Universidade Federal de São Carlos - UFSCar

Bacharelado em Ciência da Computação

Trabalho da Disciplina de Computação Gráfica

Jogo de Adivinhação: Welcome to Space!
Quanto tempo o astronauta demorou?

Carlos Tadeu Castilho dos Santos	726505
Sofia de Almeida Machado da Silveira	726589
Vinícius Silva Salinas	726594

2018/1

Sumário

1. Apresentação

Essa seção irá apresentar as principais informações sobre o trabalho: o que é, como funciona, quais foram as motivações para a realização do mesmo.

1.1. Tema

O tema escolhido para o trabalho é “Espaço”. Por essa razão, o trabalho consiste em um jogo com objetos como espaçonaves, um astronauta, o planeta Terra e a Lua.

1.2. Objetivo

O principal objetivo do trabalho foi implementar os conceitos estudados na Disciplina de Computação Gráfica. Para isso, o grupo optou por implementar tais conceitos por meio da criação de um jogo de adivinhação, no qual o usuário deve observar a trajetória de astronauta do Planeta Terra até a Lua e adivinhar o tempo que ele levou para sair do ponto inicial e chegar ao ponto final.

1.3. Funcionalidades

As funcionalidades do jogo são apresentar o movimento de um astronauta que vai desde o Planeta Terra até a Lua com velocidades diferentes a cada iteração, contar o tempo que o astronauta demora, receber um valor de entrada do usuário com um palpite de quanto tempo o trajeto do astronauta demorou, comparar a entrada do usuário com o tempo decorrido, calcular e mostrar na tela a pontuação do usuário com base na proximidade do palpite com o tempo que realmente demorou o trajeto.

2. Desenvolvimento

Essa seção irá descrever como ocorreu o desenvolvimento do projeto ao longo do semestre de forma que fosse possível implementar os requisitos obrigatórios e colocar em prática o jogo imaginado pelo grupo.

2.1. Divisão de Tarefas

De forma geral, ao longo de todas as fases, o grupo deu preferência a encontros presenciais com todos os integrantes a fim de facilitar a comunicação e a troca de ideias para o desenvolvimento do código do projeto.

Dessa forma, a divisão de tarefas ocorreu somente quando o encontro presencial não era possível ou quando os requisitos daquela fase não puderam ser terminados durante o encontro, fazendo com que cada integrante ficasse responsável por uma parte dos requisitos restantes.

Contudo, independente dos encontros presenciais e da divisão de tarefas, a comunicação do grupo permaneceu boa ao longo de todo o semestre, permitindo que o

desenvolvimento fosse fácil, de forma cooperativa e sem conflitos, já que todos os integrantes estavam cientes do que estava sendo implementado.

2.2. Dificuldades

A principal dificuldade encontrada ao longo do desenvolvimento, independente da fase do projeto, foi entender como os conceitos estudados nas aulas teóricas deveriam ser utilizados na prática na implementação dos requisitos. Para sanar as dúvidas que surgiram ao longo do desenvolvimento, além dos slides fornecidos pelo Professor Mario e das anotações feitas em sala de aula, o grupo buscou outros meios e recursos, como, por exemplo, o livro “Web GL - Up and Running” e também a troca de ideias com colegas de sala.

Outro fator que tornou o desenvolvimento um pouco mais difícil foi o fato de um dos integrantes não cursar a Disciplina Desenvolvimento para Web e dos outros integrantes ainda estarem aprendendo como programar para Web, já que essa disciplina foi oferecida no mesmo semestre que Computação Gráfica. Para sanar dúvidas sobre HTML5 e JavaScript foram utilizados o material da disciplina de Desenvolvimento para Web e os sites https://www.w3schools.com e https://stackoverflow.com.

2.3. Bibliotecas Utilizadas

Optamos por utilizar a `API WebGL`, implementando, portanto, o trabalho em `HTML5` e `JavaScript`. Outra API escolhida para a realização do trabalho foi a biblioteca `JavaScript Three.js`, que utiliza `WebGL`. Essa biblioteca possui recursos importantes que nos auxiliaram ao longo do desenvolvimento do trabalho, como, por exemplo, as funções `OBJLoader()`, para carregar um arquivo `.obj` e `RequestAnimationFrame()`, que cria animações suaves, fazendo com que o browser administre a taxa de quadros. Vale ressaltar que essa biblioteca está disponível sob a licença MIT.

3. Manual

Esta seção irá apresentar o passo a passo de como jogar o jogo de adivinhação desenvolvido pelo grupo, de forma a promover uma melhor experiência do usuário.

