Unity 3D, kamera, postać, bump mapping

1. Wstęp:

Zapoznanie ze sposobem dodania postaci z **Asset Store**, utworzenia podążającej za postacią kamery, ruchomych obiektów, tekstur z bump mappingiem.

2. Oprogramowanie:

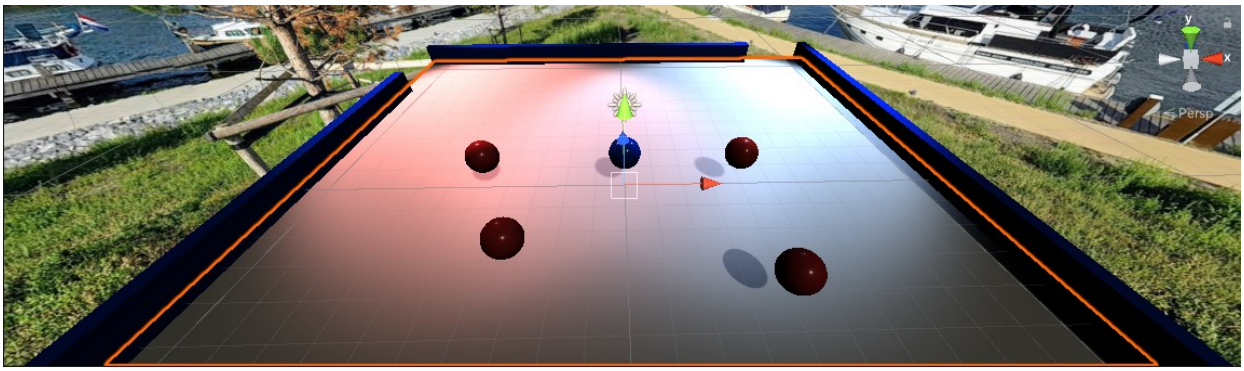
Do wykonania projektu konieczne jest zainstalowanie Unity3D, co wiąże się z utworzeniem konta, oraz dowolnego edytora kodu (IDE).

3. Ćwiczenie:

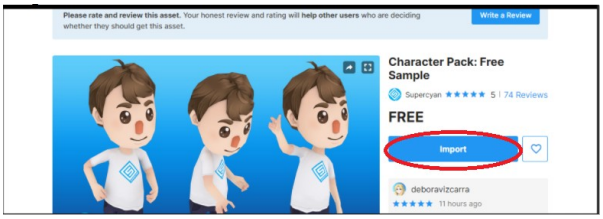
Wykonać projekt sceny z zaimportowaną animowaną postacią sterowaną klawiszami strzałek, kamerę podążającą za postacią, ruchome dowolne obiekty, tekstury z bump mappingiem.

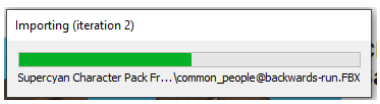
4. Wykonanie zadania:

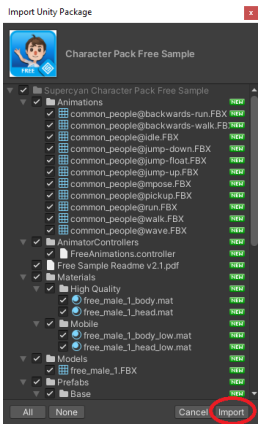
1. Uruchomić Unity3D.
2. Otworzyć projekt z poprzednich zajęć.



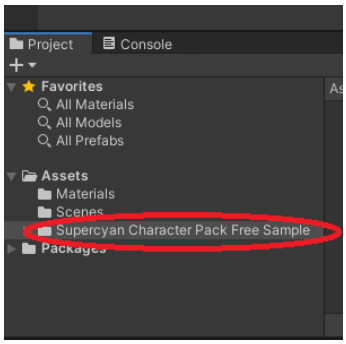
3. Otworzyć okno Asset Store i w wyszukiwarce wpisać: „Free 3d character” i wybrać asset jak poniżej klikając przycisk **Import** i w nowym oknie **Unity Import Package** kliknąć przycisk **Import.**



