ipvc estg

22as JORNADAS da COMPUTAÇÃO GRÁFICA e MULTIMÉDIA

Unity 3D

Unity®

1. Índice

2. Apresentação

Bem-vindos às nossas Jornadas da Computação Gráfica e Multimédia!

Este Workshop foi desenvolvido para vocês poderem aprender um bocadinho sobre utilizar Unity 3d e C# um dos muitos conhecimentos que nos adquirimos ao longo da nossa Formação académica.

Nós alunos da Licenciatura em **Engenharia da Computação Gráfica e Multimédia** (ECGM), e do **Curso Técnico Superior Profissional** (CTeSP), e de **Desenvolvimento de Web e Multimédia** (DWM) fizemos e estaremos a Demonstrar este Workshop para vocês.

Se desejarem saber mais sobre as jornadas, ou sobre o Instituto Politécnico de Viana do Castelo (IPVC) , podem visitar os respetivos sites http://jcgm.estg.ipvc.pt/ e https://www.ipvc.pt/.

3. Tecnologias

3.1. Unity

Unity 3D é um programa de desenvolvimento de jogos e aplicações interativas, conhecido como uma Game Engine.
Este oferece um conjunto abrangente de ferramentas e recurso que permitem aos programadores criar experiências imersivas em 2D, 3D, realidade virtual (VR) e realidade aumentada (AR).
Com o Unity, pode se criar jogos simples até experiências complexas e multiplataforma para uma ampla variedade de dispositivos, incluindo PC, consolas, telefones e muitos mais.

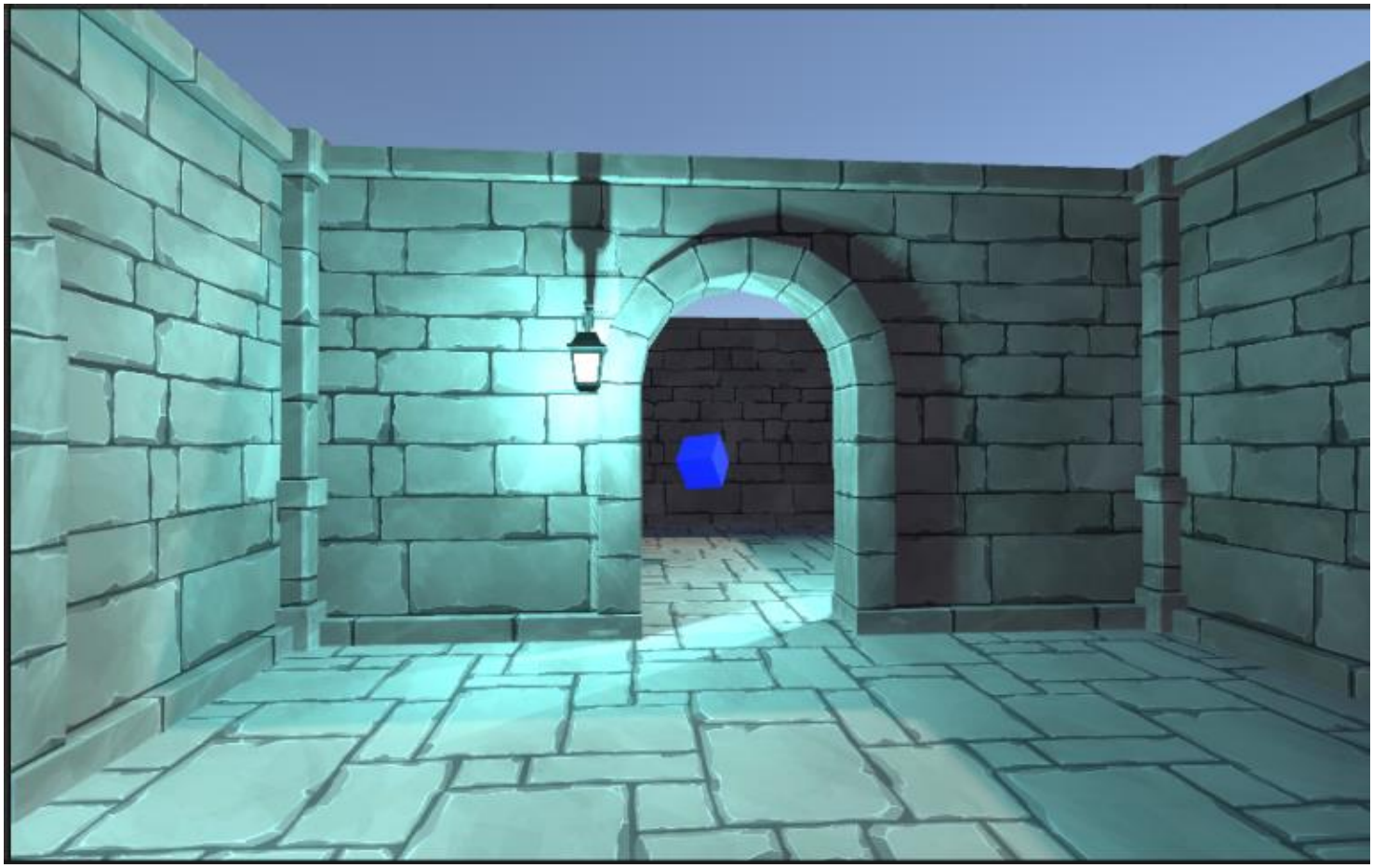
3.2. C#

C# é uma linguagem de programação orientada a objetos desenvolvida pela Microsoft, amplamente utilizada na criação de aplicações, jogos e soluções empresariais. Com uma sintaxe similar ao C e C++, é especialmente popular no desenvolvimento de jogos na Unity 3D devido à sua robustez e integração nativa com o Unity.

4. Workshop

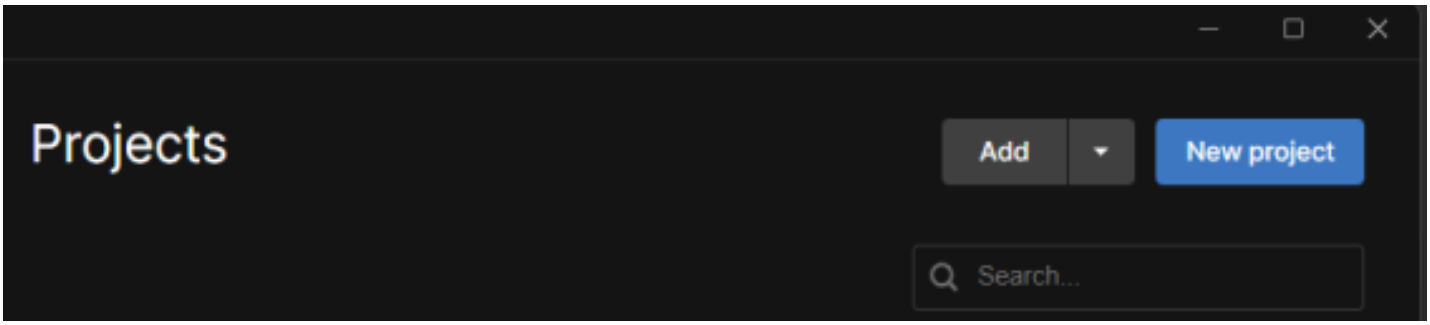
Aqui esta o resultado do nosso Jogo final que vamos fazer neste Workshop usando Unity 3D e C#.

