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

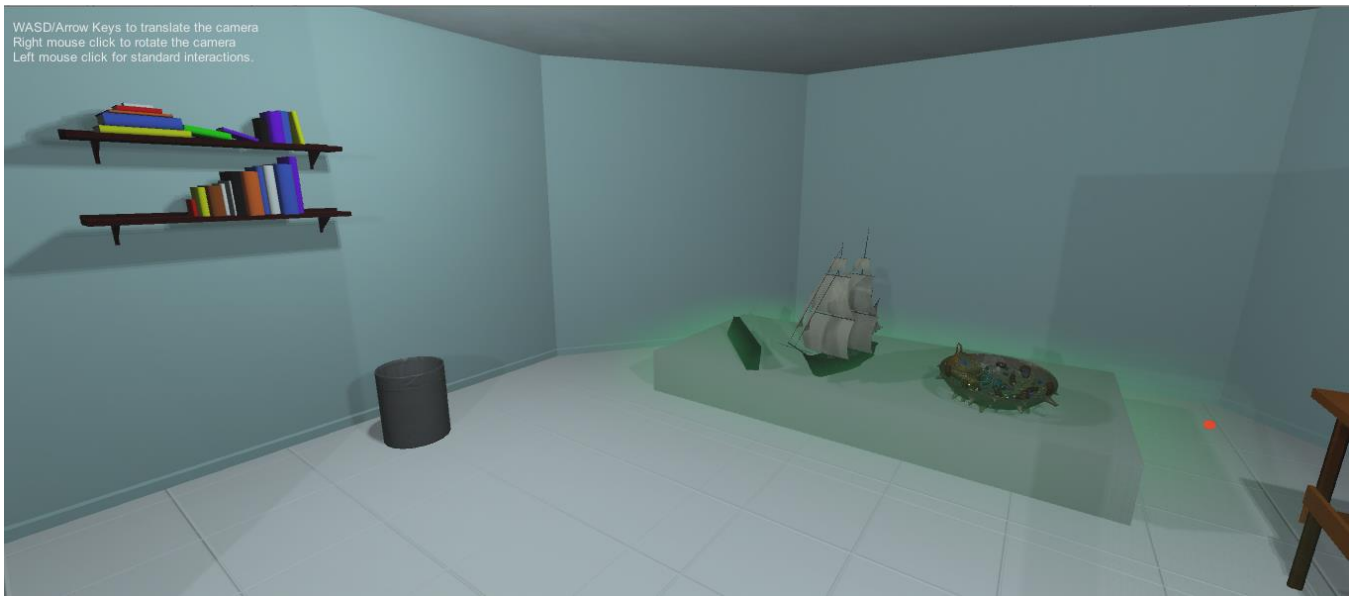
Cenas

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.

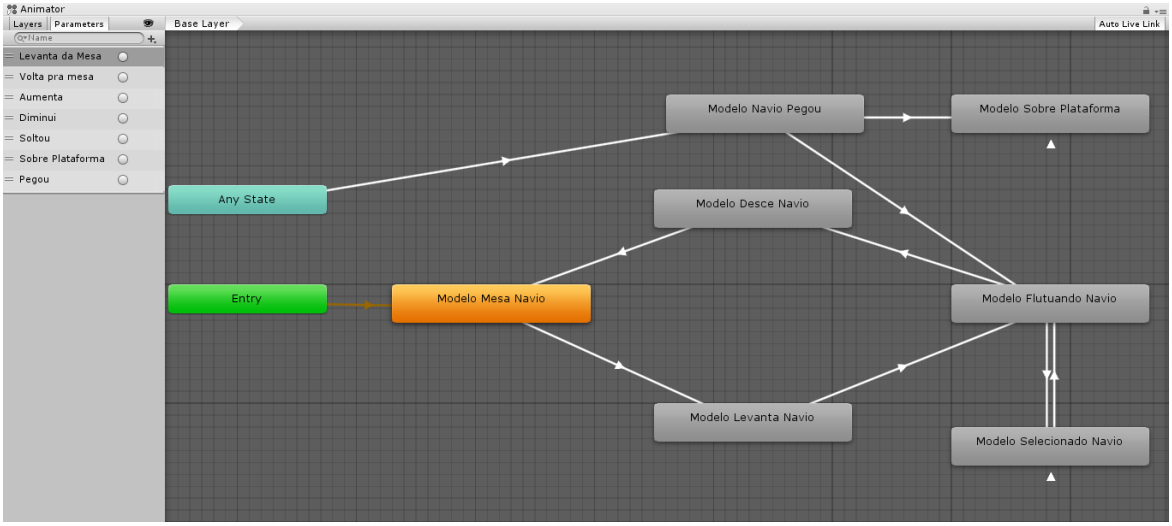
Será também utilizado prefab de luzes volumétricas do plugin Aura - Volumetric Lighting para criar as luzes volumétricas e alguns prefabs prontos do asset Morgue Room PBR para modelar a sala.