3.1. Como Jogar

Primeiramente, o usuário deve clicar no botão “Jogar” na tela inicial. Em seguida, ele será redirecionado à página do jogo, onde deve observar a trajetória do astronauta da Terra até a Lua. Quando o astronauta chegar à Lua e parar, aparecerá na tela do usuário um alerta informando-o que ele deve adivinhar o tempo que o astronauta levou de um ponto a outro. Haverá um campo no rodapé da página para que esse valor seja inserido. Após a inserção do palpite do usuário, é feito um cálculo da pontuação com base na proximidade do palpite ao tempo realmente decorrido. O usuário pode fazer, no máximo, 1000 pontos.

Em suma, os principais passos para jogar o jogo de adivinhação podem ser descritos como na tela inicial:

1. Clique em “Jogar” para começar o jogo
2. Espere o astronauta ir da Terra até a Lua
3. Tente adivinhar quanto tempo ele levou

Imagem 1: Tela inicial do Jogo de Adivinhação

4. Requisitos Atendidos

Essa seção irá mostrar os requisitos obrigatórios que foram atendidos e também irá indicar na prática como esses requisitos aparecem na versão final do jogo.

4.1. Dois objetos carregados de arquivos

Esse requisito foi atendido, pois todos os objetos exceto a Terra e a Lua foram carregados de arquivos. Portanto, um total de 5 objetos foram carregados de arquivos (4 espaçonaves e um astronauta). Isso pode ser observado nos seguintes trechos de código:

```
function addSpaceship()
{
    var loader = new THREE.OBJLoader();
    loader.load('./Spaceship.obj', function(object){
        spaceship_obj = object;
        spaceship_obj.position.set(50, -40, 10);
        spaceship_obj.scale.set(4.2, 4.2, 4.2);
        spaceship_obj.rotation.y = 3.2;
        spaceship_obj.rotation.x = 0.4;
        scene.add( spaceship_obj );
    });
}
```

Imagem 2: Carregamento da primeira nave `Spaceship.obj`

```
function addCraft()
{
    THREE.Loader.Handlers.add( /\.dds$/i, new THREE.DDSLoader());
    var mtlLoader = new THREE.MTLLoader();
    mtlLoader.setPath( './obj/craft/' );
    mtlLoader.load( 'craft.mtl', function( materials ) {
        materials.preload();
        var loader = new THREE.OBJLoader();
        loader.setMaterials( materials );
        loader.setPath( './obj/craft/' );
        loader.load( 'craft.obj', function ( object ) {
            object.position.set(-100, -30, 0);
            object.scale.set(2.2, 2.2, 2.2);
            object.rotation.y = 2.5;
            object.rotation.x = 0.9;
            craft = object;
            scene.add( craft );

        });
    });
}
```

Imagem 3: Carregamento da segunda nave `craft.obj`

```
function addScifiFighter()
{
    THREE.Loader.Handlers.add( /\.dds$/i, new THREE.DDSLoader());
    var mtlLoader = new THREE.MTLLoader();
    mtlLoader.setPath( './obj/SciFi-Fighter/' );
    mtlLoader.load( 'SciFi_Fighter_AK5.mtl', function( materials ) {
        materials.preload();
        var loader = new THREE.OBJLoader();
        loader.setMaterials( materials );
        loader.setPath( './obj/SciFi-Fighter/' );
        loader.load( 'SciFi_Fighter_AK5.obj', function ( object ) {
            object.position.set(200, -50, 100);
            object.scale.set(0.015, 0.015, 0.015);
            object.rotation.y = 1;
            object.rotation.x = 1;
            object.rotation.z = 1.5;
            scifi_fighter = object;
            scene.add( scifi_fighter );

        });
    });
}
```

Imagem 4: Carregamento da terceira nave `SciFi_Fighter_AK5.obj`

```
function addAircraft()
{
    THREE.Loader.Handlers.add( /\.dds$/i, new THREE.DDSLoader());
    var mtlLoader = new THREE.MTLLoader();
    mtlLoader.setPath( './obj/E-45-Aircraft/' );
    mtlLoader.load( 'E 45 Aircraft_obj.mtl', function( materials ) {
        materials.preload();
        var loader = new THREE.OBJLoader();
        loader.setMaterials( materials );
        loader.setPath( './obj/E-45-Aircraft/' );
        loader.load( 'E 45 Aircraft_obj.obj', function ( object ) {
            object.position.set(160, 28, 0);
            object.scale.set(4.2, 4.2, 4.2);
            object.rotation.y = 2.4;
            object.rotation.x = 0.8;
            aircraft = object;
            scene.add( aircraft );

        });
    });
}
```

