

1. Cel zadania oraz wariant:

Opracować grę typu 2D Platformer z takimi parametrami

Wariant 1.

“Player” - enemy1 z Prefabs/Characters

Kolor platform – żółty

Obiekty „Coins” dwóch typów (w dowolnej postaci)

Reguły gry: trafiając w obiekt Coin typu 1 – 1 punkt

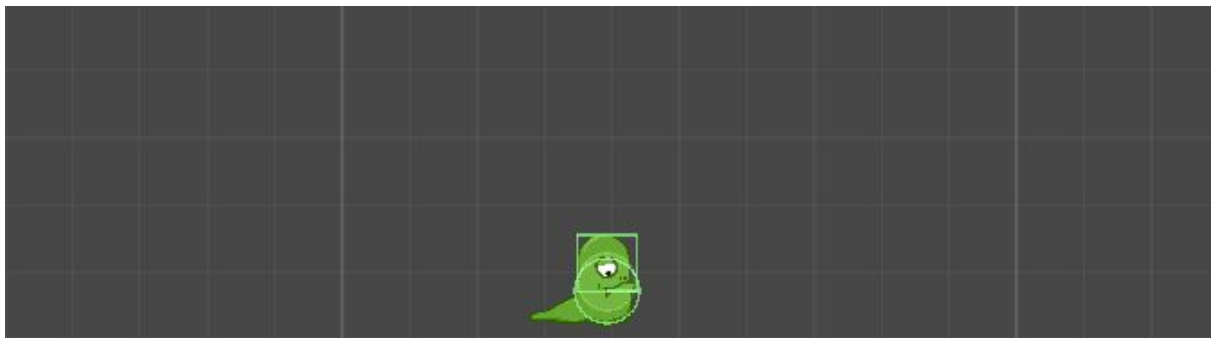
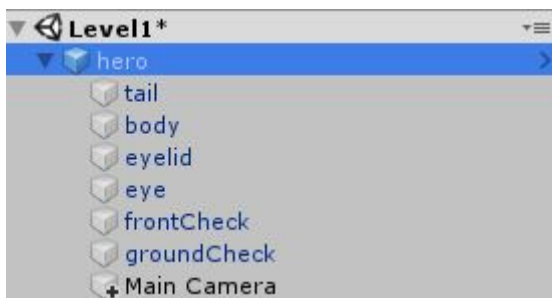
trafiając w obiekt typu 2 – 3 punkty

warunek zakończenia gry – 20 punktów

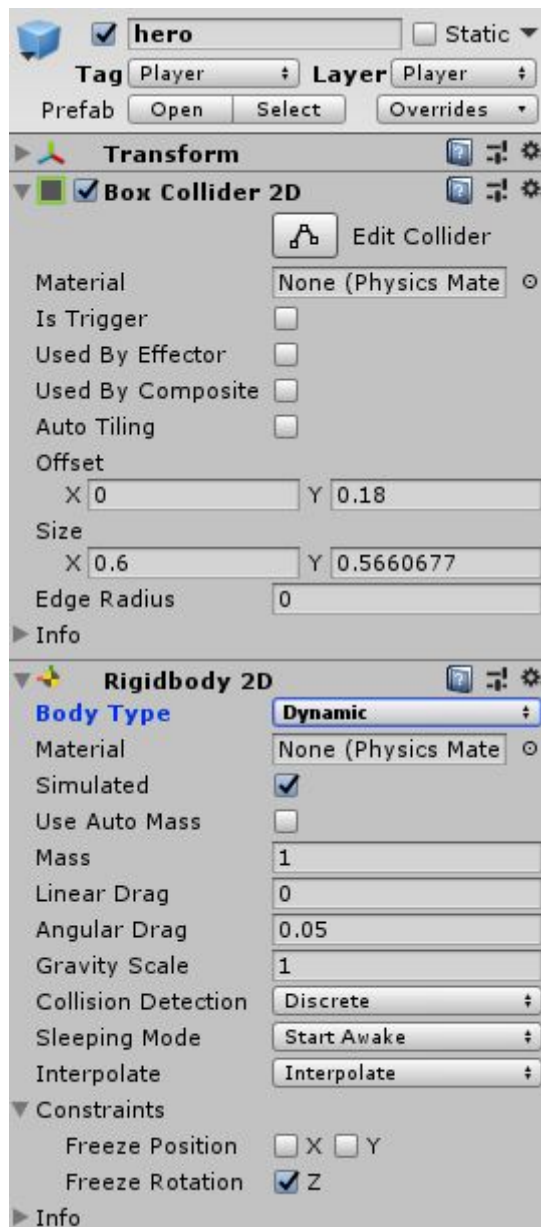
2. Przebieg ćwiczenia:

Na początek tworzymy projekt oraz dodajemy paczkę “Assets”.

