Motor para jogos 2D utilizando HTML5

MARCOS HARBS

ORIENTADOR: DALTON SOLANO DOS REIS

FURB – UNIVERSIDADE REGIONAL DE BLUMENAU

Roteiro

- Introdução
- Objetivos
- Fundamentação teórica
- Desenvolvimento
- Resultados e discussão
- Conclusão
- Extensões

Introdução

- HTML5
  - Javascript, CSS e HTML
  - *Canvas*
- Motores de jogos
  - Surgiu em meados dos anos 90 com jogos de FPS
  - Vários gêneros
  - Abstrai a parte comum do desenvolvimento

Objetivos

- Visualizar o funcionamento do motor e editor de jogos
- Visualizar a execução do jogo Tangram
- Apresentar performance da ferramenta

Fundamentação teórica

- Jogos eletrônicos
- Motores de jogos
- HTML5 e Javascript
- Box2DJS
- Kinect e Zigfu
- Tangram
- Trabalhos correlatos

Jogos Eletrônicos

- Experiência interativa
- Desafios e aprendizagem
- Vários tipos de jogos
- Mundo virtual manipulado pelo computador
- Aproximação e simplificação
- Simulações temporais
- Respostas em tempo real

Motores de Jogos

- Abstração de tarefas comuns
- Conjunto de ferramentas
- Construído em camadas
- Problema de alto acoplamento
- Solução da orientação a componentes

HTML5 e Javascript

- HTML5
  - Será o novo padrão para HTML
  - Elemento *canvas*
- Javascript
  - Manipula elementos HTML
  - Linguagem para web
  - Linguagem leve
  - Fracamente tipada
  - Fácil aprendizagem

Box2DJS

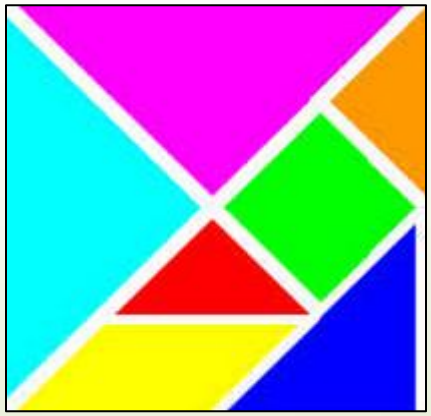
- Motor de simulações físicas de corpos rígidos
- Desenvolvida em Javascript
- Porte do motor Box2D desenvolvido em C++
- Código fonte aberto
- Detecção de colisão contínua
- Colisão por categorias e grupos
- Polígonos convexos, círculos e retângulos
- Contato, fricção e restituição
- Gravidade
- Pontos de junção

Kinect e Zigfu

- Kinect
  - Sensor de movimento
  - Criado pela Microsoft
- Zigfu
  - Biblioteca de comunicação com Kinect
  - Possui uma interface Javascript
  - Plugin para vários navegadores
  - Multi-plataforma
  - Licença de uso

Tangram

- Quebra cabeça geométrico
- Criado na China por volta de VII a.C.
- Formado por sete polígonos
- Pode-se formar uma grande variedade de representações geométricas, letras e figuras
- Apenas cenário de testes

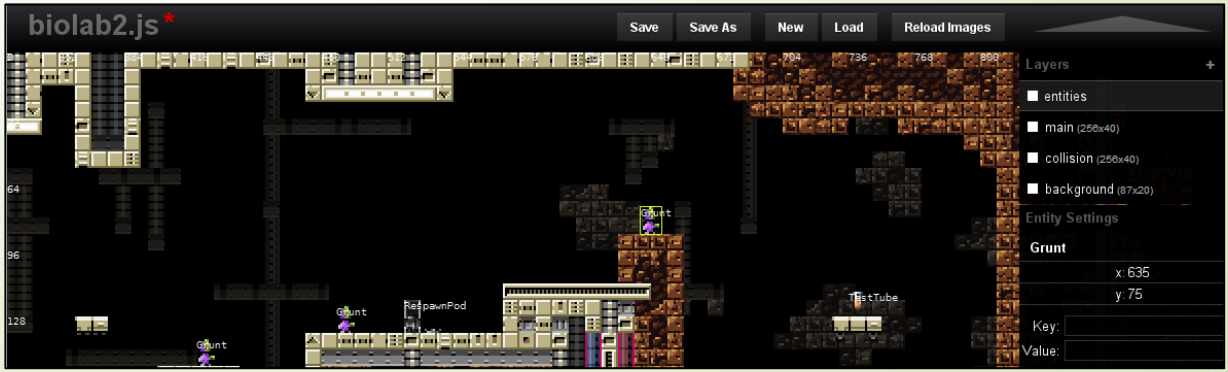


Trabalhos correlatos

- ImpactJS
- Unity 3D

ImpactJS

- Motor de jogos 2D Javascript e HTML5
- Utiliza a Box2D na camada de física
- Criação de níveis
- Detecção de colisão
- Gerenciamento de recursos
- Gerenciamento de camadas
- Renderização de objetos
- Dispositivos móveis
- É paga



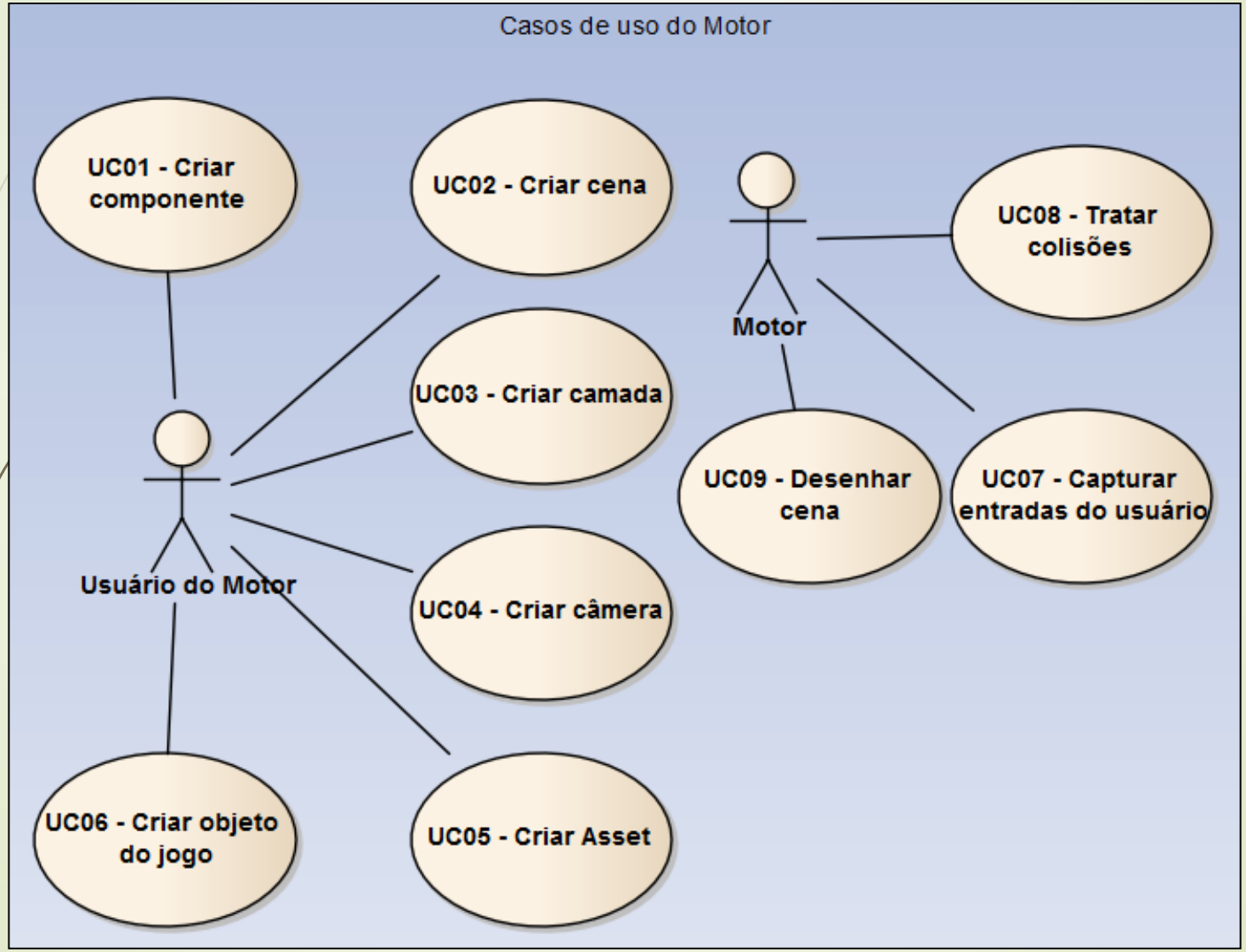
Unity 3D

- Motor e editor de jogos 3D
- Física de corpos rígidos
- Gerenciamento de recursos
- Orientada a componentes
- Gerenciamento de personagens
- Multi-plataforma
- Ferramentas de depuração
- Gerenciamento de animações
- É paga