Imagem 5: Carregamento da última nave `E 45 Aircraft_obj.obj`

```
function addSpaceman()
{
    THREE.Loader.Handlers.add( /\.dds$/i, new THREE.DDSLoader());
    var mtlLoader = new THREE.MTLLoader();
    mtlLoader.setPath( './obj/Astronauta/' );
    mtlLoader.load( 'spaceman.mtl', function( materials ) {
        materials.preload();
        var loader = new THREE.OBJLoader();
        loader.setMaterials( materials );
        loader.setPath( './obj/Astronauta/' );
        loader.load( 'spaceman.obj', function ( object ) {
            object.position.set(20, 28, 32);
            object.scale.set(7.2, 7.2, 7.2);
            object.rotation.y = -2.4;
            object.rotation.x = 0.5;
            spaceman = object;
            scene.add( spaceman );

        });
    });
}
```

Imagem 6: Carregamento do astronauta `spaceman.obj`

4.2. Cinco objetos no total

Esse requisito foi atendido, já que o jogo possui um total de 7 objetos principais: quatro espaçonaves, um astronauta, o planeta Terra e a Lua. Além desses objetos, também podemos considerar os objetos simples de nuvens da Terra e de estrelas no espaço.

4.3. Dois movimentos distintos

Esse requisito foi implementado. Todos os objetos exceto a Lua possuem movimento. Três objetos movimentam-se com a implementação da Curva de Bézier, as espaçonaves e o astronauta. Essas duas naves também possuem um movimento de rotação enquanto fazem a Curva de Bézier. A Terra, as nuvens da Terra e outra espaçonave também possuem movimento de rotação. A última espaçonave possui ações curtas de translação por frame, o que dá a impressão de que ela está movendo-se no espaço. Esta mesma ação foi aplicada à espaçonave que possui movimento de rotação.

4.4. Uma curva de Bézier

Esse requisito foi atendido, pois a Curva de Bézier foi implementada como um movimento do objeto, como pode ser observado nos trajetos em azul da espaçonave mais à esquerda da tela e da espaçonave mais à direita, que caminha do canto superior direito e vai até o canto inferior esquerdo.

Outra Curva de Bézier implementada foi do movimento do astronauta do Planeta Terra até a Lua. Esta curva, diferentemente das outras, não está explícita na tela, visto que se trata de um movimento significativo à funcionalidade do jogo. Dessa forma, o jogador pode determinar o tempo de chegada do astronauta da Terra à Lua, sem analisar detalhadamente o movimento que o astronauta faz. Além disso, para que o jogo indique tempos de chegada diferentes em cada jogada, o astronauta caminha na curva em velocidades diferentes. Isso é possível utilizando uma posição aleatória no vetor que determina a posição do objeto na curva, o que pode ser analisado com maior facilidade na figura abaixo:

```
spaceman.position.x = curveLineC.geometry.vertices[m].x;
spaceman.position.y = curveLineC.geometry.vertices[m].y;
spaceman.position.z = curveLineC.geometry.vertices[m].z;
k = k + random;
m = Math.floor(k);
```

Imagem 7: Posição do astronauta na curva por frame

Como observado na Imagem X, a velocidade do objeto na curva varia de acordo com o valor da variável `random` que pode receber valores reais entre 1 e 10. O cálculo deste valor é descrito a seguir:

```
random = (Math.random() * (9) + 1);
```

Imagem 8: Cálculo do número aleatório

Contudo, podemos comparar a posição do objeto `spaceman` com a posição do objeto `E 45 Aircraft_obj` em outra curva, que apresenta a mesma velocidade a cada rodada pois o incremento deste no vetor de vértices da Curva de Bézier é constante (`j++`) como descreve a Imagem X.

```
aircraft.position.x = curveLineB.geometry.vertices[j].x;
aircraft.position.y = curveLineB.geometry.vertices[j].y;
aircraft.position.z = curveLineB.geometry.vertices[j].z;
j++;
```

Imagem 9: Posição da espaçonave na curva por frame

4.5. Textura em algum modelo objeto

Esse requisito foi atendido. Foram utilizadas texturas em 4 objetos: nas espaçonaves `craft`, `SciFi_Fighter_AK5` e `E 45 Aircraft_obj` e no objeto do astronauta `spaceman`.

