Desenvolvimento de um motor de Jogos 3D, utilizando WebGL

Daniel Pereira
prof. Dalton S. dos Reis

FURB - Universidade Regional de Blumenau
DSC - Departamento de Sistemas e Computação
Grupo de Pesquisa em Computação Gráfica e Entretenimento Digital
2012/I1 - http://www.inf.furb.br/gcg/

Roteiro

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

Introdução

- Internet
  - acessada por bilhões de pessoas
  - navegador web é sua principal forma de acesso
  - rápida evolução dos navegadores web
- Entretenimento através dos jogos
- WebGL
  - OpenGL para navegadores web
  - não requer plugin
  - é suportado pela maioria dos navegadores

Objetivos

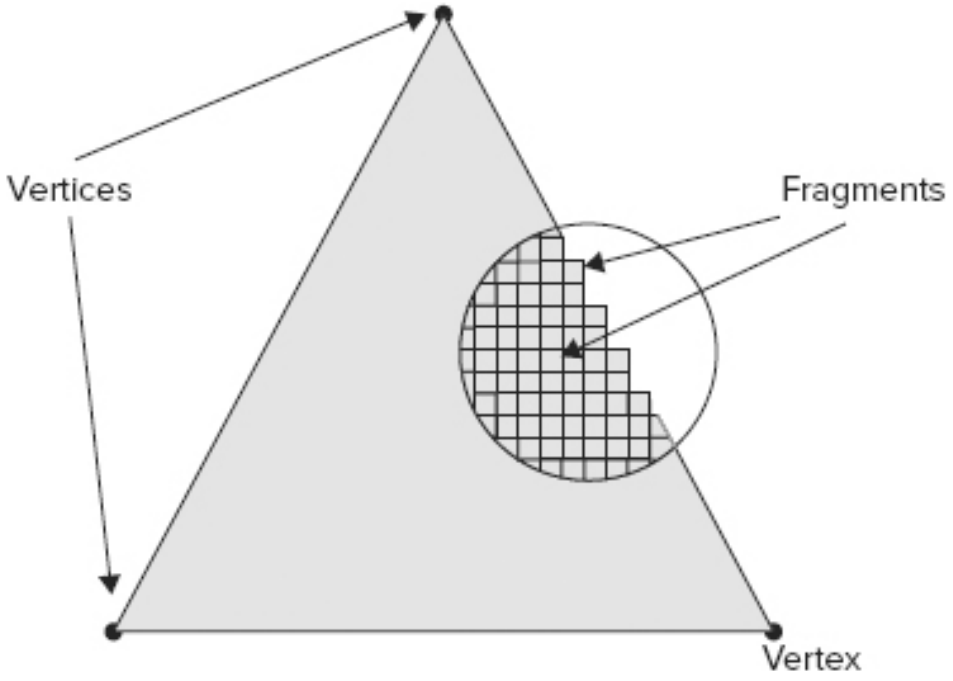
- criar um motor de jogos com suporte a:
  - gerenciador de objetos
  - grafo de cena
  - leitura dos objetos através de arquivos
  - movimentação da câmera
  - Iluminação
  - textura
- criar uma aplicação exemplo

