Dokumentacja techniczna
gry VISION RUN

Klasa Wait

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


public class Wait : MonoBehaviour
{
    public float waitTime = 5f;


    void Start()
    {
        StartCoroutine(WaitForIntro());
    }
    /// <summary>
    /// Korutryna wykorzystujaca SceneManager do przechodzenia miedzy scenami
    /// Wykorzystuje ona funkcje GetAcyiveScene i buildIndex do przejscia o 1
    /// Aby przejsc do nastepnej sceny
    /// </summary>

    IEnumerator WaitForIntro()
    {
        yield return new WaitForSeconds(waitTime);

        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
    }
}
```

Klasa Key

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

public class Key : MonoBehaviour
{

    [SerializeField] private KeyType keyType;
    /// <summary>
    /// \enum Typy wyliczeniowe kluczy
    /// </summary>
    public enum KeyType
    {
        Red,
        Green,
        Yellow
    }
    /// <summary>
    /// Pozyskanie klucza
    /// </summary>

    public KeyType GetKeyType()
    {
        return keyType;
    }

}
```

Klasa KeyDoor

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

public class KeyDoor : MonoBehaviour
{
    [SerializeField] private Key.KeyType keyType;

    public Key.KeyType GetKeyType()
    {
        return keyType;
    }
    /// <summary>
    /// \fn Funkcja ta reprezentuje stan drzwi czy sa otwarte czy nie
    /// </summary>
    public void OpenDoor()
    {
        gameObject.SetActive(false);
    }
}
```

Klasa KeyHolder

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

public class KeyHolder : MonoBehaviour
{
    public event EventHandler OnKeysChanged;
    private List<Key.KeyType> keyList;

    private void Awake()
    {
        keyList = new List<Key.KeyType>();
    }

    public List<Key.KeyType> GetKeyList()
    {
        return keyList;
    }
    /// <summary>
    /// Funkcja ta opdpowiada za dodanie klucza do ekwipunku
    /// Sprawdza potem czy gracz nie uzyl klucza
    /// </summary>
    /// <param name="keyType"></param>
    public void AddKey(Key.KeyType keyType)
    {
        Debug.Log("Added key:" + keyType);
        keyList.Add(keyType);
        OnKeysChanged?.Invoke(this, EventArgs.Empty);
    }
```

```
        /// <summary>
        /// Funkcja odpowiada za usuniecie klucza z ekwipunku
        /// Sprawdza potem czy gracz nie uzyl klucza
        /// </summary>

        public void RemoveKey(Key.KeyType keyType)
    {
        keyList.Remove(keyType);
        OnKeysChanged?.Invoke(this, EventArgs.Empty);
    }
    /// <summary>
    /// Funkcja ta sprawdza stan w jakim jest klucz
    /// </summary>


    public bool ContainsKey(Key.KeyType keyType)
    {
        return keyList.Contains(keyType);
    }
    /// <summary>
    /// Funkcja odpowiada za sprawdzenie czy to z
    /// czym koliduje gracz ma komponent key
    /// Jezeli tak dodaje klucz do ekwipunku
    /// I niszczy gameobject klucza
    /// </summary>

        private void OnTriggerEnter2D(Collider2D collision)
    {
        Key key = collision.GetComponent<Key>();
        if (key != null && !collision.CompareTag("GroundChecker"))
        {
            AddKey(key.GetKeyType());

            Destroy(key.gameObject);
        }

        /// <summary>
        /// Funkcja ta sprawdza czy to z czym koliduje gracz ma komponent KeyDoor
        /// Jezeli gracz posiada dany klucz drzwi sie otwieraja
        /// Usuwany zostanie klucz z ekwipunku
        /// </summary>

        KeyDoor keyDoor = collision.GetComponent<KeyDoor>();
        if (keyDoor != null)
        {
            if (ContainsKey(keyDoor.GetKeyType()))
            {
                RemoveKey(keyDoor.GetKeyType());
                keyDoor.OpenDoor();
            }
        }
    }
}
```