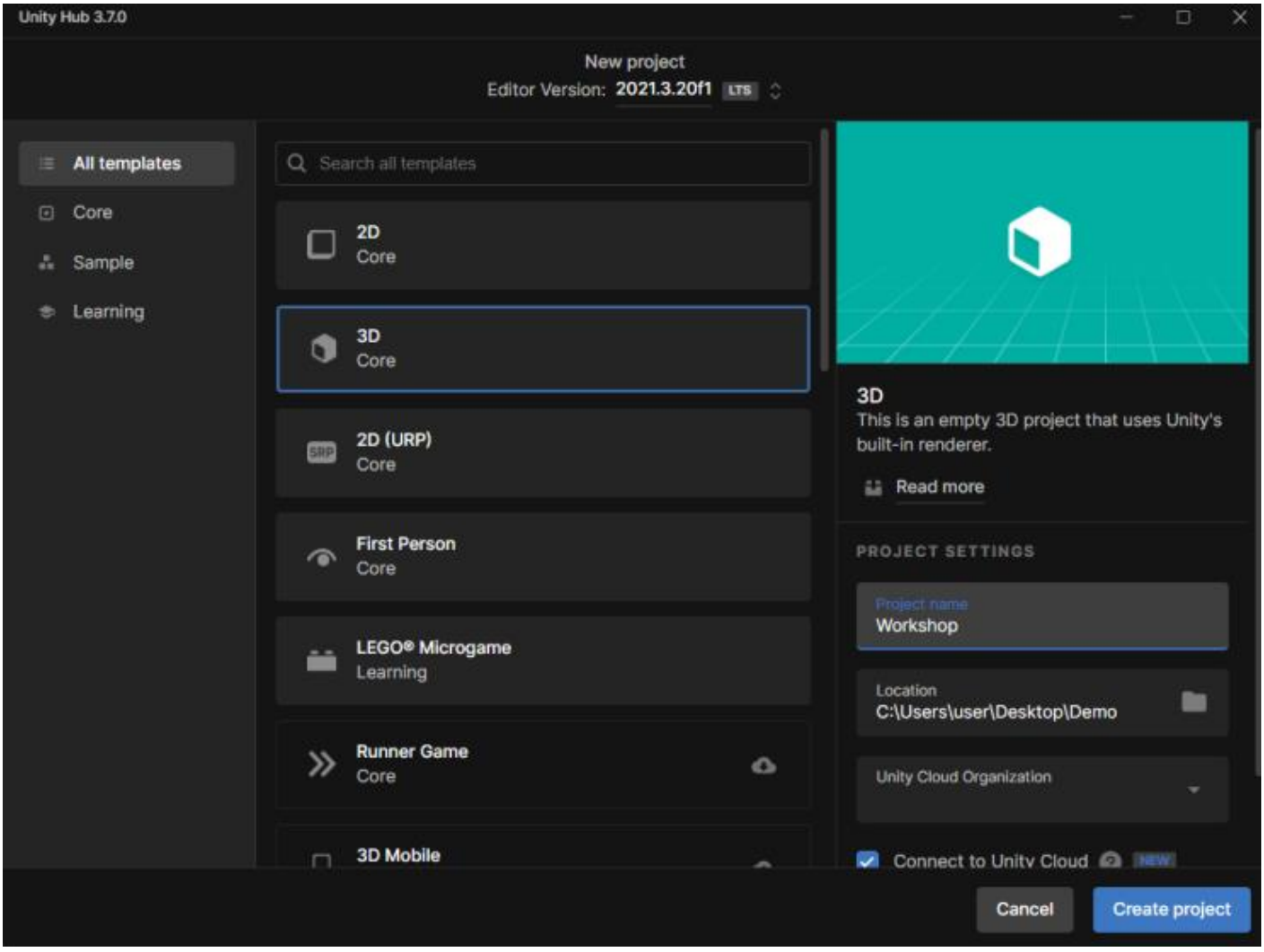
Neste Jogo vai haver Geração processual de uma masmorra mas o nosso objectivo neste Workshop será usar o movimento e colisões da nossa personagem que iremos controlar em 3D terá uma perspectiva de 1º Pessoa.



4.1. Criação de Projeto

4.1.1.Para começar abram o “Unity Hub” que esta no Ambiente de Trabalho .
4.1.2.Vamos criar um novo projeto Unity 3D.
4.1.3.Clicar em New Project -> Selecionar 3D Core -> “Escolher um nome para o projeto” -> Clicar em Create Project.