WebGL

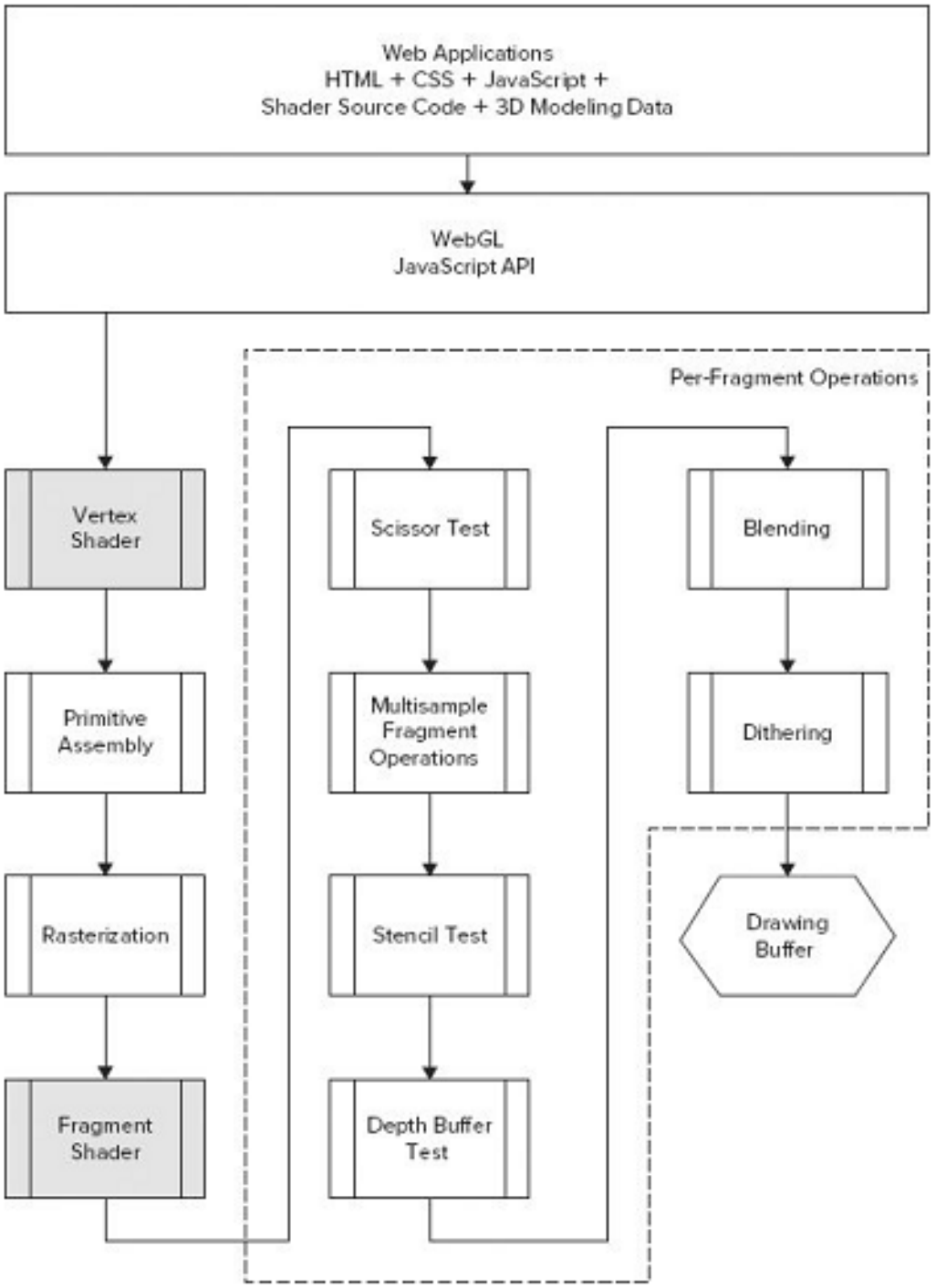
- Baseado no OpenGL ES 2.0
- API acessada via JavaScript
- *Shaders* descritos através da GLSL

WebGL

- Possui dois tipos de *shaders*:
  - *vertex shader*
  - *fragment shader*

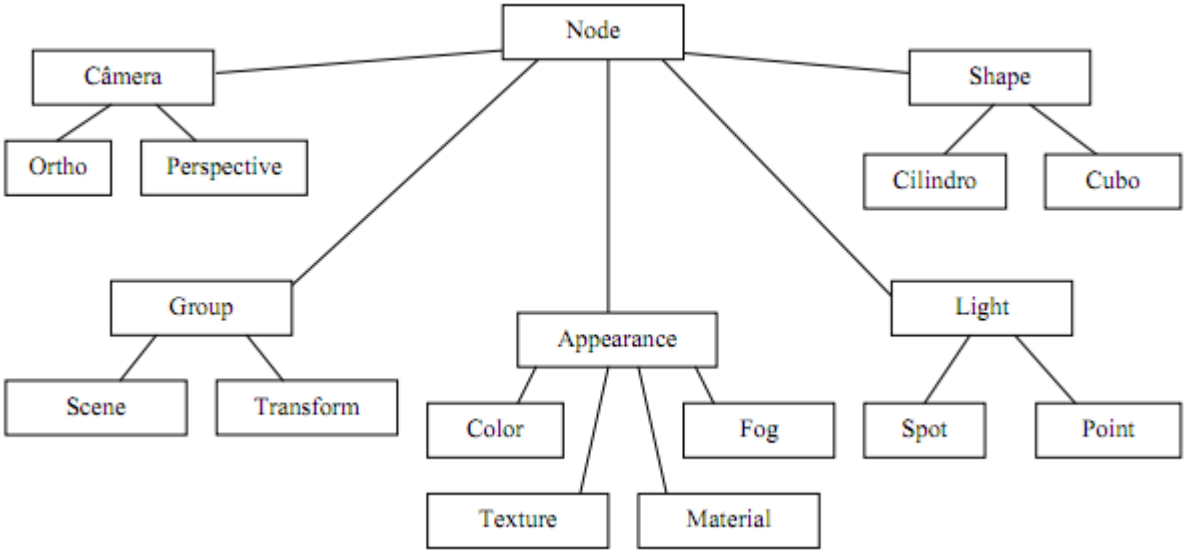


WebGL *pipeline*



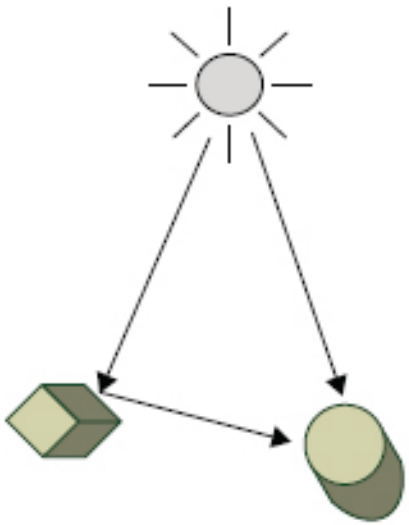
Grafo de cena

- Estrutura de dados
- Organiza os objetos da cena em hierarquia
- Simplifica a manipulação de cenas complexas

