CG 12/13 – Relatório

Puzzle Bubble

(Tema inicialmente proposto por Alexandra Cunha)

Grupo 24

Armindo Cunha

Rui d'Orey

Resumo

A ideia inicial deste trabalho proveio da colega Alexandra Cardoso, baseada no seu trabalho Didactic Bubble[1], realizado na disciplina de Sistemas Multimedia no ano lectivo de 2009/2010. Por sua vez, o jogo referido anteriormente é baseado noutro jogo famoso de nome **Puzzle Bobble** (a.k.a. **Bust-a-Move**).
Trata-se de um jogo de puzzles 2D que surgiu em 1994, criado pela empresa Taito Corporation[2] originalmente em máquinas arcade.
O objectivo do trabalho foi desenvolver uma versão 3D onde pudessem ser exploradas as várias temáticas requeridas. De forma a explorar a API OpenGL, mudou-se o tema de **desenhos animados japoneses** para **espaço** e inseriram-se alguns elementos visuais adicionais ao jogo.

Mecânica de Jogo

Foram estudadas as mecânicas e as regras de jogo original. As características que foram implementadas no trabalho em questão são:

- no início de cada jogada, uma arena de jogo rectangular contém, na parte superior, um padrão pré-estabelecido de esferas de cores variadas;
- na parte inferior da arena, o jogador controla um dispositivo chamado "pointer", que aponta e dispara esferas individuais para a parte superior da arena (onde se encontram as outras esferas);
- a esfera disparada viaja em linha recta de acordo com o ângulo dado ao apontador podendo mudar a sua direcção se embater nos limites laterais da arena e parando apenas quando colide com uma esfera da arena ou com o limite superior do ecrã;
- se a esfera disparada parar adjacentemente a uma ou mais esferas da mesma cor, e por sua vez formar com outras esferas adjacentes entre si, um conjunto de 3 ou mais elementos da mesma cor , estas são removidas do jogo, assim como todas as esferas que se encontram em jogo exclusivamente ligadas ao conjunto eliminado;
- a cor das esferas disparadas é gerada e escolhida a partir das cores das esferas que ainda existem na arena;
- quando não houver mais esferas em jogo, um novo nível será iniciado.

Alguns detalhes sobre a implementação

As estruturas usadas para representar o tabuleiro e as esferas são baseadas em outras usadas no trabalho de Tecnologias Web 2012/2013 efectuado por Armindo Cunha e Nuno Dias.

A implementação do sistema de jogo caracteriza-se pelos seguintes pontos:

- Matrizes multi-dimensionais representam os objectos;
- A posição da esfera em jogo é dada pelas suas componentes x e y. O movimento obtém-se ao modificar estes valores todos os frames, incrementando-os de acordo com o ângulo de lançamento. Este é calculado após ser pressionada a tecla 'SPACE' depois de usadas as teclas 'SETA PARA ESQUERDA' e 'SETA PARA A DIREITA', para escolher o ângulo. No movimento da esfera a velocidade da bola é constante, independentemente do ângulo de lançamento, como no jogo original.
- Quando ocorre uma combinação de esferas da mesma cor, são executados algoritmos que separam as esferas a sair do jogo, das que continuam em jogo. As esferas que fazem parte da combinação, movem-se para fora da área visível por deslocamento no eixo z. As esferas que estão ligadas exclusivamente às anteriores, movem-se para fora da área visível por deslocamento no eixo y, neste caso, com movimento acelerado.
- Quando o jogador limpa as peças de uma arena, um novo nível é iniciado. Uma matriz dedicada garante facilidade em inserir níveis personalizados;
- Se forem colocadas mais de 10 peças em linha desde o topo, o jogo entrará em modo game over, a cor das bolas mudará e o teclado será bloqueado.

Sistema de Colisões

O Sistema de Colisões presente no jogo assenta em trigonometria característica do jogo original Puzzle Bubble.

Cada vez que é efectuada uma modificação na posição horizontal da esfera em jogo, é verificado se o seu novo x excede os limites definidos pela borda lateral do jogo. Se ocorrer, a esfera inverte o sentido horizontal desse movimento, simulando o efeito de bater numa parede.