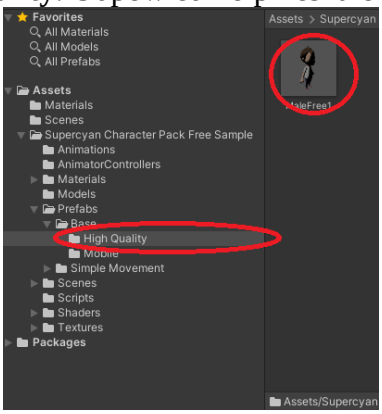




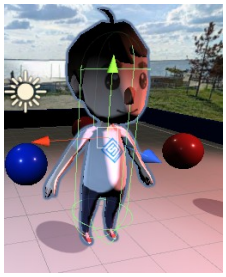
Po zaimportowaniu assetu powinien być on widoczny w oknie Project w katalogu Assets.



4. W celu dodania postaci do sceny należy przeciągnąć asset z katalogu **Prefabs → Base → High Quality**. Odpowiednio przeskalować model.



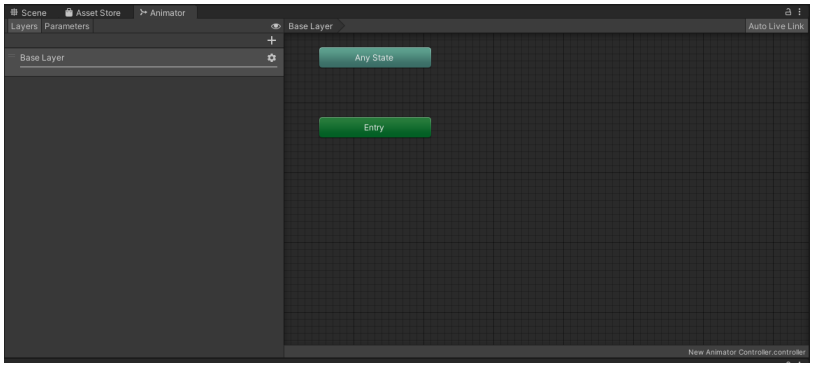
5. Do postaci dodać component **RigidBody** oraz **CapsuleCollider** i dopasować do rozmiarów. Ograniczyć dla **RigidBody** możliwość obrotów.



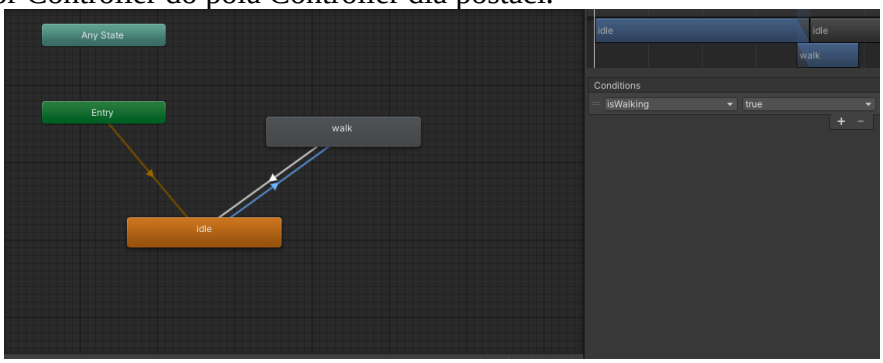
6. Utworzyć nowy skrypt **CharacterMovement**. Przeciągnąć MaleFree1 z okna Hierarchy na pole Rb w skrypcie Character Movement w oknie Inspector. Poniższy skrypt pozwala na poruszanie postacią względem lokalnego układu współrzędnych.

