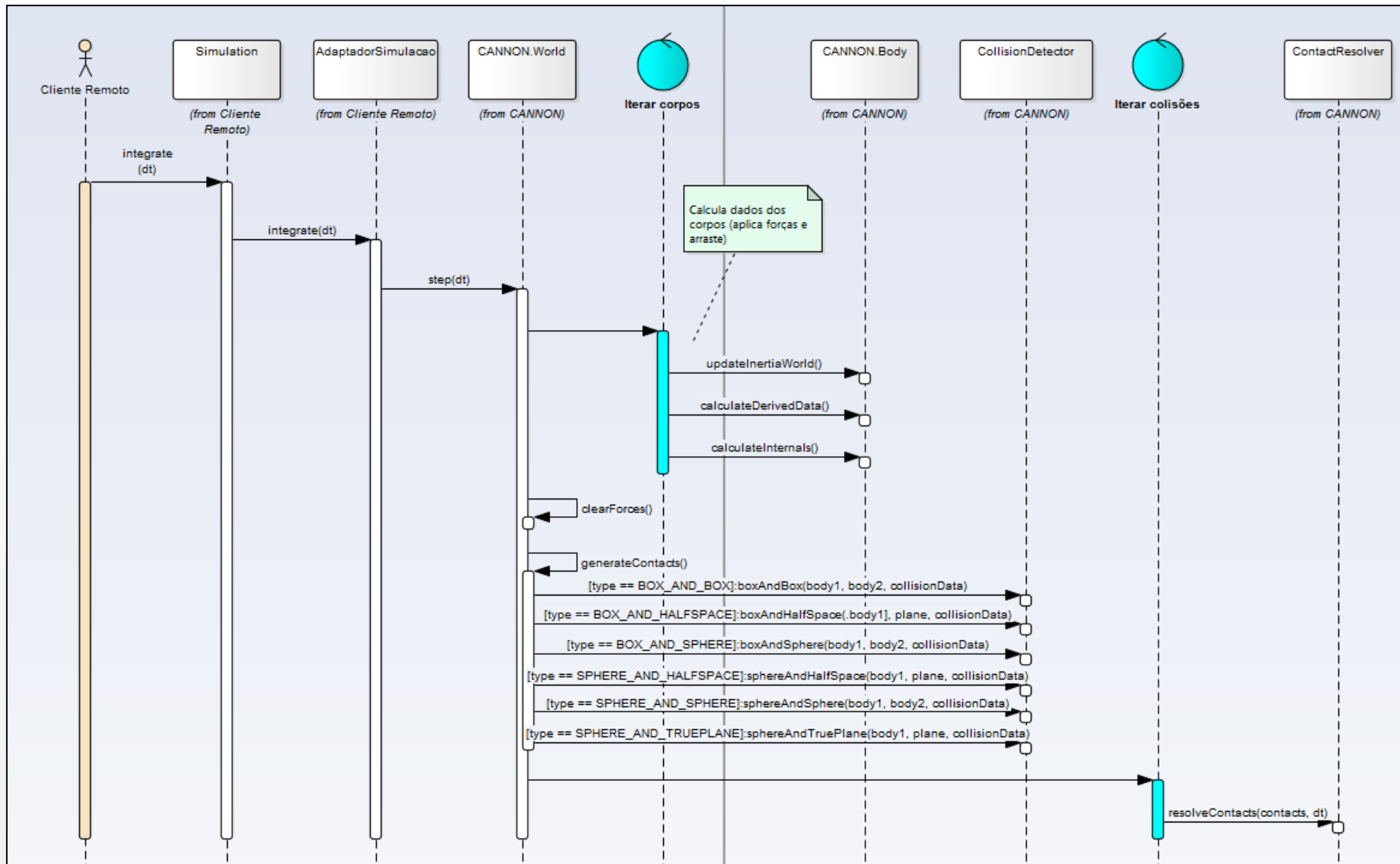
Especificação



Especificação







Implementação

```
102 bindRigidBody: function (body) {  
103     this._rigidBodys = this.adaptador.bindRigidBody(body, this._rigidBodys);  
104 },
```

```
20     this.bindRigidBody = function (body, corpos) {  
21         var corpoCannon = this.adaptadorCorpoRigido.converter(body);  
22  
23         corpoCannon.calculateDerivedData();  
24         corpoCannon.calculateInternals();  
25  
26         this.world.addBody(corpoCannon);  
27         this.world.bodiesIndex[corpoCannon.id] = corpoCannon;  
28  
29         corpos[body.id] = body;  
30         return corpos;  
31     },
```

```

3  this.converter = function(corpo)
4  {
5      var c = corpo;
6      var orientacao = new CANNON.Quaternion(-c.orientation.x, -c.orientation.y, -c.orientation.z, c.orientation.w);
7
8      var corpoCannon = new CANNON.Body({
9          mass: c.mass,
10         position: new CANNON.Vec3(parseFloat(c.position.x), parseFloat(c.position.y), parseFloat(c.position.z)),
11         velocity: new CANNON.Vec3(parseFloat(c.velocity.x), parseFloat(c.velocity.y), parseFloat(c.velocity.z)),
12         angularVelocity: new CANNON.Vec3(parseFloat(c.rotation.x), parseFloat(c.rotation.y), parseFloat(c.rotation.z)),
13         linearDamping: c.linearDamping,
14         angularDamping: c.angularDamping,
15         quaternion: orientacao,
16     });
17
18     corpoCannon.setInertiaTensor(c.inertiaTensor);
19     corpoCannon.ignoreIntegration = c.ignoreIntegration;
20     corpoCannon.useWorldForces = c.useWorldForces;
21     corpoCannon.id = c.id;
22
23     if (c.ignoreIntegration) {
24         corpoCannon.type = CANNON.Body.STATIC;
25     }
26
27     if (c.radius > 0) {
28         var forma = new CANNON.Sphere(c.radius);
29         corpoCannon.addShape(forma);
30     } else {
31         var hs = c.halfSize;
32         var forma = new CANNON.Box(new CANNON.Vec3(hs.x, hs.y, hs.z));
33         corpoCannon.addShape(forma);
34     }
35
36     if (c.bindContactData &&
37         c.bindContactData !== null) {
38         corpoCannon.bindContactData = c.bindContactData;
39     }
40
41     aceleracaoX = parseFloat(c.acceleration.x);
42     aceleracaoY = parseFloat(c.acceleration.y);
43     aceleracaoZ = parseFloat(c.acceleration.z);
44
45     corpoCannon.acceleration = new CANNON.Vec3(aceleracaoX, aceleracaoY, aceleracaoZ);
