МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ «ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНО-ЭКОНОМИЧЕСКИЙ КОЛЛЕДЖ»

Специальность 2-40 01 01 «Программное обеспечение

информационных технологий»

**КУРСОВОЙ ПРОЕКТ**

по дисциплине: «Основы алгоритмизации и программирования»

**на тему: «Разработка игрового приложения»**

Пояснительная записка

|  |  |  |  |
| --- | --- | --- | --- |
| Разработал |  |  | Доморацкий Р.К. |
|  | (подпись) |  | (ФИО) |
| Руководитель |  |  | Серикова С.А. |
|  | (подпись) |  | (ФИО) |

Гомель, 2023

**СОДЕРЖАНИЕ**

[Введение 3](#_Toc106276733)

[1 Двумерный массив 4](#_Toc106276734)

[2 Описание алгоритмов технологий решения 7](#_Toc106276735)

[2.1 Постановка задачи 7](#_Toc106276736)

[2.2 Описание алгоритмов и технологий решений 8](#_Toc106276737)

[2.3 Описание компонентов 9](#_Toc106276738)

[3 Описание интерфейса 13](#_Toc106276739)

[3.1 Описание интерфейса. 13](#_Toc106276740)

[3.2 Описание алгоритмов каждой формы. 16](#_Toc106276741)

[3.3 Контроль корректности данных 20](#_Toc106276742)

[4 Результаты работы программе 21](#_Toc106276743)

[Заключение 29](#_Toc106276744)

[Список использованных источников 30](#_Toc106276745)

[Приложение А 31](#_Toc106276746)

Изм.

Лист

№ докум.

Подпись

Дата

Лист

2

*01.2023.2-400101.1.ПО-31.XXXX ПЗ*

Разраб.

*Доморацкий*

Провер.

Серикова

Н. Контр.

Утв.

*Разработка*

*информационно-поисковой системы*

*«Коллекция музея»*

Лит.

Листов

2

*ГГАЭК*

*у*

у

# **ВВЕДЕНИЕ**

C# — объектно-ориентированный язык программирования общего назначения. Разработан в 1998—2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для платформы Microsoft .NET Framework и .NET Core. Впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270.

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, переменные, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

Язык программирования C# считается одним из самых универсальных. Он применяется в самых разных сферах. Например, для создания продвинутых бизнес-приложений, видеоигр, функциональных веб-приложений, приложений для Windows, macOS, мобильных программ для iOS и Android.

C Sharp изначально был придуман компанией Microsoft для собственных целей и служб. Он предусматривает следующие преимущества:

* строгую типизацию;
* сохранение концепций объектно-ориентированного программирования;
* функциональность;
* достаточно мощный инструментарий;
* стабильную работу через Visual Studio;
* компактный и легко читаемый код;
* понятный даже новичкам синтаксис.

При использовании этого языка можно насладиться обработкой исключений, а также наличием сборщика мусора. Здесь все продумано так, чтобы программеру было легко писать и считывать итоговые кодификации.

Unity — кроссплатформенная среда разработки компьютерных игр, разработанная американской компанией Unity Technologies. Unity позволяет создавать приложения, работающие на более чем 25 различных платформах, включающих персональные компьютеры, игровые консоли, мобильные устройства, интернет-приложения и другие. Выпуск Unity состоялся в 2005 году и с того времени идёт постоянное развитие.

Основными преимуществами Unity являются наличие визуальной среды разработки, межплатформенной поддержки и модульной системы компонентов. К недостаткам относят появление сложностей при работе с многокомпонентными схемами и затруднения при подключении внешних библиотек.

# **1 ДВУМЕРНЫЙ МАССИВ**

Структурированные типы данных массив — это пронумерованная последовательность величин одинакового типа, обозначаемая одним именем. Элементы массива располагаются в последовательных ячейках памяти, обозначаются именем массива и индексом.

Массив – это структурированный тип данных, состоящий из фиксированного числа элементов одного типа. Тип элементов массива называется базовым. Число элементов массива фиксируется при описании и в процессе выполнения программы не меняется.

Доступ к элементу массива реализуется указателем имени массива и в квадратных индекса. Индексы элементов массива — это выражения любого скалярного типа кроме вещественного. Двумерный массив представляет собой массив, в котором положение элементов определяется двумя индексами.

Массив в С# представляет собой набор однотипных данных. Объявление массива похоже на объявление переменной за тем исключением, что после указания типа ставятся квадратные скобки. Пример описания массива представлен на рисунке 1.1.

Массивы характеризуются таким понятием как ранг или количество измерений. Выше мы рассматривали массивы, которые имеют одно измерение – такие массивы можно представлять в виде ряда элемента. Но массивы также бывают многомерными. У таких массивов количество измерений больше 1.

Основные определения массива:

* ранг – количество измерений массива;
* длина измерения – длина отдельного измерения массива;
* длина массива – количество всех элементов массива.

Массивы, которые имеют два измерения называют двухмерными. Например, создадим одномерный и двухмерный массивы, которые имеют одинаковые элементы:

Пример описания массива представлен на рисунке 1.1.

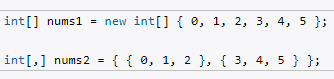


Рисунок 1.1 – Многомерный массив

Массивы могут иметь и большее количество измерений. Объявление трехмерного массива могло бы выглядеть так:



Рисунок 1.2 – Трехмерный массив

Соответственно могут быть и четырехмерные массивы, и массивы с большим количеством измерений. Но на практике обычно используются одномерные и двухмерные массивы.

Определенную сложность может представлять перебор многомерного массива. Прежде всего надо учитывать, что длина такого массива — это совокупное количество элементов Пример описания перебора массива представлен на рисунке 1.3.

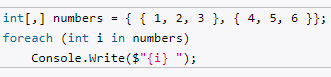


Рисунок 1.3 – Перебор массива через «forEach»

Но что если мы хотим отдельно пробежаться по каждой строке в таблице? В этом случае надо получить количество элементов в размерности. В частности, у каждого массива есть метод «GetUpperBound», который возвращает индекс последнего элемента в определенной размерности. И если мы говорим непосредственно о двухмерном массиве, то первая размерность, по сути, это и есть таблица. И с помощью выражения «numbers.GetUpperBound(0) + 1», можно получить количество строк таблицы, представленной двухмерным массивом. А через «numbers.Length / количество\_строк» можно получить количество элементов в каждой строке:

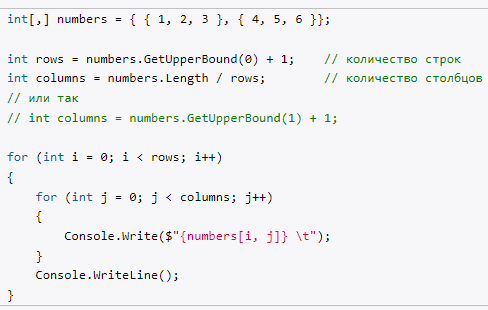


Рисунок 1.4 – Перебор массива через «for»

От многомерных массивов надо отличать массив массивов или так называемый "зубчатый массив":

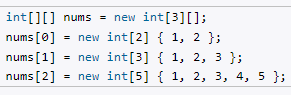


Рисунок 1.5 – Массив массивов

Здесь две группы квадратных скобок указывают, что это массив массивов, то есть такой массив, который в свою очередь содержит в себе другие массивы. Причем длина массива указывается только в первых квадратных скобках, все последующие квадратные скобки должны быть пусты: «new int[3][]». В данном случае у нас в массиве содержит три массива. Причем размерность каждого из этих массивов может не совпадать.

