Universidade de Aveiro

departamento de electrónica, telecomunicações e informática

47053- Computação Visual

Puzzle

Implementação em WebGL de um Puzzle

8240 - mestrado integrado em engenharia de computadores e telemática

António Rafael da Costa Ferreira
NMec: 67405

Rodrigo Lopes da Cunha
NMec: 67800

Docente: Joaquim João Estrela Ribeiro Silvestre Madeira

Dezembro de 2015
2015-2016

Conteúdos

1 Introdução

O trabalho proposto para o projeto da unidade curricular de Segurança é um IEDCS: Identity Enabled Distribution Control System. Para o efeito foi necessário implementar uma Ebook Webstore, um WebService e um Player de reprodução dos Ebooks em formato de texto.

O objetivo deste sistema é garantir a máxima e possível segurança do serviço, utilizando os conhecimentos adquiridos na unidade curricular de Segurança. Para isso são necessários vários processos como por exemplo, a utilização de certificados HTTPS, a cifragem de todo o material existente, derivação de chaves e registo de utilizadores.

O relatório reflete todos os passos e decisões tomadas na criação do sistema, assim como uma análise ao que foi mostrado na primeira apresentação e decisões que se tomaram depois desta, tecnologias utilizadas, descrição dos vários processos existentes e conclusão.

2 JavaScript e WebGL

2.1 Arquitectura da implementação

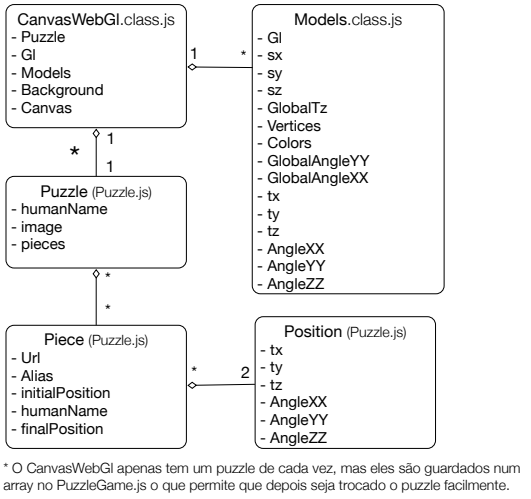

Figura 1: Diagrama da implementação desenvolvida

Na implementação desenvolvida procurou-se uma solução que permitisse reutilizar o código e instanciar quantas peças e puzzles fosse preciso. Para isso criou-se uma class Models, que tem como atributos os que estão descritos em cima, que instância um modelo, independente dos outros que irá fazer uso das translações, rotações e outros métodos usados durante as aulas práticas. Para isso, esta class irá, no seu construtor instanciar uma única vez dois buffers, um chamado *triangleVertexPositionBuffer* e outro *triangleVertexColorBuffer*, de resto, o *initBuffers* é sempre chamado sempre que for feito um *drawScene* para os arrays de buffers serem atualizados.

Já na class *CanvasWebGl*, é onde o puzzle instância todos os modelos (peças), aplica as translações globais e independentes, é desenhada a cena, inicializado o modelo de fundo e inicializado o WebGl.

Para alimentar a class *CanvasWebGl*, foi criada uma class *Puzzle* que tem todos os atributos necessários para instanciar um puzzle. O puzzle irá ter o seu *humanName* que será apresentado ao utilizador, a *image* que é a imagem final da solução do puzzle, e as *pieces* que são as várias peças do puzzle.

A peça, terá o URL sendo este onde será obtido a lista de vértices e de cores para a peça. O alias é o usado para identificar a peça, tem de ser único para todas as peças existentes no Puzzle, o *initialPosition* que identifica a posição inicial da peça e o *finalPosition* que identifica a posição final da figura.

3 Organização da implementação

3.1 Modelos

Foram criados vários modelos para cada puzzle, estes depois são usados para peças e estão localizados na pasta "modelos". Ests são ficheiros .txt como usados nas aulas onde armazenam a lista de vértices e de cores.

