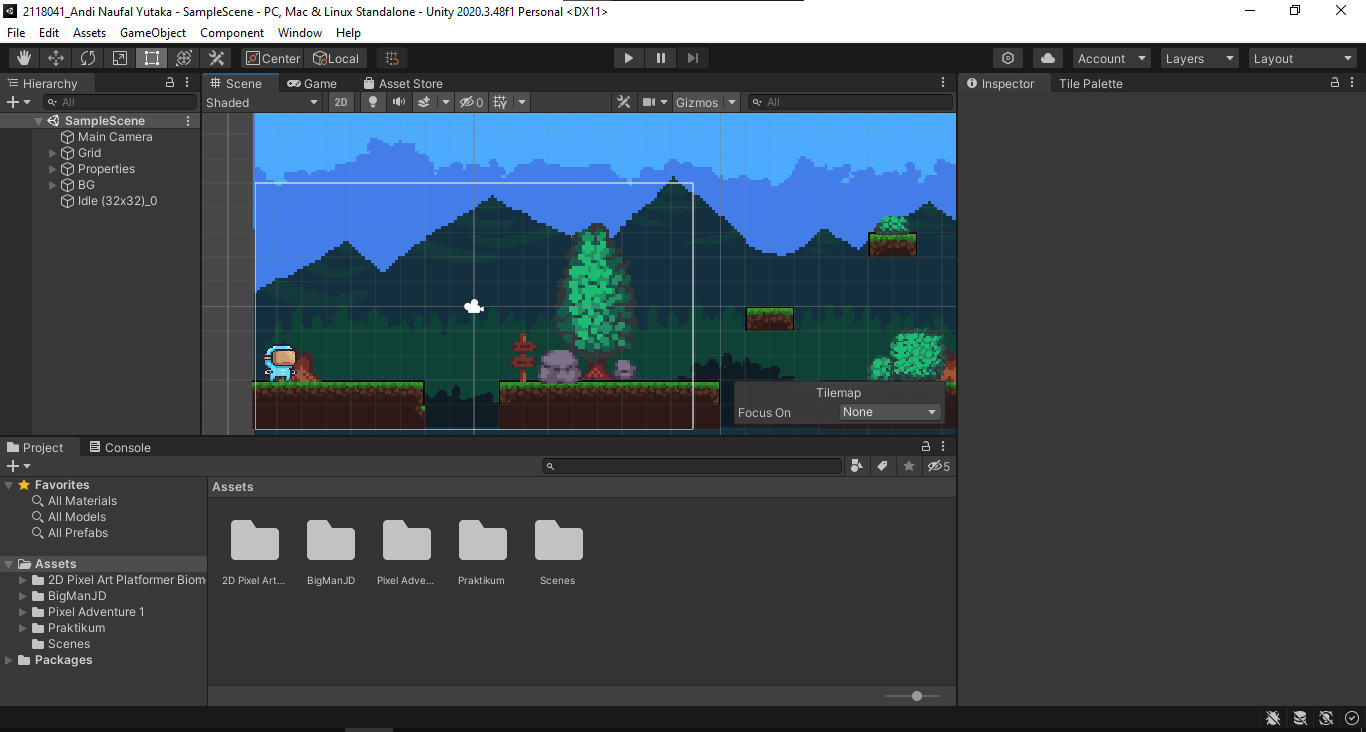
# 8 CAMERA AND CHARACTER MOVEMENT

|  |  |  |
| --- | --- | --- |
| **NIM** | : | 2118041 |
| **Nama** | : | Andi Naufal Yutaka |
| **Kelas** | : | D |
| **Asisten Lab** | : | Mohammad Akbar Ilham (2118091) |

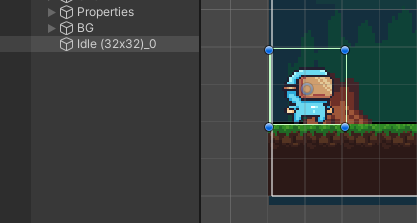
## Tugas 8: Camera & Character Movement

1. **Membuat Pergerakan Player**
2. Buka file projek *Unity* sebelumnya pada bab 7 untuk digunakan kembali.



