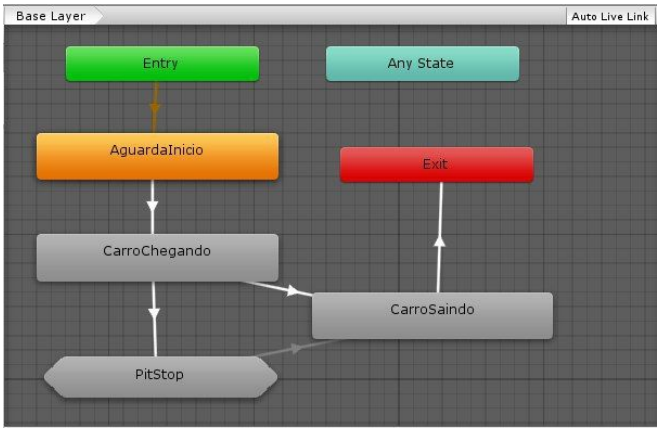
Tema: Aplicações para esporte e saúde

Projeto: Stock Car Pit Stop Experience

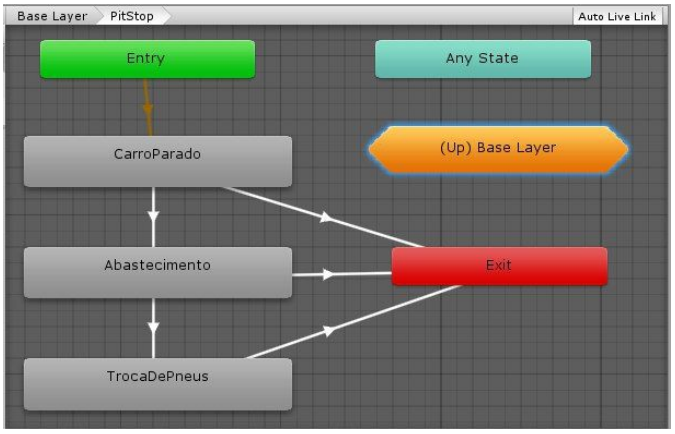
Membros	**NUSP**
Antonio Crespo	7380071
Bruno Akio Shirasuna	9778800
Bruno da Costa Braga	8993480
Gustavo Kimura	10334165
Henrique Uhelszki Yoshida	10314854
Thiago Perroni Meletti	8992482
Vinicius Henriques de Freitas	8992012

1. Dinâmica da Aplicação

Diante das escolhas feitas durante o planejamento das últimas aulas e nas últimas tarefas, a equipe esquematizou os seguintes diagramas para a dinâmica geral das atividades do projeto:



AnimatorController do Carro

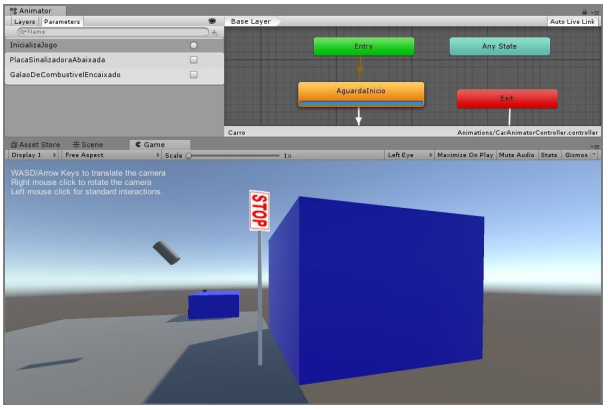


Máquina de estado relacionada ao pitstop

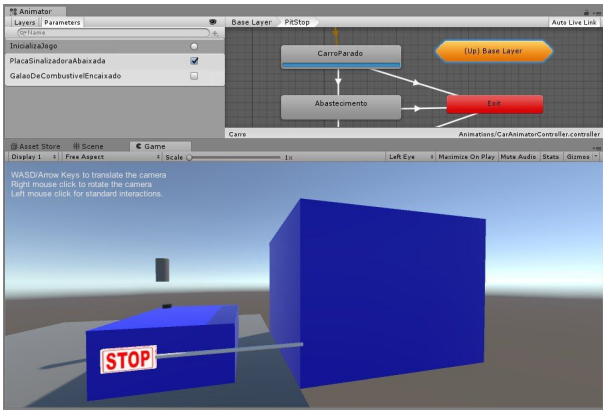
2. Dinâmica de Objetos, Ações e Tarefas

Com as máquinas de estado gerais definidas, os membros do grupo iniciaram a criação dos scripts e do fluxo das ações para determinadas atividades do projeto. Como nem todo o material gráfico está finalizado, foi decidido também utilizar, em paralelo, objetos simplificados para não haver desperdício de tempo.

