Gra platformowa 2d

**Spis treści**

2. Opis gry

3. Elementy gry

4. Sterowanie

5. Kod źródłowy

5-7. Kontroler gracza

8. Kontroler kamery

8-9. Menu główne

9-10. Przeciwnik Żaba

11. Niszczenie przeciwników

12. Narzędzia użyte do stworzenia gry

13. Schemat klas UML

**Opis Gry**

**Gra jest prostą „platformówką” celem każdego poziomu jest przejście z jednego końca mapy na drugi. Sposobem na przegraną (powrotu na początek mapy) jest wypadnięcie z mapy lub strata wszystkich żyć poprzez otrzymanie obrażeń od przeciwników. Jedynym ułatwienie jest wykorzystywanie mapy. Największą przeszkodą są przeciwnicy i sam gracz nie staraj się wygrać mapy na czas lecz na spryt pamiętaj „gdy się człowiek spieszy to się diabeł cieszy”.**

**Elementy Gry**

**GraczGłówny element gry którym sterujemy.**

**Przeciwnicy:**

**-Żaby  mogą nas zranić i przeszkadzają w przejściu na dalsze etapy mapy.**

**-Orły  służą głównie do przejścia na niedostępne miejsca na mapie**

**Wiśnie  służą do zbierania nie dają żadnego profitu w grze ale urozmaicają rozgrywkę.**

**Relację miedzy obiektami**

**Gracz  może zostać zraniony przez Żabę  oraz Orła  co spowoduję odepchnięcie  i utratę jednego życia. Gracz może zniszczyć przeciwników poprzez skoczenie na ich głowy (po zniszczeniu przeciwnika można się z niego wybić).**

**Sterowanie**

**A/D Poruszanie się w Lewo/Prawo**

**Spacja skok**

**Przytrzymanie Shift przyśpieszenie**

**Esc powrót do Menu**

**Kod Źródłowy**

**Kontrola gracza – PlayerMove.cs**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using TMPro;

using UnityEngine.SceneManagement;

public class PlayerMove : MonoBehaviour

{

//Start Valables/Wartości początkowe

private Rigidbody2D rb;

private Animator anim;

private Collider2D coll;

//FSM/Wartości służące zmianą animacji postaci

private enum State { idle, runing, jumping, falling, hurt }

private State state = State.idle;

//Player Valibles/Wartości służące zmiany w postaci

[SerializeField] private LayerMask ground;

[SerializeField] private float speed = 5f;

[SerializeField] private float jumpFroce = 15f;

[SerializeField] private int cherries = 0;

[SerializeField] private TextMeshProUGUI CherryText;

[SerializeField] private float hurtforce = 10f;

[SerializeField] private int health;

[SerializeField] private Text healthAmount;

private void Start()//Pobieranie komponentów gracza do skryptu

{

rb = GetComponent<Rigidbody2D>();

anim = GetComponent<Animator>();

coll = GetComponent<Collider2D>();

healthAmount.text = health.ToString();

}

private void Update()//wykonuje się w czasie rzeczywistym gry

{

if (state != State.hurt)

{

Movment();

}

AnimationState();

anim.SetInteger("state", (int)state);

if (gameObject.transform.position.y <= -30) //Restrt mapy po upadku

{

SceneManager.LoadScene(SceneManager.GetActiveScene().name);

}

if (Input.GetButtonDown("Cancel") && coll.IsTouchingLayers(ground)) //powrót do menu po wciśnięciu Esc

{

UnityEngine.SceneManagement.SceneManager.LoadScene(0);

}

}

private void OnTriggerEnter2D(Collider2D collision) //Zliczanie zebranych wiśni

{

if (collision.tag == "Collectable")

{

Destroy(collision.gameObject);

cherries += 1;

CherryText.text = cherries.ToString();

}

}

private void OnCollisionEnter2D(Collision2D other) //Interakcja z przeciwnikiem

{

Enemy enemy = other.gameObject.GetComponent<Enemy>();

if (other.gameObject.tag == "Enemy")//niszczenie przeciwnika

{

if (state == State.falling)