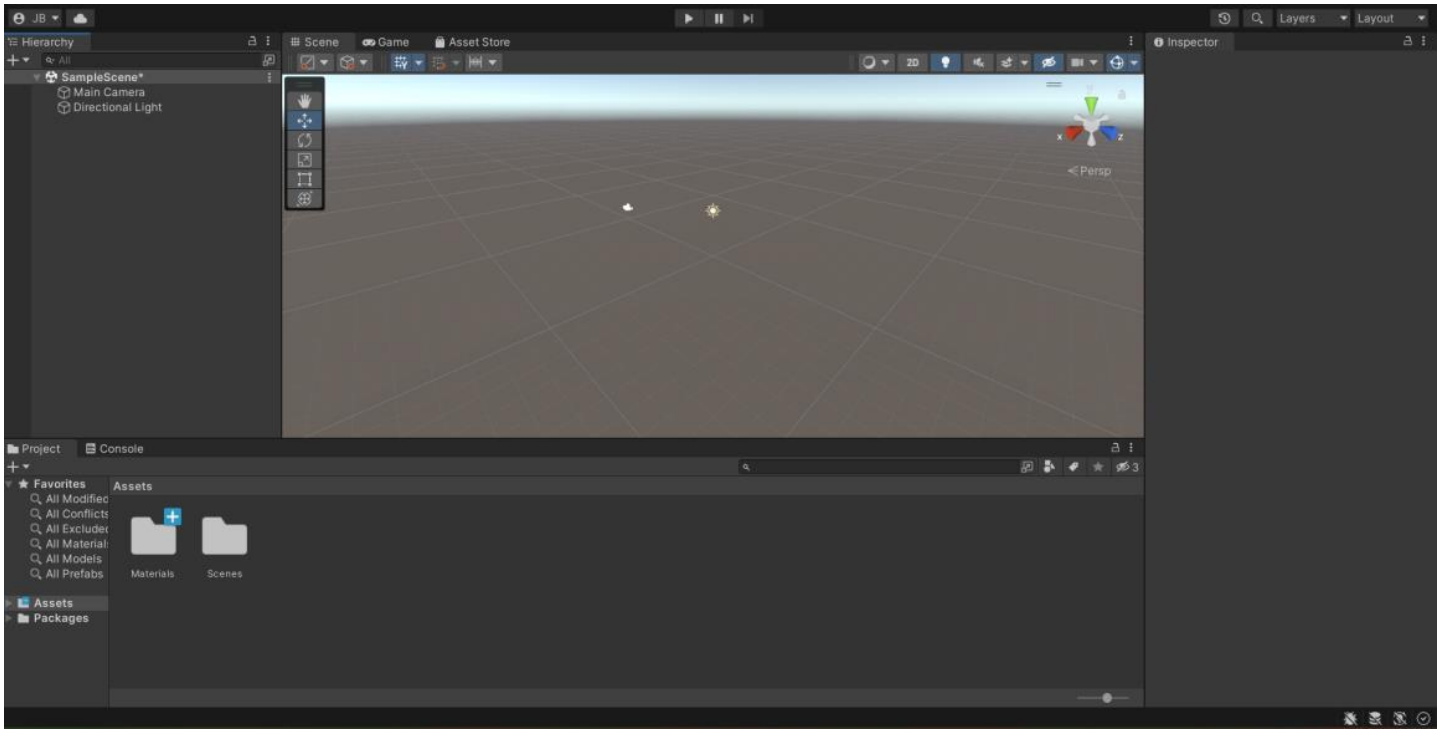


4.2. Preparação dos materiais

4.2.1.Após a criação do novo projeto.

4.2.2.Deverão importar os materiais que estão na pasta junto a este ficheiro chamada de “Material” que e importá-los para o projeto 3D.

4.2.3.Para isso terão de arrastar esses assets para a pasta “Material”” dentro do projeto.



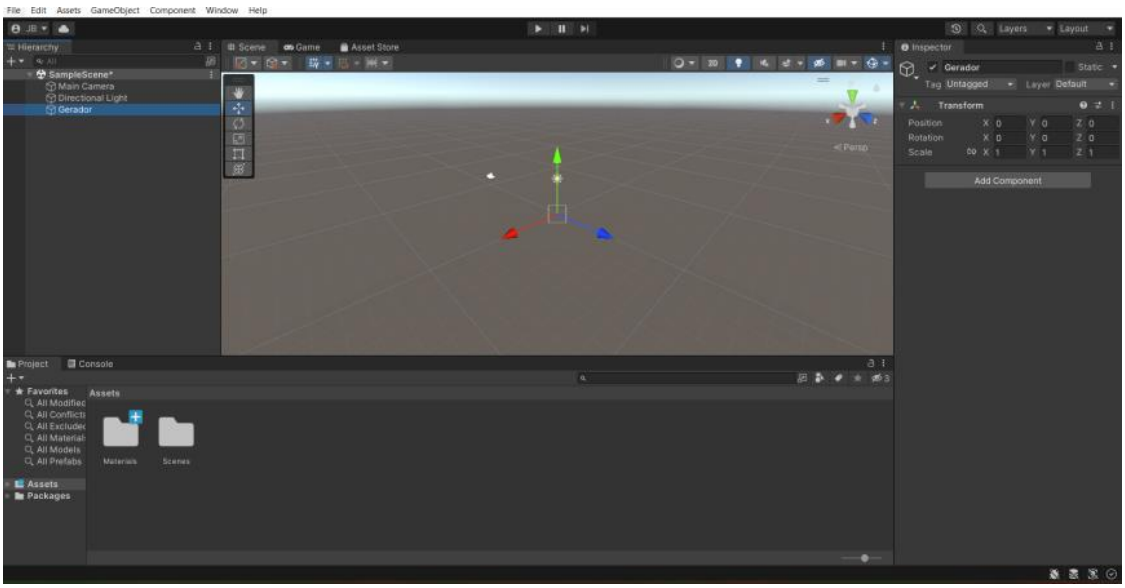
4.3. Criar o gerador da Masmorra

Agora vamos Começar o projeto por criar o gerador da masmorra.

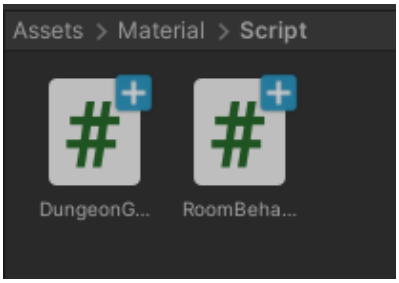
4.3.1. Posicionem o vosso rato dentro da “Hierarchy” (canto superior esquerdo) e façam um Right-Click.

4.3.2. De seguida escolhe a Opção Create Empty e deem o nome “Gerador” ao Objeto criado.

4.3.3. Cliquem no Objecto “Gerador” na Hierarchy e confirmem se o vosso objecto esta como na respectiva imagem no “Inspector” (Canto Esquerdo).



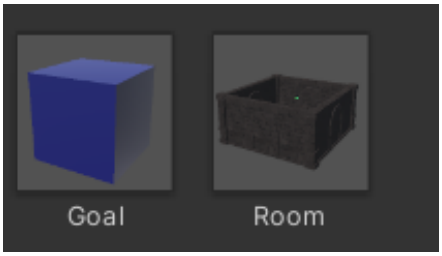
4.3.4. Agora vão a pasta “Material”, cliquem na pasta “Script”, e vão encontrar 2 ficheiros C# Script que são o “DungeonGenerator” , “Goal” o “RoomBehaviour”.



