Министерство образования и науки Российской Федерации

федеральное государственное бюджетное образовательное

учреждение высшего образования

«Алтайский государственный технический университет

им. И.И. Ползунова»

Факультет \_\_информационных технологий\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра \_\_\_прикладной математики\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Специальность (направление, профиль)\_\_ПИ\_\_\_\_\_\_\_\_\_\_\_\_\_

Курсовой проект

защищен с оценкой\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В.С. Троицкий

(подпись руководителя проекта)(инициалы, фамилия)

“\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

Курсовой проект

Разработка 2D игры «CursGolf» в жанре минигольф с элементами головоломки

(тема курсового проекта)

Пояснительная записка

по дисциплине \_\_Программирование\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_КП 09.03.04.21.000 ПЗ\_\_\_\_\_\_\_

(обозначение документа)

Студенты группы ПИ-32 Балдин\_А.А.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_23.09.2024

(фамилия, имя, отчество) (подпись) (дата)

Студенты группы ПИ-32 Кучковский\_С.В.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_23.09.2024

(фамилия, имя, отчество) (подпись) (дата)

Руководитель проекта преподаватель.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В.С. Троицкий

(должность, ученое звание) (подпись) (инициалы, фамилия)

БАРНАУЛ 2025

Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО «Алтайский государственный технический университет

имени И.И. Ползунова»

Факультет информационных технологий

Кафедра «Прикладная математика»

З А Д А Н И Е

на курсовой проект по дисциплине «Программирование»

студентам группы ПИ-32 Балдину Андрею Андреевичу, Кучковскому Станиславу Владимировичу

Тема курсового проекта: «Разработка 2D игры «CursGolf» в жанре минигольф с элементами головоломки »

Календарный план работы:

|  |  |  |
| --- | --- | --- |
| № этапа | Содержание этапа | Недели семестра |
| 1 | Получение задания | 1 |
| 2 | Постановка задачи. Работа с документацией | 2 |
| 3 | Проектирование программы | 3-4 |
| 4 | Реализация программы | 5-13 |
| 5 | Оформление пояснительной записки | 14 |
| 6 | Защита курсового проекта | 15-16 |

Руководитель проекта \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.С. Троицкий, доцент

подпись

Дата выдачи задания «23» сентября 2024 г.

число месяц год

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Балдин А.А.

Подпись

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Кучковский С.В.

подпись

**Содержание**

[Введение 4](#_Toc41169870)

[1 Обзор предметной области и постановка задачи 5](#_Toc41169871)

[1.1 Обзор предметной области](#_Toc41169872) 5

[1.2 Постановка задачи 5](#_Toc41169873)

[2 Проектирование 7](#_Toc41169874)

[2.1 Укрупненный алгоритм решения 7](#_Toc41169875)

[2.2 Структура данных 8](#_Toc41169876)

[3 Реализация 9](#_Toc41169877)

[3.1 Выбор средств реализации 9](#_Toc41169878)

[3.2 Структура программы 9](#_Toc41169879)

[3.3 Состав программы 9](#_Toc41169880)

[Заключение 10](#_Toc41169881)

[Список использованных источников 11](#_Toc41169882)

[Приложение А 12](#_Toc41169883)

[Приложение Б 27](#_Toc41169884)

Введение

Разработка компьютерных игр представляет собой одну из самых динамично развивающихся отраслей информационных технологий. Игры находят широкое применение не только в индустрии развлечений, но и в образовании, маркетинге и даже в медицинских исследованиях. Особый интерес вызывает жанр казуальных игр, таких как минигольф, которые отличаются простотой освоения, увлекательностью и доступностью для широкой аудитории. Этот проект ставит перед собой одну из задач создания интересного и интуитивно понятного игрового процесса, который позволит игрокам испытать элементы стратегического мышления и точности в увлекательной игровой форме.

В процессе работы над проектом будут рассмотрены ключевые аспекты разработки компьютерных игр, включая дизайн уровней, реализацию физики движения объектов, управление пользователем и создание визуального оформления. Кроме того, особое внимание будет уделено выбору инструментов и технологий для разработки, а также методам оптимизации игрового процесса.

Цель данной работы: изучение принципов объектно-ориентированного программирования и обучение их применению на реальной задаче.

Главная задача: разработать 2D игру в жанре минигольф с пиксельной графикой и элементами головоломки с помощью языка c++ с библиотекой для отрисовки графики SDL, написать прототип игры на java с использованием библиотеки LWJGL (Lightweight Java Game Library).

1 Обзор предметной области и постановка задачи

* 1. **Обзор предметной области**

Объектно-ориентированное программирование (ООП) — это подход, при котором программа рассматривается как набор объектов, взаимодействующих друг с другом. У каждого есть свойства и поведение.

Объектно-ориентированное программирование является парадигмой разработки программного продукта с использованием трех принципов: инкапсуляция, полиморфизм, наследование.

Управляемость для иерархических систем предполагает минимизацию избыточности данных (аналогичную нормализации) и их целостность, поэтому созданное удобно управляемым — будет и удобно пониматься. Таким образом, через тактическую задачу управляемости решается стратегическая задача — транслировать понимание задачи программистом в наиболее удобную для дальнейшего использования форму.

Инкапсуляция или «сокрытие информации» предполагает, что пользователь обращается к созданному в программе объекту через открытый интерфейс взаимодействия, а не напрямую, детали от пользователя скрыты.

Полиморфизм — это свойство программного кода изменять свое поведение в зависимости от ситуации, возникающей при выполнении программы. Полиморфизм позволяет повторно использовать уже написанный ранее код, что повышает эффективность разработки.

Наследование даёт возможность объекту (классу) использовать переменные и методы другого объекта (класса) как свои собственные, что также повышает эффективность разработки.

Без следования любому из этих принципов модель не будет объектно-ориентированной.

1.2 Постановка задачи

Разработать 2D игру в жанре минигольф используя парадигму объектно-ориентированного программирования под платформу Windows 10 или выше, на языке С++ с использованием библиотеки SDL2.

Игра должна включать в себя 10 уровней. Должно быть разработано меню с выбором «Game», «Info» и «Exit». В пункте «Game» пользователь начинает игру с первого уровня. В пункте «Info» пользователь может посмотреть информацию об игре. У игрока в любой момент времени должна быть возможность выйти в главное меню и начать игру заново. В пункте «Records» содержится информация о лучших результатах игрока за каждый уровень. Лучший результат – наименьшее количество ударов по мячу. Пользователь должен управлять мячом с помощью мыши. В игре необходимо правильно составить логику взаимодействия пользователя и мяча.

**Схема уровней для игры:**

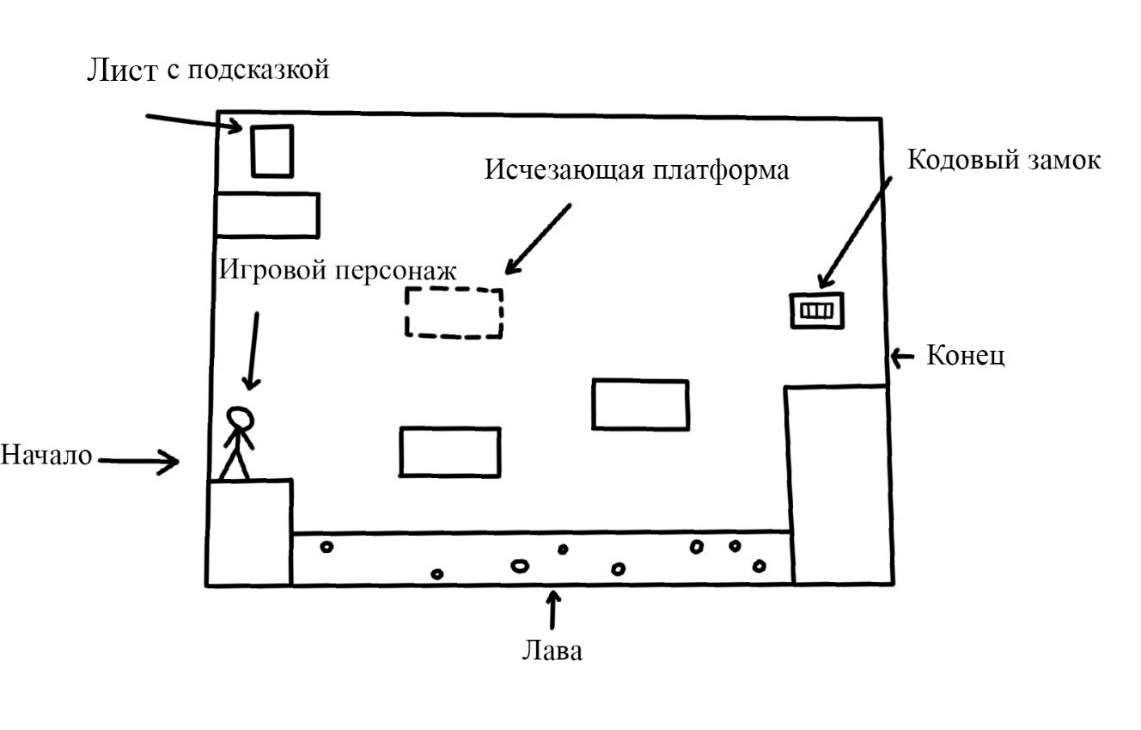


Рисунок 1 Схема первого игрового уровня

На схеме отмечены: шарик, которым управляет игрок, лунка с флажком, в которую должен попасть мячик для перехода на следующий уровень, а также различные текстуры, имеющие определенные свойства. Песок (встречается на других уровнях) и земля замедляют шарик. Через серые блоки запрещено проходить, при столкновении с ними шарик отскакивает, имитируется физика мяча. По краям карты расположено время, отсчет ведется от начала уровня, количество ударов по мячу (Strokes), ограничитель ударов (16), а также лучший результат на этом уровне. Для выхода в главное меню нужно нажать ESC.