Desenvolvimento

- Casos de uso
- Diagramas
- Implementação

Casos de uso do motor de jogos



Casos de uso do editor de jogos

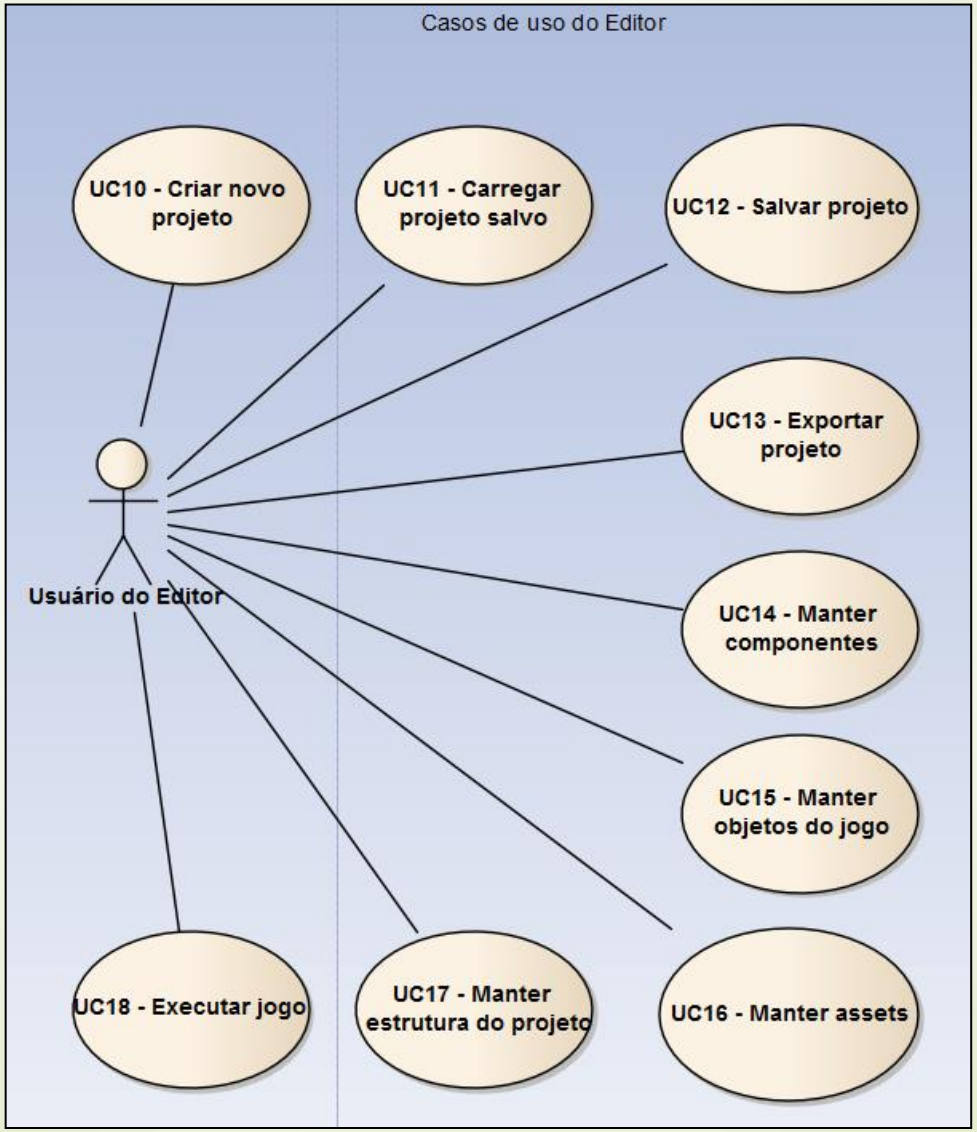


Diagrama: Pacotes do motor de jogos

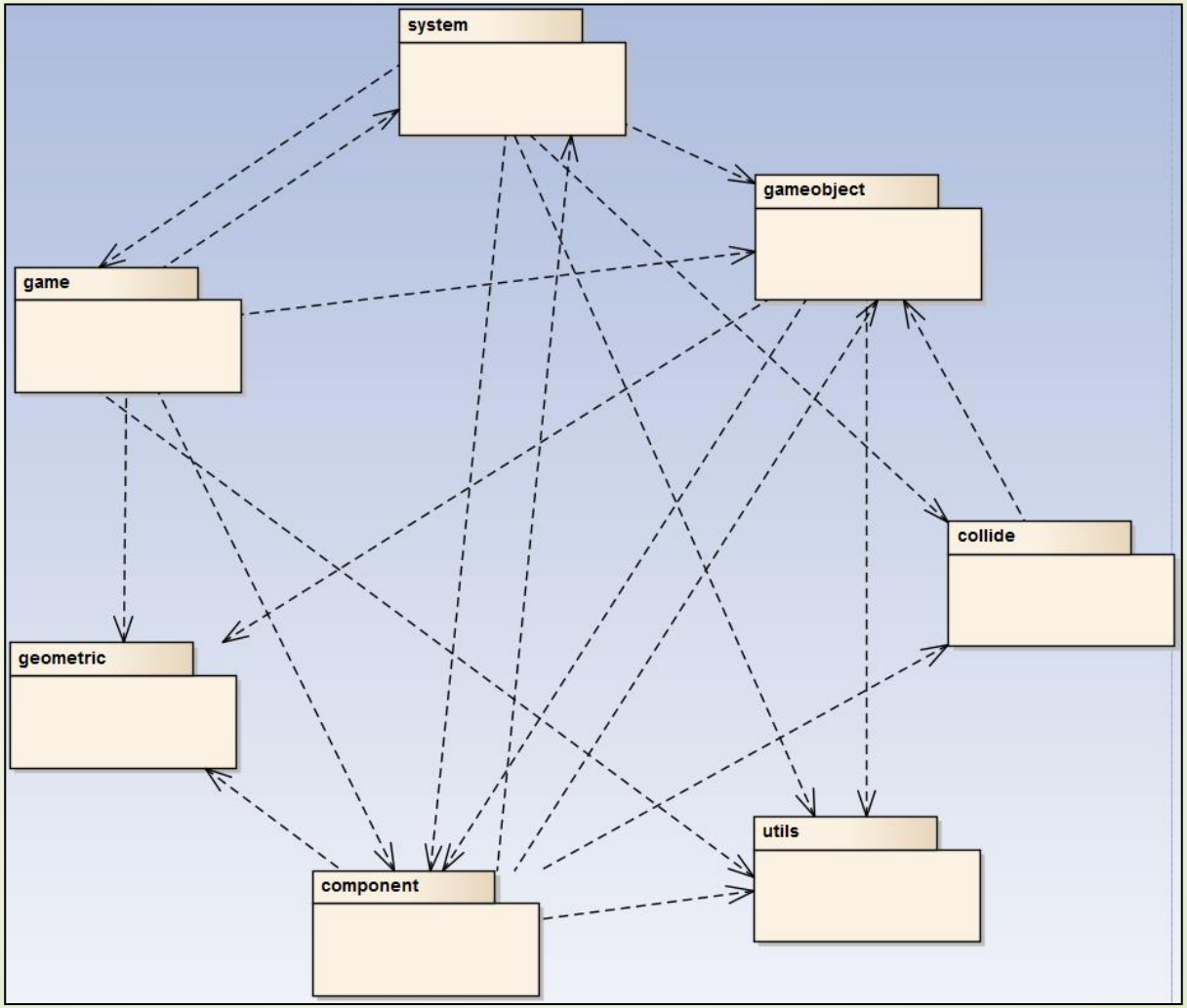


Diagrama classe: collide

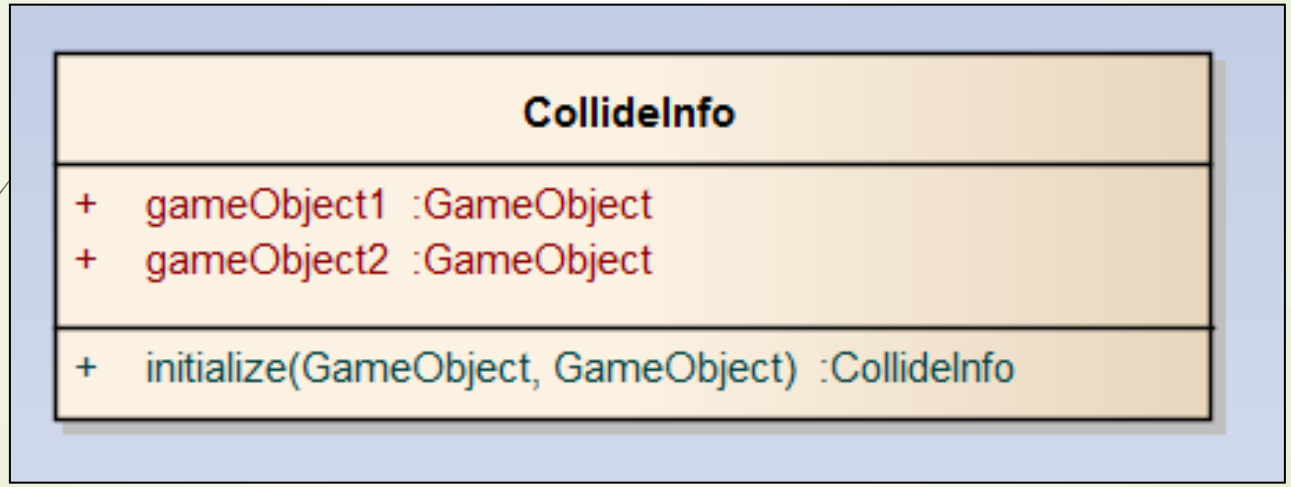