Po stworzeniu projektu, należy dodać model “Playera”. W tym celu z folderu Prefabs wybieramy model “enemy1” (zgodnie z wariantem zadania) oraz przenosimy kamerę jako child gracza.








Ustawienia Playera:



Dodanie “bindu” skoku:

▼ Jump	
Name	Jump
Descriptive Name	
Descriptive Negative Name	
Negative Button	
Positive Button	up
Alt Negative Button	
Alt Positive Button	up
Gravity	1000
Dead	0.001
Sensitivity	1000
Snap	<input type="checkbox"/>
Invert	<input type="checkbox"/>
Type	Key or Mouse Button
Axis	X axis
Joy Num	Get Motion from all Joysticks

Dodanie skryptu SimplePlatformController:

▼   Simple Platform Contr	  
Script	SimplePlatformC
Move Force	365
Max Speed	5
Jump Force	1000
Ground Check	None (Transform)
<div>Add Component</div>	

Skrypt umożliwiający obrót postaci wokół jednej z osi, skok oraz aktualizacja pozycji.

```
public class SimplePlatformController : MonoBehaviour
{
    [HideInInspector]
    public bool facingRight = true;
    [HideInInspector]
    public bool jump = false;
    public float moveForce = 365f;
    public float maxSpeed = 5f;
    public float jumpForce = 1000f;
    public Transform groundCheck;

    private bool grounded = false;
    private Animator anim;
    private Rigidbody2D rb2d;

    void Awake()
    {
        anim = GetComponent<Animator>();
        rb2d = GetComponent<Rigidbody2D>();
    }

    void Update()
    {
        grounded = Physics2D.Linecast(transform.position, groundCheck.position, 1 << LayerMask.NameToLayer("Ground"));
        if (Input.GetButtonDown("Jump") && grounded)
        {
            jump = true;
        }
    }

    void Flip()
    {
        facingRight = !facingRight;
        Vector3 tempScale = transform.localScale;
        tempScale.x *= -1;
        transform.localScale = tempScale;
    }
}
```

```

void FixedUpdate()
{
    float h = Input.GetAxis("Horizontal");
    anim.SetFloat("Speed", Mathf.Abs(h));

    if (h * rb2d.velocity.x < maxSpeed)
    {
        rb2d.AddForce(Vector2.right * h * moveForce);
