**МИНИСТЕРСТВО СЕЛЬСКОГО ХОЗЯЙСТВА**

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ**

**ИМЕНИ И. Т. ТРУБИЛИНА»**

**Факультет прикладной информатики**

**Кафедра системного анализа и обработки информации**

**КУРСОВОЙ ПРОЕКТ**

по дисциплине «Технологии программирования»

на тему Программирование игрового приложения “Дудл джамп”

Направление подготовки 09.03.02 «Информационные системы и технологии»

Направленность «Создание, модификация и сопровождение информационных систем, администрирование баз данных»

Выполнил:

Шелудько А.А

группа ИТ2202

Руководитель:

*К.п.н., доцент кафедры*

*системного анализа и*

*обработки информации*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Салий В.В.

**Дата защиты\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Шелудько А.А

**Краснодар 2023**

**РЕФЕРАТ**

43 с., 21 рис., 9 прил.

ИГРА, ГРАФИЧЕСКИЙ ДИЗАЙН, ПЛАТФОРМЕР, ПРЫЖКИ

Целью работы является разработка игрового приложения «Дудл Джамп» в среде MS Visual Studio 2022.

Объект исследования – разработка игры.

Предмет исследования – объектно-ориентированные средства языка программирования для реализации механики игры и интерфейса.

Разработанная программа имеет вид приложения для развлечения пользователя. Относится к виду платформенных игр. Прохождение включает в себя различных персонажей, препятствий и бонусов, с помощью которых пользователь добивается лучшего счета очков.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc553901374)

[1. ОСНОВНЫЕ ПОЛОЖЕНИЯ ПРОГРАММИРОВАНИЯ ИГРОВЫХ ПРИЛОЖЕНИЙ. 6](#_Toc1140895148)

[1.1 Общие сведения 6](#_Toc1354700142)

[1.2 Теория построения игрового приложения 8](#_Toc715559715)

[1.3 Постановка задачи 10](#_Toc1029319871)

[2. ТЕХНОЛОГИЯ РАЗРАБОТКИ ПРИЛОЖЕНИЯ 12](#_Toc453130464)

[2.1 Алгоритм решения 12](#_Toc905143673)

[2.2 Описание приложения 12](#_Toc871498961)

[2.3 Написание программы 15](#_Toc724896759)

[2.4 Создание файла формата exe. 22](#_Toc1394791228)

[3. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЮ 25](#_Toc424087601)

[ЗАКЛЮЧЕНИЕ 27](#_Toc1112429540)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 29](#_Toc1193151839)

[ПРИЛОЖЕНИЕ 1. Листинг программы Form1.cs 31](#_Toc1311603002)

[ПРИЛОЖЕНИЕ 2. Листинг программы класс Bonus 34](#_Toc1390793555)

[ПРИЛОЖЕНИЕ 3. Листинг программы класс Bullet 34](#_Toc6853055)

[ПРИЛОЖЕНИЕ 4. Листинг программы класс Enemy 36](#_Toc322604466)

[ПРИЛОЖЕНИЕ 5. Листинг программы класс Physics 37](#_Toc1524329164)

[ПРИЛОЖЕНИЕ 6. Листинг программы класс Platform 39](#_Toc1776151430)

[ПРИЛОЖЕНИЕ 7. Листинг программы класс PlatformController 41](#_Toc2136536268)

[ПРИЛОЖЕНИЕ 8. Листинг программы класс Player 44](#_Toc1468075594)

[ПРИЛОЖЕНИЕ 9. Листинг программы класс Transform 45](#_Toc1473561508)

# ВВЕДЕНИЕ

В наше время игры являются одним из самых популярных и распространенных способов развлечения. Из-за своей простоты и доступности они захватывают сердца многих людей и удерживают их в своих виртуальных мирах на длительное время. Разработка игр – это захватывающий процесс, позволяющий не только отдохнуть от повседневной рутины, но и раскрыть свои творческие и программистские способности, а также применить знания и навыки в области программирования.

В курсовом проекте рассмотрен процесс разработки игры в среде Visual Studio с использованием технологии Windows Forms — это одна из самых популярных интегрированных сред разработки, которая позволяет разрабатывать приложения и игры для различных операционных систем.

Windows Forms — это библиотека классов, которая позволяет создавать графические пользовательские интерфейсы для приложений на платформе .NET. В процессе работы мы рассмотрим основные компоненты Windows Forms и научимся создавать игровой процесс с использованием элементов управления и функций программирования в Visual Studio.

Игра,которая описывается в данной работе, относится к типу платформер. Разработка платформенных игр — это сложный и многогранный процесс, который включает в себя множество задач, начиная от проектирования уровней и подбора арт-стиля, заканчивая программированием и оптимизацией. Актуальность темы: Разработка игр - одна из самых популярных и быстроразвивающихся областей в индустрии программного обеспечения. Сегодня игры разрабатываются для различных платформ, включая персональные компьютеры, мобильные устройства и игровые консоли. Одним из самых популярных инструментов для разработки игр является Visual Studio в среде Windows Forms. Его использование позволяет разработчикам создавать игры, которые хорошо интегрируются с другими Windows-приложениями и имеют простой, понятный пользовательский интерфейс.

Цель: Данная работа направлена на изучение основных принципов программирования игр, использование базовых техник и технологий, необходимых для создания игры, а также на практический опыт работы с Visual Studio в среде Windows Forms.

Задачи: Изучение теории, принципов программирования игровых приложений, основных компонентов и элементов Windows Forms в Visual Studio; разработка концепции игры, определение ее целей и основных механик; создание пользовательского интерфейса для игры на основе Windows Forms; программирование функциональности игры, включая игровую логику, управление игроком и противниками; создание графики для игрового приложения.

# 1. ОСНОВНЫЕ ПОЛОЖЕНИЯ ПРОГРАММИРОВАНИЯ ИГРОВЫХ ПРИЛОЖЕНИЙ.

# 1.1 Общие сведения

Платформенные игры остаются одними из самых популярных и любимых жанров игр в мире. Они представляют собой игры, где игрок управляет персонажем, который бегает, прыгает, собирает предметы и преодолевает препятствия на уровнях. И хотя этот жанр появился еще в 80-х годах прошлого века, он до сих пор остается актуальным и популярным у игроков всех возрастов.

Платформенные игры продолжают развиваться, добавляя новые уровни, новые возможности и новые персонажи. Они также перекочевали с консолей на мобильные устройства и ПК, что делает их еще более доступными для широкой аудитории.

Одной из причин популярности платформенных игр является их уникальный стиль и механика игры. Они могут быть простыми и легкими для игры, но в то же время требовательными к ловкости и реакции игрока. Эти игры могут быть как одиночными, так и многопользовательскими.

Платформенные игры являются одним из самых прибыльных жанров в мире видеоигр. Наиболее популярными платформами, на которых играются платформенные игры, являются PlayStation, Xbox, Nintendo Switch и PC. В настоящее время популярность многих платформенных игр растет, и многие из них становятся бестселлерами.

На торговых площадках, таких как Steam, PlayStation Store, Xbox Store, Nintendo eShop и других, платформенные игры также пользуются огромным спросом. Например, игра Hollow Knight на платформе Nintendo Switch уже продана в количестве более 1 миллиона копий.

