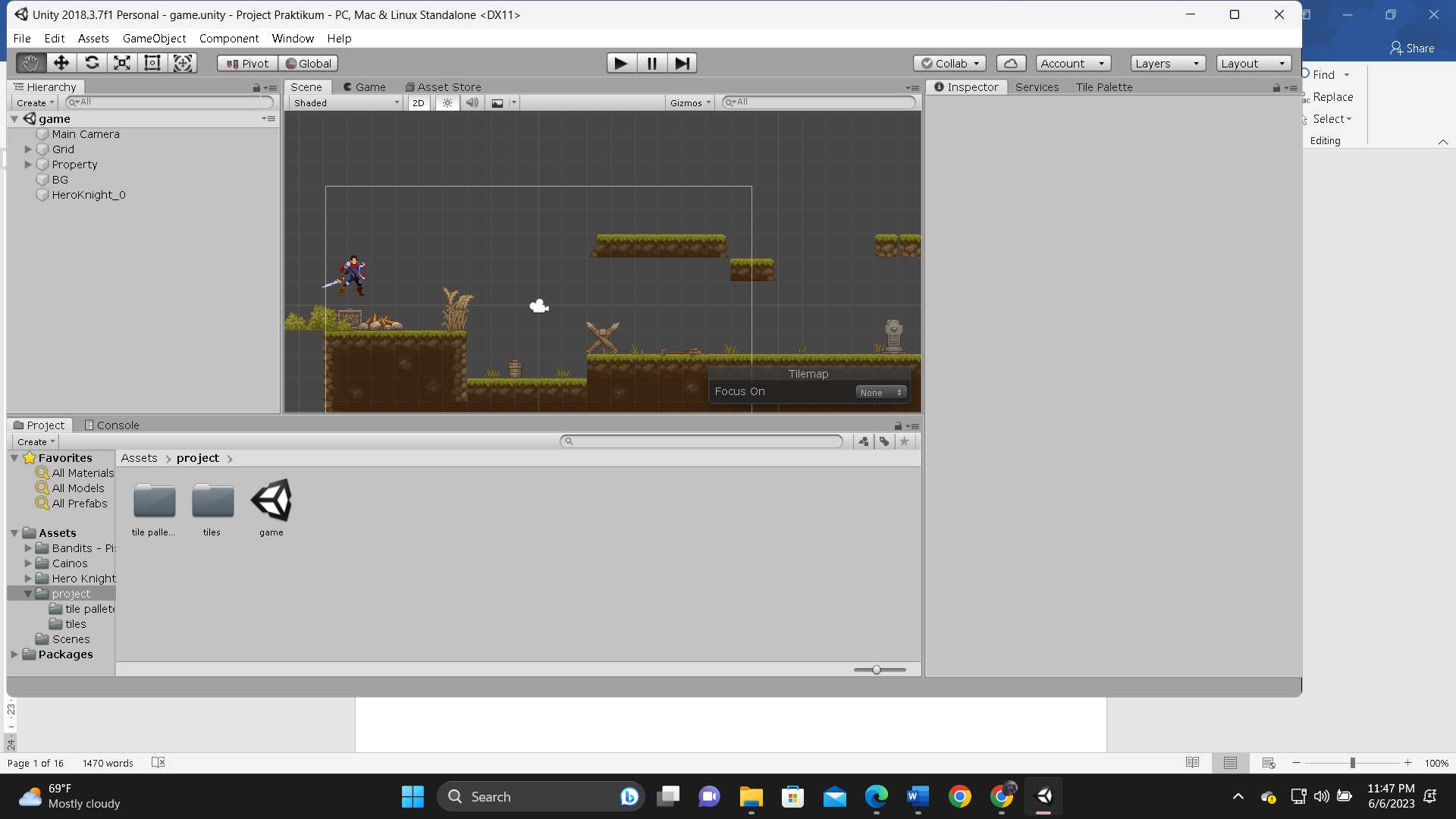
# PRAKTIKUM ANIMASI DAN GAME PERTEMUAN : 8 Character dan Camera Movement

|  |  |  |
| --- | --- | --- |
| NIM | : | 2118015 |
| NAMA | : | Martho Klemens Nahak |
| MATERI | : | Character dan Camera Movement |
| KELAS | : | B |
| TUGAS | : | 1. Membuat pergerakan karakter player2. Membuat pergerakan kamera mengikuti player |

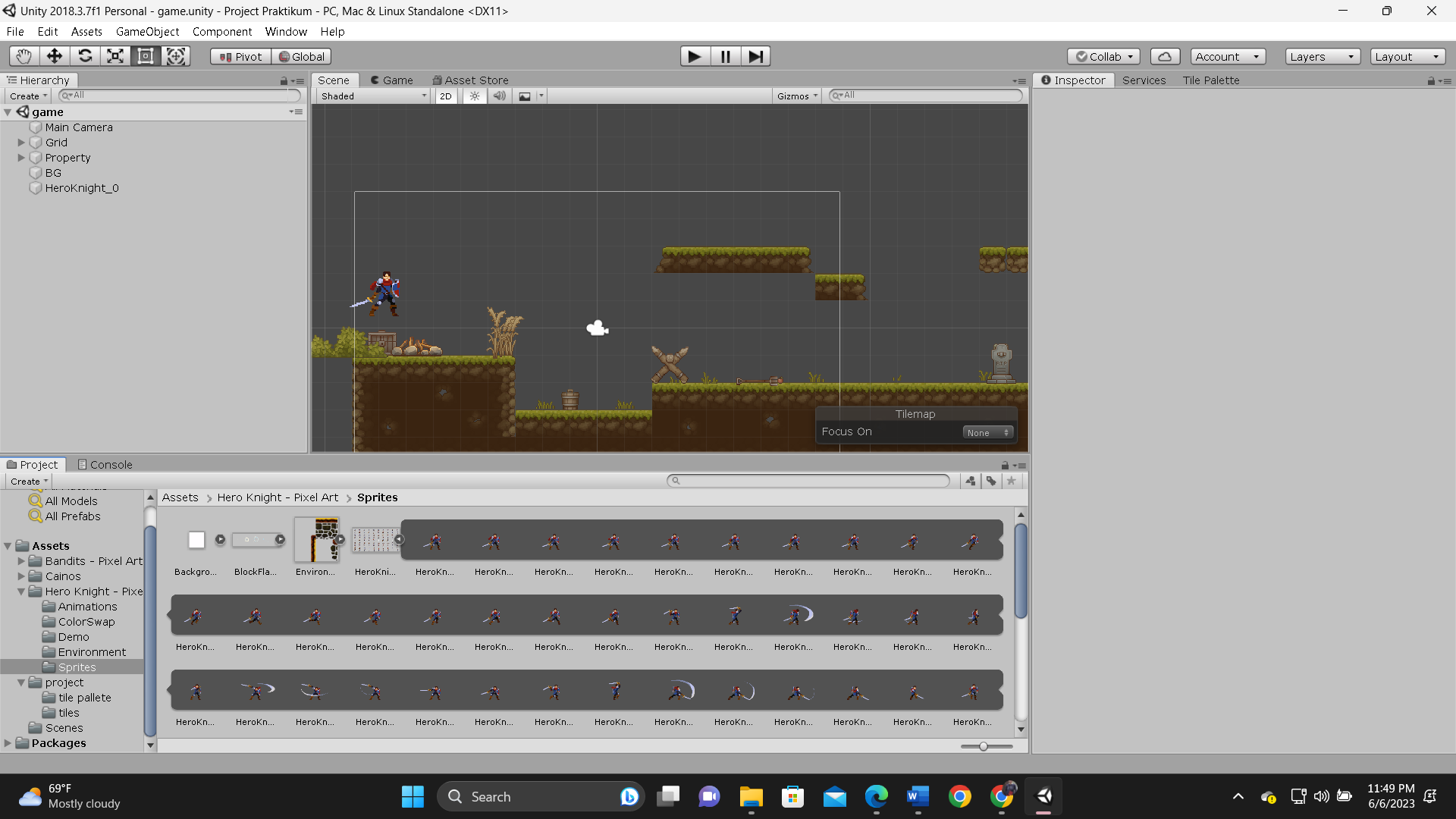
## Langkah-Langkah Membuat Tugas

1. **Moving *Character***
2. Buka file projek Unity sebelumnya pada bab 7 untuk digunakan Kembali.



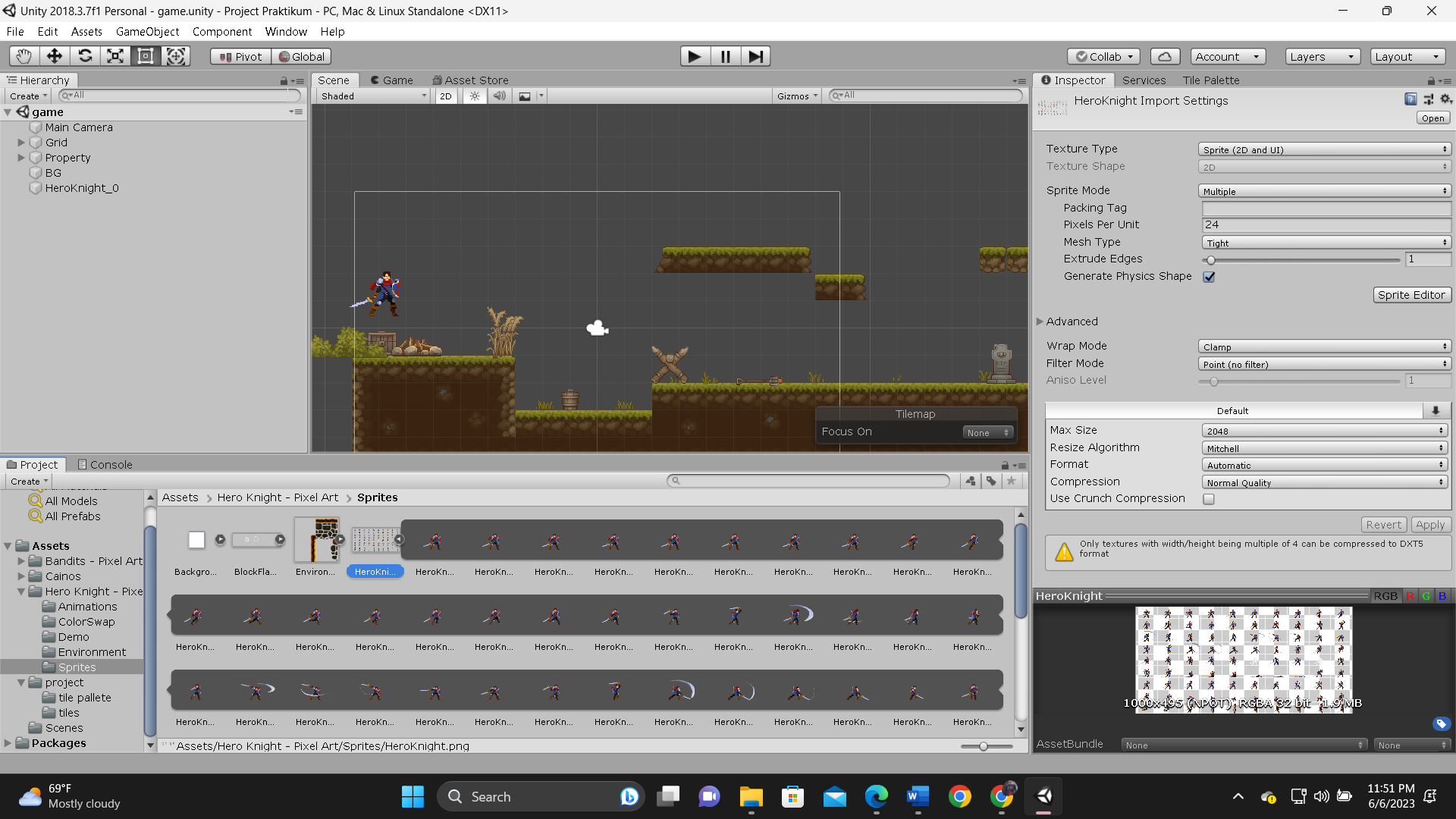
### Tampilan *project* Unity sebelumnya

1. Cari folder *Hero knight* lalu *Sprite* Sheet, lihat gambar berikut, kemudian klik bagian panah kecil dibagian kanan file tersebut, maka akan muncul banyak file *sprite* pada gambar dibawah ini.



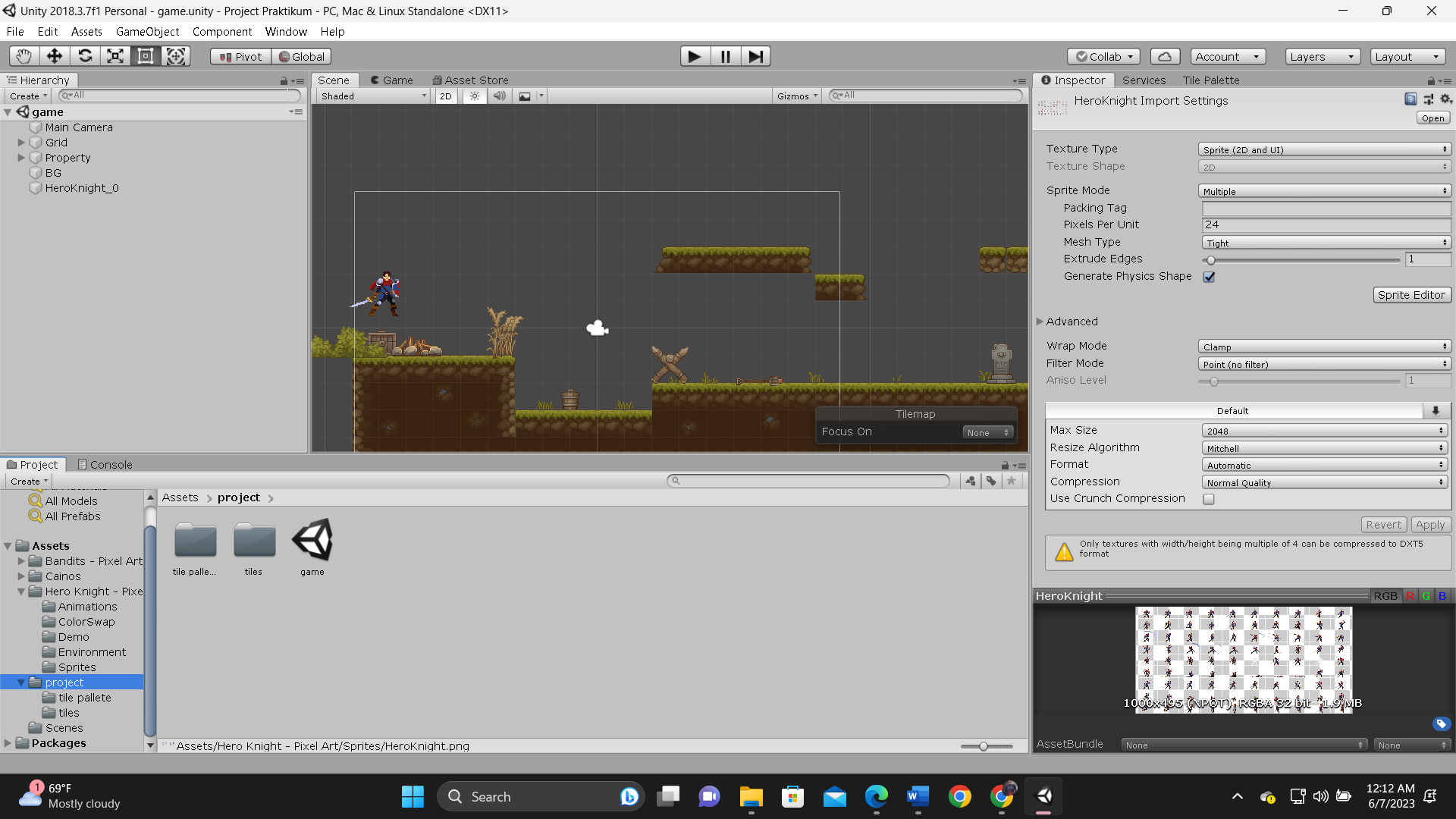
### Tampilan *sprite* sheet

1. Samakan pixel per unit pada karakter menjadi 24 seperti ukuran pixel per unit pada *asset* yang digunakan pada *tile*map. Seperti pada gambar dibawah ini



