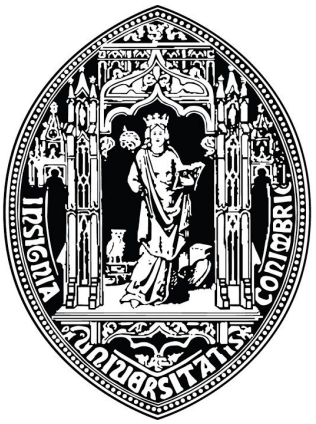
Departamento de Engenharia Informática

Computação Gráfica

2016/2017 - Cubo Rubik

• U  C •

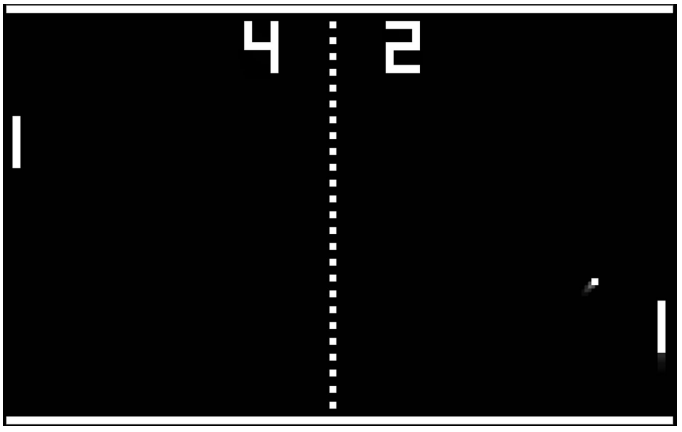
Relatório

Joaquim Ramires Ferrer 2015260670

Introdução

Neste projeto decidi implementar um jogo que tanto fosse apelativo visualmente como jogável e divertido. A ideia original, demasiado complexa para o tempo que dispunha, consistia numa espécie de Pong 3D, para apenas um jogador, dentro dum cubo Rubik cujos quadrados mexiam de acordo com as regras desse tipo de cubos. A plataforma seria transparente, as paredes refletiriam a bola e a plataforma e haveria um pequeno cubo que iluminaria a sala.

Infelizmente este projeto revelou-se demasiado ambicioso na parte das colisões , que não queria que fosse o foco principal do meu trabalho, tendo por isso deixado para trás a ideia de a bola embater diretamente no cubo rubik (com tudo o que isso tinha associado) tendo em vez disso construído um cubo transparente dentro do cubo rubik que se mexe normalmente. Adicionei também algumas features que mencionarei durante o relatório.



Pong original

Implementação

O jogo foi implementado recorrendo às bibliotecas do OpenGL para C. O código foi escrito num paradigma orientado a objetos recorrendo a C++. Foi também feito um esforço para o código ser o mais modular e abstrato possível de forma a ser fácil de adicionar e retirar features, assim como mudar parâmetros.

O código consiste para num header file para todas as declarações de objetos e funções e três ficheiro ‘cpp’:

- ‘Projeto.cpp’ que contém a chamada do main e as funções chamadas pelo OpenGl para ativar os modos necessários, desenhar a cena, inicializar texturas, chamar um timer que redesenha a cena e tratar dos eventos do rato e do teclado.
- ‘Rubik.cpp’ cujo conteúdo resume-se a todas as classes criadas por mim. São elas:
    - Rubik - Cubo Rubik com lógica associada
    - Platform - Uma plataforma transparente que se mexe num plano x=d.

    - Ball - Uma bola que se mexe e embate com o cubo abaixo.
    - Cube - Um cubo centrado na origem também transparente.
    - Particle - Classe para particulas com uma posição, velocidade e tempo de vida.
- 'RgbImage' que é o código dado pelo professor para fazer load de imagens.

Características e funcionalidades

As seguintes features estão por ordem cronológica de desenvolvimento e são as que, na minha opinião, dão valor ao trabalho.

1. Desenho e movimento do cubo Rubik.

   O cubo Rubik foi desenhado "tile" a "tile", face a face, utilizando a primitiva 'GL_QUADS' e de acordo com o estado que correspondem às cores de cada "tile". Associado a ele também está um eixo de rotação ativo(de 1 a 6 pois as filas centrais não estão a rodar) e o número de graus que esse eixo de rotação foi rodado. A cada momento de desenho, de acordo com o eixo de rotação ativo, determinadas faces são rodadas o número de graus correspondentes assim como as "tiles que estão coladas a essa face. A cada momento de update, o ângulo de rotação é aumentado e, caso chege a 90, o estado do cubo (e.g. as cores de cada face) é alterado de acordo com as o eixo de rotação que estava ativo. O novo eixo de rotação ativo é escolhido aleatoriamente depois disso e a rotação posta a 0.

   O update do estado foi algo que foi feito de forma "bruta". Para o estudo usei uma planificação do cubo que me ajudava a perceber que cores iam para onde após a rotação.
2. Desenho de um cubo transparente com textura dentro do cubo rubik.

   Este cubo transparente tem exatamente o tamanho do cubo rubik multiplicado por raiz de 2. Este tamanho garante que é o maior possível sem entrar em sobreposição com o cubo exterior.

   O cubo é desenhado sem face da frente de forma a ver melhor o que se passa dentro dele (dentro vai ser a área de jogo). De forma a podermos ver através do cubo exterior foi necessário aumentar a distância mínima que um objeto tem de estar do observador para ser projetado. Esta distância relacionou-se com a distância do observador ao cubo que precisava de ser grande o suficiente para ver cubo todo fazendo o jogo mais apelativo.

   O cubo, por ter uma cor transparente, teve de ser desenhado depois do cubo. À sua côr foi sobreposta uma textura para fazer parecer um vidro.
3. Desenho duma bola que salta dentro do cubo Rubik e se transforma em partículas quando "morre".

   A bola é desenhada com a função glutSolidSphere é atualizada de acordo com uma velocidade. De cada vez que se mexe testa sobreposições com as paredes do cubo e da plataforma. Caso saia do cubo então o jogador perdeu e a bola ganha uma animação com partículas que são criadas no momento com uma posição random, que forma uma bola do mesmo tamanho, saindo disparadas de

seguida. Isto é feito de forma não muito eficiente, criando partículas no uniformemente num cubo que circunscreve a bola e descartando a partícula caso não esteja dentro da bola.

4. Reflexão da bola em cada superfície do cubo interior.

   A bola é refletida em todas as superfícies do cubo. Para isso usamos o GL_STENCIL em cada parede desenhando uma bola correspondente à sua reflexão em cada parede.

5. Desenho duma plataforma transparente que se mexe de acordo com o rato.

   Para isto é feito um mapeamento das coordenadas do rato no viewport para as coordenadas desejadas no ambiente de jogo.

Bugs e Limitações

Em termos de bugs, felizmente, nada foi descoberto.

Em termos de limitações, as principais são o cubo apenas girar cada face num sentido e a velocidade da bola em termos horizontais e verticais não serem afetadas pelo sítio da plataforma onde tocam. Isto traria muito mais dinâmica ao jogo.

Conclusão

Durante o desenvolvimento deste projeto aprendi bastante sobre vários temas teóricos lecionados durante o assim como como trabalhar com biblioteca do OpenGL em si.

Sou da opinião que o meu trabalho aborda muitos dos tópicos importantes, sendo que apenas tive pena de não implementar luminosidade mas não consegui arranjar forma de a encaixar de forma não muito complexa no jogo.