4.3.5. Cliquem e arrestem o “DungeonGenerator” para cima do objecto Gerador.

4.3.6. Voltem a pasta “Material”, cliquem na pasta “Prefab”, onde o ficheiro Prefab “Room”.

4.3.7. E repitam para a Goal.

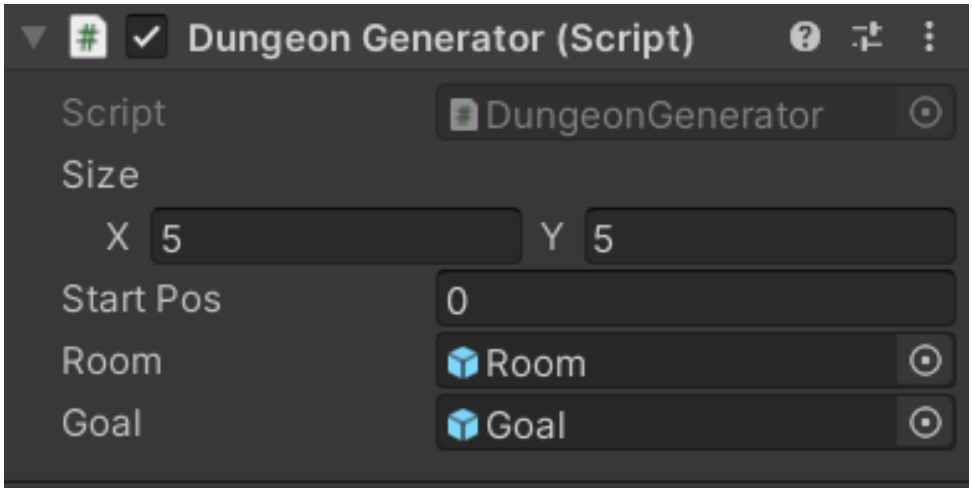


4.3.8.Agora cliquem no objeto Gerador na Hierarchy.

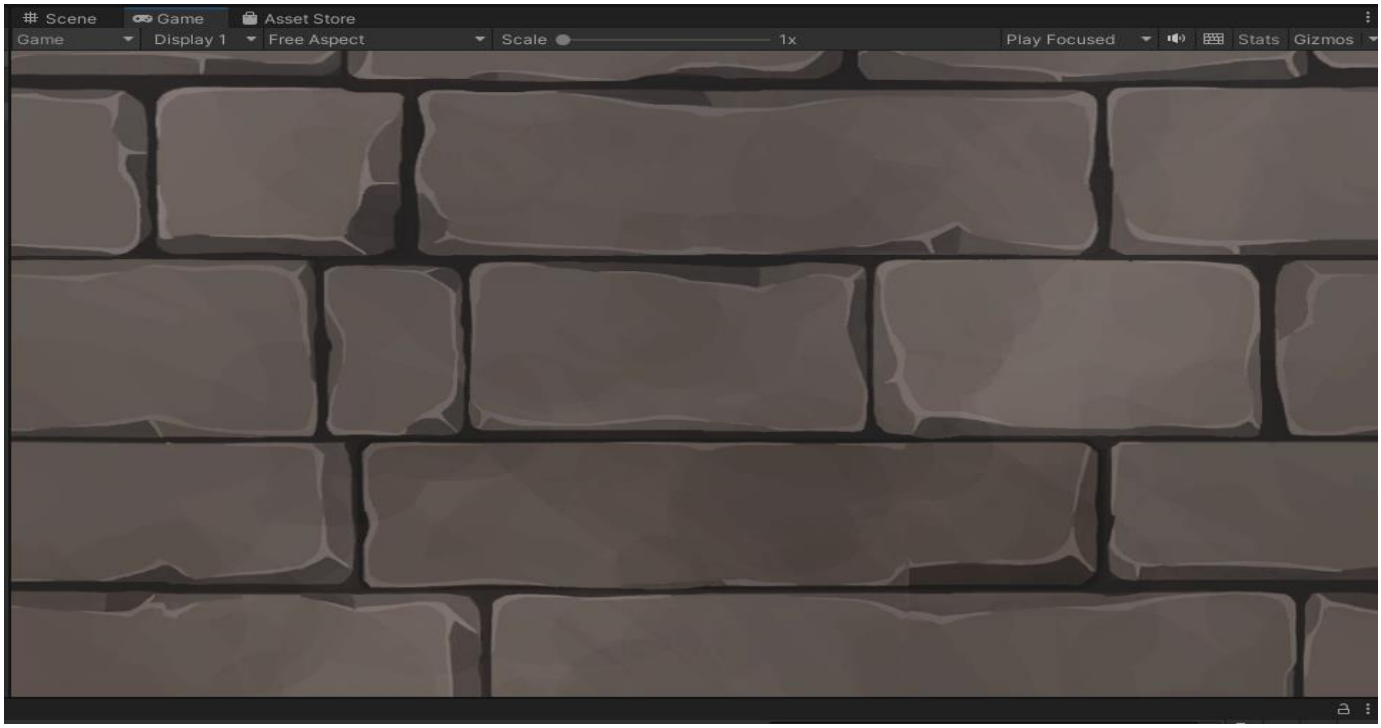
4.3.9.Podem ver o DungeonGenerator(Script) no Inspector.

4.3.10. Arrestem o ficheiro “Room” para a respectivo parâmetro dentro do DungeonGenerator Script no Inspector.

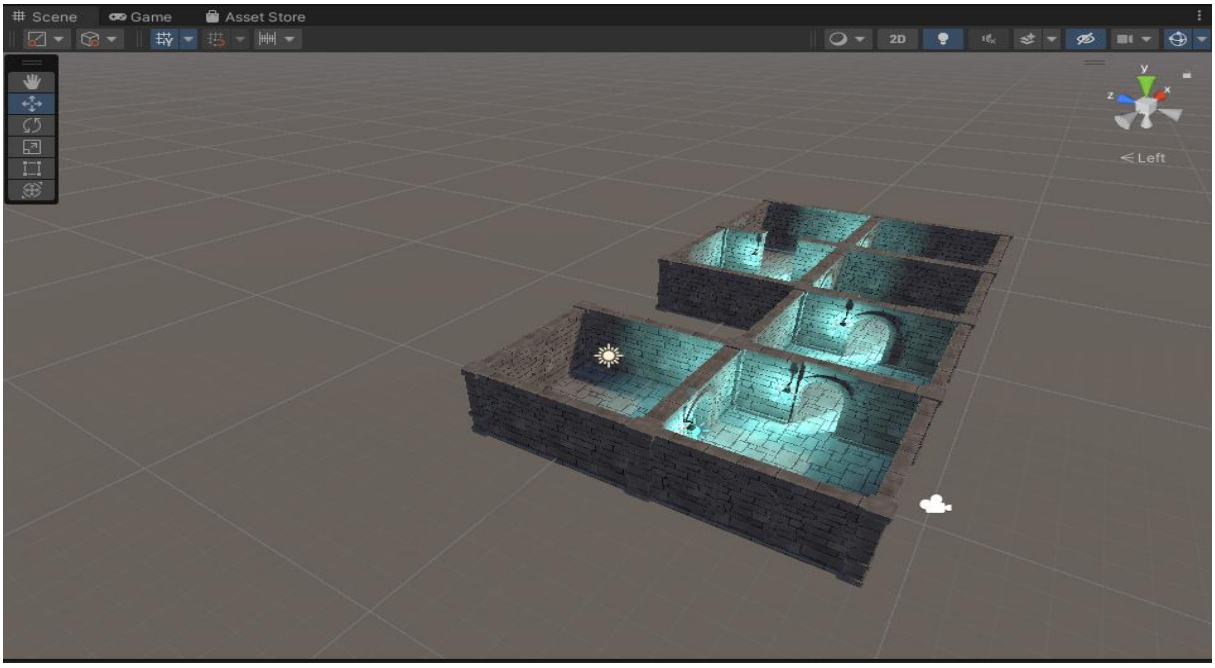
4.3.11. Ajustem os parâmetros X e Y com dois Números a vossa escolha.