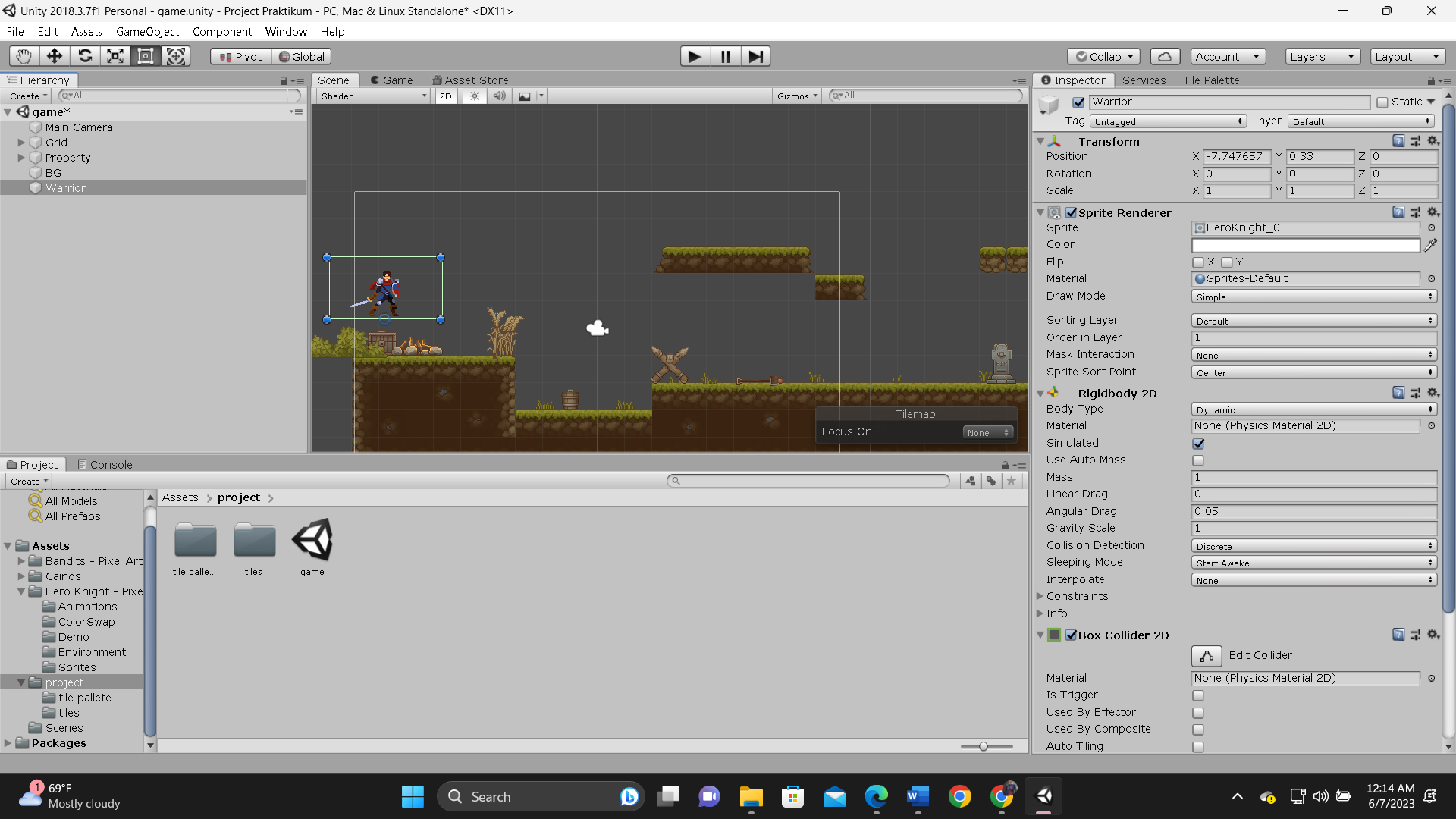
### Tampilan mengubah pixel menjadi 48

1. Pilih karakter hero seperti gambar dibawah ini, kemudian drag dan drop kedalam halaman kerja. Jika karakter tertutup oleh *asset* *tile*map, cobalah untuk mengganti nilai default Order in *Layer* dari 0 menjadi 1 pada *inspector*. Seperti pada gambar dibawah ini.



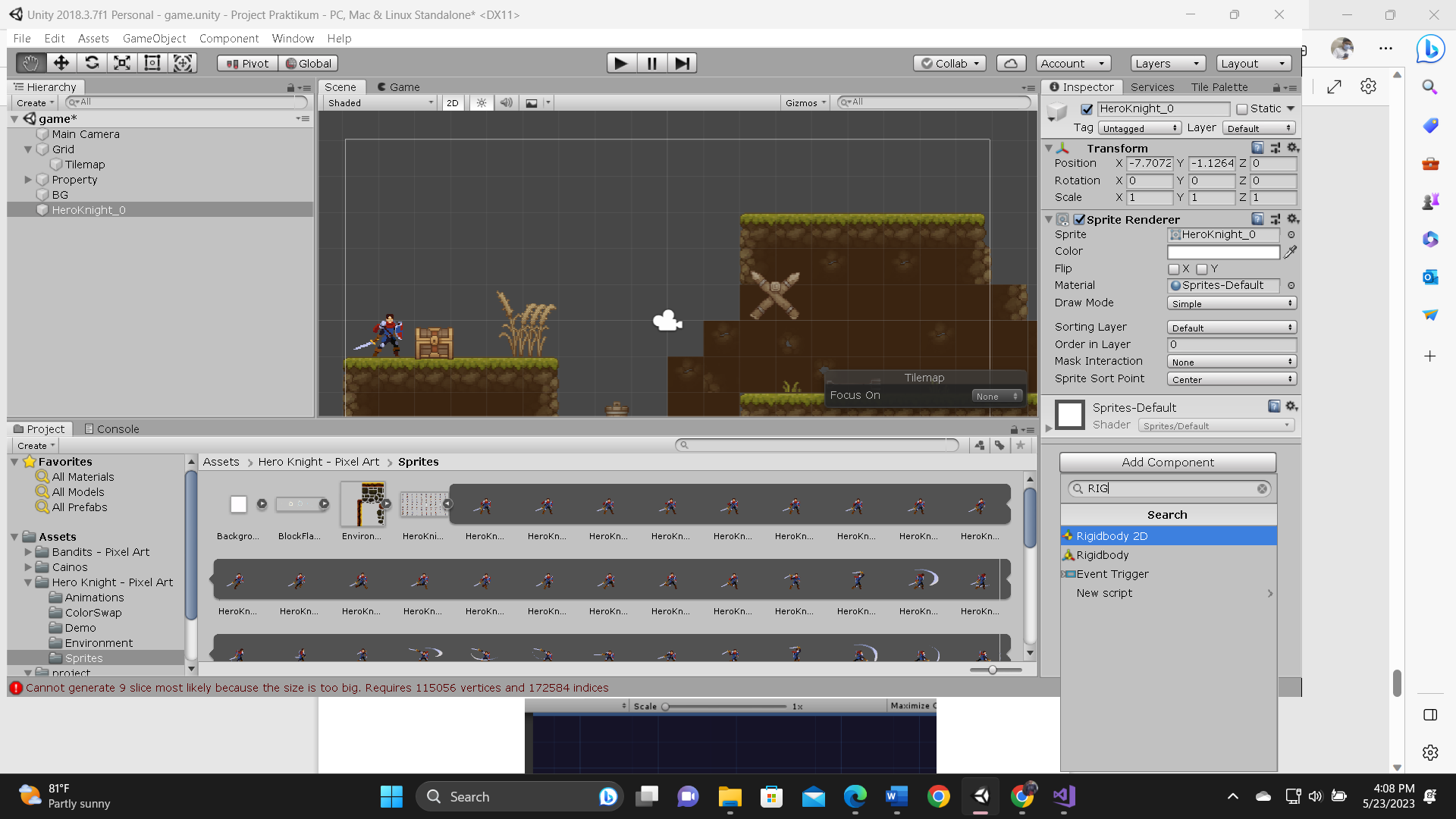
### Tampilan mengubah order in *layer* menjadi 1

1. Pada *Hierarchy*, ganti nama “*HeroKnight\_0*” menjadi “*Warrior*”, caranya klik kanan pada Run, pilih Rename. Seperti pada gambar dibawah ini



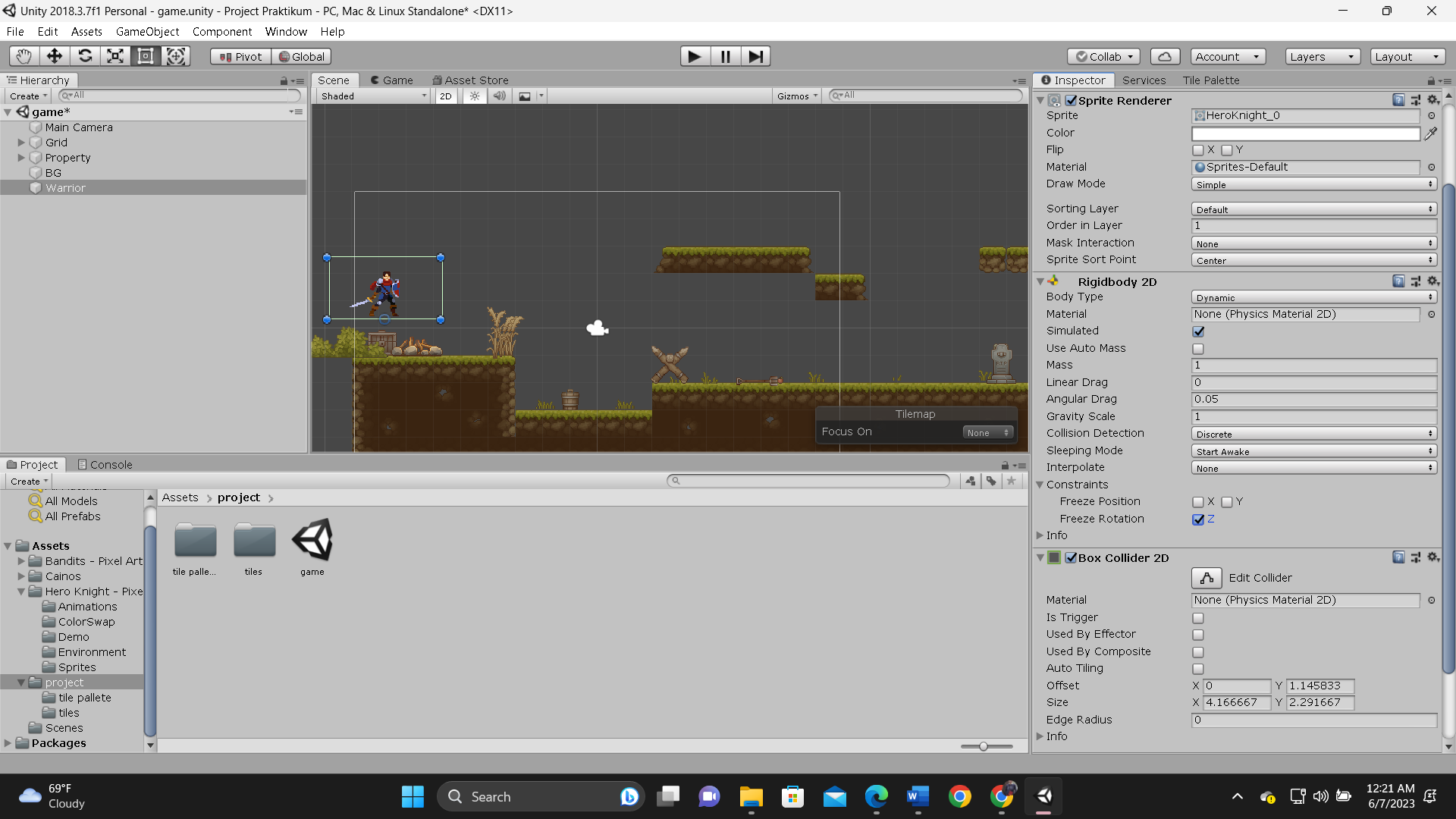
### Tampilan rename karakter

1. Klik karakter tersebut, pergi *Inspector*, klik *Add* *Component*, dan cari komp*one*n bernama *RigidBody* *2D*, klik jika ketemu. RigidBody ini berfungsi sebagai pemberian sifat fisik pada karakter. Seperti pada gambar dibawah ini



