Curso Tecnológico em Jogos Digitais

Prática em Laboratório UNITY
Roll a Ball

Murilo Boratto, Leandro Correia e Fred Adler

1 Resumo

A prática em laboratório a seguir tem como finalidade criar um simples jogo de armazenamento de objetos coletáveis. A estrutura consiste em uma estrutura 3D formada por um plano e uma esfera, de forma que o elemento esfera através do teclado ao pasar do tempo vai coletando elementos como condição de vitória. Este tutorial é baseado no tutorial ROLL-A-BALL do Unity3D [2].

Parte 1 - Construção da Cena para o jogo "ROLL A BALL"

1. Abrimos o UNITY. Criamos um novo projeto. Colocaremos como nome do projeto, "Roll a Ball". Selecionaremos como local a pasta "ProjetosUNITY/NomeAluno" que criamos no "Desktop". Selecionaremos o modo 3D.

2. Salvaremos a Cena em "File > Save scene as" ... Criaremos uma nova pasta em "Assets" chamada "Cenas" e salvaremos a cena com o nome de "Principal".

3. Criaremos o campo de jogo que será defnido com um plano. Nomearemos o plano como "Plano" e o posicionaremos na posição: $x = 2$, $y = 0$, $z = 2$.

4. Criamos uma esfera e renomeamos como "Player". Reposicionamos sua posição para as coordenadas $x = 0$, $y = 0.5$, $z = 0$. Atribuímos à esfera um componente Rigidbody.

5. Criamos uma nova pasta dentro de Assets chamada "Scripts". Uma vez criada arrastamos o script criado no ponto 5 para a pasta. Dando um doble click no script editaremos com o seguinte código C#:

```
private Rigidbody rb;
public float velocidade;

void Start () {
 rb = GetComponent<Rigidbody>();
}

void Update () {
 float movimentoHorizontal = Input.GetAxis("Horizontal");
 float movimentoVertical   = Input.GetAxis("Vertical");
```

```
 Vector3 movimento         = new Vector3(movimentoHorizontal, 0.0f, movimentoVertical);

 rb.AddForce(movimento * velocidade);
}
```

6. No Campo Inspector das propiedades de "Player" atribuindo o valor à variável *velocidade* = 10.
7. Presionamos o PLAY e podemos provar com diferentes valores da variável velocidade.

Parte 2 - Configuração da Câmara na visão do jogo

1. Reposicionamos a câmara, para os valores $x = 45$, $y = 10$, $z = 0$.
2. A idéia desta ação é arrastar a câmara junto com o jogador. Apertando o play vemos que neste momento isso não acontece.
3. Criaremos um script chamado "CamaraController" para implementar esta ação. Sendo asociado ao elemento câmara na hierarquia.

```
public GameObject player;
private Vector3 deslocamento;

void Start () {
 deslocamento = transform.position;
}

void Update () {
 transform.position = player.transform.position + deslocamento;
}
```

4. Como a varíavel "player" é pública e do tipo GameObject ela aparece no Inspector do UNITY. Sendo assim, faz-se necessário arrastar o elemento "Player" desde o ponto da Hierarquia até a propriedade da câmara . A continuação precionamos Play e provamos o movimento da câmara seguinto a bola.

Parte 3 - Construção dos Muros "The Walls"

1. Criamos um objeto vazio (Empty Game Object) e o chamamos de "Wall". Depois reinicializamos a sua posição iniocial para a origem.
2. Criamos um novo elemento cubo e o chamamos de "West Wall". Arrastamos sobre o elemento anterior para que o mesmo o contenha, fazendo-o que esteja hierarquicamente abaixo. Mudamos a escala do cubo para os valores $x = 0.5$, $y = 2$, $z = 20.5$ e o arrastamos até situar-lo à posição $x = -10$.
3. Selecionamos o objeto "West Wall" e o duplicamos. O chamaremos de "East Wall" colocando na posição $x = 10$.
4. Duplicamos "East Wall" e o chamamos de "North Wall". Podemos rotacionar o muro mudando o valor da coordenada y=90 ou aplicamos uma nova escala al cubo $x = 20.5$, $y = 2$, $z = 0.5$, estabelecendo como posição $x = 0$, $z = 10$.
5. E por último duplicamos "North Wall" e o chamamos de "South Wall" com uma posição de $z = -10$.
6. Presionamos o botão de Play e provamos como funcionam os muros no jogo.