Кроме того, многие платформенные игры, такие как Super Mario и Sonic the Hedgehog, являются очень популярными среди игроков уже нескольких поколений и продолжают оставаться бестселлерами на многих торговых площадках.

Данное игровое приложение будет разрабатываться с помощью интегрированной среды разработки Visual Studio с использованием Windows Forms.

Visual Studio Windows Forms — это интегрированная среда разработки (IDE), предоставляемая компанией Microsoft, которая позволяет разработчикам создавать графические приложения для операционной системы Windows с помощью технологии Windows Forms.

Windows Forms — это библиотека классов для создания графических пользовательских интерфейсов (GUI) в приложениях .NET Framework. Она предоставляет широкий набор элементов управления (таких как кнопки, текстовые поля, списки и т. д.), которые разработчик может использовать для создания функциональных форм и окон.

Visual Studio Windows Forms предоставляет множество инструментов и возможностей, которые помогают разработчикам создавать Windows-приложения с высокой производительностью и качеством. Эти инструменты включают в себя дизайнер форм, который позволяет дизайнерам легко создавать пользовательский интерфейс с помощью простых операций, а также множество функций для отладки и тестирования приложений.

Кроме того, Visual Studio Windows Forms предоставляет поддержку многопоточности, что позволяет создавать многопоточные приложения, которые могут работать более эффективно и быстро.

В целом, Visual Studio Windows Forms предоставляет разработчикам мощную среду для создания Windows-приложений с простым и интуитивно понятным интерфейсом. Это делает его одним из самых популярных инструментов для разработки Windows-приложений.

Целью разработки является готовая игра, которая будет создаваться с помощью объектно-ориентированного типа программирования. В ней будет сочетаться простой, но привлекающий внимание дизайн, понятные для пользователя механики, бесконечные уровни с противниками, которые будут вызывать у игрока желание вернуться в игру и продолжать ставить новые рекорды.

Задачи для достижения поставленной цели:

1. Просмотр теории по созданию игровых приложений.
2. Изучение основных компонентов и элементов Windows Forms.
3. Разработка концепции игры, определение работы основных механик.
4. Создание пользовательского интерфейса для игры
5. Построение игровой логики и управления
6. Создание графики

1.2 Теория построения игрового приложения

При создании проекта следует понимать какой шаблон проекта следует использовать. В нашем случае мы задействуем Windows Form App (.NET Framework). Платформа «.NET Framework» — это технология, которая поддерживает создание и выполнение веб-служб и приложений Windows. Она состоит из двух основных компонентов: среды CLR — механизма, управляющего выполняющимися приложениями, и библиотеки классов. У нее есть преимущества: в ней легко проектировать ранее созданные приложения, существует большой выбор инструментов, наличие кроссплатформенной функции.

В Visual Studio существует множество инструментов для разработки программ. Основными из таковых являются:

1. Редактор форм — инструмент для создания и редактирования пользовательских интерфейсов визуальных форм.  
 2. Панель свойств — позволяет настраивать свойства элементов управления, таких как размер, цвет, шрифт и т. д.  
 3. Окно «Решения» — показывает все файлы проекта, включая код, ресурсы, изображения и т. д.  
 4. Окно «Обозреватель объектов» — позволяет просматривать и управлять элементами управления на форме.  
 5. Окно «Инструменты сервера» — предоставляет доступ к инструментам и сервисам, которые могут быть использованы в приложении.  
 6. Окно «Тестовый эксплорер» — используется для запуска и управления тестами приложения.  
 7. Окно «Консоль пакетного менеджера» — используется для установки и управления пакетами NuGet.  
 8. Компоненты — представляют собой готовые элементы управления, которые можно добавить на форму и настроить их свойства.  
 9. Мастер создания форм — помогает создавать новые формы с заданными параметрами и компонентами.  
 10. Инструменты отладки — позволяют отслеживать выполнение кода, устанавливать точки останова и т. д.

У каждого проекта есть свое «решение». В нем будут находиться все наши файлы, папки, используемые в проекте. В нем будут находиться все наши характеристики или же свойства объектов, ссылки на них, конфигурации и другое.

При разработке игрового приложения будут использоваться так называемые «классы». Класс — некий шаблон, по которому определяется форма объекта, с помощью слова «class». Далее в фигурных скобках определяется содержимое класса. Мы будем использовать данное определение для каждого объекта, над которым мы будем взаимодействовать.

В разработке используется объектно-ориентированный вид программирования — cпособ кодирования, основанный на представлении программы в виде совокупности объектов, причем каждый объект будет являться экземпляром определенного класса.

Для формирования графической части приложения кроме самой формы мы будем использовать так называемые «спрайты». Это картинки формата png, которые будут выполнять функцию визуализации персонажей, платформ и другого.

Реализовать проект помогут библиотеки: using System; using System.Collections.Generic; using System.Drawing; using System.Linq; using System.Text; using System.Threading.Tasks; using System.Windows.Forms.

# 1.3 Постановка задачи

Задача проекта заключается в создании игрового проекта. Необходимо создать окно приложения, в котором будет отображаться игровой процесс. Модель должна включать персонажа, платформы, противников, бонусное продвижение вверх с помощью ускорителя, пули, убивающие противников, счетчик очков, фон приложения, привязку к клавишам клавиатуры для управления персонажем.

Задача поможет реализовать навыки программирования на языке C#, использование Windows Forms, различных библиотек, развития использования графической части.

План

1. Создание проекта с формой Windows Form App (.NET Framework).
2. Загрузка спрайтов, создание отдельной папки с классами. Постепенное создание нужных классов в процессе кодирования.
3. Реализация функционала. Написание кода к каждому классу, подтягивание нужных библиотек.
4. Тестирование. После разработки приложение нужно подвергнуть тестированию, чтобы убедиться в его работоспособности и наличии необходимой функциональности.
5. Развертывание. После разработки приложение конвертируется и становится готовым для распространения, использования пользователем.

# 2. ТЕХНОЛОГИЯ РАЗРАБОТКИ ПРИЛОЖЕНИЯ

# 2.1 Алгоритм решения

1. Создаем проект в среде Microsoft Visual Studio с помощью формы Windows Form App (.NET Framework). Открываем его и помещаем в resources.resx все спрайты.

2. Создаем 2 папки, в одной будут находиться классы, в другой изображения персонажей, фона, платформ.

3. Создаем классы: transform, player, platformcontroller, platform, physics, enemy, bullet, bonus, в папку, описанную выше.

4. Пишем код к классам: transform, player, platformcontroller, platform, physics, enemy, bullet, bonus, параллельно продумывая физику объектов, их размеры, передвижение, позиции.

5. Тестируем программу на корректность.

6. Опубликовываем программу и создаем файл формата exe.

# 2.2 Описание приложения

Игровое приложение «Дудл Джамп», написанное с помощью MS Visual Studio 2022, формы Windows Form App (.NET Framework), на языке C#, представляет собой игру, которая предназначена для занятия человека на досуге.

Суть игры заключается в том, чтобы подняться как можно выше, собрать наибольшее количество очков и победить врагов.

При запуске приложения пользователь увидит начальную позицию, платформы, счет.