3.2 Javascript

Os ficheiros javascript onde está a maior parte da implementação, desde "listeners" a código de WebGl está guardado na pasta "js".

- *bootstrap.js* contem os scripts do bootstrap que são responsáveis por exemplo pelas modalbox, tooltips entre outros.
- *CanvasWebGl.class.js* como já foi explicado, contem a automatização necessária para instanciar um puzzle em WebGl.
- *confetti.js* contem o script necessário para a animação de confettis quando um utilizador termina o puzzle.
- *bootstrap.js* contem os scripts do bootstrap que são responsáveis por exemplo pelas modalbox, tooltips entre outros.
- *document.jquery.js* contem os scripts jquery necessários para a lupa quando o utilizador passa o rato por cima da pré-visualização do puzzle quando é terminado e a animação necessária para a pontuação do utilizador.
- *initShaders.js* contem os scripts necessários para inicializar cada shader program e para inicializar o shader. Este foi um aspeto importante no nosso desenvolvimento porque a percepção do que estes elementos faziam, além de outras coisas, permitiu-nos instanciar os modelos de forma independente.
- *jquery.js* é uma framework javascript que simplifica a vida ao programador, permite criar listeners de forma mais fácil e dinâmica com seletores, modificar atributos de objetos do documento HTML, fazer animações com esses objetos entre outras coisas.
- *maths.js* contem os scripts auxiliares usados nas aulas.
- *microscope.js* contem o script para instanciar a lupa na imagem.

- *Models.class.js* tal como descrito em cima, contem toda a informação necessária para instanciar uma peça de um puzzle e fazer as translações, rotações entre outras coisas do puzzle.
- *models.js* não necessitámos de olhar para este ficheiro, mas, contem as funções necessárias para processar as "triangle mesh models".
- *parseFiles.js* adaptando o código fornecido nas aulas práticas para fazer o processamento de ficheiros txt e obj, contudo a adaptação prendeu-se em fazer download via ajax de forma assíncrona dos ficheiros.
- *puzzle.js* contem as classes que permitem fazer programação ao estilo orientada aos objetos para definir o puzzle, as peças e as posições iniciais e finais.
- *PuzzleGame.js* é responsável por carregar o ficheiro puzzles.json que tem a definição dos puzzles em JSON e disponibiliza o runWebGL que irá instanciar o CanvasWebGl e irá chamar o setScreenPuzzle e o setEventListeners.
- *setEventListeners.js* contem os event listeners necessários para os controlos do puzzle e da página.
- *setScreenPuzzle.js* contem os scripts necessários para popular a página que é apresentada ao utilizador com toda a informação necessária.
- *webgl-utils.js* Copyright 2010, Google Inc, All rights reserved.

3.3 JSON

Para uma melhor definição de todos os puzzles usados no jogo foi criado um ficheiro JSON onde é possível criar todas as peças de cada puzzle e definir todos os atributos anteriormente detalhados. Este ficheiro é carregado inicialmente sempre que o jogo é iniciado, apresentando assim ao utilizador a lista de puzzles e inicializando o jogo com o puzzle inicial.

```
1 {
2   "puzzles" : [
3     {
4       "humanName": "Level 1",
5       "image" : "img/puzzles/puzzle1.png",
6       "pieces" : [
7         {
8           "alias": "triangulo",
```

```
            "url": "modelos/trianguloBlue.txt",
            "humanName": "Triangulo Blue",
            "initialPosition" : {
              "tx": 0.2,
              "ty": 0.4,
              "tz": 0.5,
              "angleXX": 225,
              "angleYY": 45,
              "angleZZ": 45
            },
            "finalPosition" : {
              "tx": 0,
              "ty": 0,
              "tz": 0,
              "angleXX": 0,
              "angleYY": 0,
              "angleZZ": 0
            }
          }
        ]
    }
  ]
}
```

4 Interface do utilizador