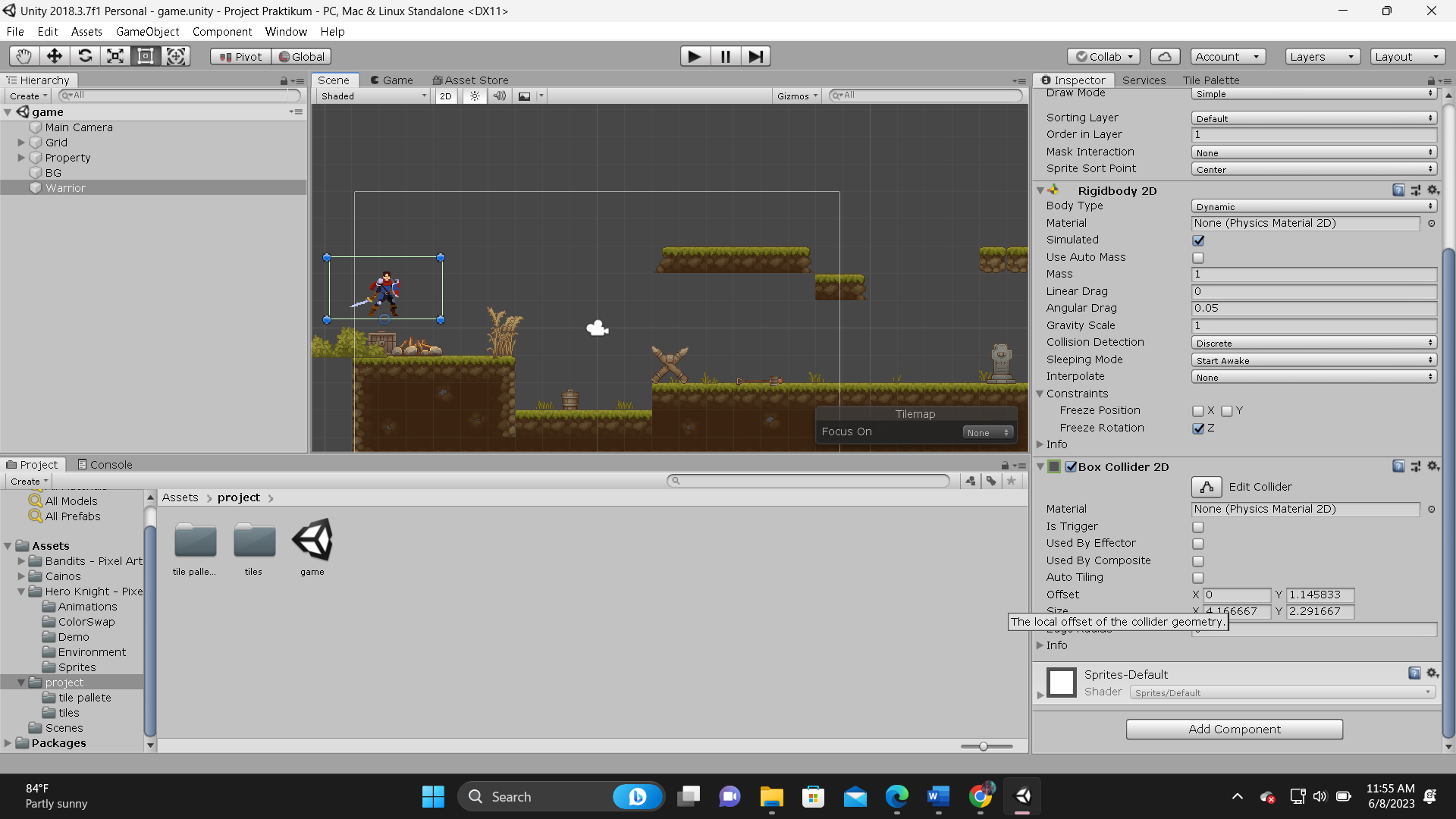
### Tampilan menambahkan komp*one*n RigidBody *2D*

1. Pada properties *Rigidbody* *2D*, klik Constraint, dan centang pada bagian Z, Freeze Rotation, kotak tersebut dicentang agar saat karakter menginjak tanah tidak terguling. Seperti pada gambar dibawah ini



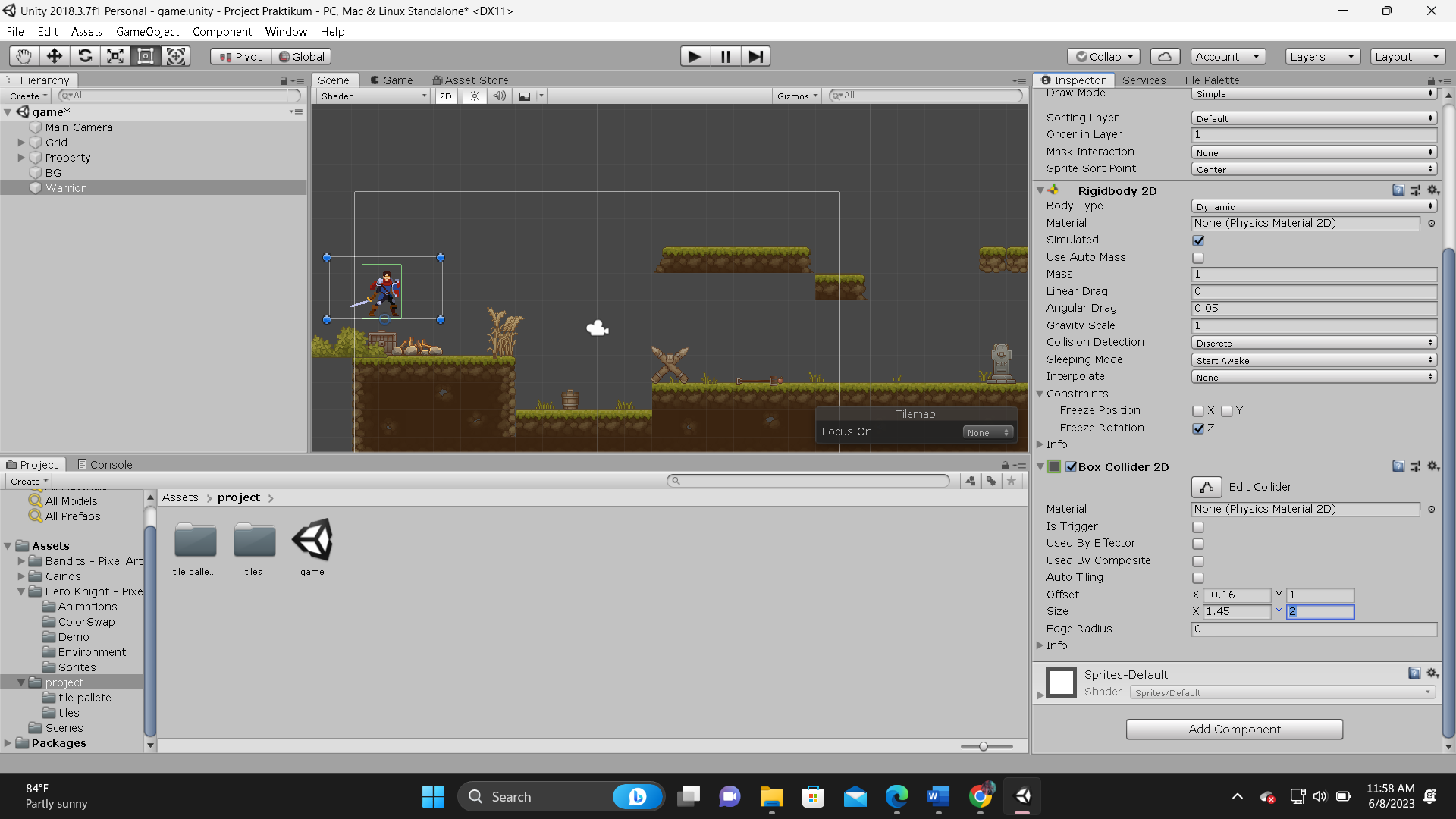
### Tampilan mengubah constraint z freeze rotation

1. Tambahkan lagi satu komp*one*n bernama *Capsule Collider* *2D*, jika sudah ketemu tinggal di klik. Collider adalah pembatas antar objek, jadi jika pada objek A dan objek B terdapat collider maka satu sama lain tidak dapat saling menembus, jika hanya salah satu memiliki collider makan dapat menembus antar objek. Seperti pada gambar dibawah ini



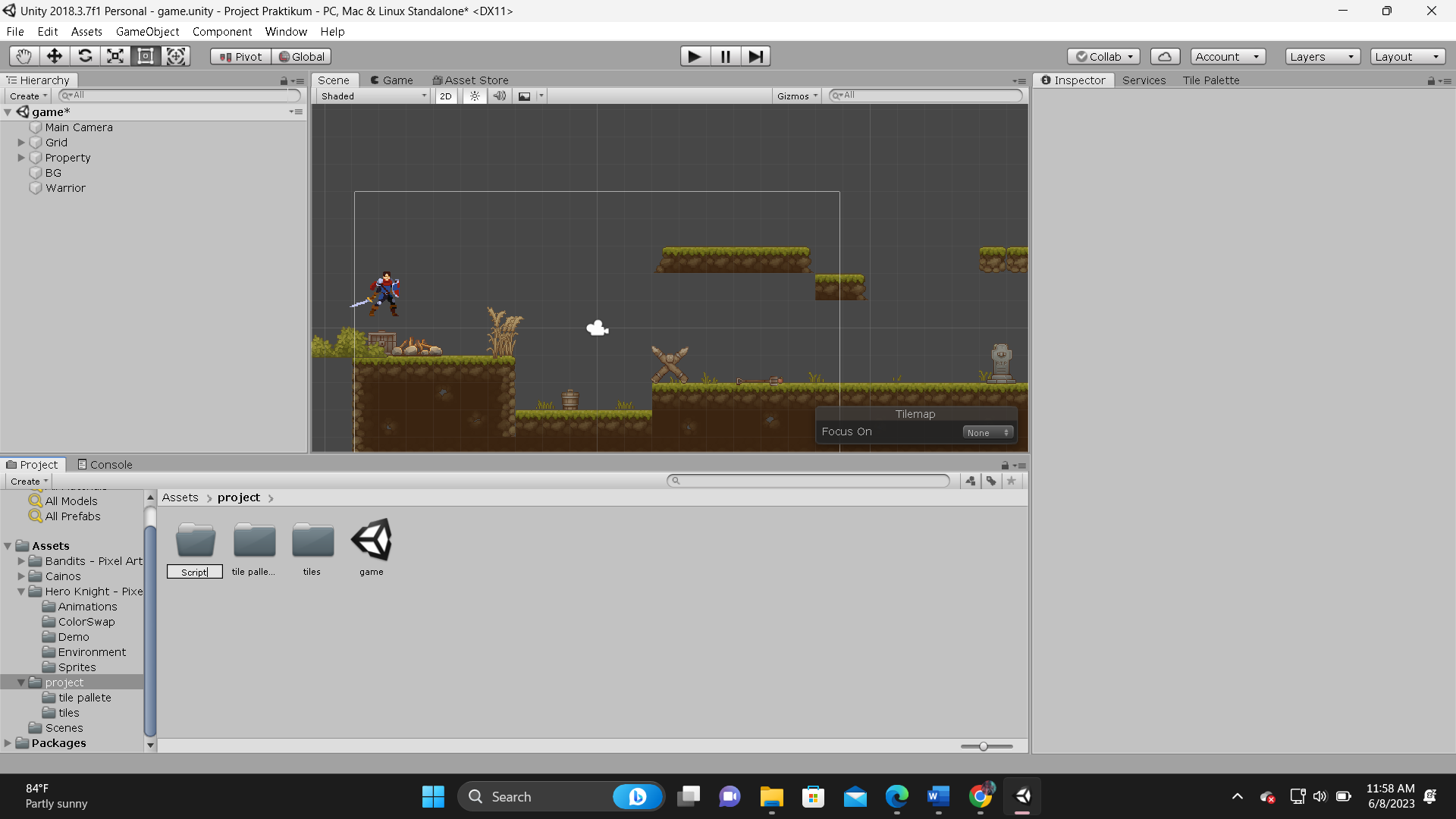
### Tampilan menambah *Capsule Collider* *2D*