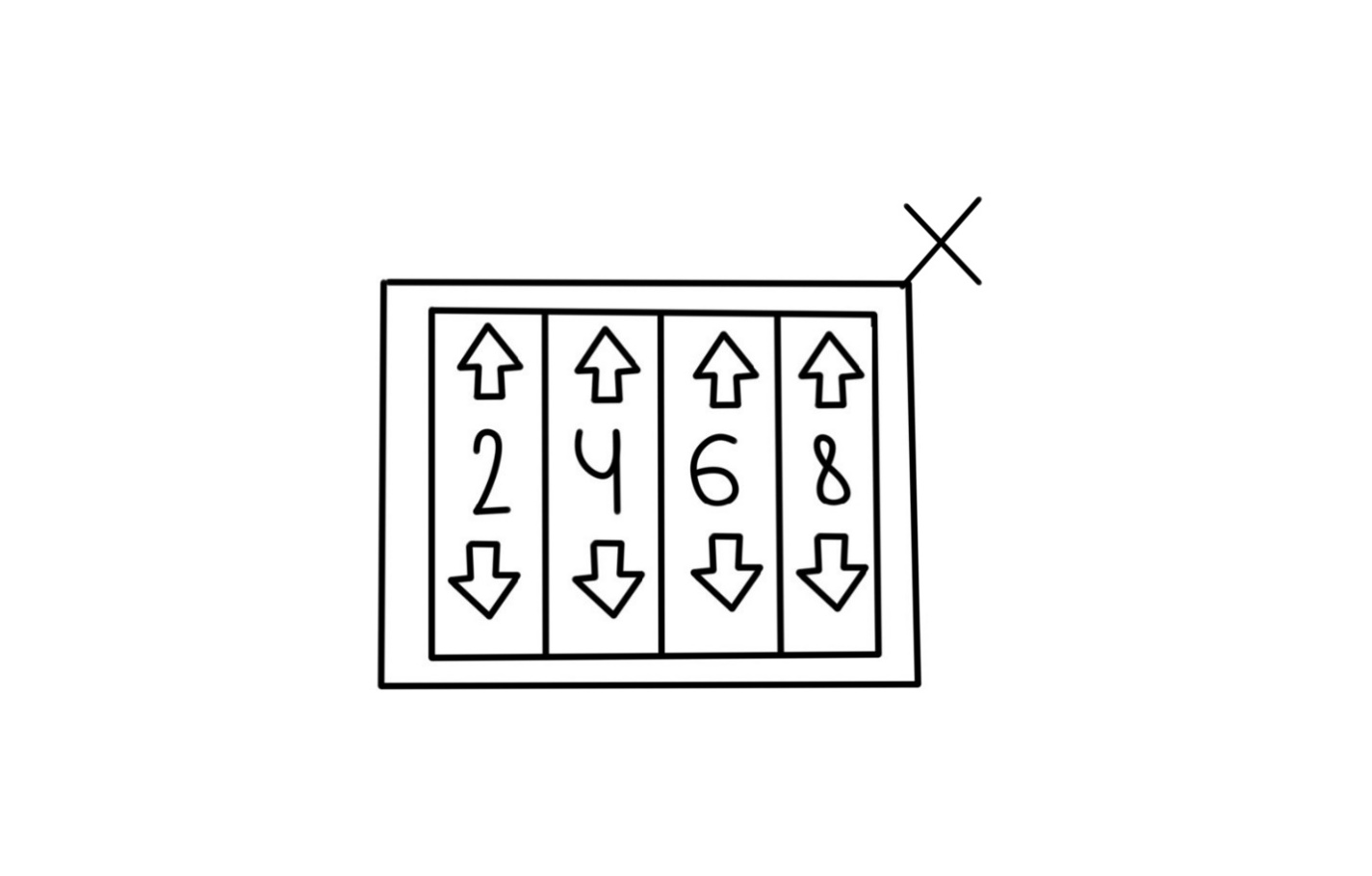


Рисунок 2 Схема второго уровня

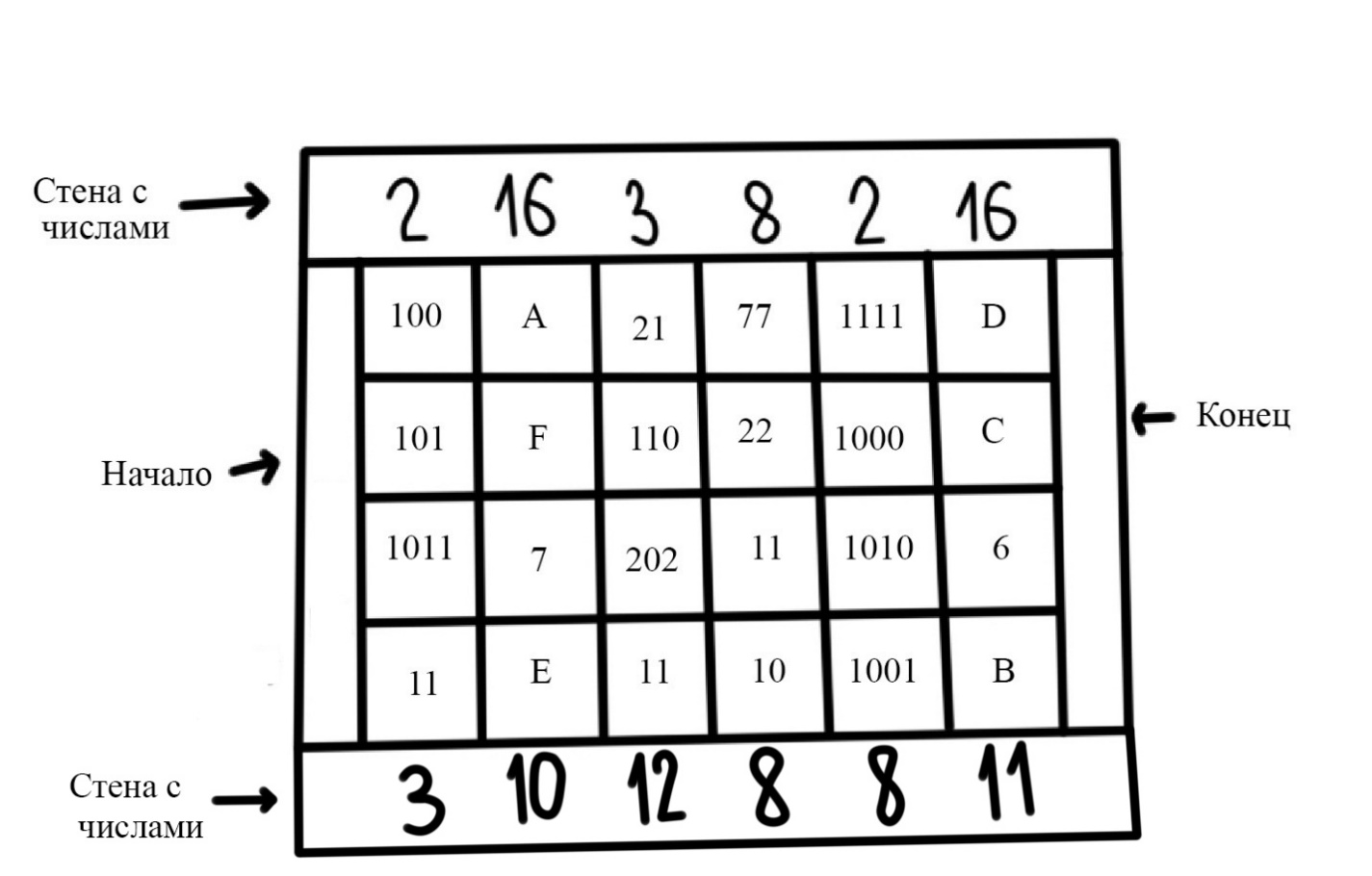


Рисунок 3 Схема третьего уровня

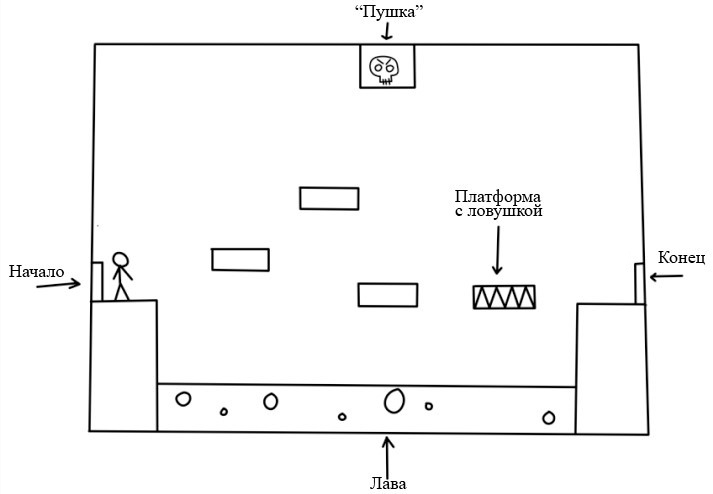


Рисунок 4 Схема четвертого уровня

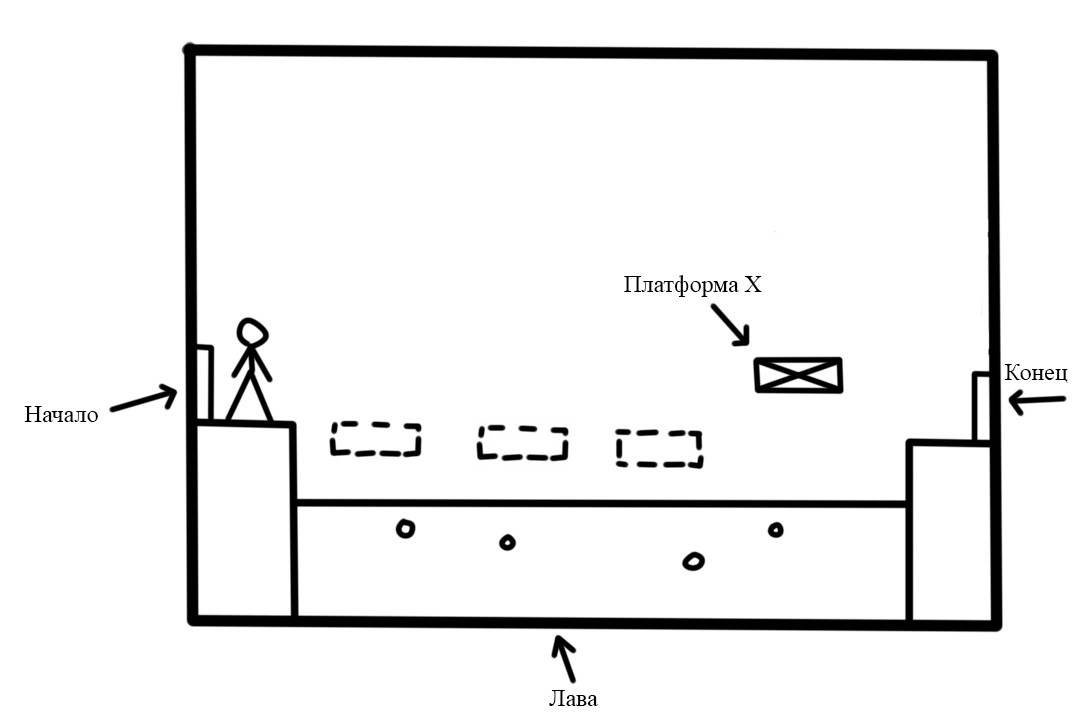


Рисунок 5 Схема 5го уровня

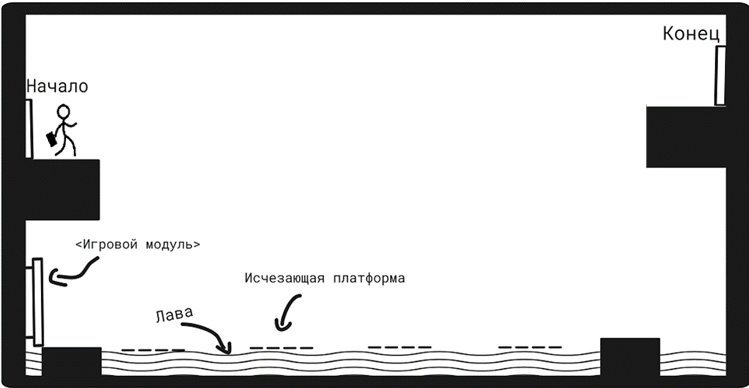


Рисунок 6 Схема 6-го уровня

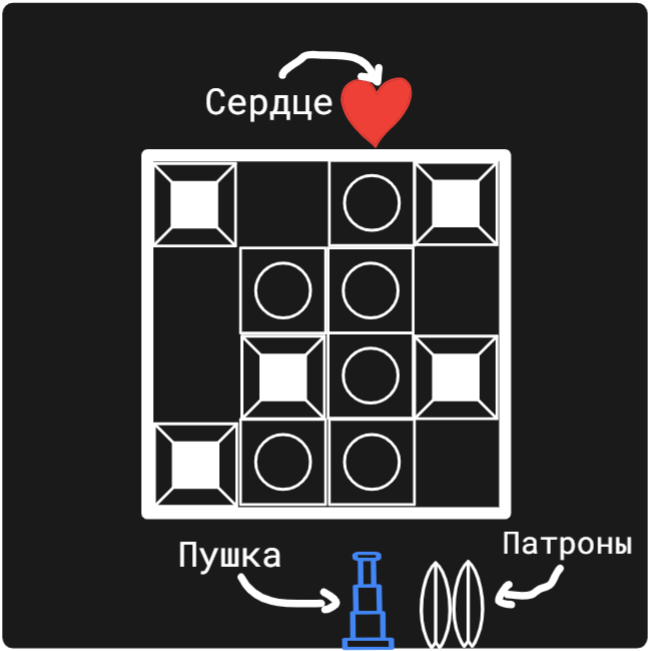


Рисунок 7 Схема 7-го уровня

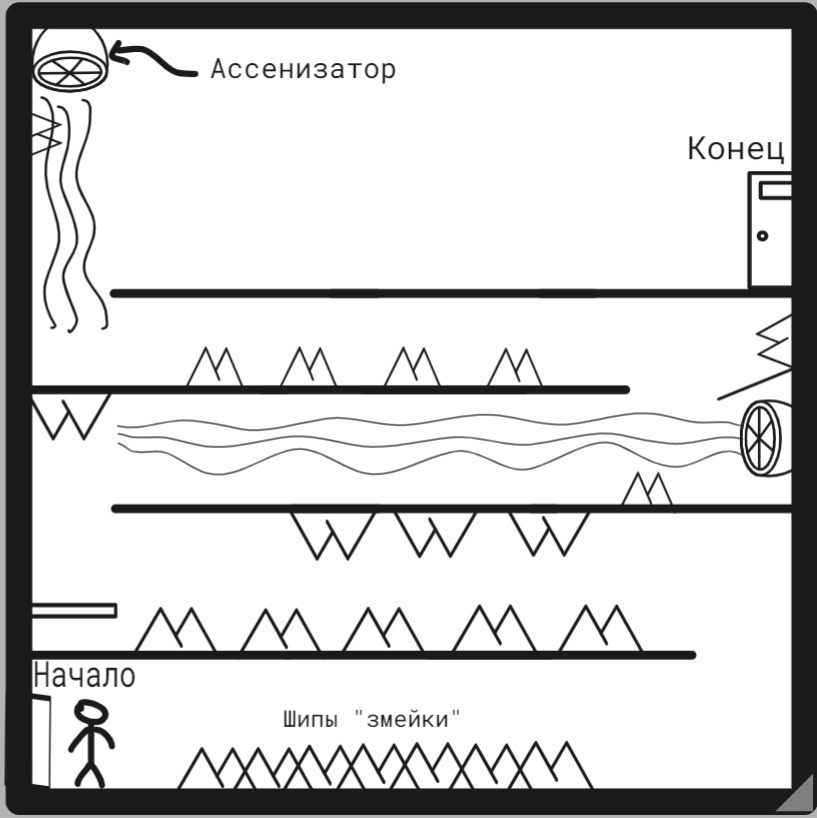


Рисунок 8 Схема 8-го уровня

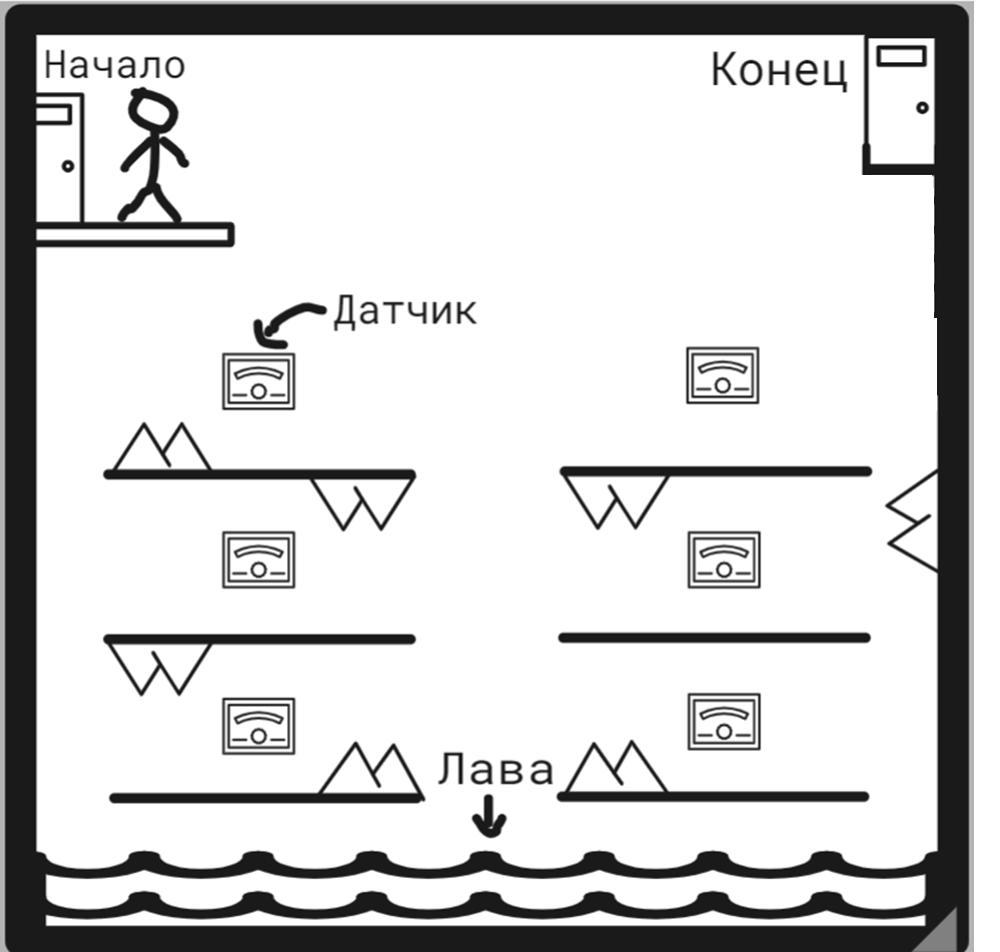


Рисунок 9 Схема 9-го уровня

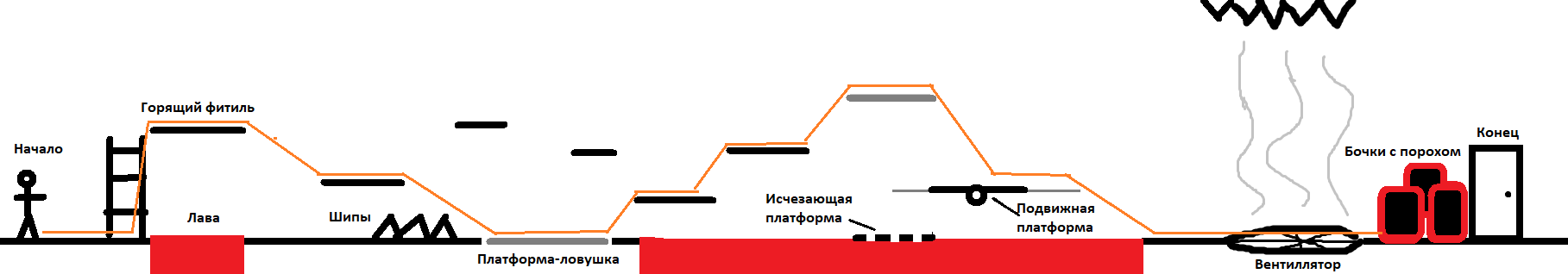


Рисунок 10 Схема 10-го уровня

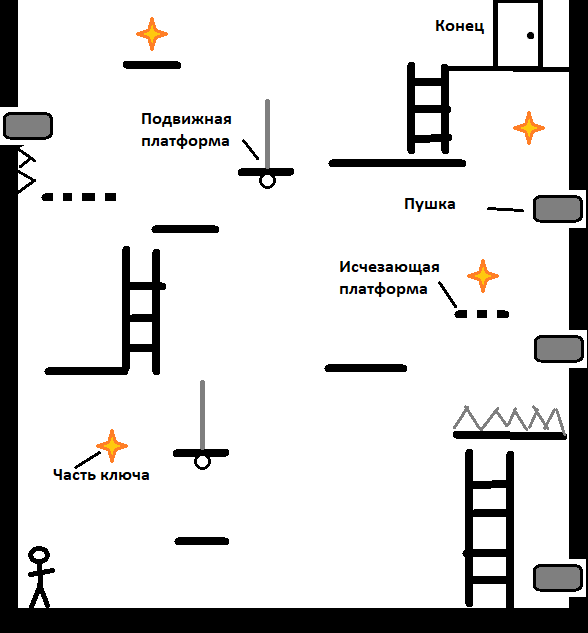


Рисунок 11 Схема меню

Меню состоит из четырех опций: «Game», «Records», «Info» , «Exit». Текущая выбранная строка выделена зеленым цветом. Для перемещения по опциям используются стрелочки на клавиатуре. Чтобы выбрать опцию, нажмите ENTER. Выбор на первой опции начинает игру с первого уровня. При выборе второй опции выводится окно, содержащая лучшие результаты игрока за каждый уровень. Третья опция выводит основную информацию об игре. Для выхода из игры выберите «Exit».

Изображение выглядит как желтый, звезда, творческий подход, оригами

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 12 Информация об игре

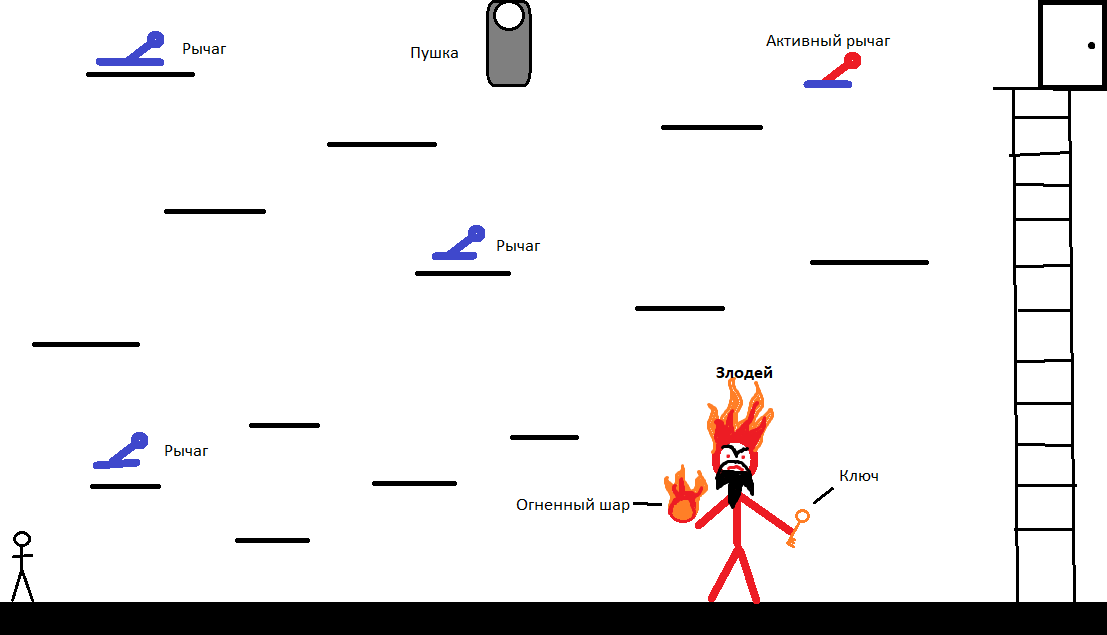


Рисунок 13 Рекорды

**2 Проектирование**

**2.1 Укрупненный алгоритм решения**

Программа работает в диалоговом режиме. При исполнении программы пользователь видит перед собой меню с возможностью выбора из 4 вариантов: «Играть», «Рекорды», «Информация» и «Выход». При нажатии кнопки «Играть» пользователь перейдет на 1ый уровень. Уровни идут по порядку, для получения доступа к каждому новому уровню необходимо пройти предыдущий. Информация о лучшем результате хранится в разделе «Рекорды». При нажатии на кнопку «Информация» пользователь увидит основную информацию об игре: управление, описание. Из любого выше описанного окна меню можно вернуться в основное меню, нажав кнопку «ESC» на клавиатуре.

2.2 Структура данных

**Распределение задач:**

Разработка классов для правильного взаимодействия пользователя с мячом, для физики мяча, для создания полноценного игрового движка; оформление курсовой работы

– Балдин А.А и Кучковский С.В.

Разработка алгоритма добавления уровней и создание уровней 6-10

– Кучковский С.В.

Разработка взаимодействия пользователя с меню, создание уровней 1-5

– Балдин А.А

**3 Реализация**

**3.1 Выбор средств реализации**

Данное ПО реализовано на языке C++ с использованием среды разработки Microsoft Visual Studio 2022 и библиотеки SDL2 для отрисовки объектов игры, включая дополнительные библиотеки SDL2\_ttf для шрифта, SDL2\_mixer звуков. Для реализации используется парадигма объектно-ориентированного программирования.

3.2 Структура программы

В программе реализованы следующие модули и взаимодействие между ними, изображенные на рисунке 14.

Рисунок 14 Структура программы в виде схемы и иерархии модулей

**3.3 Состав программы**

**Модули (функции) программы:**

**Entity-Component System (система сущностей и компонентов):**

**BallMechanic –** компонент, реализующий физику мяча.

**ColiderComponent –** компонент, реализующий механику взаимодействия объектов. При столкновении мяча и любого другого объекта (имеющий этот компонент), мяч отскакивает.

**Components.h –** заголовочный файл, содержащий в себе все компоненты

**ECS.h –** система, в которой описаны: класс Сущности, базовый класс Компонентов, а также Менеджер сущностей (класс, управляющий сущностями).

**ECS.cpp –** файл, в котором описана реализация некоторых методов, принадлежащих ECS.

**SpriteComponent.h –** компонент, реализующий работу со спрайтами (графические объекты) в игре.

**TileComponent.h –** компонент,работающий с мелкими блоками, из которых составлена карта.

**TransformComponent.h –** компонент, объединяющий в себе логику движения объектов.

**Модули, взаимодействующие с ECS:**

**Сollision.cpp и Collision.h –** модуль, описывающий наличия коллизий, т.е. взаимодействие между объектами, а если точнее, то их столкновение.

**Game.cpp и Game.h –** модуль, реализующий поэтапную логику игры: инициализацию объектов, рендер, обновление, переход на новый уровень и др.

**InfoWindow.cpp и InfoWindow.h –** модуль, который создает отдельный блок в меню с описанием программного продукта

**Main.cpp –** файл, в котором описана логика выполнения программы.

