

ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO

PSI3502 – Realidade Virtual

Tarefa 4

GRUPO: APLICAÇÕES PARA O ENSINO

ALEXANDRE HOPPE INOUE	8988942
BRUNO HARLLEN	9871051
LUCAS SALLABERRY	8038514
LUCAS SPONCHIADO	9373924
RAUL DA SILVA SOUZA	8630540
VITOR AUGUSTO MARTIN	8993100

São Paulo, 05/11/2018

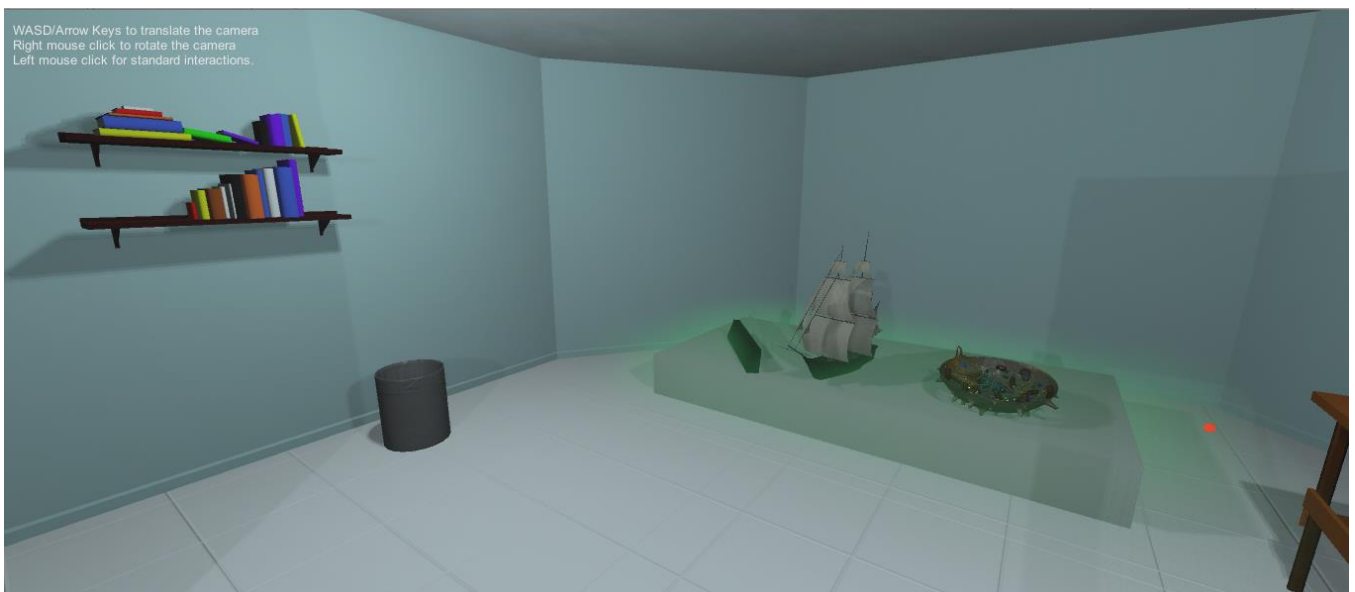
Cenas

1) Sala Inicial

Esboço 3D:

O aluno é colocado em uma sala simples com uma mesa e alguns objetos de ambiente. O aluno é introduzido a uma rápida explicação de como a cena funciona (provavelmente por uma narrativa) e se dirige à uma plataforma onde encontra os ícones das cenas (modelos 3D representativos). Para facilitar a orientação do aluno na sala, uma luz volumétrica suave foi colocada sobre a plataforma.

Sobre a plataforma estão colocados os modelos de cada um dos desafios. Ao se aproximar, esses ícones se levantam da mesa, flutuando à frente do aluno. Ao colocar a mão sobre um dos desafios, ele começa a girar no próprio eixo, pulsando (aumentando e diminuindo de escala) para indicar que está sendo selecionado. Caso o aluno decida que quer fazer este desafio, ele deve segurar o ícone do desafio e o colocar sobre a mesa ao lado.