Используя вложенные циклы, можно перебирать зубчатые массивы. Например:

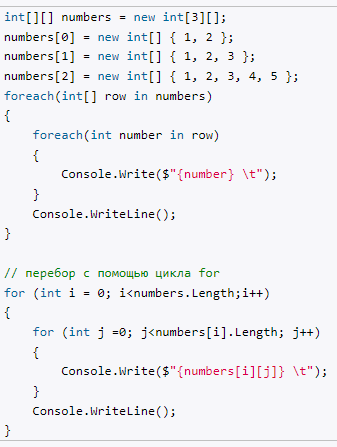


Рисунок 1.6 – Перебор массива массивов

# **2 ОПИСАНИЕ АЛГОРИТМОВ ТЕХНОЛОГИЙ РЕШЕНИЯ**

## **2.1 Постановка задачи**

В курсовом проекте необходимо разработать игровое приложение на языке программирования «C#» и с использованием игрового движка «Unity».

При начале запуска игры должна присутствовать главное меню, на котором выбрать перечень функции: «Старт», «Настройка», «Выйти».

При нажатии на кнопку «Старт», создается реестр памяти, на котором будет храниться информацию о прохождение, количество очков и скорость передвижения персонажа.

Для того чтобы реализовать такую логику игры нужно:

* управления персонажем;
* анимированные персонажи и объекты;
* взаимодействия с объектами для передвижения;
* взаимодействия с ловушками на персонажем;
* сохранение статистику игрока и текущее сохранение.

Необходимо произвести настройку разрешение, окно паузы, регулирование музыки и звука, а также организовать адаптивный интерфейс для любых устройств.

Исходными данными для игрока в данной игровой приложении являются:

* скорость передвижения;
* фиксированная и предельная скорость (для ускорения);
* кол-во улучшения скорости;
* кол-во очков (фруктов);
* параметры прыжка.

Также входят данные, через которые будут реализованы индивидуальные функции программы:

* ускорения персонажа, при нажатии на кнопку игрок со временем будет ускоряться до его предельной скорости;
* двойной прыжок персонажа, при нажатии пробела идёт отсчет время последнего нажатии, после чего выполняет дополнительный прыжок.



Рисунок 2.1 – Описание игрока

С помощью атрибутов записи реализуется работа персонажа с этими данными:

* атрибут под названием «speed», отвечает за скорость передвижения игрока, также этот атрибут будет меняться, если атрибут «doubleSpeed» будет истинным, то при передвижения набирает ускорения;
* атрибут под названием «fixedSpeed», предназначен для первоначальной скорости передвижения игрока, и используется для сброса атрибута «speed»;
* атрибут под названием «limitSpeed», предназначен для ограничения скорости передвижения игрока взаимосвязанный с атрибутом «doubleSpeed»;
* атрибут под названием «countUpdate», отвечает за ограничения покупки улучшения в магазине;
* атрибут под названием «fruits», отвечает за количество собранных фруктов и предназначены для обмена на улучшения способности персонажа;
* атрибут под названием «jumpForce», отвечает, насколько прыгнет персонаж;
* атрибут под названием «doubleJump» представляет собой логический тип, который отвечает за дополнительный прыжок игрока, а также рассчитывает время нажатия клавиши, чтобы персонаж не делал несколько прыжков;
* атрибут под названием «doubleSpeed» представляет собой логический тип, который отвечает за ускорение игрока, при этом ускорения рассчитывается от скорости игрока до предела за секунду, и это показатель плавно растёт, если скорость равна или больше предела, то скорость не возрастает, но для этого пользователю нужно зажать клавишу, при отпускании скорость становиться первоначальной.

## **2.2 Описание алгоритмов и технологий решений**

Алгоритм — это система точных и понятных предписаний о содержании и последовательности выполнения конечного числа действий, необходимых для решения любой задачи данного типа.

Функция — фрагмент программного кода, к которому можно обратиться из другого места программы. С именем функции неразрывно связан адрес первой инструкции, входящей в функцию, которой передаётся управление при обращении к функции. После выполнения функции управление возвращается

обратно в адрес возврата — точку программы, где данная функция была вызвана.

Для программы были перечислены и описаны такие функции как:

* функция «передвижения», при котором пользователь нужно нажимать на кнопки (wasd), после их нажатия персонаж будет передвигаться и будет сменять каждый спрайт
* функция «сохранение», при котором пользователь может выйти из игры, и после очередного запуска, загружается та сцена которая была последний момент выхода пользователя;
* функция «прохождения уровня», позволяет при завершения данного уровня перейти на другой со сдержкой;
* функция «передвижения объекта по точкам», отвечает за передвижения объекта, где включают массив точек, по который будет передвигаться объект зависимости от поставленной скорости;
* функция «собирание фруктов» предназначен для сбора фруктов, когда входит персонаж в коллайдер фрукта, то срабатывает функция и проигрывается анимация сбора и после этого уничтожается объект и начисляется очки;
* функция «нажатия плитки» задача игрока перетащить куб на нужную плитку, чтобы можно продолжить проходить следующее препятствие.

## **2.3 Описание компонентов**

В данной игре будем использовать такие компоненты как:

* компонент «Animation»;
* компонент «Controller»;
* компонент «GameObject»;
* компонент «Script»;
* компонент «Tilemap»;
* компонент «Canvas»;
* компонент «Scenes»;
* компонент «Rigibody2D»;
* компонент «BoxCollider2D»;
* компонент «CircleCollider2D»;
* компонент «AudioSource»;
* компонент «Tilemap Collider»;
* компонент «Platform Effector2D»;
* компонент «Composite Collider2D»;
* компонент «Sprite Render».