**Map.cpp и Map.h –** модуль, в котором описаны методы для работы с картой: создание, отображение, инициализация. Карта представляет собой отдельный текстовый файл, где записаны 2 двумерных массива чисел. Первый массив отвечает за загрузку текстур, 2ой – за инициализацию препятствий, за которые шарик не может заходить. Каждой цифре соответствует своя текстура. Все tileset’ы доступны в директории assets главного каталога.

**Menu.cpp и Menu.h –** модуль, отвечающий за представление меню.

**TextureManager.cpp и TextureManager.h –** модуль, отвечающий за загрузку и прорисовку текстур.

**Vector2D.cpp и Vector2D.h –** модуль, реализующий структуру данных двумерного вектора и операции с векторами.

Заключение

Был создан игровой движок, соответствующий проекту и актуальный на рынке игровой индустрии. На основе собственного движка написана игра. Реализованы уровни с различными механиками и сценариями игры.

В процессе выполнения задания была подробно изучена парадигма объектно-ориентированного программирования на языке программирования C++.

Преимущества приложения:

* Легкий интерфейс;
* Работает без доступа к сети Интернет;
* Не требует высокой производительности устройства;
* Не требует установки;
* Есть звуковое сопровождение;
* Простое интуитивно понятное управление;
* Интересный геймплей;

Недочёты приложения:

* Физика мяча требует доработки;
* Отсутствие настроек и параметров;
* Возможны некоторые мелкие баги;

Список использованных источников

1 Сайт CyberForum [Электронный ресурс]. Режим доступа: http://www.cyberforum.ru/, свободный.

2 Сайт Itch.Io [Электронный ресурс]. Режим доступа: <https://itch.io/> , свободный.

3 Сайт Learn.Unity [Электронный ресурс]. Режим доступа: <https://learn.unity.com/>, свободный.

4 Сайт Assetstore.Unity [Электронный ресурс]. Режим доступа: <https://assetstore.unity.com/>, свободный.

5 Сайт docs.unity3d [Электронный ресурс]. Режим доступа: <https://docs.unity3d.com/>, свободный.

Приложение А

**Исходный код программы**

CheckSound:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using System.IO;

using UnityEngine.UI;

public class CheckSound : MonoBehaviour

{

public float check;

public GameObject toggle;

void Start()

{

StreamReader streamReader = new StreamReader("Sound.txt");

check = System.Convert.ToSingle(streamReader.ReadLine());

if (check == 1)

toggle.GetComponent<Toggle>().isOn = true;

else toggle.GetComponent<Toggle>().isOn = false;

streamReader.Close();

}

}

Exit:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Exit : MonoBehaviour

{

public void onClick()

{

Application.Quit();

}

}

Level1:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Level1 : MonoBehaviour

{

public void onClick()

{

SceneManager.LoadScene("Lvl1");

}

}

Level2:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Level2 : MonoBehaviour

{

public void onClick()

{

SceneManager.LoadScene("Lvl2");

}

}

Level3:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Level3 : MonoBehaviour

{

public void onClick()

{

SceneManager.LoadScene("Lvl3");

}

}

Level4:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Level4 : MonoBehaviour

{

public void onClick()

{

SceneManager.LoadScene("Lvl4");

}

}

Level5:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Level5 : MonoBehaviour

{

public void onClick()

{

SceneManager.LoadScene("Level-5");

}

}

Level6:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Level6 : MonoBehaviour

{

public void onClick()

{

SceneManager.LoadScene("Level-6");

}

}

Level7:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Level7 : MonoBehaviour

{

public void onClick()

{

SceneManager.LoadScene("Level-7");

}

}

Level8:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Level8 : MonoBehaviour

{

public void onClick()

{

SceneManager.LoadScene("Level-8");

}

}

Level9:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Level9 : MonoBehaviour

{

public void onClick()

{

SceneManager.LoadScene("Level-9");

}

}

Level10:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Level10 : MonoBehaviour

{

public void onClick()

{

SceneManager.LoadScene("Level-10");

}

}

LevelFinished:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using System.IO;

using UnityEngine.SceneManagement;

using System.Linq;

public class LevelFinished : MonoBehaviour

{

List<string> lines = new List<string>();

private string filename;

public void LevelIsFinished(int scene)