Klasa KeyHolder

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

public class UI_KeyHolder : MonoBehaviour
{

    [SerializeField] private KeyHolder keyHolder;
    [SerializeField] private Sprite redKey;
    [SerializeField] private Sprite greenKey;
    [SerializeField] private Sprite yellowKey;

    private Transform container;
    private Transform keyTemplate;

    private void Awake()
    {
        container = transform.Find("container");
        keyTemplate = container.Find("keyTemplate");
        keyTemplate.gameObject.SetActive(false);
    }

    private void Start()
    {
        keyHolder.OnKeysChanged += KeyHolder_OnKeysChanged;
    }

    private void KeyHolder_OnKeysChanged(object sender, System.EventArgs e)
    {
        UpdateVisual();
    }

    private void UpdateVisual()
    {
        /// <summary>
    /// Usuwanie starych kluczy
    /// </summary>

        foreach (Transform child in container)
        {
            if (child == keyTemplate) continue;
            Destroy(child.gameObject);
        }

        /// <summary>
    /// Inicjalizowanie nowej listy kluczy
    /// </summary>
        List<Key.KeyType> keyList = keyHolder.GetKeyList();
        for (int i = 0; i < keyList.Count; i++)
        {
            Key.KeyType keyType = keyList[i];
            Transform keyTransform = Instantiate(keyTemplate, container);
            keyTransform.gameObject.SetActive(true);
            keyTransform.GetComponent<RectTransform>().anchoredPosition = new
Vector2(50 * i, 0);
            Image keyImage = keyTransform.Find("image").GetComponent<Image>();
            switch (keyType)
            {
                default:
```

```
                case Key.KeyType.Red: keyImage.sprite = redKey; break;
                case Key.KeyType.Green: keyImage.sprite = greenKey; break;
                case Key.KeyType.Yellow: keyImage.sprite = yellowKey; break;
            }
        }
    }

}
```

Klasa LevelSelector

```
using UnityEngine;
using UnityEngine.UI;

public class LevelSelector : MonoBehaviour
{
    public SceneFader fader;

    public Button[] levelButtons;

    private void Start()
    {
      /// <summary>
    /// Inicjalizacja menu wyboru poziomu
    /// Gdy gracz nie przejdzie poziomu nastepny jest zablokowany
    /// </summary>

        int levelReached = PlayerPrefs.GetInt("levelReached", 1);

        for (int i = 0; i < levelButtons.Length; i++)
        {
            if (i + 1 > levelReached)
            {
                levelButtons[i].interactable = false;

            }

        }
    }

    public void Select(string levelName)
    {
/// <summary>
    /// Zablokowanie poziomu
    /// </summary>

        fader.FadeTo(levelName);
    }
}
```

Klasa Battery

```
using UnityEngine;

public class Battery : MonoBehaviour
{
    public string playerTag = "Player";

    private void OnTriggerEnter2D(Collider2D collision)
    {
    /// <summary>
    /// Funkcja dodajaca zycie dla gracza
    /// Sprawdza czy colider sprawdza czy baterii dotknął gracz
    /// Jeżeli tak Zycie zwieksza się a obiekt w postaci baterii niszczy sie
    /// </summary>

        if (collision.CompareTag(playerTag))
        {
            PlayerStats.Health++; ;
            Destroy(gameObject);
        }
    }
}
```

Klasa Bullet

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

public class Bullet : MonoBehaviour
{
    public float speed = 20f;
    public int damage = 40;
    public Rigidbody2D rb;


    /// <summary>
    /// Funkcja generuje naboj i nadale mu predkosci
    /// </summary>
    void Start()
    {
        rb.velocity = transform.right * speed;
    }

    private void OnTriggerEnter2D(Collider2D hitInfo)
    {
        //hitInfo.GetComponent<Ene>
        //Destroy(gameObject);
    }
}
```