{

enemy.JumpOn();

Jump();

}

else//Otrzymywanie obrażeń od przeciwnika

{

state = State.hurt;

HealthLose();

if (other.gameObject.transform.position.x > transform.position.x)

{

rb.velocity = new Vector2(-hurtforce, rb.velocity.y);

}

else

{

rb.velocity = new Vector2(hurtforce, rb.velocity.y);

}

}

}

}

private void HealthLose()//Odejmowanie/restart mapy po otrzymaniu obrażeń

{

health -= 1;

healthAmount.text = health.ToString();

if (health <= 0)

{

SceneManager.LoadScene(SceneManager.GetActiveScene().name);

}

}

private void Movment()//Poruszanie się postaci i nadawanie wartości

{

float hDirection = Input.GetAxis("Horizontal");

// Przyśpieszenie po przytrzymaniu shfit zmian wartości prędkości

if (Input.GetKey(KeyCode.LeftShift))

{

speed = 7f;

}

else

{

speed = 5f;

}

//poruszanie się w prawo

if (hDirection < 0)

{

rb.velocity = new Vector2(-speed, rb.velocity.y);

transform.localScale = new Vector2(-1, 1);

}

//poruszanie się w lewo

else if (hDirection > 0)

{

rb.velocity = new Vector2(speed, rb.velocity.y);

transform.localScale = new Vector2(1, 1);

}

//skok

if (Input.GetButtonDown("Jump") && coll.IsTouchingLayers(ground))

{

Jump();

}

}

private void Jump()//skok

{

rb.velocity = new Vector2(rb.velocity.x, jumpFroce);

state = State.jumping;

}

private void AnimationState()//wywoływanie animację w danych sytacjach

{

if (state == State.jumping) //Animacja skoku

{

if (rb.velocity.y < .1f)

{

state = State.falling;

}

}

else if (state == State.falling) //Animacja opadania

{

if (coll.IsTouchingLayers(ground))

{

state = State.idle;

}

}

else if (state == State.hurt) //Animacja otrzymywania obrażeń

{

if (Mathf.Abs(rb.velocity.x) < .1f)

{

state = State.idle;

}

}

else if (Mathf.Abs(rb.velocity.x) > 2f) //animacja biegu

{

state = State.runing;

}

else

{

state = State.idle;

}

}

}

**Kontrola Kamery - CameraControler.cs**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class CameraControler : MonoBehaviour

{

public Transform player; //Wartość gracza

void Update()//Przypięcie kamery do gracza

{

transform.position = new Vector3(player.position.x, player.position.y, transform.position.z);

}

}

**Menu Główne – MainMenu.cs**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class MainMenu : MonoBehaviour

{

public void StartGame()//po wywołaniu wczytuje scene1(level1) przypięte do przycisku start

{

UnityEngine.SceneManagement.SceneManager.LoadScene(1);

}

public void QuitGame()()//po wywołaniu wyłącza grę przypięte do przycisku quit

{

Application.Quit();

}

public void level1 ()()//Wczytuje scene1(level1) w menu levels przypięte do przycisku Level1

{

UnityEngine.SceneManagement.SceneManager.LoadScene(1);

}

public void level2()()()//Wczytuje scene2(level2) w menu levels przypięte do przycisku Level2

{

UnityEngine.SceneManagement.SceneManager.LoadScene(2);

}

public void level3()()()//Wczytuje scene3(level3) w menu levels przypięte do przycisku Level3

{

UnityEngine.SceneManagement.SceneManager.LoadScene(3);

}

}

**Przeciwnik Żaba – Frog.cs**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Frog : Enemy//Klasa Frog dziedziczy Elementy klasy Eenemy

{

//Wartości Żaby

[SerializeField] private float rightCap;

[SerializeField] private float leftCap;

[SerializeField] private float jumpLenght = 10f;

[SerializeField] private float jumpHeight = 15f;

