**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| СОГЛАСОВАНО  Научный сотрудник МЛ ИССА факультета компьютерных наук, канд. техн. наук  \_\_\_\_\_\_\_\_\_\_\_ О.В. Максименкова  "\_\_\_" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г. | |  | УТВЕРЖДАЮ  Академический руководитель образовательной программы "Программная инженерия"  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Шилов  "\_\_\_" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020 г. | |
| |  |  | | --- | --- | | Подп. и дата |  | | Инв. № дубл. |  | | Взам. Инв. № |  | | Подп. и дата |  | | Инв. № подл. |  | | **КОМПЬЮТЕРНАЯ ИГРА ВДОХНОВЛЁННАЯ LEGENDS OF ZELDA**  **Текст программы**  **ЛИСТ УТВЕРЖДЕНИЯ**  **RU.17701729.04.01-01 ТЗ 01-1-ЛУ**  Исполнитель  Студент группы БПИ191  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д.П.Архаров  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020 г. | | | | |
|  | |  | | |

УТВЕРЖДЕН

RU.17701729.04.01-01 ТЗ 01-1-ЛУ

|  |  |
| --- | --- |
| Подп. и дата |  |
| Инв. № дубл. |  |
| Взам. инв. № |  |
| Подп. и дата |  |
| Инв. №подл |  |

**КОМПЬЮТЕРНАЯ ИГРА ВДОХНОВЛЁННАЯ LEGENDS OF ZELDA**

**Текст программы  
  
RU.17701729.04.01-01 ТЗ 01-1  
  
Листов 35**

СОДЕРЖАНИЕ

[ТЕКСТ ПРОГРАММЫ 4](#_Toc40631273)

[Класс BoolValue 4](#_Toc40631274)

[Класс CameraMovement 4](#_Toc40631275)

[Класс Coin 5](#_Toc40631276)

[Класс CoinTextManager 6](#_Toc40631277)

[Класс ContextClue 6](#_Toc40631278)

[Класс Door 7](#_Toc40631279)

[Класс DungeonEnemyRoom 8](#_Toc40631280)

[Класс Enemy 11](#_Toc40631281)

[Класс FloatValue 13](#_Toc40631282)

[Класс Heart 13](#_Toc40631283)

[Класс HeartManager 14](#_Toc40631284)

[Класс Interactable 15](#_Toc40631285)

[Класс Inventory 16](#_Toc40631286)

[Класс Item 17](#_Toc40631287)

[Класс ProgressBar 17](#_Toc40631288)

[Класс log 18](#_Toc40631289)

[Класс Loot 19](#_Toc40631290)

[Класс LootTable 20](#_Toc40631291)

[Класс OverworldRoom 20](#_Toc40631292)

[Класс PlayerMovement 21](#_Toc40631293)

[Класс pot 23](#_Toc40631294)

[Класс Projectile 24](#_Toc40631295)

[Класс Room 25](#_Toc40631296)

[Класс RoomMove 26](#_Toc40631297)

[Класс RoomTransfer 27](#_Toc40631298)

[Класс SceneTransition 27](#_Toc40631299)

[Класс Sign 29](#_Toc40631300)

[Класс Signal 29](#_Toc40631301)

[Класс SignalListener 30](#_Toc40631302)

[Класс Switch 31](#_Toc40631303)

[Класс TreasureChest 31](#_Toc40631304)

[Класс TurretEnemy 33](#_Toc40631305)

[Класс VectorValue 34](#_Toc40631306)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 35](#_Toc40631307)

[ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ 36](#_Toc40631308)

# ТЕКСТ ПРОГРАММЫ

## [Класс](https://github.com/isp13/Accretion) BoolValue

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

[CreateAssetMenu]

public class BoolValue : ScriptableObject, ISerializationCallbackReceiver

{

public bool initialValue;

[HideInInspector]

public bool RuntimeValue;

public void OnAfterDeserialize()

{

RuntimeValue = initialValue;

}

public void OnBeforeSerialize() { }

}

## Класс CameraMovement

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class CameraMovement : MonoBehaviour {

[Header("Position Variables")]

public Transform target;

public float smoothing;

public Vector2 maxPosition;

public Vector2 minPosition;

[Header ("Animator")]

public Animator anim;

[Header("Position Reset")]

public VectorValue camMin;

public VectorValue camMax;

// Use this for initialization

void Start () {

maxPosition = camMax.initialValue;

minPosition = camMin.initialValue;

anim = GetComponent<Animator>();