Для работы с объектами взаимодействия используются следующие функции и операции над ними:

* функция «Invoke», вызывает метод в секундах времени;
* функция «StartCoroutine», запускает сопрограмму. Выполнение сопрограммы можно приостановить в любой момент с помощью оператора yield. Когда используется инструкция yield, сопрограмма приостанавливает выполнение и автоматически возобновляет выполнение в следующем кадре;
* функция «Update» обновление вызывается каждый кадр, если включен «MonoBehaviour».
* Функция «OnCollisionEnter2D» отправляется, когда входящий коллайдер вступает в контакт с коллайдером этого объекта (только для 2D-физики). Дополнительная информация о столкновении сообщается в параметре «Collision2D», передаваемом во время вызова.
* Функция «OnCollisionExit2D» отправляет, когда коллайдер другого объекта перестает касаться коллайдера этого объекта (только для 2D-физики. дополнительная информация об задействованных объектах сообщается в параметре Collision2D, передаваемом во время вызова.
* Функция «CompareTag» возвращает истину, если «GameObject» имеет такой же тег. В противном случае возвращает ложь.
* Функция «GetComponent» получает ссылку на компонент типа “T” в том же «GameObject», что и указанный компонент.
* Функция «Destoy» удаляет GameObject, компонент или актив.
* Функция «OnTriggerEnter2D» отправляет, когда другой объект входит в триггерный коллайдер, прикрепленный к этому объекту (только для 2D-физики). Дополнительная информация о другом коллайдере сообщается в параметре «Collider2D», передаваемом во время вызова.
* Функция «OnTriggerExit2D» отправляет, когда другой объект покидает триггерный коллайдер, прикрепленный к этому объекту (только для 2D-физики). Дополнительная информация о другом коллайдере сообщается в параметре Collider2D, передаваемом во время вызова.
* Функция «SetActive» активирует или деактивирует «GameObject» в зависимости от заданного значения истина или ложно.

Макет вступительной формы интерфейса информационно-поисковой системы «Коллекция музея» показан на рисунке 2.2.

Макет паузы аналогично выглядит, как в главном меню только без названия.

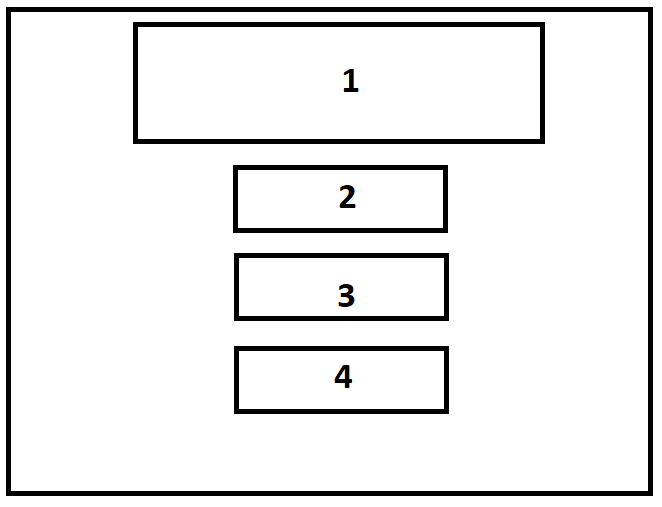


Рисунок 2.2 – Макет главного меню:

1 – блок название игры; 2 – блок начало игры; 3 – кнопка настройки;

4 – кнопка выхода;

Макет настройки представлен ан рисунке 2.3. В этой форме присутствуют настройки музыки и звуки, разрешения игры, качество и полноэкранный режим.

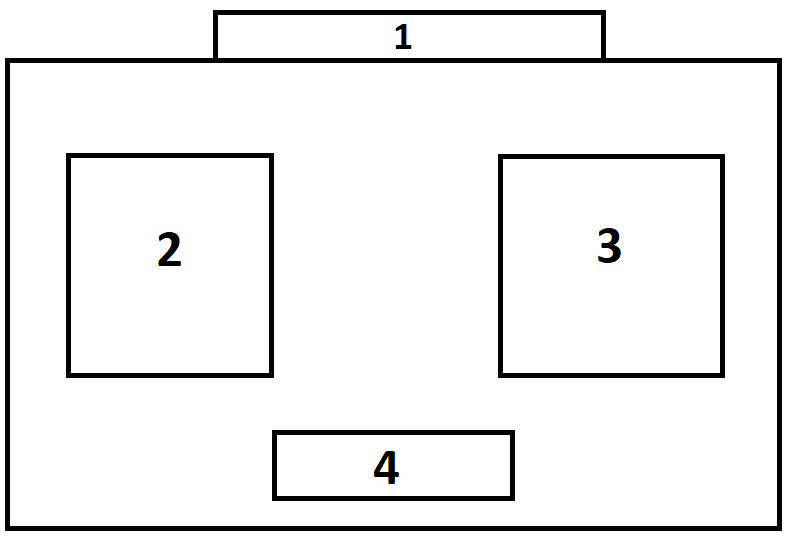


Рисунок 2.3 – Макет главной формы:

1 – блок название окна; 2 – блок настройки музыки; 3 – блок настройки разрешения и качества; 4 – кнопка выхода из текущего окна;

# **3 ОПИСАНИЕ ИНТЕРФЕЙСА**

## **3.1 Описание интерфейса.**

На рисунке 3.1 показана главное меню игры, с помощью которой пользователь может либо настроить, либо перейти сразу в игру.

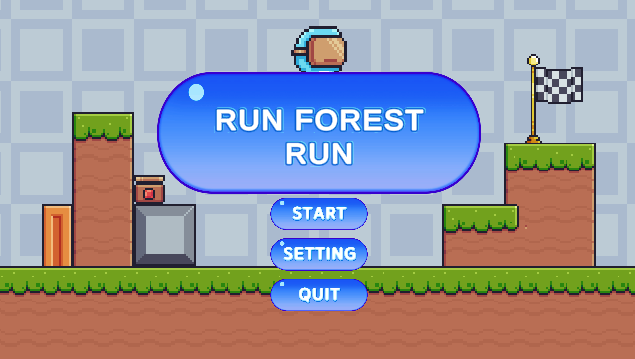


Рисунок 3.1 – Главное меню

После плавного перехода, пользователю представляют главное меню игры, на котором располагается задний вид игры. Также в главном экране игры представлен несколько кнопок на которых можно открыть окно настройки, либо начать игру сразу при этом игра поддерживает адаптивный интерфейс что позволяет менять на полноэкранную игры, при это присутствует задний фон музыки и звуки нажатии кнопки. Закрытие программы осуществляется через кнопку главного меню на форме.

При выборе кнопки «Настройка», пользователю показывают окно настройки, на котором расположены в левом части настройка музыки и звука в игре, а справа разрешения экрана, которые представлен виде список выбора

разрешений, качество аналогично с разрешений экрана и полноэкранный режим. При нажатии кнопки выхода, текущее окно исчезает и появляется главное меню.

Графическое окно настройки представлен на рисунке 3.2.

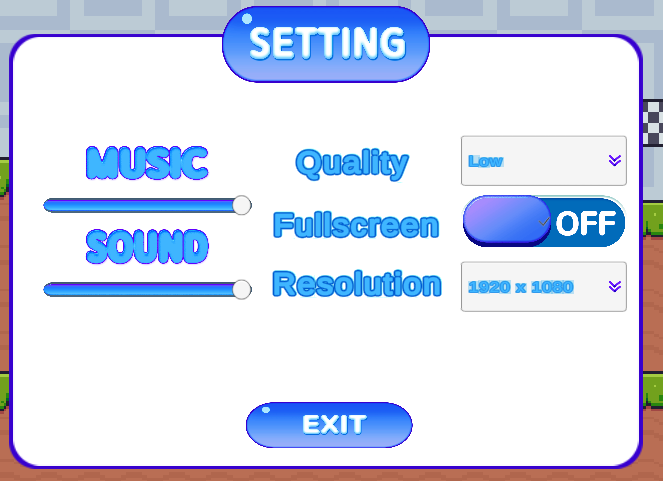


Рисунок 3.2 – Окно настройки

После нажатия на кнопки «Старт» начинается ожидания перехода в игру. Сразу после ожидания загружается игра и появляется персонаж при которым, можно за него управлять и проходить обычные и логические препятствии. При этом у персонажа имеет стандартные параметры и возможность возраста этих значений при прохождения игры.