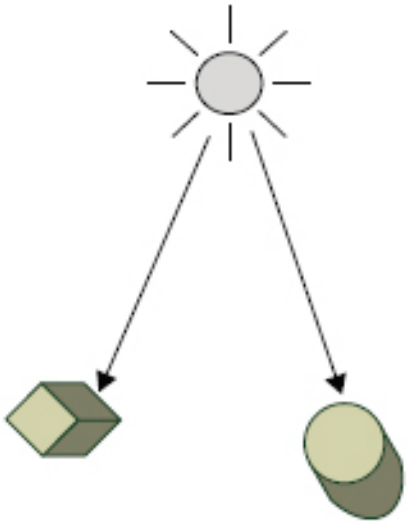


Iluminação

- Modos de iluminação
  - Global
  - Local



Modelo de Luz
Global



Modelo de Luz
Local

Modelo de reflexão de Phong

- A cor é baseada em três reflexões:
  - ambiente
    - reflete por toda a cena de maneira equivalente
  - difusa
    - sua reflexão depende da direção da luz de entrada
  - especular
    - é calculada considerando a direção do observador

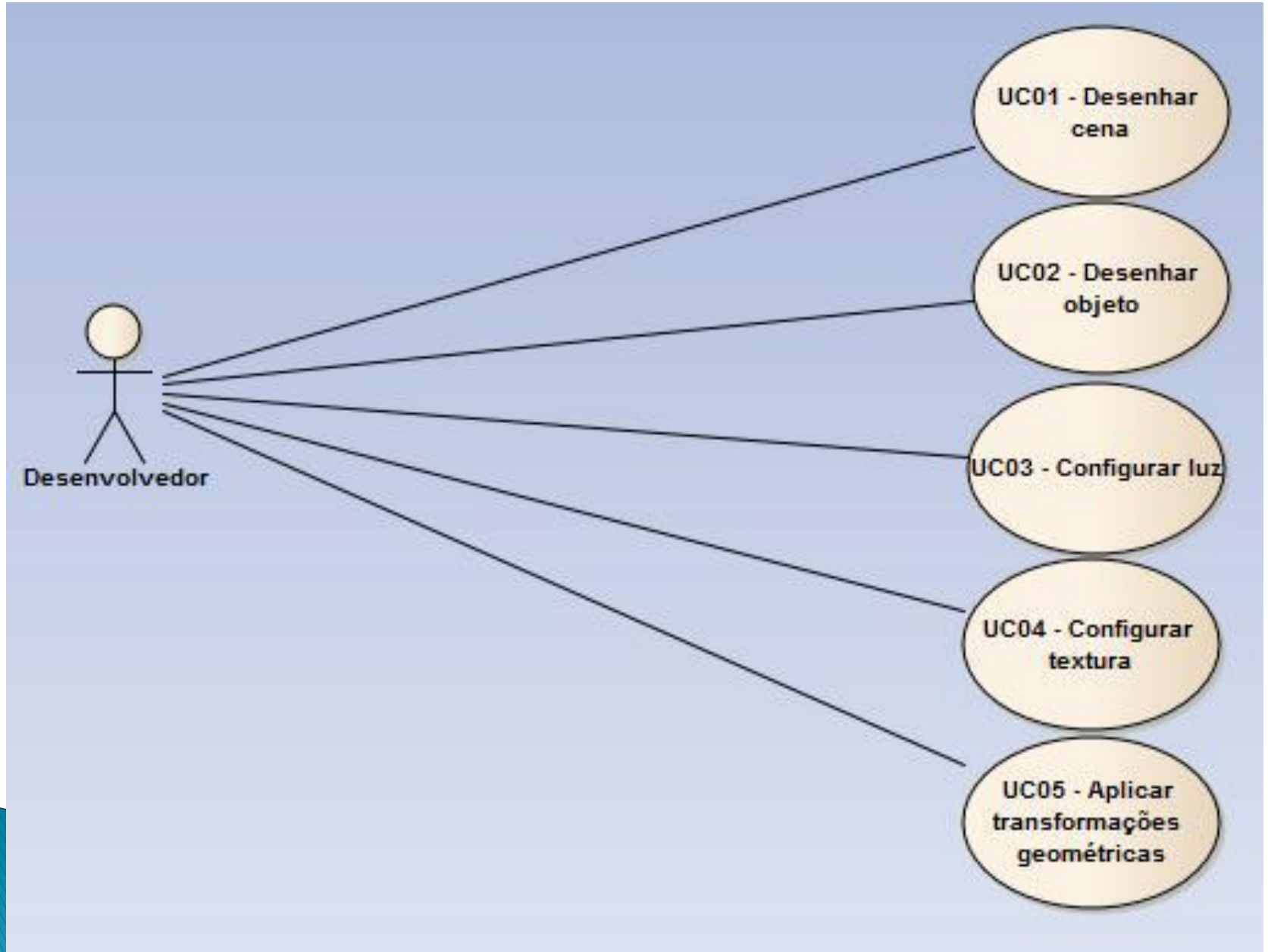
Requisitos Funcionais

- fornecer um *framework* para desenvolvimento em WebGL