[SerializeField] private LayerMask ground;

private Collider2D coll;

//Zaba początkowa porusza się w prawo

private bool facingLeft = true;

//Pobieranie komponentów Żaby z projektu

protected override void Start()

{

base.Start();

coll = GetComponent<Collider2D>();

}

//aktualizacja w czasie rzeczysity

private void Update()

{

if(anim.GetBool("jumping"))

{

if(rb.velocity.y < .1)

{

anim.SetBool("falling", true);

anim.SetBool("jumping", false);

}

}

if(coll.IsTouchingLayers(ground) && anim.GetBool("falling"))

{

anim.SetBool("falling", false);

}

}

//poruszanie się Żaby

private void Move()

{

if (facingLeft) //Poruszanie się w lewo

{

if (transform.position.x > leftCap)

{

if (transform.localScale.x != 1)

{

transform.localScale = new Vector3(1, 1);

}

if (coll.IsTouchingLayers(ground))

{

rb.velocity = new Vector2(-jumpLenght, jumpHeight);

anim.SetBool("jumping", true);

}

}

else

{

facingLeft = false;

}

}

else//Zmian kierunku i poruszanie się w prawo

{

if (transform.position.x < rightCap)

{

if (transform.localScale.x != -1)

{

transform.localScale = new Vector3(-1, 1);

}

if (coll.IsTouchingLayers(ground))

{

rb.velocity = new Vector2(jumpLenght, jumpHeight);

anim.SetBool("jumping", true);

}

}

else

{

facingLeft = true;

}

}

}

}

**Niszczenie przeciwników – Enemy.cs**

public class Enemy : MonoBehaviour

{

//Wartości chronione które dziedziczy Frog.cs

protected Animator anim;

protected Rigidbody2D rb;

//Pobieranie wartości od obiektu do którego podjęty jest skrypt lub który dziedziczy od Enemy.cs

protected virtual void Start()

{

anim = GetComponent<Animator>();

rb = GetComponent<Rigidbody2D>();

}

//Aktywuje animację zniszczenia blokuję element i pozwala się z niego wybić

public void JumpOn()

{

anim.SetTrigger("Death");

rb.velocity = Vector2.zero;

rb.bodyType = RigidbodyType2D.Kinematic;

GetComponent<Collider2D>().enabled = false;

}

//Usuwa obiekt z gry ( niszczy przeciwnika)

private void Death()

{

Destroy(this.gameObject);

}

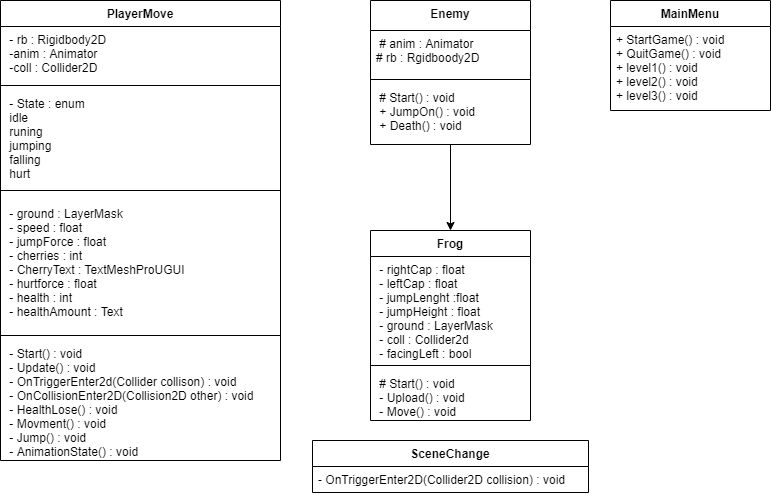
}

**Narzędzia Użyte Do Stworzenia Gry**

**Program do stworzenia mapy obiektów animacji użyłem Unity.**

**Program do stworzenia w skryptów języku c# użyłem Visual Studio 2019.**

**Schemat klas UML**

****