4.3.12. Agora Podem testar o Gerador clicando neste Botão Play  que se encontra no topo debaixo da nossa barra de ferramentas. Vai vos aparecer assim mas não se assustem.



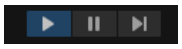
4.3.13. Cliquem na Tab Scene que encontra no canto superior da Janela para ver o resultado do nosso gerador.



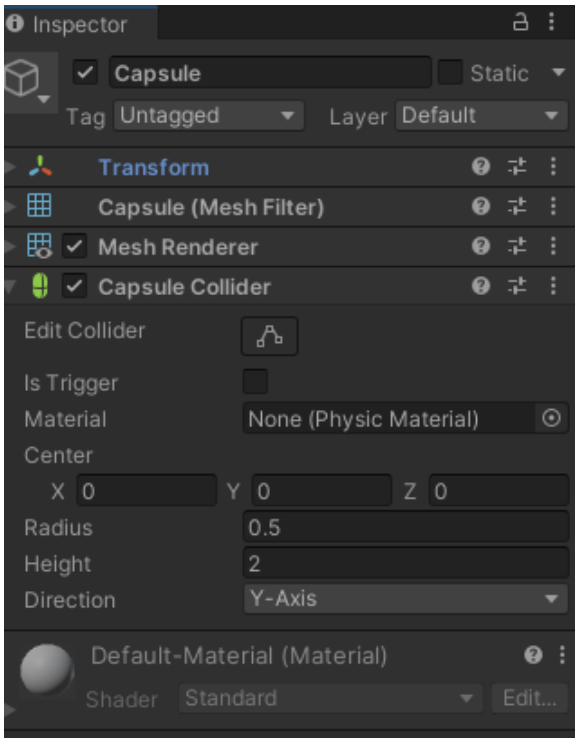
Para explorar o vosso projeto. Usem os atalhos de teclas:

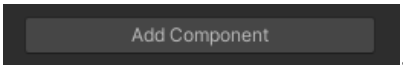
- ALT + LEFT-CLICK -> rodar ou ver livremente o cenário (scene) sem sair da mesma posição
- ALT + SCROLL BUTTON -> Mover posição ao qual se está a ver o cenário (scene)
- SCROLL UP/DOWN -> zoom in e zoom out

4.4. Criar o Player

Cliquem no Botão play  **de novo para Voltar mos a editar o nosso Projeto, se não o fizer o projeto não vai gravar as vossas alterações quando clicarem de novo.**

4.4.1.Voltem a "Hierarchy" (canto superior esquerdo) e façam um Right-Click.
4.4.2.De seguida escolhe a Opção 3D Object > Capsule e deem o Nome "Player" ao Objeto criado.
4.4.3.No Inspector (Canto Esquerdo) vão ver que as propriedades do Objecto tem um Capsule Collider.

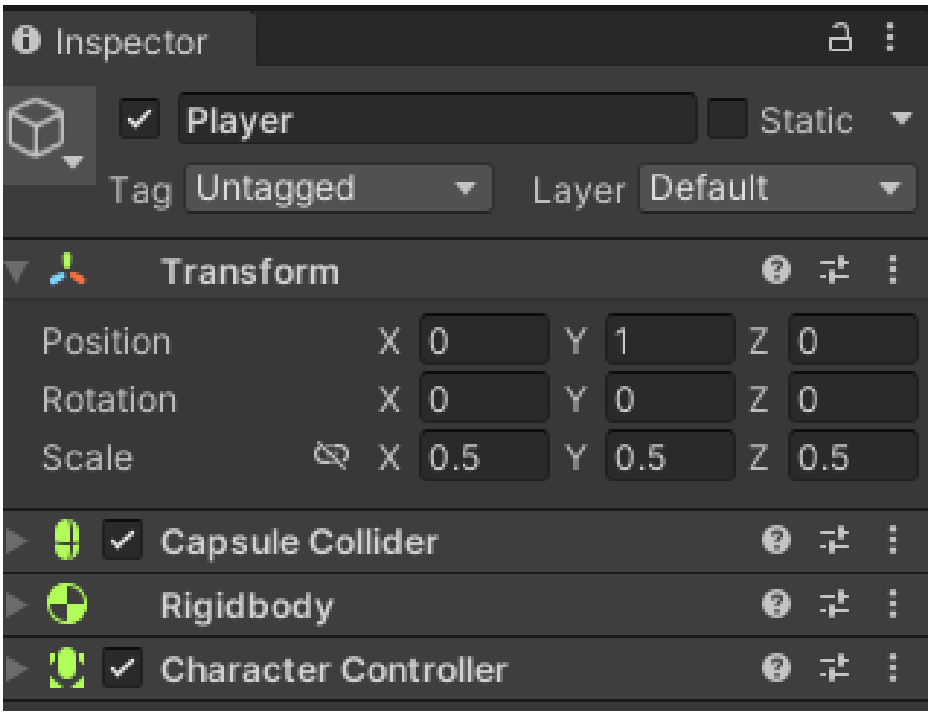


4.4.4.Selecionem o Player na Hierarchy e cliquem no Botão 

.
4.4.5.Usando a barra de navegação procurem pelo RigidBody e Selecionem o.
4.4.6.Repitam o processo mais outra vez para o Componente Characther Controller.

4.4.7. Agora reajustem o tamanho de 1 para 0.5 do Player no Inpscetor no componente Transform parâmetros Scale X,Y e Z.

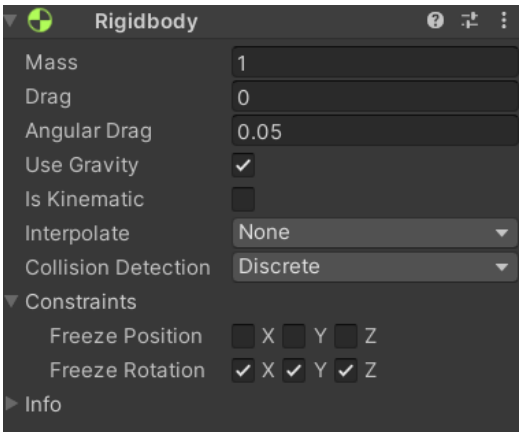
4.4.8. Verifiquem se os parâmetros do Transform estão iguais a imagem.