- fornecer um grafo de cena para gerenciar e manipular os objetos hierarquicamente
- fornecer recursos para a movimentação da câmera
- fornecer recursos para que sejam feitas transformações geométricas
- fornecer recursos para o gerenciamento de luz no ambiente
- fornecer recursos para importar objetos que sigam o padrão estabelecido

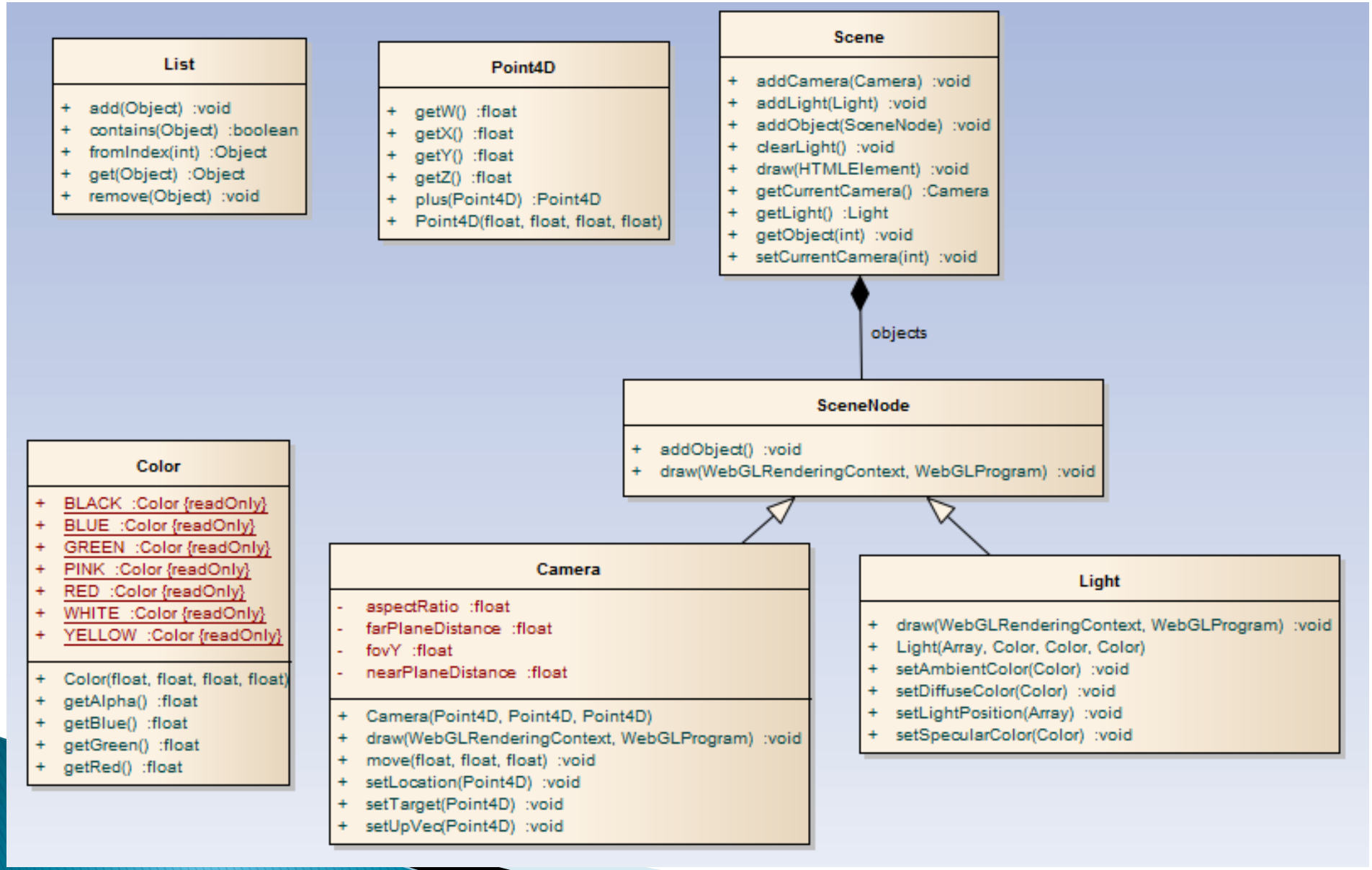
Requisitos não funcionais

- utilizar a linguagem JavaScript para implementação
- utilizar a biblioteca WebGL para desenvolvimento da parte gráfica
- funcionar de maneira equivalente nos navegadores que suportam WebGL