Экран начало первой уровня игры представлен на рисунке 3.3.



Рисунок 3.3 – Первый уровень игры

На рисунке 3.4 показано препятствие виде шипов при это над ним есть фрукт, который предназначен для показать очков и после прохождения можно обменяться с торговцем на усиления, что позволяет проходить препятствие и игру намного быстрее. При взаимодействия шипом, у персонажа проигрывается анимация проигрыша и пользовать ожидает воскрешения персонажа, но при этом его очки возвращают текущее значение.



Рисунок 3.4 – Препятствие с вишней

На рисунке 3.5 представлен передвигающая платформа на котором игрок может на него постоять, и он будет передвигать из-за одной точки в другой. Также учитывается, что, если персонаж находиться, на этой платформе, то сам персонаж будет вместе с ним передвигаться. Аналогично работает с препятствиями.

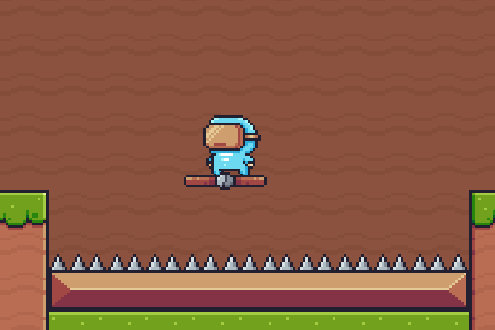


Рисунок 3.5 – Передвигающая платформа

На рисунке 3.6 изображено персонаж, которому нужно передвинуть данный куб к плитке, чтобы заработал механизм открытие ворот, но сам игрок не сможет задействовать на механизм.



Рисунок 3.6 – Механизм открытие ворота

Также присутствует меню пауза, что позволяет игре остановить свой игровой процесс. При вызове паузы появляется стандартное меню, при этом есть выбор «Возобновить», «Настройка», либо «Выйти» из игры. При выходе из игры сохраняется сцена на котором остановился игрок и после того, как игрок хочет возобновить сцену, загружается из реестра памяти сцена.

Экран меню паузы игры представлен на рисунке 3.7.

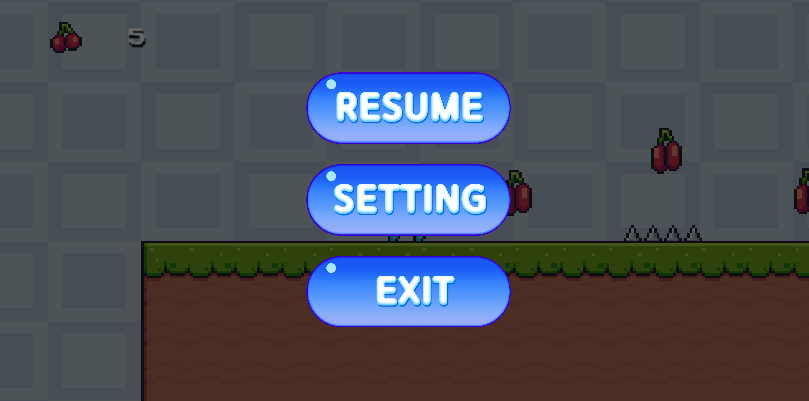


Рисунок 3.7 – Меню паузы

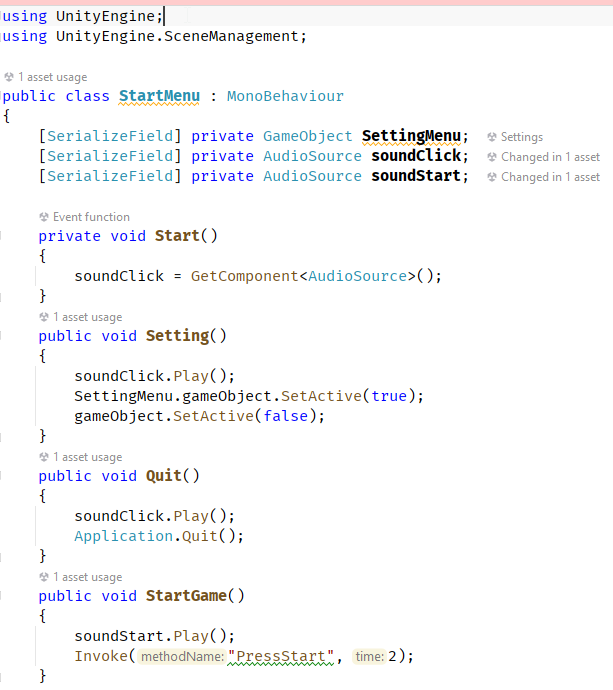
На рисунке 3.8 изображен интерфейс магазина, который предназначен для покупки улучшения персонажа или способности. Для того, чтобы отобразить окно магазина нужно подойти к продавцу и нажать на кнопку «E».



Рисунок 3.8 – Меню магазина

## **3.2 Описание алгоритмов каждой формы.**

При запуске программы появляется главное меню с присутствуют кнопки выбора, каждая кнопка отвечает за отдельную функцию, также присутствуют поля взаимодействия с объектом «Настройки» и со звуками. Код главного меню представлен на рисунке 3.9.



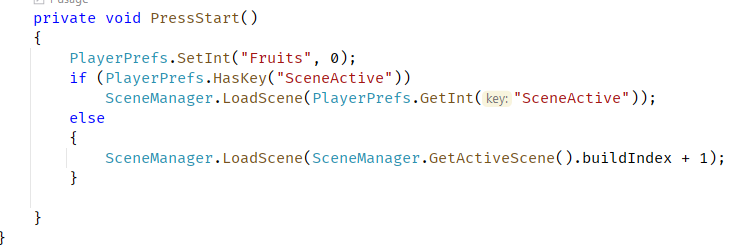


Рисунок 3.9 – Код главного меню

Когда игрок нажал на кнопку «Настройки» появляется окно с настройками. При этом у настройки включают другие функции взаимодействия с компонентами. Код окно настройки игры представлен рисунке 3.10.