Quando uma esfera muda a sua posição, além de o programa verificar se ultrapassa os limites laterais, também verifica outro tipo de colisões.

As colisões da bola que está em movimento com outras bolas e com o topo do cenário são feitas da seguinte forma para cada frame:

- A cada uma das componentes do **centro**(x,y) da esfera em jogo, é somado à componente y um valor correspondente ao raio. À componente x é somado e subtraído esse valor. Desta forma pode-se obter as coordenadas dos extremos da esfera;

- É verficado, numa matriz auxiliar que contém a correspondência entre coordenadas OpenGL e posições relativas do tabuleiro (10 linhas e 8 colunas), quais as posições relativas do tabuleiro que "tocam" nas extremidades da esfera;
- Para cada uma dessas posições é verificado noutra matriz se lá existem bolas desenhadas ou se se trata do limite superior do tabuleiro;
- Ao encontrar uma destas posições ocupadas, o algoritmo anterior é interrompido, e é executado um novo algoritmo que verifica quais as posições livres adjacentes à bola em jogo e a bola com que esta colide;
- A bola em jogo é então colocada nessa posição.
- Se não for encontrada nenhum colisão, a bola em jogo altera a sua posição.

No desenvolvimento do sistema verificou-se que em casos muito ocasionais a colisão pode não ser correctamente calculada e a bola poderá inadvertidamente sair do tabuleiro. Por esse motivo implementou-se uma rotina extra de verificação da posição da bola. Esta rotina garante que se as coordenadas da esfera forem inválidas, esta voltará à posição de lançamento, de forma ao jogador poder repetir a jogada.

Modelos

Foram usados os seguintes ficheiros obj para os elementos gráficos presentes no trabalho :

- **esfera1.obj** - esfera com mapeamento de texturas usado para as esferas do jogo, o planeta e o cenário.
- **cube2.obj** - cubo com materiais e mapeamento de texturas usado para a arena de jogo e para o cubo extra terrestre (que está nas proximidades da zona do planeta com sistema de partículas).
- **textured_cube.obj** - cubo com mapeamento de texturas. Usado apenas para mostrar a informação sobre as teclas na vista normal.
- **triangle.obj** - triangulo usado para representar as partículas do sistema de partículas.
- **pointer.obj** - Toróide com cone acoplado e materiais que serve como apontador para o jogo.

Texturas

Foram utilizadas três texturas para o jogo, sendo que além da textura **base.dds**, que foi providenciada na framework, duas foram criadas para o jogo.

Figura 1 – Textura sky.dss

A primeira textura criada para o jogo - **sky.dds** -foi obtida através da imagem PNG retirada de uma pesquisa na Internet e representada na Figura 1. Esta foi convertida para o formato *dds* usando o programa ***Easy2Convert PNG to DDS***[3].
Inicialmente foi produzida para ser usada como cenário de jogo apenas, mapeada a um círculo e este aumentado. Depois de experimentada noutros objectos e com manipulações de luz, concluiu-se que poderia ser usada para outros fins sem parecer repetida porque criava objectos com aspecto distinto.
Esta foi usada nas esferas do jogo, que, combinada com variações da iluminação, serviu para obter o efeito de "berlinde" que emitia luz própria.
Foi usada também apenas para suporte de efeitos de luz nas barras que delimitam a arena do jogo e no sistema de partículas, tanto nas partículas como no planeta no centro.

Idealmente deveria ter sido feito mapeamento específico dos modelos para a textura sky.dds, especialmente para a esfera, de forma a evitar que os limites da mesma ficassem acentuados ao longo do objecto. No entanto, consideramos que a forma visual do mapeamento obtido por defeito era satisfatória.

