ColorPlay

Universidade de Aveiro

Licenciatura em Engenharia Informática

UC 45423 - Introdução à Computação Gráfica

Aveiro, 21 de Julho de 2021

Projeto desenvolvido por:

79900 – Frederico Avó

universidade de aveiro
theoria poiesis praxis

deti universidade de aveiro
departamento de eletrónica,
telecomunicações e informática

Índice

Introdução

O presente documento tem como principal objetivo descrever detalhadamente o projeto desenvolvido no âmbito da unidade curricular de Introdução à Computação Gráfica da Universidade de Aveiro.

O principal objetivo pretendido com a realização deste projeto seria o de o estudante aplicar diversas práticas de computação gráfica, lecionadas durante o semestre, tais como:

- Modelação e animação de objetos.

-Utilização de animação.

-Interatividade/controlo por parte do utilizador.

Objetivo do jogo

Foi desenhado um percurso, através de diferentes formas geométricas de diferentes cores, onde o objetivo é fazer com que o jogador chegue ao final do percurso (objeto de cor branca). Ao longo do percurso o jogador tem que adivinhar a cor da forma para onde se vai deslocar. Se o jogador não ativar a cor certa perde e o jogo começa automaticamente de novo.

O jogador perde também se não se deslocar por cima dos objetos.

O jogo é constituído por 3 níveis, sendo o nível 1 intuitivo e o mais fácil que tem como objetivo fazer com que o jogador entenda a lógica do jogo. O nível 2 será relativamente um pouco mais complicado e o nível 3 terá vários caminhos que não levam o jogador à peça branca, tendo o jogador que saber qual o caminho certo.

O jogador tem 60 segundos para conseguir completar cada nível.

Existem 5 cores possíveis: Branco, preto, azul, vermelho e verde.

Workflow

Inicialmente configurei a cena do three js, criei a camara perspetiva, escolhi a cor de background da cena, defini o tamanho do render do webgl, inseri o render webgl no html, ativei as sombras no webgl.

De seguida inicializei variáveis para a lógica do jogo. Defini uma variável que vai armazenar a função que conta o tempo, outra variável que controla qual é a fase atual, uma que controla se a lógica do jogo pode acontecer, uma que armazena a quantidade de passos dada em cada nível, outra que armazena o tempo, em segundos, em cada nível, em que caso passe de 60s o jogador perde.

Inicializei também uma variável que define se o jogador está em contacto com o “chão”, criei outra para definir qual a cor de blocos está ativa. Existe outra variável que armazena os objetos adicionados a cena, exceto as luzes. Implementei também uma variável para controlar a posição do jogador no mundo.

Por fim existe uma variável que define as cores de cada camada.

Principais funções

1. setupLight

Onde eu crio e adiciono luz ambiente indireta na cena. Criei também luz direta para a gerar sombras. Defini a posição x,y e z da luz direta. Configurei esta luz para que seja possível projetar sombras. Criei também uma simulação de camara ortográfica para a projeção das sombras.

2. setLayer

Função auxiliar que define qual cor de blocos está ativa. A variável "**activeLayer**" define que cor está ativa. Implementei um ciclo for que percorre cada elemento que foi adicionado à cena, na variável "**sceneObj**" e armazenei numa variável auxiliar "**b**".

Depois, através de uma instrução condicional, verifico se o item atual do ciclo for é um array ou um objeto único. Se for um array ele retorna e sai da função.

Se for um array, percorremos o array para deixar com transparência os blocos que não estão ativos. Verifico se o bloco não é branco nem preto. Se a cor do objeto não for a cor ativa deixo o bloco com opacidade 0.2. Verifico a cor ativa, baseado na definição de "camada" que criei quando instanciei o objeto.
Se a cor do objeto for a ativa, deixo o bloco com opacidade 1.

3. createObj

Nesta função instanciei os objetos, passando as seguintes variáveis para a função:

geo- objeto javascript no padrão {x,y,z} que usei para definir o tamanho do bloco;

color- string, onde defino qual cor está ativa;

pos- objeto javacript, no padrão {x,y,z} que usei para definir a posição do bloco;

name- defini o nome do objeto na cena.