{

string filename = "Levels.txt";

string item = "1";

lines = File.ReadAllLines(filename).ToList();

int index = SceneManager.GetActiveScene().buildIndex-scene;

lines.RemoveAt(index);

lines.Insert(index, item);

File.WriteAllLines(filename, lines);

}

}

LevelManager:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using System.IO;

public class LevelManager : MonoBehaviour

{

public GameObject level1;

public GameObject level2, level3, level4, level5, level6, level7, level8, level9, level10;

float[] L = new float[10];

void Start()

{

StreamReader streamReader = new StreamReader("Levels.txt");

for (int i = 0; i < 10; i++)

{

L[i] = System.Convert.ToSingle(streamReader.ReadLine());

}

if (L[1] == 1)

level2.GetComponent<Button>().interactable = true;

else level2.GetComponent<Button>().interactable = false;

if (L[2] == 1)

level3.GetComponent<Button>().interactable = true;

else level3.GetComponent<Button>().interactable = false;

if (L[3] == 1)

level4.GetComponent<Button>().interactable = true;

else level4.GetComponent<Button>().interactable = false;

if (L[4] == 1)

level5.GetComponent<Button>().interactable = true;

else level5.GetComponent<Button>().interactable = false;

if (L[5] == 1)

level6.GetComponent<Button>().interactable = true;

else level6.GetComponent<Button>().interactable = false;

if (L[6] == 1)

level7.GetComponent<Button>().interactable = true;

else level7.GetComponent<Button>().interactable = false;

if (L[7] == 1)

level8.GetComponent<Button>().interactable = true;

else level8.GetComponent<Button>().interactable = false;

if (L[8] == 1)

level9.GetComponent<Button>().interactable = true;

else level9.GetComponent<Button>().interactable = false;

if (L[9] == 1)

level10.GetComponent<Button>().interactable = true;

else level10.GetComponent<Button>().interactable = false;

}

}

MainMenu:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class MainMenu : MonoBehaviour

{

public void onClick()

{

SceneManager.LoadScene("MainMenu");

}

}

Pause:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Pause : MonoBehaviour

{

public bool pause;

public GameObject pauseGameMenu;

// Update is called once per frame

void Update()

{

if (Input.GetKeyDown(KeyCode.Escape))

{

if (pause)

{

Resume();

}

else

{

Paused();

}

}

}

public void Resume()

{

pauseGameMenu.SetActive(false);

Time.timeScale = 1f;

pause = false;

}

public void Paused()

{

pauseGameMenu.SetActive(true);

Time.timeScale = 0f;

pause = true;

}

public void LoadMenu()

{

Time.timeScale = 1f;

SceneManager.LoadScene("MainMenu");

}

}

Pause1:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Pause1 : MonoBehaviour

{

public bool pause;

public GameObject pauseGameMenu;

public GameObject gameMenu;

// Update is called once per frame

void Update()

{

if (Input.GetKeyDown(KeyCode.Escape))

{

if (pause)

{

Resume();

}

else

{

Paused();

}

}

}

public void Resume()

{

pauseGameMenu.SetActive(false);

gameMenu.SetActive(true);

Time.timeScale = 1f;

pause = false;

}

public void Paused()

{

pauseGameMenu.SetActive(true);

gameMenu.SetActive(false);

Time.timeScale = 0f;

pause = true;

}

public void LoadMenu()

{

Time.timeScale = 1f;

SceneManager.LoadScene("MainMenu");

}

}

Play:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Play : MonoBehaviour

{

public void onClick()

{

SceneManager.LoadScene("Levels");

}

}

Settings:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Settings : MonoBehaviour

{

public void onClick()

{

SceneManager.LoadScene("Settings");

}

}

Sound:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using System.IO;

public class Sound : MonoBehaviour

{

public float check;

public GameObject Audio;

void Start()

{

StreamReader streamReader = new StreamReader("Sound.txt");

check = System.Convert.ToSingle(streamReader.ReadLine());

if (check == 1)

Audio.GetComponent<AudioSource>().volume = 1;

else Audio.GetComponent<AudioSource>().volume = 0;

}

}

SoundController:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using System.IO;

public class SoundController : MonoBehaviour

{

public GameObject toggle;

private string filename;

public void CheckSound()

{

filename = "Sound.txt";

StreamWriter sw = new StreamWriter(filename);

if (toggle.GetComponent<Toggle>().isOn == true)

sw.WriteLine("1");

if (toggle.GetComponent<Toggle>().isOn == false)

sw.WriteLine("0");

sw.Close();

}

}

FinishTilelvl4:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class FinishTilelvl4 : MonoBehaviour

{

public SpriteRenderer doorspritestart;

public SpriteRenderer doorspritefinish;

public SpriteRenderer textsprite;

private void OnTriggerEnter2D(Collider2D other)

{

if (other.tag == "Hero")

{

doorspritestart.enabled = false;

doorspritefinish.enabled = false;

textsprite.enabled = true;

}

}

}

Fire:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Fire : MonoBehaviour

{

[SerializeField] private Transform post1, post2, post3, post4, post5, post6, post7, post8;

[SerializeField] private float speed = 1f;

[SerializeField] private Transform startPost;

[SerializeField] private Vector3 nextPost;

void Start()

{

nextPost = startPost.position;

}

private void OnTriggerEnter2D(Collider2D other)

{

if (other.tag == "Respawn")

{

SceneManager.LoadScene("Level-8");

speed = 0;

}

}

// Update is called once per frame

void Update()

{

transform.position = Vector3.MoveTowards(transform.position, nextPost, speed \* Time.deltaTime);

if (transform.position == post1.position)

{

nextPost = post2.position;

}

if (transform.position == post2.position)

{

nextPost = post3.position;

}

if (transform.position == post3.position)

{

nextPost = post4.position;

}

if (transform.position == post4.position)

{

nextPost = post5.position;

}

if (transform.position == post5.position)

{

nextPost = post6.position;

}

if (transform.position == post6.position)

{

nextPost = post7.position;

}

if (transform.position == post7.position)

{

nextPost = post8.position;

}

}

}

Fireball:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Fireball : MonoBehaviour

{

public Transform player;

public float speed = 4;

public float Lifetime = 6;

void Start()

{

player = GameObject.FindGameObjectWithTag("Hero").GetComponent<Transform>();

}

void Update()

{

transform.position = Vector2.MoveTowards(transform.position, player.position, speed\*Time.deltaTime);

if (Lifetime <= 0)

Destroy(this.gameObject);

else

Lifetime -= Time.deltaTime;

}

private void OnTriggerEnter2D(Collider2D other)

{

if (other.tag == "Hero")

{

other.GetComponent<Hero>().Death();

Destroy(this.gameObject);

}

}

}

Gun:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Gun : MonoBehaviour

{

public GameObject bullet;

public Transform shotPoint;

public float TimeShots;

public float StartTimeShots;

// Update is called once per frame

void Update()

{

if (TimeShots <= 0)

{

Instantiate(bullet, shotPoint.position,transform.rotation);

TimeShots = StartTimeShots;

}

else

{

TimeShots -= Time.deltaTime;

}

}

}

Gun\_mblock:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Gun\_mblock : MonoBehaviour

{

private GameObject[] MoveBlocks;

private GameObject Hearth;

private GameObject[] Bullets;

private int \_bullets = 2;

private float length = 1f;

private float \_up = 2.5f;

void Update()

{

if (Input.GetKeyDown(KeyCode.E))

{

MoveBlocks = GameObject.FindGameObjectsWithTag("MoveBlock");

Bullets = GameObject.FindGameObjectsWithTag("BulletMod");

this.GetComponent<AudioSource>().Play();

FindAndShoot();

}

}

void FindAndShoot()

{

var position = this.gameObject.transform.position;

var y = position.y;

var x = position.x;

int k = 1; int f = 1;

while (f == 1 && ((y + (k \* length)) < \_up))

{

for (int i = 0; (i < MoveBlocks.Length) && (f == 1); i++)

{

if (((y + (length \* k)) == (MoveBlocks[i].gameObject.transform.position.y)) && ((x) == (MoveBlocks[i].gameObject.transform.position.x)))

{

f = 0;

Destroy(MoveBlocks[i]);

MoveBlocks = GameObject.FindGameObjectsWithTag("MoveBlock");

\_bullets--;

Destroy(Bullets[0]);

if (\_bullets < 1) SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);

}

}

if (f == 1) k++;

}

if (f == 1)

{

Hearth = GameObject.Find("Hearth");

Destroy(Hearth);

SceneManager.LoadScene("Level-6");

GetComponent<LevelFinished>().LevelIsFinished(1);

}

}

}

GunButton:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class GunButton : MonoBehaviour

{

public float ofset;

public GameObject bullet;

public Transform shotPoint;

public float TimeShots;

public float StartTimeShots;

public GameObject Buttons;

public float TimeButton=3;

void Update()

{

if (TimeShots <= 0 && Buttons.GetComponent<Buttons>().on)

{

Instantiate(bullet, shotPoint.position, transform.rotation);

TimeShots = StartTimeShots;

}

else

{

TimeShots -= Time.deltaTime;

}

if (TimeButton <= 0)

{

Buttons.GetComponent<Buttons>().on = false;

TimeButton = 3;

}

else

{

TimeButton -= Time.deltaTime;

}

}

}

HatchPlatform:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class HatchPlatform : MonoBehaviour

{

public float Timer,Timer1;

public float StartTimer;

public SpriteRenderer sr;

void Update()

{

if (Timer <= 0)

{

if (gameObject.GetComponent<Collider2D>().enabled == true)

{

gameObject.GetComponent<Collider2D>().enabled = false;

sr.enabled = false;

}

else

{

gameObject.GetComponent<Collider2D>().enabled = true;

sr.enabled = true;

}

Timer = StartTimer;

}

else

{

Timer -= Time.deltaTime;

}

}

}

Hero:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Hero : MonoBehaviour

{

[SerializeField] public float speed = 4f;

[SerializeField] public float jumpForce = 0.3f;

[SerializeField] public static int level;

private bool isGrounded;

private bool HaveKey = false;

private int keyPiece=0;

private Rigidbody2D rb;

private Animator anim;

private SpriteRenderer sprite;

public static Hero Instance { get; set; }

void Start()

{

string sceneName = SceneManager.GetActiveScene().name;

if (sceneName == "lvl1")

{

level = 1;

speed = 4f;

jumpForce = 0.6f;

}

if (sceneName == "lvl2")

{

level = 2;

speed = 4f;

jumpForce = 0.7f;

}

if (sceneName == "lvl3")

{

level = 3;

HaveKey = true;

speed = 4f;

jumpForce = 0.8f;

}

if (sceneName == "lvl4")

{

level = 4;

speed = 1.5f;

jumpForce = 0.3f;

}

if (sceneName == "Level-5")

{

level = 5;

HaveKey = true;

speed = 5f;

jumpForce = 0.4f;

}

if (sceneName == "Level-6")

{

level = 6;

HaveKey = true;

speed = 1.5f;

jumpForce = 0.3f;

}

if (sceneName == "Level-7")

{

level = 7;

HaveKey = true;

speed = 1.5f;

jumpForce = 0.3f;

}

if (sceneName == "Level-8")

{

level = 8;

speed = 5f;

jumpForce = 0.5f;

}

if (sceneName == "Level-9")

{

level = 9;

speed = 5f;

jumpForce = 0.6f;

}

if (sceneName == "Level-10")

{

level = 10;

speed = 5f;

jumpForce = 0.6f;

}

}

private void Awake()

{

rb = GetComponent<Rigidbody2D>();

anim = GetComponent<Animator>();

sprite = GetComponentInChildren<SpriteRenderer>();

}

private void Run()

{

if (isGrounded) State = States.run;

Vector3 dir = transform.right \* Input.GetAxis("Horizontal");

transform.position = Vector3.MoveTowards(transform.position, transform.position + dir, speed \* Time.deltaTime);

sprite.flipX = dir.x < 0.0f;

}

private void Update()

{

if (isGrounded) State = States.idle;

if (Input.GetButton("Horizontal"))

Run();

if (isGrounded && Input.GetButton("Jump"))

Jump();

}

private void Jump()

{

rb.AddForce(transform.up \* jumpForce, ForceMode2D.Impulse);

}

private void CheckGround()

{

Collider2D[] colider = Physics2D.OverlapCircleAll(transform.position, 0.4f);

isGrounded = colider.Length > 1;

if (!isGrounded) State = States.jump;

}

private void FixedUpdate()

{

CheckGround();

}

private States State

{

get { return (States)anim.GetInteger("state"); }

set { anim.SetInteger("state", (int)value); }

}

private void OnTriggerEnter2D(Collider2D other)

{

if(other.tag == "Death" || other.tag == "Lava")

{

Death();

}

if (other.tag == "Key" || other.tag == "FinishTile")

{

if(level != 9)

HaveKey = true;

else

{

keyPiece++;

if(keyPiece == 4)

HaveKey = true;

}

}

if (other.tag == "Finish" && HaveKey)

{

Finish();

}

}

public void Death()

{

SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);

}

private void Finish()

{

if (level == 3)

{

SceneManager.LoadScene("lvl4");

GetComponent<LevelFinished>().LevelIsFinished(0);

}

if (level == 4)

{

SceneManager.LoadScene("Level-5");

GetComponent<LevelFinished>().LevelIsFinished(0);

}

if (level == 6)

{

SceneManager.LoadScene("Level-7");

GetComponent<LevelFinished>().LevelIsFinished(1);

}

if (level == 7)

{

SceneManager.LoadScene("Level-8");

GetComponent<LevelFinished>().LevelIsFinished(1);

}

if (level == 8)

{

SceneManager.LoadScene("Level-9");

GetComponent<LevelFinished>().LevelIsFinished(1);

}

if (level == 9)

{

SceneManager.LoadScene("Puzzle");

GetComponent<LevelFinished>().LevelIsFinished(1);

}

if (level == 10)

{

SceneManager.LoadScene("End");

GetComponent<LevelFinished>().LevelIsFinished(1);