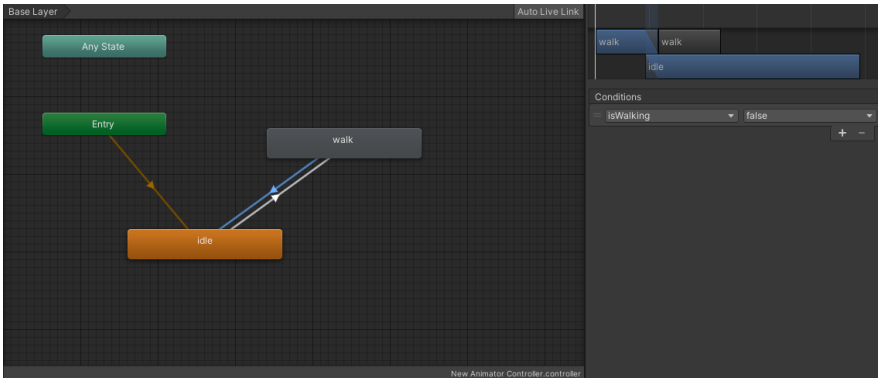
```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class CharacterMovement : MonoBehaviour
6  {
7      public Rigidbody rb;
8
9      public float accel = 400.0F;
10     public float rotateSpeed = 2.0F;
11     private Vector3 moveDirection = Vector3.zero;
12
13     // Update is called once per frame
14     void FixedUpdate()
15     {
16         moveDirection = new Vector3(0,0 , Input.GetAxis("Vertical"));
17         moveDirection = transform.TransformDirection(moveDirection);
18         moveDirection *= accel;
19
20         rb.AddForce(moveDirection* Time.deltaTime);
21
22         //Rotate Player
23         transform.Rotate(0, Input.GetAxis("Horizontal")*rotateSpeed, 0);
24
25     }
26 }
27
```

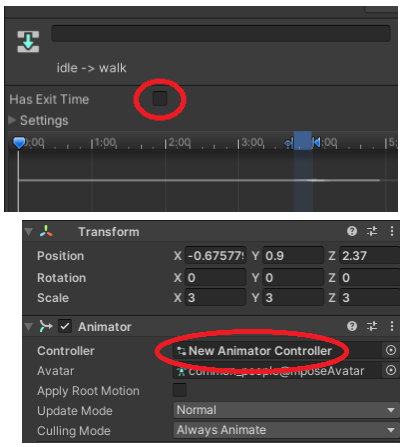
7. Usunąć ze sceny wcześniej sterowaną sferę.
8. Kolejny etap to uruchomienie animacji postaci. W tym celu konieczne jest stworzenie kontrolera animacji. Prawy klawisz myszy w obszarze **Project**, **Create → AnimatorController.** Następnie podwójnie kliknąć utworzony asset, co otworzy okno kontrolera.



9. Odszukać w katalogu Animations dwie animacje **idle** i **walk** i przeciągnąć do okna **Base Layer.** Utworzyć przejścia z **idle** do **walk** i z powrotem. (**Prawy klawisz myszy → MakeTransition**).
10. Następnie należy utworzyć warunki przejść. W karcie **Parameters**, znak **+**,**Bool** i wpisać nazwę **isWalking.**
11. Kliknąć przejście z **idle** do **walk** i z powrotem dodać utworzony warunek klikając znak + zakładce Conditions w oknie Inspector. W pierwszym przypadku zmienna pozostaje na **true** w drugim na **false.** Dla przejść wyłączyć opcję **Has Exit Time.** Przeciągnąć utworzony Animator Controller do pola Controller dla postaci.







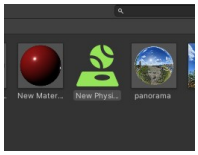
12. Następnie należy dodać kod uruchamiający animację w skrypcie kontrolującym ruch postaci, oraz ograniczający maksymalną szybkość. Przeciągnąć MaleFree1 do pola Anim skryptu.

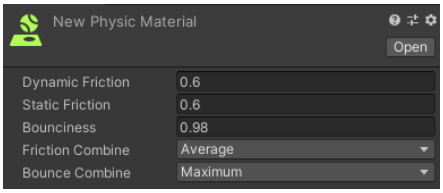
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CharacterMovement : MonoBehaviour
{
    public Rigidbody rb;

    public float accel = 400.0F;
    public float maxSpeed = 2.0F;
    public float rotateSpeed = 2.0F;
    private Vector3 moveDirection = Vector3.zero;
    public Animator anim;