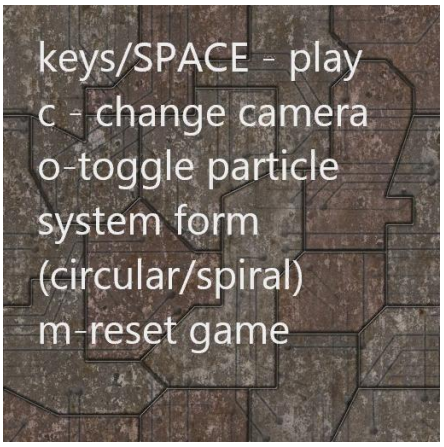

Figura 2 Textura info.dss

A segunda textura própria do jogo - **info.dds** - é uma cópia da textura **base.dds**, incluída na framework. Essa cópia foi feita no editor de texturas do **Visual Studio 2012**, tendo sido inserido texto com indicação das teclas do jogo por cima da imagem de textura original. É usada unicamente num cubo sem profundidade que está "colado" ao ecrã (visível quando se usa a câmara clássica).

A textura incluída no framework - **base.dss** - foi utilizada no cubo que navega aleatoriamente no espaço e no apontador de esferas.

Iluminação

Relativamente à iluminação, convém começar por indicar que os componentes em x e y do vector principal de posição de luz oscilam entre limites mínimos e máximos de respectivamente -1.0 e 1.0. Quando o valor minimo é atingido, a esse valor é somado, por frame, 0.003 até chegar ao máximo de 1.0. Aí inicia-se uma subtracção por frame de 0.003 até se atingir o valor de -1.0, recomeçando novamente o ciclo. Esta variação inicialmente foi implementada para ter impacto directo na luz das esferas do jogo, de forma que, ao sofrerem uma variação de direcção de luz ao longo do tempo, as esferas pudessem transmitir a ideia de que a luz vinha de dentro delas. No entanto, estes valores variáveis foram úteis também, para calcular valores de luz ambiente e difusa, rotação e posição de outros objectos.

De seguida, apresentam-se as características de luz de elementos específicos:

- **Globo/cenário,barras da arena e cubo com movimento aleatório**: a direcção da luz não muda, mas todos os seus componentes(excepto alfa) de luz difusa e ambiente oscilam entre 0.3 e 0.7 em simultâneo, de forma a dar um efeito subtil de luz no ambiente local;

- **esferas de jogo**: a direcção da luz muda conforme indicada no início desta secção, de acordo com a variação das componentes x e y da direcção da luz. Para dotar cada esfera de uma das três cores possíveis (azul, verde, vermelha), é alterada a sua luz difusa, colocando numa das 3 componentes de luz RGB o valor 1 e nas outras duas o valor 0, consoante a cor pretendida. Desta forma a textura **sky.dss** é descaraterizada da sua aparência original.
  Quando existe uma combinação de esferas e se dá a animação de destruição de esferas, as esferas destruídas são alvo de uma variação intermitente da sua componente principal de luz difusa até saírem do ecrã. As esferas que ainda se encontram em jogo têm a sua luz ambiente reduzida até 0.1, simulando um efeito de sombra ou perda de vida, sendo que o valor original de luz ambiente, 1.0, é recuperado após o final da sequência de destruição de esferas.

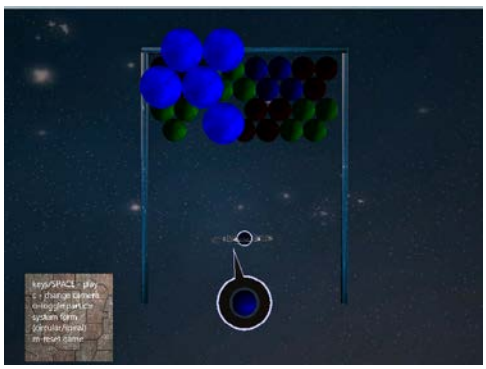

Figura 3 - Cor de esferas numa destruição.

- **Quadrado informativo com teclas:** este elemento tem luz constante em todas as suas componentes.

- **Partículas do sistema de partículas:** de forma que todas as partículas deste sistema tivessem uma luz dinâmica, todos os seus componentes, tanto de luz difusa, como ambiente são calculados de acordo com a distância da partícula ao centro do sistema de partículas. Parte-se de um valor 2.0 e subtrai-se o seu número de ordem do sistema a dividir por 15.