    }

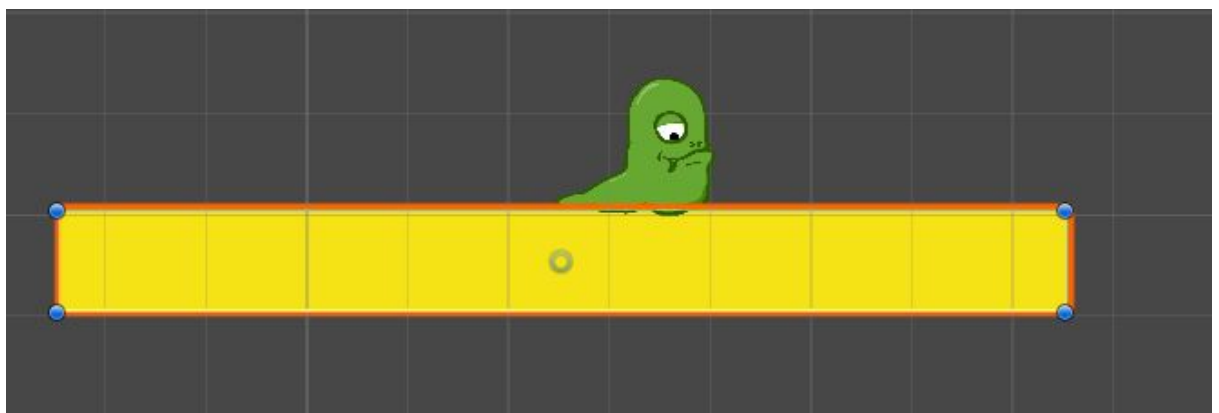
    if (Mathf.Abs(rb2d.velocity.x) > maxSpeed)
    {
        rb2d.velocity = new Vector2(Mathf.Sign(rb2d.velocity.x) * maxSpeed, rb2d.velocity.y);
    }

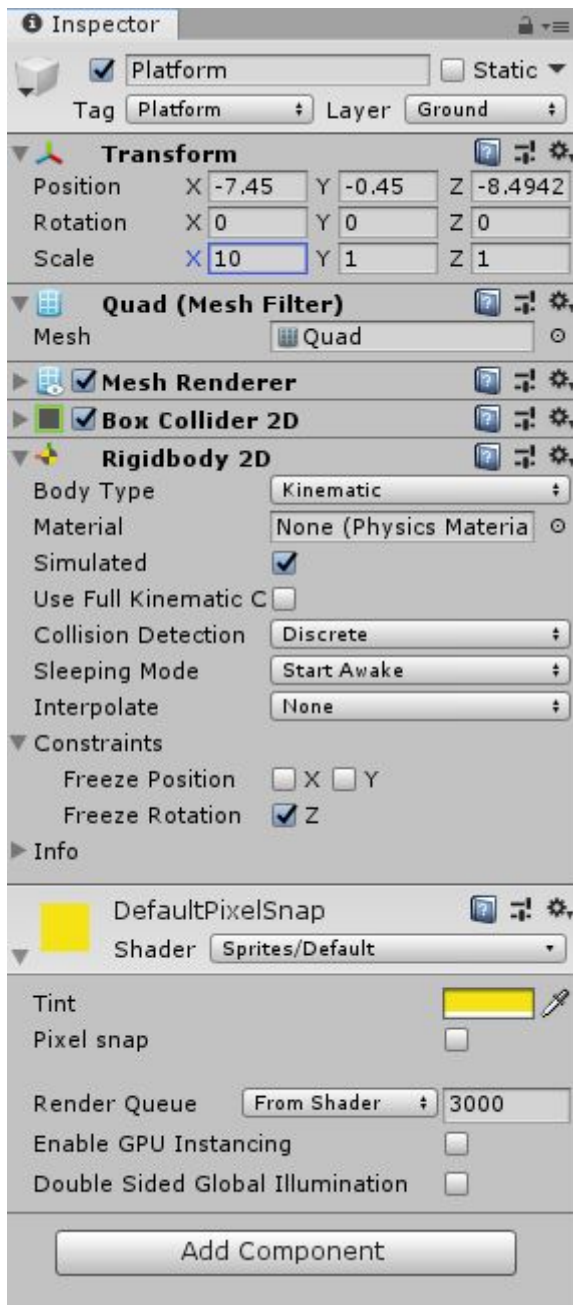
    if (h > 0 && !facingRight)
    {
        Flip();
    }
    else if (h < 0 && facingRight)
    {
        Flip();
    }

    if (jump)
    {
        anim.SetTrigger("Jump");
        rb2d.AddForce(new Vector2(0f, jumpForce));
        jump = false;
    }
}

```

Dodanie platformy (kolor: żółty zgodnie z wariantem):





Następnie tworzymy skrypt spadania platform oraz spawner (obiekt, którego zadaniem będzie wygenerowanie określonej ilości platform).

Spadek platform (skrypt przypisany do obiektu "platform"):

```
public class PlatformFall : MonoBehaviour
{
    public float fallDelay = 1f;
    private Rigidbody2D rb2d;

    void Awake()
    {
        rb2d = GetComponent<Rigidbody2D>();
    }

    void OnCollisionEnter2D(Collision2D other)
    {
        if (other.gameObject.CompareTag("Player"))
        {
            Invoke("Fall", fallDelay);
        }
    }

    void Fall()
    {
        rb2d.isKinematic = false;
    }
}
```

Generator platform:

```
public class SpawnPlatform : MonoBehaviour
{
    public int maxPlatforms = 20;
    public GameObject platform;
    public float horizontalMin = 7.5f;
    public float horizontalMax = 14f;
    public float verticalMin = -6f;
    public float verticalMax = 6f;
    private Vector2 originPosition;

    void Start()
    {
        originPosition = transform.position;
        Spawn();
    }

    void Spawn()
    {
        for (int i = 0; i < maxPlatforms; i++)
        {
            Vector2 randomPosition = originPosition + new Vector2(Random.Range(horizontalMin, horizontalMax), Random.Range(verticalMin, verticalMax));
            Instantiate(platform, randomPosition, Quaternion.identity);
            originPosition = randomPosition;
        }
    }
}
```