46
47     return corpoCannon;
48 }

```

```

86  this.integrate = function (duration, corpos) {
87      this.world.step(duration);
88      var corposCannon = Object.values(this.world.bodiesIndex);
89
90      for (var i = 0; i < corposCannon.length; i++) {
91          var _rb = corposCannon[i];
92          var rb = corpos[_rb.id];
93          var p = new THREE.Vector3();
94          p.x = _rb.position.x;
95          p.y = _rb.position.y;
96          p.z = _rb.position.z;
97          rb.position = p;
98          // caso nao possua mesh, nao precisa ajustar valores de exibicao
99          if (rb.mesh == undefined) {
100              continue;
101          }
102          rb.mesh.position.x = p.x;
103          rb.mesh.position.y = p.y;
104          rb.mesh.position.z = p.z;
105          //tentar setar o orientacao e posicao
106          rb.mesh.matrix.identity();
107          var utils = new AdaptadorUtils();
108          var a = utils.calcularMatrizTransformacao(_rb.position, _rb.quaternion);
109          var m = new THREE.Matrix4();
110          m.set(
111              a[0], a[1], a[2], a[3], a[4], a[5], a[6], a[7],
112              a[8], a[9], a[10], a[11], a[12], a[13], a[14], a[15]);
113          rb.mesh.applyMatrix(m);
114          rb.mesh.updateMatrix();
115          rb.mesh.matrix.setPosition(p);
116          if (this._contactListener != undefined && this._contactListener != null) {
117              var contacts = this.world.contacts;
118              for (j = 0; j < contacts.length; j++) {
119                  this._contactListener(contacts[j]);
120              }
121          }
122          log('Integrate simulation: ' + this._simulationId);
123      }
124      return corpos;
125  },

```

Operacionalidade da Implementação

The screenshot displays the 'Ballistic' simulation interface. At the top left is the 'Ballistic' logo. The top navigation bar includes buttons for 'Canhão', 'Disparar', 'Altera Gravidade', 'Tutorial', 'Sobre', and a settings icon. The control panel features a spring constant $F = K \cdot x$ with input fields for K (value 1) and x (value 1). The 'Material do projétil' section shows a selection of materials with a 'Atual' button. The 'Ângulo do canhão' is set to 0 with up/down arrows. A red stop button is also present. The main 3D view shows a black cannon on a grid floor and a stack of red and white balls.