Métodos de locomoção:

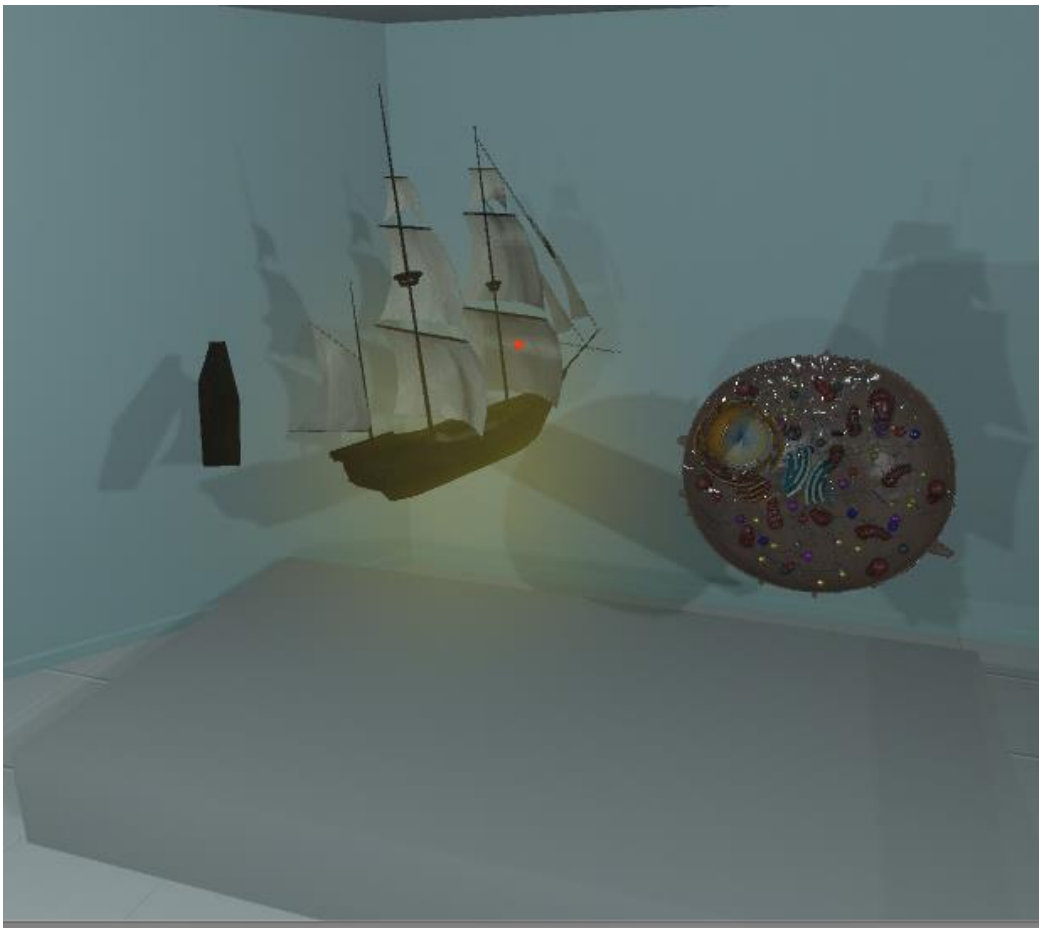
A Locomoção nesta cena será feita dentro da área 4x4m prevista, visando locomoção natural.

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 Selecionado”).



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:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Video;
using AuraAPI;

    //Script que ativa as animações dos modelos
public class AtivarAnimacoesIcones : MonoBehaviour {

    //Agrega todos os Icones na cena
    private Animator[] animators;

    //OlharParaJogador é o script que faz o Icone olhar sempre para o jogador
    //Esta função não é mais usada para ícones 3D
    private OlharParaJogador[] ativarOlhar;

    private GameObject player;

    //auraCubo é a luz volumetrica que fica sobre icones no inicio da cena
    private AuraVolume auraCubo = null;
    public float fadeSpeed = 50f;

    //sobreAMesa define se ha algum modelo sobre a mesa
    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
```

```
    //caso jogador tire a mão do ícone
    public void DiminuiIcone(GameObject modelo){
        if (modelo != sobreAMesa) {
            modelo.GetComponent<Animator> ().SetTrigger ("Diminui");
            ParaVideo (modelo);
            DesligaAura (modelo);
        }
    }

