Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО «Алтайский государственный университет»

Школа развития цифровых компетенций «Digital Up» (цифровая кафедра)

**ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ**

**на итоговый проект «Разработка игры в жанре «Платформер» TheLostForest-master»**

**по ДПП ПП «Основы Gamedev и VR-разработки»**

|  |  |  |  |
| --- | --- | --- | --- |
| п/п  № | Задание | Исполнитель | Рабочий график (план) выполнения |
| 1 | Разработать главную цель игры и продумать механики геймплея | Дергачев Захар Константинович. Плотников Владимир Сергеевич Придчин Илья Александрович Таратынкин Евгений Евгеньевич Синицын Дмитрий Михайлович | 01.05.2025-02.05.2025 |
| 2 | Создать уровни и проработать их наполнение, написать скрипты для механик геймплея. | 03.05.2025-10.06.2025 |
| 3 | Подготовка отчета и видео-  презентации | 11.06.2025-12.06.2025 |

Руководитель проекта   
Петров Е.П.

\_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / «\_\_\_ » \_\_\_\_\_\_\_\_\_ 2025 г.

(подпись)

СОГЛАСОВАНО:

Руководитель ДПП ПП

канд. физ.-мат. наук, доцент Козлов Д. Ю.

\_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_2024 г.

(подпись)

Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО «Алтайский государственный университет»

Школа развития цифровых компетенций «Digital Up» (цифровая кафедра)

Отчет о выполнении группового итогового проекта по ДПП ПП

«Основы Gamedev и VR-разработки»

**«Разработка игры в жанре «Платформер» Путешествие через забытый лес»**

Исполнители:

Дергачев З.К.  
Плотников В.С.  
Придчин И.А.  
Таратынкин Е.Е.  
Синицын Д.М.

«\_\_\_\_» \_\_\_\_\_\_\_\_ 2025 г.

Руководитель проекта

Петров Е.П.

«\_\_\_\_» \_\_\_\_\_\_\_\_ 2025 г.

г. Барнаул, 2025

# Цель проекта

Цель данного проекта разработка игры в жанре «Платформер» Путешествие через забытый лес

# Задачи проекта и исполнители

Для достижения поставленной цели требовалось решить ряд задач:

1. Разработать концепцию игры, определить главную цель игрока и продумать основные механики геймплея, характерные для платформера (движение, прыжки, взаимодействие).
2. Создать игровые уровни, проработать их дизайн, визуальное наполнение и расстановку препятствий/платформ; написать скрипты на C# для реализации механик геймплея, управления персонажем и объектов уровня.
3. Подготовка отчёта и видео-презентации.

# Актуальность и востребованность проекта

Актуальность разработки игры "Путешествие через забытый лес" в жанре платформера обусловлена несколькими факторами. Гейминг продолжает активно развиваться и проникать в различные сферы, включая не только развлечения, но и образование, развитие когнитивных навыков и моторики. Платформеры, как один из классических и популярных жанров, обладают значительной аудиторией.

Востребованность проекта связана с тем, что платформеры предлагают доступный, динамичный и увлекательный игровой опыт. Они способствуют развитию реакции, координации движений, пространственного мышления и настойчивости в достижении цели. В условиях, когда пользователи часто ищут способы расслабиться и отвлечься от повседневных задач, хорошо продуманный платформер с приятной атмосферой и сбалансированной сложностью может стать отличным способом релаксации и получения положительных эмоций. "Путешествие через забытый лес", как представитель этого жанра, отвечает этим потребностям.

# Общие сведения о проделанной работе

Этапы выполнения проекта:

* Разработать главную цель игры и продумать механики геймплея.
* Создать уровни и проработать их наполнение, написать скрипты для механик геймплея.
* Подготовка отчёта и видео-презентации.

Для разработки игры был выбран игровой движок Unity и язык программирования C#. Этот выбор обусловлен следующими преимуществами:

* Unity предоставляет мощный и удобный редактор для создания 2D игр, включая инструменты для работы со спрайтами, анимациями, физикой (включая 2D физику) и построения уровней.
* C# является основным языком программирования для Unity. Он обладает понятным синтаксисом, обширной документацией, большим сообществом и богатым набором библиотек, необходимых для реализации игровой логики платформера. Его относительная простота освоения и производительность идеально подходят для проектов такого масштаба.
* Unity обеспечивает кроссплатформенность и облегчает процесс сборки проекта под различные устройства.

# Результаты проекта

В результате проекта была разработана игра-плафтомер, которая начинается с первого уровня (рисунок 1), где видно табличку-чекпоинт.



1 рисунок

Уровень довольно линейный и простой. По пути встречаются звездочки, мотивирующие и направляющие игрока. Вскоре нам встречается новый чекпоинт, где нас ожидает первый противник.



2 рисунок

По пути игрок сталкивается с различными этапами плафторминга, которые задействуют механики прыжков, навык игрока подбирать тайминги и т.д. Каждая такая “головоломка” предлагает уникальный вызов, где-то нужно отпрыгнуть от противника, где-то вовремя перепрыгнуть с быстрой платформы на другую. Это делает игровой процесс разнообразным и интересным.



3 рисунок



4 рисунок

Изображение выглядит как снимок экрана, текст, мультфильм

Содержимое, созданное искусственным интеллектом, может быть неверным.

5 рисунок

Изображение выглядит как снимок экрана, Компьютерная игра, Программное обеспечение для видеоигр, Стратегическая видеоигра

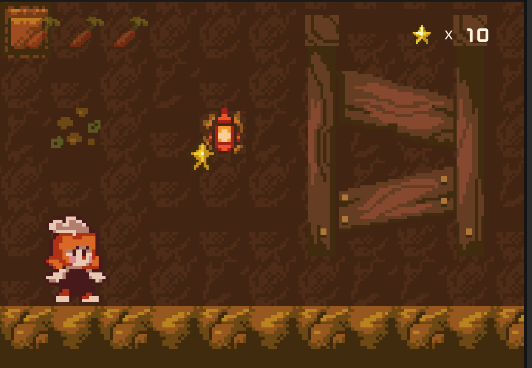
Содержимое, созданное искусственным интеллектом, может быть неверным.

