Dokumentacja techniczna
gry VISION RUN

W projekcie identyfikacja obiektów odbywa się za pomocą różnych tagów. Obiekt Player, który reprezentuje gracza ma przypisany tag Player, każdy z przeciwników ma przypisany tag Enemy, cała tailemapa odpowiedzialna za podłoże ma przypisany tag Ground. Obiekty z którymi gracz nie ma styczności są ustawione na BackGround by nie kolidowały podczas rozgrywki ale by urozmaiciły krajobraz poziomu. W wielu skryptach jest użyty ten mechanizm do identyfikacji obiektów i implementacji zachowań pomiędzy poszczególnymi obiektami. Dzięki temu można łatwiej zarządzać obiektami i mechaniką gry.

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());
   }

   /// Korutryna wykorzystujaca SceneManager do przechodzenia miedzy scenami
   /// Wykorzystuje ona funkcje GetAcyiveScene i buildIndex do przejscia o 1
   /// Aby przejsc do nastepnej sceny


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

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

Klasa ta odpowiada za przechodzenie między scenami. Pole waitTIme odpowiada ile czasu ma minąć między przejściem do kolejnej sceny.

Klasa Key

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

public class Key : MonoBehaviour
{

    [SerializeField] private KeyType keyType;

    /// \enum Typy wyliczeniowe kluczy
        public enum KeyType
    {
```

```
        Red,
        Green,
        Yellow
    }
        /// Pozyskanie klucza


    public KeyType GetKeyType()
    {
        return keyType;
    }

}
```
Klasa odpowiada za stworzenie typu enum dla każdego klucza. Zwraca dany klucz.

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;
    }
        /// \fn Funkcja ta reprezentuje stan drzwi czy sa otwarte czy nie

    public void OpenDoor()
    {
        gameObject.SetActive(false);
    }
}
```

Klasa KeyDoor odpowiada za drzwi na poziomie. Gdy funkcja ma stan false drzwi są zamknięte.

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;
}

/// Funkcja ta opdpowiada za dodanie klucza do ekwipunku
/// Sprawdza potem czy gracz nie uzyl klucza

public void AddKey(Key.KeyType keyType)
{
    Debug.Log("Added key:" + keyType);
    keyList.Add(keyType);
    OnKeysChanged?.Invoke(this, EventArgs.Empty);
}


  /// Funkcja odpowiada za usuniecie klucza z ekwipunku
  /// Sprawdza potem czy gracz nie uzyl klucza


  public void RemoveKey(Key.KeyType keyType)
{
    keyList.Remove(keyType);
    OnKeysChanged?.Invoke(this, EventArgs.Empty);
}

/// Funkcja ta sprawdza stan w jakim jest klucz



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

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

        Destroy(key.gameObject);
    }


    /// 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


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

```
        }
    }
}
```

Klasa odpowiada za zarządzanie kluczami. Dodaje je od ekwipunku i usuwa gdy gracz zetknie się z odpowiednimi drzwiami. Sprawdza też czy dany klucz ma odpowiedni nameTag dla danych drzwi.

Klasa KeyHolder UI

```
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()
    {

    /// Usuwanie starych kluczy


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


    /// Inicjalizowanie nowej listy kluczy

        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 odpowiada za wyświetlanie jakie gracz ma klucze. Gdy gracz zbierze klucz ekwipunek odświeża się i pokazuje zawartość.

Klasa LevelSelector

```
using UnityEngine;
using UnityEngine.UI;

public class LevelSelector : MonoBehaviour
{
    public SceneFader fader;

    public Button[] levelButtons;

    private void Start()
    {

    /// Inicjalizacja menu wyboru poziomu
    /// Gdy gracz nie przejdzie poziomu nastepny jest zablokowany


        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)
    {
    /// Zablokowanie poziomu


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

Klasa odpowiada za wybór danego poziomu. Blokuje poziom który nie jest odblokowany i gracz nie ma do niego dostępu.