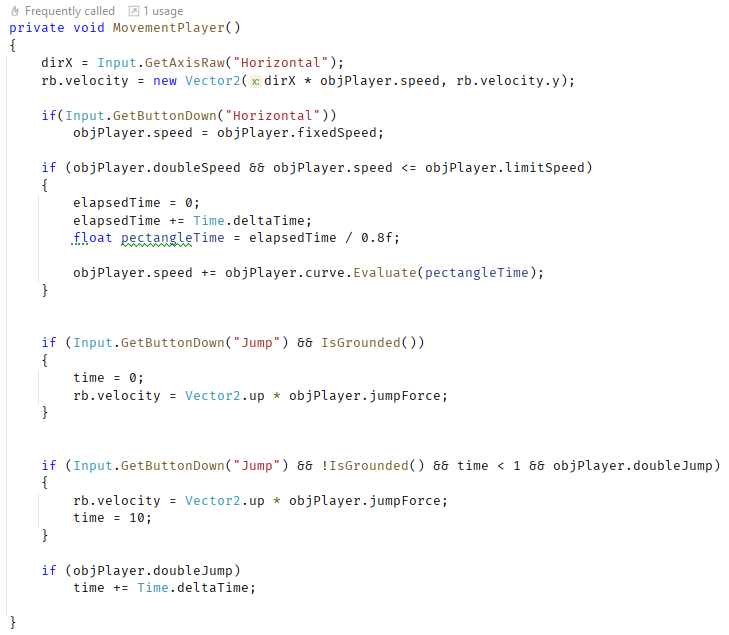




Рисунок 3.10 – Код окна настройки

После запуска игры, игрок может передвигать персонажа, который использует один скрипт для передвижения, прыжка, использование способностей, проигрывание анимации, ограничение прыжка, и надстройка слоя для платформы. Код окно настройки игры представлен рисунке 3.11.





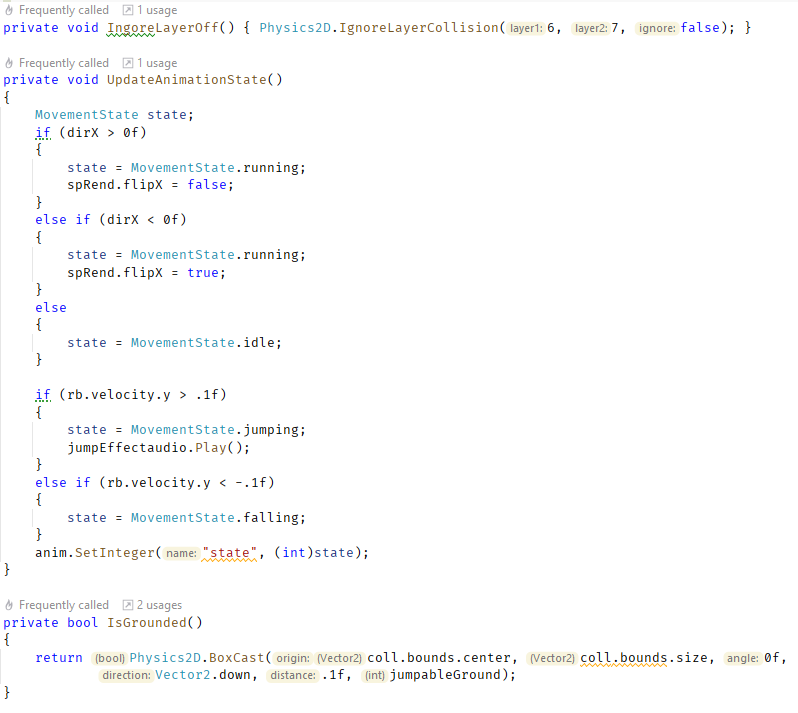


Рисунок 3.11 – Код передвижения персонажа

Персонаж не только передвигается по объектам, но и собирает фрукты, которые будут нужны для купли способности или прокачки персонажа. Код сбора фруктов представлен рисунке 3.12.



Рисунок 3.12 – Код сбора фруктов

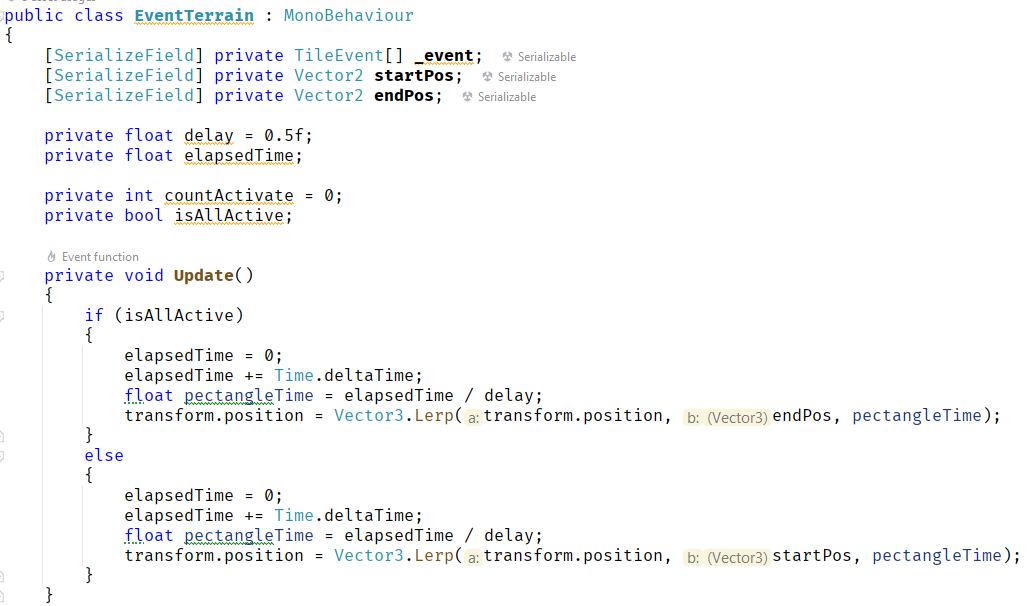
Персонаж может взаимодействовать с ловушками который при взаимодействии проигрывается анимация проигрыша и перезагружается сцена, но собранных фруктов сбросятся до первоначального. Код проигрыша персонажа представлен рисунке 3.13.





Рисунок 3.13 – Код проигрыша игрока

В некоторые прохождения уровней могут присутствовать логическая игра, которая предназначена для открытие следующей препятствие, вырабатывание логики. Код активации препятствие представлен рисунке 3.14.



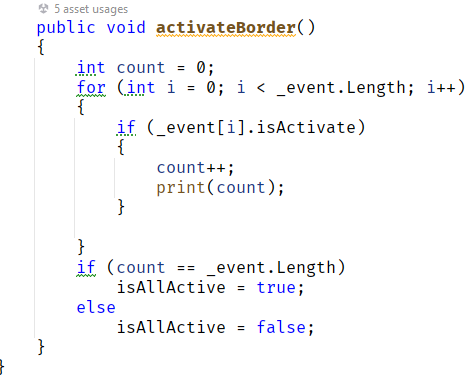


Рисунок 3.14 – Код активации препятствие

Для того чтобы отобразился меню паузы нужно нажать на кнопку «Escape», после чего, затемняется фон и появляется стандартные кнопки «Возобновить», «Настройки», «Выход». Код появление меню паузы представлен рисунке 3.15.



