Universidade Federal de São Carlos
Campus Sorocaba
Computação Gráfica

Atividade Prática
Entrega Final

Prof.: Mário Lizier
Discentes:
RA:489999 Gustavo de Almeida Rodrigues
RA:552585 João Vitor de Sá Medeiros Santos

Sumário

Introdução

O projeto consiste em desenvolver uma pequena animação em ThreeJS, que é uma biblioteca de Javascript para a API gráfica WebGL.

A animação, por definição de tema de projeto, deve ser iterativa, portanto os alunos decidiram que seria feito um jogo.

O jogo desenvolvido nesse projeto é baseado na franquia de jogos Megaman, da companhia de jogos Capcom. O jogo foi lançado na plataforma “Super Nintendo”, implicando que o jogo é 2D.

Objetivo

O projeto tem como foco principal concretizar as propostas e expandir ensinamentos adquiridos em sala de aula, na matéria de Computação Gráfica, colocando-as em execução num projeto prático. Portanto, o jogo é um fruto do esforço mútuo da dupla de alunos, que durante o semestre desenvolveu o projeto em 3 etapas incrementais. A intenção é que, com os conhecimentos adquiridos, seja dada continuidade no projeto fora da disciplina.

Estrutura do Jogo

O jogo, à uma primeira impressão, parecer ser um jogo apenas em duas dimensões, como nos antigos consoles para qual a franquia do jogo Megaman foi desenvolvido. Porém desenvolver um jogo simplesmente em duas dimensões não atingiria a maioria de técnicas 3D abordadas em sala de aula, nem mesmo de câmera. Por isso dizemos que é uma simulação em 2D. Se a câmera orbitar ao centro do “mundo” do jogo, veremos o segredo do truque de simulação: uma subreposição de planos. Os planos mais ao fundo, são o que chamamos de *background* e *foreground*, responsáveis pela sensação estar dentro de uma cidade futurística.

Não obstante, representamos também o personagem principal (Megaman), através de um plano. Neste caso, ele é o plano mais a frente dos outros e posicionado na eixo Y de forma que pareça estar sobre o *foreground*. Os disparos realizados ficam na mesma profundidade do personagem, para que se possa detectar colisão (seja com parede ou com inimigos). E assim como o Megaman, os possíveis inimigos

também são um plano, porém com a mesma prerrogativa de estar com a mesma profundidade, para serem “atingíveis” por projéteis.

Para que a simulação seja fiel ao jogo original, tivemos de lançar mão de uma tarefa um pouco dificultosa: para todo plano, uma *sprite* teve de ser aplicada. Uma *sprite* consiste em uma imagem “mapa de bits” bidimensional, que era amplamente usada em desenvolvimento de jogos 2D. As sprites foram facilmente encontradas em *http://www.sprites-inc.co.uk/sprite.php?local=/X/,* entretanto teve-se de recortar e tratar cada sprite usada utilizando o programa Photoshop e atribuir manualmente no código. A movimentação do personagem, faz com que as *sprites* sejam trocadas rapidamentes, de forma que pareça que ele está animado.

O único objeto na cena que não obedece esta regra de planos, é a esfera que simula uma lua. O motivo de ela estar no jogo é para que possamos trabalhar com luzes e superfícies com diversas normais. Sendo assim, ela é suscetível à uma aparente distorção, devido à câmera ser uma projeção perspectiva. Então, para causar um efeito mais apocalíptico ao cenário futurista, uma sombra ambiente foi colocada na cor carmim, e a luz que a lua recebe, alguns tons mais claro. Note que a textura da lua é apenas cinza, e é todo o trabalho da luz que a deixou com a aparência avermelhada.

Etapas de Desenvolvimento do Projeto

Entregas

As entregas foram incrementais e divididas em 3 etapas:

Entrega 1 – Consistiu da cena inicial básica com os planos de fundo e o Megaman, com uma interação do usuário, que era fazer o Megaman andar. A animação era a do próprio Megaman, enquanto ele se movimentava.

Entrega 2 – Consistiu da adição de iluminação e um custom Shader, e a implementação da curva de Bézier.

Entrega 3 – É referente à entrega final, onde adiciona-se ao menos uma textura (já havíamos usado o tempo todo), correções e melhorias das fases anteriores, e colisão entre objetos.

Divisão de Tarefas

A divisão de tarefas era atualizada à cada nova entrega. Mas inicialmente o tema do jogo foi proposto pelo aluno Gustavo, dentre diversos outros temos sugeridos por ambos discentes. Havendo concordância, o projeto começou a ser executado. Boa parte da primeira entrega, que consistia em sua maioria no tratamento das *sprites*

e definições de planos, ficou à cargo do aluno Gustavo, por ter mais habilidade com o programa Photoshop.