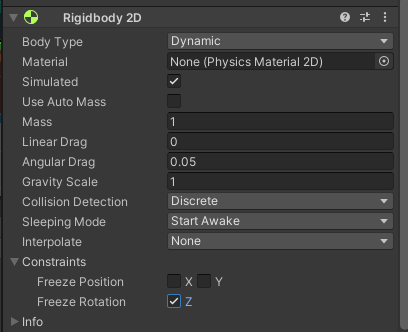
### 8.1 Membuka Project Unity

1. Pilih karakter yang telah di-*download*, lalu *import* kedalam Hirarki pada bagian *idle*.



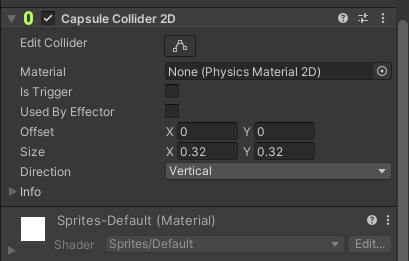
### 8.2 *Import Character*

1. Klik karakter lalu tambahkan Component *Rigidbody* 2D, sesuaikan *settingan*-nya seperti gambar berikut, Centang pada *Freeze* *Rotation* Z.



### 8.3 *Rigidbody*

1. Lalu tambahkan komponen*Capsule* *Colider* pada karakter, lalu klik *icon* sebelah kanan edit *colider*.



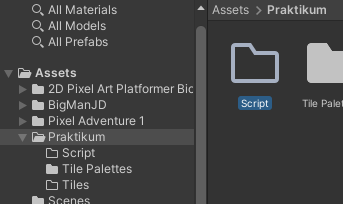
### 8.4 *Capsule* *Colider*

1. Lalu cocokan garis oval degan karakternya atau bisa di *input*-kan *Offset* X, Y dan juga *Size* X, Y nya.



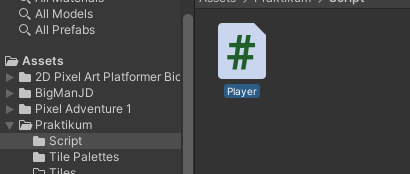
### 8.5 *Capsule* *Colider*

1. Buka *Folder* praktikum, lalu bikin *folder* baru bernama *Script.*



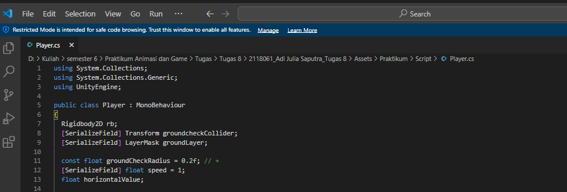
### 8.6 *Script Folder*

1. Masuk kedalam *folder Script*, lalu buat C# *Script*, beri nama *Player*.



### 8.7 *Script*

1. *Drag* & *drop* *script player* kedalam Hirarki *character*, lalu klik 2x pada *script player* maka akan masuk kedalam *text* *editor* seperti ini.