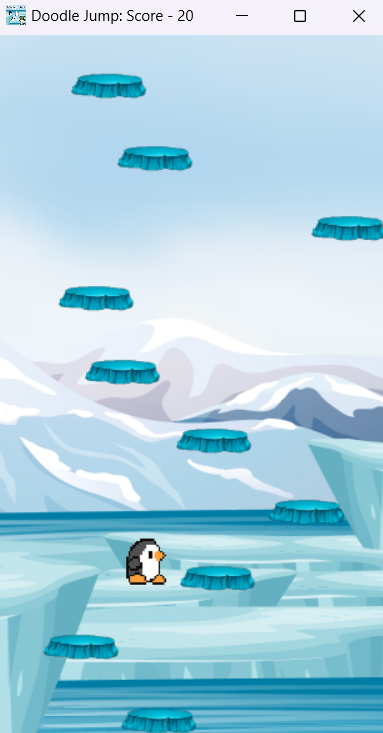


рисунок 2.2.1 - начало игры

Во время прохождения игры пользователю будет показывать счет, который будет увеличиваться с каждым шагом персонажа.

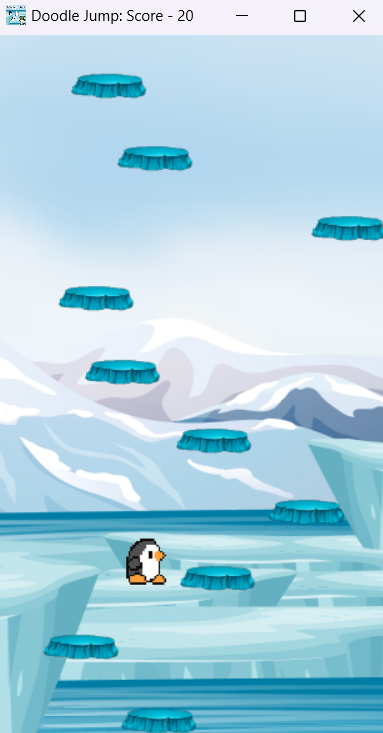


рисунок 2.2.2 - счет игрока

Уровень за уровнем игроку будут предлагаться различные ускорители, которые будут ему помогать пройти выше.



рисунок 2.2.3 - пружина, которая перебрасывает персонажа вверх

Как пользователь пройдет определенный отрезок пути, ему станут мешать подняться противники, стоящие на платформах сверху него

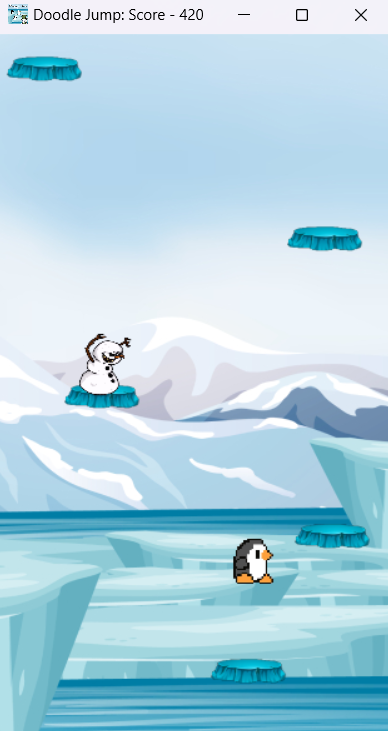


рисунок 2.2.4 - противники

Чтобы побороть антагонистов игры, главный персонаж использует функцию стрельбы для поражения противников.

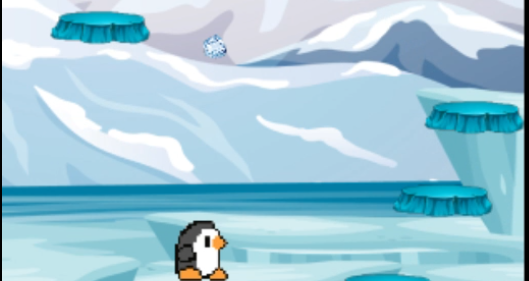


рисунок 2.2.5 - стрельба льдом

Одним из бонусов для получения большего количества очков является ранец, с помощью которого игрок может пролететь определенное расстояние мимо врагов.



рисунок 2.2.6 - ранец, который помогает пролететь выше

# 2.3 Написание программы

Добавление графической части

При создании проекта с помощью .NET Framework существует изначальный файл для хранения графических объектов - Resources.resx. В котором будут находиться наши спрайты. Чтобы их добавить, нужно: открыть Resources.resx, выбрать добавить ресурс, выбрать добавить существующий файл. Можно создать отдельную папку в проекте со спрайтами или же добавлять из своей галереи.

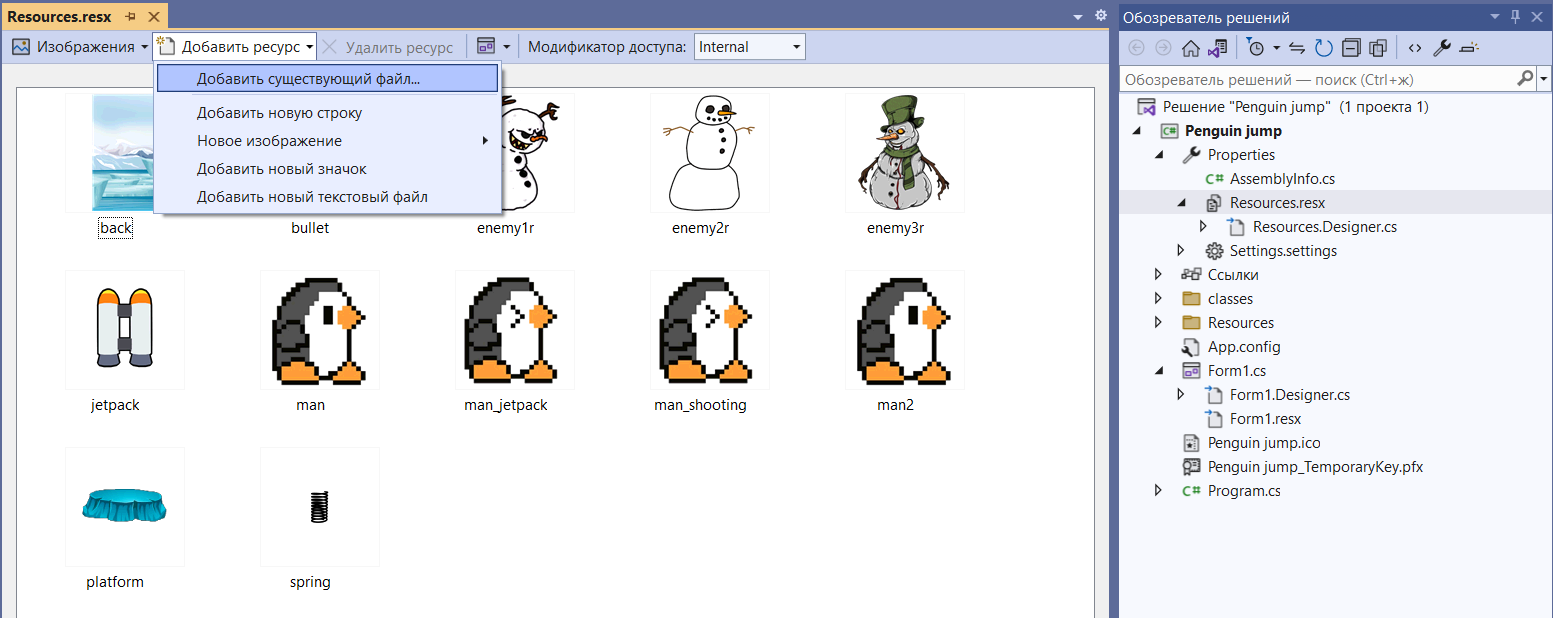


рисунок (1.7) - использование ресурсов

Создаем новую папку в обозревателе решений и называем ее classes.