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

}

// Update is called once per frame

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);

//transform.position = Vector3.Lerp(transform.position,

// targetPosition, smoothing);

}

}

private Vector3 RoundPosition(Vector3 position)

{

float xOffset = position.x % .0625f;

if(xOffset != 0)

{

position.x -= xOffset;

}

float yOffset = position.y % .0625f;

if(yOffset != 0)

{

position.y -= yOffset;

}

return position;

}

public void BeginKick()

{

Debug.Log("Kick");

anim.SetBool("kick\_active", true);

StartCoroutine(KickCo());

}

public IEnumerator KickCo()

{

yield return null;

anim.SetBool("kick\_active", false);

}

}

## Класс Coin

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Coin : Powerup

{

public Inventory playerInventory;

// Start is called before the first frame update

void Start()

{

powerupSignal.Raise();

}

// Update is called once per frame

void Update()

{

}

public void OnTriggerEnter2D(Collider2D other)

{

if (other.CompareTag("Player") && !other.isTrigger)

{

playerInventory.coins += 1;

powerupSignal.Raise();

Destroy(this.gameObject);

}

}

}

## Класс CoinTextManager

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using TMPro;

public class CoinTextManager : MonoBehaviour

{

public Inventory playerInventory;

public TextMeshProUGUI coinDisplay;

private void Start()

{

UpdateCoinCount();

}

public void UpdateCoinCount()

{

coinDisplay.text = "" + playerInventory.coins;

}

}

## Класс ContextClue

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class ContextClue : MonoBehaviour {

public GameObject contextClue;

public bool contextActive = false;

public void ChangeContext()

{

contextActive = !contextActive;

if(contextActive)

{

contextClue.SetActive(true);

}else{

contextClue.SetActive(false);

}

}

}

## Класс Door

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public enum DoorType

{

key,

enemy,

button

}

public class Door : Interactable

{

[Header("Door variables")]

public DoorType thisDoorType;

public bool open = false;

public Inventory playerInventory;

public SpriteRenderer doorSprite;

public BoxCollider2D physicsCollider;

private void Update()

{

if(Input.GetButtonDown("attack"))

{

if(playerInRange && thisDoorType == DoorType.key)

{

//Does the player have a key?

if(playerInventory.numberOfKeys > 0)

{

//Remove a player key

playerInventory.numberOfKeys--;

//If so, then call the open method

Open();

}

}

}

}

public void Open()

{

//Turn off the door's sprite renderer

doorSprite.enabled = false;

//set open to true

open = true;

//turn off the door's box collider

physicsCollider.enabled = false;

}

public void Close()

{

//Turn off the door's sprite renderer

doorSprite.enabled = true;

//set open to true

open = false;

//turn off the door's box collider

physicsCollider.enabled = true;

}

}

## Класс DungeonEnemyRoom

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

using UnityEngine.UI;

public class DungeonEnemyRoom : DungeonRoom

{

public Door[] doors;

public GameObject dialogBox;

public Text dialogText;

public string dialog;

private void Start()

{

OpenDoors();

}

public int EnemiesActive()

{

int activeEnemies = 0;

for (int i = 0; i < enemies.Length; i++)

{

if (enemies[i].gameObject.activeInHierarchy)

{

activeEnemies++;

}

}

return activeEnemies;

}

public void CheckEnemies()

{

if (EnemiesActive() == 1)

{

OpenDoors();

dialogBox.SetActive(true);

dialogText.text = dialog;

StartCoroutine(Co\_ForEnd(2));

}

}

public override void OnTriggerEnter2D(Collider2D other)

{

if (other.CompareTag("Player") && !other.isTrigger)

{

//Activate all enemies and pots

for (int i = 0; i < enemies.Length; i++)

{

ChangeActivation(enemies[i], true);

}

for (int i = 0; i < pots.Length; i++)

{

ChangeActivation(pots[i], true);

}

CloseDoors();

StartCoroutine(Co\_OnEnter(2));

virtualCamera.SetActive(true);

}

}

public override void OnTriggerExit2D(Collider2D other)

{

if (other.CompareTag("Player") && !other.isTrigger)

{

//Deactivate all enemies and pots

//Activate all enemies and pots

for (int i = 0; i < enemies.Length; i++)

{

ChangeActivation(enemies[i], false);

}

for (int i = 0; i < pots.Length; i++)