Parte 4 - Cubos Coletáveis

1. Criamos um novo elemento cubo e o chamamos de "PickUp". Reinicializamos à origem e centralizaremos à vista da cena neste elemento.
2. Selecionamos o Player e desmarcamos seu nome no Inspector para que não haja interferencia na visão do novo cubo criado.
3. Situamos o cubo na posição $y = 0.5$, e o redimensionamos para o tamanho $x = 0.5$, $y = 0.5$ y $z = 0.5$. Logo, após o rotacionamos em 45 graus para os eixos x, y e z.
4. Criamos um script para o cubo chamado de "Rotator". E neste script utilizaremos a classe "Transform" com os métodos "Translate" y "Rotate".

```
void Update () {
   transform.Rotate(new Vector3(15, 30, 45) * Time.deltaTime);
}
```

5. Precionar Play e comprovar o giro do elemento cubo.
6. Criamos uma pasta chamada Prefabricados chamada "Prefabs". Arrasta-se o objeto "PickUp" da Hierárquica até a nova pasta de prefabricados na vista dos projetos.
7. Criamos um novo elemento vazio "Empty Game Object" chamado "PickUps". Arrastamos o objeto "PickUp" para a pasta PickUps.
8. Criar vários objetos PickUp e situar-los em posições aleatórias. Presionar o Play e comprovar que todos rotem.

Parte 5 - Colisões

1. Edite o script "PlayerController". Iremos utilizar a Classe "gameObject" e os métodos "tag" e "SetActive". Ambosos elementos irão identificar a colisão entre os elementos. O script a seguir ilustra isso:

```
void OnTriggerEnter(Collider other) {
   if (other.gameObject.tag == "PickUp")
     other.gameObject.SetActive(false);
}
```

2. Selecionaremos o prefabricado "PickUp" no visor do projeto e vamos a propiedad "Tag" no Inspector. Selecionaremos "Add Tag". Presionaremos o ícone "+" para inserir e no lugar da tag0 escreveremos "PickUp". Voltaremos a seleccionar o prefabricado "PickUp" no visor de projetos e no Inspetoor selecionaremos a tag, agora visível, "PickUp".
3. Para que funcione desativaremos a propriedade "Mesh Renderer" e selecionaremos o prefabricado "PickUp" e em seu componente "Box Collider" marcar "Is Trigger" para que as colisões sejam executadas pelo script.
4. Atribuímos al prefabricado "PickUp" um componente Rigidbody considerando um elemento do tipo "collider dinamic".
5. Para que os cubos não caiam no vazio ao presionar play, desativamos a gravidade no prefabricado "PickUp" e selecionamos "Is Kinematic" para que os cubos sejam submetidos as forças do motor físico.

Parte 6 - Gamificação da condição de vitória

1. No script "Player Controller" inserir:

```
private int count;

count = 0; //Inserir no método Start

count = count + 1; //Inserir depois de other.gameObject.SetActive(false);
```

2. Criaremos um novo objeto de tipo tipo UI > Text e o nomearemos "Count Text". O situaremos na posição "top", "center" e estabeleceremos as seguintes margens para os eixos: Pos $x = 20$ e Pos $y = -10$, para um tamanho da fonte da letra 20.

3. Inserir o seguinte código no script "PlayerController":

```
public Text countText; void SetCountText() {
  countText.text = "Count: " + count.ToString();
}
```

4. No método "Start" inserir: SetCountText();

5. Inserir também em "OnTriggerEnter", na continuação colocaremos $count = count + 1$;

6. Arrasta-se o novo objeto de texto sobre o Inspector da propriedade de player, onde neste momento se visualiza a propriedade chamada "Count Text". Presione Play e prove o funcionamento.

7. Criamos um novo objeto do tipo UI > Text chamado "Win Text". O situaremos na posição "middle" e "center", e estabelecemos o tamanho da fonte para 60. Estabeleceremos a posição do elemento em $x = 200$, $y = 60$ com tamanhos $Width = 668$ e $Height = 100$.

8. Inserimos no script "PlayerController": public Text winText;

9. No método "Start" inserimos: $winText.text =$;

10. No método "SetCountText" inserimos:

```
if (count >= 12)
  winText.text = "YOU WIN!!";
```

11. Arrastamos o novo objeto "Win Text" sobre a nova propriedade que aparece no Inspector para elemento "Player". Salve as alterações, aperte o Play e prove.

Referências

[1] Michelle Menard. *Game Development with Unity*. Course Technology Press, Boston, MA, United States, 2014.

[2] UNITY: Roll-a-ball tutorial, https://unity3d.com/es/learn/tutorials/projects/roll-ball-tutorial