Klasa Coin

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

public class Coin : MonoBehaviour
{
    public string playerTag = "Player";

    private void OnTriggerEnter2D(Collider2D collision)
    {
        if(collision.CompareTag(playerTag))
        {
            PlayerStats.Points+=10;
            Destroy(gameObject);
        }
    }
}
```

Klasa CompleteLevel

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

public class CompleteLevel : MonoBehaviour
{
    public string menuStringName = "MainMenu";

    public string nextLevel = "Level_02";
    //public int levelToUnlock = 2;

    public SceneFader sceneFader;

    public void Continue()
    {
        //PlayerPrefs.SetInt("levelReached", levelToUnlock);  //Nie wywali errora
jezeli pomyle nazwe zmiennej tylko zapisze do innej
        sceneFader.FadeTo(nextLevel);
    }

    public void Menu()
    {
      /// <summary>
    /// Przyciemnienie obrazu po zakończeniu levela
    /// </summary>

        sceneFader.FadeTo(menuStringName);
    }
}
```

Klasa FinishPoint

```
using UnityEngine;

public class FinishPoint : MonoBehaviour
{
    public GameManager gameManager;

    private void OnTriggerEnter2D(Collider2D collision)
    {
       /// <summary>
    /// Funkcja reaguje z colidarem jeżeli napotka tag gracza
    /// jeżeli wykryje gracza wygrywa poziom
    /// </summary>

        if(collision.tag == "Player")
        {
            gameManager.WinLevel();
        }
    }
}
```

Klasa GameOver

```
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class GameOver : MonoBehaviour
{
    public string menuStringName = "MainMenu";

    public SceneFader sceneFader;

    public void Restart()
    {
    /// <summary>
    /// Rozjasnienie poziomu po restarcie
    /// </summary>

        sceneFader.FadeTo(SceneManager.GetActiveScene().name);
    }

    public void Menu()
    {
    /// <summary>
    /// Przyciemnienie poziomu
    /// </summary>

        sceneFader.FadeTo(menuStringName);
    }
}
```

Klasa GroundChecker

```
using UnityEngine;

public class GroundChecker : MonoBehaviour
{
    [SerializeField]
    private string groundTag = "Ground";
    private bool isGrounded;

    private void OnTriggerEnter2D(Collider2D collision)
    {
    /// <summary>
    /// Sprawdzanie czy gracz porusz się po podlozu
    /// </summary>

        if (collision.CompareTag(groundTag))
        {
            isGrounded = true;
        }
    }

    private void OnTriggerExit2D(Collider2D collision)
    {
      /// <summary>
    /// Sprawdzenie czy gracz znajduje się na podlozu
    /// </summary>

        if (collision.CompareTag(groundTag))
        {
            isGrounded = false;
        }

    }
/// <summary>
    /// Funkcja zwraca prawde jeżeli znajduje się na podlozu
    /// Jezeli nie zwraca falsz
    /// </summary>

    public bool IsGrounded()
    {
        return isGrounded;
    }
}
```

Klasa LivesUI

```
using UnityEngine;
using UnityEngine.UI;

public class LivesUI : MonoBehaviour
{
    public Text livesText;

    private void Update()
    {
       /// <summary>
    /// Przepisanie punktow zycia na tekst
    /// </summary>

        livesText.text = PlayerStats.Health.ToString();
    }
}
```

Klasa MainMenu

```
using UnityEngine;

public class MainMenu : MonoBehaviour
{
    public string levelToLoad = "Level_01";

    public SceneFader sceneFader;

    public void Play()
    {  /// <summary>
    /// Wczytanie pierwszego poziomu
    /// </summary>

        sceneFader.FadeTo(levelToLoad);
    }

    public void Quit()
    {
/// <summary>
    /// Zamkniecie aplikacji
    /// </summary>

        Debug.Log("Exciting...");
        Application.Quit();
    }
}
```