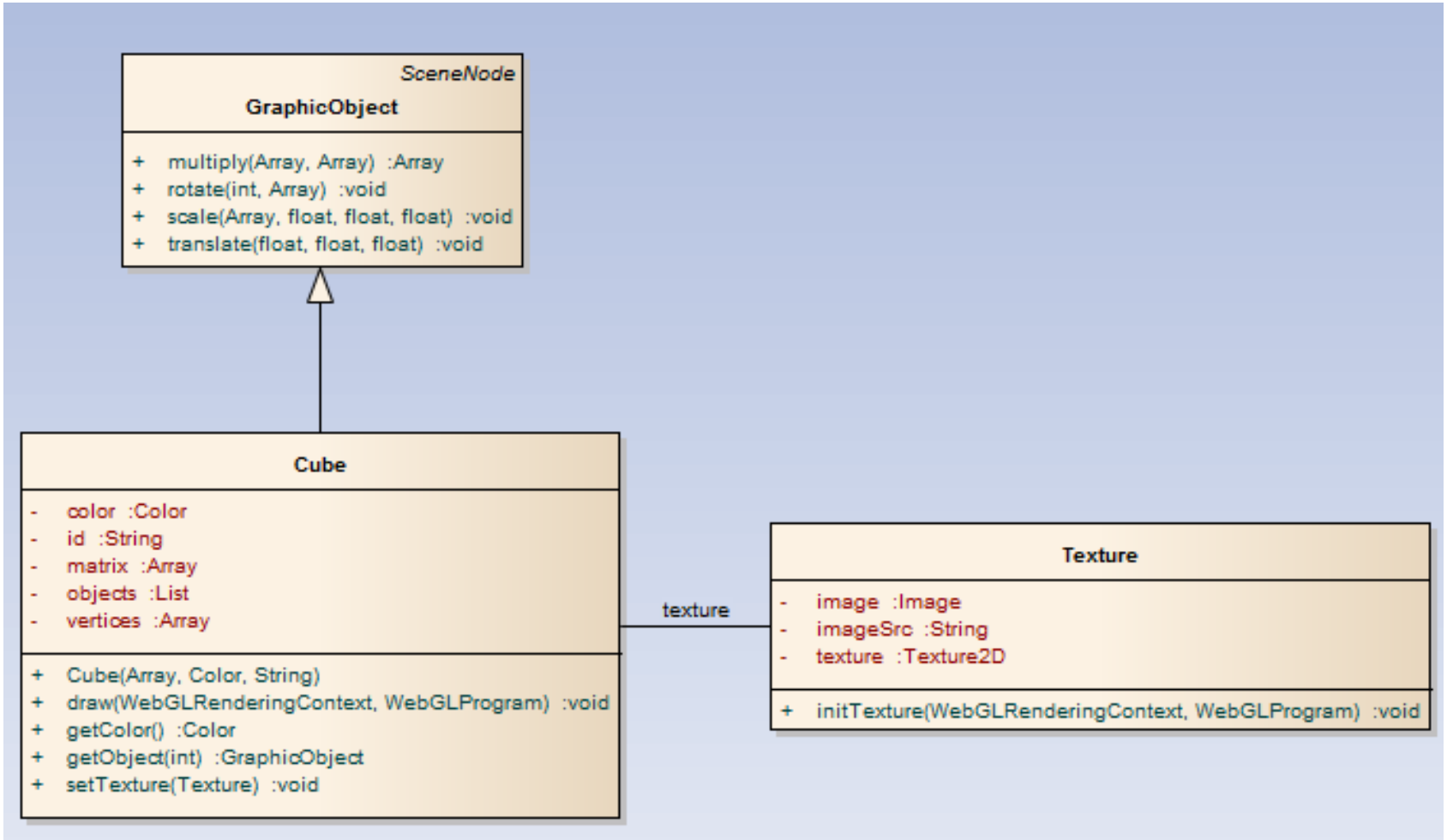
Casos de Uso


UC01 - Desenhar cena
UC02 - Desenhar objeto
UC03 - Configurar luz
UC04 - Configurar textura
UC05 - Aplicar transformações geométricas
Desenvolvedor