Diagrama classe: component

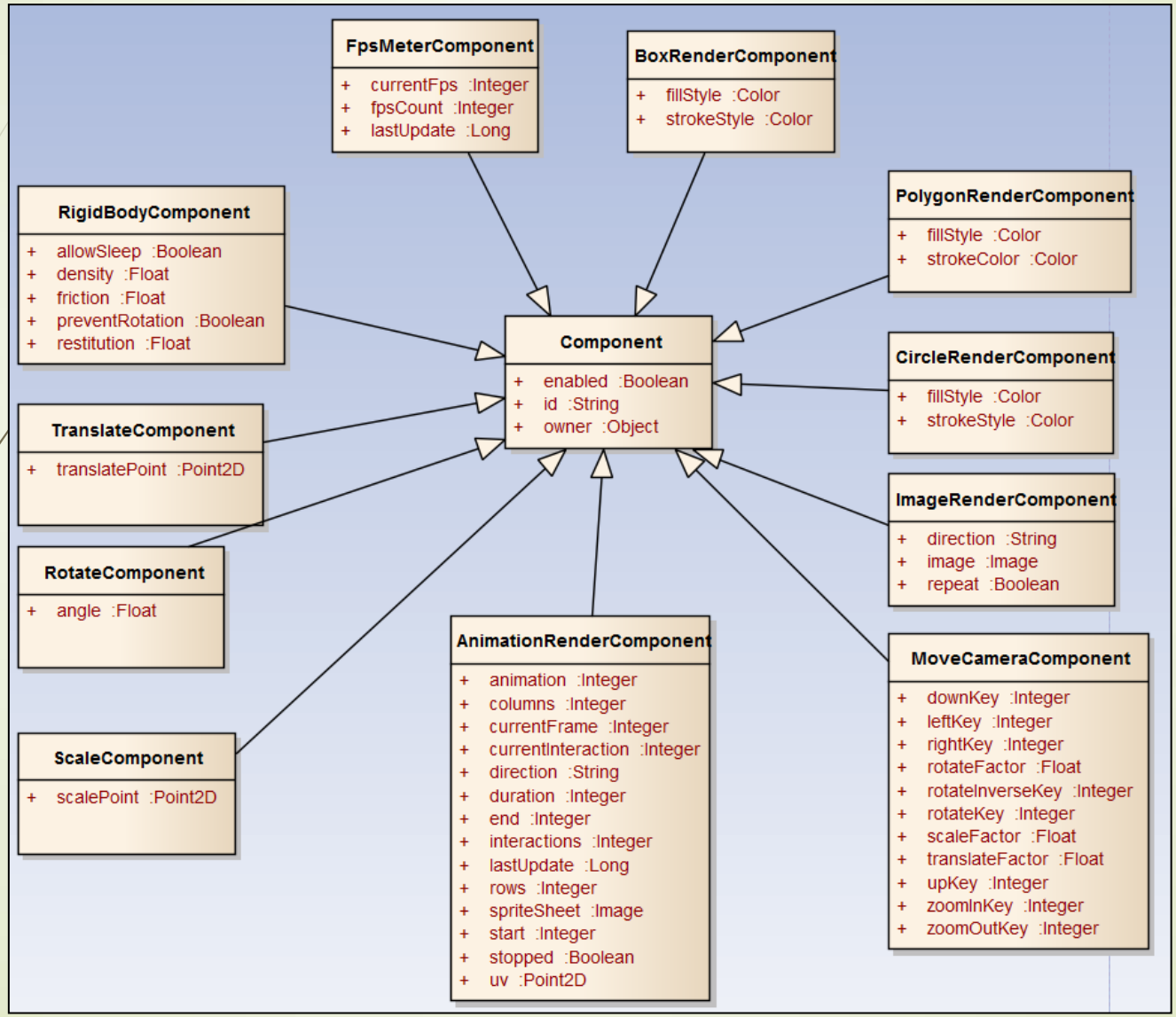


Diagrama classe: game

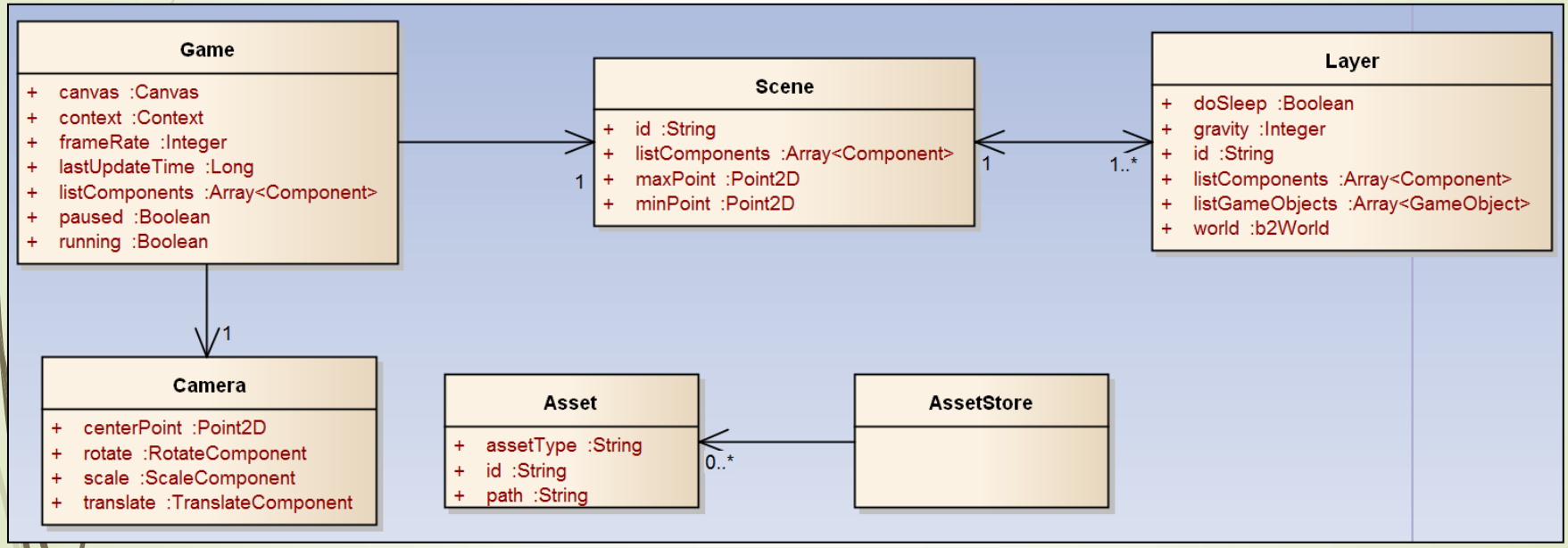


Diagrama classe: gameobject

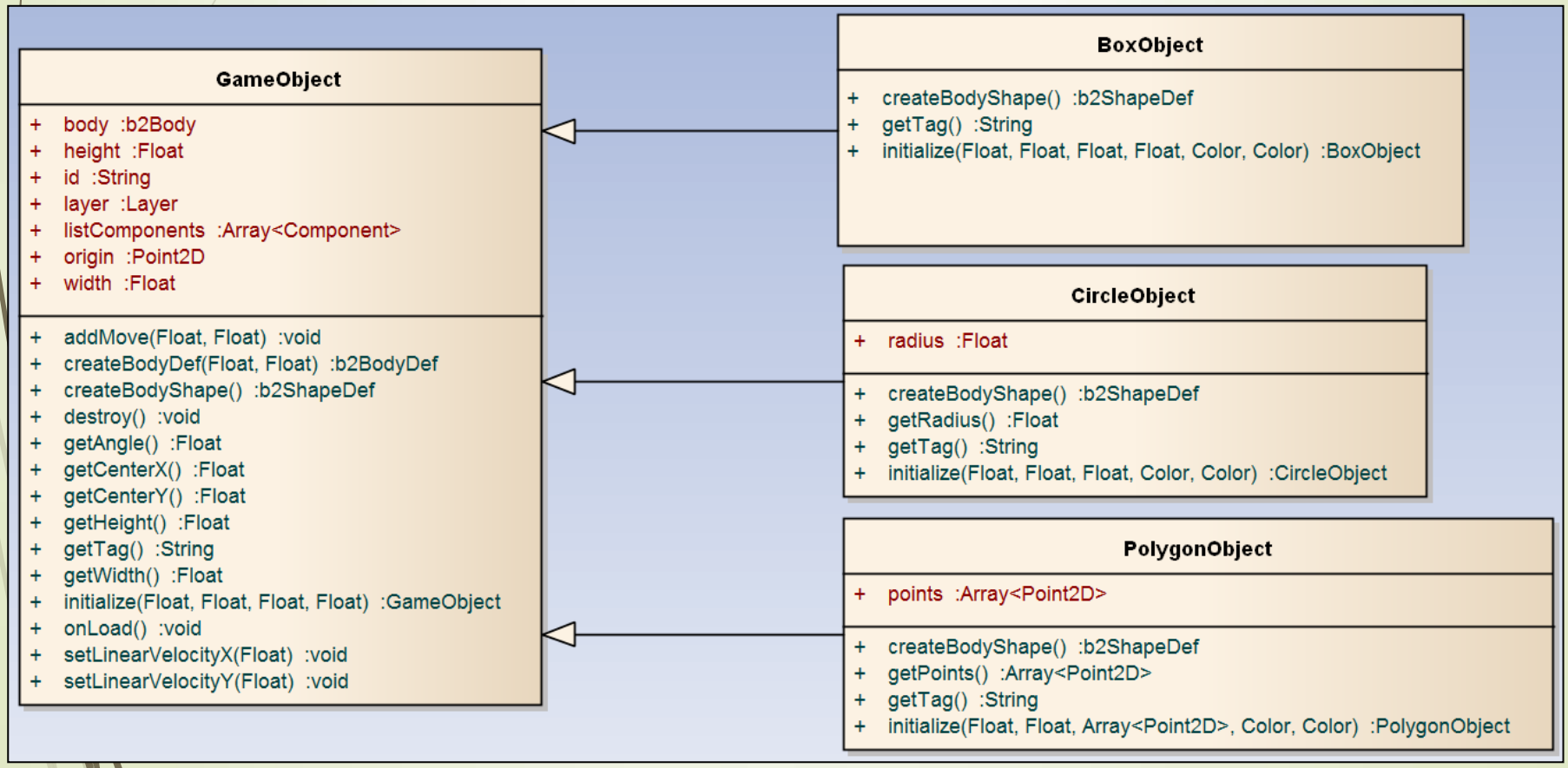


Diagrama classe: system

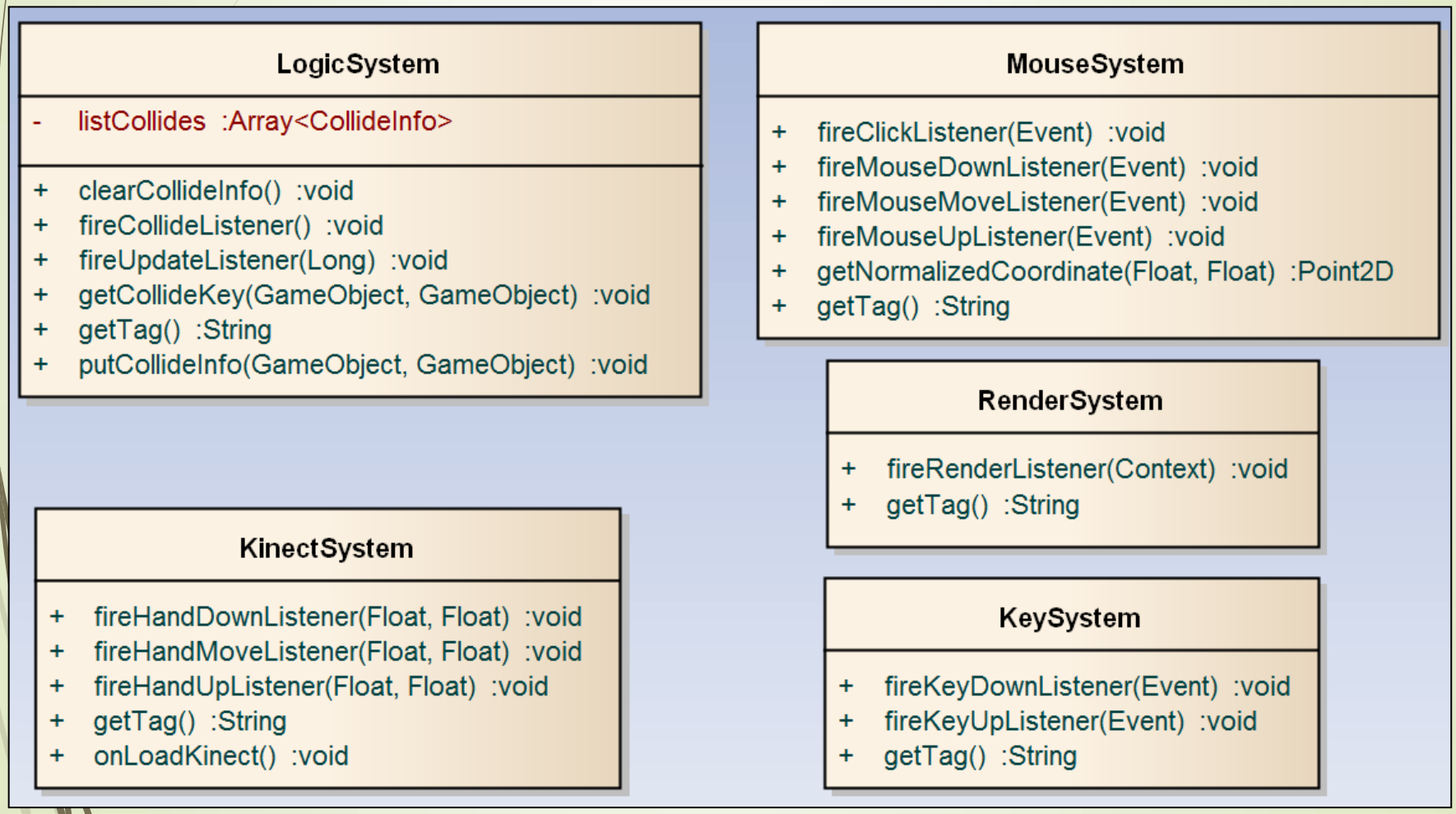