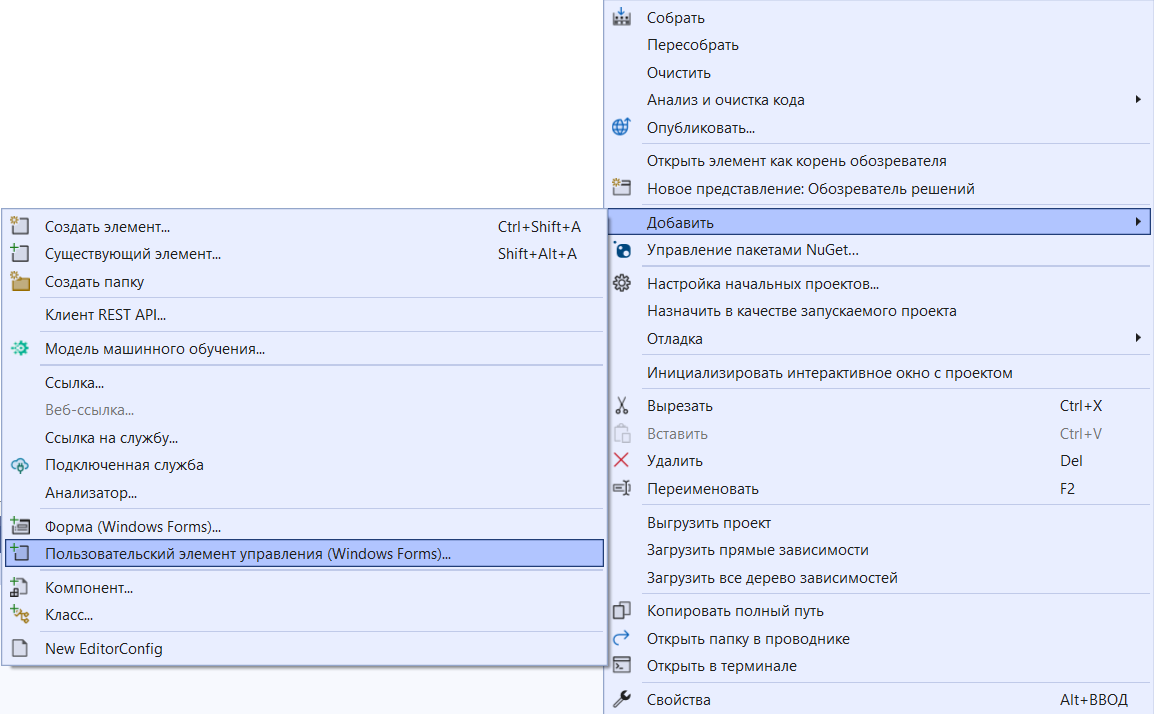
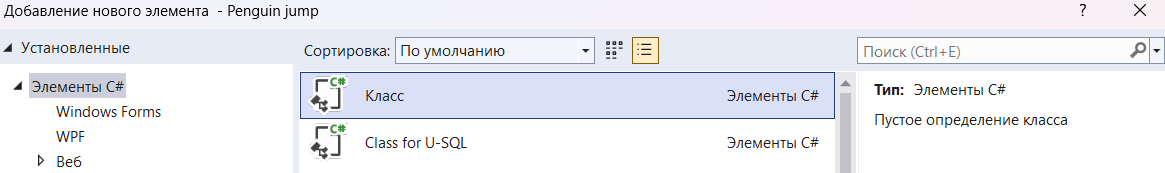


рисунок 2.3.1 - создание папки с классами

Далее кликаем правой кнопкой мыши по папке и выбираем добавить, после пользовательский элемент управления.

рисунок 2.3.2 - добавление классов



Переименовываем с начального названия Class1 в: transform. Повторяем создание таких классов: player, platformcontroller, platform, physics, enemy, bullet, bonus.

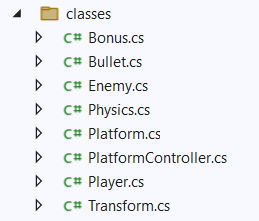


рисунок 2.3.3 - результат

Написание кода к классам

Первый класс, в котором мы напишем код, будет transform. В этом классе будет храниться координаты или позиции объектов, с которыми мы будем взаимодействовать в будущем, а также размер объектов. Присваиваем параметры переменным position и size.

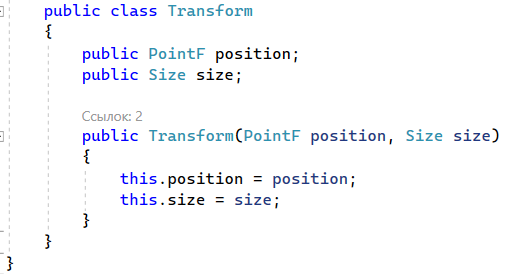


рисунок 2.3.4 - класс Transform

Следующий класс, с которым мы будем взаимодействовать - Platform. Этот класс отвечает за платформы, по которым будет прыгать персонаж. Он будет рисовать спрайт (ледяную платформу в нашем случае), также в нем мы создаем переменную transform, которая будет помогать сохранять размер и позицию, sizeX и sizeY, которые отвечают за координаты платформы и переменная IsTouchByPlayer, которая отвечает за касание персонажа платформы. С помощью функции DrawSprite прорисовываем платформу.