- **esfera do sistema de partículas e apontador de esferas:** para estes elementos poderem ficar com um rebordo de luz, desenhou-se, para cada um deles, dois objectos. Inicialmente desenha-se uma versão dos mesmos normal (tanto o planeta como a seta têm uma ligeira variação com o tempo das suas componentes difusas e ambiente, direccionadas pela variação do vector direcção de luz, referido anteriormente). De seguida, desenha-se uma versão com **Blending**, ligeiramente maior, no mesmo local e com alfa difuso próximo de 0. Desta forma, se não houver mais elementos desenhados, irá ser criado um elemento com uma cor uniforme e visível apenas na área exterior ao primeiro elemento. Isto acontece porque, como a transparência é total, o segundo elemento torna-se invisível, tendo o primeiro "dentro" dele. Como o calculo da transparência só é feito sobre os objectos que já estão no buffer, ao desenhar o resto dos elementos de seguida, o rebordo do segundo objecto mantém-se visível nos restantes elementos do cenário.

Figura 4 - esfera do sistema de partículas composta

Animação

Neste trabalho foram usadas as seguintes animações:

- rotação das esferas enquanto se deslocam e mudança da mesma quando ocorre uma colisão. Esta rotação é um efeito tridimensional não presente na versão original de Puzzle Bubble;
- escalonamento de esferas no inicio de cada tabuleiro. As esferas de um novo tabuleiro iniciam sempre com escala 0 e são dimensionadas para a sua escala normal antes do controlo ser dado ao jogador;
- destruição de esferas. Quando é realizada uma combinação, as esferas que fazem parte da mesma avançam em direcção à câmara clássica e as que ficam isoladas, caem;
- planeta que faz parte do sistema de partículas. As duas esferas que compõem o "planeta" do sistema de partículas são alvo de uma variação de escala dessincronizada, de forma a dar a noção de se tratar de uma fonte de energia. A esfera que aguarda ser jogada também usa a variável que controla a escala das anteriores para "piscar" e dessa forma chamar a atenção sobre si.

Sistema de partículas

Foi implementado um pequeno sistema de partículas que simula um sistema de asteróides à volta de um planeta. O sistema apresentado tem 120 partículas. É possível aumentar o número de partículas mas tal não foi feito pois, para manter alguma fluidez, seria necessário optimizações extra. As partículas movem-se de uma forma ordenada numa trajetória circular ou em espiral, consoante a escolha do jogador.

Cada partícula é representada por um triangulo que é desenhado em modo GL_POINTS, aparentando tratar-se de três partículas distintas quando de facto se trata de um objecto. Com isto pretende-se reduzir os cálculos realizados a cada frame de cor, posição, etc. A partícula tem uma valorização que representa o seu tempo de vida. Essa valorização, que aumenta todos os frames, é também usada como:

- raio no calculo da sua posição no circulo ou na espiral;
- intensidade da cor ambiente e difusa da partícula;
- controlo sobre a sua vida, sendo que quando é atingindo um valor máximo estipulado pelo programador, o seu tempo de vida é reiniciado, voltando a partícula ao estado inicial e consequentemente posição e cor;
- as partículas com valorização menor que 50 terão anexas a si mais duas partículas que serão deslocadas no eixo y, uma para cima e outra para baixo, um valor proporcional à sua distancia do centro (quando mais próximas, mais distantes entre si). A ideia é dar volume ao sistema.

Para efeitos de animação da transição de um estado circular para um estado em espiral, definiu-se um tempo de transição de 100 frames. Ao pressionar a tecla de alteração de estado, a contagem de tempo de vida das partículas é parada e, para cada partícula, é calculada a distancia entre o ponto actual e o ponto de destino e dividida pelo numero de frames de transição (100). A cada turno é percorrida a distância calculada. Ao terminar de percorrer a distancia, a contagem de tempo de vida é retomada.