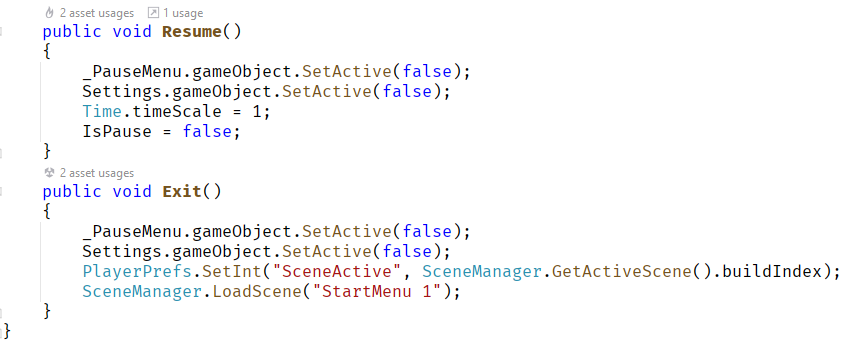


Рисунок 3.15 – Код паузы меню

## **3.3 Контроль корректности данных**

При взаимодействии с магазином происходить корректность находиться ли персонаж в диапазоне, также, когда появляется окно магазина, но игрок не может осуществить паузу. Код события корректности диапазона взаимодействия на рисунке 3.17.

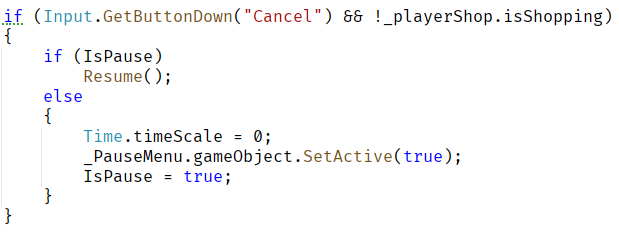




Рисунок 3.17 – Код корректности взаимодействия

При покупки способности либо улучшения сначало проверяется сколько нужно фруктов для покупки, если есть нужное количество, то тратиться фрукты перезаписывается сохранение фруктов и улучшается параметр, либо активации способности. Код события корректности диапазона взаимодействия на рисунке 3.18.



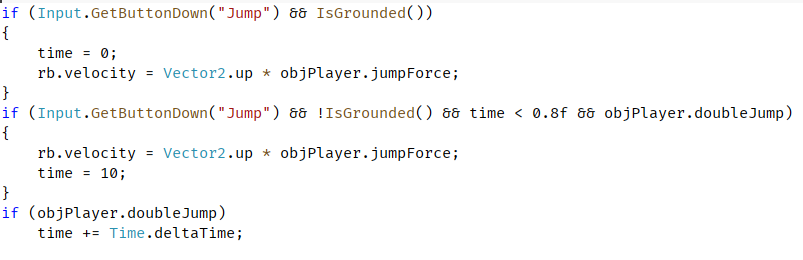


Рисунок 3.18 – Код покупки прокачки

# **4 РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЕ**

При запуске игрового приложения отображается вступительная форма игрового движка и после чего появляется плавное отображения главного меню игры. Главное меню программы показана на рисунке 4.1.

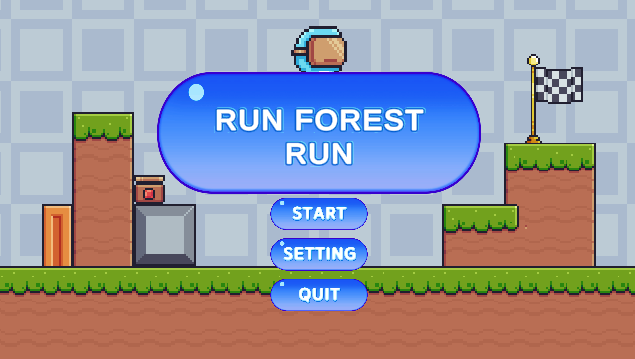


Рисунок 4.1 – Главное меню

После плавной смены, представлен пользовательский интерфейс, которому пользователю предлагают стандартные кнопки взаимодействия «Старт», «Настройки» и «Выход». Также поддерживается адаптивность игры, где можно изменять разрешения игры, полноэкранный режим и настройка качества, ещё присутствуют настройки музыки и звуки в игре. Окно настройки представлен на рисунке 4.2.

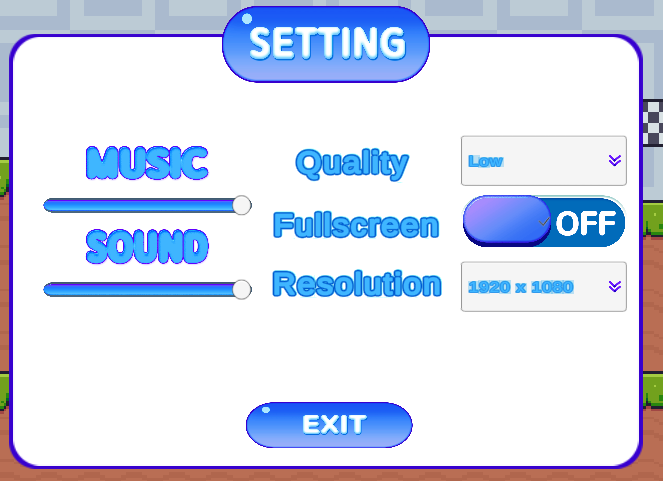


Рисунок 4.2 – Окно настройки

Для начала игры нужно нажать на кнопку «Старт», ожидается загрузка с промежутком времени после чего плавное появляется сцена игры вместе с персонажем и препятствием. Также в игре присутствуют сохранение данных в реестре, предназначен для возобновление игры, очков и сцена. Также отображается количество очков, которые пригодиться для улучшения персонажа. Сцена игры показана на рисунке 4.3.

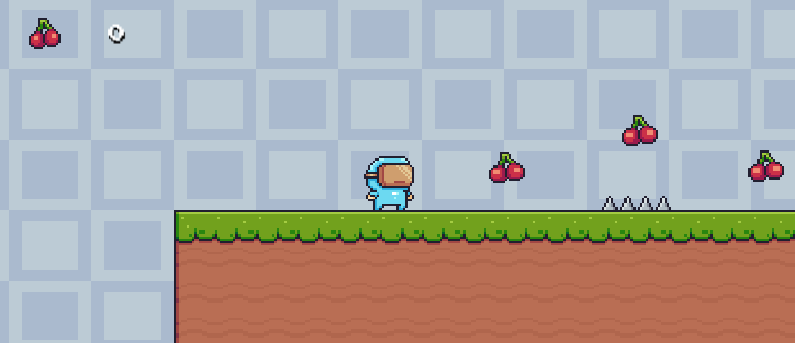


Рисунок 4.3 – Сцена игры

Для того, чтобы поставить игру на паузу, нужно нажать на кнопку «Escape», после этого затемнеет задний фон и отобразиться кнопки взаимодействия как: «Возобновить», «Настройки», «Выйти». При выходе из-за игры сохраняются статистика игрока. Меню паузы изображен на рисунке 4.4.