### 8.8 *Text Editor*

1. Masukan *source code* dibawah ini, pastikan nama *public class* harus sama dengan nama *file* yang dibuat.

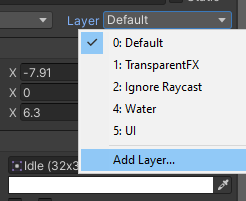
|  |
| --- |
| using System.Collections;  using System.Collections.Generic;  using UnityEngine;    public class Player : MonoBehaviour  {    Rigidbody2D rb;      [SerializeField] float speed = 1;    float horizontalValue;    bool facingRight;      private void Awake()    {      rb = GetComponent<Rigidbody2D>();    }      void Update ()    {      horizontalValue = Input.GetAxisRaw("Horizontal");    }      void FixedUpdate()    {      Move(horizontalValue);    }      void Move(float dir)    {      #region gerak kanan kiri      float xVal = dir \* speed \* 100 \* Time.fixedDeltaTime;      Vector2 targetVelocity = new Vector2(xVal,  rb.velocity.y);      rb.velocity = targetVelocity;  if (facingRight && dir < 0)  {  // ukuran player  transform.localScale = new Vector3(-1, 1, 1);  facingRight = false;  }  else if (!facingRight && dir > 0)  {  // ukuran player  transform.localScale = new Vector3(1, 1, 1);  facingRight = true;  }  #endregion  }  } |

1. Untuk mencoba *Source code* diatas berhasil, Tekan dikeyboard “a” atau “left arrow” untuk ke arah kiri, tekan “d” atau “right arrow” untuk ke arah kanan.



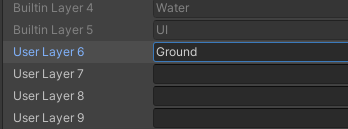
### 8.9 *Testing*

1. Untuk membuat *player* loncat menggunakan spasi, kita perlu membuat *GorundCheck* dengan cara, klik *Grid* pada *Hierarchy*, pergi ke *inspector*, pilih *Layer*, Klik *Add Layer*.



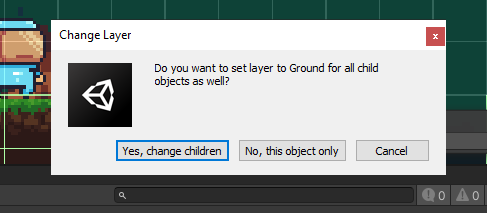
### 8.10 *Add Layer*

1. Lalu isi “*Ground*” pada *User* *Layer* 6.



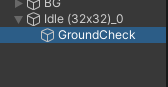
### 8.11 *Ground*

1. Ubah *Layer* pada *grid* menjadi *Ground*, jika muncul *pop up Change* *Layer*, klik *yes* saja*.*



### 8.12 *Pop up Change*

1. Klik kanan pada *character*, lalu *Create empty*, beri nama *GroundCheck.*



### 8.13 *GroundCheck*

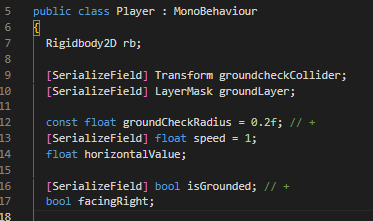
1. Klik pada Hirarki *GroundCheck*, lalu gunakan “*Move Tools*” untuk memindahkan ke bagian bawah *Player* seperti gambar berikut.



### 8.14 *GroundCheck*

1. Kembali ke *script* *Player* tambahkan *source code* seperti ini*.*

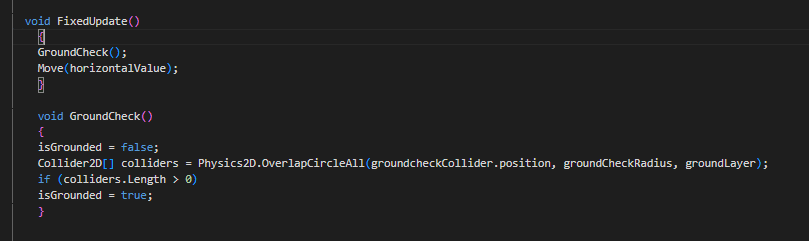
|  |
| --- |
| [SerializeField] Transform groundcheckCollider;  [SerializeField] LayerMask groundLayer;  const float groundCheckRadius = 0.2f; // +  [SerializeField] float speed = 1;  float horizontalValue;  [SerializeField] bool isGrounded; // +  bool facingRight; |