Sala Inicial com modelos que representam cada um dos desafios.

Prefabs:

A cena conta com um prefab dos ícones. Esses prefabs contém a lógica de animação dos ícones, sendo apenas necessário mudar o modelo contido em cada um.

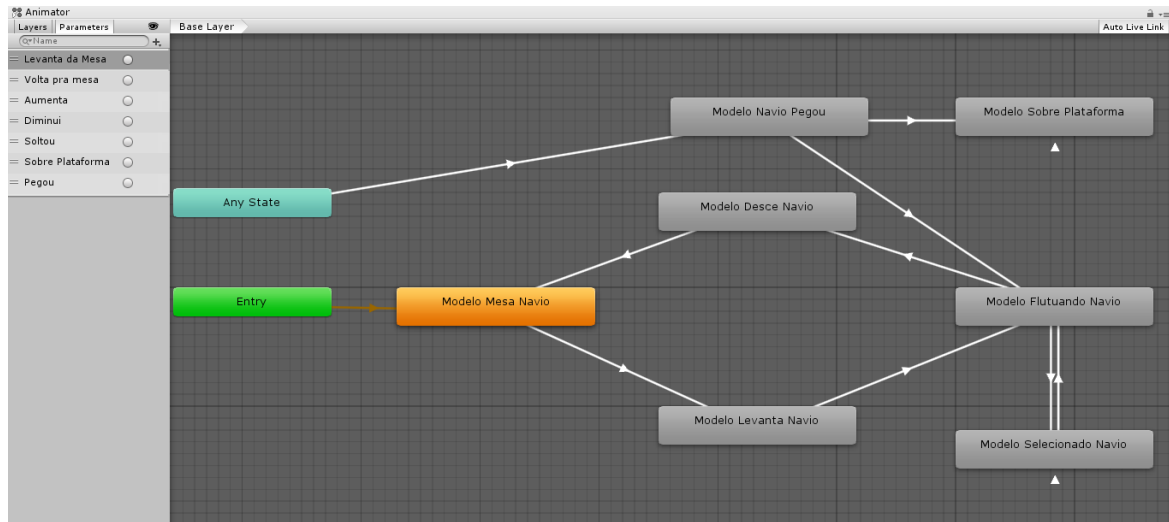
Será também utilizado prefab de luzes volumétricas do plugin Aura - Volumetric Lighting para criar as luzes volumétricas e alguns prefabs prontos do asset Morgue Room PBR para modelar a sala.

Métodos de locomoção:

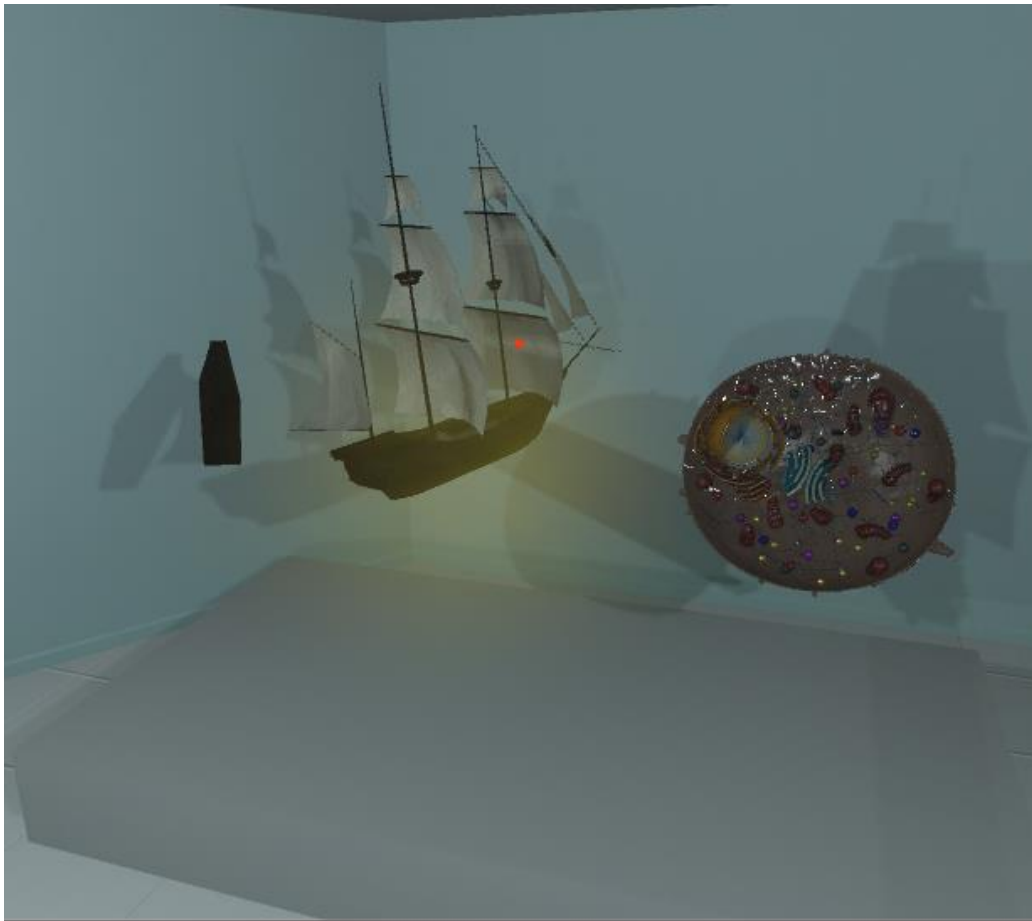
A Locomoção nesta cena será feita dentro da área 4x4m prevista, visando locomoção natural.

Interações:

As interações feitas com os ícones seguirão a lógica de animações:



Inicialmente, os modelos estarão sobre a plataforma (estado “Modelo Mesa”). Quando o aluno se aproxima e entra em uma área definida por um collider, o modelo passa a flutuar (“Modelo Levanta” → “Modelo Flutuando”). Quando a mão do aluno é colocada sobre o modelo, a função “On Hand Hover” faz com que o modelo passe a girar no lugar, emitindo luz e pulsando (“Modelo Seleccionado”).



Modelo selecionado pelo aluno girando e emitindo luz.

A qualquer momento o aluno pode segurar um dos modelos. Caso o faça, ele é colocado na mão do aluno (“Modelo Pegou”) e, caso seja solto, irá voltar ao local original. Porém, se o aluno soltar o modelo sobre uma mesa que se encontrará ao seu lado, o modelo será colocado sobre uma plataforma que está na mesa e irá rotacionar emitindo uma luz. Isso significa que este modelo foi selecionado e, caso o aluno aperte o botão “Iniciar” sobre a mesa, o desafio escolhido será iniciado.



Modelo escolhido colocado sobre a mesa. Ainda bem simples, o modelo da mesa deve ser alterado, além de se colocar uma plataforma sob o ícone selecionado e um botão “Iniciar”.

As interações são controladas pelos scripts:

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.Video;
5 using AuraAPI;
6
7 //Script que ativa as animações dos modelos
8 public class AtivarAnimacoesIcones : MonoBehaviour {
9
10     //Agrega todos os Icones na cena
11     private Animator[] animators;
12
13     //OlharParaJogador é o script que faz o Icone olhar sempre para o jogador
14     //Esta função não é mais usada para ícones 3D
15     private OlharParaJogador[] ativarOlhar;
16
17     private GameObject player;
18
19     //auraCubo é a luz volumetrica que fica sobre ícones no início da cena
20     private AuraVolume auraCubo = null;
21     public float fadeSpeed = 50f;
22
23     //sobreAMesa define se ha algum modelo sobre a mesa
24     private GameObject sobreAMesa = null;
```