Diagramas de classe



Diagramas de classe



Diagramas de classe

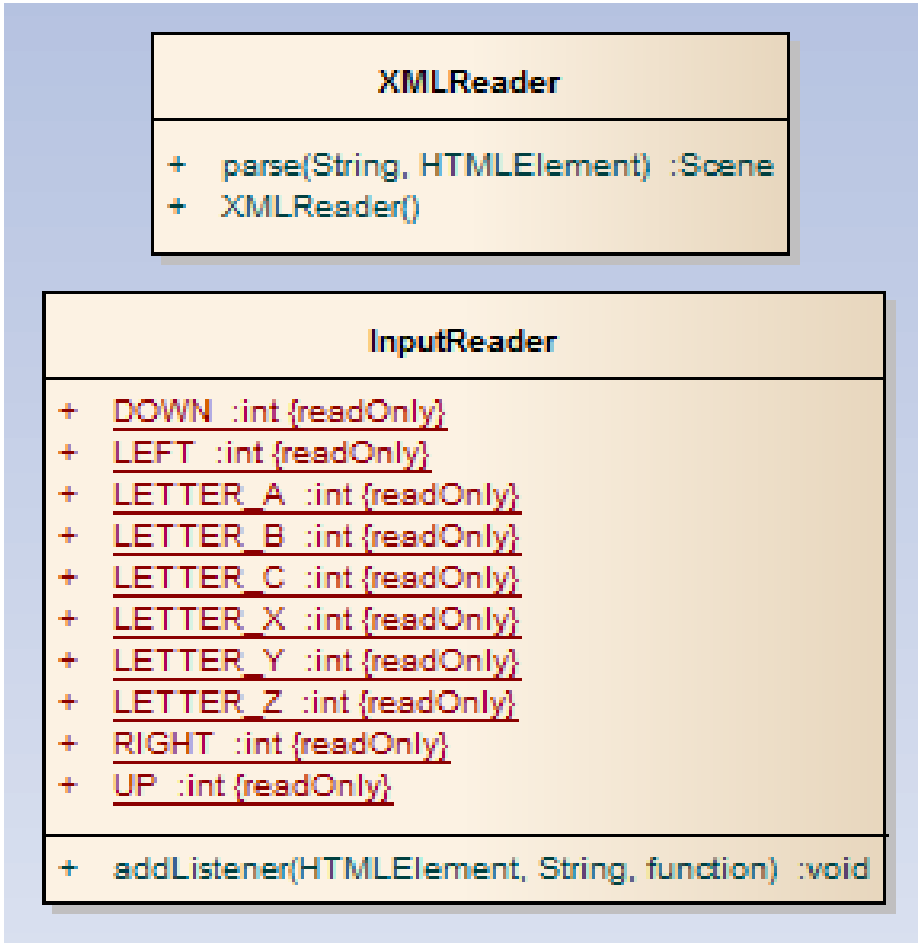
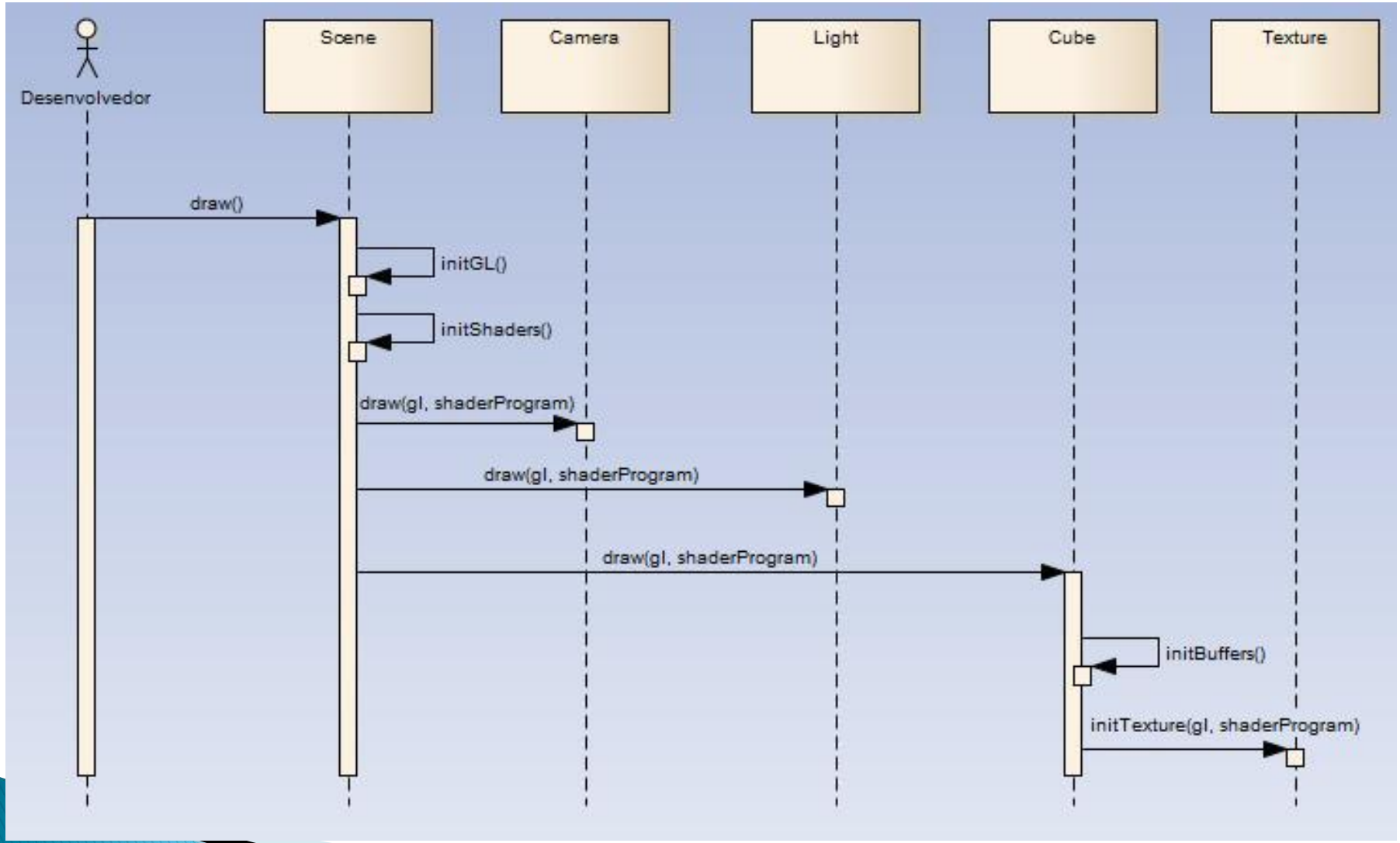