Diagrama: Pacote do editor de jogos

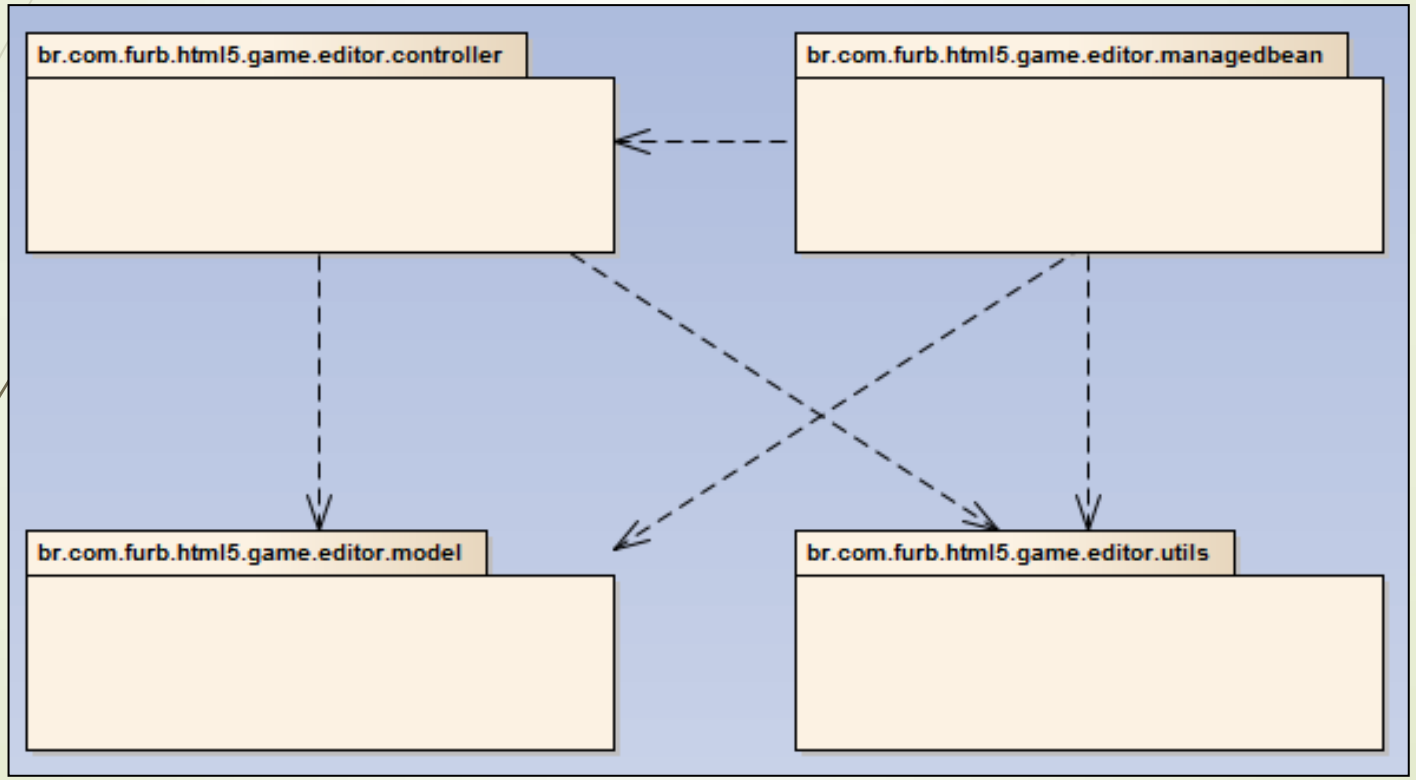


Diagrama classe: editor.controller

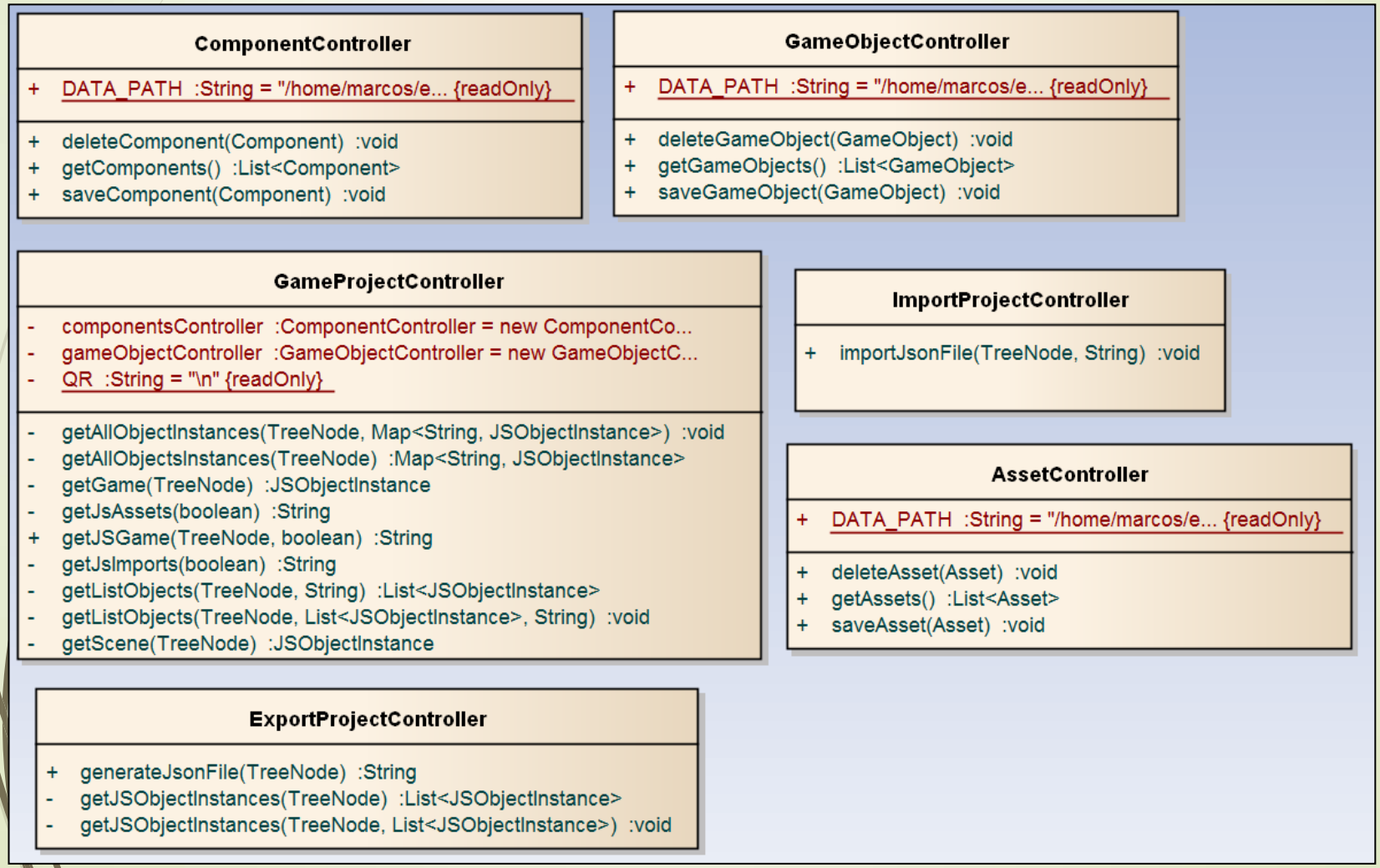


Diagrama classe: editor.managedbean

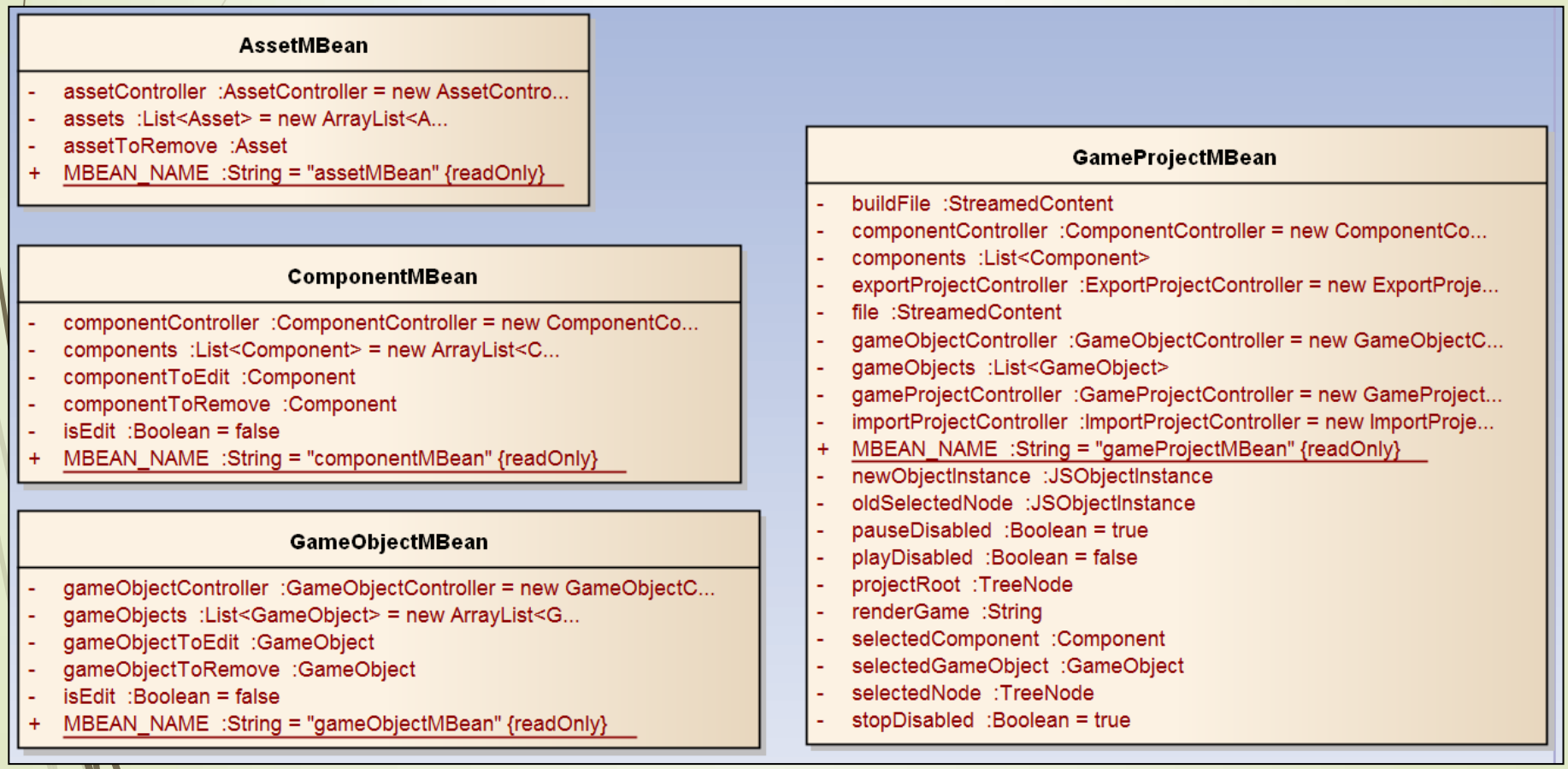


Diagrama classe: editor.model

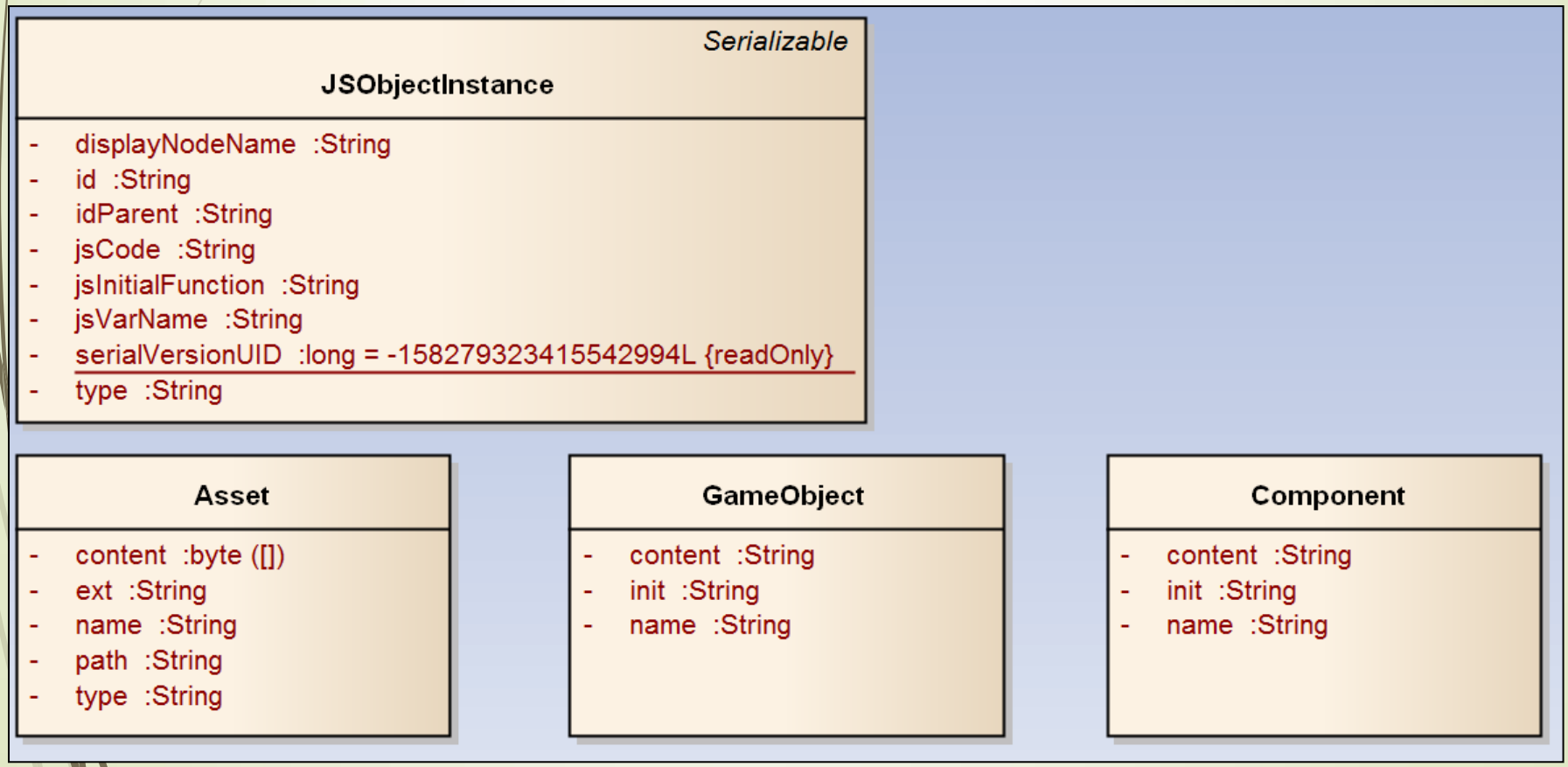