4.5. Movimentar o Player

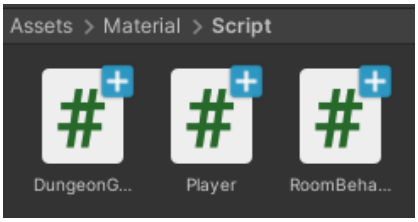
4.5.1. Podem testar o Projeto clicando neste Botão Play  que se encontra no topo debaixo da nossa barra de ferramentas.

4.5.2. Vão ao rigidbody > Constrains e verfiquem o Frezze rotation em X , Y e Z se esta marcado.



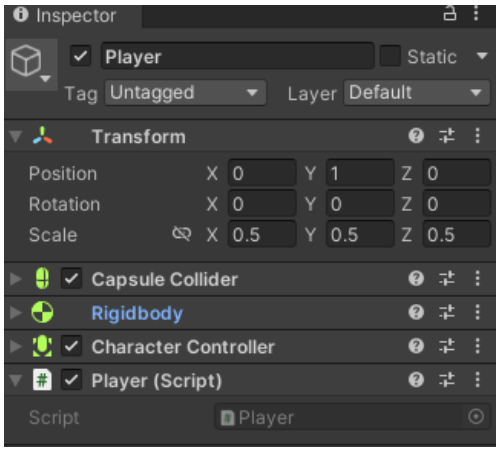
4.5.3. Vão a pasta “Material”, cliquem na pasta “Script”.

4.5.4. Façam Right-Click e selecione a opção Create > C# Script.



4.5.5. Dem o Nome de “Player” ao C# Script.

4.5.6. Arrastem o Script para o Inpscetor com Player selecionado.



4.5.7.Façam Double-Left--Click sobre o “Player” Script.

4.5.8. Vai vos abrir o vosso editor de código com este Código inicial.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class Player : MonoBehaviour
{// Start is called before the first frame update
    void Start(){  }
    // Update is called once per frame
    void Update(){}
}
```

4.5.9.Agora vamos começar Programar

4.5.10. Antes do Void Start vão introduzir as varáveis:

```
    // Referência ao CharacterController para controlar o movimento do jogador

    public CharacterController control;

    // Referência ao Rigidbody para aplicar a gravidade

    public Rigidbody rb;

    // Velocidade de movimento do jogador

    public float speed = 5f;

    // Variáveis para armazenar a entrada do teclado

    private float movementX;

    private float movementZ;

    // Variável para armazenar a velocidade atual do jogador

    private Vector3 velocity;
```

4.5.11. Dentro da Função void Start ()

```
    void Start()
    {
        // Verifica se o Rigidbody está presente e torna-o cinemático para evitar
    interferência da física
        if (rb)
        {
            rb.isKinematic = true;
        }
    }
```

4.5.12. Na Função void Update () introduzam

```
void Update()
    {
        // Obtém a entrada do teclado para o movimento horizontal e vertical
        movementX = Input.GetAxisRaw("Horizontal");
        movementZ = Input.GetAxisRaw("Vertical");
        // Calcula o vetor de movimento do jogador com base na entrada do teclado
        Vector3 move = transform.right * movementX + transform.forward * movementZ;
        // Move o jogador usando o CharacterController
        control.Move(move * speed * Time.deltaTime);
        // Aplica a gravidade ao jogador
        ApplyGravity();
    }
```

4.5.13. Dentro da Função void ApplyGravity()

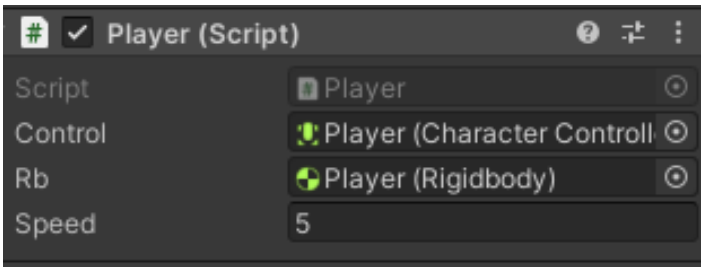
```
 void ApplyGravity()
    { // Adiciona a gravidade à velocidade vertical do jogador
        velocity.y += Physics.gravity.y * Time.deltaTime;
        // Move o jogador verticalmente de acordo com a velocidade
        control.Move(velocity * Time.deltaTime);