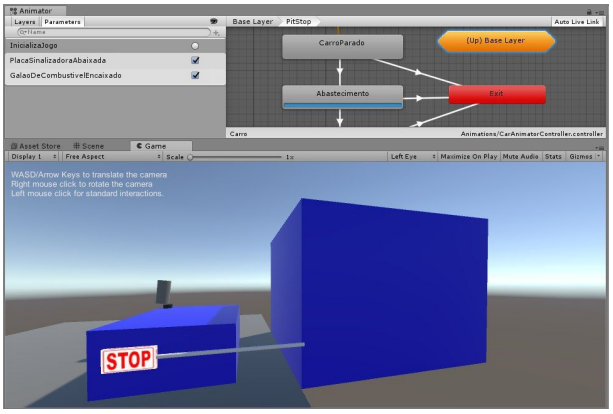
As atividades implementadas foram aquelas referentes à placa de "STOP", abastecimento da gasolina e parada do carro e todas elas estão ilustradas nas imagens abaixo:



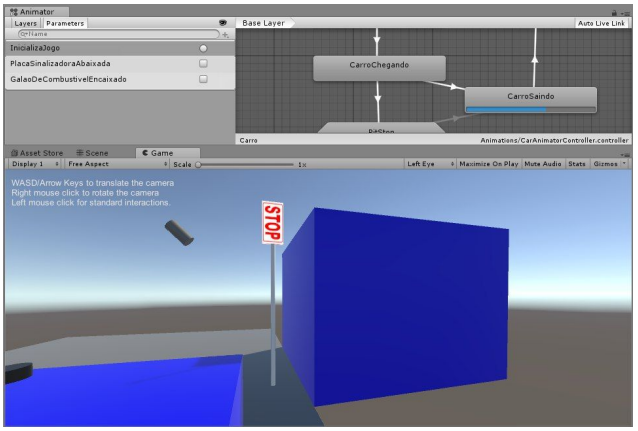
Estado que aguarda o trigger InicializaJogo



Placa abaixada e carro parado



Galão encaixado no local de abastecimento



Galão desencaixado, placa levantada e carro voltando a andar

Para todas estas 3 ações, foram criados os seguintes scripts:

PitStopCronometro.cs

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

public class PitStopCronometro : StateMachineBehaviour {

    float pitStopStartTime;
    float pitStopEndTime;
    float messageTime = 20.0f;
    bool  messagemDada = false;

    public void OnStateMachineEnter(Animator animator, int stateMachinePathHash){
        pitStopStartTime = Time.realtimeSinceStartup;
    }

    public void OnStateMachineExit(Animator animator, int stateMachinePathHash){
        pitStopEndTime = Time.realtimeSinceStartup;
        Debug.Log("Tempo da parada: " + (pitStopEndTime - pitStopStartTime) + " segundos");
    }

    // OnStateEnter is called when a transition starts and the state machine starts to evaluate this state
    //override public void OnStateEnter(Animator animator, AnimatorStateInfo stateInfo, int layerIndex) {
    //
    //}

    // OnStateUpdate is called on each Update frame between OnStateEnter and OnStateExit callbacks
    override public void OnStateUpdate(Animator animator, AnimatorStateInfo stateInfo, int layerIndex) {
        if(((Time.realtimeSinceStartup - pitStopStartTime) > messageTime) && (messagemDada == false)){
            messagemDada = true;
            Debug.Log("Ja se passaram 20 segundos...");
        }
    }
```

Script que conta o tempo do Pit Stop, ou seja, o tempo em que o carro está na máquina de estado Pit Stop.

C# PitStopAbastecimento.cs

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

public class PitStopAbastecimento : StateMachineBehaviour {

    float abastecimentoStartTime;
    float capacidade = 10.0f;
    bool  tanqueCheio;

    // OnStateEnter is called when a transition starts and the state machine starts to evaluate this state
    override public void OnStateEnter(Animator animator, AnimatorStateInfo stateInfo, int layerIndex) {
        abastecimentoStartTime = Time.realtimeSinceStartup;
        tanqueCheio = false;
    }

    // OnStateUpdate is called on each Update frame between OnStateEnter and OnStateExit callbacks
    override public void OnStateUpdate(Animator animator, AnimatorStateInfo stateInfo, int layerIndex) {
        if(((Time.realtimeSinceStartup - abastecimentoStartTime)  > capacidade) && (tanqueCheio == false)) {
            tanqueCheio = true;
            Debug.Log("Tanque cheio!");
        }
    }

    // OnStateExit is called when a transition ends and the state machine finishes evaluating this state
    override public void OnStateExit(Animator animator, AnimatorStateInfo stateInfo, int layerIndex) {
        float tanqueDeGasolina;

        if(tanqueCheio) tanqueDeGasolina = capacidade;
        else tanqueDeGasolina = Time.realtimeSinceStartup - abastecimentoStartTime;

        Debug.Log("Nivel do tanque: " + (tanqueDeGasolina/capacidade)*100.0f + " %");
    }
```