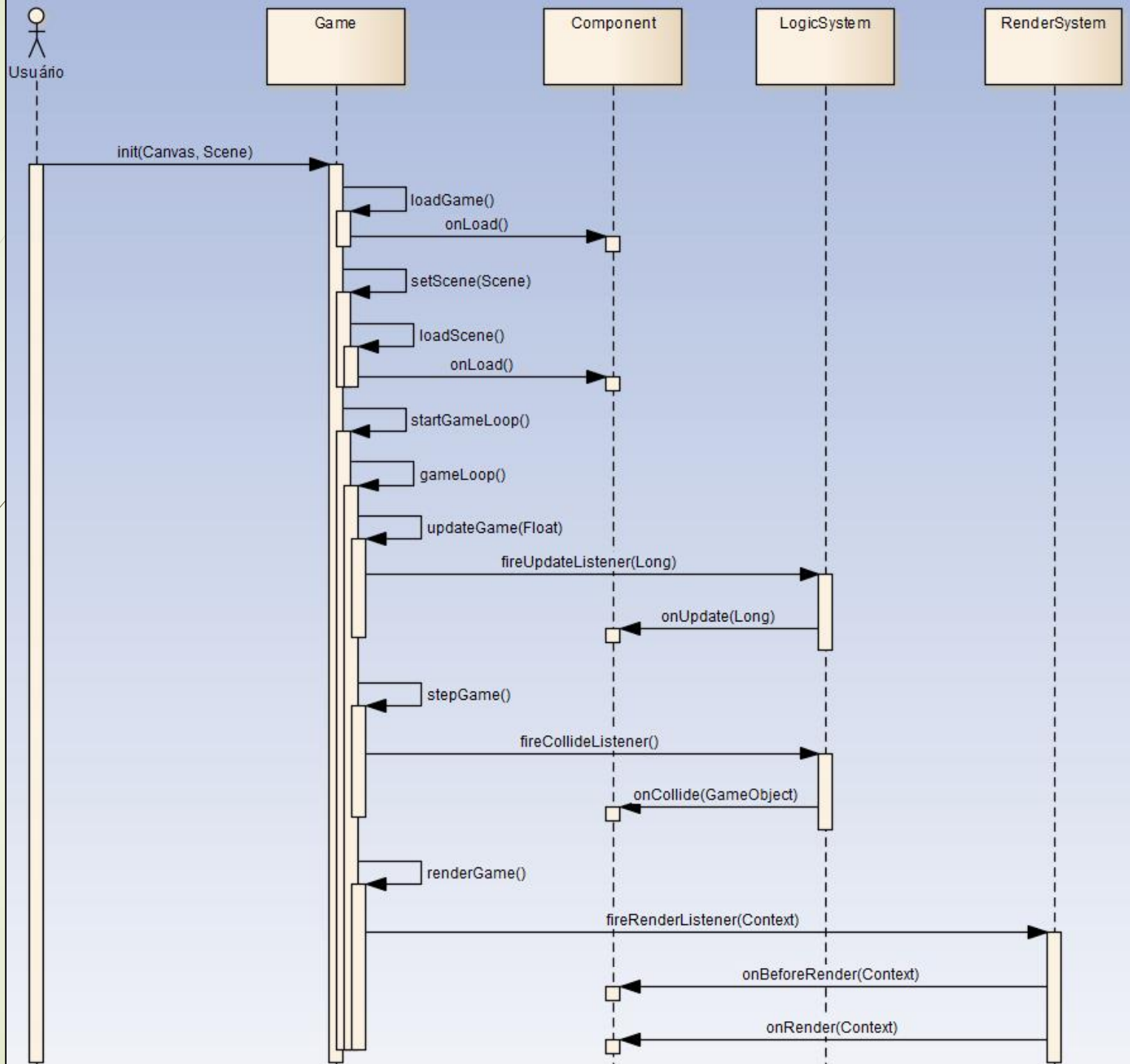


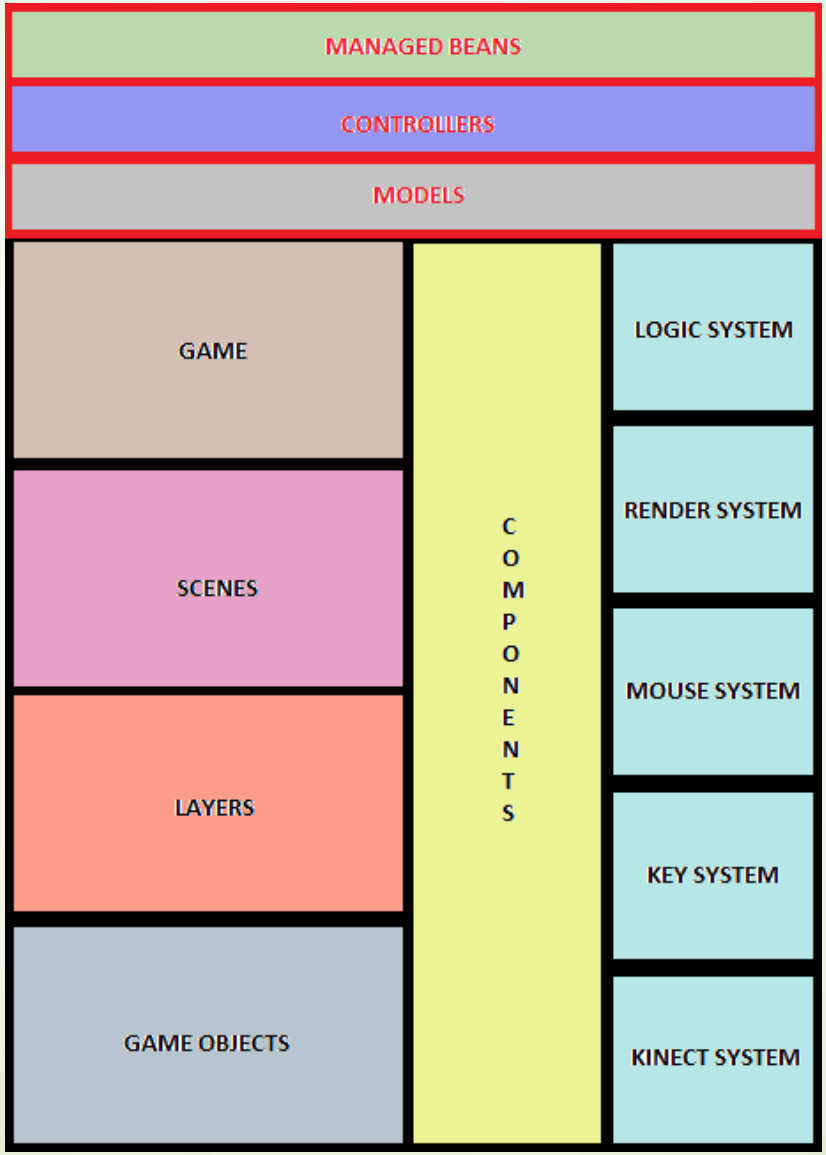
Diagrama seqüência: gameloop



Implementação

- Orientação a objetos
- Arquitetura orientada a componentes
- HTML5 e Javascript
- Box2DJS
- Zigfu
- SublimeText
- Java Server Faces e Primefaces
- JBoss
- Eclipse IDE