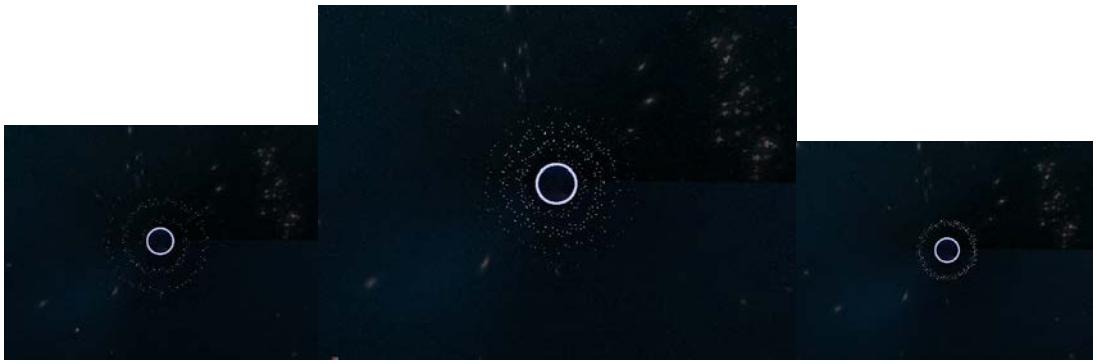

Figura 5a, 5b, 5c - Forma espiral(a), transição(b), forma circular(c)

Curvas de Bezier

De forma a dar resposta à necessidade de implementar curvas de Bezier e devido à falta de tempo, foi inserido no cenário do jogo uma espécie de cubo cuja razão de lá estar deverá ser vista não como falta de criatividade dos programadores mas como um desafio e estimulo à imaginação do jogador. Pode convencionar-se que se trata de uma entidade que realiza órbitas irregulares nas redondezas do sistema de partículas com algum intuito cósmico pouco claro.

Para sua implementação usou-se o algoritmo de Casteljau's [4].

O cubo desloca-se nos eixos x e y, sendo as coordenadas de destino e origem geradas de forma aleatória dentro de um determinado intervalo mínimo e máximo permitido (para evitar que o cubo se desloque para fora do ecrã).

Ao iniciar o jogo é definida uma curva de Bezier utilizando quatro pontos fixos. As coordenadas do cubo são determinadas utilizando os 4 pontos e a variação do valor de interpolação entre pontos, único valor que aumenta por frame, entre 0 e 1. No final do deslocamento (quando o valor da interpolação é 1), um novo conjunto de pontos é calculado.

Para manter a fluidez de movimento e evitar alterações bruscas de direcção e sentido, no algoritmo de geração de novas coordenadas, a coordenada inicial será sempre a coordenada de destino da curva anterior. Para manter a direcção, a primeira coordenada de controlo da curva será a última coordenada da curva anterior somada à distância entre essa e o segundo ponto de controlo da curva anterior .

Para inserir mais alguma dinâmica ao cubo, a sua rotação também varia em proporção à variação da componente x da direção da luz referida anteriormente.

Câmaras

No jogo foram implementadas 5 câmaras com alterações fluídas.

câmara clássica

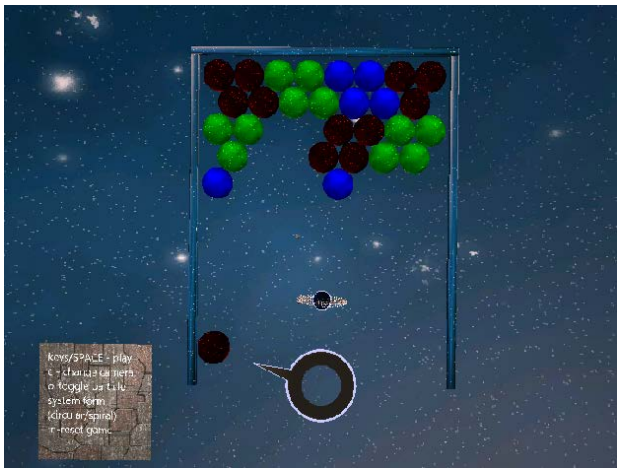

Figura 6.a - câmara clássica

Visão clássica do jogo Puzzle-Bubble em que o tabuleiro é totalmente visível na parte central do ecrã. Inicialmente a câmara está fixada numa posição em frente ao tabuleiro. Uma forma que permite ter algum movimento dinâmico da câmara nesta visão é utilizando o ponto de foco da câmara como o ponto de localização da mesma, adicionando co-seno e seno do ângulo actual de movimento da esfera às coordenadas x e y do foco. Isto permite que a câmara acompanhe o movimento da esfera sem variações muito bruscas de posição.