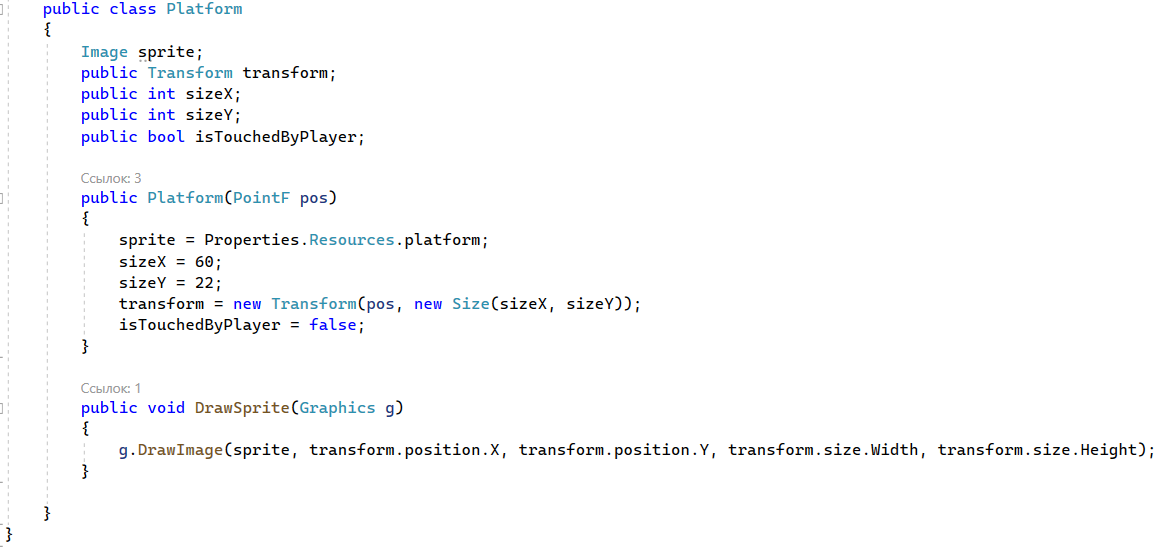


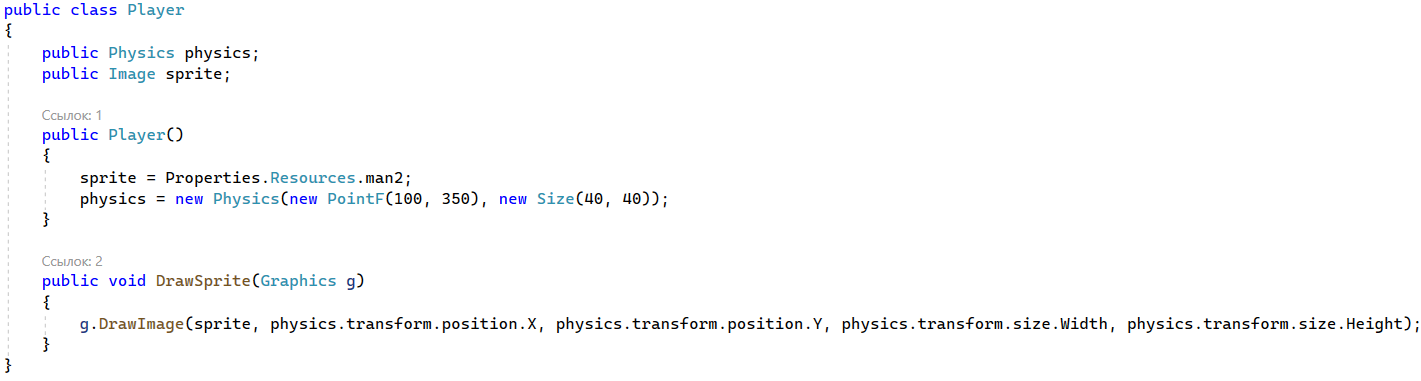
рисунок 2.3.5 - класс Platform

Класс PlatformController. Через этот класс мы управляем всеми движениями платформ, создаем новые или удаляем старые во время движения персонажа. Класс объявляем статическим, чтобы не создавать экземпляр класса. В нем будет содержаться статические листы (List) для платформ, противников, пули, бонусам по очкам. Это позволит им храниться на карте. Переменная startPlatformPosY отвечает за генерацию платформ, а score за очки. Далее мы создаем функции: AddPlatform( отвечающая за добавление платформ), CreateBullet (создание пули), GenerateStartSequence(В начале игры создает определенное количество платформ на карте по способу рандома. Задействуем смещение по y, чтобы подниматься выше), GenerateRandomPlatform( Создание с помощью случайных чисел платформ на листе, а также добавление с помощью switch противников и бонусов на случайных платформах), CreateBonus(создание рандомных бонусов и указание их координат, расположения), CreateEnemy(создание рандомных противников на случайных платформах и указание их координат, расположения), RemoveEnemy, RemoveBullet, ClearPlatforms(эти 3 класса отвечают за отчистку спрайтов с листа). Их код будет указан в приложении из-за объемности.

Рассмотрим класс Physics. Он отвечает за физическое устройство объектов во всем приложении, их передвижение. Например, прыжок персонажа и взаимодействие с платформами. Воспользуемся переменными transform (отвечает за позицию), gravity (гравитация для постоянного прыжка), а (ускорение), dx (переменная для передвижения персонажа влево и вправо). В методе Physics присваиваем начальные значения переменным. CalculatePhysics будет отвечать за передвижение по x персонажем, а также реализуем прыжок, задаем там координаты при различных позициях персонажа, StandartCollidePlayerWithObjects( взаимодействие с противниками или с бонусами, используем условные if циклы и вложенные for), ApplyPhysics (вызываем CalculatePhysics), Collide (функция, отвечающая за передвижение игрока по платформам, при условиях, когда игрок коснулся платформы или этого не произошло, прибавляем очки, а также создаем взамен пройденной платформы новую), Addforce( механика прыжка), StandartCollide(функция пролета пули). Также часть кода будет приведена в приложении к курсовой проекте.

Следующий класс - Player. Класс отвечает за прорисовку персонажа в разработанном проекте. Здесь прописывается размер персонажа, рисуется спрайт и привязывается к нему новый компонент Physics.

рисунок 2.3.6 - класс Player



Теперь переходим к классам, в которых реализовано средство атаки противников (назовем его bullet) и к самим антогонистам игры.

Рассмотрим код класса bullet. На данном этапе создаем новый компонент Physics и спрайт. Берем картинку и привязываем к ней указанный размер и позицию. Как всегда, прописываем функцию прорисовки картинки.

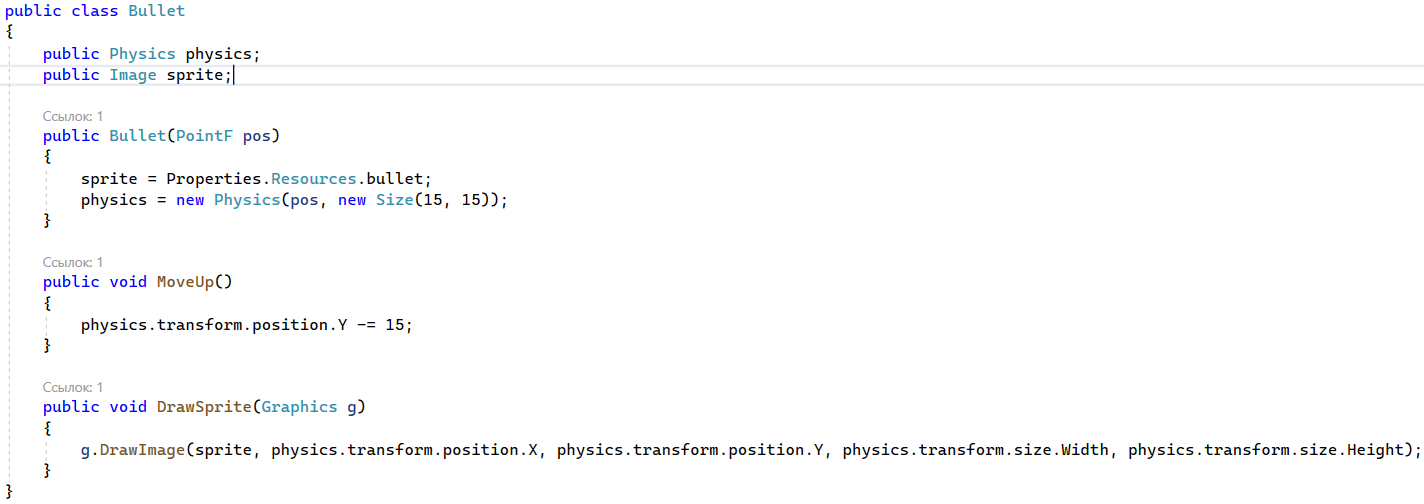


рисунок 2.3.7 - класс Bullet

Класс Enemy. Он отвечает за противников. Унаследуем его от класса Player. В нем создаем компонент Physics, указываем размер спрайтов, откуда их берем, привязываем позиции. Делаем это через оператора Switch.