{

ChangeActivation(pots[i], false);

}

virtualCamera.SetActive(false);

}

}

public void CloseDoors()

{

for(int i = 0; i < doors.Length; i ++)

{

doors[i].Close();

}

Debug.Log("Close Doors");

}

public void OpenDoors()

{

for (int i = 0; i < doors.Length; i++)

{

doors[i].Open();

}

Debug.Log("Open Doors");

}

private IEnumerator Co\_ForEnd(float delay)

{

yield return new WaitForSeconds(delay);

Application.Quit();

}

private IEnumerator Co\_OnEnter(float delay)

{

dialogBox.SetActive(true);

dialogText.text = "Ha-ha, you will have to destroy all enemies if you want to exit this room !";

yield return new WaitForSeconds(delay);

dialogBox.SetActive(false);

}

}

## Класс Enemy

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public enum EnemyState{

idle,

walk,

attack,

stagger

}

public class Enemy : MonoBehaviour {

[Header("State Machine")]

public EnemyState currentState;

[Header("Enemy Stats")]

public FloatValue maxHealth;

public float health;

public string enemyName;

public int baseAttack;

public float moveSpeed;

public Vector2 homePosition;

[Header("Death Effects")]

public GameObject deathEffect;

private float deathEffectDelay = 1f;

public LootTable thisLoot;

[Header("Death Signals")]

public Signal roomSignal;

private void Awake()

{

health = maxHealth.initialValue;

}

private void OnEnable()

{

transform.position = homePosition;

health = maxHealth.initialValue;

currentState = EnemyState.idle;

}

private void TakeDamage(float damage)

{

health -= damage;

if(health <= 0)

{

DeathEffect();

MakeLoot();

if (roomSignal != null)

{

roomSignal.Raise();

}

this.gameObject.SetActive(false);

}

}

private void MakeLoot()

{

if(thisLoot != null)

{

Powerup current = thisLoot.LootPowerup();

if(current != null)

{

Instantiate(current.gameObject, transform.position, Quaternion.identity);

}

}

}

private void DeathEffect()

{

if(deathEffect != null)

{

GameObject effect = Instantiate(deathEffect, transform.position, Quaternion.identity);

Destroy(effect, deathEffectDelay);

}

}

public void Knock(Rigidbody2D myRigidbody, float knockTime, float damage)

{

StartCoroutine(KnockCo(myRigidbody, knockTime));

TakeDamage(damage);

}

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;

}

}

}

## Класс FloatValue

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

[CreateAssetMenu]

public class FloatValue : ScriptableObject, ISerializationCallbackReceiver {

public float initialValue;

[HideInInspector]

public float RuntimeValue;

public void OnAfterDeserialize(){

RuntimeValue = initialValue;

}

public void OnBeforeSerialize(){}

}

## Класс Heart

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Heart : Powerup

{

public FloatValue playerHealth;

public FloatValue heartContainers;

public float amountToIncrease;

// Start is called before the first frame update

void Start()

{

}

// Update is called once per frame

void Update()

{

}

public void OnTriggerEnter2D(Collider2D other)

{

if(other.CompareTag("Player") && !other.isTrigger)

{

playerHealth.RuntimeValue += amountToIncrease;

if(playerHealth.initialValue > heartContainers.RuntimeValue \* 2f)

{

playerHealth.initialValue = heartContainers.RuntimeValue \* 2f;

}

powerupSignal.Raise();

Destroy(this.gameObject);

}

}

}

## Класс HeartManager

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

using UnityEngine.UI;

public class HeartManager : MonoBehaviour {

public Image[] hearts;

public Sprite fullHeart;

public Sprite halfFullHeart;

public Sprite emptyHeart;

public FloatValue heartContainers;

public FloatValue playerCurrentHealth;

public GameObject dialogBox;

public Text dialogText;

public string dialog;

// Use this for initialization

void Start() {

InitHearts();

}

public void InitHearts()

{

for (int i = 0; i < heartContainers.initialValue; i++)

{

hearts[i].gameObject.SetActive(true);

hearts[i].sprite = fullHeart;

}

}

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;

}

}

if (tempHealth <= 0)

{

dialogBox.SetActive(true);

dialogText.text = dialog;

StartCoroutine(Co\_ForEnd(2));

}

}

private IEnumerator Co\_ForEnd(float delay)

{

yield return new WaitForSeconds(delay);

Application.Quit();