Aktualny stan gry:



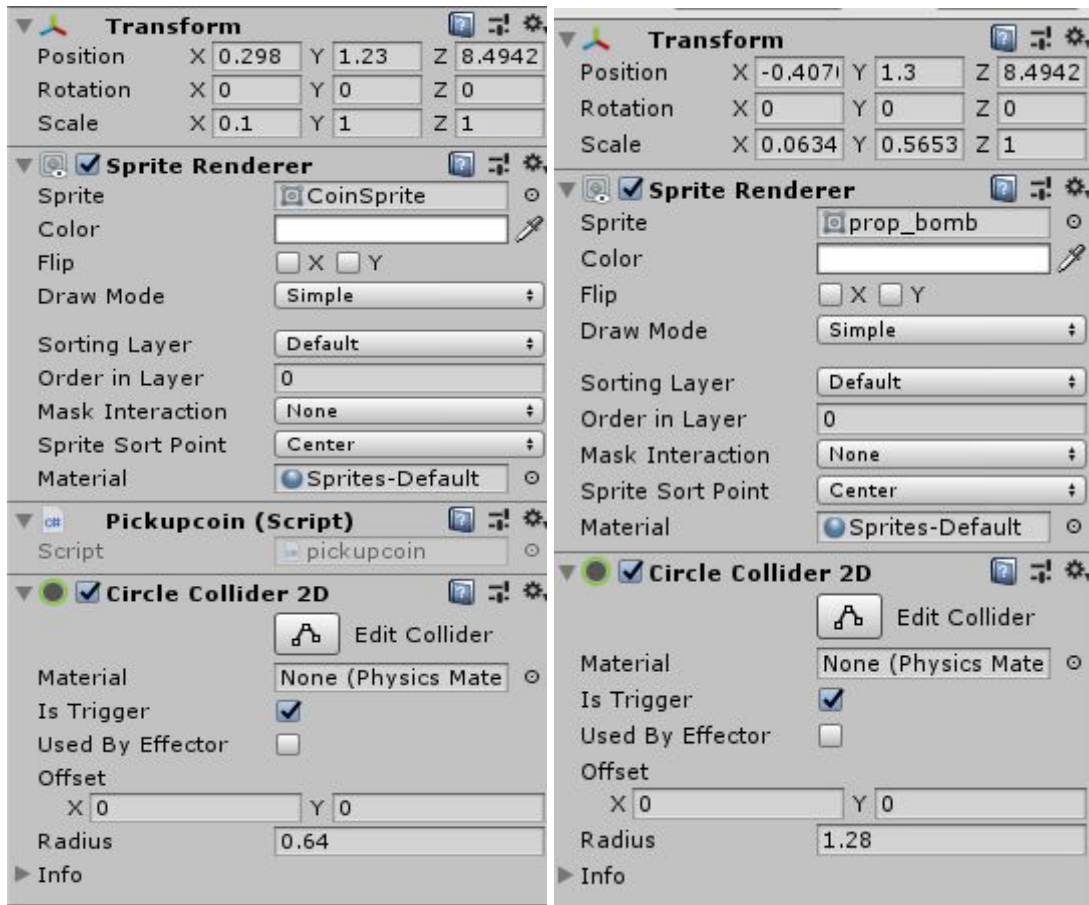
Dodanie warunku porażki poprzez dodanie obiektu "Cube" na dole planszy. Przy dotknięciu owego Cube, poziom zostaje zresetowany. W przypadku dotknięcia Cube przez platformę, platforma znika.

```
public class DestroyScript : MonoBehaviour
{
    void OnTriggerEnter2D (Collider2D other)
    {
        if (other.gameObject.CompareTag("Player"))
        {
            Application.LoadLevel(Application.loadedLevel);
        }
    }
    void OnCollision2D (GameObject other)
    {
        if (other.gameObject.CompareTag("Platform"))
        {
            Destroy(other);
        }
    }
}
```


Dodanie monet:

Jako monet został użyty model "CoinSprite" (element typu coin), jako element typu 2 został użyty model "prop_bomb".

