Projekt PO - Grupa 1

Założyciele - Dawid Feler, Konrad Kozłowski, Jan Jarząbek

Temat: gra roguelike z wykorzystaniem języka C#

Opis projektu:

gra roguelike, w której pokonujemy przeciwników, przechodząc z pomieszczenie do pomieszczenia.
Gracz oraz przeciwnicy posiadają własne animacje w zależności od stanu, w którym się znajdują oraz trzy osobne mapy. Dodatkowo przeciwnicy posiadają własne AI, które pozwala im gonić gracza oraz patrolować określony obszar.
Dodatkowo występują drobne interakcje z przedmiotami oraz prosty interfejs, na którym wyświetlane jest życie oraz ilość monet.
Gra wykorzystuje silnik Unity, projekt 2D, wersja 2018.

Opisane najważniejsze skrypty

Skrypty gracza:

1. **Poruszanie się, animacje oraz życie**

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

// stan, w którym znajduje się gracz
public enum PlayerState
{
    walk,
    attack,
    interact,
    stagger,
    idle
}
```

```
public class PlayerMovement : MonoBehaviour
{
    // dla systemu poruszania się i animacji
    public PlayerState currentState;
    public float speed;
    private Rigidbody2D myRigidbody;
    private Vector3 change;
    private Animator animator;
    public SpriteRenderer receivedItemSprite;

    // dla systemu zdrowia i ekwipunku
    public FloatValue currentHealth;
    public SignalSend playerHealthSignal;
    public Inventory playerInventory;

    // dla pozycji startowej przy przejściach
    public VectorValue startingPosition;


    // Start is called before the first frame update
    void Start()
    {
// określenie podstaw gracza przy starcie
        currentState = PlayerState.walk;
        animator = GetComponent<Animator>();
        myRigidbody = GetComponent<Rigidbody2D>();
        animator.SetFloat("moveX", 0);
        animator.SetFloat("moveY", -1);
        transform.position = startingPosition.initialValue;
    }

    // Update is called once per frame
    void Update()
    {
        // sprawdzenie interakcji
        if (currentState == PlayerState.interact)
        {
            return;
        }
```

```
// poruszanie przy użyciu klawiatury
        change = Vector3.zero;
        change.x = Input.GetAxisRaw("Horizontal");
        change.y = Input.GetAxisRaw("Vertical");


// pętla stanu przy ataku

        if(Input.GetButtonDown("attack") && currentState != PlayerState.attack &&
currentState != PlayerState.stagger)
        {
            StartCoroutine(AttackCo());
        }
        else if (currentState == PlayerState.walk || currentState == PlayerState.idle)
        {
            UpdateAnimationAndMove();
        }
    }
//funkcja odpowiedzialna za atak
    private IEnumerator AttackCo()
    {
        animator.SetBool("attacking", true);
        currentState = PlayerState.attack;
        yield return null;
        animator.SetBool("attacking", false);
        yield return new WaitForSeconds(.33f);
        if (currentState != PlayerState.interact)
        {
            currentState = PlayerState.walk;
        }
    }
// sygnał do podniesienia przedmiotu
    public void RaiseItem()
    {
        if (playerInventory.currentItem != null)
        {
            if (currentState != PlayerState.interact)
            {
                animator.SetBool("receive item", true);
                currentState = PlayerState.interact;
                receivedItemSprite.sprite = playerInventory.currentItem.itemSprite;
            }
```

```
            else
            {
                animator.SetBool("receive item", false);
                currentState = PlayerState.idle;
                receivedItemSprite.sprite = null;
                playerInventory.currentItem = null;
            }
        }
    }


    // kod obsługujący animację
    void UpdateAnimationAndMove()
    {
        if (change != Vector3.zero)
        {
            MoveCharacter();
            animator.SetFloat("moveX", change.x);
            animator.SetFloat("moveY", change.y);
            animator.SetBool("moving", true);

        }
        else
        {
            animator.SetBool("moving", false);
        }
    }
// funkcja poruszająca postać i obliczająca wartość prędkości
    void MoveCharacter()
    {
        change.Normalize();
        myRigidbody.MovePosition(
            transform.position + change * speed * Time.deltaTime
            );
    }
```

// funkcja odejmująca życie gracza