### 8.15 *Script Player*

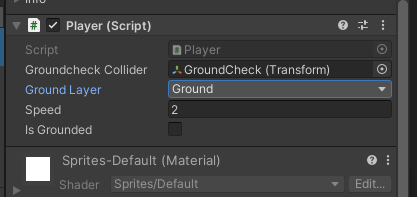
1. Buat *void ground check* dibawah *void fixedUpdate* & tambahkan *GorunCheck*(); pada *void fixedUpdate*.

|  |
| --- |
| void FixedUpdate()  {  GroundCheck();  Move(horizontalValue);  }  void GroundCheck()  {  isGrounded = false;  Collider2D[] colliders = Physics2D.OverlapCircleAll(groundcheckCollider.position, groundCheckRadius, groundLayer);  if (colliders.Length > 0)  isGrounded = true;  } |



### 8.16 *Script Fixed*

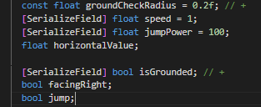
1. Klik *character*, lalu ke *inspector* ke *effect Player script* di bagian “*Goruncheck collider*” tekan *icon* bulat titik di sebelah kanan lalu pilih yang *GorundCheck* *Transform*, dan pada *Ground Layer* pilih *Ground*.



### 8.17 *GroundCheck*

1. Lalu untuk membuat *player* melompat tambahkan *script*berikut.

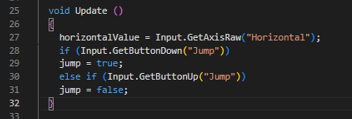
|  |
| --- |
| [SerializeField] float jumpPower = 100;  bool jump; |



### 8.18 *Jump Script*

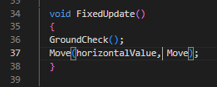
1. Tambahkan juga *script* berikut di bagian *void update*.

|  |
| --- |
| if (Input.GetButtonDown("Jump"))  jump = true;  else if (Input.GetButtonUp("Jump"))  jump = false; |



### 8.19 *Jump Script*

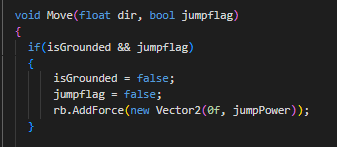
1. Tambahkan juga *jump* pada parameter *Move*.



### 8.20 *Move Script*

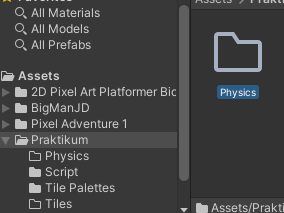
1. Tambahkan *script* berikut pada *void Move*.

|  |
| --- |
| bool jumpflag  if(isGrounded && jumpflag)  {  isGrounded = false;  jumpflag = false;  rb.AddForce(new Vector2(0f, jumpPower));  } |



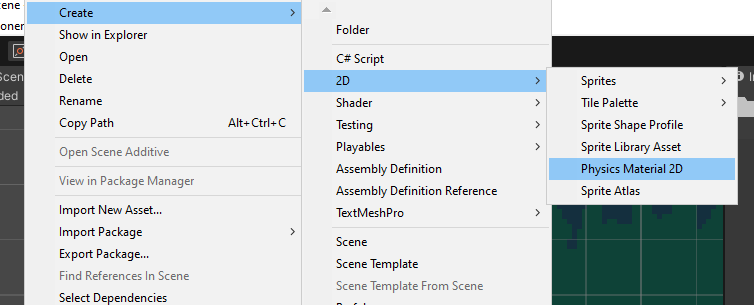
### 8.21 *Void Move*

1. Buat *folder* baru di praktikum bernama “*Physics*”.



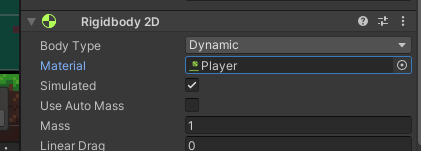
### 8.22 *Folder Physics*

1. Didalam *folder**Pyshics* *create* > 2d > *physical material* 2d, dan beri nama “*Player*”.



### 8.23 *Physical Material* 2d

1. Klik *Player* (*Physics Material 2D*), dibagian *menu* *inspector*, *friction* & *bounces* ubah menjadi 0. Klik *Hierarchy* pilih *layer* *character*, pada *Inspector* Cari *Rigidbody* 2D lalu klik *icon* untuk membuka *box* *select* *physhics* *material* 2d, lalu pilih *asset* *Player* yang sudah kita buat tadi.