    //Chamado por "On Hand Hover Begin"
    public void Pegou(GameObject modelo){
        modelo.GetComponent<Animator>().SetTrigger ("Pegou");
        DesligaAura (modelo);
    }

    public void IniciaVideo(GameObject objeto){
        Transform player = objeto.transform.GetChild (0).GetChild (0);
        if (player.GetComponent<VideoPlayer> () != null) {
            player.GetComponent<VideoPlayer> ().Play ();
        }
    }

    public void ParaVideo(GameObject objeto){
        Transform player = objeto.transform.GetChild (0).GetChild (0);
        if (player.GetComponent<VideoPlayer> () != null) {
            player.GetComponent<VideoPlayer> ().Stop ();
        }
    }

    //Ativa a luz no modelo selecionado
    public void LigaAura(GameObject objetoAura){
        GameObject aura = objetoAura.transform.GetChild (1).gameObject;
        if (aura.name == "Aura Sphere Volume") {
            aura.SetActive (true);
        }
    }

    //Desativa a luz no modelo selecionado
    public void DesligaAura(GameObject objetoAura){
        GameObject aura = objetoAura.transform.GetChild (1).gameObject;
        if (aura.name == "Aura Sphere Volume") {
            aura.SetActive (false);
        }

    }

    public void SetSobreAMesa(GameObject objetoSobreMesa){
        sobreAMesa = objetoSobreMesa;

    }

    //liga/desliga a luz do auraCubo de maneira suave
    //para intensidadeAlvo --> -1 desliga e 1 liga
    public IEnumerator FadeLuz(float intensidadeAlvo, AuraVolume aura){
        if (aura != null) {
            if (intensidadeAlvo > aura.density.injectionParameters.strength) {
                while (aura.density.injectionParameters.strength < intensidadeAlvo) {
                    aura.density.injectionParameters.strength += 0.05f * fadeSpeed * Time.deltaTime;
                    yield return null;
                }
            } else {
                while (aura.density.injectionParameters.strength > intensidadeAlvo) {
                    aura.density.injectionParameters.strength -= 0.05f * fadeSpeed * Time.deltaTime;
                    yield return null;
                }
            }
        }
    }

}
```

E também pelo Script:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

//Script que realiza as interações dos modelos com a mesa que muda de cena
public class MudarACena : MonoBehaviour {

    //Agrega todos os Icones na cena
    private GameObject icones;

    private GameObject desafioEscolhido = null;

    //Script ativo AtivarAnimacoesIcones
    private AtivarAnimacoesIcones ativarAnimacoesScript;

    void Start () {
        icones = GameObject.Find ("Icones");
        ativarAnimacoesScript = GameObject.Find ("Controlador Icones").GetComponent<AtivarAnimacoesIcones>();
    }


    //Reconhece se algum modelo foi colocado sobre a mesa
    void OnTriggerEnter(Collider other)
    {
        if (other.tag == "Icone" && desafioEscolhido == null) {
            desafioEscolhido = other.gameObject;
            ativarAnimacoesScript.SetSobreAMesa(desafioEscolhido);
        }
    }

    //Reconhece se algum modelo foi tirado da mesa
    void OnTriggerExit(Collider other)
    {
        if (other.gameObject == desafioEscolhido) {
            desafioEscolhido = null;
            ativarAnimacoesScript.SetSobreAMesa(desafioEscolhido);
        }
    }

    //Chamado no "On Hand Hover End"
    public void ColocarIconeNaPlataforma (GameObject modelo){
        if (desafioEscolhido == modelo) {
            desafioEscolhido.transform.parent = transform;
            desafioEscolhido.GetComponent<Animator> ().SetTrigger ("Sobre Plataforma");

            ativarAnimacoesScript.LigaAura (modelo);

        } else {
            modelo.GetComponent<Animator>().SetTrigger ("Soltou");
            modelo.transform.parent = icones.transform;

            ativarAnimacoesScript.DesligaAura (modelo);
        }

    }
}

```

2) Desafio Tiro de canhão (em negrito trechos respondendo ao feedback)

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:

- Conservação de energia. Adicionar vetor vertical que diminui conforme a cinética diminui ao passo que a bola de canhão ganha altura e energia potencial e aumenta conforme ela retorna para a forma cinética.
- 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. Considera-se ainda uma possível narrativa para servir como pretexto para atirar nos navios.

Esboço 3D:

A cena inicia com o aluno sendo inserido no seu navio. Haverá uma narração que explicará os comandos básicos ao desafiante bem como o contexto da cena. O aluno terá apenas um número limitado de tiros e um tempo para acertar os oponentes antes que eles consigam fugir para fora do alcance dos canhões. Para induzir o aprendizado, o usuário poderá assistir replays dos seus tiros realizados até então para planejar o próximo disparo. Na análise terá ao seu dispor informações de força do tiro, trajetória da bala bem como a ação da gravidade no tiro, que diminuirá o vetor vertical da velocidade enquanto essa sobe e aumentará o vetor quando ela começar a descer. A ideia até agora, para poder abordar o conceitos de que existem sempre 2 ângulos que conseguem percorrer uma mesma distância horizontal, é que vão ter mais de um canhão no barco. Alguns dos canhões só vão poder inclinar em ângulos de 45º até 90º enquanto os outros poderão inclinar de 0º até 45º. Após o aluno acertar um navio que está a uma distância x, ele terá que acertar em outro canhão com apenas um tiro em outro navio também a uma distância x.