Diagrama de sequência



Desenvolvimento

- Ferramentas utilizadas
  - Eclipse Juno
  - Apache Tomcat
  - Google Chrome
  - Mozilla Firefox
  - Opera

Desenvolvimento

- Baseado na arquitetura do V-ART
- Abstrai totalmente a API do WebGL
- Contém os arquivos *shaders* pré-configurados
- Permite a importação de arquivos no padrão XML

Operacionalidade

```
//cria a cena
var scene = new Scene();

//cria a luz
var lightPosition = [0.0, 0.0, 1.0];
var ambientColor = new Color(0.5, 0.5, 0.5);
var diffuseColor = new Color(0.7, 0.7, 0.7);
var specularColor = new Color(0.8, 0.8, 0.8);
var light = new Light(lightPosition, ambientColor, diffuseColor,
                      specularColor);

//cria a câmera
var location = new Point4D(3.0, 3.0, 5.0);
var target = new Point4D(0.0, 0.0, 0.0);
var upVec = new Point4D(0.0, 1.0, 0.0);
var camera = new Camera(location, target, upVec);

//adiciona a luz e a câmera à cena
scene.addCamera(camera);
scene.addLight(light);
```

Operacionalidade

```
//cria os vertices do cubo
var vertices = [new Point4D(1.0,  1.0,  1.0), new Point4D(-1.0,  1.0,
1.0),
new Point4D(-1.0, -1.0,  1.0), new Point4D(1.0, -1.0,  1.0),
new Point4D(1.0,  1.0, -1.0), new Point4D(-1.0,  1.0, -1.0),
new Point4D(-1.0, -1.0, -1.0), new Point4D(1.0, -1.0, -1.0),
new Point4D(-1.0,  1.0,  1.0), new Point4D(-1.0,  1.0, -1.0),
new Point4D(-1.0, -1.0, -1.0), new Point4D(-1.0, -1.0,  1.0),
new Point4D(1.0,  1.0,  1.0), new Point4D(1.0, -1.0,  1.0),
new Point4D(1.0, -1.0, -1.0), new Point4D(1.0,  1.0, -1.0),
new Point4D(1.0,  1.0,  1.0), new Point4D(1.0,  1.0, -1.0),
new Point4D(-1.0,  1.0, -1.0), new Point4D(-1.0,  1.0,  1.0),
new Point4D(1.0, -1.0,  1.0), new Point4D(1.0, -1.0, -1.0),
new Point4D(-1.0, -1.0, -1.0), new Point4D(-1.0, -1.0,  1.0)];
//cria o cubo e texture adiciona a texture ao cubo o cubo à cena
var cube = new Cube(vertices);
var texture = new Texture("crate.gif");
cube.setTexture(texture);
scene.addObject(cube);
```

```
var canvas = document.getElementById("myCanvas");
scene.draw(canvas);
```

Operacionalidade

Resultados e discussão

Renderização dos objetos

- 1ª estratégia: sempre criar novos *buffers*
  - os *buffers* são descartados após a renderização
  - baixo desempenho
- 2ª estratégia: *cache* de *buffers*
  - os *buffers* são criados apenas quando necessário
  - melhora considerável no desempenho