4.6. Textura em algum objeto simples

Esse requisito foi atendido. Foram adicionadas três texturas para formar o objeto Terra e uma textura para criar as nuvens da atmosfera terrestre, fornecendo um efeito mais realístico à Terra. Dessa forma, a Terra é composta por uma esfera com três imagens que simulam os oceanos e os continentes e outra esfera com uma imagem que simula as nuvens, como pode ser observado nas funções apresentadas abaixo:

```
function createEarth()
{
    // Create our Earth with texture
    earthGroup = new THREE.Object3D();

    var surfaceMap = THREE.ImageUtils.loadTexture( "../images/earth_surface_2048.jpg" );
    var normalMap = THREE.ImageUtils.loadTexture( "../images/earth_normal_2048.jpg" );
    var specularMap = THREE.ImageUtils.loadTexture( "../images/earth_specular_2048.jpg" );

    var geometry = new THREE.SphereGeometry(30, 32, 32);

    var material = new THREE.MeshPhongMaterial({
        map: surfaceMap,
        normalMap: normalMap,
        specularMap: specularMap});

    earth = new THREE.Mesh( geometry, material );

    earth.position.set(-200, 90, -100);
    //earth.scale.set(1.4, 1.4, 1.4);
    earthGroup.add(earth);
}

function createClouds()
{
    var cloudsMap = THREE.ImageUtils.loadTexture( "../images/earth_clouds_1024.png" );
    var cloudsMaterial = new THREE.MeshLambertMaterial({
        color: 0xffffff,
        map: cloudsMap,
        transparent:true
    });

    var cloudsGeometry = new THREE.SphereGeometry(30.5, 32, 32);
    cloudsMesh = new THREE.Mesh( cloudsGeometry, cloudsMaterial );
    cloudsMesh.position.set(-200, 90, -100);
    earthGroup.add(cloudsMesh);
}
```

Imagem 10: funções `createEarth()` e `createClouds()`

4.7. Shader próprio com cálculo de iluminação Phong

Esse requisito foi implementado, já que a tonalização Phong foi utilizada no objeto `Moon` criada com shader próprio, sendo o modelo de iluminação obtido a partir da soma entre a luz ambiente, reflexão difusa e reflexão especular.

```
var uniforms = ({
    timeDelta : {type: 'f', value: 0},
    emissive : {type: 'c', value: new THREE.Color(0x282828)},
    specular : {type: 'c', value: new THREE.Color(0xB0B0B0)}
});
var phongShader = THREE.ShaderLib.phong;
var mUniforms = THREE.UniformsUtils.merge([phongShader.uniforms, uniforms]);

var materialMoon = new THREE.ShaderMaterial( {
    uniforms : mUniforms,
    vertexShader: document.getElementById('vsMoon').textContent,
    fragmentShader: phongShader.fragmentShader,
    lights: true,
    side : THREE.DoubleSide
});
```

Imagem 11: Cálculo de Iluminação Phong na função `addMoon()`

4.8. Duas posições distintas de câmeras

Esse requisito também foi atendido. A primeira câmera está posicionada de forma a mostrar os objetos de frente, com o continente americano aparecendo na Terra, como pode ser observado na primeira imagem apresentada a seguir:

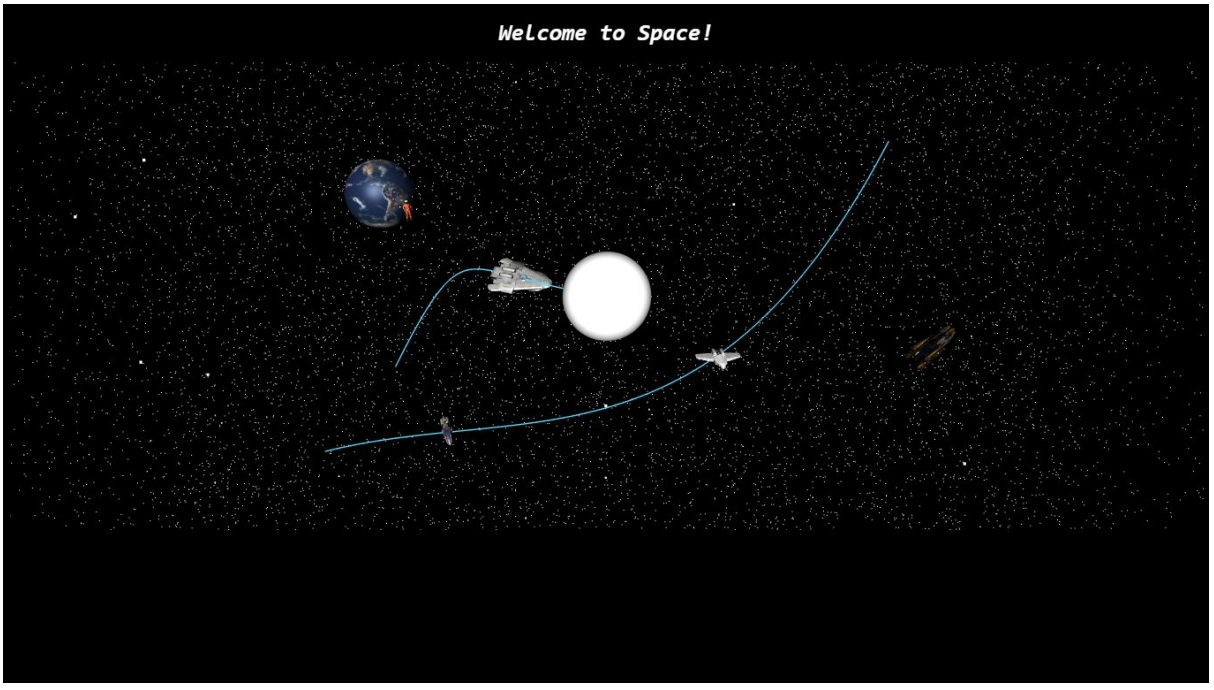

Imagem 12: Câmera na primeira posição

Já a segunda câmera está posicionada de forma que apresenta os objetos vistos de cima, permitindo uma visualização do Pólo Norte da Terra.

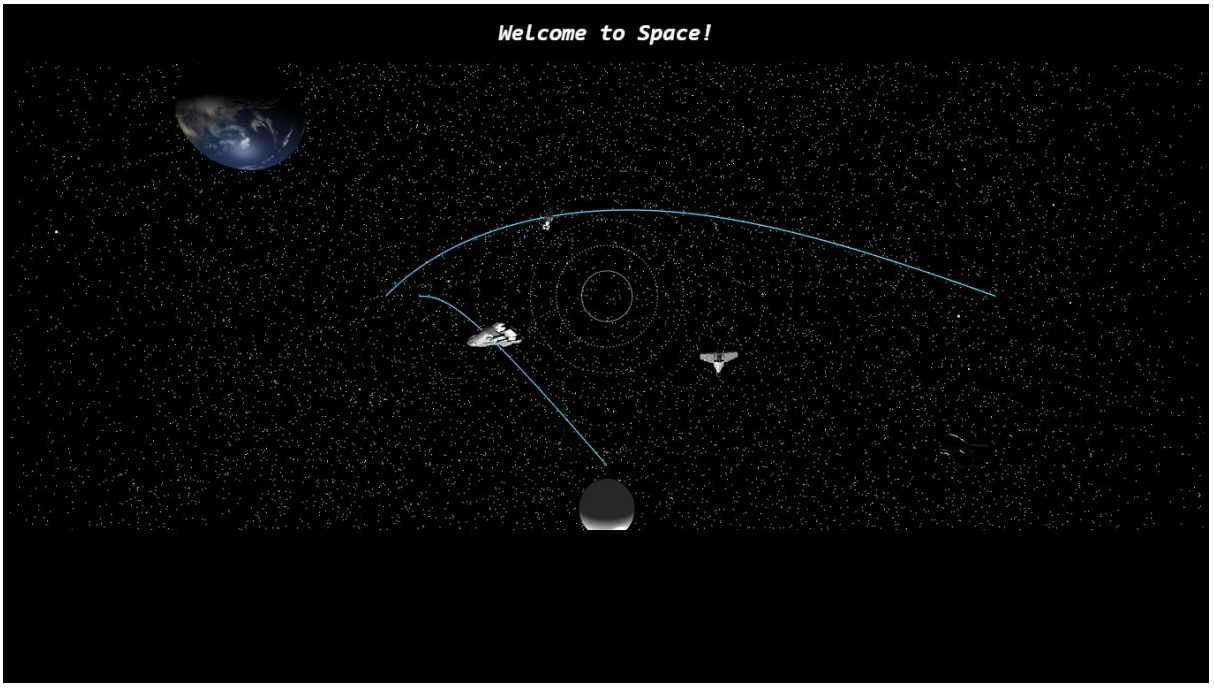

Imagem 13: Câmera na segunda posição

É possível observar a mudança da câmera pela posição das espaçonaves; da Lua, que fica parcialmente escura e, principalmente, do astronauta, que na primeira câmera estava de frente com todo o corpo aparecendo na frente da Terra e na segunda não consegue ser visto de cima. Optou-se por alternar a câmera por meio do clique do mouse em qualquer parte da tela.