Далее встречается босс уровня, и очередная награда. (рисунок 5)

Весь проект фокусируется на построении увлекательных механик, которые удерживают внимание игрока и стимулируют его продолжать проходить уровни. Игра обладает интуитивно понятным интерфейсом. В результате проект полностью соответствует поставленным целям и задачам, обеспечивая игрокам приятный и увлекательный опыт!

**Приложение 1.  
Скриншоты**











**Приложение 2.  
Скрипты**

Скрипт CharacterControl:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class CharacterControl : MonoBehaviour

{

    public float jumpForce = 2.0f;

    public float speed = 1.0f;

    private bool grounded = true;

    private bool jump;

    private bool fly;

    public static bool moving;

    private Rigidbody2D \_rigidbody2D;

    private Animator anim;

    private SpriteRenderer \_spriteRenderer;

    private float moveDirection;

    private float vertical;

    [SerializeField] public GameObject resumeScreen;

    public static bool resume = false;

    private void Awake()

    {

        anim = GetComponent<Animator>();

    }

    private void Start()

    {

        \_rigidbody2D = GetComponent<Rigidbody2D>();

        \_spriteRenderer = GetComponent<SpriteRenderer>();

        moving = true;

    }

    private void FixedUpdate()

    {

        \_rigidbody2D.linearVelocity = new Vector2(speed \* moveDirection, \_rigidbody2D.linearVelocity.y);

        if (jump)

        {

            \_rigidbody2D.linearVelocity = new Vector2(\_rigidbody2D.linearVelocity.x, jumpForce);

            jump = false;

            fly = true;

        }

        if (Enemy.isEnemyDeath)

        {

            \_rigidbody2D.linearVelocity = new Vector2(\_rigidbody2D.linearVelocity.x, jumpForce);

            Enemy.isEnemyDeath = false;

        }

        if (Climb.isClimbing)

        {

            \_rigidbody2D.gravityScale = 0f;

            \_rigidbody2D.linearVelocity = new Vector2(\_rigidbody2D.linearVelocity.x, Input.GetAxis("Vertical") \* 1.5f \* speed);

        }

        else

        {

            \_rigidbody2D.gravityScale = 0.9f;

        }

        if (Death.isDeath)

        {

            \_rigidbody2D.linearVelocity = new Vector3(0, jumpForce/3, 0);

            \_rigidbody2D.isKinematic = false;

            moving = false;

        }

        else

        {

            CharacterControl.moving = true;

        }

    }

    private void Update()

    {

        if(moving && !resume){

            if (grounded && (Input.GetAxis("Horizontal") != 0))

            {

                if (Input.GetKey(KeyCode.A) || Input.GetKey(KeyCode.LeftArrow))

                {

                    moveDirection = -1.0f;

                    \_spriteRenderer.flipX = true;

                    anim.SetFloat("speed", speed);

                }

                else if (Input.GetKey(KeyCode.D) || Input.GetKey(KeyCode.RightArrow))

                {

                    moveDirection = 1.0f;

                    \_spriteRenderer.flipX = false;

                    anim.SetFloat("speed", speed);

                }

            }

            else if (grounded)

            {

                moveDirection = 0.0f;

                anim.SetFloat("speed", 0.0f);

            }

            if (fly)

            {

                if (Input.GetAxis("Horizontal") != 0)

                {

                    if (Input.GetKey(KeyCode.A) || Input.GetKey(KeyCode.LeftArrow))

                    {

                        moveDirection = -1.0f;

                        \_spriteRenderer.flipX = true;

                    }

                    else if (Input.GetKey(KeyCode.D) || Input.GetKey(KeyCode.RightArrow))

                    {

                        moveDirection = 1.0f;

                        \_spriteRenderer.flipX = false;

                    }

                }

            }

            if (grounded && (Input.GetKey(KeyCode.W) || Input.GetKey(KeyCode.Space) || Input.GetKey(KeyCode.UpArrow)))

            {

                if (Climb.isLadder)

                {

                    Climb.isClimbing = true;

                    anim.SetTrigger("climb");

                }

                else

                {

                    jump = true;

                    anim.SetTrigger("jump");

                }

                grounded = false;

                anim.SetBool("grounded", false);

            }

            if (Climb.isLadder && !grounded && (Input.GetKey(KeyCode.A) || Input.GetKey(KeyCode.S) || Input.GetKey(KeyCode.RightArrow) || Input.GetKey(KeyCode.LeftArrow)))

            {

                grounded = true;

                anim.SetBool("grounded", true);

                Climb.isClimbing = false;

            }

        }

        if (Input.GetKeyDown(KeyCode.Escape))

        {

            if (!resume)

            {

                resume = true;

                Time.timeScale = 0;

                resumeScreen.SetActive(true);

            }

            else

            {

                resume = false;

                Time.timeScale = 1;

                resumeScreen.SetActive(false);

            }

        }

    }

    private void OnCollisionEnter2D(Collision2D other)

    {

        if (other.gameObject.CompareTag("Ground"))

        {

            grounded = true;

            fly = false;

            anim.SetBool("grounded", true);

        }

    }

}

**Приложение 3.  
Презентация**

https://disk.yandex.ru/d/9IOi8URwO247Lw

**Приложение 4.  
Видеоотчет**

<https://disk.yandex.ru/i/JL8hkP5Kkg2NHA>