        // Verifica se o jogador está no chão
        if (control.isGrounded && velocity.y < 0)
        {
            // Reseta a velocidade vertical se estiver no chão
            velocity.y = 0f;
        }
    }
```

4.5.14. Que mover o Player pelo input feito por o jogador e Pela sua velocidade e tempo.

4.5.15. Verifiquem se salvam o Script no vosso editor de Código ou usem o atalho Crtl+S para salvar.

4.5.16. Voltem a Unity e vão vem no Inspcetor com o Player selecionado que o Script e as variáveis publicas.

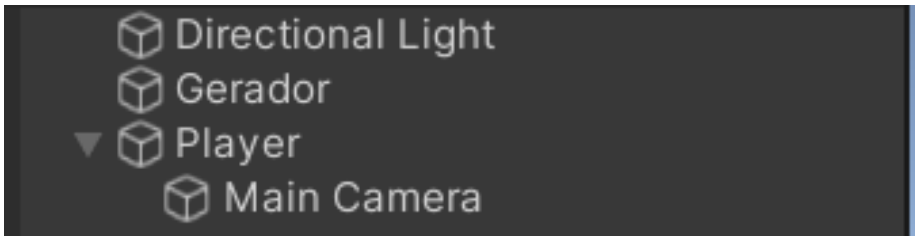
4.5.17. Arrestem o Player da Heirarchy para a variável Control para podermos aplicar o movimento ao Player.



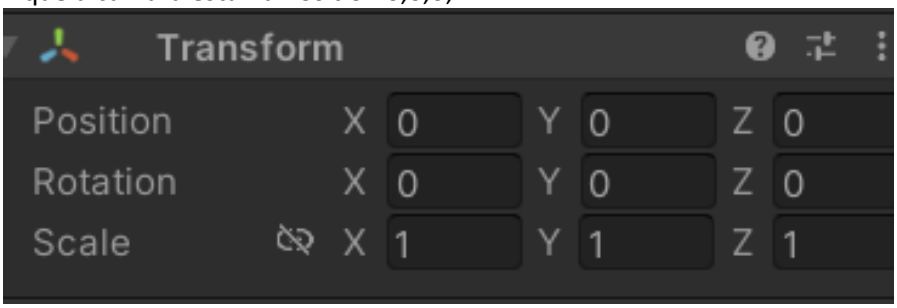
4.5.18. Podem testar outra vez o projeto mas vão perceber que não conseguem ver o projeto. Como a câmara esta fixa numa posição.

4.6. Movimentar a Câmara

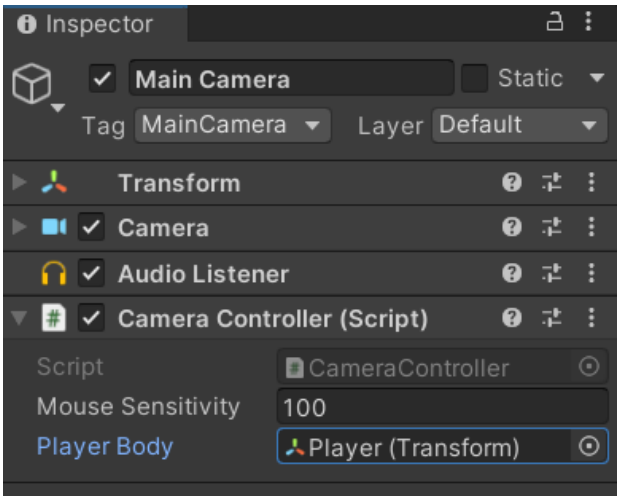
4.6.1.Primeiro arrastem a Main Camera que esta na Heirarchy para o player.



4.6.2.Verifiquem que a câmara esta na Position 0,0,0;



4.6.3.Vão a pasta “Material”, cliquem na pasta “Script”.
4.6.4.Façam Right-Click e selecione a opção Create > C# Script.
4.6.5.Dem o Nome de “CameraController” ao C# Script.
4.6.6.Arrastem o Script para o Inpscetor com Main Camera selecionada.
4.6.7.Façam Double-Right-Click sobre o “CameraController” Script
4.6.8.Vai vos abrir o vosso editor de código .
4.6.9.Aqui esta o código para ser inserido esta no documento “CameraScript.Txt” na mesma pasta do nosso guião.
4.6.10. Copiem do ficheiro texto código para o vosso CameraController script.
4.6.11. Salvem o Código e voltem para o Unity.
4.6.12. Arrestem o jogador para a variável Player no Main Camera no local da Script.



4.6.13. Podem testar agora o movimento do Player outra vez.

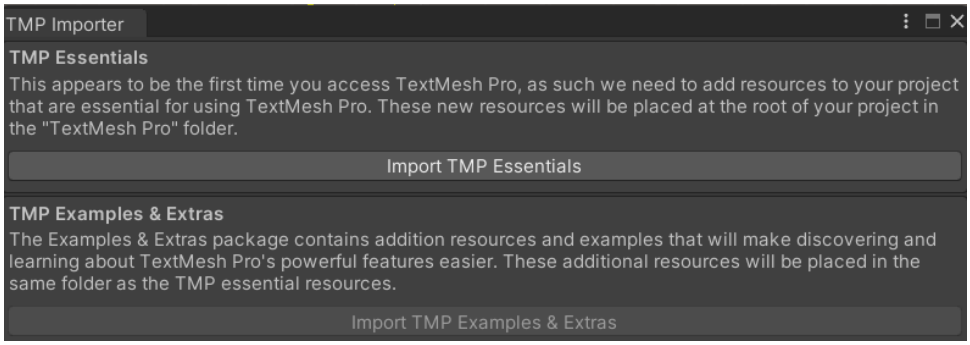
4.7. Ecrã Final

Agora vamos finalizar o nosso jogo criando um Ecrã para demonstramos que acabo-mos.

4.7.1.Voltem a “Hierarchy” (canto superior esquerdo) e façam um Right-Click.

4.7.2.De seguida escolhe a UI >Text Mesh Pro.

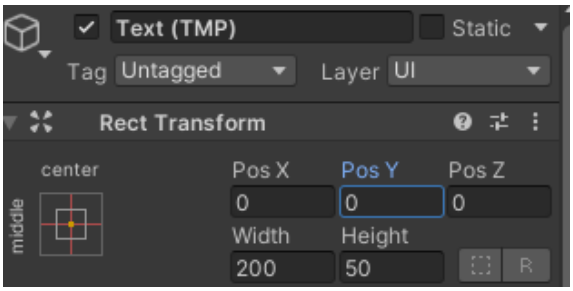
4.7.3.Vai aparecer esta notificação Clique Import TMP Essentials.



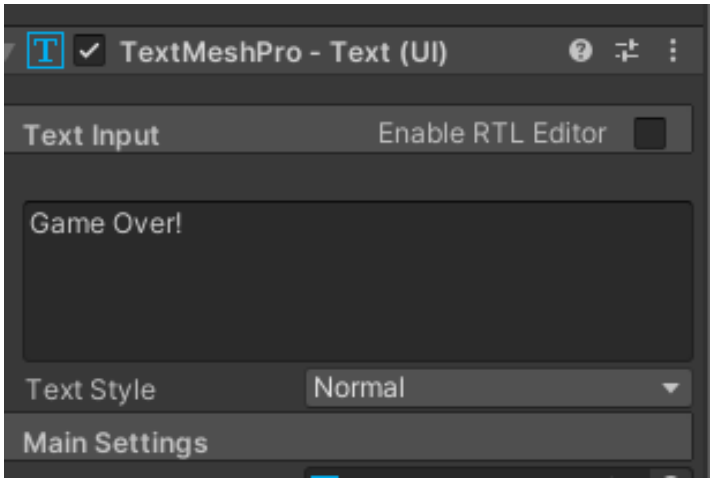
4.7.4.Vai aparecer na Hierarchy Canvas, o Text(TMP) e EventSystem.

4.7.5.Mas o que nos interessa agora e o TXT (TMP).

4.7.6.No Inspector e reposiciona para as condenadas na imagem.



4.7.7.No TextMeshPro – Text (UI) ,limpa o texto e Escreve “Game Over!”.



4.7.8.Agora volta ao Player Script e Introduz estas novas nas variáveis.