```

25
26 void Start(){
27
28     GameObject[] Icones = GameObject.FindGameObjectsWithTag ("Icone");
29     animators = new Animator[Icones.Length];
30
31
32     ativarOlhar = new OlharParaJogador[Icones.Length];
33
34     for(int a = 0; a < Icones.Length; a++){
35         animators [a] = Icones [a].GetComponent<Animator> ();
36         ativarOlhar [a] = Icones [a].transform.GetChild(0).gameObject.GetComponent<OlharParaJogador>
37     }
38
39     if(transform.GetChild(0) != null){
40         auraCubo = transform.GetChild (0).gameObject.GetComponent<AuraVolume>();
41     }
42 }
43
44 //jogador entrou na área de ativação
45 //Modelos levantam da mesa
46 //Luz volumetrica apaga
47 void OnTriggerEnter(Collider other)
48 {
49     if(other.gameObject.tag == "Player")
50     {
51         foreach (Animator anim in animators) {
52             anim.SetTrigger ("Levanta da Mesa");
53         }
54         foreach (OlharParaJogador component in ativarOlhar) {
55             component.ativar = true;
56         }
57
58         StartCoroutine(FadeLuz(-1f,auraCubo));
59     }
60 }
61
62 //jogador saiu da área de ativação
63 //Modelos voltam pra mesa
64 //Luz volumetrica apaga
65 void OnTriggerExit(Collider other)
66 {
67     if(other.gameObject.tag == "Player")
68     {
69         foreach (Animator anim in animators) {
70             anim.SetTrigger ("Volta pra mesa");
71         }
72         foreach (OlharParaJogador component in ativarOlhar) {
73             component.ativar = false;
74         }
75         StartCoroutine(FadeLuz(1f,auraCubo));
76     }
77 }
78
79 //caso jogador colocar a mão sobre ícone
80 public void AumentaIcone(GameObject modelo){
81     if (modelo != sobreAMesa) {
82         modelo.GetComponent<Animator> ().SetTrigger ("Aumenta");
83         IniciaVideo (modelo);
84         LigaAura (modelo);
85     }
86 }
87

```