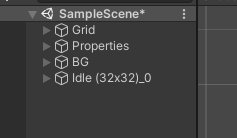
### 8.24 *Physical Material* 2d

1. Tekan *play*, maka *player* bisa melompat dengan menekan spasi



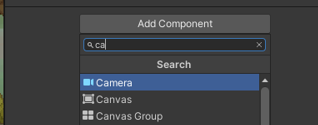
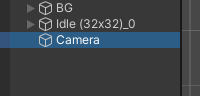
### 8.25 *Play*

1. **Camera Movement**
2. Hapus *Main Camera.*



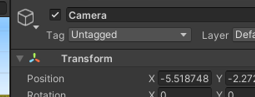
### 8.26 Hapus *Main Camera*

1. *Create Empty* pada *Hirarki*, dan Rename Menjadi *Camera*. Lalu *add component camera*.



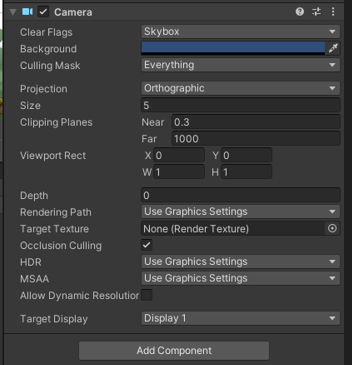
### 8.27 *Component Camera*

1. Pada *Hirarki* *Property* Ubah *Inspector* pada *tag camera* Menjadi *untaged*.



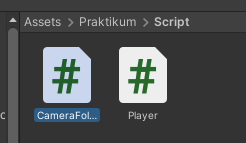
### 8.28 *Untagged*

1. Sesuaikan *Setting Layer Camer*a seperti gambar dibawah ini.



### 8.29 *Setting Camera*

1. Buat *file* *script* baru di *folder* *Script*dengan nama ”*CameraFollow*”.

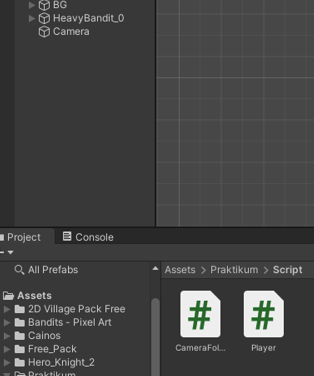


### 8.30 *Script Camera*

1. Lalu tuliskan *script* berikut ini.

|  |
| --- |
| using System.Collections;  using System.Collections.Generic;  using UnityEngine;  public class CameraFollow : MonoBehaviour  {      public float xMargin = 0.5f; // Margin di sumbu x      public float yMargin = 0.5f; // Margin di sumbu y      public float xSmooth = 4f;   // Kecepatan kamera mengikuti di sumbu x      public float ySmooth = 4f;   // Kecepatan kamera mengikuti di sumbu y      public Vector2 maxXAndY;     // Batas maksimum posisi kamera di sumbu x dan y      public Vector2 minXAndY;     // Batas minimum posisi kamera di sumbu x dan y      private Transform player;    // Referensi ke transformasi pemain      void Awake()      {          // Mencari GameObject dengan tag "Player" dan mendapatkan transformasinya          player = GameObject.FindGameObjectWithTag("Player")?.transform;          // Memastikan bahwa player ditemukan          if (player == null)          {              Debug.LogError("Pemain tidak ditemukan! Pastikan objek pemain memiliki tag 'Player'.");          }      }      bool CheckXMargin()      {          // Memeriksa apakah posisi x pemain sudah melewati margin yang ditentukan          return Mathf.Abs(transform.position.x - player.position.x) > xMargin;      }      bool CheckYMargin()      {          // Memeriksa apakah posisi y pemain sudah melewati margin yang ditentukan          return Mathf.Abs(transform.position.y - player.position.y) > yMargin;      }      void FixedUpdate()      {          if (player != null)          {              TrackPlayer();          }      }      void TrackPlayer()      {          // Mendapatkan posisi target awal (posisi kamera saat ini)          float targetX = transform.position.x;          float targetY = transform.position.y;          // Jika pemain melewati margin di sumbu x, kamera akan bergerak menuju posisi pemain          if (CheckXMargin())              targetX = Mathf.Lerp(transform.position.x, player.position.x, xSmooth \* Time.deltaTime);          // Jika pemain melewati margin di sumbu y, kamera akan bergerak menuju posisi pemain          if (CheckYMargin())              targetY = Mathf.Lerp(transform.position.y, player.position.y, ySmooth \* Time.deltaTime);          // Membatasi posisi target kamera agar tidak melewati batas maksimum dan minimum yang ditentukan          targetX = Mathf.Clamp(targetX, minXAndY.x, maxXAndY.x);          targetY = Mathf.Clamp(targetY, minXAndY.y, maxXAndY.y);          // Mengatur posisi kamera ke posisi target baru          transform.position = new Vector3(targetX, targetY, transform.position.z);          // Tambahkan pernyataan Debug.Log untuk memantau nilai targetX dan targetY          Debug.Log($"targetX: {targetX}, targetY: {targetY}");      }  } |

1. *Drag & drop script* *CameraFollow* Kedalam *Layer Camera*.