}

level++;

}

}

public enum States

{

idle,

run,

jump,

fall

}

Key:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Key : MonoBehaviour

{

void OnTriggerEnter2D(Collider2D other)

{

if (other.tag == "Hero")

{

Destroy(this.gameObject);

}

}

}

ModulePlatform:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class ModulePlatform : MonoBehaviour

{

private GameObject[] KGreen;

private Renderer rend;

private int maxObj;

void Start()

{

maxObj = GameObject.Find("Sensors").GetComponent<DadSens>().maxObj;

rend = gameObject.GetComponent<Renderer>();

rend.enabled = false;

Debug.Log(maxObj);

gameObject.GetComponent<BoxCollider2D>().enabled = false;

}

void Update()

{

FindKGreen();

if ((KGreen.Length) == maxObj)

{

rend.enabled = true;

gameObject.GetComponent<BoxCollider2D>().enabled = true;

}

}

void FindKGreen()

{

KGreen = GameObject.FindGameObjectsWithTag("GreenS");

}

}

Modules:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Modules : MonoBehaviour

{

private GameObject[] StatBlocks;

private float base\_length = 1f; // смещение при перемещении

private GameObject[] MoveBlocks;

private float \_down = -1.5f; // нижняя граница карты

private float \_up = 1.5f; // верхняя граница карты

private float \_left = -3.5f; // граница слева

private float \_right = -0.5f; // граница справа

void Start()

{

StatBlocks = GameObject.FindGameObjectsWithTag("StatBlock");

MoveBlocks = GameObject.FindGameObjectsWithTag("MoveBlock");

}

void Update()

{

MoveBlocks = GameObject.FindGameObjectsWithTag("MoveBlock");

if (Input.GetKeyDown(KeyCode.W))

{

char KeyC = 'W';

FindAndMove(KeyC);

}

else if (Input.GetKeyDown(KeyCode.S))

{

char KeyC = 'S';

FindAndMove(KeyC);

}

else if (Input.GetKeyDown(KeyCode.D))

{

char KeyC = 'D';

FindAndMove(KeyC);

}

else if (Input.GetKeyDown(KeyCode.A))

{

char KeyC = 'A';

FindAndMove(KeyC);

}

}

void FindAndMove(char KeyC)

{

Vector2 vec;

var position = this.gameObject.transform.position;

var y = position.y;

var x = position.x;

float length = base\_length;

if (KeyC == 'A' || KeyC == 'S') length = length \* (-1);

int f = 1;

if (KeyC == 'W' || KeyC == 'S')

{

int k = 1; // коэффициент

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

{

if (((y + (length \* k)) == (MoveBlocks[i].gameObject.transform.position.y)) && ((x) == (MoveBlocks[i].gameObject.transform.position.x)))

{

k++;

i = 0;

}

}

if (((y + (length \* k)) > \_up) || ((y + (length \* k)) < \_down)) f = 0;

for (int j = 0; (j < StatBlocks.Length) && (f == 1); j++)

{

if (((y + (length \* k)) == (StatBlocks[j].gameObject.transform.position.y)) && ((x) == (StatBlocks[j].gameObject.transform.position.x)))

{

f = 0;

}

}

if (f == 1)

{

vec = new Vector2(x, y + length);

this.gameObject.transform.position = vec;

}

}

else if (KeyC == 'A' || KeyC == 'D')

{

int k = 1; // коэффициент

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

{

if (((x + (length \* k)) == (MoveBlocks[i].gameObject.transform.position.x)) && ((y) == (MoveBlocks[i].gameObject.transform.position.y)))

{

k++;

i = 0;

}

}

if (((x + (length \* k)) > \_right) || ((x + (length \* k)) < \_left)) f = 0;

for (int j = 0; (j < StatBlocks.Length) && (f == 1); j++)

{

if (((x + (length \* k)) == (StatBlocks[j].gameObject.transform.position.x)) && ((y) == (StatBlocks[j].gameObject.transform.position.y)))

{

f = 0;

}

}

if (f == 1)

{

vec = new Vector2(x + length, y);

this.gameObject.transform.position = vec;

}

}

}

}

MovPlatform:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class MovPlatform : MonoBehaviour

{

[SerializeField] private Transform post1, post2;

[SerializeField] private float speed = 3.5f;

[SerializeField] private Transform startPost;

[SerializeField] private Vector3 nextPost;

private void Start()

{

nextPost = startPost.position;

}

private void Update()

{

transform.position = Vector3.MoveTowards(transform.position,nextPost, speed\* Time.deltaTime);

if(transform.position==post1.position)

{

nextPost = post2.position;

}

if (transform.position == post2.position)

{

nextPost = post1.position;

}

}

}

NumbersRow:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class NumbersRow : MonoBehaviour

{

public SpriteRenderer zero;

public SpriteRenderer one;

public SpriteRenderer two;

public SpriteRenderer three;

public SpriteRenderer four;

public SpriteRenderer five;

public SpriteRenderer six;

public SpriteRenderer seven;

public SpriteRenderer eight;

public SpriteRenderer nine;

private SpriteRenderer[] numbers = new SpriteRenderer[10];

private void Start()

{

numbers[0] = zero;

numbers[1] = one;

numbers[2] = two;

numbers[3] = three;

numbers[4] = four;

numbers[5] = five;

numbers[6] = six;

numbers[7] = seven;

numbers[8] = eight;

numbers[9] = nine;

close();

}

public void show()

{

numbers[9].enabled = true;

}

public void close()

{

for (int i = 0; i < 10; i++)

{

numbers[i].enabled = false;

}

}

public void spinUp()

{

for (int i = 0; i < 10; i++)

{

if (numbers[i].enabled == true && i != 9)

{

numbers[i].enabled = false;

numbers[i + 1].enabled = true;

break;

}

if (numbers[i].enabled == true && i == 9)

{

numbers[i].enabled = false;

numbers[0].enabled = true;

}

}

}

public void spinDown()

{

for (int i = 0; i < 10; i++)

{

if (numbers[i].enabled == true && i != 0)

{

numbers[i].enabled = false;

numbers[i - 1].enabled = true;

break;

}

if (numbers[i].enabled == true && i == 0)

{

numbers[i].enabled = false;

numbers[9].enabled = true;

break;

}

}

}

}

Obstacle:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Obstacle : MonoBehaviour

{

[SerializeField] private Transform post1, post2;

[SerializeField] private float speed = 0.1f;

[SerializeField] private Transform startPost;

[SerializeField] private Vector3 nextPost;

private void Start()

{

nextPost = startPost.position;

}

private void Update()

{

transform.position = Vector3.MoveTowards(transform.position, nextPost, speed \* Time.deltaTime);

if (transform.position == post1.position)

{

nextPost = post2.position;

speed = 0.1f;

}

if (transform.position == post2.position)

{

nextPost = post1.position;

speed = 1f;

}

}

}

Paper:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Paper : MonoBehaviour

{

public SpriteRenderer sr;

public SpriteRenderer closecode;

private bool HeroIsHere = false;

private void Start()

{

sr.enabled = false;

closecode.enabled = false;

}

private void OnMouseDown()

{

if (HeroIsHere)

{

sr.enabled = true;

closecode.enabled = true;

}

}

private void OnTriggerEnter2D(Collider2D other)

{

if (other.tag == "Hero")

{

HeroIsHere = true;

}

}

private void OnTriggerExit2D(Collider2D other)

{

if (other.tag == "Hero")

{

HeroIsHere = false;

}

}

}

Paperlvl2:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class paperlvl2 : MonoBehaviour

{

public SpriteRenderer sr;

public SpriteRenderer closecode;

private void Start()

{

sr.enabled = false;

closecode.enabled = false;

}

private void OnMouseDown()

{

sr.enabled = true;

closecode.enabled = true;

}

}

PressNextScene:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class PressNextScene : MonoBehaviour

{

[SerializeField] private Transform hero;

private float distance; // дистанция до игрока

void Update()

{

if (Input.GetKeyDown(KeyCode.E))

{

distance = Vector3.Distance(hero.position, transform.position);

if (distance < 1.5)

{

SceneManager.LoadScene("Level-5-1");

}

}

}

}

Puzzle:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Puzzle : MonoBehaviour

{

bool move;

bool finish;

Vector2 mousePosition;

public GameObject form;

int k = 0;

void OnMouseDown()

{

if (Input.GetMouseButtonDown(0))

move = true;

}

void OnMouseUp()

{

move = false;

if (Mathf.Abs(this.transform.localPosition.x - form.transform.localPosition.x) <= 5f && (this.transform.localPosition.y - form.transform.localPosition.y) <= 5f)

{

this.transform.position = new Vector2(form.transform.position.x, form.transform.position.y);

finish = true;

if (k == 0)

{

PuzzleWin.AddElement();

k++;

}

}

}

void Update()

{

if (move && finish != true)

{

Vector2 сhangePos = transform.position;

сhangePos = Camera.main.ScreenToWorldPoint(Input.mousePosition);

transform.position = сhangePos;

}

}

}

PuzzleWin:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class PuzzleWin : MonoBehaviour

{

public int elements;

public static int myElements;

public GameObject myPuzzle;

public GameObject myPanel;

public GameObject winPanel;

// Start is called before the first frame update

void Start()

{

winPanel.SetActive(false);

elements = myPuzzle.transform.childCount;

}

// Update is called once per frame

void Update()

{

if (elements== myElements)

{

myPanel.SetActive(false);

winPanel.SetActive(true);

}

}

public static void AddElement()

{

myElements++;

}

}

Rowlslvl2:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Rowslvl2 : MonoBehaviour

{

public int row;

public bool IsItCorrect;

private static int[] counter = new int[5];

private AudioSource audiosouce;

public SpriteRenderer sr;

private float Timer = 0f;

private float StartTime = 3f;

private void Start()

{

audiosouce = GetComponent<AudioSource>();

}

// Update is called once per frame

void Update()

{

if (Timer <= 0)

{

if (sr.enabled)

{

sr.enabled = false;

}

Timer = StartTime;

}

else

{

Timer -= Time.deltaTime;

}

}

private void OnMouseDown()

{

if (counter[row] != 1)

{

if (row == 0 || counter[row - 1] == 1)

{

audiosouce.Play();

itscorrect();

}

}

}

private void itscorrect()

{

if (IsItCorrect)

{

counter[row] = 1;

if (row == 4)

{

GetComponent<LevelFinished>().LevelIsFinished(0);

SceneManager.LoadScene("lvl3");

}

}

else

{

for (int i = 0; i < 5; i++)

counter[i] = 0;

sr.enabled = true;

}

}

}

Sensor:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Sensor : DadSens

{

[SerializeField] private Transform hero;

[SerializeField] private Transform dad;

[SerializeField] private Sprite sprite1;

[SerializeField] private Sprite sprite2;

[SerializeField] private float Timing = 2f;

private SpriteRenderer spriteRenderer;

private float distance; // дистанция до игрока

public int index; // текущее положение в иерархии родителя

private float TimeAll; // время, через которое должно произойти событие

private int position; // текущее положение в иерархии массива

private int action; // исполняемое событие

public int flag = 0; // если flag = 0 - красный спрайт, flag = 1 - зелёный спрайт.

private GameObject[] KGreen;

void Start()

{

spriteRenderer = GetComponent<SpriteRenderer>();

if (spriteRenderer.sprite == null)

spriteRenderer.sprite = sprite1;

maxObj = GameObject.Find("Sensors").GetComponent<DadSens>().maxObj; // количество наследников

mass = GameObject.Find("Sensors").GetComponent<DadSens>().mass; // массив с упорядоченными номерами наследников

index = transform.GetSiblingIndex();

for (int i = 0; i < maxObj; i++)

{

if (mass[i] == index) position = i;

}

TimeAll = position \* Timing;

action = 1;

gameObject.tag = "RedS";

gameObject.name = position.ToString();

}

void Update()

{

switch (action)

{

case 0:

if ((TimeAll -= Time.deltaTime) < 0) action = 2;

break;

case 1:

if ((TimeAll -= Time.deltaTime) < 0)

{

spriteRenderer.sprite = sprite2;

TimeAll = 2f;

action = 0;

}

break;

case 2:

spriteRenderer.sprite = sprite1;

action = 3;

TimeAll = maxObj - position;

break;

case 3:

if ((TimeAll -= Time.deltaTime) < 0) action = 4;

break;

case 4:

CheckDist();

break;

case 5:

break;

case 6:

break;

}

}

void ChangeTheDamnSprite()

{

if (spriteRenderer.sprite == sprite1)

{

spriteRenderer.sprite = sprite2;

if (position == maxObj) action = 6;

else action = 5;

flag = 1;

gameObject.tag = "GreenS";

KGreen = GameObject.FindGameObjectsWithTag("GreenS");

if ((KGreen.Length - 1) != position)

{

SceneManager.LoadScene("Level-7");

}

}

}

void CheckDist()

{

distance = Vector2.Distance(hero.position, transform.position);

if (Input.GetKeyDown(KeyCode.E) && (distance < 0.5))

{

ChangeTheDamnSprite();

}

}

}

Shipi:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Shipi : MonoBehaviour

{

[SerializeField] private Transform post1, post2;

[SerializeField] private float speed = 0.1f;

[SerializeField] private Transform startPost;

[SerializeField] private Vector3 nextPost;

private void Start()

{

nextPost = startPost.position;

}

private void Update()

{

transform.position = Vector3.MoveTowards(transform.position, nextPost, speed \* Time.deltaTime);

if (transform.position == post1.position)

{

nextPost = post2.position;

speed = 0.1f;

}

if (transform.position == post2.position)

{

nextPost = post1.position;

speed = 1f;

}

}

}

SnaekyShip:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class SneakyShip : MonoBehaviour

{

[SerializeField] private Transform post1, post2;

[SerializeField] private float kSpeed = 0.0001f;

[SerializeField] private Transform startPost;

[SerializeField] private Vector3 nextPost;

private float speed = 1f;

private float TimeIer = 0f;

private float TimeAll = 0f;

private void Start()