Script que conta o tempo em que o galão está encaixado, ou seja, permanece no estado Abastecendo. Esse tempo é proporcional ao nível do tanque.

C# PitStopEncaixeTanqueGalao.cs ×

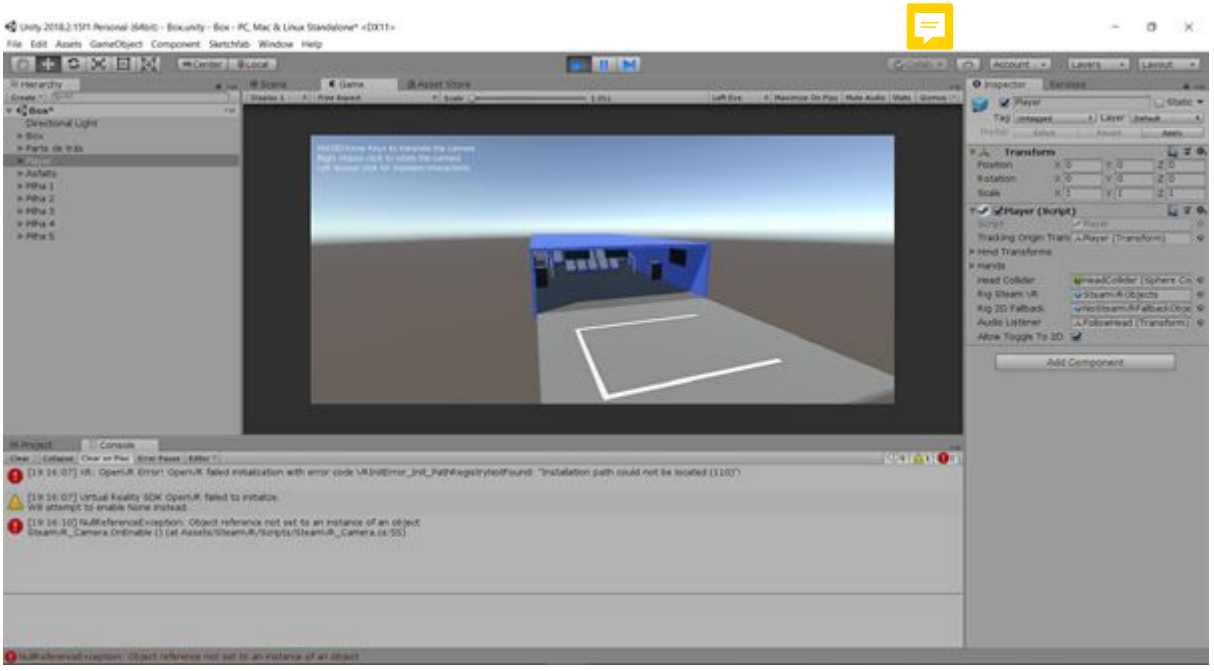
```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4   using UnityEngine.Events;
5
6   public class PitStopEncaixeTanqueGalao : MonoBehaviour {
7
8       public string FilterByName;
9       public UnityEvent aoIniciarColisao;
10      public UnityEvent aoTerminarColisao;
11
12      // Use this for initialization
13      void Start () {
14      }
15
16      // Update is called once per frame
17      void Update () {
18      }
19
20      void OnCollisionEnter(Collision info)
21      {
22          Debug.Log("Colidiu com " + info.gameObject.name );
23
24          if(info.gameObject.name == FilterByName){
25              aoIniciarColisao.Invoke();
26          }
27      }
28
29      void OnCollisionExit(Collision info)
30      {
31          if(info.gameObject.name == FilterByName){
32              aoTerminarColisao.Invoke();
33          }
34      }
35  }
```

Script de colisão do galão com o tanque de combustível para acionar a variável tanqueEncaixado