No momento da segunda entrega, ficou acordado que o aluno João Vitor implementaria a maioria das funcionalidades. Entretanto no decorrer da fase, nos deparamos com o assunto que ambos tiveram imensa dificuldade. Isto explica a simplicidade da segunda entrega

Assim sendo, no tempo de desenvolvimento da terceira fase, ambos participaram mais ativamente do que nunca, visto que é a parte mais empolgante do jogo, onde vemos as coisas se tornarem parecidas com o jogo original. Assim sendo, correções de movimentação e colisão foram mudadas e melhoradas por ambos, e também a detecção de colisão dos projéteis com os personagens inimigos, modelagem da luz e sombreamento da lua, inserção e controle de projéteis ativos e documentação do código.

Dificuldades

Logo de início, a primeira dificuldade que encontramos foi lidar com uma linguagem que nunca havíamos usado: JavaScript. Após muito tempo dedicado à pesquisas em fórums e documentações começamos a entender o JavaScritp em si, para entender o ThreeJS.

Ainda quanto a base inicial do jogo, tivemos dúvidas sobre questões quanto Computação Gráfica no que diz respeito a cena, e câmera e renderizador. Não sabíamos como conseguiríamos fazer com que a câmera ficasse exatamente de frente aos planos. Pesquisando exemplos famosos, do próprio criador do ThreeJS, tivemos um norte para seguir, o que resultou no sucesso da primeira entrega.

Quanto à segunda entrega, nos vimos próximos ao fracasso. Devido a um problema com o “tick” (tempo do jogo), que era feito com animação e muito difícil de trabalhar, existia um problema com os cálculos (da curva de Bezier, que inicialmente não tínhamos ideia de como aplicar) e com a execução (em ordem e/ou execução unitária) das sprites. Para agravar a situação em que nos encontrávamos, nos deparamos com uma limitação da biblioteca: o material dos planos. Em ThreeJS, um objeto é composto por uma geometria, e um *material* (basicamente metadados de um shader), porém não conseguimos nem ao menos utilizar uma luz comum nos objetos. Testamos até uma luz que atingia a esfera onde está a lua, mas não obtivemos sucesso. A limitação encontrada, é que devido aos elementos da cena serem apenas um Plano, eles tem apenas uma normal (duas, caso você defina que ela tenha a parte

de frente e a de trás), ou seja, mesmo que houvesse incidência de luz, o efeito não seria notado. Além disso, o material que usamos, era o *MeshBasicMaterial*, o qual não aceita efeitos de luz. Porém não podíamos usar um material diferente, pois a geometria de plano aceita apenas este tipo de material ou um material feito por um *custom shader*.

A terceira fase foi nossa redenção, após o sucesso parcial da segunda, mas mesmo assim, as implementações da segunda fase eram nossa maior dificuldade. Ao desenvolver o *custom shader*, recorremos à diversos sites e fóruns, e até mesmo encontramos um em que o desenvolvedor consegui aplicar o sombreamento num plano 2D, como o nosso, alterando as normais. Resolvemos nos basear no tal projeto, mas não obtivemos sucesso. Não descobrimos o motivo, mas o erro intermitente dizia que a nossa funcionalidade *GetElementById('ThreeJS')* retornava NULL, ou seja, não encontrava a <*div*> com este nome, mesmo ela estando declarada.
O projeto em que nos baseamos para fazer o custom shader pode ser encontrado em https://csantosbh.wordpress.com/2014/01/09/custom-shaders-with-three-js-uniforms-textures-and-lighting/ e a nossa tentativa de criar um custom shader, separadamente primeiro, para após implementar no nosso jogo, na pasta do próprio trabalho, chamada "teste-shader" ou em https://github.com/Darkkgreen/trab-CGrafica/tree/gh-pages/teste-shader

Ferramentas Utilizadas

GitHub, Three.js, Atom IDE e Browser Mozilla Firefox.

Manual

Para executar o jogo, basta abrir o arquivo .html em seu Browser, ou acessar http://darkkgreen.github.io/trab-CGrafica/

O objetivo do jogo é bem simples: destruir o inimigo!
Para isso você deve movimentar o Megaman, para que ele desvie dos ataques do oponente, e esteja em posição para atirar.

Para movimentar o Megaman, utilize as teclas W,A, e D, sendo que A e D movimentam o megaman para a esquerda e para a direta respectivamente, e W faz com que ele pule. Para atirar, basta pressionar espaço que um projétil é disparado. Você também pode realizar ações enquanto anda para os lados, seja pular ou atirar. Experimente!

Caso você morra, ou destrua o oponente, você pode clicar no botão do canto superior para reiniciar o jogo.

Para informações sobre os desenvolvedores clique no botão com “?” desenhado.