Operacionalidade da Implementação

The screenshot displays the VisEdu software interface. At the top, there is a menu bar with 'VisEdu' and 'Arquivo -'. Below the menu bar are several buttons: 'Criar Objetos -', 'Força', 'Canhão', 'Bolas do Canhão', a play button, and 'Teste'. The main area is a 3D grid with a green background and a red line. On the left, there is a panel labeled 'Objetos'. On the right, there are two panels: 'Propriedades:' and 'Propriedades Cena:'. The 'Propriedades Cena:' panel contains the following information:

Propriedades:

Propriedades Cena:

Forças

Canhão

- Velocidade Inicial: 100m/s
- Posições: x:10 y:20 z:40

At the bottom, there are two input panels. The left panel has 'X: X', 'Y: Y', 'Z: Z' and a 'Mudar Posição' button. The right panel has 'X: 20.00', 'Y: 23.00', 'Z: 20.00' and a 'Mudar Câmera' button. A small status bar in the bottom right corner of the 3D view shows '109 FPS (1-114)'.

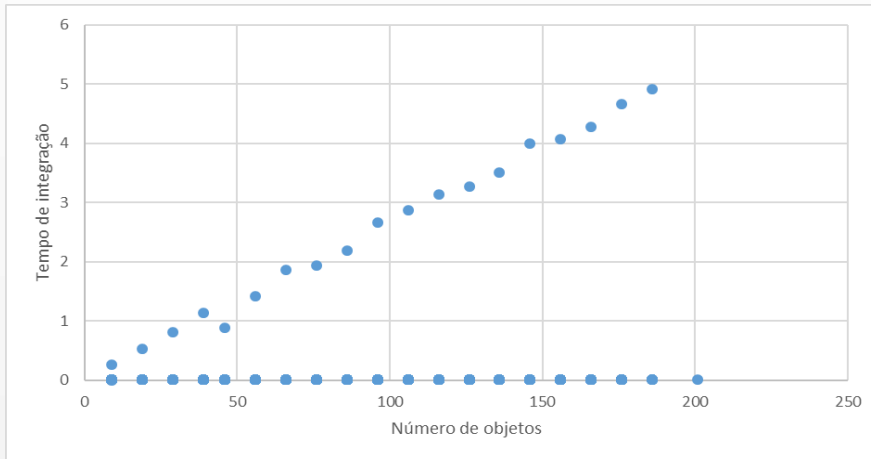
Resultados e Discussões

- Testes de performance realizados
 - coleta de dados alterando o código da aplicação para testes de tempo de integração,
 - testes de consumo de memória RAM foram realizados com a ferramenta Process Explorer.