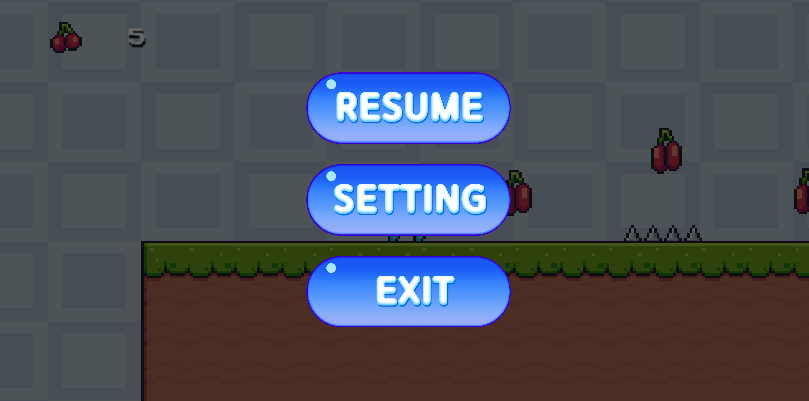


Рисунок 4.4 – Меню паузы

При прохождении препятствий игрок, может наблюдать передвигающийся платформу, которая передвигается от одной точки до другой. Передвижения платформы изображен на рисунке 4.5.

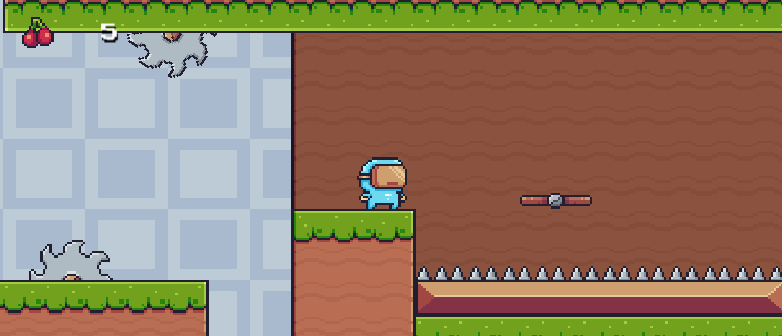


Рисунок 4.5 – Передвижения платформы

Также в некоторых объектов могут скрываться некоторые очки или спрятанные объекты для решения головоломок. Секретные фон изображен на рисунке 4.6.



Рисунок 4.6 – Передвижения платформы

В игре присутствуют препятствие, в котором, нужно передвигать объекты в нажимные плитки, зависит от сложности выполнения, чтобы дальше проходить уровень. Прохождение препятствие представлен рисунок 4.7.



Рисунок 4.7 – Прохождения препятствие

После прохождения препятствие игрок, видит перед собой флаг, который завершает этот уровень. При взаимодействии с ним проигрывается звук победы и ожидается переход на новый уровень, и сохраняется следующий уровень и количество очков. Флаг прохождения уровня изображен на рисунке 4.8.



Рисунок 4.8 – Флаг прохождения уровня

После того, как появился новый уровень нам предстоит прыгнуть на батут, который подбрасывает персонажа на вверх, однако батут имеют разную cилу. Батут изображен на рисунке 4.9.

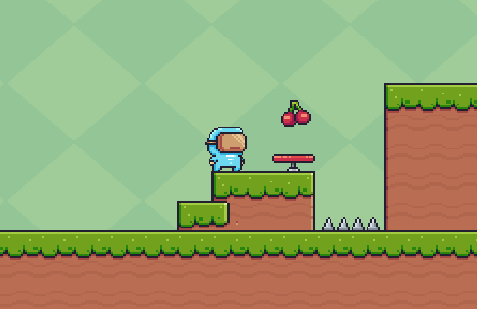


Рисунок 4.9 – Батут

Также могут присутствовать выбор прохождение игрок может пройти по легкому пути, но при этом фрукты уничтожаться, если выбрал сложный путь, то получит больше фруктов. Выбор препятствий изображен на рисунке 4.10.



Рисунок 4.10 – Выбор препятствий

Иногда в некоторых уровнях может встречать торговца, который предлагает игроку закупиться способностями осуществляется через кнопку «E». При нажатии отображает окно с цен на конкретную способность. Если игроку, есть очки для покупки, который списывает очки игрока и после чего улучшается параметры персонажа, иначе недоступна покупка. Окно магазина изображен на рисунке 4.11.



Рисунок 4.11 – Окно магазина

После прохождения всех уровней появляется окно с поздравлением о прохождение игры, но есть выбор из кнопок, игрок может сбросить все параметры на кнопку «Рестарт», если продолжить, то «Выйти». После выбора игроку возвращает в главное меню игры. Окно поздравления изображен на рисунке 4.12.



Рисунок 4.12 – Окно поздравления

# **ЗАКЛЮЧЕНИЕ**

В результате курсовой работы была разработана игровое приложение «» и изучен теоретический материал по теме «Структурированный тип данных массив. Двумерные массивы».

В процессе выполнения курсовой работы были закреплены знания о работе с различными стандартными функциями, работа с анимации персонажей, взаимодействия с объектами, создание пользовательский интерфейс, использование атрибутов для отображения в инспекторе, создание триггеров и настройка в игре с использованием средой разработки для игр «Unity», навыки создание функции и взаимодействиям с классами.

Разработанный приложение поддерживает такие функции как:

* управления персонажем;
* анимированные персонажи и объекты;
* взаимодействия с объектами для передвижения;
* взаимодействия с ловушками на персонажем;
* сохранение статистику игрока и текущее сохранение.
* ускорения персонажа, при нажатии на кнопку игрок со временем будет ускоряться до его предельной скорости;
* двойной прыжок персонажа, при нажатии пробела идёт отсчет время последнего нажатии, после чего выполняет дополнительный прыжок.

Основной функционал программы реализуется по средствам наборов модулей, которые обеспечивают разные функции, такие как, передвижения персонажа, использование способностей, улучшения параметров персонажа, вызов пользовательских интерфейсов, прохождения препятствий и сохранение данных в реестре.

Наличие данного функционала обеспечивает облегчение игровой механике, взаимодействия с объектами, сохранение данных в реестре памяти, адаптивный интерфейс по разное расширение, полноэкранный режим и управление музыкой и проигрывание анимации.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Марк Прайс. С# 10 и .NET 6. Современная кроссплатформенная разработка. – СПб.: Питер 2023. – 848 с.:
2. Васильев А. Н. Программирование на C# для начинающих. Основные сведения – Москва: Эксмо, 2018. – 592 с.:
3. Васильев А. Н. Программирование на C# для начинающих. Особенности языка – Москва: Эксмо, 2019. – 528 с.:
4. Торн Алан. Основы анимации в Unity / пер. с англ. Р. Рагимова. – М.: ДМК Пресс, 2016. – 176 с.: ил.
5. Феррон Харрисон. Изучаем С# через разработку игр на Unity. 5–е издание. – СПб.: Питер, 2022. 400 с.: ил.
6. Руководство Unity – Unity Manual [Электронный ресурс] / Unity Documentation – Unity Technologies. 2021 – Режим доступа: <https://docs.unity3d.com/Manual/index.html> – Дата доступа: 2023-06-16.