Klasa Battery

```
using UnityEngine;

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

    private void OnTriggerEnter2D(Collider2D collision)
    {

    /// 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


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

Klasa odpowiada za życie gracza. Gdy gracz wejdzie w baterię ta zbiera się życie gracza zostaje dodane do niego i gameobject zostanie zniszczony by nie marnować pamięci.

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;



    /// Funkcja generuje naboj i nadale mu predkosci

    void Start()
    {
        rb.velocity = transform.right * speed;
    }

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

Klasa odpowiada za pocisk który wystrzeliwuje gracz. Są w niej zawarte informacje z jaką prędkością leci pocisk i ile zadaje obrażeń przeciwnikowi. Generuje pocisk i nadaje mu pęd.

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 odpowiada za generowania pieniędzy jako punktów. Gdy gracz wejdzie w interakcję z monetą dostaje on pewną ilość punktów.

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()
    {

    /// Przyciemnienie obrazu po zakończeniu levela


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

Klasa odpowiada za interakcję gracza z flagą końca poziomu. Gdy gracz wejdzie w nią poziom kończy się i zostaje przeniesiony na kolejny poziom.

Klasa FinishPoint

```
using UnityEngine;


public class FinishPoint : MonoBehaviour
{
    public GameManager gameManager;

    private void OnTriggerEnter2D(Collider2D collision)
    {

    /// Funkcja reaguje z colidarem jeżeli napotka tag gracza
    /// jeżeli wykryje gracza wygrywa poziom


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

Klasa odpowiada czy z flagą końca poziomu zderzył się gracz jeżeli tak gracz wygrywa poziom.

Klasa GameOver

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

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

    public SceneFader sceneFader;

    public void Restart()
    {

        /// Rozjasnienie poziomu po restarcie


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

    public void Menu()
    {

    /// Przyciemnienie poziomu


        sceneFader.FadeTo(menuStringName);
```

```
    }
}
```

Klasa odpowiada za nawigowanie między scenami przegranej a głównym menu.

Klasa GroundChecker

```
using UnityEngine;

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

    private void OnTriggerEnter2D(Collider2D collision)
    {

    /// Sprawdzanie czy gracz porusz się po podlozu

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

    private void OnTriggerExit2D(Collider2D collision)
    {

    /// Sprawdzenie czy gracz znajduje się na podlozu


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

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


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

Klasa ta odpowiada za sprawdzenie czy gracz znajduje się na ziemi. Pokazuje też czy gracz nie znajduje się w powietrzu.

Klasa LivesUI

```
using UnityEngine;
using UnityEngine.UI;

public class LivesUI : MonoBehaviour
{
    public Text livesText;

    private void Update()
    {

    /// Przepisanie punktow zycia na tekst


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

Klasa odpowiada za pokazanie ile gracz ma aktualnie punktów życia w interfejsie gracza.

Klasa MainMenu

```
using UnityEngine;

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

    public SceneFader sceneFader;

    public void Play()
    {
    /// Wczytanie pierwszego poziomu


        sceneFader.FadeTo(levelToLoad);
    }

    public void Quit()
    {
    /// Zamkniecie aplikacji


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

Klasa ta odpowiada za menu główne gry. W niej jest zawarte przejście do pierwszego poziomu gry oraz możliwość opuszczenia rozgrywki.

Klasa PauseMenu

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class PauseMenu : MonoBehaviour
{
    public GameObject ui;

    public string menuSceneName = "MainMenu";

    public SceneFader sceneFader;

    private void Update()
    {

    /// Sprawdzenie czy zostal wciśnięty przycisk Esc


        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 odpowiada za pauzowanie gry. Gdy gracz spauzuje gre zatrzymuje się czas gry obraz zostaje przyciemniony i wyświetlone zostaje menu wyboru opcji.

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()
    {

    /// Ustawienie stanow animacji poruszania się i stania na ziemi


        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)
        {

    /// Warunek w którym jest sprawdzany jest kierunek w jakim gracz jest
///ustawiony


            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 ta odpowiada za wszystkie animacje gracza. Sprawdza w którą stronę przemieszcza się gracz i czy znajduje się on na ziemi. Na tej podstawie ustalane są przez silnik animacje dla gracza.

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()
    {
    /// Sprawdzenie czy gracz wcisnal spacje
    /// oraz czy znajduje się na ziemi


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

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


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

Klasa ta odpowiada za skok gracza. Domyślnie ta klasa jest ustawiona na false by nie zmieniac Tagu gracza. Można w niej zmienić moc skoku.