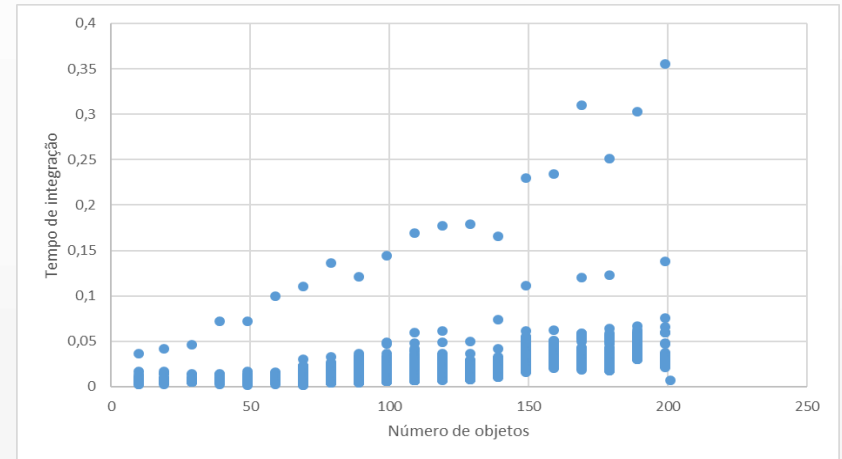
Resultados e Discussões

- Gráfico de tempo de integração/número de objetos no mundo

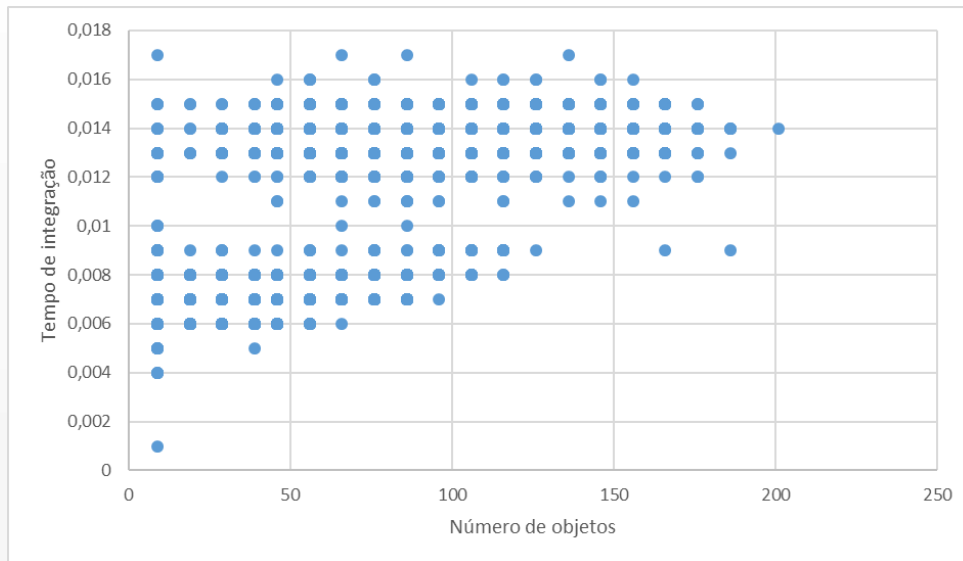
Antigo



Novo



Resultados e Discussões



- Motor antigo sem picos no tempo de integração causados pelo *bind* dos corpos no servidor

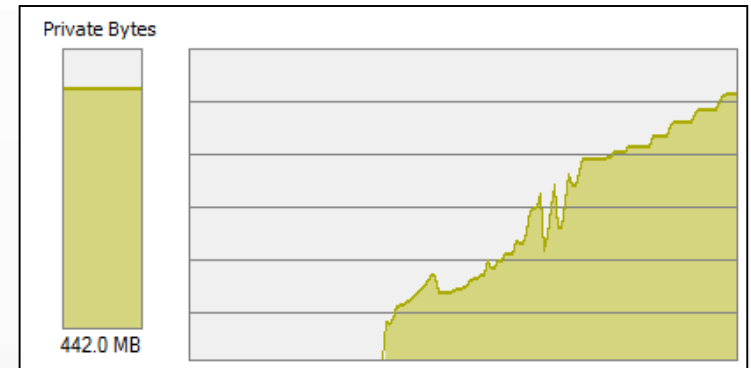
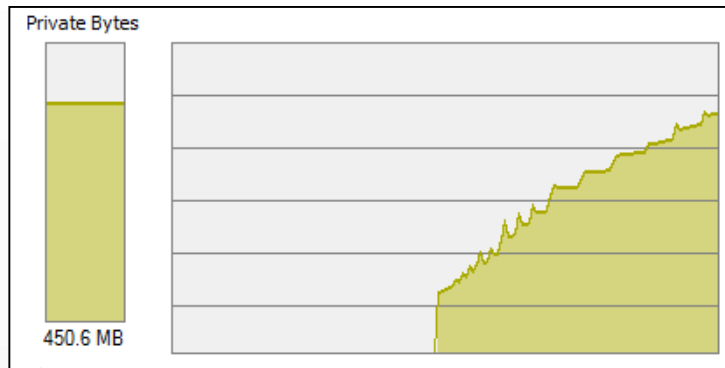
Resultados e Discussões

- Consumo de memória RAM (capturado com a ferramenta Process Explorer).

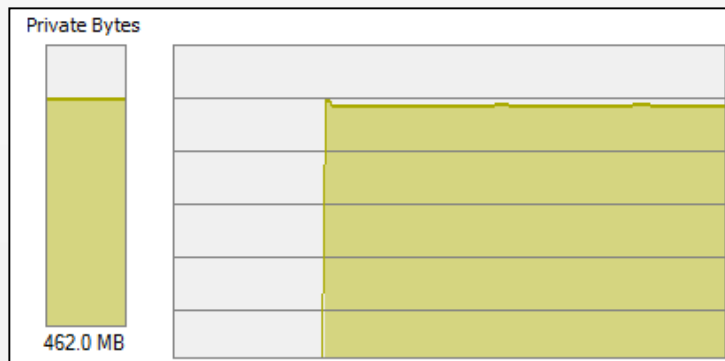
Antigo

Novo

chrome.exe



javaw.exe



Resultados e Discussões

| Característica | Brummit (2014) | Ferreira et al. (2011) | Silva (2012) | Hefesto Javascript (2017) |
|--------------------------------|----------------|------------------------|--------------|---------------------------|
| Customização de cenas 3D | - | - | X | X |
| Motor de física próprio | X | ? | X | X* |
| Executa em dispositivos móveis | X** | - | X | X** |
| Executa na Web | X | - | - | X |
| Possui aplicação própria | - | X | X | X |
| Usa realidade aumentada | - | X | - | - |

* - Junção de dois motores;

** - Executa na web logo pode executar em mobile

Conclusões e Sugestões

- Melhoria definitiva de performance na simulação de física, especialmente com muitos objetos
- Ausência de um servidor trouxe ganho no consumo de memória RAM
- Resolução de problemas decorrentes da arquitetura do trabalho anterior
- Trabalho não adicionou funcionalidade, logo limitações funcionais apontadas por Teixeira ainda se aplicam
- Somente utiliza um núcleo do processador

Demonstração prática