Klasa PauseMenu

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class PauseMenu : MonoBehaviour
{
    public GameObject ui;

    public string menuSceneName = "MainMenu";

    public SceneFader sceneFader;

    private void Update()
    {
/// <summary>
    /// Sprawdzenie czy zostal wciśnięty przycisk Esc
    /// </summary>

        if (Input.GetKeyDown(KeyCode.Escape))
        {
            Toggle();
        }
    }

    public void Toggle()
    {
        ui.SetActive(!ui.activeSelf);
//Zmieni aktualny stan UI na przeciwny

        if (ui.activeSelf)                  //aktualny stan UI
        {
            Time.timeScale = 0f;            //Pause
        }
        else
        {
            Time.timeScale = 1f;            //Przywrocenie normalnego tempa gry
        }
    }

    public void Restart()
    {
        //Wznowienie tempa
        Toggle();

        sceneFader.FadeTo(SceneManager.GetActiveScene().name);
    }

    public void Menu()
    {
        Toggle();
        sceneFader.FadeTo(menuSceneName);
    }
}
```

Klasa PlayerAnimation

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

public class PlayerAnimation : MonoBehaviour
{
    [SerializeField] private Animator animator;
    [SerializeField] private PlayerMovement movement;
    [SerializeField] private Rigidbody2D rb;
    [SerializeField] private GroundChecker groundChecker;
    [SerializeField] private string isMovingParameterName = "IsMoving";
    [SerializeField] private string isGroundedParameterName = "IsGrounded";
    [SerializeField] private string isFallingParameterName = "IsFalling";

    private int _isMovingHash;
    private int _isGroundedHash;
    private int _isFallingHash;

    private bool flip = false;

    private void Start()
    {
        _isMovingHash = Animator.StringToHash(isMovingParameterName);
        _isGroundedHash = Animator.StringToHash(isGroundedParameterName);
        _isFallingHash = Animator.StringToHash(isFallingParameterName);
    }

    private void Update()
    {
      /// <summary>
    /// Ustawienie stanow animacji poruszania się i stania na ziemi
    /// </summary>

        animator.SetBool(_isMovingHash, movement.IsMoving());
        animator.SetBool(_isGroundedHash, groundChecker.IsGrounded());

        bool isFalling = rb.velocity.y < -0.01f;
        animator.SetBool(_isFallingHash, isFalling);


        float horizontalInput = movement.GetInput().x;

        if (horizontalInput > 0.01f && flip)
        {
      /// <summary>
    /// Warunek w którym jest sprawdzany jest kierunek w jakim gracz jest
///ustawiony
    /// </summary>

            transform.rotation = Quaternion.Euler(0, 0, 0);
            flip = false;
        }
        else if (horizontalInput < -0.01f && !flip)
        {
            transform.rotation = Quaternion.Euler(0, 180, 0);
            flip = true;
        }
    }
}
```

Klasa PlayerJump

```
using UnityEngine;

public class PlayerJump : MonoBehaviour
{
    [SerializeField]
    private float jumpPower = 6f;
    [SerializeField]
    private Rigidbody2D rb;
    [SerializeField]
    private GroundChecker groundChecker;

    private bool isJumping = false;

    private void Update()
    {
/// <summary>
    /// Sprawdzenie czy gracz wcisnal spacje
    /// oraz czy znajduje się na ziemi
    /// </summary>

        if(Input.GetKeyDown(KeyCode.Space) && groundChecker.IsGrounded()) {
            isJumping = true;
        }
    }

    private void FixedUpdate()
    {
        if(isJumping)
        {
    /// <summary>
    /// Sprawdzanie czy gracz skoczyl jeżeli tak nadawana jest mu
    /// prędkość i fizyka z gry po skoku ustawiany jest stan na false
    /// </summary>

            rb.AddForce(new Vector2(0, jumpPower), ForceMode2D.Impulse);
            isJumping= false;
        }
    }
}
```