1. Pada properties *Capsule Collider* *2D*, isikan nilai Offset dan Size seperti gambar dibawah ini.



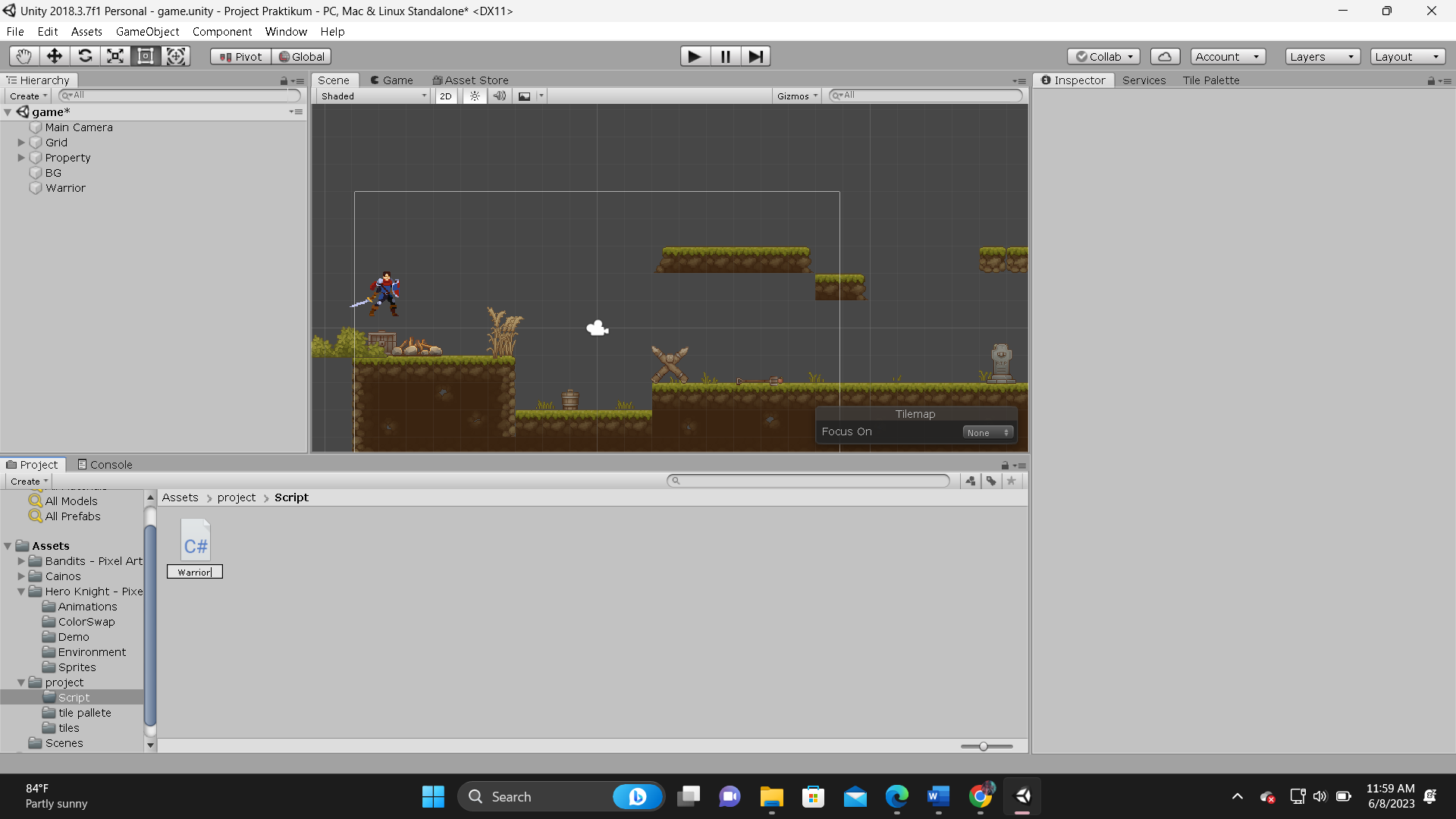
### Tampilan mengubah nilai offset

1. Buat folder baru bernama “*Script*” dalam folder praktikum.



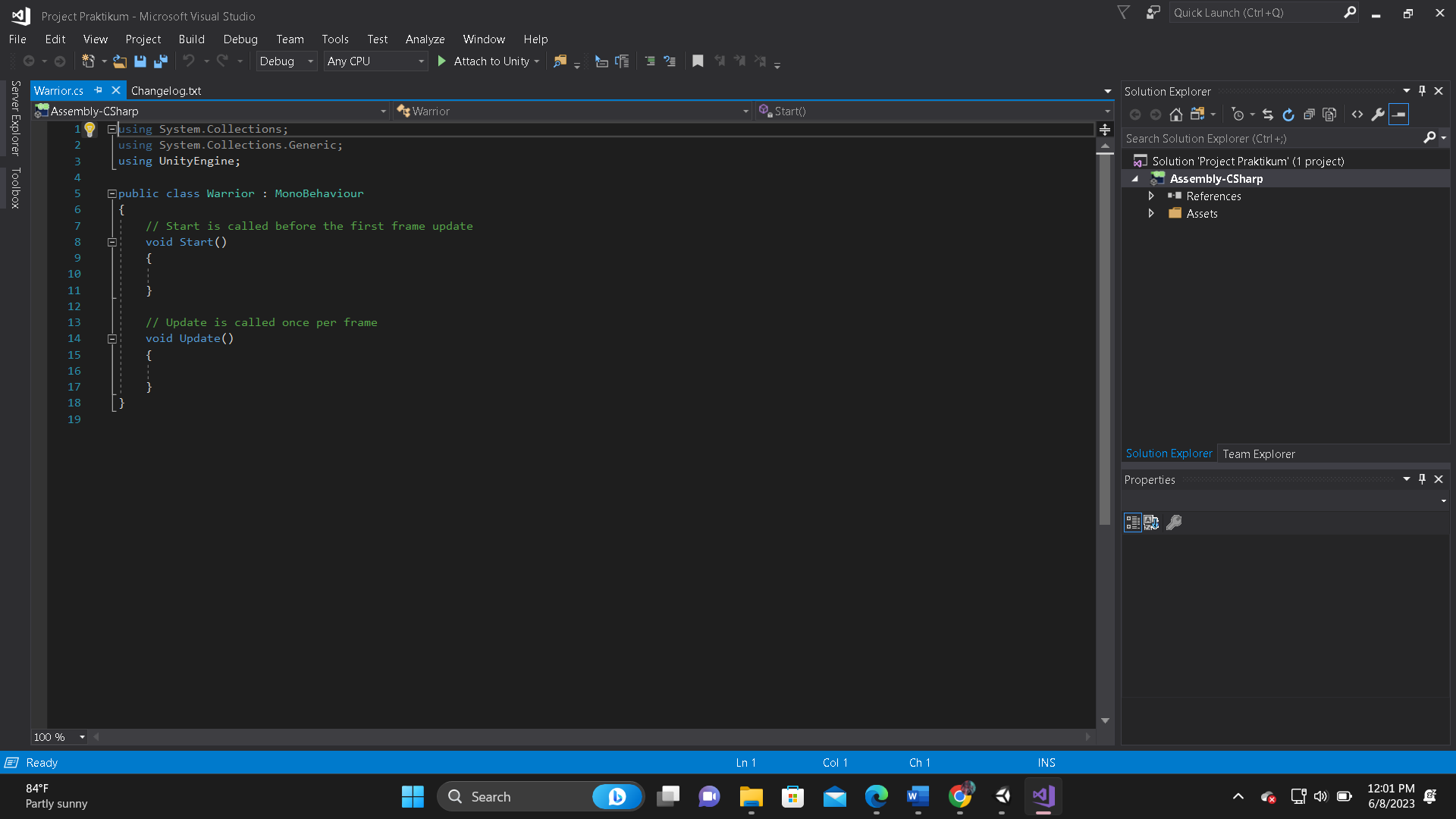
### Tampilan membuat folder *script*

1. Buat file *Script* baru, caranya klik kanan folder *Script*, pilih *Create* lalu C# *Script* dan namai file *script* tersebut dengan “*Warrior*”.



### Tampilan membuat file *script* C# dengan nama warrior

1. Drag file *script* tersebut kedalam *Game*Object “*Warrior*”. Kemudian klik 2X file *script* tersebut, maka akan dialihkan ke visual studio 2017.

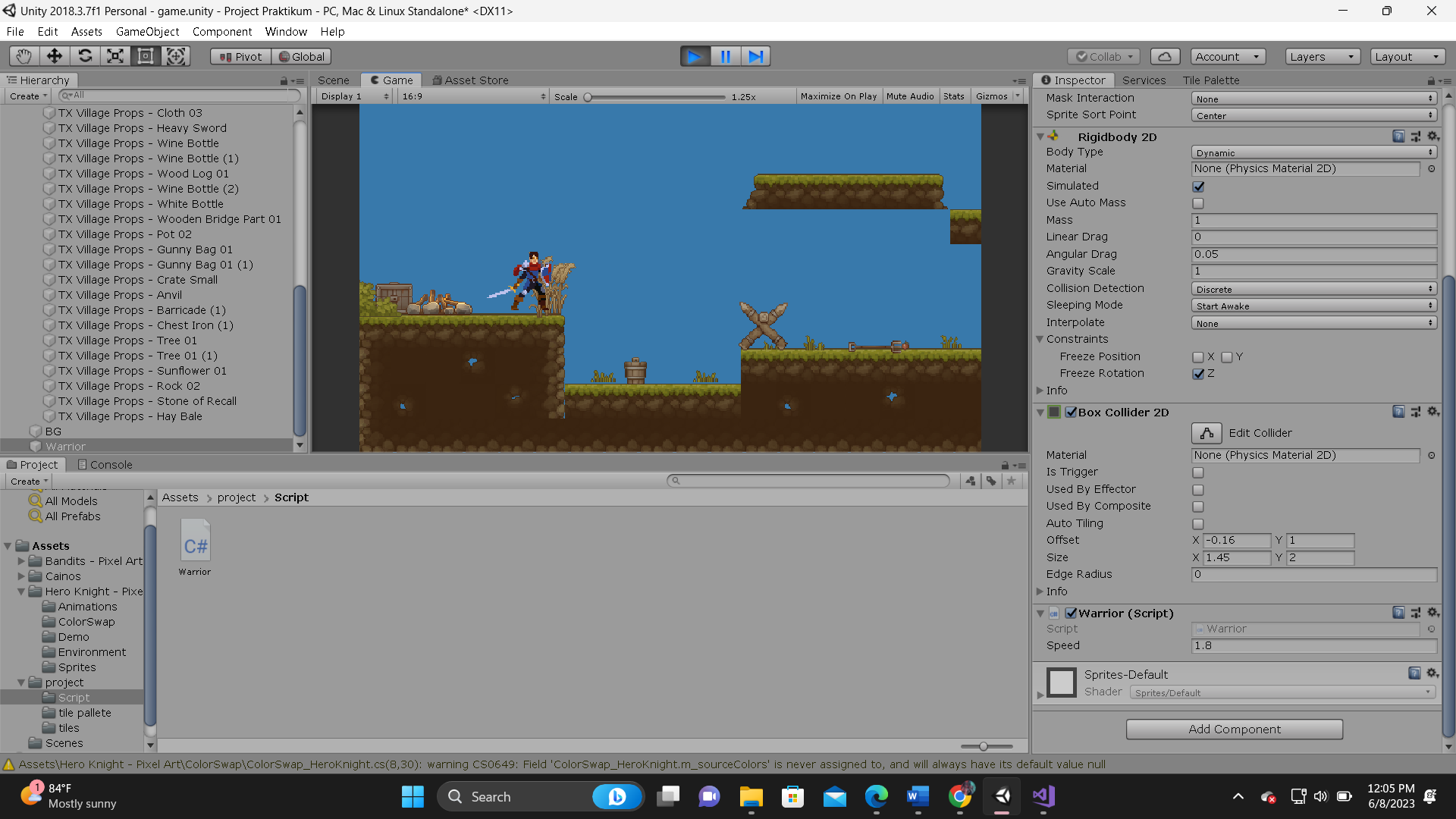


### Tampilan *workspace* visual studio 2017

1. Masukan source code dibawah ini, jangan lupa untuk mengganti Hero pada public class Hero dengan nama yang sama seperti nama file.

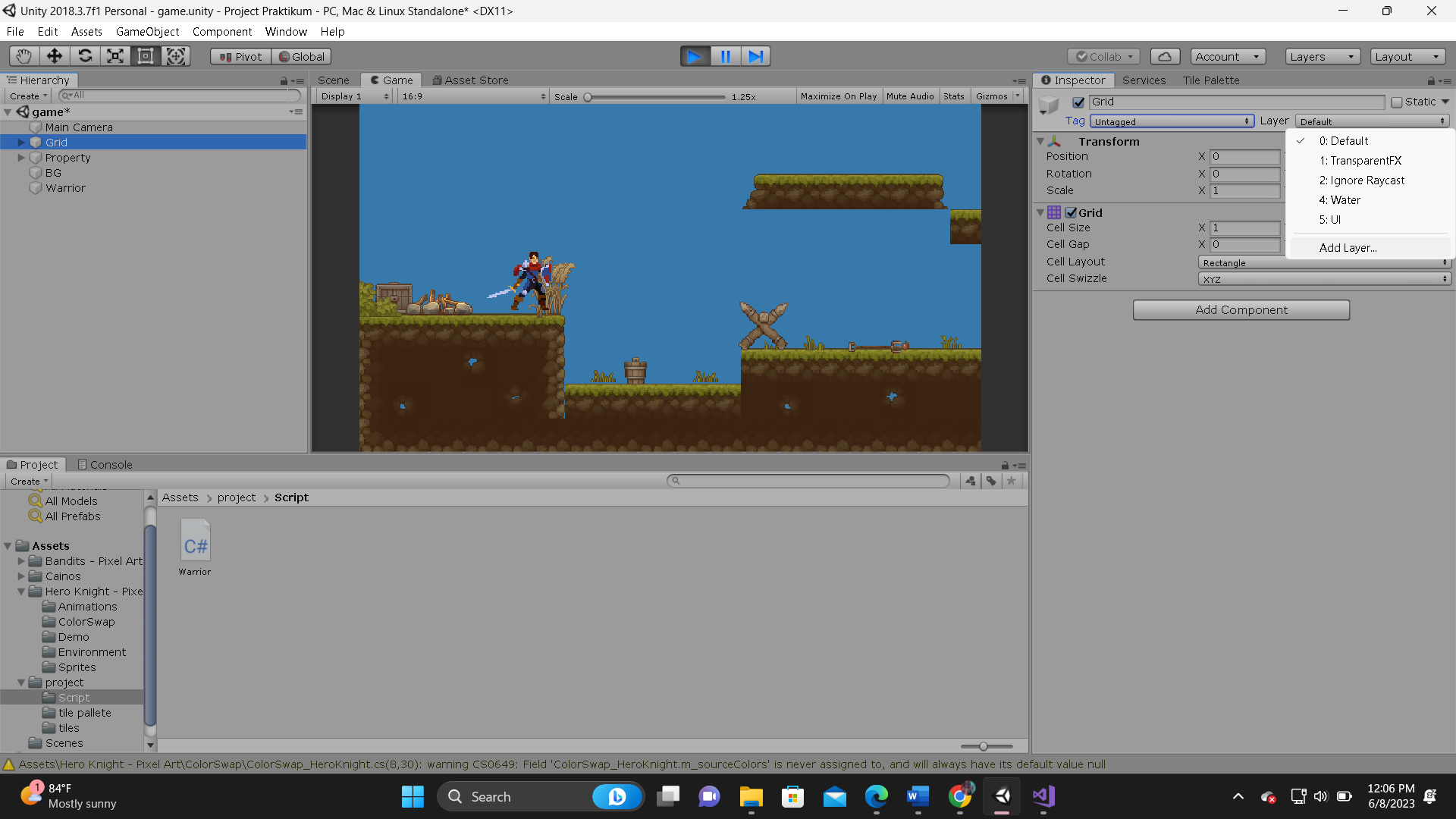
|  |
| --- |
| using System.Collections;  using System.Collections.Generic;  using UnityEngine;  public class Warrior : MonoBehaviour  {  Rigidbody2D rb;  [SerializeField] float speed = 1;  float horizontalValue;  bool facingRight;  private void Awake()  {  rb = GetComponent<Rigidbody2D>();  }  // Update is called once per frame  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)  {  transform.localScale = new Vector3(-1, 1, 1);  facingRight = false;  }  else if (!facingRight && dir > 0)  {  transform.localScale = new Vector3(1, 1, 1);  facingRight = true;  }  #endregion  }  }  } |

1. Jika diplay, karakter sudah bisa bergerak ke kanan dan ke kiri dengan menekan arah kanan kiri di keyboard. Seperti pada gambar dibawah ini