```

88 //caso jogador tire a mão do ícone
89 public void DiminuiIcone(GameObject modelo){
90     if (modelo != sobreAMesa) {
91         modelo.GetComponent<Animator> ().SetTrigger ("Diminui");
92         ParaVideo (modelo);
93         DesligaAura (modelo);
94     }
95 }
96
97 //Chamado por "On Hand Hover Begin"
98 public void Pegou(GameObject modelo){
99     modelo.GetComponent<Animator>().SetTrigger ("Pegou");
100     DesligaAura (modelo);
101 }
102
103 public void IniciaVideo(GameObject objeto){
104     Transform player = objeto.transform.GetChild (0).GetChild (0);
105     if (player.GetComponent<VideoPlayer> () != null) {
106         player.GetComponent<VideoPlayer> ().Play ();
107     }
108 }
109
110 public void ParaVideo(GameObject objeto){
111     Transform player = objeto.transform.GetChild (0).GetChild (0);
112     if (player.GetComponent<VideoPlayer> () != null) {
113         player.GetComponent<VideoPlayer> ().Stop ();
114     }
115 }
116
117 //Ativa a Luz no modelo selecionado
118 public void LigaAura(GameObject objetoAura){
119     GameObject aura = objetoAura.transform.GetChild (1).gameObject;
120     if (aura.name == "Aura Sphere Volume") {
121         aura.SetActive (true);
122     }
123 }
124
125 //Desativa a luz no modelo selecionado
126 public void DesligaAura(GameObject objetoAura){
127     GameObject aura = objetoAura.transform.GetChild (1).gameObject;
128     if (aura.name == "Aura Sphere Volume") {
129         aura.SetActive (false);
130     }
131 }
132
133
134 public void SetSobreAMesa(GameObject objetoSobreMesa){
135     sobreAMesa = objetoSobreMesa;
136 }
137
138
139 //liga/desliga a luz do auraCubo de maneira suave
140 //para intensidadeAlvo --> -1 desliga e 1 liga
141 public IEnumerator Fadeluz(float intensidadeAlvo, AuraVolume aura){
142     if (aura != null) {
143         if (intensidadeAlvo > aura.density.injectionParameters.strength) {
144             while (aura.density.injectionParameters.strength < intensidadeAlvo) {
145                 aura.density.injectionParameters.strength += 0.05f * fadeSpeed * Time.deltaTime;
146                 yield return null;
147             }
148         } else {
149             while (aura.density.injectionParameters.strength > intensidadeAlvo) {
150                 aura.density.injectionParameters.strength -= 0.05f * fadeSpeed * Time.deltaTime;
151                 yield return null;
152             }
153         }
154     }
155 }
156
157 }
158

```

E também pelo Script:

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 //Script que realiza as interações dos modelos com a mesa que muda de cena
6 public class MudarACena : MonoBehaviour {
7
8     //Agrega todos os Icones na cena
9     private GameObject icones;
10
11     private GameObject desafioEscolhido = null;
12
13     //Script ativo AtivarAnimacoesIcones
14     private AtivarAnimacoesIcones ativarAnimacoesScript;
15
16     void Start () {
17         icones = GameObject.Find ("Icones");
18         ativarAnimacoesScript = GameObject.Find ("Controlador Icones").GetComponent<AtivarAnimacoesIcones>();
19     }
20
21
22     //Reconhece se algum modelo foi colocado sobre a mesa
23     void OnTriggerEnter(Collider other)
24     {
25         if (other.tag == "Icone" && desafioEscolhido == null) {
26             desafioEscolhido = other.gameObject;
27             ativarAnimacoesScript.SetSobreAMesa(desafioEscolhido);
28         }
29     }
30
31     //Reconhece se algum modelo foi tirado da mesa
32     void OnTriggerExit(Collider other)
33     {
34         if (other.gameObject == desafioEscolhido) {
35             desafioEscolhido = null;
36             ativarAnimacoesScript.SetSobreAMesa(desafioEscolhido);
37         }
38     }
39
40     //Chamado no "On Hand Hover End"
41     public void ColocarIconeNaPlataforma (GameObject modelo){
42         if (desafioEscolhido == modelo) {
43             desafioEscolhido.transform.parent = transform;
44             desafioEscolhido.GetComponent<Animator> ().SetTrigger ("Sobre Plataforma");
45
46             ativarAnimacoesScript.LigaAura (modelo);
47
48         } else {
49             modelo.GetComponent<Animator>().SetTrigger ("Soltou");
50             modelo.transform.parent = icones.transform;
51
52             ativarAnimacoesScript.DesligaAura (modelo);
53         }
54     }
55 }
56 }
57
```

2) Desafio Tiro de canhão

Para evitar ter que sair do ambiente virtual para realizar contas e resolver o desafio, resolvemos mudar o desafio do foguete para o desafio proposto inicialmente dos canhões no

barco. Só que em vez do aluno ter que calcular as inclinações do canhão, serão abordados conceitos como:

- Maior distância horizontal atingida a 45 graus.
- Um navio que pode ser atingido com um canhão a 30 graus pode ser atingido a 60 graus.