```
    public void Knock(float knockTime, float damage)
    {
        currentHealth.RuntimeValue -= damage;
        playerHealthSignal.Raise();

        if (currentHealth.RuntimeValue > 0)
        {
            StartCoroutine(KnockCo(knockTime));
        }
        else
        {
            this.gameObject.SetActive(false);
        }
    }
```

// funkcja odpowiadająca za odbicia się po uderzeniu

```
    private IEnumerator KnockCo(float knockTime)
    {
        if (myRigidbody != null)
        {
            yield return new WaitForSeconds(knockTime);
            myRigidbody.velocity = Vector2.zero;
            currentState = PlayerState.idle;
            myRigidbody.velocity = Vector2.zero;
        }
    }
}
```

2. **System obejmujący wyświetlanie życia**

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

```
// podanie wartości oraz rodzaje serc
public class HeartManager : MonoBehaviour
{
    public Image[] hearts;
    public Sprite fullHeart;
    public Sprite halfFullHeart;
    public Sprite emptyHeart;
    public FloatValue heartContainers;
    public FloatValue playerCurrentHealth;

    // Start is called before the first frame update
    void Start()
    {
        InitHearts();
    }
// funkcja wyświetlająca startową ilość życia
    public void InitHearts()
    {
        for(int i = 0; i < heartContainers.initialValue; i++)
        {
            hearts[i].gameObject.SetActive(true);
            hearts[i].sprite = fullHeart;
        }
    }
// funkcja odpowiadająca za aktualizację paska zdrowia
    public void UpdateHearts()
    {
        float tempHealth = playerCurrentHealth.RuntimeValue / 2;
        for(int i = 0; i< heartContainers.initialValue; i++)
        {
            if(i <= tempHealth-1)
            {
                //Full Heart
                hearts[i].sprite = fullHeart;
            }else if ( i >= tempHealth)
            {
                //Empty Heart
                hearts[i].sprite = emptyHeart;
            }
            else
            {
                //Half full heart
```

```
            hearts[i].sprite = halfFullHeart;
        }
      }
   }
}
```

Skrypty przeciwnika:

1. **Skrypt klasy przeciwnik**

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

//stan przeciwnika
public enum EnemyState
{
   idle,
   walk,
   attack,
   stagger
}
//bazowe wartości przeciwnika
public class Enemy : MonoBehaviour
{

   public EnemyState currentState;
   public FloatValue maxHealth;
   public float health;
   public string enemyName;
   public int baseAttack;
   public float moveSpeed;
//odczytanie życia przeciwnika
   private void Start()
   {
      health = maxHealth.initialValue;
   }
```

```
   //funkcja odpowiadająca za życie przeciwnika
private void TakeDamage(float damage)
   {
      health -= damage;
      if(health <= 0)
      {
         this.gameObject.SetActive(false);
      }
   }
//odbicie przeciwnika po uderzeniu i odebranie życia
   public void Knock(Rigidbody2D myRigidbody, float knockTime, float damage)
   {
      StartCoroutine(KnockCo(myRigidbody, knockTime));
      TakeDamage(damage);
   }
//funkcja licząca jak daleko odbić przeciwnika
   private IEnumerator KnockCo(Rigidbody2D myRigidbody, float knockTime)
   {
      if (myRigidbody != null)
      {
         yield return new WaitForSeconds(knockTime);
         myRigidbody.velocity = Vector2.zero;
         currentState = EnemyState.idle;
         myRigidbody.velocity = Vector2.zero;
      }
   }
}
```

2. **Przeciwnik “LOG”**

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


//dziedziczenie i potrzebne wartości
public class log : Enemy
{
    public Rigidbody2D myRigidbody;
    public Transform target;
    public float chaseRadius;
    public float attackRadius;
    public Transform homePosition;
    public Animator anim;

    private void Awake()
    {
        health = maxHealth.initialValue;
    }

    // Start is called before the first frame update
    void Start()
    {
        currentState = EnemyState.idle;
        myRigidbody = GetComponent<Rigidbody2D>();
        anim = GetComponent<Animator>();
        target = GameObject.FindWithTag("Player").transform;
        anim.SetBool("wakeUp", true);
    }

    // Update is called once per frame
    void FixedUpdate()
    {
        CheckDistance();
    }
```