}

}

## Класс Interactable

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Interactable : MonoBehaviour {

public Signal context;

public bool playerInRange;

// Use this for initialization

void Start () {

}

// Update is called once per frame

void Update () {

}

private void OnTriggerEnter2D(Collider2D other)

{

if (other.CompareTag("Player") && !other.isTrigger)

{

context.Raise();

playerInRange = true;

}

}

private void OnTriggerExit2D(Collider2D other)

{

if (other.CompareTag("Player") && !other.isTrigger)

{

context.Raise();

playerInRange = false;

}

}

}

## Класс Inventory

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

[CreateAssetMenu]

public class Inventory : ScriptableObject {

public Item currentItem;

public List<Item> items = new List<Item>();

public int numberOfKeys;

public int coins;

public void AddItem(Item itemToAdd)

{

// Is the item a key?

if(itemToAdd.isKey)

{

numberOfKeys++;

}

else

{

if(!items.Contains(itemToAdd))

{

items.Add(itemToAdd);

}

}

}

}

## Класс Item

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

[CreateAssetMenu]

public class Item : ScriptableObject {

public Sprite itemSprite;

public string itemDescription;

public bool isKey;

}

## Класс ProgressBar

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Knockback : MonoBehaviour {

public float thrust;

public float knockTime;

public float damage;

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);

}

}

}

}

}

}

## Класс log

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class log : Enemy {

public Rigidbody2D myRigidbody;

[Header("Target Variables")]

public Transform target;

public float chaseRadius;

public float attackRadius;

[Header("Animator")]

public Animator anim;

// Use this for initialization

void Start () {

currentState = EnemyState.idle;

myRigidbody = GetComponent<Rigidbody2D>();

anim = GetComponent<Animator>();

target = GameObject.FindWithTag("Player").transform;

}

// Update is called once per frame

void FixedUpdate () {

CheckDistance();

}

public virtual void CheckDistance()

{

if(Vector3.Distance(target.position,

transform.position) <= chaseRadius

&& Vector3.Distance(target.position,

transform.position) > attackRadius)

{

if (currentState == EnemyState.idle || currentState == EnemyState.walk

&& 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);

}

}

public 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);

}

}

}

public void ChangeState(EnemyState newState){

if(currentState != newState)

{

currentState = newState;

}

}

}

## Класс Loot

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

[System.Serializable]

public class Loot

{

public Powerup thisLoot;

public int lootChance;

}

## Класс LootTable

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

[CreateAssetMenu]

public class LootTable : ScriptableObject

{

public Loot[] loots;

public Powerup LootPowerup()

{

int cumProb = 0;

int currentProb = Random.Range(0, 100);

for(int i = 0; i < loots.Length; i ++)

{

cumProb += loots[i].lootChance;

if(currentProb <= cumProb)

{

return loots[i].thisLoot;

}

}

return null;

}

}

## Класс OverworldRoom

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class OverworldRoom : Room

{

public BoolValue starter;

public GameObject dialogBox;

public Text dialogText;

public string dialog;

// Start is called before the first frame update

void Start()

{

if (starter.RuntimeValue)

{

dialogBox.SetActive(true);

dialogText.text = dialog;

starter.RuntimeValue = false;

StartCoroutine(Co\_OnEnter(6));

}

}

private IEnumerator Co\_OnEnter(float delay)

{

yield return new WaitForSeconds(delay);

dialogBox.SetActive(false);

}

// Update is called once per frame

void Update()

{

}

}

## Класс PlayerMovement

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public enum PlayerState{

walk,

attack,

interact,

stagger,

idle

}

public class PlayerMovement : MonoBehaviour {

public PlayerState currentState;

public float speed;

private Rigidbody2D myRigidbody;

private Vector3 change;

private Animator animator;

public FloatValue currentHealth;

public Signal playerHealthSignal;

public VectorValue startingPosition;

public Inventory playerInventory;

public SpriteRenderer receivedItemSprite;

public Signal playerHit;

// Use this for initialization

void Start () {

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 () {

// Is the player in an interaction

if(currentState == PlayerState.interact)

{

return;

}

change = Vector3.zero;

change.x = Input.GetAxisRaw("Horizontal");

change.y = Input.GetAxisRaw("Vertical");

if(Input.GetButtonDown("attack") && currentState != PlayerState.attack

&& currentState != PlayerState.stagger)

{

StartCoroutine(AttackCo());

}

else if (currentState == PlayerState.walk || currentState == PlayerState.idle)