{

TimeIer = transform.GetSiblingIndex();

TimeAll = transform.childCount;

nextPost = startPost.position;

TimeAll += TimeIer;

}

private void Update()

{

transform.position = Vector3.MoveTowards(transform.position, nextPost, speed \* Time.deltaTime + (TimeAll \* kSpeed));

if (transform.position == post1.position && (TimeIer -= Time.deltaTime) < 0)

{

nextPost = post2.position;

speed = 0.1f;

}

if (transform.position == post2.position && (TimeAll -= Time.deltaTime) < 0)

{

nextPost = post1.position;

speed = 0.5f;

TimeAll = transform.childCount;

}

}

}

StartTile:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class StartTile : MonoBehaviour

{

private void checkFinish()

{

if (Hero1.Instance.getIsFinished() == true)

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

}

}

Textlvl2:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Textlvl2 : MonoBehaviour

{

public SpriteRenderer sr;

// Start is called before the first frame update

void Start()

{

sr.enabled = false;

}

}

Textlvl4:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Textlvl4 : MonoBehaviour

{

public SpriteRenderer textsprite;

// Start is called before the first frame update

void Start()

{

GetComponentInChildren<SpriteRenderer>();

textsprite.enabled = false;

}

}

UpDownClick:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class UpDownClick : MonoBehaviour

{

private static int[] counter = new int[4];

private int[] N = new int[4];

public int row;

public NumbersRow Numbers;

public bool UpDown;

// Start is called before the first frame update

void Start()

{

N[0] = 1;

N[1] = 2;

N[2] = 3;

N[3] = 2;

}

private void OnMouseDown()

{

counter[row] += 1;

if (UpDown)

Numbers.spinUp();

else

Numbers.spinDown();

if (counter[row] > N[row])

{

CounterToZero();

}

if (counter[0] == 1 && counter[1] == 2 && counter[2] == 3 && counter[3] == 2)

{

SceneManager.LoadScene("lvl2");

GetComponent<LevelFinished>().LevelIsFinished(0);

}

}

public void CounterToZero()

{

for (int i = 0; i < 4; i++)

{

counter[i] = 0;

}

}

}

Vere:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Vere : MonoBehaviour

{

private void OnTriggerEnter2D(Collider2D other)

{

if (other.tag == "Lava")

{

Destroy(this.gameObject,0.5f);

}

}

}

XPlatform:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class XPlatform : HatchPlatform1

{

private bool isTouched = false;

// Update is called once per frame

void Update()

{

if (!isTouched)

{

HideAndReturn();

}

else

{

sr.enabled = false;

sr.gameObject.GetComponent<Collider2D>().enabled = false;

}

}

private void OnTriggerExit2D(Collider2D other)

{

if (other.tag == "Hero")

{

isTouched = true;

}

}

}

Bullet:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Bullet : MonoBehaviour

{

public float speed;

public float offset;

public LayerMask Solid;

// Update is called once per frame

void Update()

{

RaycastHit2D hitInfo = Physics2D.Raycast(transform.position, transform.up, Solid);

if(hitInfo.collider != null)

{

if (hitInfo.collider.CompareTag("Hero"))

{

hitInfo.collider.GetComponent<Hero>().Death();

}

if (hitInfo.collider.CompareTag("Evil"))

{

hitInfo.collider.GetComponent<Evil>().TakeDamage();

}

Destroy(this.gameObject);

}

transform.Translate(Vector2.left \* speed \* Time.deltaTime);

}

}

Buttons:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Buttons : MonoBehaviour

{

public SpriteRenderer sr;

public Sprite sprite1, sprite2;

public bool on = false;

private void OnTriggerEnter2D(Collider2D other)

{

if (on == true && other.tag =="Hero")

{

on = false;

}

else

{

on = true;

}

}

void Update()

{

if (on == true)

{

sr.sprite = sprite2;

}

else

{

sr.sprite = sprite1;

}

}

}

CameraController:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class CameraController : MonoBehaviour

{

[SerializeField] private Transform player;

private Vector3 pos; //координаты движения

private void Awake()

{

//проверка найден игрок или нет

if (!player)

player = FindObjectOfType<Hero>().transform;

}

// Update is called once per frame

private void Update()

{

pos = player.position; //присваивает переменной позицию игрока

pos.z = -10f; //фиксация камеры по координате z (иначе камера уйдет в 1 плоскость с игроком)

transform.position = Vector3.Lerp(transform.position, pos, Time.deltaTime); //плавно перемещает камеру

}

}

CloseCode:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class CloseCode : MonoBehaviour

{

public SpriteRenderer sr;

private SpriteRenderer closecode;

private void Awake()

{

closecode = GetComponentInChildren<SpriteRenderer>();

}

private void OnMouseDown()

{

sr.enabled = false;

closecode.enabled = false;

}

}

CloseCodeLock:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class CloseCodeLock : MonoBehaviour

{

public SpriteRenderer sr;

private SpriteRenderer closecodelock;

public SpriteRenderer OneUp;

public SpriteRenderer TwoUp;

public SpriteRenderer ThreeUp;

public SpriteRenderer FourUp;

public SpriteRenderer OneUDown;

public SpriteRenderer TwoDown;

public SpriteRenderer ThreeDown;

public SpriteRenderer FourDown;

public NumbersRow NumbersRow0;

public NumbersRow NumbersRow1;

public NumbersRow NumbersRow2;

public NumbersRow NumbersRow3;

private UpDownClick updownclick;

private void Awake()

{

closecodelock = GetComponentInChildren<SpriteRenderer>();

updownclick = new UpDownClick();

}

private void OnMouseDown()

{

sr.enabled = false;

closecodelock.enabled = false;

OneUp.enabled = false;

TwoUp.enabled = false;

ThreeUp.enabled = false;

FourUp.enabled = false;

OneUDown.enabled = false;

TwoDown.enabled = false;

ThreeDown.enabled = false;

FourDown.enabled = false;

NumbersRow0.close();

NumbersRow1.close();

NumbersRow2.close();

NumbersRow3.close();

updownclick.CounterToZero();

}

}

CodeLock:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class CodeLock : MonoBehaviour

{

public SpriteRenderer codelock;

public SpriteRenderer closecodelock;

public SpriteRenderer OneUp;

public SpriteRenderer TwoUp;

public SpriteRenderer ThreeUp;

public SpriteRenderer FourUp;

public SpriteRenderer OneUDown;

public SpriteRenderer TwoDown;

public SpriteRenderer ThreeDown;

public SpriteRenderer FourDown;

private bool HeroIsHere = false;

public NumbersRow NumbersRow0;

public NumbersRow NumbersRow1;

public NumbersRow NumbersRow2;

public NumbersRow NumbersRow3;

private void Start()

{

codelock.enabled = false;

closecodelock.enabled = false;

OneUp.enabled = false;

TwoUp.enabled = false;

ThreeUp.enabled = false;

FourUp.enabled = false;

OneUDown.enabled = false;

TwoDown.enabled = false;

ThreeDown.enabled = false;

FourDown.enabled = false;

}

private void OnMouseDown()

{

if (HeroIsHere)

{

codelock.enabled = true;

closecodelock.enabled = true;

OneUp.enabled = true;

TwoUp.enabled = true;

ThreeUp.enabled = true;

FourUp.enabled = true;

OneUDown.enabled = true;

TwoDown.enabled = true;

ThreeDown.enabled = true;

FourDown.enabled = true;

NumbersRow0.show();

NumbersRow1.show();

NumbersRow2.show();

NumbersRow3.show();

}

}

private void OnTriggerEnter2D(Collider2D other)

{

if (other.tag == "Hero")

{

HeroIsHere = true;

}

}

private void OnTriggerExit2D(Collider2D other)

{

if (other.tag == "Hero")

{

HeroIsHere = false;

}

}

}

DadSens:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class DadSens : MonoBehaviour

{

public const int N = 10;

public int[] mass = new int[N];

public int maxObj;

// Start is called before the first frame update

void Start()

{

Generator();

}

void Generator()

{

int value = 0;

maxObj = transform.childCount;

for (int i = 0; i < maxObj; i++)

{

int f = 1;

do

{

f = 1;

value = Random.Range(0, maxObj);

for (int j = 0; j < i; j++)

{

if (value == mass[j]) f = 0;

}

} while (f == 0);

mass[i] = value;

}

}

}

Evil:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Evil : MonoBehaviour

{

public int Life = 5;

public SpriteRenderer sr;

public SpriteRenderer an;

public Sprite sprite1;

private Animator anim;

private SpriteRenderer sprite;

public Transform post1, post2;

public float speed = 3f;

public Transform startPost;

public Vector3 nextPost;

public GameObject Fireball;

public GameObject Key;

public Transform shotPoint;

public float Timer=1.03f;

public GameObject Gun1, Gun2, R1, R2;

int a = 0;

private void Start()

{

anim = GetComponent<Animator>();

nextPost = startPost.position;

}

void AnimRun()

{

State = States1.EvilRun;

}

public void TakeDamage()

{

Life -= 1;

if(Life>0)

State = States1.EvilTakeHit;

if(Life == 2)

{

Destroy(Gun1);

Destroy(R1);

}

if (Life == 1)

{

Destroy(Gun2);

Destroy(R2);

}

if (Life == 0)

{

Death();

}

}

void Run()

{

transform.position = Vector3.MoveTowards(transform.position, nextPost, speed \* Time.deltaTime);

if (transform.position == post1.position && Life > 0)

{

nextPost = post2.position;

an.flipX = false;

Attack();

}

if (transform.position == post2.position && Life > 0)

{

nextPost = post1.position;

an.flipX = true;

Attack();

}

}

void Update()

{

if(Life>0) Run();

if(Life==0) sr.sprite = sprite1;

}

public void Attack()

{

State = States1.EvilAttack;

if (a == 0)

{

Instantiate(Fireball, shotPoint.position, transform.rotation);

a++;

}

if (Timer > 0)

{

Timer -= Time.deltaTime;

speed = 0;

}

else

{

speed = 3;

Timer = 1.03f;

State = States1.EvilRun;

a = 0;

}

}

void Death()

{

Instantiate(Key, new Vector3(93f, 0f, 0), Quaternion.identity);

State = States1.EvilDeath;

sr.sprite = sprite1;

gameObject.GetComponent<Collider2D>().enabled = false;

speed = 0f;

transform.position = transform.position + new Vector3(0, -0.5f, 0);

}

private States1 State

{

get { return (States1)anim.GetInteger("States1"); }

set { anim.SetInteger("States1", (int)value); }

}

}

public enum States1

{

EvilRun,

Evil\_Jump,

EvilTakeHit,

EvilDeath,

EvilAttack

}

FinishDoor:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class FinishDoor : MonoBehaviour

{

private SpriteRenderer sprite;

private Color color;

// Start is called before the first frame update

void Start()

{

sprite = GetComponent<SpriteRenderer>();

color = sprite.material.color;

color.a = 1f;

sprite.material.color = color;

}

// Update is called once per frame

void Update()

{

}

public static FinishDoor Instance { get; set; }

public void setcolor(float f)

{

color.a = f;

}

}

FinishTile:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class FinishTile : MonoBehaviour

{

public SpriteRenderer doorsprite;

public SpriteRenderer textsprite;

private GameObject startDoor;

private void OnTriggerEnter2D(Collider2D other)

{

if (other.tag == "Hero")

{

doorsprite.enabled = false;

textsprite.enabled = true;

startDoor = GameObject.Find("StartDoor");

startDoor.tag = "Finish";

}

}

}

Приложение Б

**Тесты работы программы:**



Рисунок 15 Главное меню

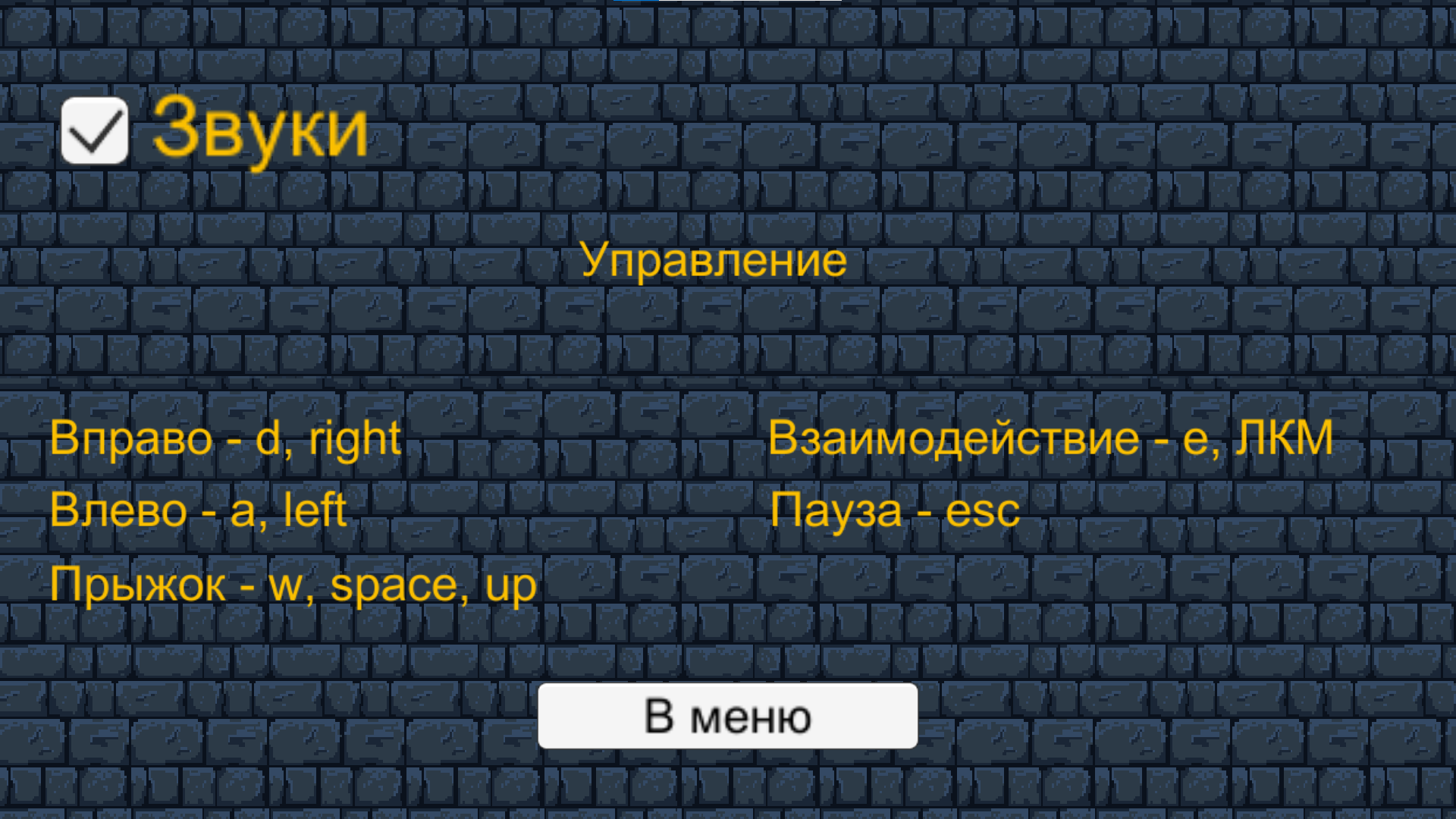


Рисунок 16 Меню «Настройки»