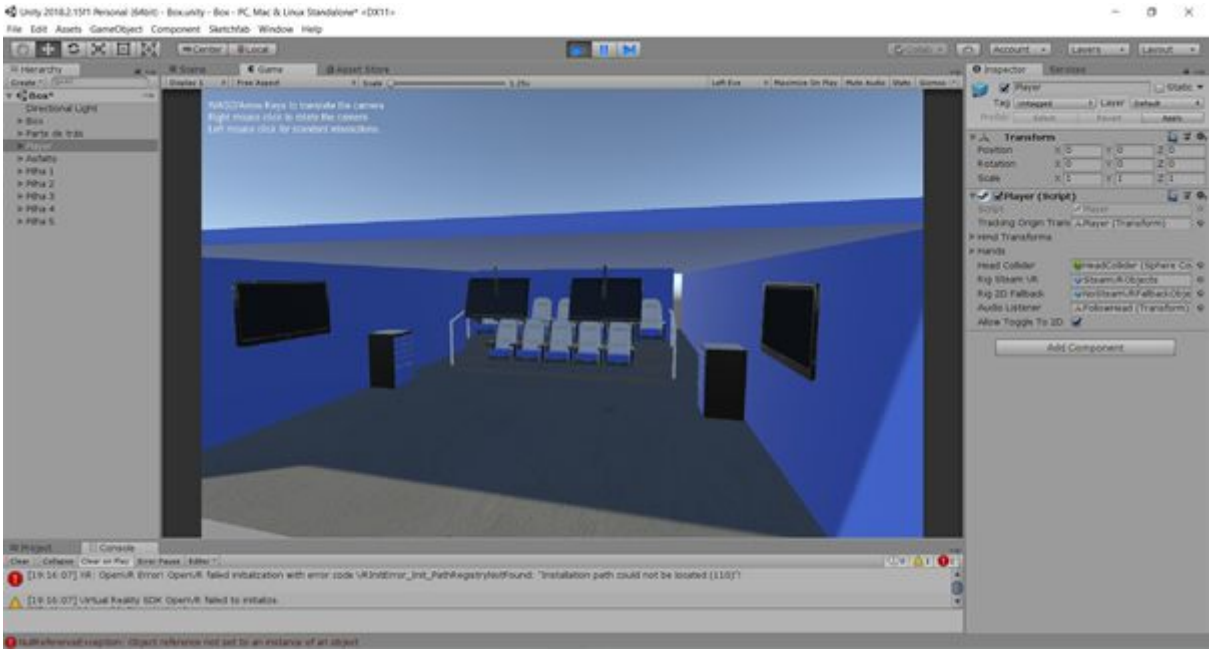
Finalmente, vale dizer também que a variável PlacaAbaixada é acionada no OnMinAngle() do componente CircularDrive.

3. Cenas/Modelos/Prefabs/Materiais/Efeitos

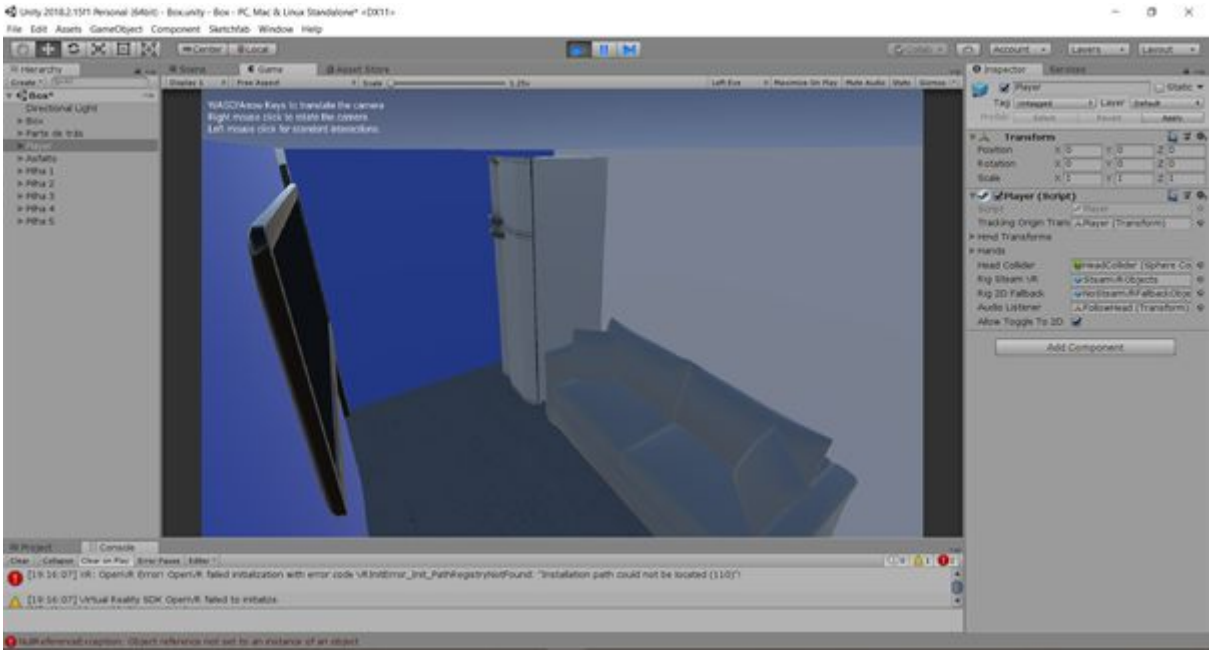
Em relação ao cenário, o grupo evoluiu significativamente pois conseguiu já criar todo um ambiente interno para o Box. As interações com portas, monitores e computadores estão sendo implementadas de tal maneira que aumente a realidade imersiva do usuário em relação ao jogo. O resultado da criação do ambiente pode ser encontrado nas figuras abaixo:



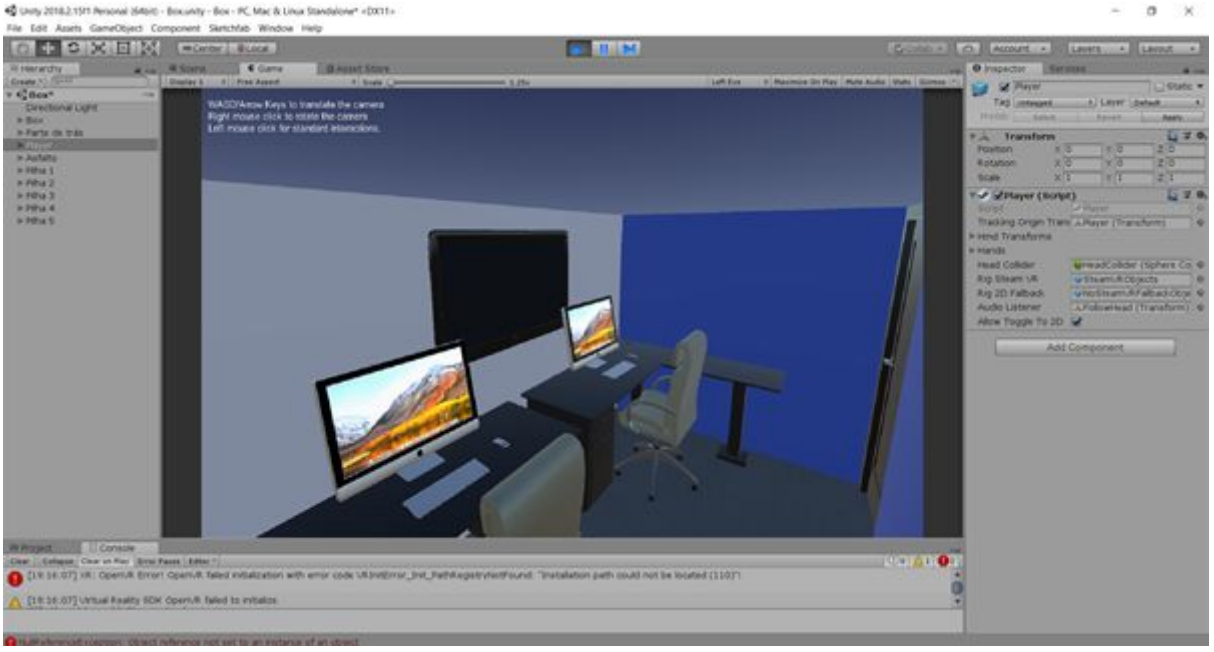
Visão externa da área do Box



Visão frontal da entrada do Box



Sala do piloto



Sala da Engenharia

4. Testes de Interação em Sala

Devido ao fato de o grupo ainda estar implementando os comandos e interações de cada atividade do jogo, os testes até o momento estão direcionados aos comandos simples do teclado (teclas ASDW) e do mouse. Assim, não houve interações com os equipamentos mais complexos disponibilizados durante as aulas.

Finalmente, vale dizer que todas as interações testadas até então estão se mostrando bastante positivas e importantes para o desenvolvimento.