    // Update is called once per frame
    void FixedUpdate()
    {

        moveDirection = new Vector3(0,0 , Input.GetAxis("Vertical"));
        moveDirection = transform.TransformDirection(moveDirection);
        moveDirection *= accel;

        rb.AddForce(moveDirection* Time.deltaTime);


        //Rotate Player
        transform.Rotate(0, Input.GetAxis("Horizontal")*rotateSpeed, 0);


        Vector3 vel = rb.velocity;
        if (vel.magnitude > maxSpeed){
            rb.velocity = vel.normalized* maxSpeed;
        }

        if (vel.magnitude > 0.0F)
        anim.SetBool("isWalking",true);
        else
        anim.SetBool("isWalking",false);

    }
}

```

13. Kolejny zagadnienie to kamera podążająca za postacią. Istniejący kod skryptu dla kamery należy nadpisać poniższym kodem. Offset Position ustawić na x=0, y1.2, z-1.8

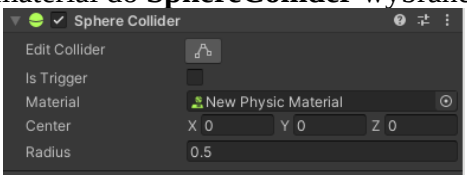
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class FollowCamera : MonoBehaviour
{

    public Transform target;
    public Vector3 offsetPosition;
    public bool lookAt = true;

    private void Update()
    {
        if(target == null)
        {
            Debug.LogWarning("Missing target ref !", this);
            return;
        }
            transform.position = target.TransformPoint(offsetPosition);

        // compute rotation
        if(lookAt)
        {
            transform.LookAt(target);
        }
        else
        {
            transform.rotation = target.rotation;
        }

    }
}

```

14. Kolejne zagadnienie to odbijająca się sfera od podłoża. Należy utworzyć nowy materiał fizyczny. Prawy klawisz myszy w obszarze **Assets → Create → PhysicMaterial.** Zmienić parametry materiału zgodnie z rysunkiem poniżej.





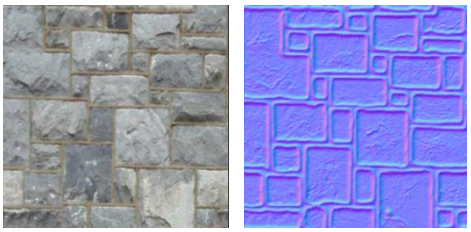
Przeciągnąć utworzony materiał do **SphereCollider** wybranego obiektu jako **Material.**



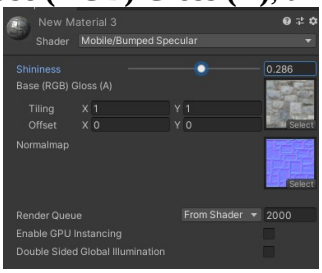
15. Kolejny element to rotacja obiektu względem lokalnego układu współrzędnych. Dodać obiekt Cylinder, obrócić 90°, przeskalować, dodać skrypt z kodem:

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

public class Rotation : MonoBehaviour
{
    public float speed = 50.0F;
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        transform.Rotate(Vector3.forward * speed * Time.deltaTime);
    }
}

```

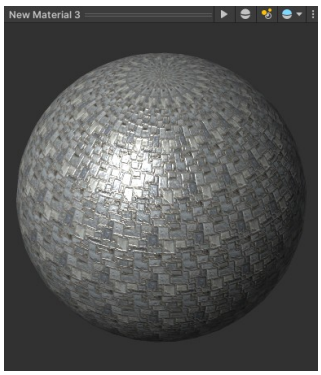
16. Kolejny element to dodawanie tekstur z BumpMappingiem. W tej technice konieczne jest użycie tekstury i jej mapy wektorów normalnych (NormalMap).



1. Utworzyć nowy materiał. Zmienić parametr **Shader** na **Mobile → BumpedSpecular**.
2. Zmienić typ tekstury z mapą normalnych na **TextureType → NormalMap**
3. Przeciągnąć teksturę na **Base (RGB) Gloss (A)**, a mapę na **NormalMap**



Wyregulować parametr Shinines. Oraz Tiling.



Przeciągnąć materiał na wybrany obiekt sceny.