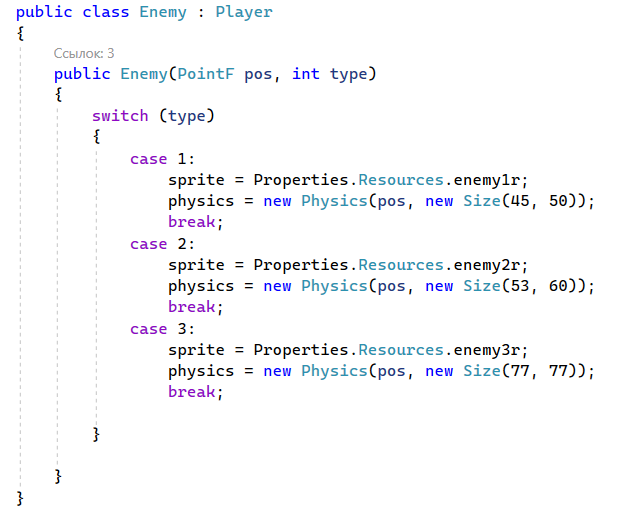
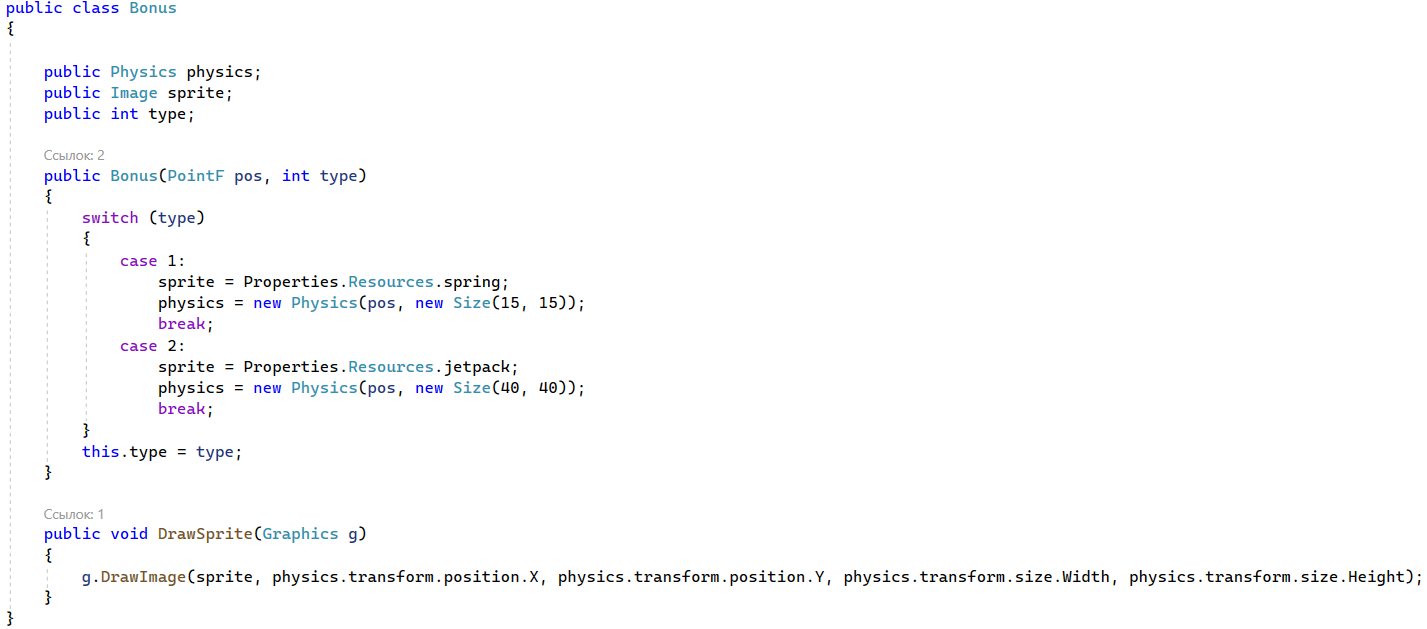


рисунок 2.3.8 - класс Enemy

И наконец последний, но не по важности класс - Bonus. Этот класс непосредственно отвечает за появления пружины и джетпака для героя, которые помогут ему собрать как можно больше очков и пролететь определенный участок. Очень похож на предыдущий класс. Первое, как всегда, создаем компонент Physics, далее объявляем компонент для спрайта и прописываем общедоступную переменную type типа integer. Опят же через switch прописываем для пружины и джетпака позицию и размеры спрайта. Не забываем о функции DrawSprite.

рисунок 2.3.9 - класс Bonus



Написание кода в форме.

Когда все персонажи, платформы и физика продуманы, то можно перейти к коду самой формы. В нем нам предстоит поработать с фоном нашей игры, привязки игры к клавишам, создать имитацию камеры, которая будет следить за нашем персонажем, создать таймер и многое другое. Начнем поэтапно.

Заводим переменную игрока и таймера. Прописываем обработчики клавиш, таймера и размеры фона.



рисунок 2.3.1.1 - Форма

Прописываем функцию Init( т. е. инициализация). Создаем новый лист для платформ. Добавляем начальную платформу с ее координатами. Переменной startPlatformPosY задаем значение, обнуляем очки. Вызываем функцию GenerateStartSequence, которая сгенерирует платформы. Создаем новый экземпляр класса Player. Отчистим элементы из колекиции List: bonuses, bullets, enemies. А также присвоим player.

Переходим к обработчикам клавиш. В OnKeyboardUp прописываем, что, когда пользователь отпускает клавишу, то персонаж будет прыгать на одном месте. Прописываем спрайт. С помощью Switch реализуем нажатие на клавишу Space, при которой персонаж будет стрелять льдом.

Второй обработчик OnKeyboardPressed будет отвечать за передвижение персонажа вправо и влево, а также за стрельбу при нажатии этих клавиш.

Переходим к обработчику Update. Прописываем текст, который будет находиться вверху формы и делаем обновление счетчика score. Теперь пропишем условия поражения пользователя, функцию столкновения пользователя и противника,

Теперь напишем код для обработчика имитации камеры слежения ха персонажем, создаим переменную offset, которая будет выступать за камеру, добавим условия для слежения за пулями, врагами, бонусами и платформами.

В обработчике OnRepaint укажем прорисовку спрайтов.

Код к форме находится в приложении к работе.

# 2.4 Создание файла формата exe.

Для этого нужно нажать правой кнопкой мыши на решение проекта, выбрать категорию опубликовать.

Далее нужно выбрать, где будет находиться приложение и вариант его установки.