# **ПРИЛОЖЕНИЕ А**

(обязательное)

**Листинг игрового приложения «Тыры-пыры»**

Код модуля «PlayerMovement»

using UnityEngine;  
  
public class **PlayerMovement** : MonoBehaviour  
{  
 private Rigidbody2D rb;  
 private Animator anim;  
 private BoxCollider2D coll;  
 private SpriteRenderer spRend;  
 private AudioSource jumpEffectaudio;  
  
 [SerializeField] private ScriptableObjPlayer **objPlayer**;  
 [SerializeField] private LayerMask **jumpableGround**;  
   
 private float dirX;  
 private float time;  
 private float elapsedTime;  
  
 private enum MovementState { idle, running, jumping, falling }  
 private void **Start**()  
 {  
 rb = GetComponent<Rigidbody2D>();  
 anim = GetComponent<Animator>();  
 coll = GetComponent<BoxCollider2D>();  
 spRend = GetComponent<SpriteRenderer>();  
 jumpEffectaudio = GetComponent<AudioSource>();  
 }  
   
 private void **Update**()  
 {  
   
 if (Input.GetKeyDown(KeyCode.S))  
 {  
 Physics2D.IgnoreLayerCollision(6, 7, true);  
 Invoke("IngoreLayerOff", 0.35f);  
 }  
 UpdateAnimationState();  
 MovementPlayer();  
 }  
  
 private void MovementPlayer()  
 {  
 //print(time);  
 print(objPlayer.speed);  
 dirX = Input.GetAxisRaw("Horizontal");  
 rb.velocity = new Vector2(dirX \* objPlayer.speed, rb.velocity.y);  
  
 if(Input.GetButtonDown("Horizontal"))  
 objPlayer.speed = objPlayer.fixedSpeed;  
  
 if (objPlayer.doubleSpeed && objPlayer.speed <= objPlayer.limitSpeed)  
 {  
 elapsedTime = 0;  
 elapsedTime += Time.deltaTime;  
 float pectangleTime = elapsedTime / 0.8f;  
   
 objPlayer.speed += objPlayer.curve.Evaluate(pectangleTime);  
 }  
   
   
 if (Input.GetButtonDown("Jump") && IsGrounded())  
 {  
 time = 0;  
 rb.velocity = Vector2.up \* objPlayer.jumpForce;  
 }  
   
  
 if (Input.GetButtonDown("Jump") && !IsGrounded() && time < 1 && objPlayer.doubleJump)  
 {  
 rb.velocity = Vector2.up \* objPlayer.jumpForce;  
 time = 10;  
 }  
   
 if (objPlayer.doubleJump)  
 time += Time.deltaTime;  
  
 }  
 private void IngoreLayerOff() { Physics2D.IgnoreLayerCollision(6, 7, false); }  
  
 private void UpdateAnimationState()  
 {  
 MovementState state;  
 if (dirX > 0f)  
 {  
 state = MovementState.running;  
 spRend.flipX = false;  
 }  
 else if (dirX < 0f)  
 {  
 state = MovementState.running;  
 spRend.flipX = true;  
 }  
 else  
 {  
 state = MovementState.idle;   
 }  
  
 if (rb.velocity.y > .1f)  
 {  
 state = MovementState.jumping;  
 jumpEffectaudio.Play();  
 }  
 else if (rb.velocity.y < -.1f)  
 {  
 state = MovementState.falling;  
 }  
 anim.SetInteger("state", (int)state);  
 }  
  
 private bool IsGrounded()  
 {  
 return Physics2D.BoxCast(coll.bounds.center, coll.bounds.size, 0f, Vector2.down, .1f, jumpableGround);  
 }  
}

Код модуля «ScriptableObjPlayer»

using UnityEngine;

[CreateAssetMenu(fileName = "PlayerScriptableObject", menuName = "ScriptableObjects/Player")]

public class ScriptableObjPlayer : ScriptableObject

{

[Header("Настройка скорости персонажа: ")]

public float speed = 7f;

public float fixedSpeed = 7f;

public float limitSpeed = 10f;

[SerializeField] public AnimationCurve curve;

[Header("Кол-во улучшений")]

public int countUpdate = 3;

public int fruits = 0;

[Space(20)]

[Header("Настройка прыжка персонажа: ")]

public float jumpForce = 14f;

[Space(20)]

[Header("Усиления персонажа: ")]

public bool doubleJump = false;

public bool doubleSpeed = false;

public void Reset()

{

speed = 7f;

fixedSpeed = 7f;

limitSpeed = 10f;

countUpdate = 3;

fruits = 0;

jumpForce = 14f;

doubleJump = false;

doubleSpeed = false;

}

}

Код модуля «ItemCollector»

using UnityEngine;  
using UnityEngine.UI;  
  
public class **ItemCollector** : MonoBehaviour  
{  
 [SerializeField] private Text **cherryText**;  
 [SerializeField] private AudioSource **collectCherry**;  
 [SerializeField] private ScriptableObjPlayer **Player**;  
  
 private void **Start**()  
 {  
 cherryText.text = PlayerPrefs.GetInt("Fruits").ToString();  
 Player.fruits = PlayerPrefs.GetInt("Fruits");  
 }  
 private void **OnTriggerEnter2D**(Collider2D col)  
 {  
 switch (col.gameObject.tag)  
 {  
 case "Cherry":  
 collectCherry.Play();  
 col.GetComponent<BoxCollider2D>().enabled = false;  
 Player.fruits++;  
 cherryText.text = Player.fruits.ToString();  
 break;  
   
 case "StrawBerry":  
 collectCherry.Play();  
 col.GetComponent<BoxCollider2D>().enabled = false;  
 Player.fruits+=5;  
 cherryText.text = Player.fruits.ToString();  
 break;  
 }  
 }  
}

Код модуля «PlayerLife»

using UnityEngine;

using UnityEngine.SceneManagement;

public class PlayerLife : MonoBehaviour

{

private Rigidbody2D rb;

private Animator anim;

private float timeRestart = 0f;

private bool inSpace = true;

private bool isPressed = false;

[SerializeField] private AudioSource deathEffect;

[SerializeField] private GameObject enabledBG;

private void Start()

{

anim = GetComponent<Animator>();

rb = GetComponent<Rigidbody2D>();

}

private void OnCollisionEnter2D(Collision2D collision)

{

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

{

Die();

}

}

private void Update()

{

if (Input.GetKeyDown(KeyCode.R) && !isPressed)

{

Invoke("Die", 0.3f);

isPressed = true;

}

if (transform.position.y <= -10 && inSpace)

{

Die();

inSpace = false;

}

}

private void RestartLevel()

{

SceneManager.LoadScene(SceneManager.GetActiveScene().name);

}

private void Die()

{

enabledBG.SetActive(false);

deathEffect.Play();

anim.SetTrigger("death");

rb.bodyType = RigidbodyType2D.Static;

}

}

Код модуля «CameraController»

using UnityEngine;

public class CameraController : MonoBehaviour

{

[SerializeField] private Transform player;

private void Update()

{

if(player.position.y !> -3)

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

else

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

}

}