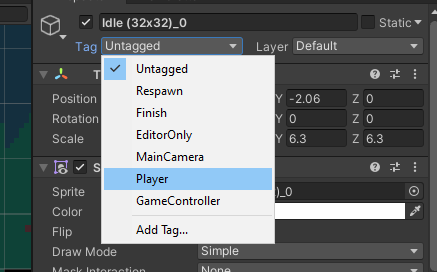
### 8.31 *Drag & Drop Script Camera*

1. Lalu klik pada *camera*, buka *inspector* *setting* menjadi berikut.



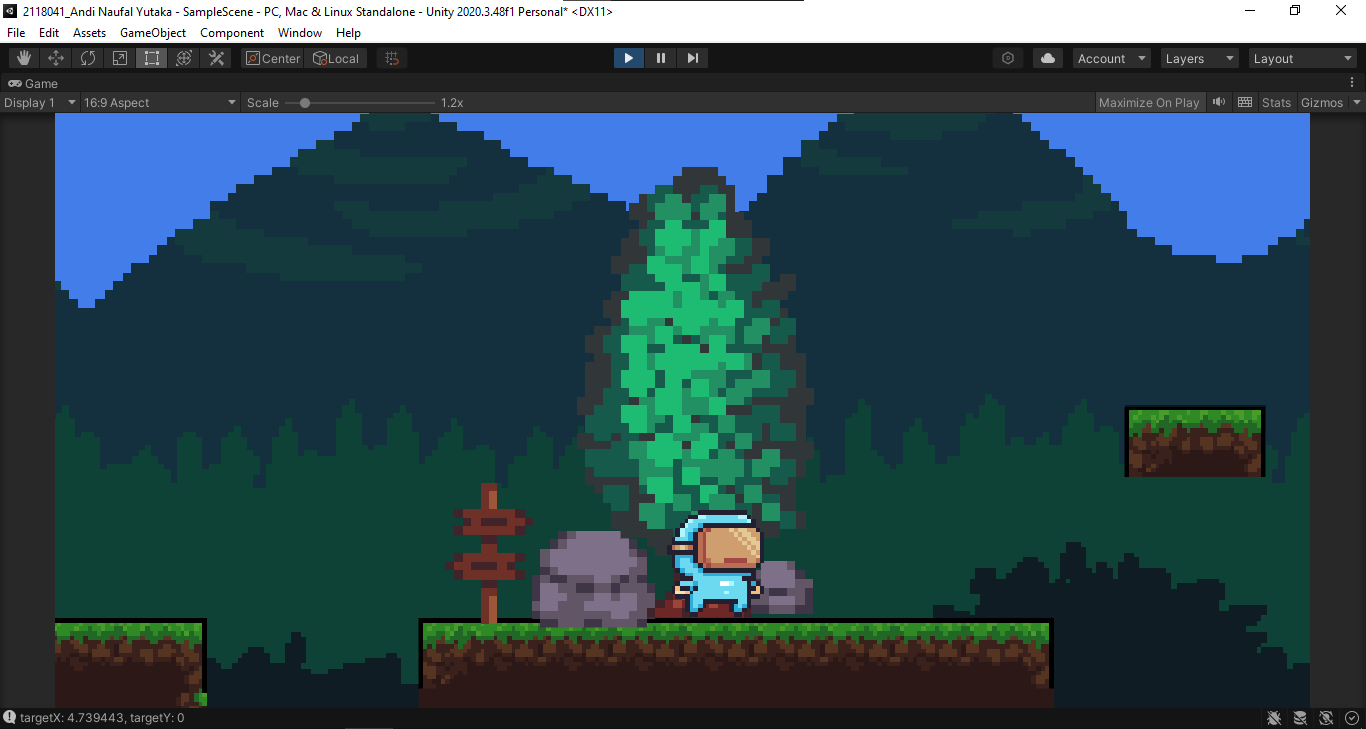
### 8.32 *Camera Setting*

1. Ubah *tag* di *character* *Untagged* menjadi ”Player”.



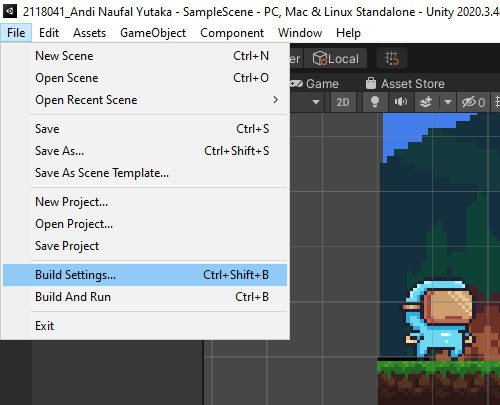
### 8.33 *Player*

1. Tekan *play* untuk menjalankan, maka sekarang kamera akan mengikuti pergerakan karakter.



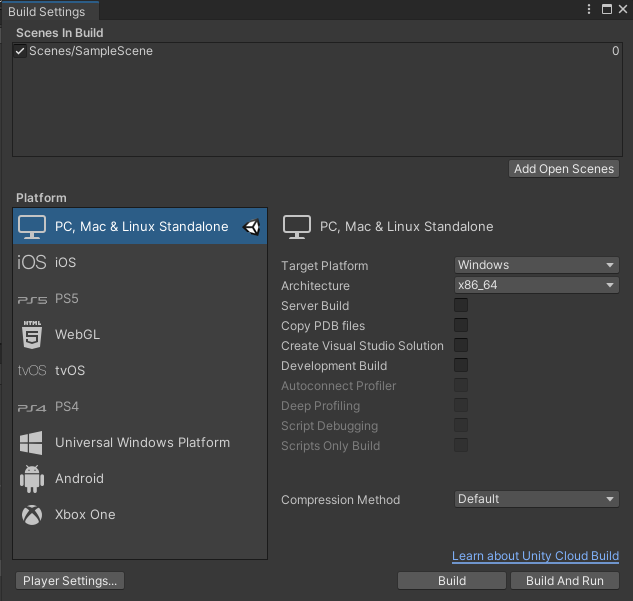
### 8.34 *Play*

1. **Render**
2. Pergi ke *menu* *File* kemudian pilih *Build Setting* (Ctrl + Shift + B).