```
public GameObject winTextObject;
```

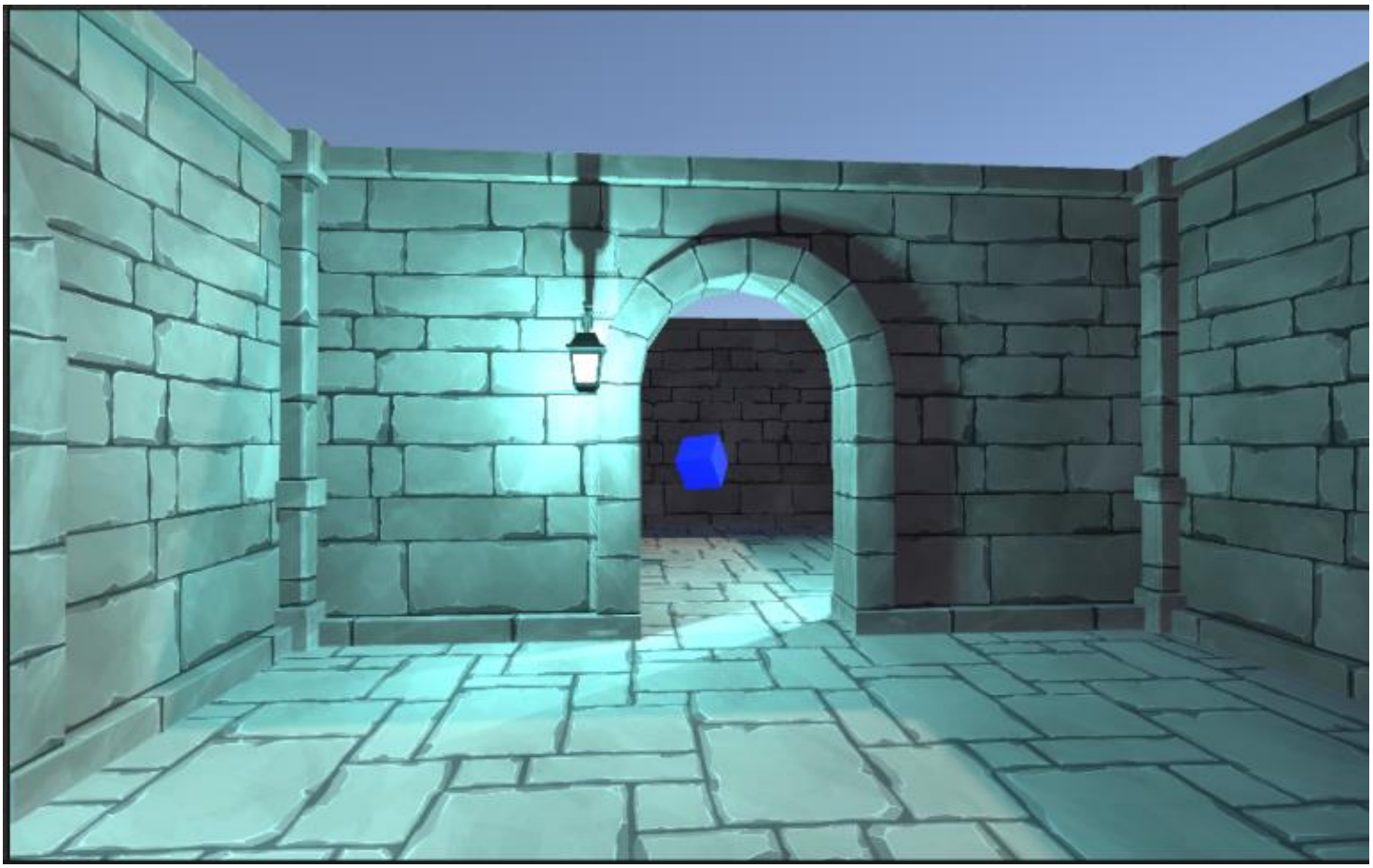
4.7.9.No void Start()

```
winTextObject.SetActive(false);
```

4.7.10. No final depois da função applyGravity poem esta nova função

```
    // Método chamado quando o jogador colide com um trigger
    void OnTriggerEnter(Collider other) {
        // Verifica se o jogador colidiu com um objeto com a tag "Finish"
        if (other.gameObject.CompareTag("Finish"))
        {
            // Desativa o objeto com a tag "Finish"
            other.gameObject.SetActive(false);
            // Ativa o texto de vitória
            winTextObject.SetActive(true);
        }
    }
```

Vamos testar novamente o nosso projeto, clicando em "Play". Agora sim, podemos dar por concluído este workshop em Unity! Esperamos que agora tenham uma compreensão das imensas possibilidades e projetos que se podem criar com a Unity Engine!



5. Extras

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

public class Player : MonoBehaviour
{
    // Referência ao CharacterController para controlar o movimento do jogador
    public CharacterController control;

    public GameObject winTextObject;
    // Referência ao Rigidbody para aplicar a gravidade
    public Rigidbody rb;
    // Velocidade de movimento do jogador
    public float speed = 5f;
    // Variáveis para armazenar a entrada do teclado
    private float movementX;
    private float movementZ;
    // Variável para armazenar a velocidade atual do jogador
    private Vector3 velocity;

    void Start()
    {  winTextObject.SetActive(false);
        // Verifica se o Rigidbody está presente e torna-o cinemático para evitar interferência da
física
        if (rb)
        {
            rb.isKinematic = true;
        }
    }
    void Update()
    {
        // Obtém a entrada do teclado para o movimento horizontal e vertical
        movementX = Input.GetAxisRaw("Horizontal");
        movementZ = Input.GetAxisRaw("Vertical");

        // Calcula o vetor de movimento do jogador com base na entrada do teclado
        Vector3 move = transform.right * movementX + transform.forward * movementZ;
        // Move o jogador usando o CharacterController
        control.Move(move * speed * Time.deltaTime);

        // Aplica a gravidade ao jogador
        ApplyGravity();
    }

    // Método para aplicar a gravidade ao jogador
```

```
    void ApplyGravity()
    {
        // Adiciona a gravidade à velocidade vertical do jogador
        velocity.y += Physics.gravity.y * Time.deltaTime;
        // Move o jogador verticalmente de acordo com a velocidade
        control.Move(velocity * Time.deltaTime);

        // Verifica se o jogador está no chão
        if (control.isGrounded && velocity.y < 0)
        {
            // Reseta a velocidade vertical se estiver no chão
            velocity.y = 0f;
        }
    }
    // Método chamado quando o jogador colide com um trigger
    void OnTriggerEnter(Collider other) {
        // Verifica se o jogador colidiu com um objeto com a tag "Finish"
        if (other.gameObject.CompareTag("Finish"))
        {
            // Desativa o objeto com a tag "Finish"
            other.gameObject.SetActive(false);
            // Ativa o texto de vitória
            winTextObject.SetActive(true);
        }
    }

}
```

6. Conclusão

Espero que tenho gostado do nosso workshop e tenham aprendido um bocadinho como usar Unity 3d e C#. Com muito Gosto do curso de **Engenharia de Computação Gráfica e Multimédia**, espero que tenham gosta e podem jogar o vosso jogo a vontade.