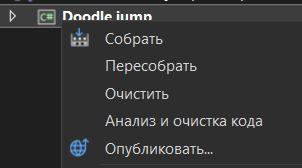


рисунок (2.4.1) - Публикация

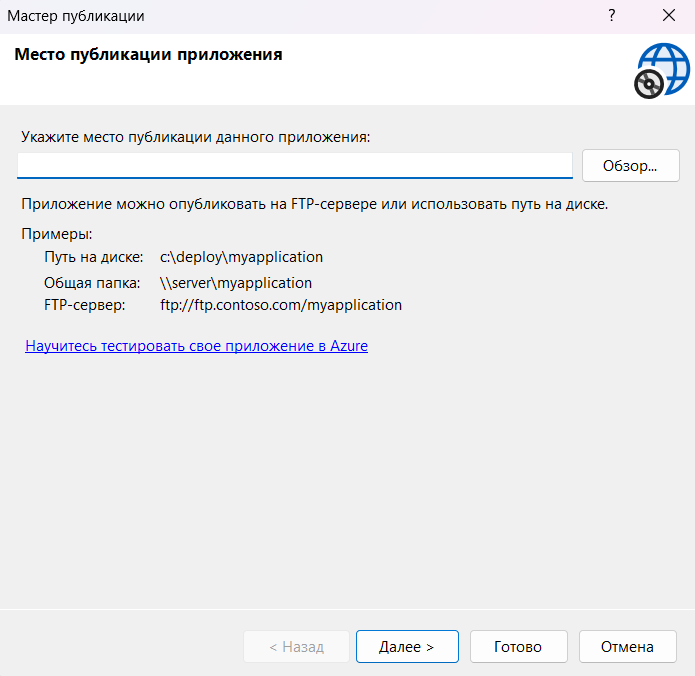


рисунок 2.4.2 - Выбор пути расположения

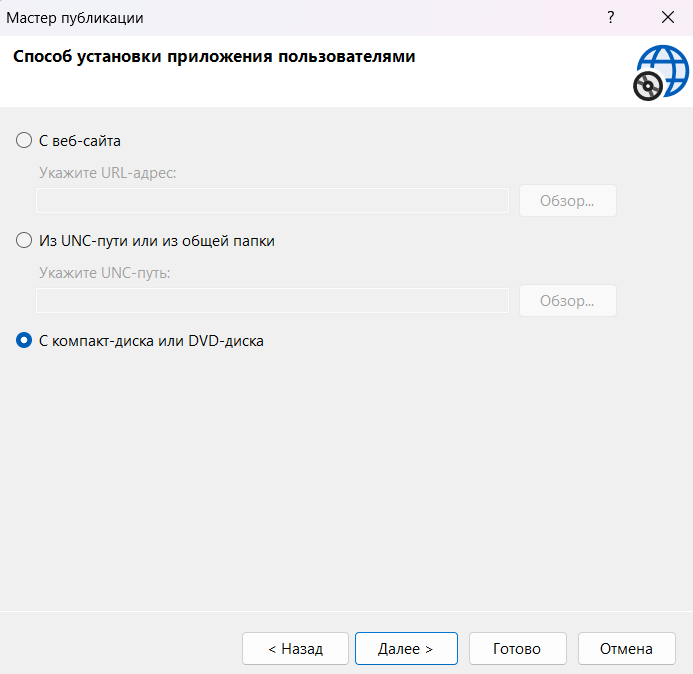


рисунок 2.4.3- Способ установки

В конечном результате должна получиться папка с данными файлами

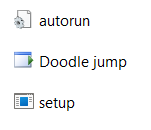


рисунок 2.4.2 - Результат

3. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЮ

Для запуска проекта следует:

Установить официальную версию Microsoft Visual Studio 2022 c официального сайта “ https://visualstudio.microsoft.com/ru/”, нажмите кнопку “Cкачать” (рисунок 3.1)

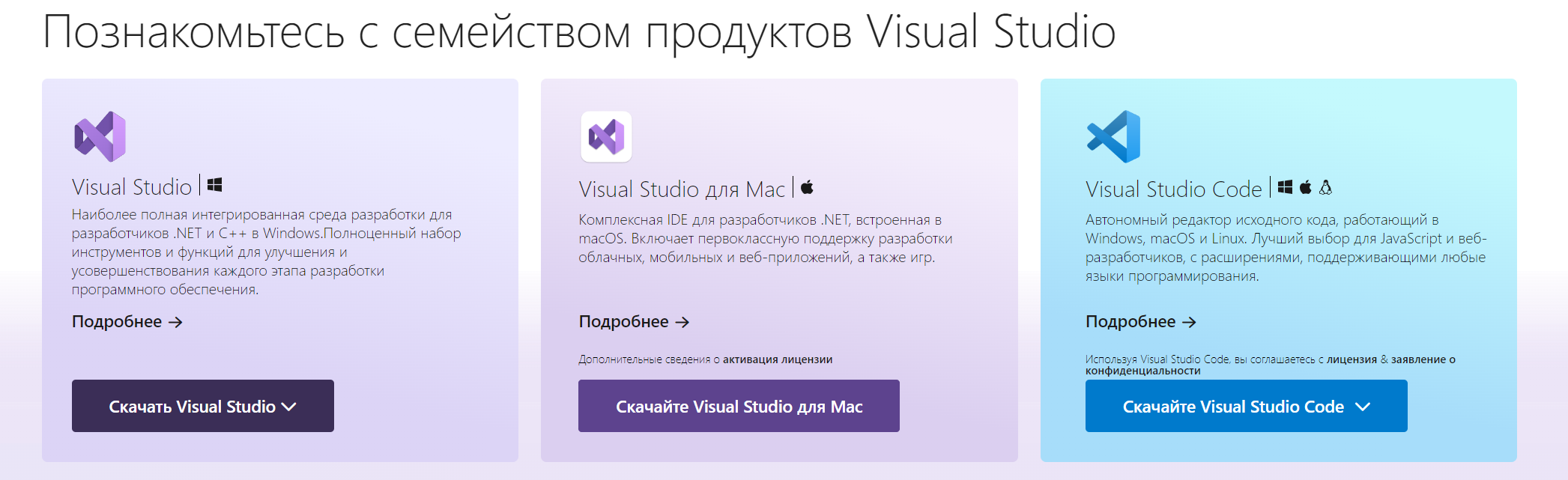


рисунок 3.1 - скачивание Visual Studio

Далее нужно выбрать необходимую версию продукта и, кликнув на нее, начнется автоматическая загрузка программы.