{

UpdateAnimationAndMove();

}

}

private IEnumerator AttackCo()

{

animator.SetBool("attacking", true);

currentState = PlayerState.attack;

yield return null;

animator.SetBool("attacking", false);

yield return new WaitForSeconds(.3f);

if (currentState != PlayerState.interact)

{

currentState = PlayerState.walk;

}

}

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;

}

}

}

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);

}

}

void MoveCharacter()

{

change.Normalize();

myRigidbody.MovePosition(

transform.position + change \* speed \* Time.deltaTime

);

}

public void Knock(float knockTime, float damage)

{

currentHealth.RuntimeValue -= damage;

playerHealthSignal.Raise();

if (currentHealth.RuntimeValue > 0)

{

playerHit.Raise();

StartCoroutine(KnockCo(knockTime));

}

else

{

this.gameObject.SetActive(false);

}

}

private IEnumerator KnockCo(float knockTime)

{

if (myRigidbody != null)

{

yield return new WaitForSeconds(knockTime);

myRigidbody.velocity = Vector2.zero;

currentState = PlayerState.idle;

myRigidbody.velocity = Vector2.zero;

}

}

}

## Класс pot

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class pot : MonoBehaviour {

private Animator anim;

// Use this for initialization

void Start () {

anim = GetComponent<Animator>();

}

// Update is called once per frame

void Update () {

}

public void Smash()

{

anim.SetBool("smash", true);

StartCoroutine(breakCo());

}

IEnumerator breakCo()

{

yield return new WaitForSeconds(.3f);

this.gameObject.SetActive(false);

}

}

## Класс Projectile

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Projectile : MonoBehaviour

{

[Header("Movement Stuff")]

public float moveSpeed;

public Vector2 directionToMove;

[Header("Lifetime")]

public float lifetime;

private float lifetimeSeconds;

public Rigidbody2D myRigidbody;

// Start is called before the first frame update

void Start()

{

myRigidbody = GetComponent<Rigidbody2D>();

lifetimeSeconds = lifetime;

}

// Update is called once per frame

void Update()

{

lifetimeSeconds -= Time.deltaTime;

if(lifetimeSeconds <= 0)

{

Destroy(this.gameObject);

}

}

public void Launch(Vector2 initialVel)

{

myRigidbody.velocity = initialVel \* moveSpeed;

}

public void OnTriggerEnter2D(Collider2D other)

{

if (!other.isTrigger || other.CompareTag("Player"))

{

Destroy(this.gameObject);

}

}

}

## Класс Room

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Room : MonoBehaviour

{

public Enemy[] enemies;

public pot[] pots;

public GameObject virtualCamera;

public virtual void OnTriggerEnter2D(Collider2D other)

{

if(other.CompareTag("Player") && !other.isTrigger)

{

//Activate all enemies and pots

for(int i = 0; i < enemies.Length; i ++)

{

ChangeActivation(enemies[i], true);

}

for(int i = 0; i < pots.Length; i ++)

{

ChangeActivation(pots[i], true);

}

virtualCamera.SetActive(true);

}

}

public virtual void OnTriggerExit2D(Collider2D other)

{

if (other.CompareTag("Player") && !other.isTrigger)

{

//Deactivate all enemies and pots

//Activate all enemies and pots

for (int i = 0; i < enemies.Length; i++)

{

ChangeActivation(enemies[i], false);

}

for (int i = 0; i < pots.Length; i++)

{

ChangeActivation(pots[i], false);

}

virtualCamera.SetActive(false);

}

}

public void ChangeActivation(Component component, bool activation)

{

component.gameObject.SetActive(activation);

}

}

## Класс RoomMove

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class RoomMove : MonoBehaviour {

public Vector2 cameraChange;

//public Vector2 cameraMinChange;

//public Vector2 cameraMaxChange;

public Vector3 playerChange;

private CameraMovement cam;

public bool needText;

public string placeName;

public GameObject text;

public Text placeText;

// Use this for initialization

void Start () {

cam = Camera.main.GetComponent<CameraMovement>();

}

// Update is called once per frame

void Update () {

}

private void OnTriggerEnter2D(Collider2D other)

{

if(other.CompareTag("Player") && !other.isTrigger)

{

cam.minPosition += cameraChange;

cam.maxPosition += cameraChange;

other.transform.position += playerChange;

if(needText)

{

StartCoroutine(placeNameCo());