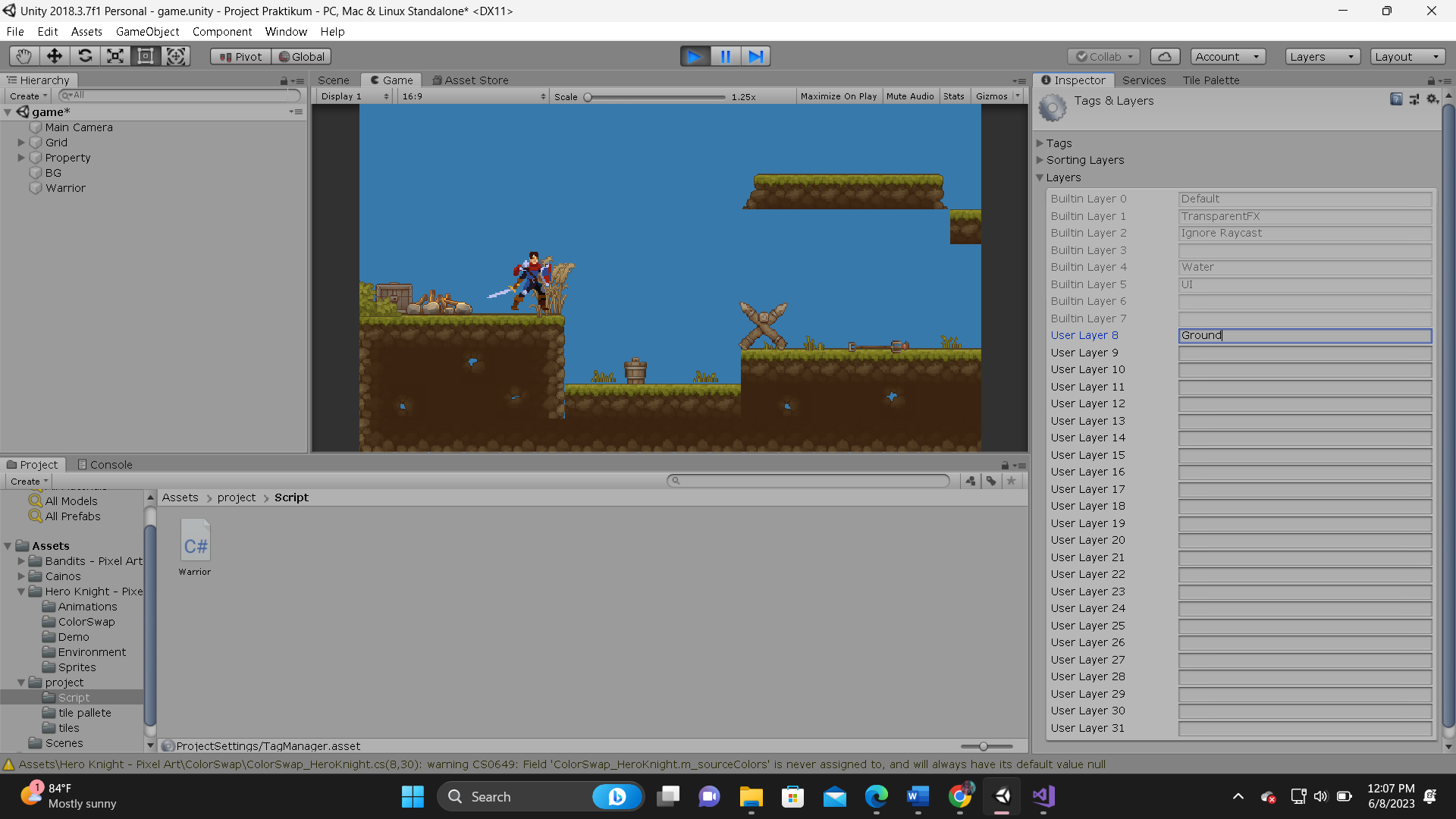
### Tampilan karakter sudah bisa bergerak ke kanan dan kiri

1. Untuk membuat Ground Check, klik Grid pada *Hierarchy*, pergi ke *Inspector*, pilih Layer, klik *Add* Layer



### Tampilan membuat *layer* baru pada grid

1. Isikan User *Layer* 8 dengan nama Ground. Seperti pada gambar dibawah ini



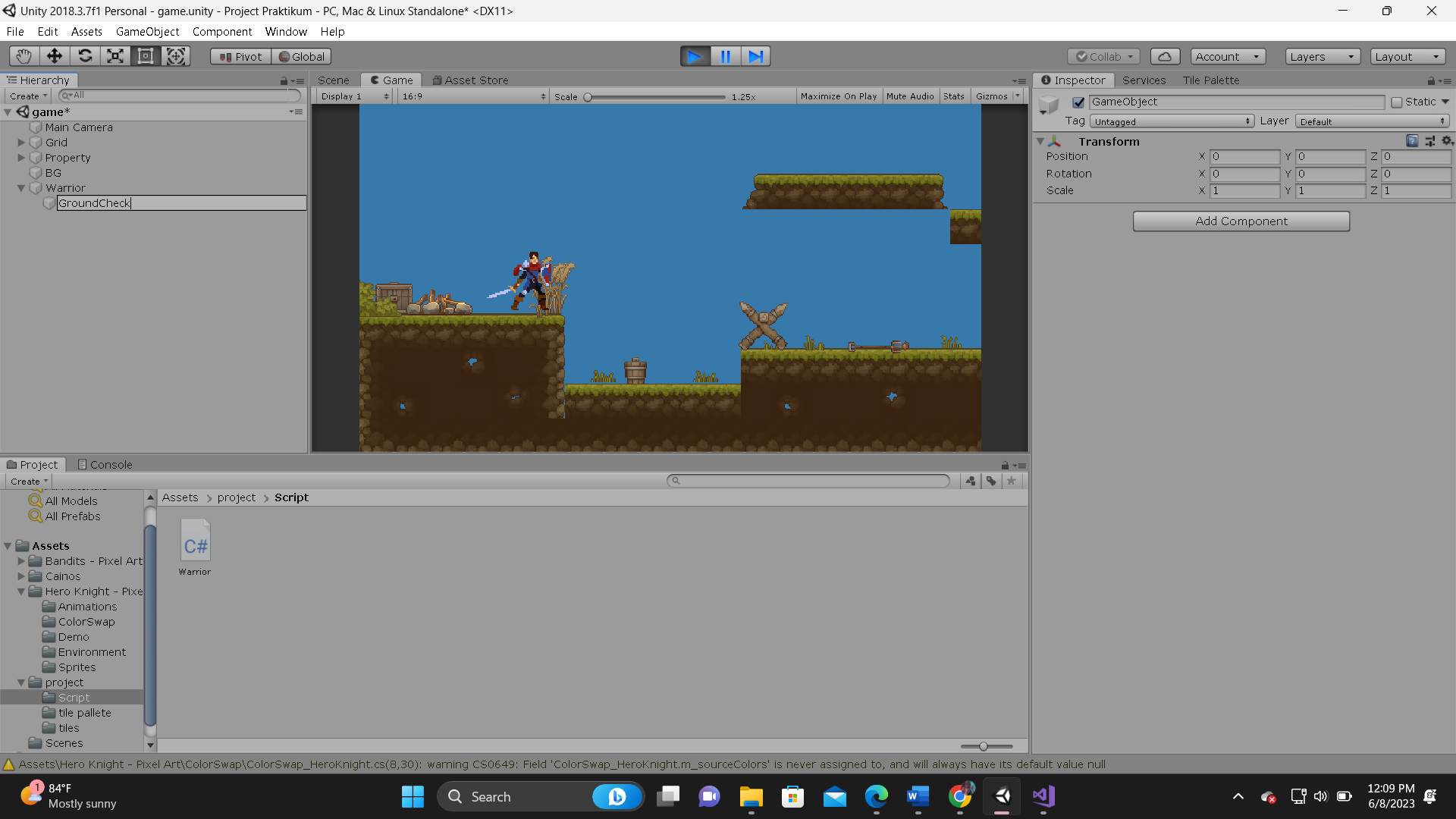
### Tampilan mengubah *layer* 8 menjadi Ground

1. Ganti *Layer* Default menjadi Ground, klik Yes, change children.



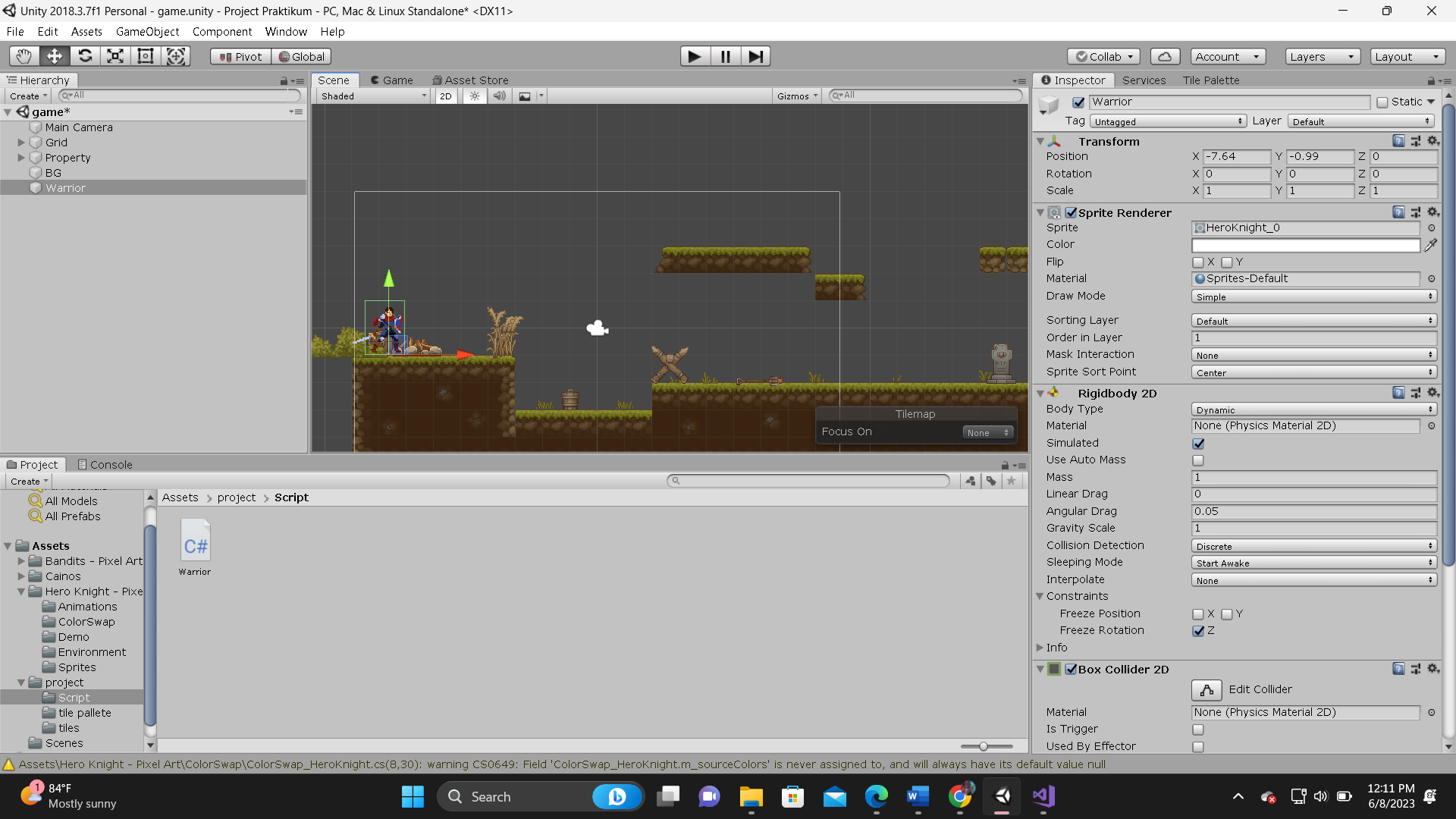
### Tampilan mengubah *layer* default menjadi ground

1. Klik kanan *Warrior* kemudian pilih *Create* *Empty* dan ubah namanya menjadi GroundCheck.



### Tampilan menambahkan GroundCheck pada warrior

1. Klik GroundCheck, kemudian gunakan Move *Tool* untuk untuk menggeser panah berwarna hijau-merah pada objek karakter seperti gambar dibawah ini.



### Tampilan menggeser panah

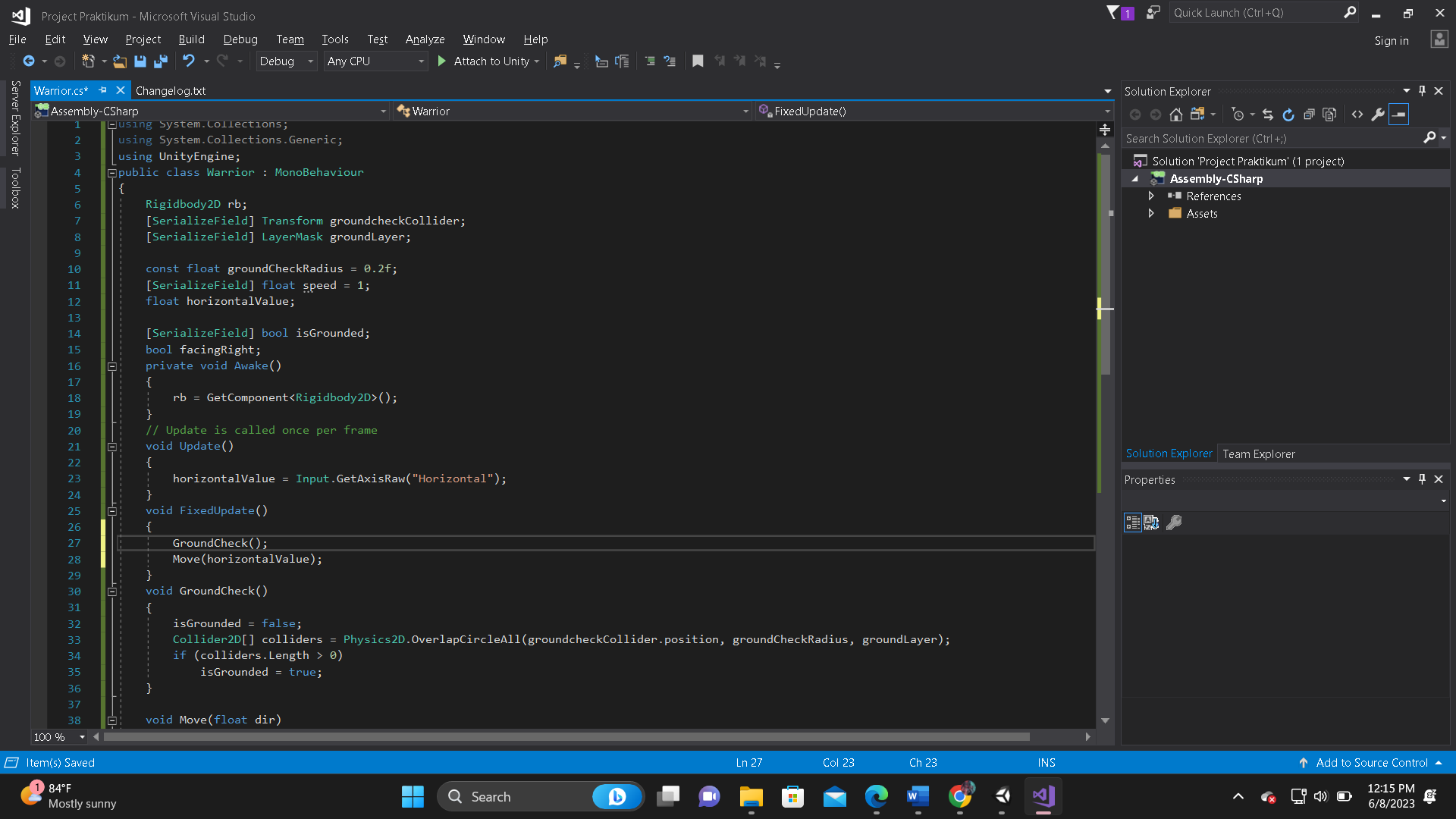
1. Kembali ke visual studio, tambahkan baris kode seperti gambar dibawah ini.