Рисунок 17 Меню «Выбор уровней»

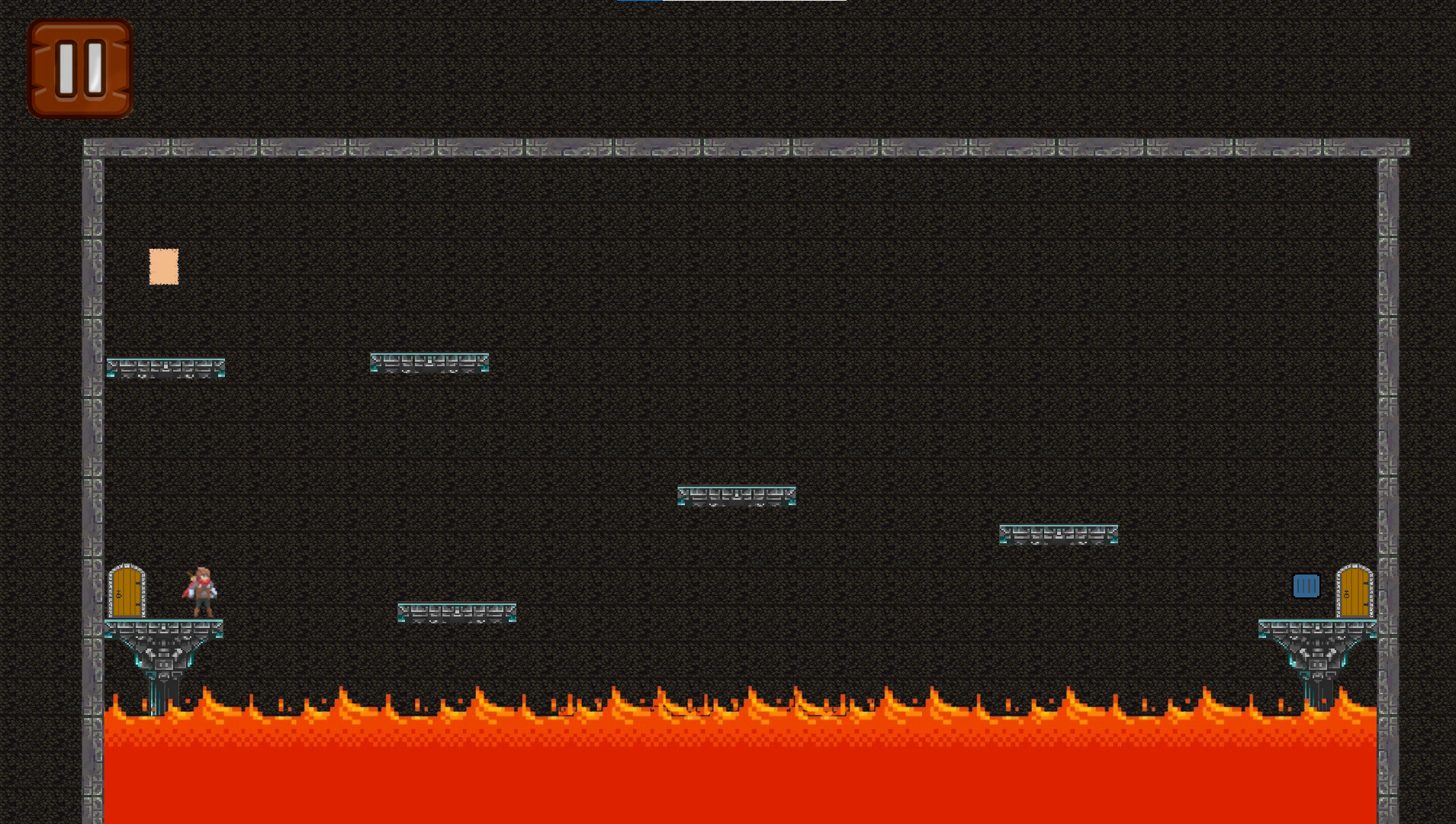


Рисунок 18 Уровень 1

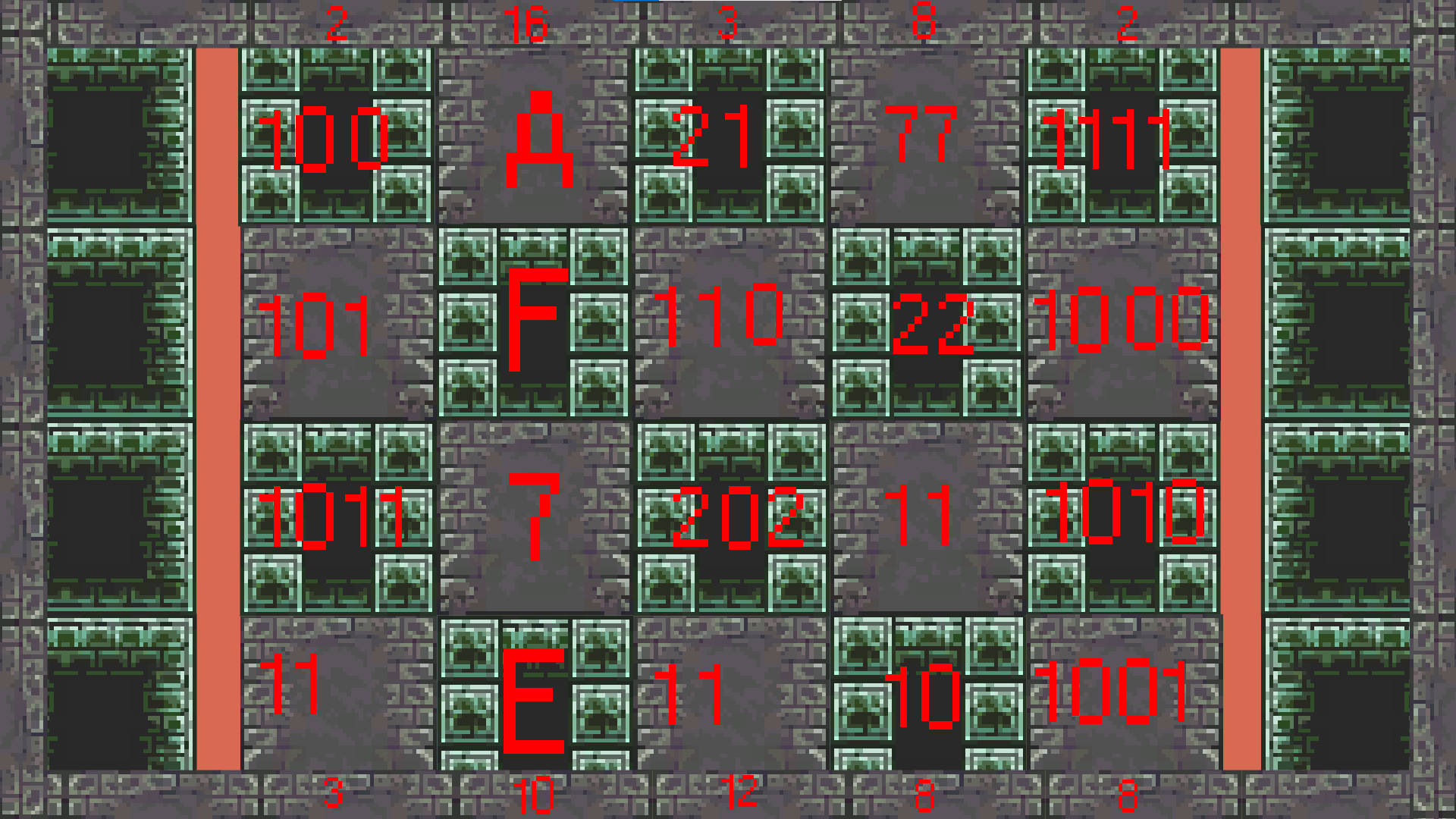


Рисунок 19 Уровень 2



Рисунок 20 Уровень 3

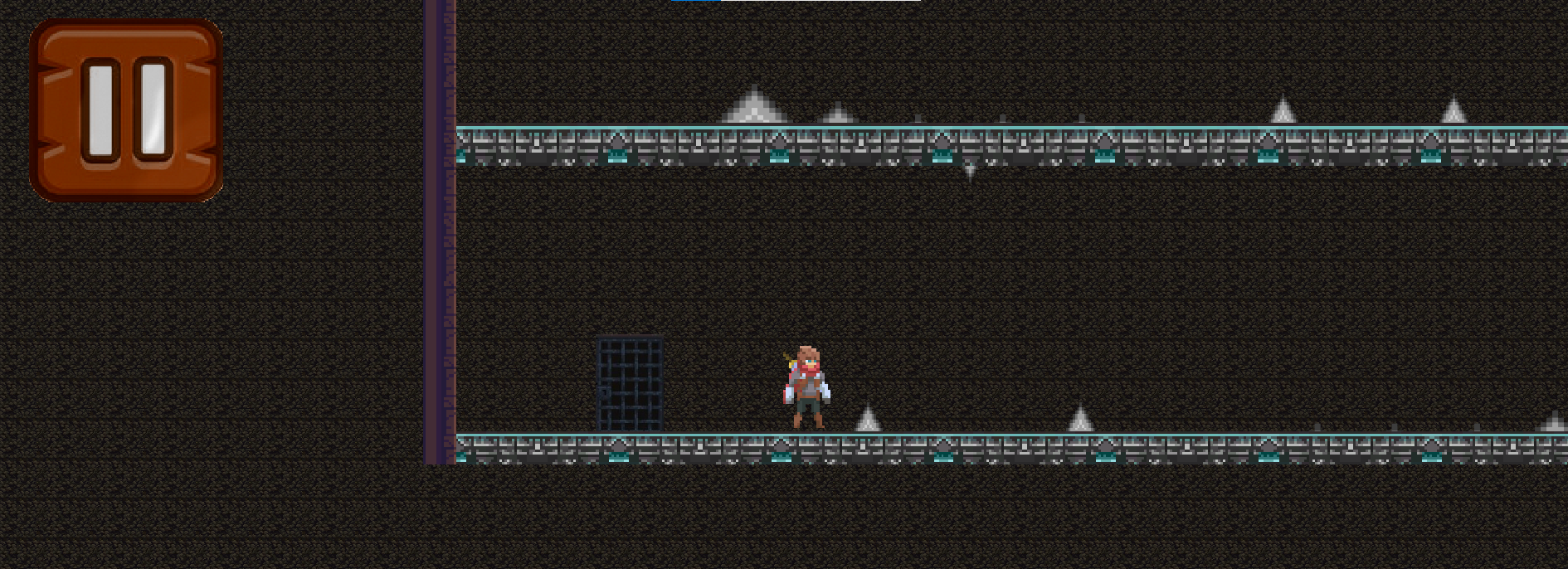
 Рисунок 21 Уровень 4

 *Рисунок 22 Уровень 5*

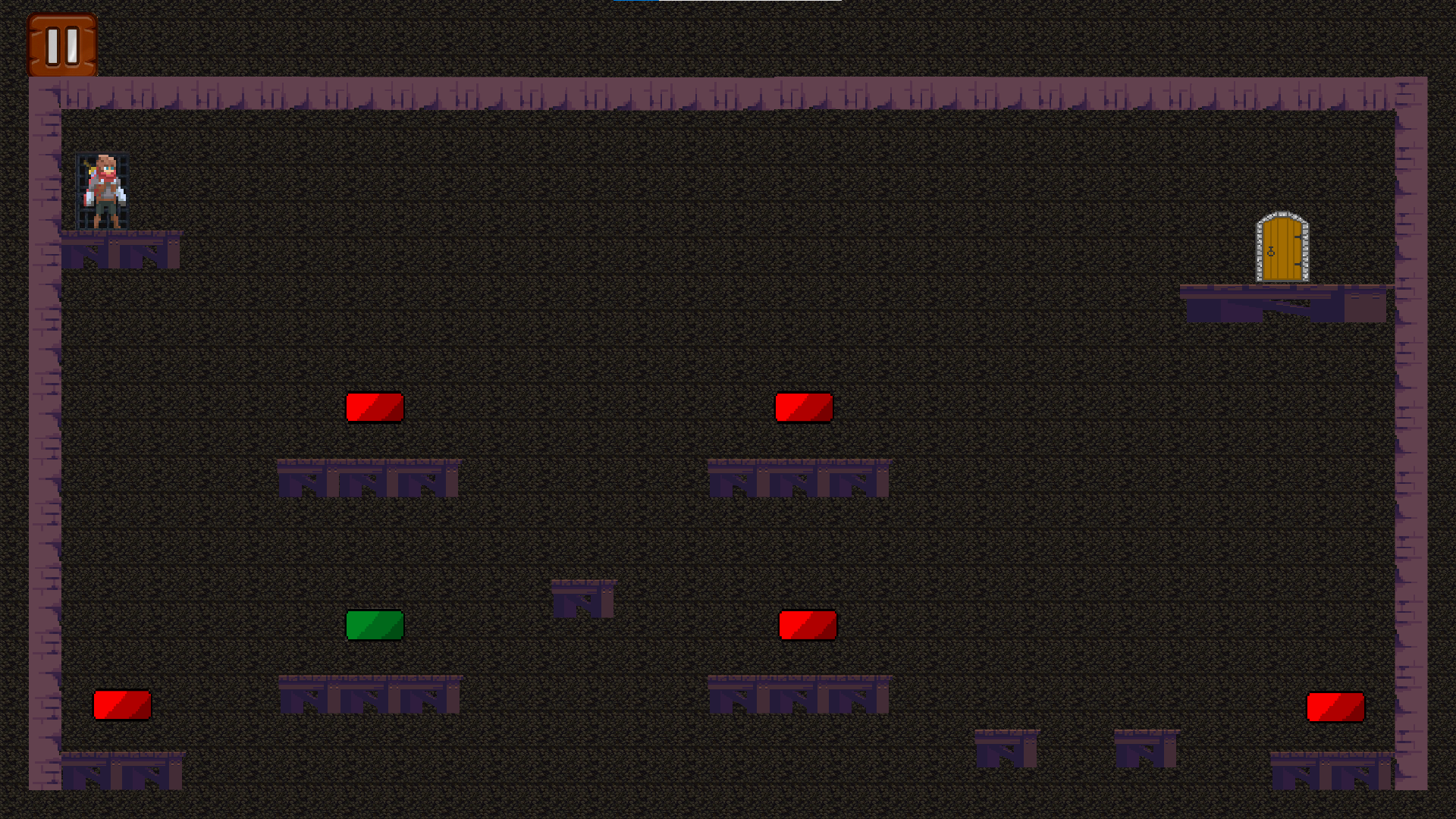
Изображение выглядит как снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