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 ta odpowiada za poruszanie się gracza. Są w niej zawarte wektory w których porusza się gracz. Można zmienić w niej szybkość poruszania się gracza.

Klasa PlayerStats

```
using UnityEngine;
public class PlayerStats : MonoBehaviour
{
    public static int Points;
    public static int Health;

    /// Ustawienie danych poczatkowych dla gracza


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

```
        Points = startPoints;
        Health = startHealth;
    }
}
```
Klasa ta odpowiada za początkowe statystyki gracza na każdym poziomie.

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)
        {
    /// Petla w ktorej punkty sa dodawane do gracza


            point++;
            pointsText.text = point.ToString();

    /// Powrót do pola po czasie


            yield return new WaitForSeconds(.01f);
        }


    }
}
```

Klasa ta odpowiada za zliczanie punktów jakie gracz zdobył podczas przechodzenia konkretnego poziomu. Jest w niej zawarta rama czasowa by nie pokazywać od razu wszystkich punktów lecz by można było dać animacje zliczania punktów.

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()
    {
    /// Powtor do korutyny


        StartCoroutine(FadeIn());
    }

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


        StartCoroutine(FadeOut(scene));
    }

    IEnumerator FadeIn()
    {
        float t = 1f;

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


            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)
        {
    /// Petla w ktorej jest przejscie ze stanu pauzy


            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 ta odpowiada za przemieszczanie się między poszczególnymi scenami np. menu główne pauza, game over. Są w niej zawarte wznowienia rozgrywki i pauzy. Służy ona do zarządzania scenami.

Klasa Weapon

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

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

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


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

    void Shoot()
    {
    /// Funkcja generuje pocisk który strzela w zależności w ktora strone patrzy
///przeciwnik


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

Klasa ta odpowiada za strzelanie gracza. Nadaje ona przedkosc pocisku, kierunek w którym ma lecieć i sprawdza czy grać wcisnął przycisk strzału.

Klasa Enemy

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

public class Enemy : MonoBehaviour
{
    [SerializeField] private GameObject deathEffect;
    public GameObject pointA;
    public GameObject pointB;
    private Rigidbody2D rb;
    private Transform currentPoint;
    public float speed;


    private void Start()
    {
        rb = GetComponent<Rigidbody2D>();
        currentPoint = pointB.transform;
    }

    private void Update()
    {
        Vector2 point = currentPoint.position - transform.position;
        if(currentPoint == pointB.transform)
        {
            rb.velocity = new Vector2(speed, 0);
        }else
        {
            rb.velocity = new Vector2(-speed, 0);
        }

        if (Vector2.Distance(transform.position, currentPoint.position) < 0.5f &&
currentPoint == pointB.transform)
        {
            Flip();
            currentPoint = pointA.transform;
        }
        if (Vector2.Distance(transform.position, currentPoint.position) < 0.5f &&
currentPoint == pointA.transform)
        {
            Flip();
            currentPoint = pointB.transform;
        }
    }

    private void Flip()
    {
        Vector3 localScale = transform.localScale;
        localScale.x *= -1;
        transform.localScale = localScale;
    }

    private void OnDrawGizmos()
    {
        Gizmos.DrawWireSphere(pointA.transform.position, 0.5f);
```

```
        Gizmos.DrawWireSphere(pointB.transform.position, 0.5f);
        Gizmos.DrawLine(pointA.transform.position, pointB.transform.position);

    }

    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.CompareTag("Bullet") ||
collision.CompareTag("EnemyDestroyer"))
        {
            GameObject effect = (GameObject)Instantiate(deathEffect,
transform.position, Quaternion.identity);
            Destroy(effect, 2f);
            PlayerStats.Points += 50;
            Destroy(gameObject);
        }