4.9. Interações do usuário

Esse requisito foi atendido. Como mencionado na subseção anterior, o usuário pode alterar a câmera clicando no mouse, sendo esta uma das formas de interação com o usuário.

Além disso, o jogo permite que o usuário interaja ativamente com o projeto, visto que o jogo consiste em observar o trajeto do astronauta da Terra até a Lua e adivinhar o tempo decorrido, sendo solicitada uma entrada numérica que represente o tempo adivinhado pelo usuário.

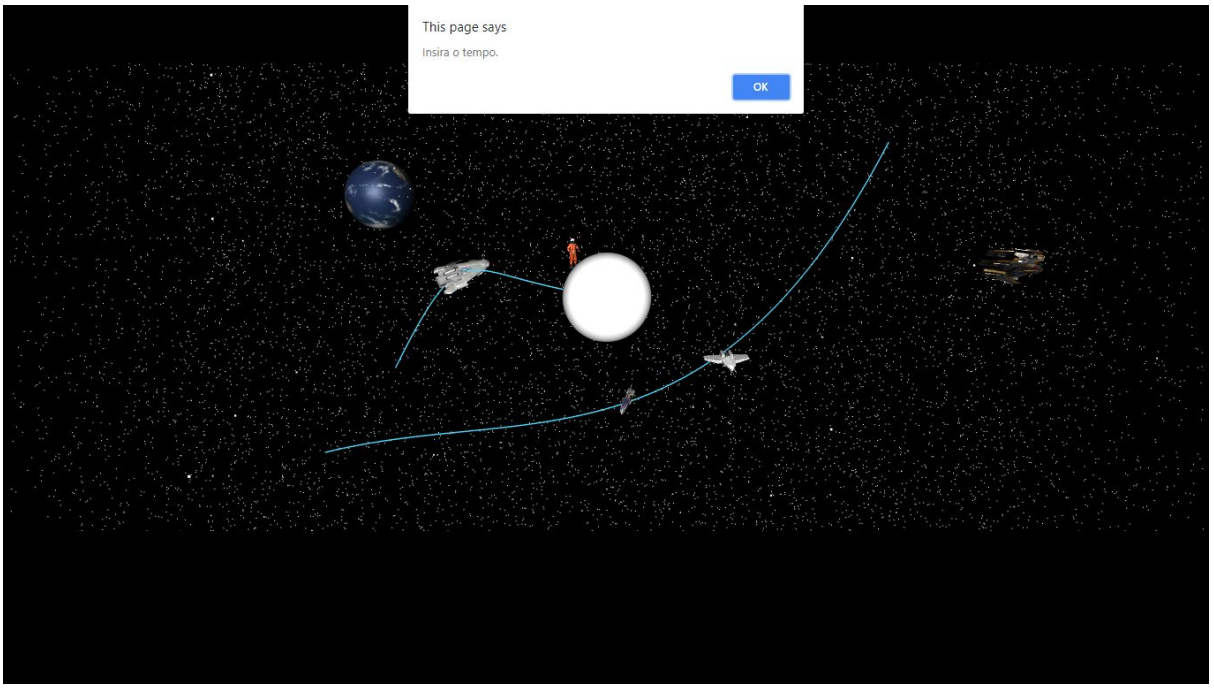

Imagem 14: Alerta pedindo ao usuário inserir um palpite do tempo do astronauta

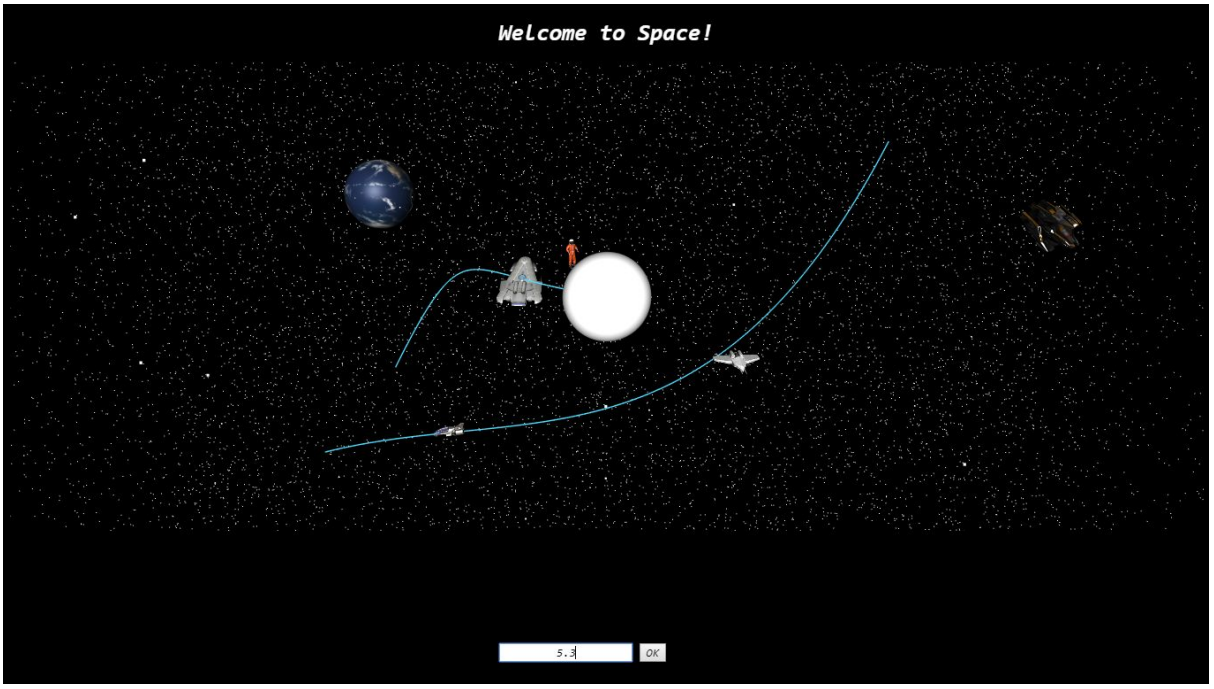