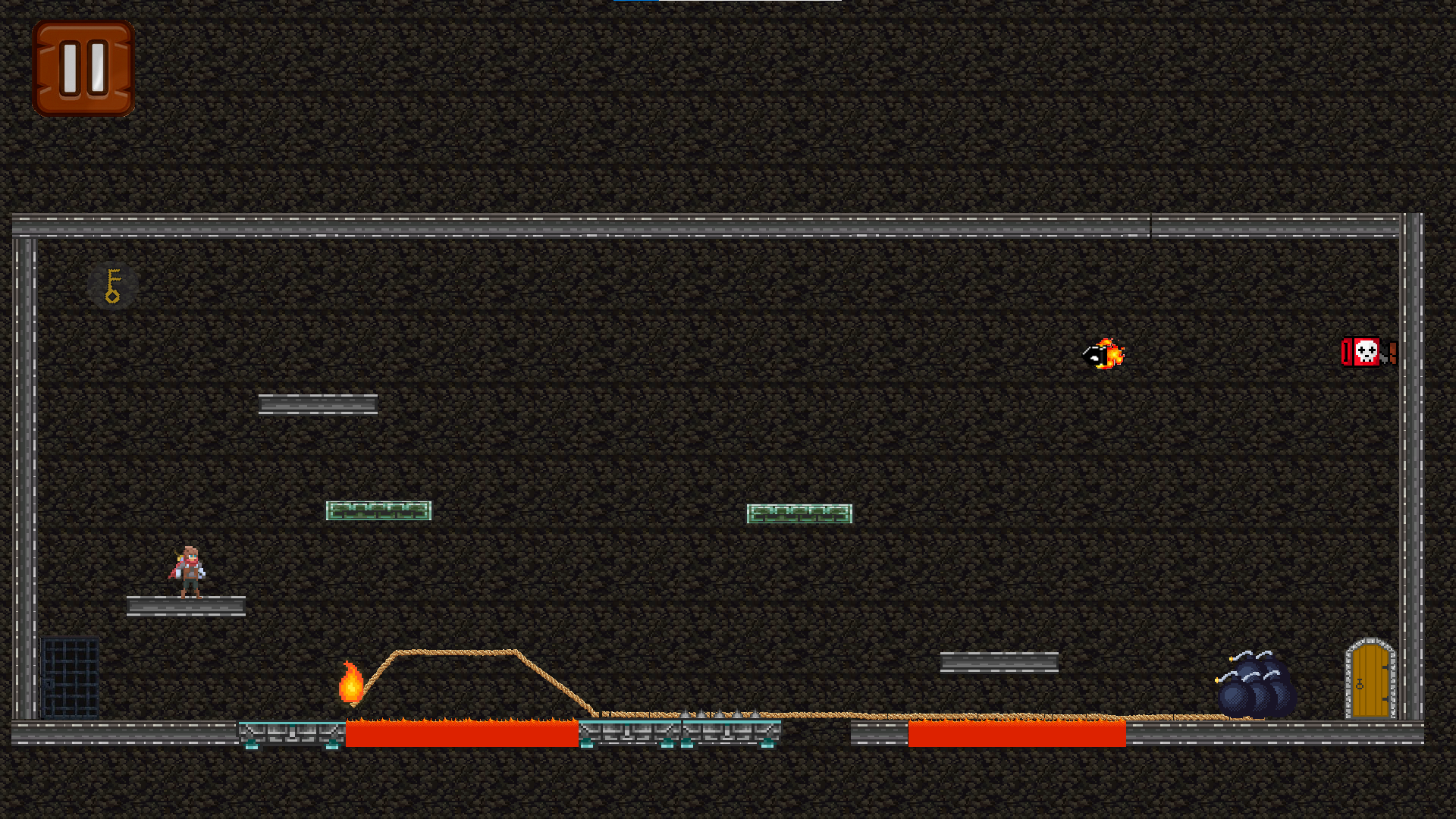
*Рисунок 23 Игровой модуль*



*Рисунок 24 Уровень 6*



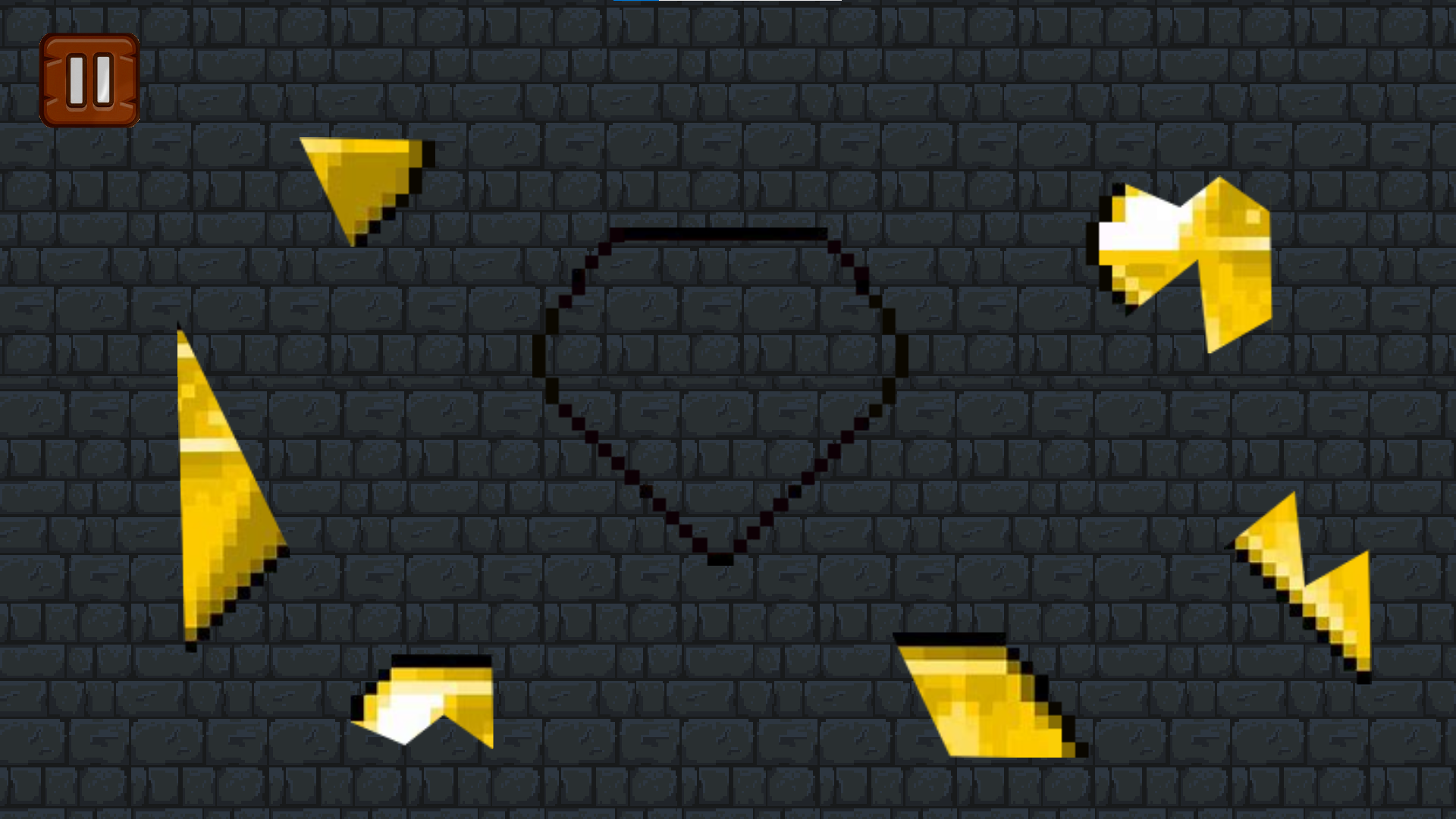
*Рисунок 25 Уровень 7*



*Рисунок 26 Уровень 8*



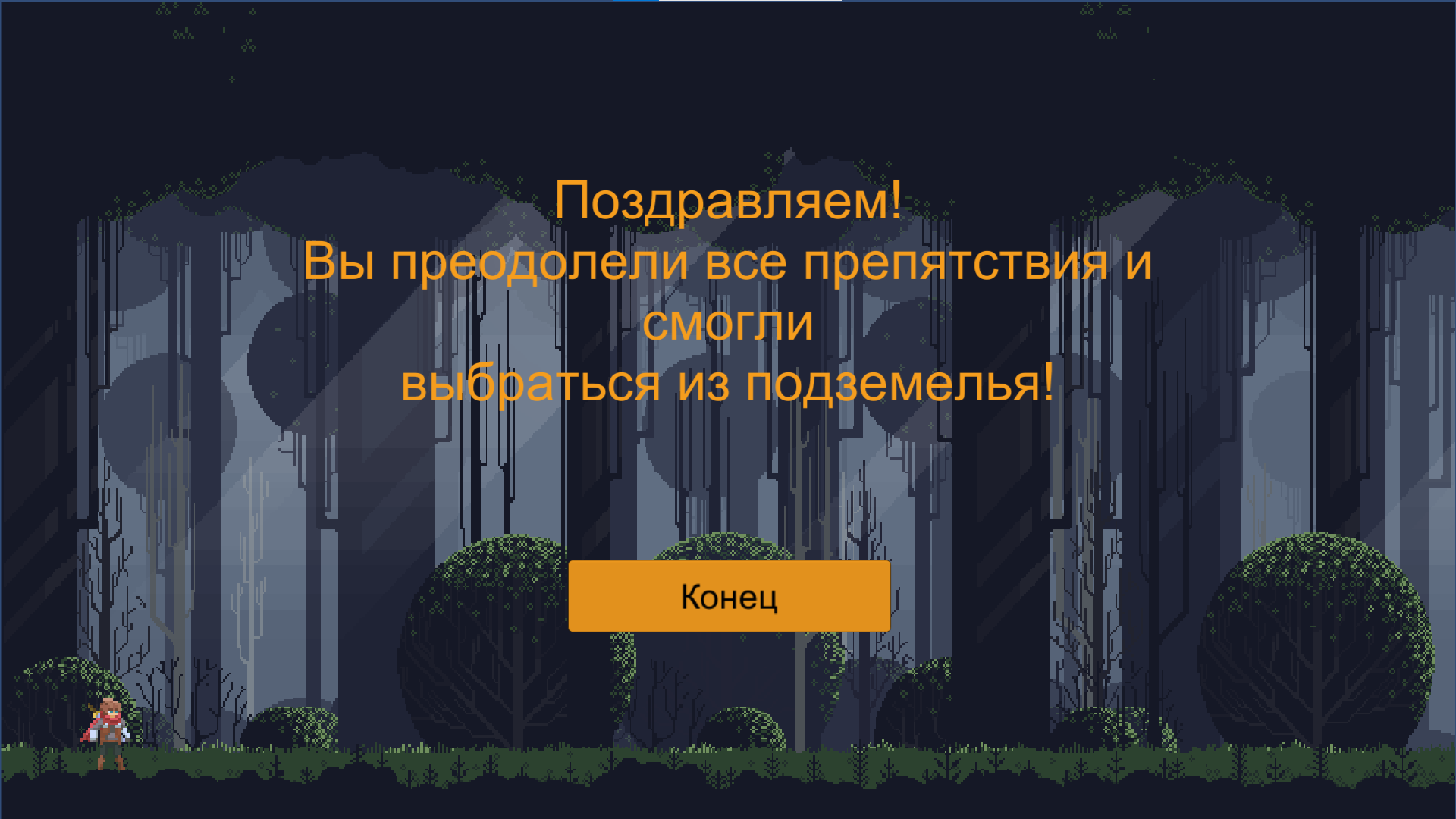
*Рисунок 27 Уровень 9*



*Рисунок 28 Пазл*



*Рисунок 29 Уровень 10*



*Рисунок 30 Финальная сцена*