```
//zasięg, w którym przeciwnik do nas podejdzie
    public virtual void CheckDistance()
    {
        if (Vector3.Distance(target.position,
            transform.position) <= chaseRadius && Vector3.Distance(target.position,
transform.position) > attackRadius)
        {
            if (currentState != EnemyState.stagger)
            {
                Vector3 temp = Vector3.MoveTowards(transform.position, target.position,
moveSpeed * Time.deltaTime);

                changeAnim(temp - transform.position);
                myRigidbody.MovePosition(temp);
                ChangeState(EnemyState.walk);
                anim.SetBool("wakeUp", true);
            }
        }
        else if (Vector3.Distance(target.position,
                        transform.position) > chaseRadius)
        {
            anim.SetBool("wakeUp", false);
        }
    }
//animacja chodzenia i spania
    private void SetAnimFloat(Vector2 setVector)
    {
        anim.SetFloat("moveX", setVector.x);
        anim.SetFloat("moveY", setVector.y);
    }
    public void changeAnim(Vector2 direction)
    {
        if (Mathf.Abs(direction.x) > Mathf.Abs(direction.y))
        {
            if (direction.x > 0)
            {
                SetAnimFloat(Vector2.right);
            }
            else if (direction.x < 0)
            {
                SetAnimFloat(Vector2.left);
            }
```

```
        }
        else if (Mathf.Abs(direction.x) < Mathf.Abs(direction.y))
        {
            if (direction.y > 0)
            {
                SetAnimFloat(Vector2.up);
            }
            else if (direction.y < 0)
            {
                SetAnimFloat(Vector2.down);
            }
        }
    }
//zmiana stanu (pomoc)
    private void ChangeState(EnemyState newState)
    {
        if (currentState != newState)
        {
            currentState = newState;
        }
    }
}
```

Skrypty ogólne:

1. **Skrypt kamery**

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

public class CameraMovement : MonoBehaviour
{
    public Transform target;
    public float smoothing;
    public Vector2 maxPosition;
    public Vector2 minPosition;
```

```
// Start is called before the first frame update
//kamera na pozycji gracza
    void Start()
    {
        transform.position = new Vector3(target.position.x, target.position.y,
transform.position.z);
    }

    // Update is called once per frame
//podążanie za graczem oraz płynne podążanie
    void LateUpdate()
    {
        if(transform.position != target.position)
        {
            Vector3 targetPosition = new Vector3(target.position.x, target.position.y,
transform.position.z);

            targetPosition.x = Mathf.Clamp(targetPosition.x, minPosition.x,
maxPosition.x);
            targetPosition.y = Mathf.Clamp(targetPosition.y, minPosition.y,
maxPosition.y);

            transform.position = Vector3.Lerp(transform.position, targetPosition,
smoothing);
        }
    }
}
```

2. **Odbicie (ang. knockback)**

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

public class Knockback : MonoBehaviour {

    public float thrust;
    public float knockTime;
    public float damage;
```

//sprawdzenie rodzaju obiektu (czy to przeciwnik) oraz obliczanie przesunięcia

```
private void OnTriggerEnter2D(Collider2D other)
   {
      if (other.gameObject.CompareTag("breakable") &&
this.gameObject.CompareTag("Player"))
      {
         other.GetComponent<Pot>().Smash();
      }
      if (other.gameObject.CompareTag("enemy") ||
other.gameObject.CompareTag("Player"))
      {
         Rigidbody2D hit = other.GetComponent<Rigidbody2D>();
         if(hit != null)
         {
            Vector2 difference = hit.transform.position - transform.position;
            difference = difference.normalized * thrust;
            hit.AddForce(difference, ForceMode2D.Impulse);
            if (other.gameObject.CompareTag("enemy") && other.isTrigger)
            {
               hit.GetComponent<Enemy>().currentState = EnemyState.stagger;
               other.GetComponent<Enemy>().Knock(hit, knockTime, damage);
            }
            if(other.gameObject.CompareTag("Player"))
            {
               if (other.GetComponent<PlayerMovement>().currentState !=
PlayerState.stagger)
               {
                  hit.GetComponent<PlayerMovement>().currentState =
PlayerState.stagger;
                  other.GetComponent<PlayerMovement>().Knock(knockTime,
damage);
               }
            }
         }
      }
   }
}
```

3. **Skrypt menu**

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

//określenie wartości dla przycisków
public class MainMenu : MonoBehaviour
{
    public void NewGame()
    {
        SceneManager.LoadScene("SampleScene");
    }

    public void Quit()
    {
        Application.Quit();
    }
}
```

Przydatne linki:

Unity Learn
Unity
Wykorzystane grafiki na zerowej licencji