Para aumentar a interatividade, além de adicionar mais canhões para o aluno percorrer pelo navio, ele deverá carregar os canhões com pólvora antes de cada disparo. Quanto mais pólvora ele fornecer ao canhão, maior será a força. Ao interagir com o canhão depois de carrega com a pólvora, ele poderá inclinar a arma de fogo com os seus controles da Oculus e finalmente atirar para tentar acertar os navios.

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:

colisaoSimples.cs | ExplosionNavio.cs | CannonFire.cs

AirshipEscapeVelocity | colisaoSimples | FilterByName

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Events;


public class colisaoSimples : MonoBehaviour {

    public string FilterByName;
    public string FilterByTag;
    public UnityEvent onStartColision;
    public UnityEvent onEndColision;

    void OnCollisionEnter (Collision info){
        if (info.gameObject.name == FilterByName || info.gameObject.tag == FilterByTag) {
            onStartColision.Invoke ();
        }
    }

    void OnCollisionExit (Collision info){
        if (info.gameObject.name == FilterByName || info.gameObject.tag == FilterByTag) {
            onEndColision.Invoke ();
        }
    }
}

```

100 %

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 e 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:

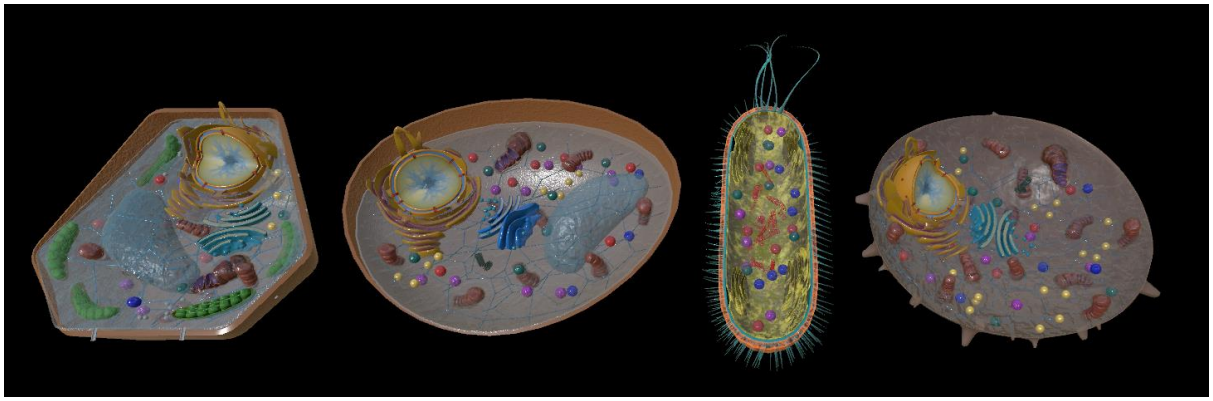
A locomoção será com a utilização do Oculus Rift usando a área 4x4, uma vez que o espaço para se explorar a sala é razoavelmente pequeno.

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.

4) Células Biológicas

Esta cena ainda está na etapa inicial do seu desenvolvimento. Para esta, serão utilizados modelos do Asset "BiologyCellsPack", que fornece os prefabs de células de fungos, animais, vegetais e bactérias, todas completas com suas respectivas organelas.



Modelos das células de vegetais, fungos, bactérias e animais, respectivamente, presentes no Asset "BiologyCellsPack"

O objetivo deste desafio é possibilitar que o aluno visualize, de forma tridimensional, imersiva e interativa, cada um dos tipos de células, as quais ele geralmente apenas tem contato através de desenhos e ilustrações. Não apenas será possível observar e reconhecer as organelas, mas pretende-se ensinar ao aluno a função de algumas delas na dinâmica celular. Para isto, ele realizará o papel de agente facilitador de processos celulares, auxiliando as organelas a cumprirem suas funções. As funções da célula que poderão ser realizadas pelo aluno ainda estão sendo estudadas.

Espera-se, também, que o aluno saiba diferenciar ao menos entre as células de animal, bactéria e vegetal, sendo pedido que diferencie entre cada uma delas com base no formato e nas organelas presentes.

A locomoção do aluno será realizada no espaço 4x4m disponível.