Aby punkty mogły zostać zebrane należy dodać collider do elementów utworzonych w celu ich kolekcji oraz zaznaczyć opcję "isTrigger".

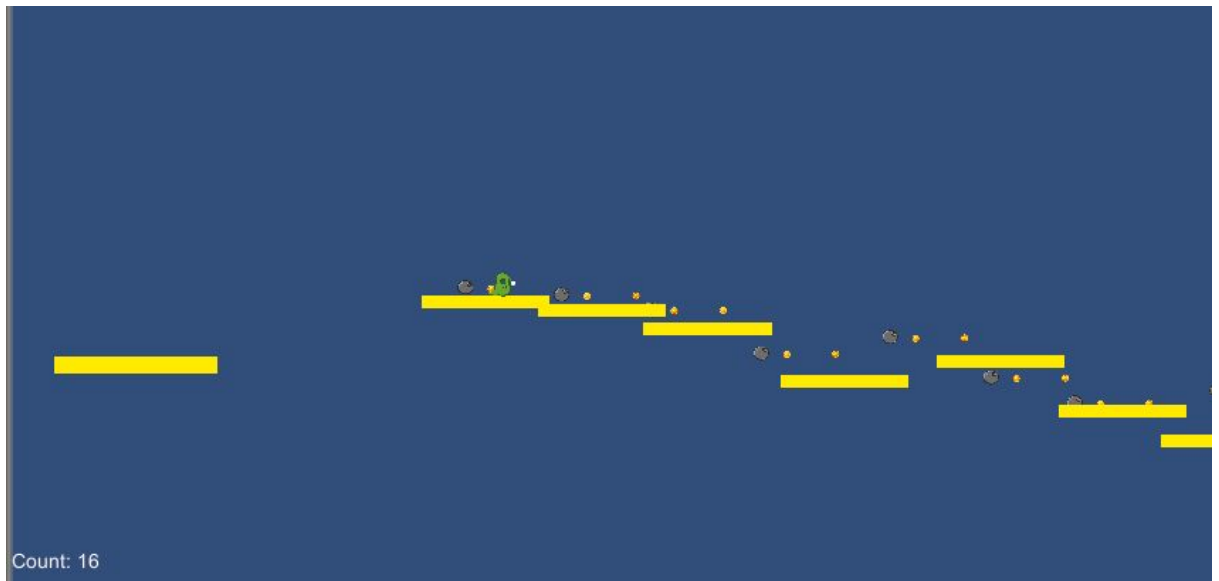


Skrypt umożliwiający zliczanie punktów oraz usuwanie obiektu przy kolizji z graczem.

```
void OnTriggerEnter2D(Collider2D other)
{
    if (other.gameObject.CompareTag("Coin"))
    {
        Destroy(other.gameObject);
        Debug.Log("Coin was picked up");
        count=count+1;
        SetCountText();
    }
    if (other.gameObject.CompareTag("PickUp2"))
    {
        Destroy(other.gameObject);
        Debug.Log("PickUp was picked up");
        count=count+3;
        SetCountText();
    }
}
```

Funkcja SetCountText aktualizuje pole tekstowe zawierające aktualną ilość punktów.

Stan gry przed zwycięstwem:



Zdobyty warunek minimum 20 punktów:



Napis "You Win!" jest wyświetlany przy osiągnięciu odpowiedniej ilości punktów dzięki skryptowi.

```
void SetCountText()
{
    countText.text = "Count: " + count.ToString();
    if (count >= 20)
    {
        winText.text = "You Win!";
    }
}
```

3. Wnioski:

Napisanie prostej gry platformowej 2D przy użyciu Unity nie jest zadaniem skomplikowanym. Dzięki dużej ilości aktualnych poradników oraz dzięki licznym forum, na których można znaleźć mnóstwo rozwiązanych problemów, wyszukanie potrzebnych informacji staje się proste.

W niniejszym ćwiczeniu najważniejszym elementem było utworzenie "colliderów", dzięki którym warunki gry mogły zostać zaimplementowane. Przykładem może być możliwość zbierania punktów, poprzez wywołanie kolizji z obiektem typu "Coin" lub "typu 2". Warunek przegranej również jest oparty na kolizji niewidzialnego obiektu na dole planszy oraz gracza (model "enemy1").