Implementação: Camadas arquitetura



Implementação: Eventos disponíveis

```
// Callback chamado quando algum componente enviar uma mensagem para este.
Component.prototype.onReceiveMessage = function(message, extras){}

// Callback chamado quando o usuário apertar uma tecla.
Component.prototype.onKeyDown = function(keyCode){}

//Callback chamado quando o usuário soltar uma tecla.
Component.prototype.onKeyUp = function(keyCode){}

//Callback chamado quando o usuário clicar com o mouse.
Component.prototype.onClick = function(x, y, wich){}

//Callback chamado quando o usuário pressionar o mouse.
Component.prototype.onMouseDown = function(x, y, wich){}

//Callback chamado quando o usuário soltar o mouse.
Component.prototype.onMouseUp = function(x, y, wich){}

//Callback chamado quando o usuário mover o mouse.
Component.prototype.onMouseMove = function(x, y){}

//Callback chamado antes do objeto ser renderizado.
Component.prototype.onBeforeRender = function(context){}

//Callback chamado quando o objeto for renderizado.
Component.prototype.onRender = function(context){}

//Callback chamado quando o objeto for atualizado.
Component.prototype.onUpdate = function(delta){}

//Callback chamado quando o objeto colidir com outro objeto.
Component.prototype.onCollide = function(otherGameObject){}

//Callback chamado quando o component é carregado.
Component.prototype.onLoad = function(){}

//Callback chamado quando o component é destruído.
Component.prototype.onDestroy = function(){}
```

Implementação: Corpos rígidos

```
//Implementação da classe BoxObject.
BoxObject.prototype.createBodyShape = function(){
    var shape = new b2BoxDef();
    var xb = this.getWidth();
    var yb = this.getHeight();
    var scale = ComponentUtils.getComponent(this, "SCALE_COMPONENT");
    if(scale){
        xb *= Math.abs(scale.scalePoint.x);
        yb *= Math.abs(scale.scalePoint.y);
    }
    shape.extents.Set(xb/2, yb/2);
    return shape;
}

//Implementação da classe CircleObject.
CircleObject.prototype.createBodyShape = function(){
    var shape = new b2CircleDef();
    var rb = this.radius;
    var scale = ComponentUtils.getComponent(this, "SCALE_COMPONENT");
    if(scale){
        rb = this.radius * Math.abs(scale.scalePoint.x);
    }
    shape.radius = rb;
    return shape;
}

//Implementação da classe PolygonObject.
PolygonObject.prototype.createBodyShape = function(){
    var shape = new b2PolyDef();
    shape.vertexCount = this.points.length;
    var scale = ComponentUtils.getComponent(this, "SCALE_COMPONENT");
    for(var i=0; i<shape.vertexCount; i++){
        var point = this.points[i];
        if(scale){
            shape.vertices[i].Set(point.x * Math.abs(scale.scalePoint.x),
                                  point.y * Math.abs(scale.scalePoint.y));
        }else{
            shape.vertices[i].Set(point.x, point.y);
        }
    }
    return shape;
}
```

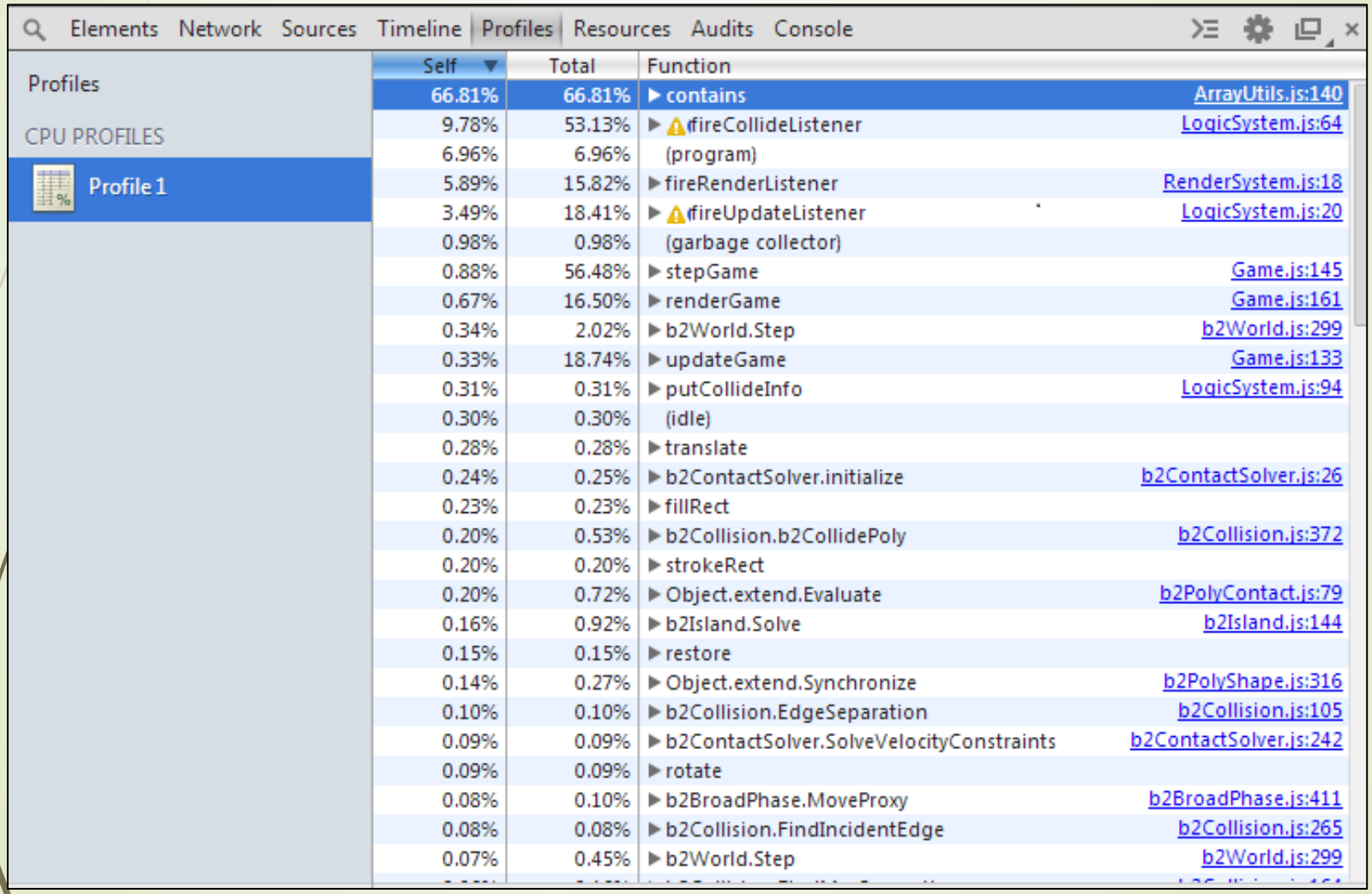
Implementação: Normalização de coordenada

```
this.getNormalizedCoordinate = function(x, y){
    //aplica o deslocamento inverso
    x -= -Game.camera.centerPoint.x+(Game.canvas.width/2);
    y -= -Game.camera.centerPoint.y+(Game.canvas.height/2);

    //aplica a escala inversa
    x -= Game.camera.centerPoint.x;
    x /= Game.camera.scale.scalePoint.x;
    x += Game.camera.centerPoint.x;

    y -= Game.camera.centerPoint.y;
    y /= Game.camera.scale.scalePoint.y;
    y += Game.camera.centerPoint.y;

    //aplica a translação inversa
    x -= Game.camera.translate.translatePoint.x;
    y -= Game.camera.translate.translatePoint.y;

    //aplica a rotação inversa
    var sine = Math.sin(-Game.camera.rotate.angle);
    var cosine = Math.cos(-Game.camera.rotate.angle);
    x -= Game.camera.centerPoint.x;
    y -= Game.camera.centerPoint.y;
    var xn = x * cosine - y * sine;
    var yn = x * sine + y * cosine;
    x = xn + Game.camera.centerPoint.x;
    y = yn + Game.camera.centerPoint.y;

    //Retorna a coordenada normalizada
    return new Point2D().initialize(x, y);
}
```

Resultados e discussões

Características	Unity 3D	Impact	Motor desenvolvido
Suporte 2D		X	X
Suporte 3D	X		
Orientada a componentes	X		X
Suporte a física	X	X	X
Editor de cena	X	X	X
Ferramentas de debug	X	X	
Geração para dispositivos móveis	X	X	
Código fonte aberto			X
Sem necessidade de licença			X