Imagem 15: Campo para inserção do palpite aparece no rodapé da página

Após o palpite do usuário sobre o tempo do movimento do astronauta, a interação continua: um alerta apresentará os resultados daquela rodada do jogo, mostrando a pontuação do usuário.

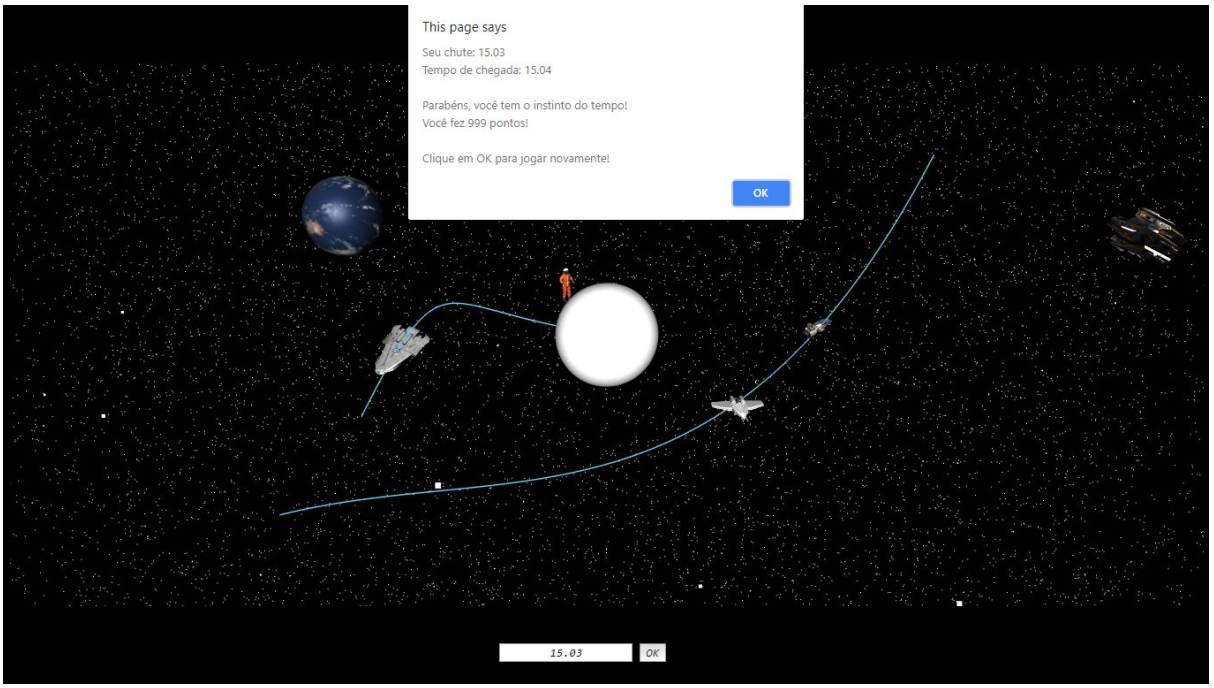

Imagem 16: Alerta para apresentar a pontuação do usuário

Essa pontuação é calculada utilizando-se uma regra de três, sendo proporcional à proximidade do palpite do usuário ao tempo que o movimento do astronauta realmente demorou. Os pontos estão no intervalo de 0 a 1000. Dessa forma, quanto mais próximo do tempo exato estiver o palpite do usuário, mais próxima a pontuação dele estará de 1000.