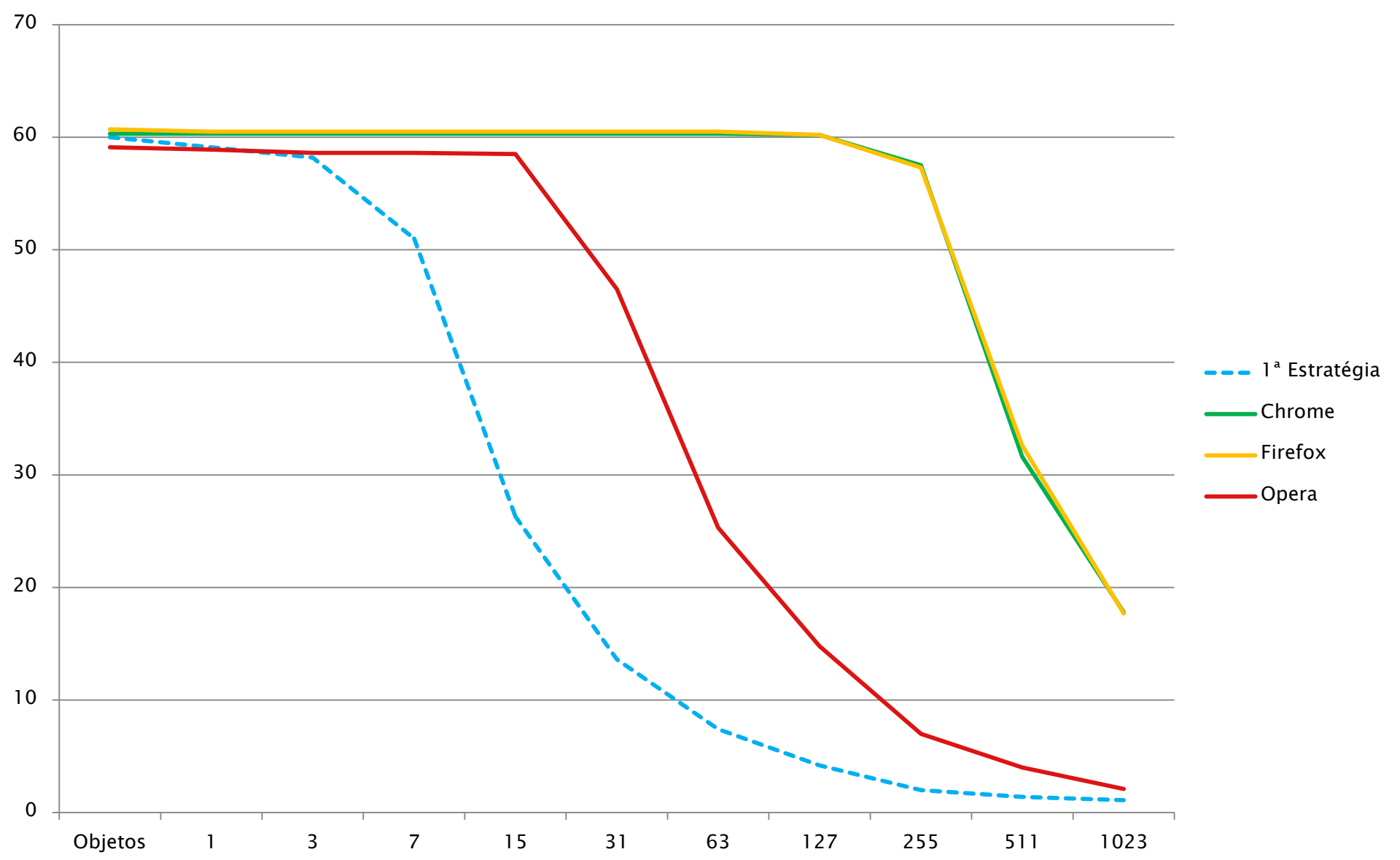
Resultados e discussão

1ª estratégia

Objetos	FPS (Chrome)
1	60,0
3	59,1
7	58,2
15	51,0
31	26,3
63	13,6
127	7,4
255	4,2
511	2,0
1023	1,4
2047	1,1

2ª estratégia

Objetos	*FPS* (Chrome)	FPS (Firefox)	FPS (Opera)
1	60,3	60,7	59,1
3	60,3	60,5	58,9
7	60,3	60,5	58,6
15	60,3	60,5	58,6
31	60,3	60,5	58,5
63	60,3	60,5	46,5
127	60,3	60,5	25,3
255	60,2	60,2	14,8
511	57,5	57,3	7,0
1023	31,6	32,6	4,0
2047	17,8	17,7	2,1


70
60
50
40
30
20
10
0
Objetos
1
3
7
15
31
63
127
255
511
1023
1ª Estratégia
Chrome
Firefox
Opera

Resultados e discussão

- O requisito de importação foi ajustado
- O requisito de textura foi adicionado
  - Textura possui problema na primeira renderização
- Não há suporte há múltiplas luzes

Conclusões

- WebGL ainda está em fase de amadurecimento
- Arquitetura adotada é simples e intuitiva
- O motor de jogos possui boa performance
- Há diferenças na implementação WebGL dos navegadores web

Extensões

- Importar arquivos OBJ
- Importar arquivos do V-ART que definem a movimentação
- criar diferentes tipos de iluminação
- transcrever código do V-ART referente aos personagens articulados

Extensões

- permitir mais de uma textura em um mesmo objeto
- permitir múltiplas iluminações em uma cena
- permitir a configuração dos *shaders*
- Abstrair o motor, deixando-o independente de WebGL

Demonstração

Obrigado