A proposta do desafio mudou um pouco. O objetivo será derrubar os 3 navios inimigos sem acertar as gaivotas presentes no cenário. Um estará bem longe e precisará angular o canhão em 45 graus. O segundo navio só poderá ser atingido atirando a 30 graus (atirando a 60 acertaria um conjunto de gaivotas) e o outro atirando a 60 graus (atirando a 30 acertaria outro conjunto de gaivotas)



Esboço 3D:

A cena inicia com o aluno sendo inserido no seu navio. Ele vai até o canhão destacado para atirar nos navios inimigos. Ao interagir com o canhão ele receberá *feedback* de quão inclinado o canhão está e poderá inclinar mais ou menos com os seus controles da Oculus e finalmente atirar para tentar acertar os navios. Além disso, demais dados relevantes do tiro podem ser apresentados ao aluno.



Prefabs:

Por enquanto, a cena possui os seguintes prefabs:

- Do canhão e bola de canhão
- Navios inimigos mais leves e pouco detalhados
- Navio em que o player está embarcado
- Elementos do ambiente como mar, e nuvens.

Ainda vão ser necessários prefabs para a fumaça do tiro de canhão e das gaivotas.

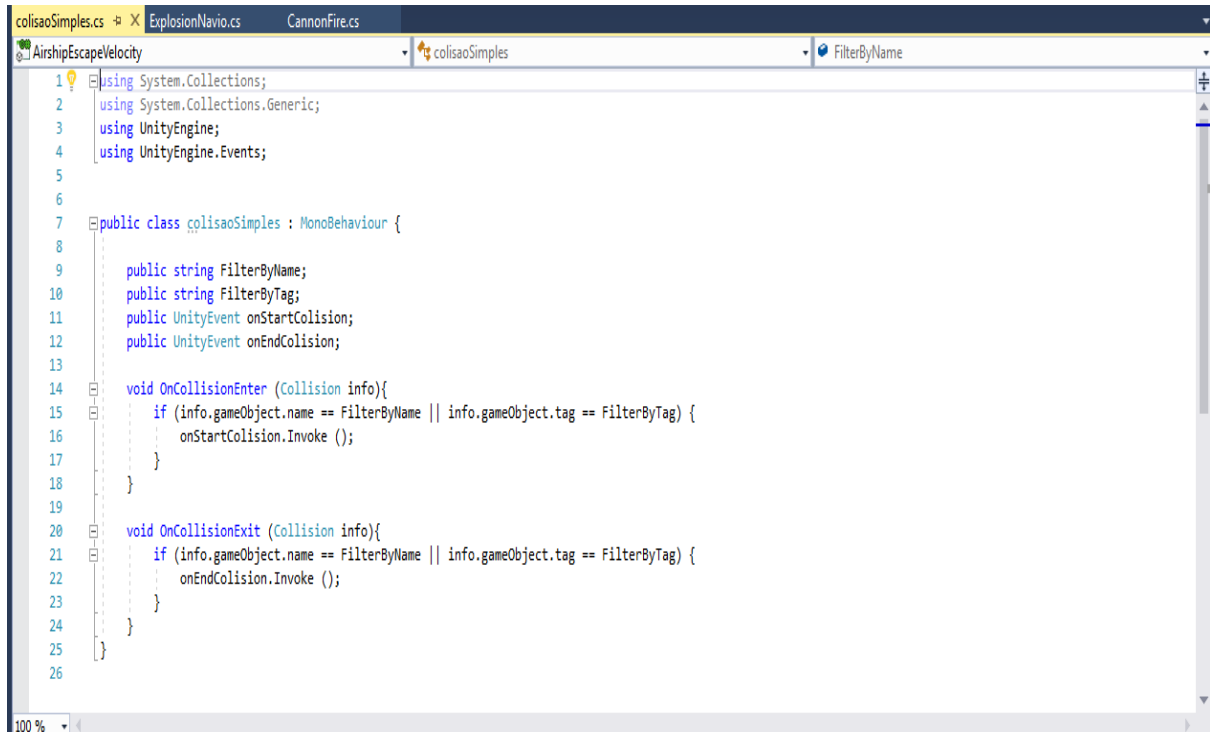
Métodos de locomoção:

A locomoção será com a utilização do Oculus Rift usando a área 4x4, uma vez que o espaço para se explorar no navio é razoavelmente pequeno.