}

}

}

private IEnumerator placeNameCo()

{

text.SetActive(true);

placeText.text = placeName;

yield return new WaitForSeconds(4f);

text.SetActive(false);

}

}

## Класс RoomTransfer

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class RoomTransfer : MonoBehaviour {

public Vector2 playerAdjust;

public Vector2 cameraAdjust;

public GameObject placeMarker;

public Text placeText;

public string placeName;

// Use this for initialization

void Start () {

}

// Update is called once per frame

void Update () {

}

private void OnTriggerEnter2D(Collider2D collision)

{

if(collision.gameObject.CompareTag("Player"))

{

GameObject player = collision.gameObject;

CameraMovement cam = Camera.main.GetComponent<CameraMovement>();

cam.minPosition += cameraAdjust;

cam.maxPosition += cameraAdjust;

player.transform.position += new Vector3(playerAdjust.x,

playerAdjust.y,

0);

StartCoroutine(textCo());

}

}

public IEnumerator textCo()

{

placeMarker.SetActive(true);

placeText.text = placeName;

yield return new WaitForSeconds(4);

placeMarker.SetActive(false);

}

}

## Класс SceneTransition

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class SceneTransition : MonoBehaviour

{

[Header("New Scene Variables")]

public string sceneToLoad;

public Vector2 playerPosition;

public VectorValue playerStorage;

public Vector2 cameraNewMax;

public Vector2 cameraNewMin;

public VectorValue cameraMin;

public VectorValue cameraMax;

public BoolValue starter;

[Header("Transition Variables")]

public GameObject fadeInPanel;

public GameObject fadeOutPanel;

public float fadeWait;

private void Awake()

{

if (fadeInPanel != null)

{

GameObject panel = Instantiate(fadeInPanel, Vector3.zero, Quaternion.identity) as GameObject;

Destroy(panel, 1);

}

}

public void OnTriggerEnter2D(Collider2D other)

{

if (other.CompareTag("Player") && !other.isTrigger)

{

playerStorage.initialValue = playerPosition;

StartCoroutine(FadeCo());

//SceneManager.LoadScene(sceneToLoad);

}

}

public IEnumerator FadeCo()

{

if (fadeOutPanel != null)

{

Instantiate(fadeOutPanel, Vector3.zero, Quaternion.identity);

}

yield return new WaitForSeconds(fadeWait);

Reset();

AsyncOperation asyncOperation = SceneManager.LoadSceneAsync(sceneToLoad);

while (!asyncOperation.isDone)

{

yield return null;

}

}

public void Reset()

{

cameraMax.initialValue = cameraNewMax;

cameraMin.initialValue = cameraNewMin;

starter.initialValue = false;

}

}

## Класс Sign

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class Sign : Interactable {

public GameObject dialogBox;

public Text dialogText;

public string dialog;

// Use this for initialization

void Start () {

}

// Update is called once per frame

void Update () {

if(Input.GetButtonDown("attack") && playerInRange)

{

if(dialogBox.activeInHierarchy)

{

dialogBox.SetActive(false);

}else{

dialogBox.SetActive(true);

dialogText.text = dialog;

}

}

}

private void OnTriggerExit2D(Collider2D other)

{

if (other.CompareTag("Player") && !other.isTrigger)

{

context.Raise();

playerInRange = false;

dialogBox.SetActive(false);

}

}

}

## Класс Signal

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

[CreateAssetMenu]

public class Signal : ScriptableObject {

public List<SignalListener> listeners = new List<SignalListener>();

public void Raise()

{

for (int i = listeners.Count - 1; i >= 0; i --)

{

listeners[i].OnSignalRaised();

}

}

public void RegisterListener(SignalListener listener)

{

listeners.Add(listener);

}

public void DeRegisterListener(SignalListener listener)

{

listeners.Remove(listener);

}

}

## Класс SignalListener

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.Events;

public class SignalListener : MonoBehaviour {

public Signal signal;

public UnityEvent signalEvent;

public void OnSignalRaised()

{

signalEvent.Invoke();

}

private void OnEnable()

{

signal.RegisterListener(this);

}

private void OnDisable()

{

signal.DeRegisterListener(this);

}

}

## Класс Switch

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Switch : MonoBehaviour

{

public bool active;

public BoolValue storedValue;

public Sprite activeSprite;

private SpriteRenderer mySprite;

public Door thisDoor;

// Start is called before the first frame update

void Start()