Câmara 3ª pessoa

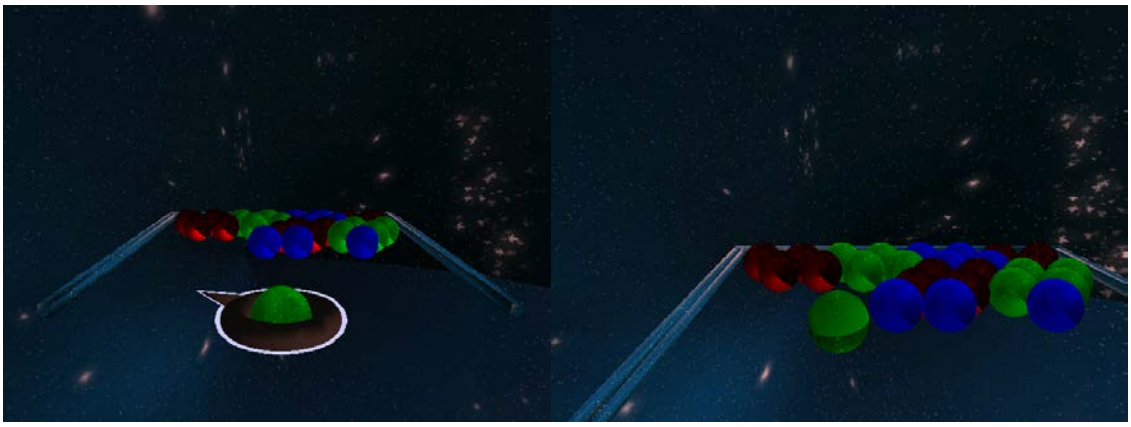

Figura 6.b.1 - Câmara 3ª pessoa com esfera no apontador.
Figura 6.b.2- Câmara 3ª pessoa com esfera em movimento.

Esta câmara foi implementada de duas formas no decurso do trabalho. Inicialmente a variação da sua posição era igual à variação da posição da esfera em jogo. Posteriormente para tentar

colocar transições com a esfera em movimento partindo da câmara clássica, implementou-se um sistema de variação de posição em que a câmara se mexe a uma velocidade fixa tentando apanhar a esfera, até a apanhar. A partir dai a sua variação é que começa a ser igual à posição da esfera. Isto por um lado evita mudanças súbitas na posição da câmara aquando de uma alteração de câmera mas pode resultar numa câmara tremula porque, tendo duas velocidades distintas a competir para irem dar ao mesmo ponto, poderá resultar numa visão tremida do jogo. No entanto, tanto uma como outra apresentam pontos a favor e contra. A direcção de câmara manteve-se fixa pois num jogo de Puzzle Bubble poderia ser pouco relevante e confusa, tendo em conta que, fora o tabuleiro que já é visível, uma mudança de câmara confundiria o jogador. Poderia, no entanto, adaptar o seu foco ao destino da esfera e não apenas só para cima.

Câmara 1ª pessoa

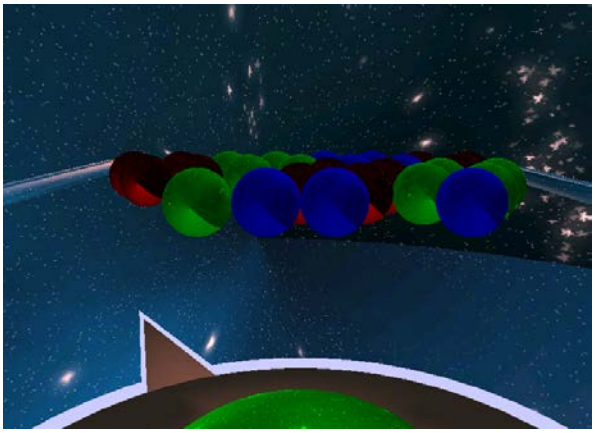

Figura 6.c- Câmara 1ª pessoa.