Klasa PlayerMovement

```
using UnityEngine;

public class PlayerMovement : MonoBehaviour
{
    [SerializeField] private float moveSpeed = 500f;
    [SerializeField] private Rigidbody2D rb;

    private Vector3 input;
```

```
    private void Update()
    {
        float inputX = Input.GetAxis("Horizontal");
        float inputY = Input.GetAxis("Vertical");

        input = new Vector3(inputX, inputY, 0);

    }

    private void FixedUpdate()
    {
        Vector3 move = input * moveSpeed * Time.fixedDeltaTime;
        rb.velocity = new Vector2(move.x, rb.velocity.y);
    }

    public Vector3 GetInput()
    {
        return input;
    }

    public bool IsMoving()
    {
        return input.x != 0;
    }
}
```

Klasa PlayerStats

```
using UnityEngine;
public class PlayerStats : MonoBehaviour
{
    public static int Points;
    public static int Health;
/// <summary>
    /// Ustawienie danych poczatkowych dla gracza
    /// </summary>

    public int startPoints = 0;
    public int startHealth = 3;
    void Start()
    {
        Points = startPoints;
        Health = startHealth;
    }
}
```

Klasa PointsObtained

```
using System.Collections;
using UnityEngine;
using UnityEngine.UI;

public class RoundsSurvived : MonoBehaviour
{
    public Text pointsText;

    private void OnEnable()
    {
        StartCoroutine(AnimateText());
    }
```

```
    IEnumerator AnimateText()
    {
        pointsText.text = "0";
        int point = 0;

        yield return new WaitForSeconds(.7f);

        while (point < PlayerStats.Points)
        {    /// <summary>
    /// Petla w ktorej punkty sa dodawane do gracza
    /// </summary>

            point++;
            pointsText.text = point.ToString();
 /// <summary>
    /// Powrót do pola po czasie
    /// </summary>

            yield return new WaitForSeconds(.01f);
        }


    }
}
```

Klasa SceneFader

```
using UnityEngine;
using System.Collections;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class SceneFader : MonoBehaviour
{
    public Image img;
    public AnimationCurve curve;

    private void Start()
    {
/// <summary>
    /// Powtor do korutyny
    /// </summary>

        StartCoroutine(FadeIn());
    }

    public void FadeTo(string scene)
    {
/// <summary>
    /// Przejscie do korutyny
    /// </summary>

        StartCoroutine(FadeOut(scene));
    }

    IEnumerator FadeIn()
    {
        float t = 1f;
```

```
        while (t > 0f)
        {
/// <summary>
    /// Petla w ktorej jest przejscie w stan pauzy
    /// </summary>

            t -= Time.deltaTime;
            float a = curve.Evaluate(t);
            img.color = new Color(0f, 0f, 0f, a);
            yield return 0;     //Skip na next frame
        }
    }

    IEnumerator FadeOut(string scene)
    {
        float t = 0f;

        while (t < 1f)
        {
/// <summary>
    /// Petla w ktorej jest przejscie ze stanu pauzy
    /// </summary>

            t += Time.deltaTime;
            float a = curve.Evaluate(t);
            img.color = new Color(0f, 0f, 0f, a);
            yield return 0;     //Skip na next frame
        }

        SceneManager.LoadScene(scene);
    }
}
```

Klasa Weapon

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

public class Weapon : MonoBehaviour
{
    public Transform firePoint;
    public GameObject bulletPrefab;

        void Update()
    {
/// <summary>
    /// Funkcja sprawdza czy nacisnieto przycisk strzalu jak tak wykonywana
    /// jest funkcja Shoot
    /// </summary>

        if (Input.GetButtonDown("Fire1"))
        {
            Shoot();
        }
    }

    void Shoot()
    {
```

```
/// <summary>
    /// Funkcja generuje pocisk który strzela w zależności w ktora strone patrzy
///przeciwnik
    /// </summary>

        Instantiate(bulletPrefab, firePoint.position, firePoint.rotation);
    }
}
```