Usei uma função padrão do three js para criar a geometria do cubo de acordo com o tamanho no eixo x,y e z. Usei também uma função padrão do three js para criar o material e "shader" do cubo, defini que o material tem transparência e fica com a cor que recebemos da função. Para definir a cor no objeto usei objeto javascript auxiliar "colors".

De seguida o objeto é criado, defini que o objeto vai receber e projetar sombras. Defini a cor do objeto na variável "layer", para consultar na função "SetLayer"**.** Defini também o nome do objeto na cena e sua respetiva posição. Por fim a função retorna o objeto.

4. isGrounded

O principal objetivo desta função é simular a física de forma simples. Usei uma função padrão do three js para emular um raio e para poder realizar a verificação de colisão. Defini a origem e a direção do raio. Através da variável “intersects” consigo verificar quais objetos estão abaixo do jogador.

Através de um if verifico se existe algum objeto abaixo do jogador, se não existir defino que o jogador não está próximo do chão e a simulação da física vai fazer com que o jogador caia.

Logo de seguida cálculo a distância do jogador para o objeto que o raio acerta, após isso arredondo para um múltiplo de 0.5.

Se a distância for igual a 1.5 e o objeto que estiver abaixo for da cor ativa na variável “activeLayer” ou se for preto defini que o jogador está no chão e então a simulação da física não faz com que o jogador caia. Se o objeto que estiver abaixo do jogador for branco significa que o jogador chegou ao fim do nível.
Por fim, se não existir nenhum objeto abaixo do jogador e a posição dele for menor que -10 no eixo Y, significa que o jogador perdeu.

5. cleanFase

Função responsável por zerar setInterval do cronómetro e apagar todos objetos da cena, menos as luzes.

6. fase1

Nesta função é criada a primeira fase. Defino por garantia que a primeira fase é a fase 1. Esta função reset às variáveis de acordo com a necessidade da fase atual. Configura a camara para olhar para o local onde o jogador é instanciado nos eixos x e y. Instanciamos o jogador, passamos o tamanho dele, a layer a que ele pertence a posição e o nome. Após instanciar o jogador esta função incrementa o jogador à variável que gere os objetos adicionados a cena.

De seguida são criados os blocos do chão da fase. Para isso temos que definir o tamanho, a cor, a posição e o nome. Na fase 1 a cor ativa é a azul.

Nas funções fase2 e fase3 o raciocínio é exatamente o mesmo, sendo apenas diferente as formas com que o chão é formado.

7. timer

Função que controla e atualiza o contador de tempo. Caso a lógia de jogo esteja ativa incremento 1 ao tempo. Caso o jogador demore mais de 1 minuto para terminar a fase, o jogador perde

8. animate

Solicito ao three js para renderizar a cena a cada frame. Caso a lógica de jogo esteja ativa, movemos sempre a camara 20 pontos à frente do jogador no eixo z, 8 pontos acima do jogador no eixo Y, e -8 pontos no eixo X. A camara é colocada a olhar para a posição do jogador abaixo 2 pontos no eixo Y. Atualizo o contador de passos de acordo com quantas vezes pressionaram alguma tecla de movimento. Depois é chamada a função que simula a física. É atualizada a posição do objeto jogador, através da variável sceneObj, na cena. Caso o jogador não esteja no chão, cai e perde. É chamada a função "fase1" que controla toda inicia todo fluxo de jogo.

Interação com o teclado

O programa controla a interação do teclado através de uma instrução condicional. As setas são responsáveis por controlar o movimento do jogador ao longo do percurso enquanto que as letras "B,R,G" são responsáveis por ativar a respetiva cor. Ao pressionar a seta para cima é alterada a posição **z** do jogador, na seta para baixo é alterada a posição **z** também, mas na direção oposta, na seta para esquerda/direita é alterada a posição **x** mas em direções opostas.

Já a tecla "B" é responsável por ativar a cor **azul**, a "R" por ativar a **vermelha** e a "G" por ativar a **verde**. O jogador clica sobre a tecla de espaço para reiniciar o jogo.

Conclusão

Durante a realização deste projeto consolidei os meus conhecimentos em three js e acho que consegui abordar todos os tópicos propostos. No futuro pretendo aumentar o número de níveis do jogo.

Referencias bibliográficas

https://riptutorial.com/three-js/example/17088/object-picking--- raycasting

https://threejs.org/~

https://blog.logrocket.com/intro-to-three-js-for-game-developers/