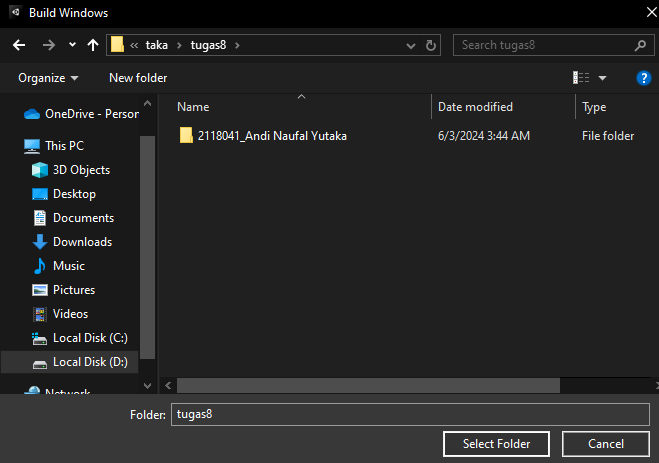
### 8.35 *Build Setting*

1. Pada *Setting Build* ini pilih PC, Mac & Linux, Tekan *Build*, pastikan pada *menu* *Scene in Build* berada pada *project* Tugas Kalian.



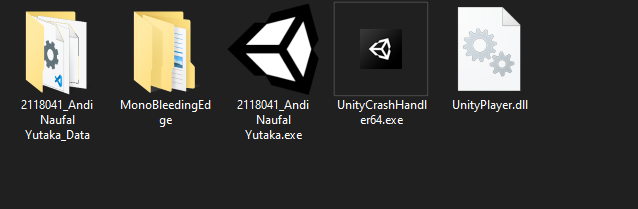
### 8.36 *Build Setting*

1. Pilih dimana *Project* disimpan, dan tunggu hasilnya.



### 8.37 Lokasi Simpan File

1. Hasil Render.



### 8.38 Hasil Render

1. Aplikasi dijalankan.



### 8.39 Aplikasi Dijalankan

1. **Kuis CameraFollow**

|  |
| --- |
| using System.Collections;  using System.Collections.Generic;  using UnityEngine;  public class CameraFollow : MonoBehaviour  {  [SerializeField] private Transform player;  void Update(){  transform.position = new Vector3(player.position.x, transform.position.y, transform.position.z);  }  } |

Penjelasan :

Kode `CameraFollow` membuat kamera mengikuti pemain dalam gerakan horizontal. Kelas ini mewarisi dari `MonoBehaviour`, memungkinkan skrip di-attach ke GameObject sebagai komponen. Variabel `player` bertipe `Transform` diatur melalui `SerializeField` agar bisa diset dari Inspector Unity. Metode `Update` dipanggil setiap frame, di mana posisi `x` kamera diatur ulang mengikuti posisi `x` pemain, sementara posisi `y` dan `z` kamera tetap konstan. Ini umum digunakan dalam permainan platformer untuk memastikan kamera mengikuti pemain secara horizontal.

Untuk membuat pergerakan kamera lebih halus, interpolasi linier (lerp) bisa digunakan. Misalnya, dengan menambahkan `smoothSpeed` dan menggunakan `Vector3.Lerp` untuk interpolasi posisi kamera secara halus menuju posisi yang diinginkan. Selain itu, jika diperlukan, kamera bisa diatur untuk mengikuti pemain di sumbu `y` juga, serta menambahkan batasan agar kamera tidak keluar dari area permainan.