4.10. Um objeto articulado

Não foi possível atender este requisito, porém, pretende-se adicionar posteriormente, em atualizações futuras. Uma possível ideia para articular um objeto de forma que não fosse necessário adicionar nenhum objeto seria articular o astronauta. Então, articulando os braços do astronauta, ao chegar na Lua, ele poderia movimentar os braços, como se estivesse comemorando.

5. Requisitos Extras

Essa seção visa apresentar quais foram os requisitos extras implementados no projeto, além dos requisitos anteriormente listados.

5.1. Mais objetos carregados de arquivos

Como mencionado na subseção 4.1, foram carregados de arquivos três objetos a mais que a quantidade pedida nos requisitos básicos. Tem-se, portanto, um total de 5 objetos carregados de arquivos.

5.2. Mais objetos no total

Como mencionado na subseção 4.2, foram inseridos quatro objetos a mais do que o necessário, totalizando 9 objetos.

5.3. Mais movimentos distintos

Como mencionado na subseção 4.3, foram implementados mais movimentos do que a quantidade exigida de dois movimentos. No total temos três curvas de Bézier (em duas espaçonaves e no astronauta), cinco movimentos de rotação (no globo da Terra, nas nuvens da Terra e em três espaçonaves) e um movimento horizontal de um lado para o outro na última espaçonave.

5.4. Mais Curvas de Bézier

Como mencionado na subseção 4.4, foram implementadas duas Curvas de Bézier a mais do que o necessário. As Curvas de Bézier foram implementadas para os objetos `craft`, `E 45 Aircraft_obj` e `spaceman`.

5.5. Mais texturas em modelo objeto

Como mencionado na subseção 4.5, foram utilizadas mais texturas em mais modelos objeto. Os requisitos básicos exigiam uma textura em um modelo objeto, porém, foram utilizadas 4 texturas em 4 modelos objeto. As texturas foram adicionadas nos objetos `craft.obj`, `SciFi_Fighter_AK5.obj`, `E 45 Aircraft_obj.obj`, `spaceman.obj`.

5.6. Mais texturas em objeto simples

Como mencionado na subseção 4.6, foram utilizadas mais texturas em mais objetos simples. Os requisitos básicos exigiam uma textura em um objeto simples, porém foram utilizadas 4 texturas em dois objetos. Três texturas para a esfera do globo da Terra e uma textura para a esfera das nuvens da Terra, que em conjunto formam o planeta Terra.

5.7. Continuações Futuras

Como mencionado na subseção 4.10, não foi possível cumprir o requisito básico de articular um objeto. Portanto, pretende-se implementar esse requisito em uma atualização futura. Outras melhorias futuras poderiam ser a detecção de colisão,a utilização do `OrbitControl` para aumentar a interação com o usuário e permitir que ele pudesse ver os objetos de vários referenciais, a utilização de um `skybox` para fazer a representação do espaço com estrelas e nebulosas ou ainda a animação dos pontos que representam as estrelas, ou seja, fazer com que a câmera mudasse de acordo com o movimento do mouse.

Ao longo da discussão e brainstorming para a elaboração do projeto foram discutidas várias ideias promissoras. Uma delas foi permitir que o jogo seja multiplayer, com dois jogadores, recebendo um palpite de cada um dele. Dessa forma o jogador que fizesse

um palpite mais próximo do tempo real decorrido ganharia a rodada. Caso ambos os palpites fossem muito distantes do tempo real, ou seja, estivessem fora da margem de acerto, haveria um empate. Essa seria outra funcionalidade a ser implementada futuramente.

Outra ideia interessante seria permitir uma competição ou campeonato entre os jogadores por meio de uma tabela na tela inicial que mostrasse os recordes batidos, isto é, os nomes dos usuários juntamente com suas respectivas pontuações. Para isso, seria necessária a utilização de um banco de dados para guardar tais dados, outra funcionalidade para continuações futuras.

Vale ressaltar que tentou-se implementar a maioria dessas funcionalidades, porém, sem êxito. Todavia, o grupo conseguiu implementar outras facilidades e implementar os conceitos estudados em aula, chegando a um resultado satisfatório e que cumpre com o proposto.