|  |
| --- |
| [SerializeField] Transform groundcheckCollider;  [SerializeField] LayerMask groundLayer;  const float groundCheckRadius = 0.2f;  [SerializeField] bool isGrounded; |

1. Buat void baru dibawah void FixedUpdate.

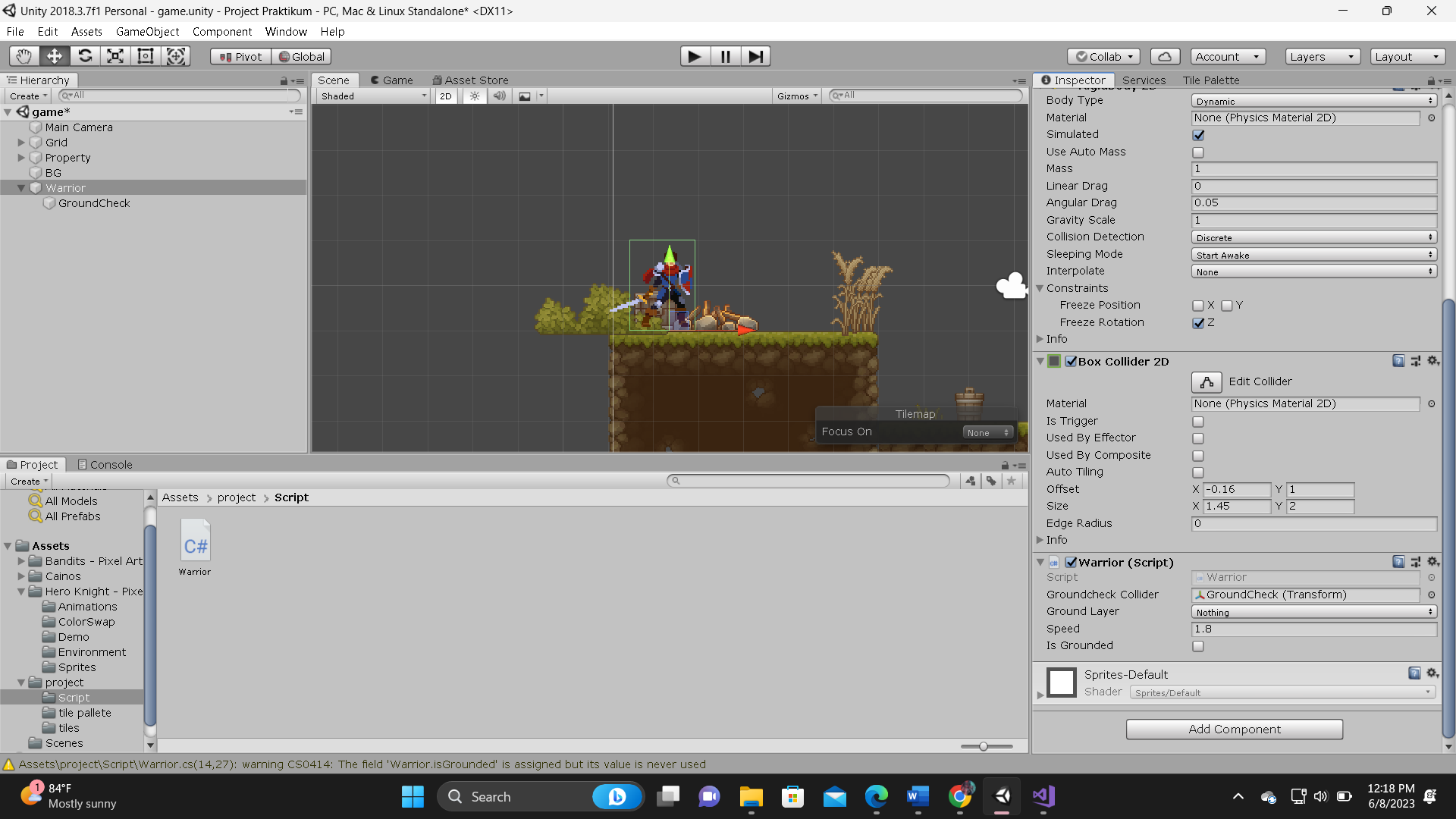
|  |
| --- |
| void GroundCheck()  {  isGrounded = false;  Collider*2D*[] colliders =  Physics*2D*.OverlapCircleAll(groundcheckCollider.position, groundCheckRadius,  groundLayer);  if (colliders.Length lalu 0)  isGrounded = true;  } |

1. Tambahkan GroundCheck(); dalam void FixedUpdate.



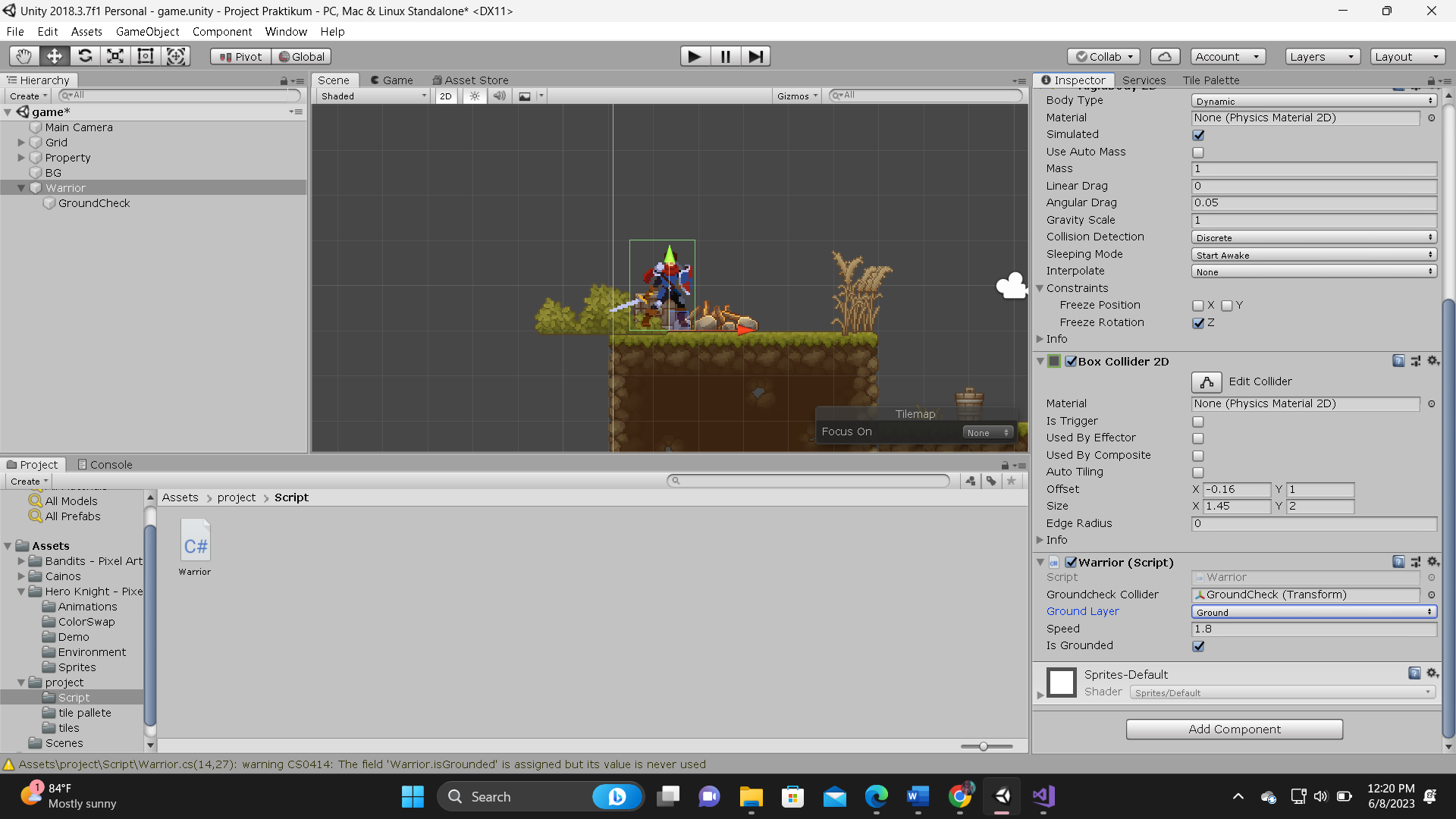
### Tampilan menambahkan fungsi GroundCheck

1. Kembali ke Unity Editor, klik *Warrior*, kemudian klik dan drag GroundCheck arahkan kedalam Groundcheck Collider yang ada pada *Inspector*.



### Tampilan drag GroundCheck collider kedalam *inspector*

1. Sesuaikan juga opsi yang lainnya.



### Tampilan opsi yang disesuaikan

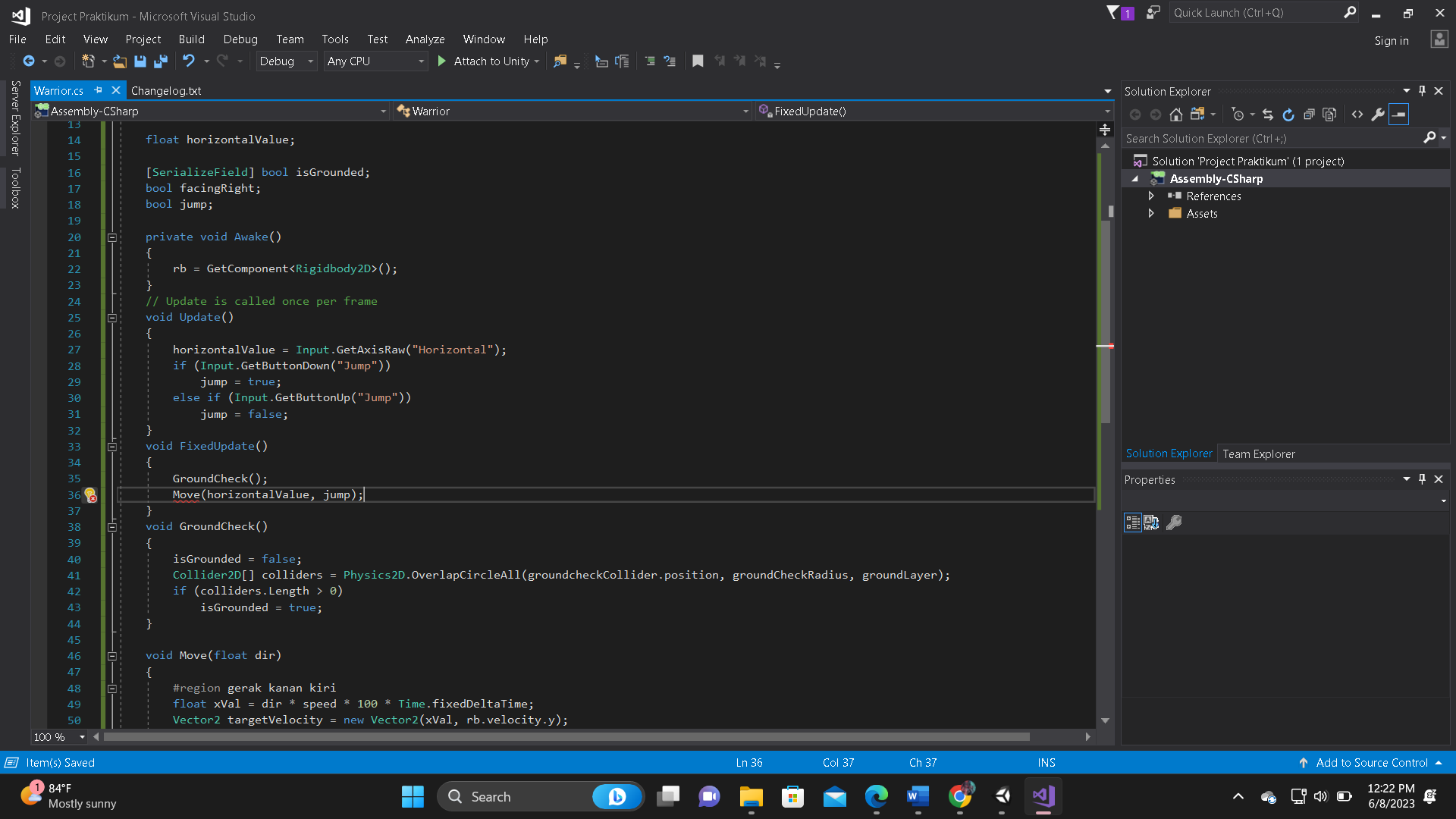
1. Untuk membuat sebuah mekanisme melompat, tambahkan baris kode seperti dibawah ini.

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

1. Tambahkan kode dibawah ini di dalam void update.

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

1. Pada void FixedUpdate tambahkan seperti yang dikotak merah.



### Tampilan menambahkan parameter jump

1. Tambahkan baris kode dibawah ini, letakkan dalam void Move.

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

1. Buat folder baru bernama “Physics” dalam folder Praktikum. Klik kanan folder “Physics” pilih *Create* lalu Physics Material *2D* dan beri nama “*Warrior*”.



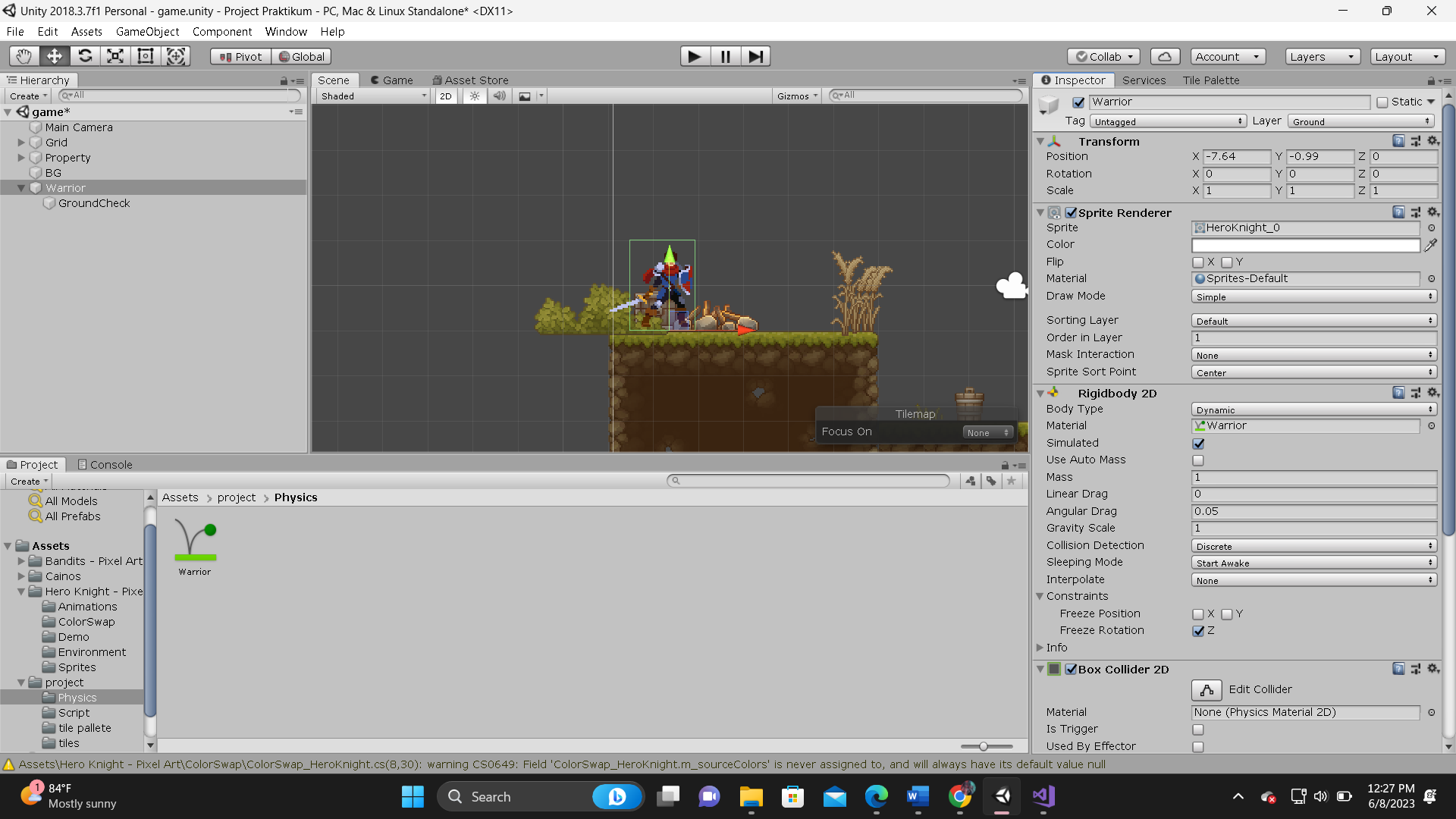
### Tampilan menambahkan folder physics

1. Klik file tersebut, dan pergi ke *Inspector*, ubah nilai Friction dan Bouncies menjadi 0. Seperti pada gambar dibawah ini



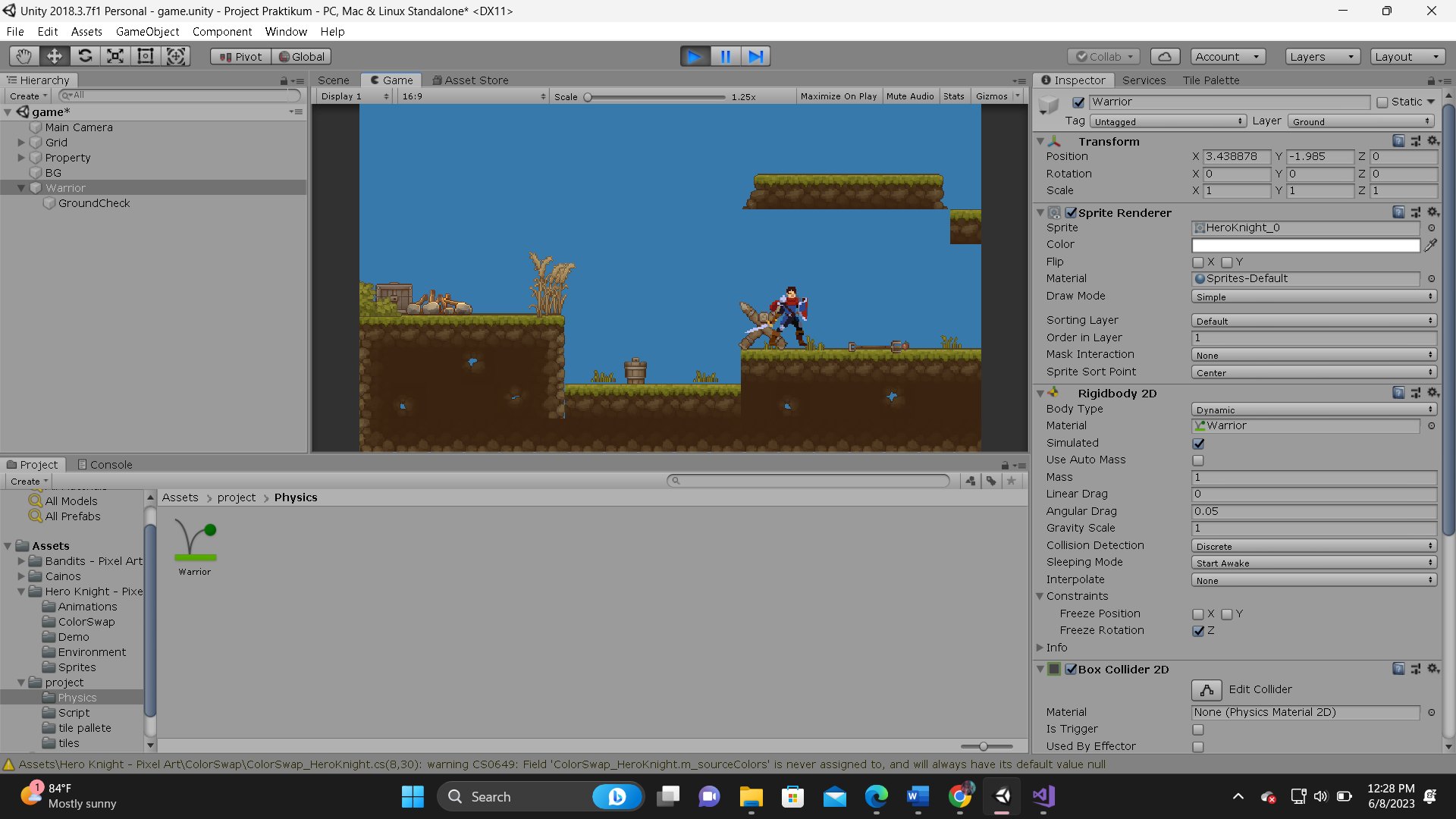
### Mengubah nilai Friction dan Bouncies menjadi 0

1. Klik objek Karakter, kemudian drag file Physics *2D* tadi kedalam Material yang ada pada *Rigidbody* *2D*.



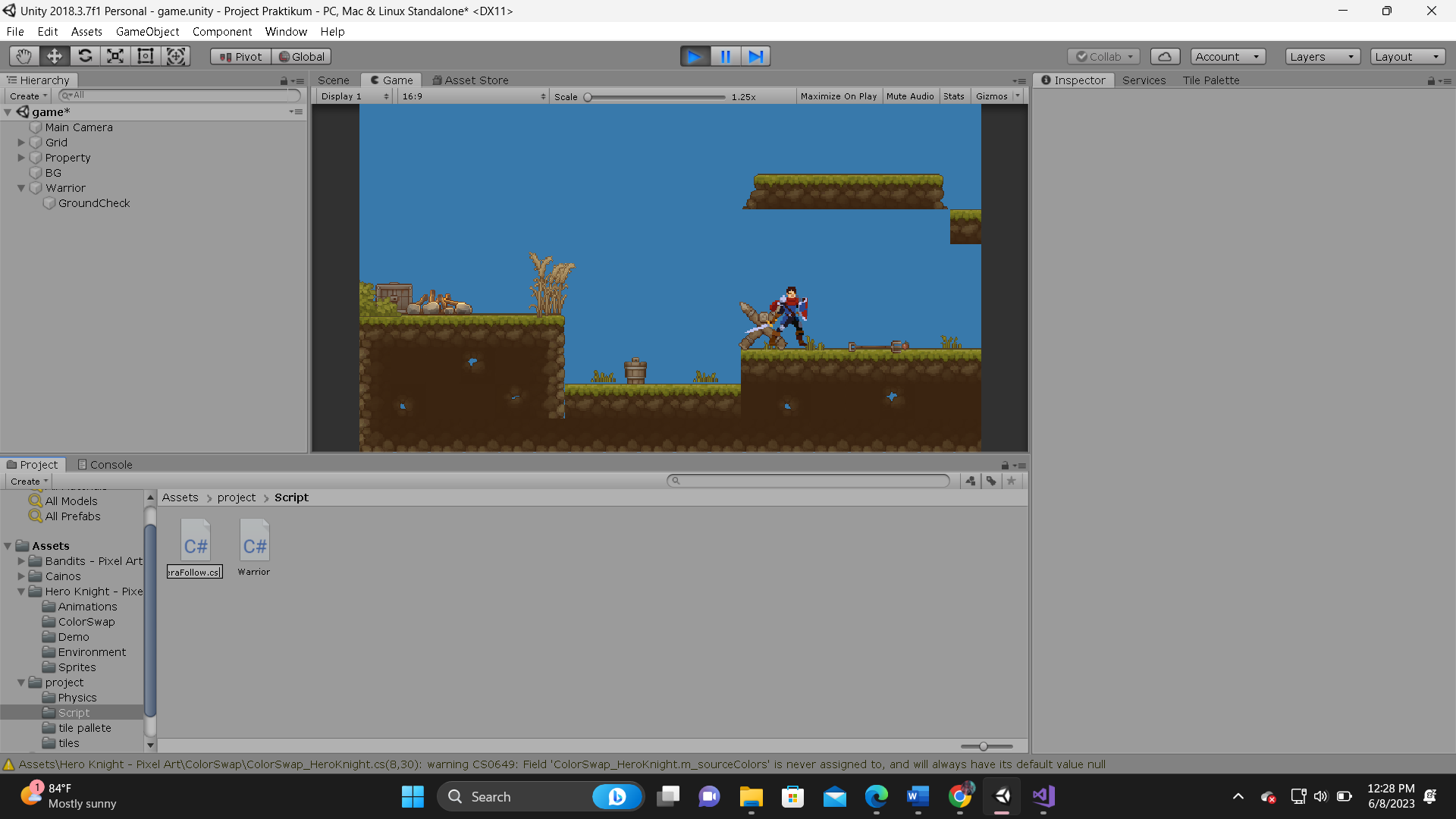
### Drag physics *2D* kedalam material RigidBody *2D*

1. Tekan play untuk mencoba hasilnya, karakter sekarang sudah bisa bergerak dan melompat.



### Tampilan karakter sudah bisa melompat

1. **Langkah-langkah *Camera* *Movement***
2. Tambahkan *script* dalam folder *Script* dan beri nama “*Camera*Follow.cs”.