Resultados e discussões

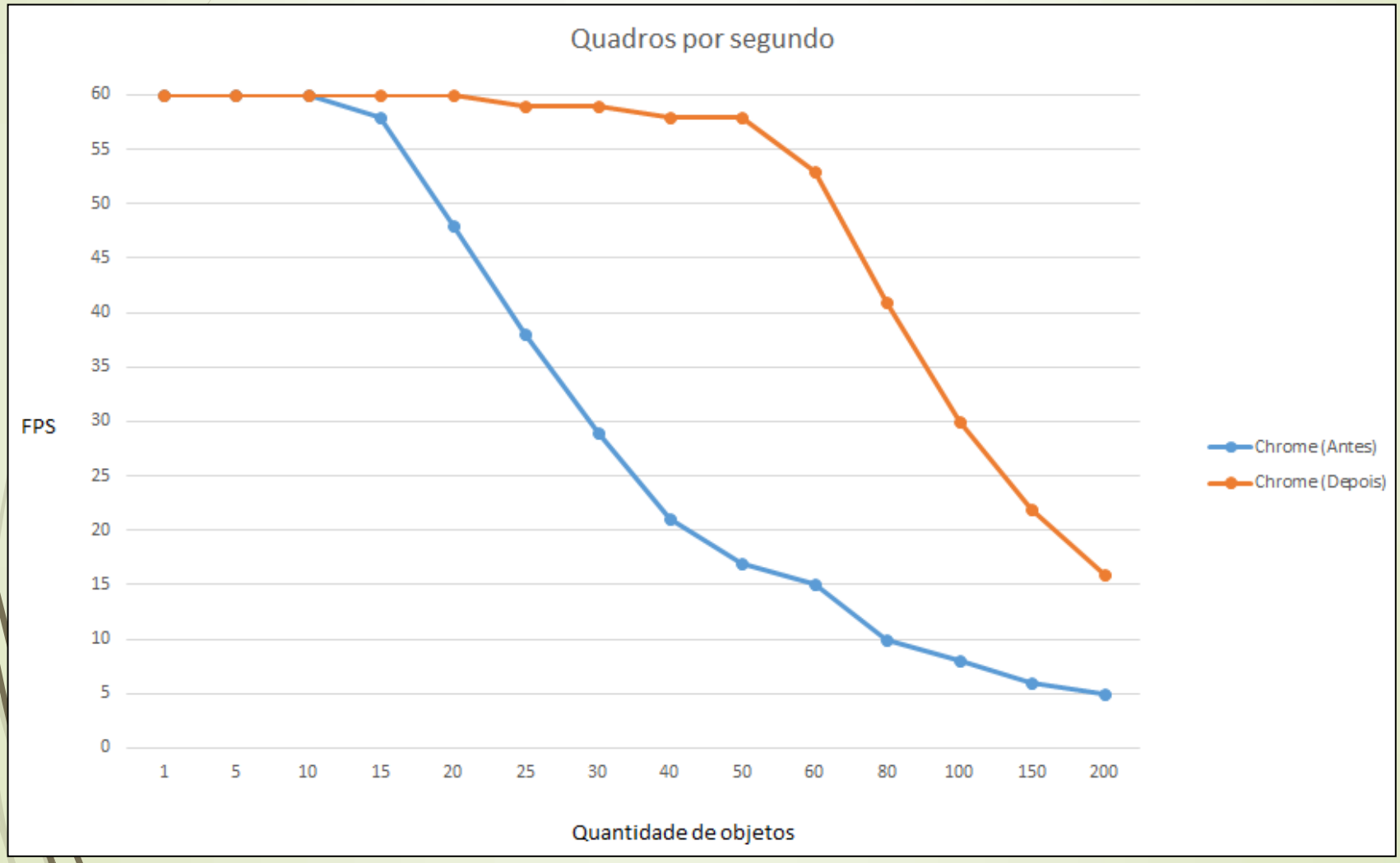


Self	Total	Function	
66.81%	66.81%	▶ contains	ArrayUtils.js:140
9.78%	53.13%	▶ ⚠ fireCollideListener	LogicSystem.js:64
6.96%	6.96%	(program)	
5.89%	15.82%	▶ fireRenderListener	RenderSystem.js:18
3.49%	18.41%	▶ ⚠ fireUpdateListener	LogicSystem.js:20
0.98%	0.98%	(garbage collector)	
0.88%	56.48%	▶ stepGame	Game.js:145
0.67%	16.50%	▶ renderGame	Game.js:161
0.34%	2.02%	▶ b2World.Step	b2World.js:299
0.33%	18.74%	▶ updateGame	Game.js:133
0.31%	0.31%	▶ putCollideInfo	LogicSystem.js:94
0.30%	0.30%	(idle)	
0.28%	0.28%	▶ translate	
0.24%	0.25%	▶ b2ContactSolver.initialize	b2ContactSolver.js:26
0.23%	0.23%	▶ fillRect	
0.20%	0.53%	▶ b2Collision.b2CollidePoly	b2Collision.js:372
0.20%	0.20%	▶ strokeRect	
0.20%	0.72%	▶ Object.extend.Evaluate	b2PolyContact.js:79
0.16%	0.92%	▶ b2Island.Solve	b2Island.js:144
0.15%	0.15%	▶ restore	
0.14%	0.27%	▶ Object.extend.Synchronize	b2PolyShape.js:316
0.10%	0.10%	▶ b2Collision.EdgeSeparation	b2Collision.js:105
0.09%	0.09%	▶ b2ContactSolver.SolveVelocityConstraints	b2ContactSolver.js:242
0.09%	0.09%	▶ rotate	
0.08%	0.10%	▶ b2BroadPhase.MoveProxy	b2BroadPhase.js:411
0.08%	0.08%	▶ b2Collision.FindIncidentEdge	b2Collision.js:265
0.07%	0.45%	▶ b2World.Step	b2World.js:299

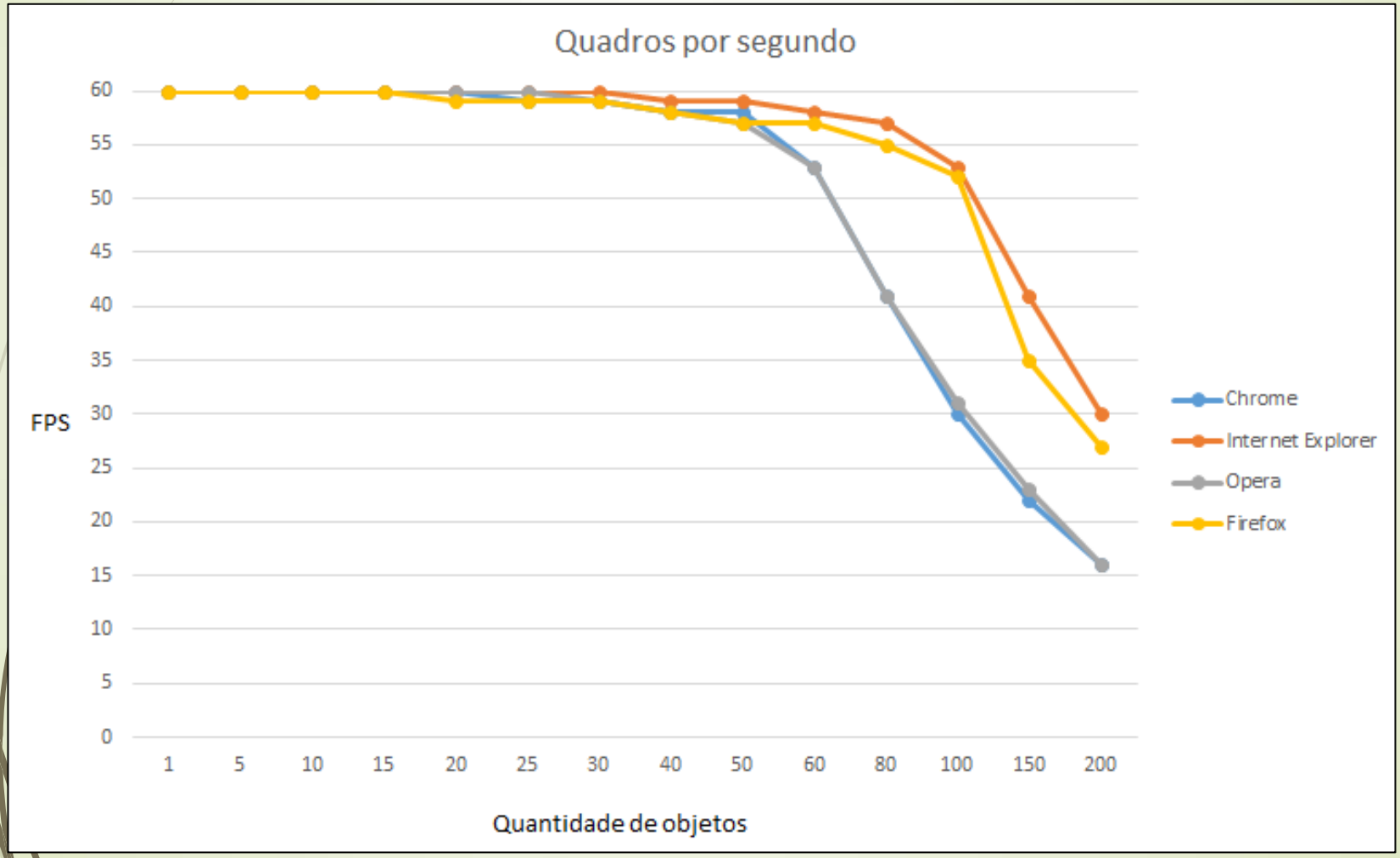
Resultados e discussões

Quantidade de objetos	Google Chrome		Internet Explorer		Opera		Mozilla Firefox	
	Antes	Depois	Antes	Depois	Antes	Depois	Antes	Depois
1	60	60	60	60	60	60	60	60
5	60	60	60	60	60	60	55	60
10	60	60	60	60	60	60	40	60
15	58	60	60	60	60	60	30	60
20	48	60	59	60	53	60	22	59
25	38	59	59	60	41	60	18	59
30	29	59	55	60	31	59	14	59
40	21	58	41	59	23	58	10	58
50	17	58	33	59	20	57	8	57
60	15	53	27	58	16	53	6	57
80	10	41	20	57	12	41	5	55
100	8	30	15	53	9	31	4	52
150	6	22	11	41	7	23	3	35
200	5	16	8	30	5	16	2	27

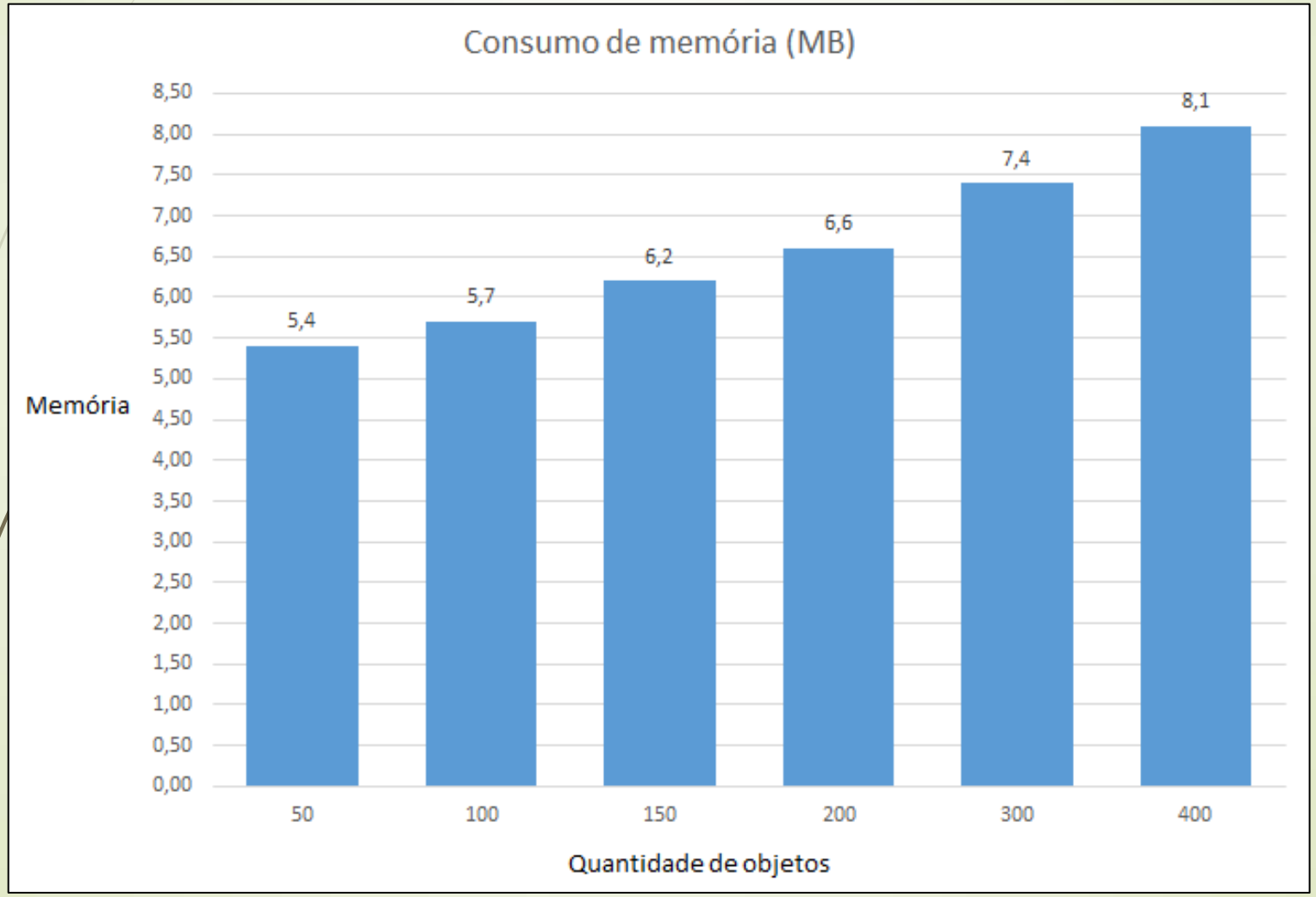
Resultados e discussões



Resultados e discussões



Resultados e discussões



Resultados e discussões

- Projeto VisEDU
- Projeto LIFE

Conclusão

- O motor de jogos atende os requisitos propostos
- Alta flexibilidade por causa da arquitetura orientada a componentes
- Desempenho superou o proposto
- O editor de jogos cumpriu os requisitos propostos
- Usuário ainda necessita conhecer programação e Javascript
- A usabilidade do editor deve ser melhorada para usuários sem conhecimento de programação e Javascript