A implementação da câmara em primeira pessoa é similar à de terceira. Devido ao facto que para o jogador decidir uma jogada é necessário ver as peças todas, de forma a saber quais combinações são possíveis , é pouco funcional uma câmara posicionada no centro da esfera, pois incapacitaria o jogador de ver esferas posicionadas no final da arena. Por isso, a câmara está posicionada em cima da esfera para uma visão completa de tabuleiro. Para se ter noção da cor da esfera em jogo, o foco foi ligeiramente ajustado para baixo.

Câmara Cubo Misterioso

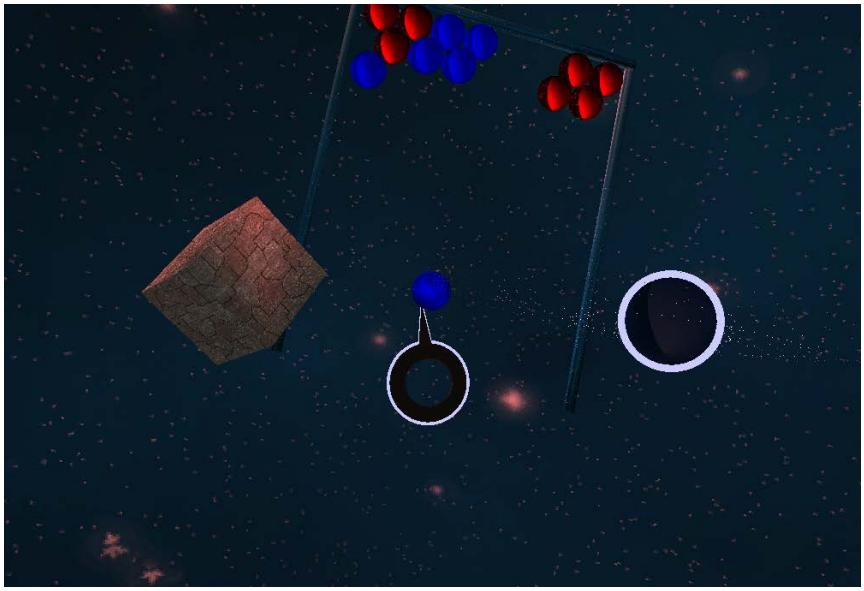

Figura 6.d- Câmara Cubo Misterioso.

Esta câmara permite acompanhar o movimento do cubo misterioso, mantendo sempre o foco principal no jogo. A câmara caracteriza-se pela sua posição no eixo x e y, que é a mesma do cubo. Mas o seu foco, por outro lado, é a esfera em jogo. Isto, aliado com o movimento fluido do cubo misterioso, permite criar um efeito mais cinemático do que as restantes câmaras.