{

mySprite = GetComponent<SpriteRenderer>();

active = storedValue.RuntimeValue;

if(active)

{

ActivateSwitch();

}

}

public void ActivateSwitch()

{

active = true;

storedValue.RuntimeValue = active;

thisDoor.Open();

mySprite.sprite = activeSprite;

}

public void OnTriggerEnter2D(Collider2D other)

{

// Is it the player?

if(other.CompareTag("Player"))

{

ActivateSwitch();

}

}

}

## Класс TreasureChest

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class TreasureChest : Interactable {

[Header("Contents")]

public Item contents;

public Inventory playerInventory;

public bool isOpen;

public BoolValue storedOpen;

[Header("Signals and Dialog")]

public Signal raiseItem;

public GameObject dialogBox;

public Text dialogText;

[Header("Animation")]

private Animator anim;

// Use this for initialization

void Start () {

anim = GetComponent<Animator>();

isOpen = storedOpen.RuntimeValue;

if(isOpen)

{

anim.SetBool("opened", true);

}

}

// Update is called once per frame

void Update () {

if (Input.GetButtonDown("attack") && playerInRange)

{

if(!isOpen)

{

// Open the chest

OpenChest();

}else

{

// Chest is already open

ChestAlreadyOpen();

}

}

}

public void OpenChest()

{

// Dialog window on

dialogBox.SetActive(true);

// dialog text = contents text

dialogText.text = contents.itemDescription;

// add contents to the inventory

playerInventory.AddItem(contents);

playerInventory.currentItem = contents;

// Raise the signal to the player to animate

raiseItem.Raise();

// raise the context clue

context.Raise();

// set the chest to opened

isOpen = true;

anim.SetBool("opened", true);

storedOpen.RuntimeValue = isOpen;

}

public void ChestAlreadyOpen()

{

// Dialog off

dialogBox.SetActive(false);

// raise the signal to the player to stop animating

raiseItem.Raise();

}

private void OnTriggerEnter2D(Collider2D other)

{

if (other.CompareTag("Player") && !other.isTrigger && !isOpen)

{

context.Raise();

playerInRange = true;

}

}

private void OnTriggerExit2D(Collider2D other)

{

if (other.CompareTag("Player") && !other.isTrigger && !isOpen)

{

context.Raise();

playerInRange = false;

}

}

}

## Класс TurretEnemy

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class TurretEnemy : log

{

public GameObject projectile;

public float fireDelay;

private float fireDelaySeconds;

public bool canFire = true;

private void Update()

{

fireDelaySeconds -= Time.deltaTime;

if(fireDelaySeconds <= 0)

{

canFire = true;

fireDelaySeconds = fireDelay;

}

}

public override void CheckDistance()

{

if (Vector3.Distance(target.position,

transform.position) <= chaseRadius

&& Vector3.Distance(target.position,

transform.position) > attackRadius)

{

if (currentState == EnemyState.idle || currentState == EnemyState.walk

&& currentState != EnemyState.stagger)

{

if (canFire)

{

Vector2 tempVector = target.transform.position - transform.position;

tempVector.Normalize();

GameObject current = Instantiate(projectile, transform.position, Quaternion.identity);

current.GetComponent<Projectile>().Launch(tempVector);

canFire = false;

ChangeState(EnemyState.walk);

anim.SetBool("wakeUp", true);

}

}

}

else if (Vector3.Distance(target.position,

transform.position) > chaseRadius)

{

anim.SetBool("wakeUp", false);

}

}

}

## Класс VectorValue

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

[CreateAssetMenu]

public class VectorValue : ScriptableObject, ISerializationCallbackReceiver {

[Header("Value running in game")]

public Vector2 initialValue;

[Header("Value by default when starting")]

public Vector2 defaultValue;

public void OnAfterDeserialize() { initialValue = defaultValue; }

public void OnBeforeSerialize(){}

}

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. ГОСТ 19.101-77 Виды программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
2. ГОСТ 19.301-79 Программа и методика испытаний. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
3. ГОСТ 19.401-78 Текст программы. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
4. ГОСТ 19.505-79 Руководство оператора. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
5. ГОСТ 19.404-79 Пояснительная записка. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
6. ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
7. ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

# ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Изм. | Номера листов (страниц) | | | | Всего листов (страниц) в документе | № документа | Входящий № сопроводительного документа и дата | Подпись | Дата |
| измененных | замененных | новых | аннулированных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |