Universidade Federal de São Carlos

Trabalho Final: Snake

Bacharelado em Ciência da Computação

Computação Grafica

Prof. Mario Lizier

André Domingues Vieira

Gustavo Eda

Thiago Borges

Sorocaba

Julho de 2017

Sumário

1 Introdução

Nosso trabalho tem como objetivo trazer um classico dos anos 90-00, o famoso snake, jogo onde controlamos uma cobra e seu objetivo e se alimentar e ir crescendo, cada vez que ela come um alimento seu tamanho aumenta, chegando ao ponto de preencher a tela toda.

2 Desenvolvimento

2.1 Ferramentas

O jogo foi desenvolvido em WebGL (Web Graphics Library), que se trata de uma API em JavaScript que oferece suporte para renderização de gráficos 2D e 3D. O WebGL é baseado no OpenGL ES 2.0 (OpenGL para Sistemas Embarcados). Uma característica marcante do WebGL é o fato que os gráficos por ele criados, rodam diretamente da GPU (Graphics Processing Unit), aliviando a carga do processador para executar outras coisas. Utilizamos também a biblioteca Three.js que é compatível com todos navegadores que suportam WebGL e que tem como objetivo abstrair as complexas funcionalidades do WebGL. O jogo roda diretamente no browser, sem a necessidade de plug-ins, o que torna a sua execução rápida.

2.2 Lógica do jogo

Antes de se preocupar com texturas, iluminação e outras coisas, o primeiro passo era entender o funcionamento do jogo e implementar bem essa parte, para não enfrentar problemas na hora de aplicar texturas e iluminação. Basicamente temos dois objetos principais: a cobra e a maçã. A cobra é o personagem principal do jogo e tem como objetivo comer o maior número possível de maçãs, aumentando assim o seu tamanho, porém ela deve não pode de maneira alguma atingir as bordas do terreno e nem "cometer suicídio", que caracteriza-se em atingir seu próprio corpo.

2.3 O começo

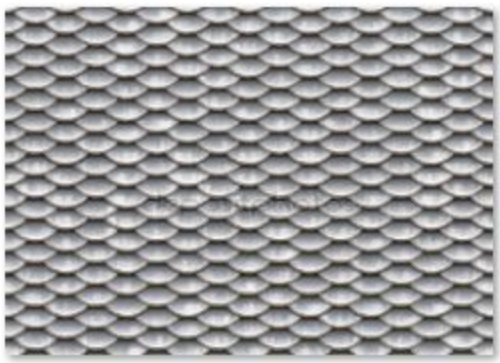
Com a lógica do jogo implementada, passou-se a pensar nas primitivas que definiriam os objetos. Foi definido que a cobra seria uma série de cubos alinhados com a cabeça de cor diferente para diferencia-la dos demais cubos. A maçã obviamente uma esfera que ficaria rotacionando todo o tempo. Inserir as primitivas no código foi uma tarefa simples, pois com poucas linhas de código era possível exibir elas na tela. O código abaixo mostra como criar um cubo amarelo e adicioná-lo a cena.

```
1        var geometry = new THREE.BoxGeometry(x,x,x);
2        var material;
3        material = new THREE.MeshBasicMaterial( { color: 0xffff00 }
             );
4    cube[qtdCube] = new THREE.Mesh( geometry, material );
5    scene.add(cube[qtdCube]);
```

2.4 Texturas

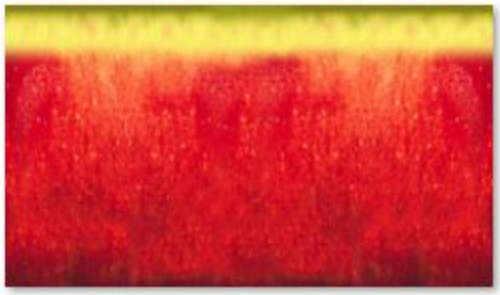
Com a lógica bem implementada e as primitivas definidas, passou-se a considerar que texturas aplicar para as primitivas. Seguindo o modelo da primeira versão, a cobra foi texturizada usando imagens de cobras de cores diferentes para cabeça e corpo.

piton



cabeca

A maçã foi a mais simples de se definir pois se trata de um objeto que não possui muitas variações em cor e textura.



apple

As paredes foram definidas como muros de tijolos(muro) e o chão um gramado(cesped), habitat natural das cobras.

muro

cesped

2.5 Iluminação

A iluminação é, sem dúvida, uma das partes mais importantes do jogo, pois torna tudo com cores mais vivas, mais intensas e uma aparência mais real aos objetos. Primeiramente foi pensado em apenas duas luzes direcionais nas posições (-0.5,1,0) e (0,1,1) e uma luz ambiente. Uma luz direcional é definida no three.js da seguinte maneira:

```
var directionalLight = new THREE.DirectionalLight(0xffffff);
directionalLight.position.set(1, 0, 0).normalize();
directionalLight.castShadow = true;
scene.add(directionalLight);
```

Contudo, depois da inserção das paredes, a esquerda ficou completamente escura, o que não dava uma aparência boa a cena. Então foi decidido colocar mais uma fonte de luz na posição (1,0,0), que iluminou a parede da esquerda.

O three.js trabalha com 3 tipos diferentes de materiais: MeshBasicMaterial, que é o tipo mais simples e que não é afetado pela luz, MeshLambertMaterial, que utiliza a lei dos cossenos vista em aula para tratar a iluminação difusa e o MeshPhongMaterial, que usa o método de iluminação phong, também visto em aula. No jogo, temos a cobra e a maçã usando Phong, pois trata-se dos objetos principais do jogo e recebem luz direta sobre eles, as paredes usam Lambert, pois a luz que chegam nelas devem ser refletidas difusamente para não atrapalhar o usuário com muitos reflexos e finalmente o terreno foi feito com o Mesh básico para reduzir ao máximo qualquer tipo de distração (como sombras) que dificultasse o usuário na hora de jogar.

3 Dificuldades

As principais dificuldades foram:

• aprender e utilizar bibliotecas gráficas, o que não estamos habituados, pois foi a primeira atividade na graduação onde trabalhamos com parte gráfica;

• pensar e resolver problemas do jogo como testar colisões da cobra com as bordas do mapa ou do seu próprio corpo, aumentar o comprimento da cobra;

• Adicionar curva de bezier no jogo;

• como posicionar as luzes de forma a iluminar a cena de forma correta;

• eliminar bugs, tais como se o usuário mudasse de direção muito rápido com o teclado, a cabeça da cobra se voltaria contra o seu corpo, o que não pode acontecer;

4 Conclusão

A implementação de um jogo relativamente simples como esse se mostrou um desafio e tanto para o grupo. Foi uma experiência muito boa para ter uma ideia da complexidade que é desenvolver jogos que possuem gráficosproximos a realidade e mesmo assim não sobrecarregam o computador. Foi bom também pelo fato de sair um pouco da teoria e partir para prática em um viés totalmente inexplorado até então na graduação. Ver todas aquelas equações, matrizes e modelos matemáticos se transformarem em algo palpável foi muito gratificante.