    }
}
```

Klasa ta odpowiada za zarządzanie przeciwnikami. Są w niej zawarte dane w jakim kierunku ma iść przeciwnik. Kolizja z pociskiem gracza lub czy gracz skoczył na głowę przeciwnika w celu jego eliminacji. Po pokonaniu przeciwnika gracz dostaje 50 punktów.

Klasa Checkpoint

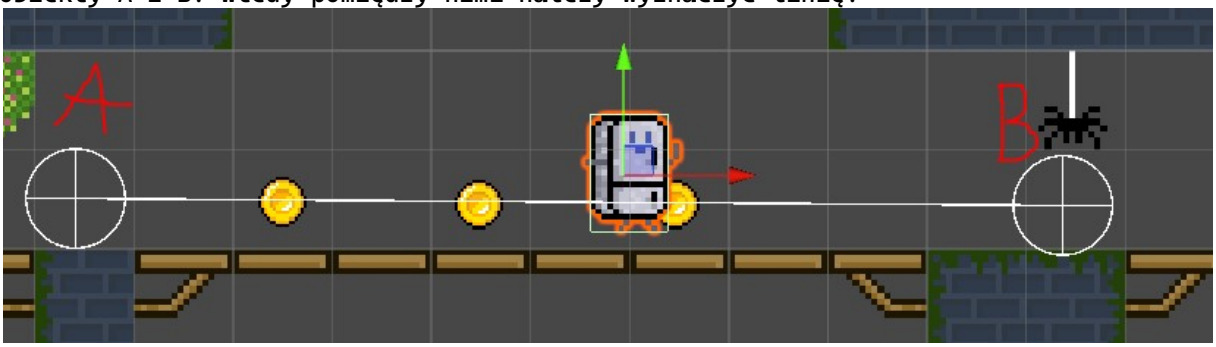
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Checkpoint : MonoBehaviour
{
    private bool isReached = false;
    public Sprite activeCheckpoint;
    public SpriteRenderer sprite;

    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.CompareTag("Player"))
        {
            sprite.sprite = activeCheckpoint;
        }
    }

    private void OnTriggerExit2D(Collider2D collision)
    {
        if (collision.CompareTag("Player"))
        {
            isReached = true;
        }
    }

    public bool IsReached()
    {
        return isReached;
    }

    public Vector3 GetCheckpointPosition()
    {
```

```
        return transform.position;
    }
}
```
Klasa ta odpowiada za checkpointy w grze. Jeżeli gracz wejdzie w checkpoint zmienia się animacja flagi oraz zostaje zapisana pozycja gracza na miejsce gdzie jest checkpoint. Zawarta w niej jest funkcja która sprawdza czy checkpoint nie został już wcześniej zajęty. Jeżeli został to stan checkpointa nie zostaje nadpisywany.

Klasa CheckPointSystem

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

public class CheckPointSystem : MonoBehaviour
{
    private Vector3 respawnPoint;
    private PlayerMovement playerMovement;

    void Start()
    {
        respawnPoint = transform.position;
        playerMovement = GetComponent<PlayerMovement>();
    }

    private void OnCollisionEnter2D(Collision2D collision)
    {
        if (collision.gameObject.tag == "Enemy")
        {
            Respawn();
        }
    }

    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.CompareTag("Checkpoint"))
        {
            if(!collision.gameObject.GetComponent<Checkpoint>().IsReached())
            {
                respawnPoint =
collision.gameObject.GetComponent<Checkpoint>().GetCheckpointPosition();
            }
        }
    }

    private void Respawn()
    {
        transform.position = respawnPoint;
        PlayerStats.Health--;

        // Zatrzymaj ruch gracza po odrodzeniu
        playerMovement.StopMovement();
    }
}
```
Klasa ta odpowiada za zarządzanie chceckpointami. Sprawdza ona czy grać nie zderzył się z przeciwnikiem jeżeli tak to odradza go w ostatnim chceckpoincie gdzie gracz był, odejmuje ona wtedy życie gracza i zatrzymuje go by gracz nie wyleciał w nieodpowiednim kierunku.

Aby wyznaczyć drogę po której ma się poruszać przeciwnik, należy zrobić dwa obiekty A i B. Wtedy pomiędzy nimi należy wyznaczyć linię.



Przeciwnik powinien poruszać się po wyznaczonym torze. Należy też wyznaczyć kierunek w którym przeciwnik ma się poruszać. Jeżeli wyznaczymy od lewej do prawej a kierunek przeciwnika będzie odwrotny będziemy mieli animację chodzenia do tyłu przeciwnika lub możliwe będzie że obiekt nie znajdzie punktu odbicia i cały czas będzie szedł w jedną stronę.