Extensões

- Explorar mais recursos da Box2DJS
- Explorar mais recursos da Zigfu
- Biblioteca de Inteligência Artificial
- Desenhar objetos 3D com WebGL
- Outros sensores de movimento (*Leap Motion*)
- Implementar Grafo de Cena
- Novos componentes
- Multiplayer com WebSockets
- Controle de acesso no editor de jogos

Extensões

- Projetos colaborativos
- Executar em dispositivos móveis
- Melhorar a usabilidade do editor de jogos

Demonstração

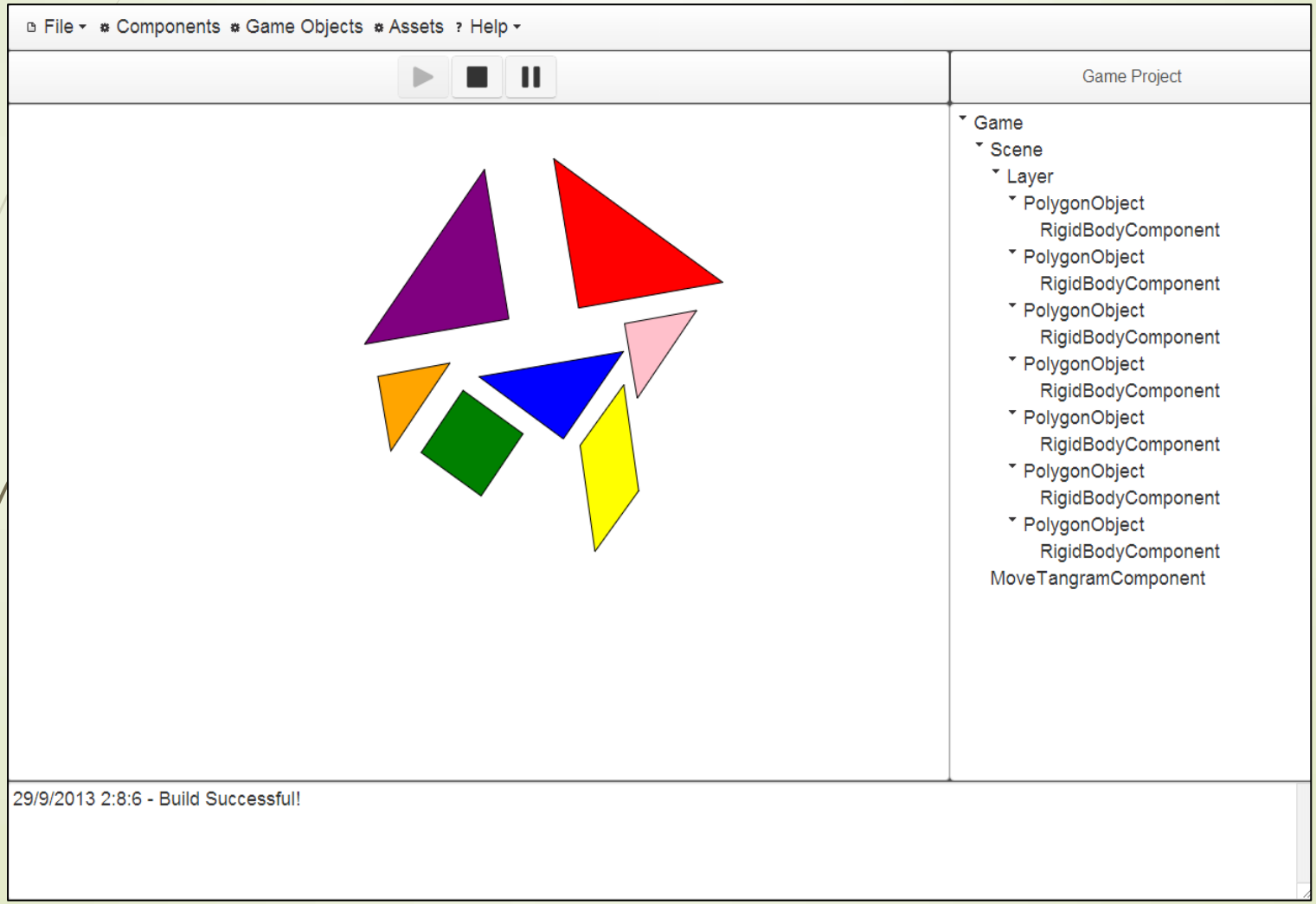


Diagrama classe: geometric

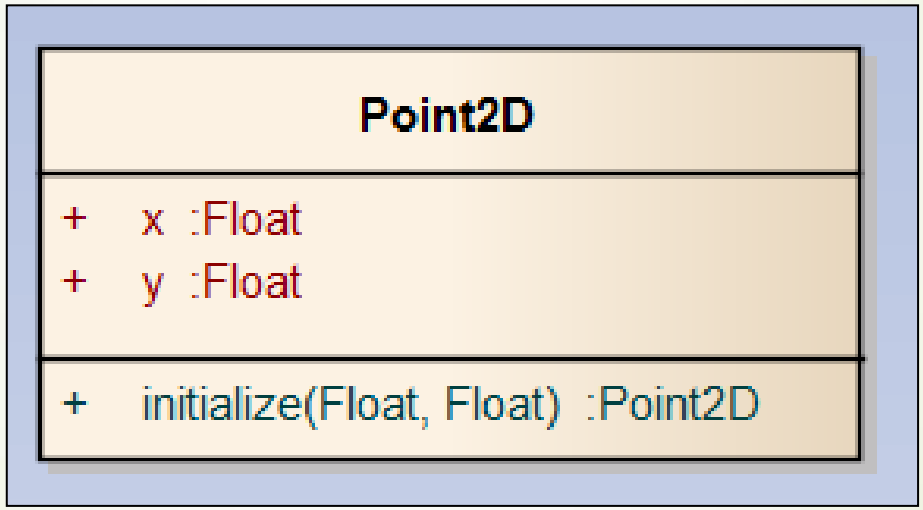


Diagrama classe: utils

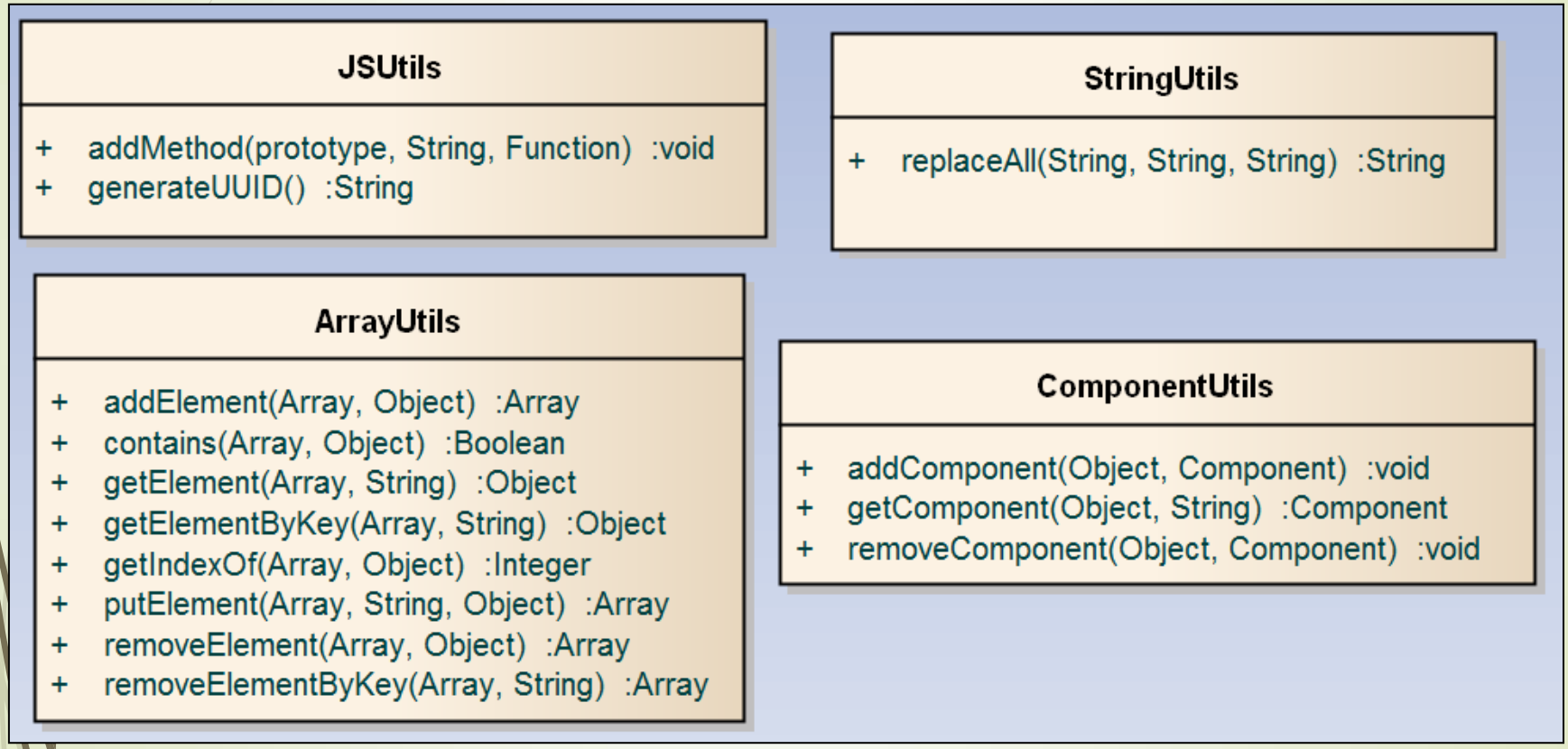


Diagrama classe: editor.utils