Câmara Sistema de Partículas

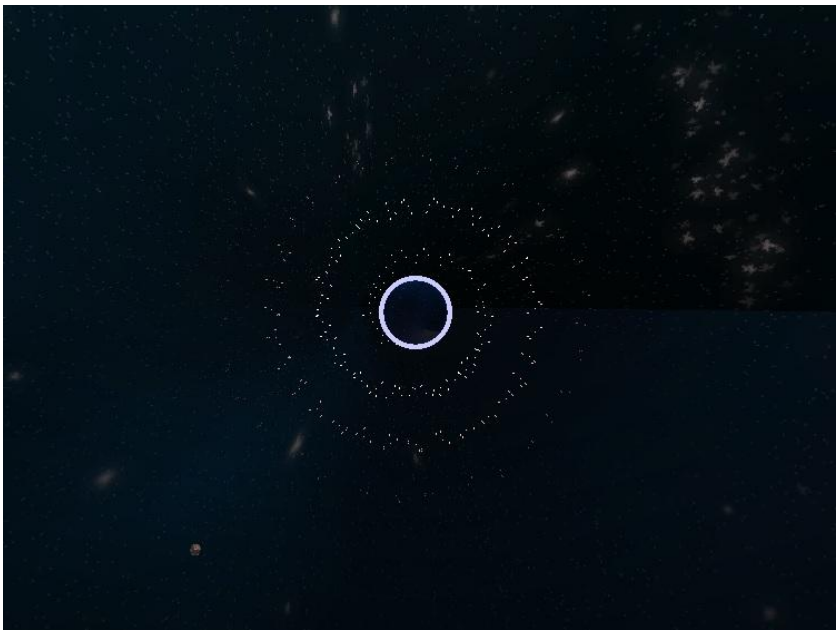

Figura 6.e- Câmara Sistema de Partículas.

A câmara posicionada por cima do sistema de partículas permite observar com mais detalhe o comportamento do sistema de partículas. A câmara é estática.

Alterações dinâmicas de câmara

Entenda-se que uma alteração dinâmica de câmara, no contexto deste trabalho, consiste na alteração não imediata de uma câmara para a seguinte, sendo visível para o jogador o deslocamento realizado de uma posição e de um foco de câmera para os correspondentes noutra câmara.

Existem dois sistemas que permitem alterações dinâmicas de câmaras no jogo:

1. Câmara clássica**:** esta câmara tem um sistema de retorno a uma posição óptima. Cada vez que a câmara se encontra com alguma alteração na sua posição ou foco, irá começar a tentar alterar a sua posição para uma posição default. Pode-se observar esse comportamento na introdução ao jogo, em que a câmara normal foi colocada numa posição e foco forçados e lentamente se posiciona no seu local definido por defeito. Outro caso em que se pode observar esse efeito é quando é concluída uma jogada e o foco passa para uma nova esfera. Tentou-se reproduzir o efeito nas outras câmaras mas devido a falta de tempo e algumas incompatibilidades com o próximo sistema, foram descartadas.

2. Outras câmaras: As restantes câmaras tem um sistema de transição animada entre si. Ao premir a tecla ‘c’, o jogo é posto em pausa, as distâncias entre a câmara actual e a próxima são calculadas e, durante 200 frames, é feito um movimento entre uma câmara e outra. No final do movimento o jogo é reposto. Esta pausa é muito pequena e ajuda a reduzir conflitos que existiam quando, por exemplo, a câmara se deslocava para o local do cubo mas quando lá chegava ele já não estava.

Extras

Para dotar o jogo de mais alguma imersão foi adicionada música e alguns efeitos sonoros utilizando chamadas MCI ao Windows.

Observações

O desenvolvimento e testes foram realizados em dois computadores com as seguintes características:
-um com processador i5, placa Intel HD4000 e 8 GB de RAM;
-outro com processador i3, placa ATI HD 5650 e 4 GB RAM.

Em ambos os computadores o jogo corre de forma fluida e sem abrandamentos. Mas nos computadores na faculdade verificamos alguns abrandamentos. Este problema poderia ser corrigido recorrendo a optimização de código, utilização de threads para separação de tarefas ou multiprocessamento que não tivemos tempo de explorar.

Embora possa ser melhorado, pensamos que este trabalho foi de grande valor por vários motivos:

- Permitiu-nos aprender a base de programação gráfica 3D;
- Explorar a nossa sensibilidade visual;
- Cooperar em equipa de forma autónoma e eficaz;
- Descobrir uma área de trabalho interessante e motivadora.

Por último, gostaríamos de agradecer ao Professor Miguel Coimbra e membros do PIC pela ajuda e orientação prestada neste trabalho.

Referências

[1] https://www.youtube.com/watch?v=NINOPgTYRqY&list=UUHzVVwnqVx30qv2NwbGrA2A&index=4

[2] http://en.wikipedia.org/wiki/Puzzle_Bobble

[3] http://www.easy2convert.com/png2dds/

[4] http://www.youtube.com/watch?v=YATikPP2q70