Interações:

O usuário pode interagir com os canhões do navio e serão necessários scripts para inclinar o canhão, atirar o projétil e identificar quando os alvos são atingidos e etc. Seguem alguns dos scripts que foram criados.

Script para identificar colisão da bola de canhão com o navio inimigo:



```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.Events;
5
6
7 public class colisaoSimples : MonoBehaviour {
8
9     public string FilterByName;
10    public string FilterByTag;
11    public UnityEvent onStartCollision;
12    public UnityEvent onEndCollision;
13
14    void OnCollisionEnter (Collision info){
15        if (info.gameObject.name == FilterByName || info.gameObject.tag == FilterByTag) {
16            onStartCollision.Invoke ();
17        }
18    }
19
20    void OnCollisionExit (Collision info){
21        if (info.gameObject.name == FilterByName || info.gameObject.tag == FilterByTag) {
22            onEndCollision.Invoke ();
23        }
24    }
25 }
26
```

Script feito para realizar o tiro de canhão. Classe com objetos relevantes como o canhão, a bola de canhão, posição de instanciação da bola, posição de instanciação de uma explosão e função de tiro:

```

public class CannonFire : MonoBehaviour {

    public GameObject bolaDeCanhao;
    private Rigidbody bolaDeCanhaoRigid;
    public Rigidbody canhao;
    public Transform shotPos;
    public Transform exploPos;
    public GameObject explosao;
    public float power;

    // Use this for initialization
    void Start () {

    }
    // Update is called once per frame
    void Update () {
        if (Input.GetKeyUp("space"))
            fire();
    }

    public void fire() {
        shotPos.rotation = transform.rotation;
        exploPos.rotation = transform.rotation;
        GameObject cannonBallCopy = Instantiate(bolaDeCanhao, shotPos.position, shotPos.rotation) as GameObject;
        bolaDeCanhaoRigid = cannonBallCopy.GetComponent<Rigidbody>();
        bolaDeCanhaoRigid.AddForce(transform.up* power, ForceMode.Impulse);
        Instantiate(explosao, exploPos.position, exploPos.rotation);
    }
}

```

Script com função para instanciar explosão no navio após contato com a bola de canhão.

```

public class ExplosionNavio : MonoBehaviour {

    public GameObject explosaoNavio;

    // Use this for initialization
    void Start() {

    }

    // Update is called once per frame
    void Update() {

    }

    public void explodeNavio(){
        Instantiate(explosaoNavio, transform.position, transform.rotation);
    }
}

```

Para realizar a inclinação do canhão, foi usado o script “circular drive” apresentado em aula.

3) Construção de Barragem

Esboço 3D:

A cena inicia com o aluno sendo inserido em uma sala onde ele encontrará uma mesa com “maquete” de um sistema de hidrelétrica com todos os elementos que a compõe, desde o lago e a natureza em volta até o vertedouro; e uma mesa com diferentes tipos de barragens, geradores, etc. Ele deverá interagir com os itens da mesa e selecionar os mais adequados para completar a maquete. Ao fazer esta interação, ele receberá *feedback* de quanto a barragem

pode suportar, quanta de energia gerar, etc de forma a escolher os componentes certos para o seu objetivo.

Prefabs:

- Elementos do ambiente e da natureza para a maquete
- Elementos da barragem
- Para a sala

Métodos de locomoção:

A locomoção será com a utilização do Oculus Rift usando a área 4x4, uma vez que o espaço para se explorar a sala é razoavelmente pequeno.

Interações:

O usuário pode interagir com os diferentes elementos que compõem uma hidrelétrica para pegá-los e colocá-los na maquete. Serão necessários scripts para encaixar as peças soltas na maquete, para a animação da maquete e para a reação da maquete de acordo com as diferentes opções de montagem.