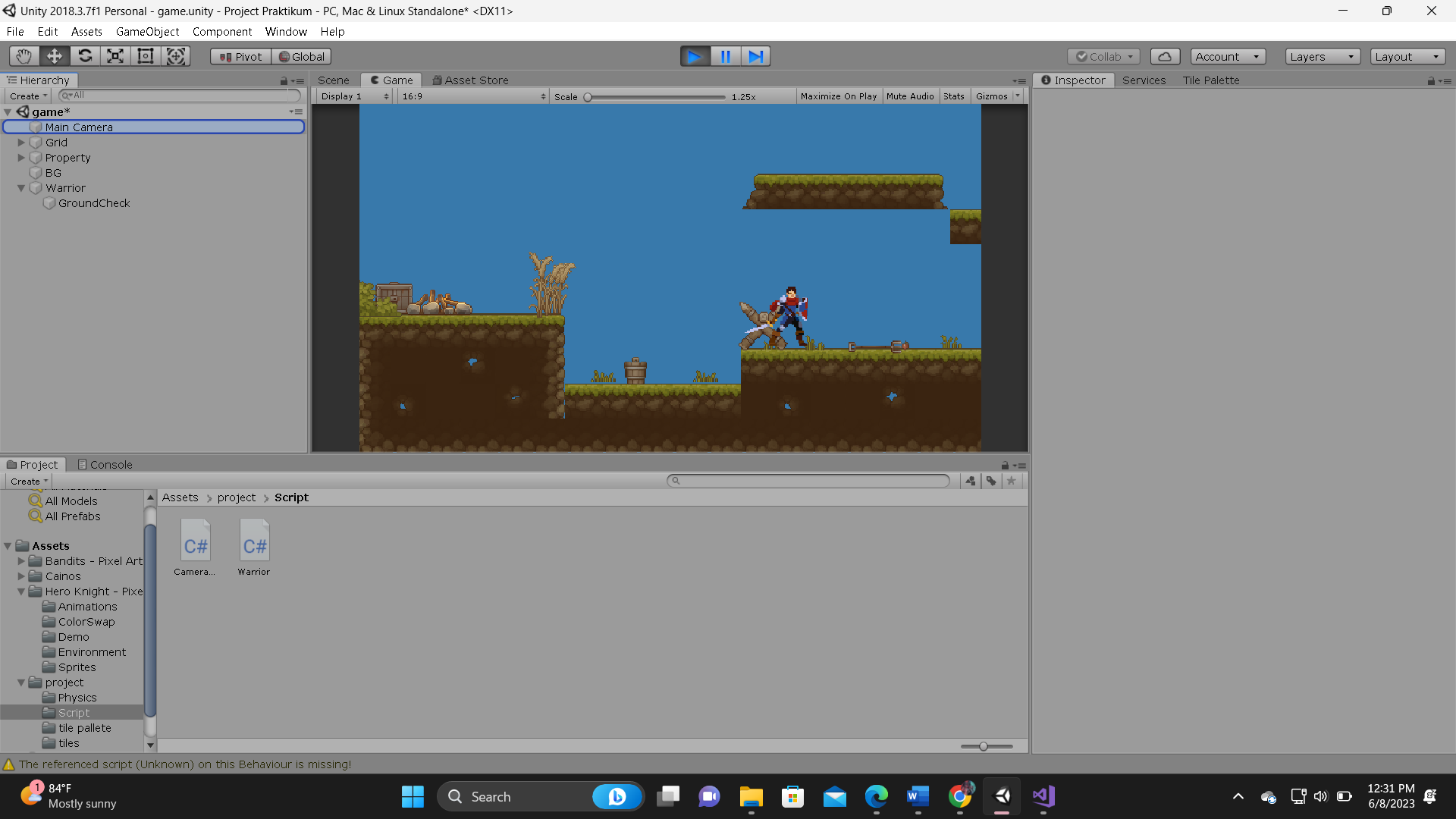


### Tampilan menambahkan *script* *CameraFollow*

1. Isikan source code dibawah ini.

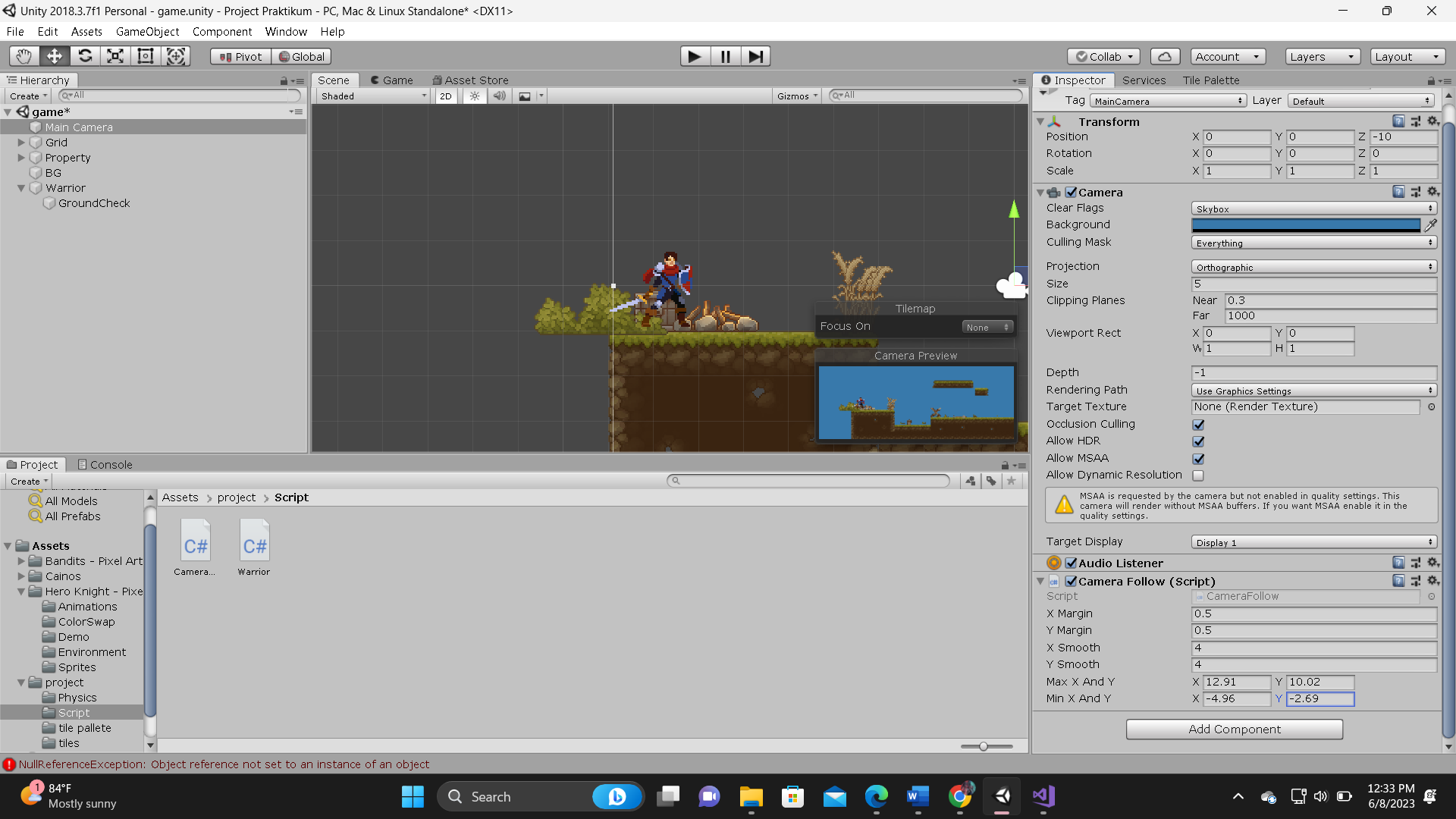
|  |
| --- |
| using System.Collections; using System.Collections.Generic; using UnityEngine; public class *Camera*Follow : MonoBehaviour {     public float xMargin = 0.5f;     public float yMargin = 0.5f;     public float xSmooth = 4f;     public float ySmooth = 4f;     public Vector2 maxXAndY;     public Vector2 minXAndY;     private Transform *player*;     void Awake()     {         *player* = *Game*Object.Find*Game*ObjectWithTag("P*layer*").transform;     }     bool CheckXMargin()     {         return Mathf.Abs(transform.position.x - *player*.position.x)  lalu  xMargin;     }     bool CheckYMargin()     {         return Mathf.Abs(transform.position.y - *player*.position.y)  lalu  yMargin;     }     void FixedUpdate()     {         TrackP*layer*();     }     void TrackP*layer*()     {         float targetX = transform.position.x;         float targetY = transform.position.y;         if (CheckXMargin())             targetX = Mathf.Lerp(transform.position.x, *player*.position.x,                 xSmooth \* *Time*.delta*Time*);         if (CheckYMargin())             targetY = Mathf.Lerp(transform.position.y, *player*.position.y,                 ySmooth \* *Time*.delta*Time*);         targetX = Mathf.Clamp(targetX, minXAndY.x, maxXAndY.x); targetY =             Mathf.Clamp(targetY, minXAndY.y, maxXAndY.y); transform.position = *new*             Vector3(targetX, targetY, transform.position.z);     } } |

1. Simpan file *script* tersebut, kemudian drag dan masukkan ke dalam *Game* Object Main *Camera*. Seperti pada gambar dibawah ini



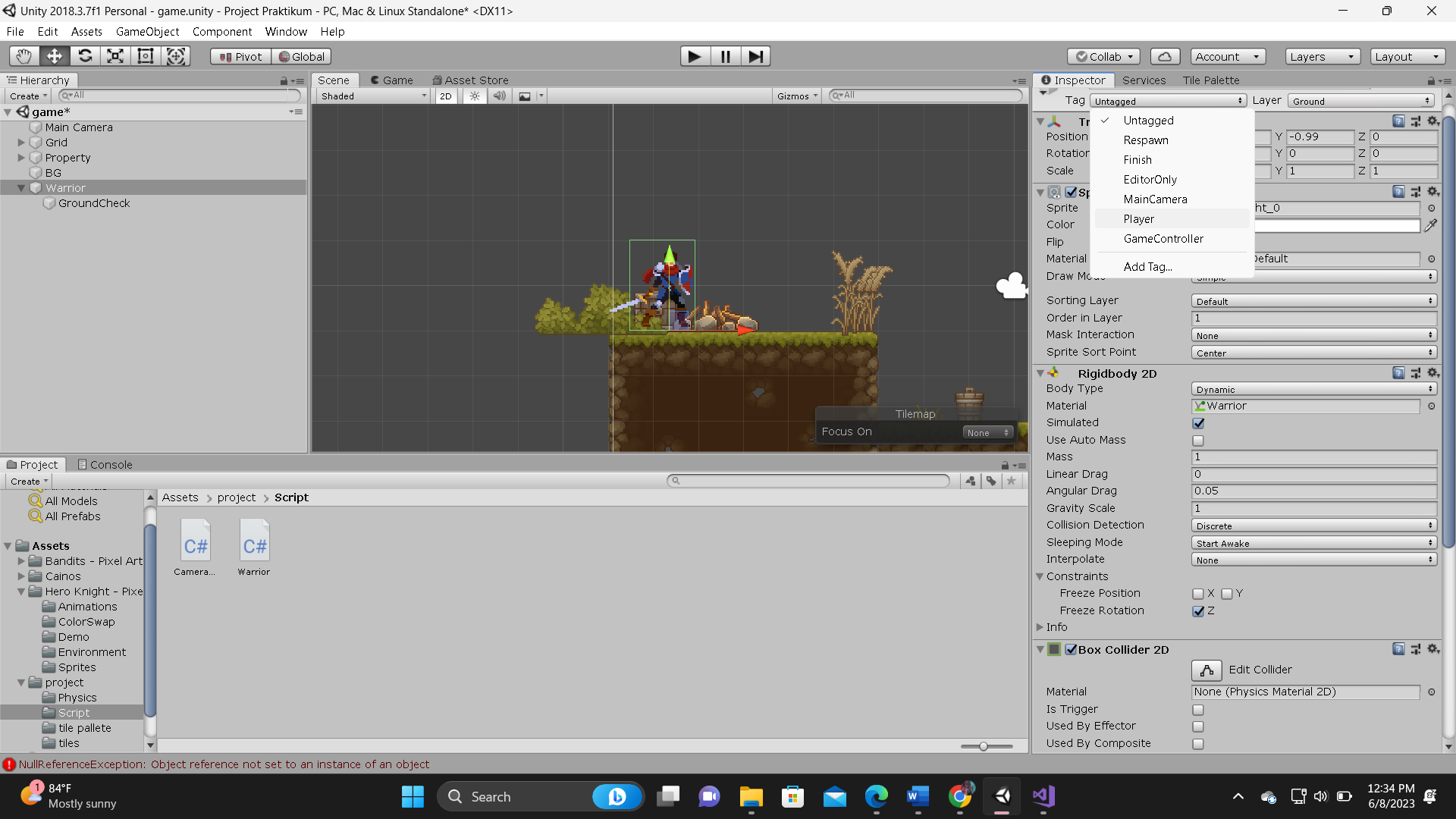
### Tampilan drag *script* kedalam Object Main *Camera*

1. Pergi ke *Inspector* Main *Camera* kemudian lakukan setting *Camera* Follow seperti gambar dibawah ini.



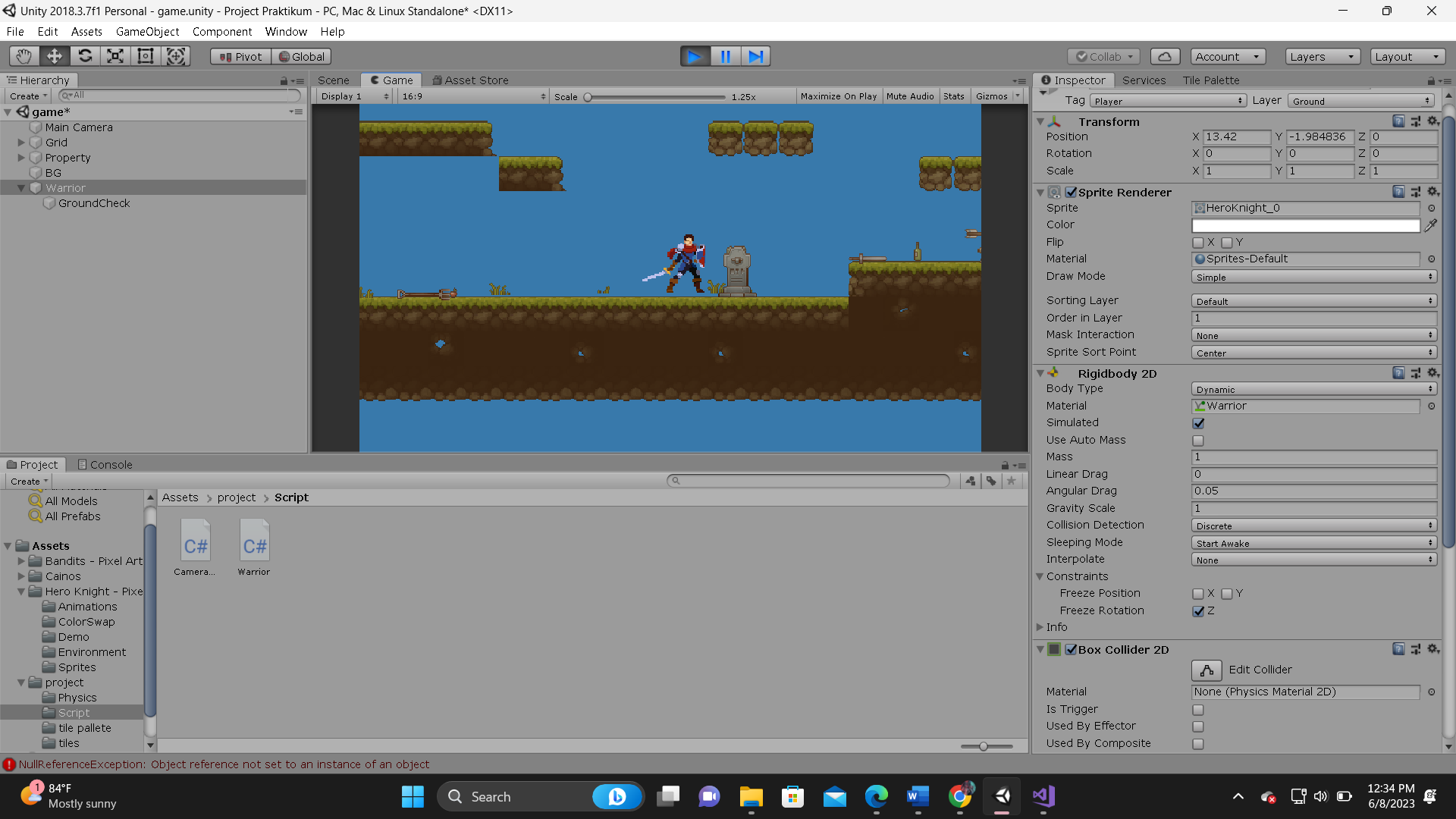
### Tampilan setting *Camera* Follow

1. Pada *inspector* karakter yang digunakan, ganti tag menjadi P*layer*.



### Mengubah tag menjadi P*layer* pada main camera

1. Tekan play, maka camera sudah bisa mengikuti karakter.



### Tampilan camera mengikuti karakter

## Kesimpulan

1. Bahasa C# adalah sebuah Bahasa pemrograman modern yang bersifat general purpose, berorientasi objek, yang dapat digunakan untuk membuat program diatas arsitektur Microsoft .Net. *Frame*work.
2. Dalam Unity Bahasa C# adalah Bahasa pemrograman default untuk pengembangan *game* menggunakan Unity Secara default pada saat pertama membuat file *script* yang baru akan ditampilkan 2 buah prosedur Monobehaviour.
3. Komp*one*n *RigidBody2D* menempatkan objek dibawah kendali sifat fisika. Objek hanya dapat bergerak di bidang XY dan hanya dapat memutar pada sumbu yang