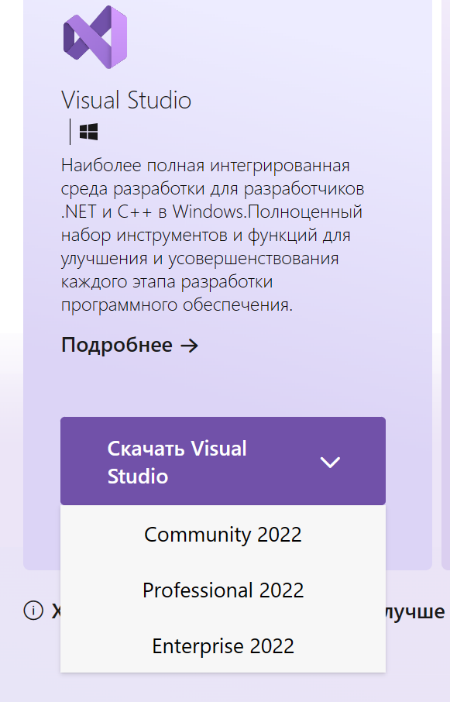


рисунок 3.2 - выбор версии программы

Далее запустите файл Doodle jump.sln., находящийся в папке Проект. После загрузки приложения нажмите кнопку пуск, которая указана на рисунке 3.3

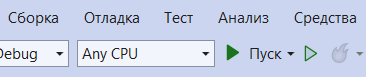


рисунок 3.3 - запуск программы

Путь к проекту: Курсовая работа Шелудько А.А. ИТ2202 / Проект /Решение /Doodle jump.sln

Для открытия файла формата exe следует:

Кликнуть левой кнопкой мыши по файлу Doodle jump, который находится: Курсовая работа Шелудько А.А. ИТ2202 / Файл формата exe/ Doodle jump

Управление игровым приложением

1. Приложение управляется левой стрелкой (передвижение влево), правой стрелкой клавиатуры (передвижение вправо), пробелом (прыжок).
2. Рекомендуемое разрешение приложения - оконный режим.
3. Закрытие приложения производится кликом по левой кнопке мыши на крест, расположенный в правом верхнем углу приложения.

# ЗАКЛЮЧЕНИЕ

В результате выполнения данной курсовом проекте было создано игровое приложение Дудл Джамп.

В процессе работы были рассмотрены основные принципы работы в приложении Microsoft Visual Studio, закреплены принципы работы с Windows Forms в среде разработки NET.Framework, а также были использованы основные инструменты создания игрового приложения. Вся программа основана на объектно-ориентированном типе программирования, что позволило узнать положительные и отрицательные стороны данного способа.

Для разработки игры были использованы следующие этапы: создание классов, работа с графической частью ресурсов, написание скриптов и создание конечного файла, готового к распространению.

В конечном результате программа представляет собой полноценную игру, которая полностью соответствует заданному плану, а ошибок или неполадок в коде не обнаружено.

В целом, создание данного приложения способствовало обогащению знаний и навыков в области разработки игр на платформе Visual Studio на языке C#. Создание игрового приложения позволило расширить знания о программировании, познакомило меня с неизвестными ранее видами инструментов и кодирования, реализацией графического дизайна и его привязке к скриптам. Был получен ценный опыт на практике по созданию игровых программ, а также: в разработке концепции игры, определении ее целей и основных механик, программировании функциональности, включая игровую логику, управление игроком и противниками, создании имитации камеры отслеживания персонажа и во многом другом.

Таким образом, созданное приложение сможет заинтересовать пользователей, которые хотят провести свободное время в игровой среде.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. <https://learn.microsoft.com/ru-ru/visualstudio/get-started/csharp/visual-studio-ide?source=recommendations&view=vs-2022> (Введение в интегрированную среду разработки Visual Studio (C#), 2023)
2. Обзор новых возможностей Visual Studio 2022 / Е. А. Пьянова, Е. В. Антонов, О. А. Климов, И. А. Гурин. – Екатеринбург: Урфу, 2022. – 322 с.
3. <https://learn.microsoft.com/ru-ru/visualstudio/designers/walkthrough-windows-forms-designer?view=vs-2022> (Учебник. Начало работы с конструктором Windows Forms, 2023)
4. РАЗРАБОТКА ПРИЛОЖЕНИЙ НА C# В СРЕДЕ VISUALSTUDIO / А.М. НУЖНЫЙ, Н. И. ГРЕБЕННИКОВА, В.Ф. БАРАБАНОВ, О. Б. КРЕМЕР. – Воронеж: Воронежский государственный технический университет, 2019. – 88 с.
5. Разработка приложений на C# в среде Visual Studio: учебное пособие пособие / А. М. Нужный, О. Б. Кремер, Н. И. Гребенникова, В. Ф. Барабанов. – В: ВГТУ, ЭБС АСВ, 2019. – 89 с.
6. Кудрина, Е. В. Основы алгоритмизации и программирования на языке c#: учеб. пособие для бакалавриата и специалитета / Е. В. Кудрина, М. В. Огнева. — М.: Издательство Юрайт, 2019. — 322 с.
7. Рихтер, Дж. CLR via C#. Программирование на платформе Microsoft. NET Framework 4.5 на языке C# / Дж. Рихтер. - СПб.: Питер, 2018. - 351 c.
8. Тюкачев, Н. А. C#. Основы программирования: учебное пособие для вузов / Н. А. Тюкачев, В. Г. Хлебостроев. — 4-е изд., стер. — Санкт-Петербург: Лань, 2021. — 272 с.
9. <https://metanit.com/sharp/windowsforms/> (Руководство по программированию в Windows Forms, 2023)
10. <https://e.lanbook.com/book/129040> (Разработка на языке C# приложений с графическим интерфейсом (использование Windows Forms): Учебник, Мурадханов С.Э., 2019. - 396 c)

# ПРИЛОЖЕНИЕ 1. Листинг программы Form1.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using ДУДЛ.classes;

namespace ДУДЛ

{

public partial class Form1 : Form

{

Player player;

Timer timer1;

public Form1()

{

InitializeComponent();

Init();

timer1 = new Timer();

timer1.Interval = 15;

timer1.Tick += new EventHandler(Update);

timer1.Start();

this.KeyDown += new KeyEventHandler(OnKeyboardPressed);

this.KeyUp += new KeyEventHandler(OnKeyboardUp);

this.BackgroundImage = Properties.Resources.back;

this.Height = 600;

this.Width = 330;

this.Paint += new PaintEventHandler(OnRepaint);

}

public void Init()

{

PlatformController.platforms = new System.Collections.Generic.List<Platform>();

PlatformController.AddPlatform(new System.Drawing.PointF(100, 400));

PlatformController.startPlatformPosY = 400;

PlatformController.score = 0;

PlatformController.GenerateStartSequence();

PlatformController.bullets.Clear();

PlatformController.bonuses.Clear();

PlatformController.enemies.Clear();

player = new Player();

}

private void OnKeyboardUp(object sender, KeyEventArgs e)

{

player.physics.dx = 0;

player.sprite = Properties.Resources.man2;

switch (e.KeyCode.ToString())

{

case "Space":

PlatformController.CreateBullet(new PointF(player.physics.transform.position.X + player.physics.transform.size.Width / 2, player.physics.transform.position.Y));

break;

}

}

private void OnKeyboardPressed(object sender, KeyEventArgs e)

{

switch (e.KeyCode.ToString())

{

case "Right":

player.physics.dx = 6;

break;

case "Left":

player.physics.dx = -6;

break;

case "Space":

player.sprite = Properties.Resources.man\_shooting;

//PlatformController.CreateBullet(new PointF(player.physics.transform.position.X + player.physics.transform.size.Width/2,player.physics.transform.position.Y));

break;

}

}

private void Update(object sender, EventArgs e)

{

this.Text = "Doodle Jump: Score - " + PlatformController.score;

if ((player.physics.transform.position.Y >= PlatformController.platforms[0].transform.position.Y + 200) || player.physics.StandartCollidePlayerWithObjects(true, false))

Init();

player.physics.StandartCollidePlayerWithObjects(false, true);

if (PlatformController.bullets.Count > 0)

{

for (int i = 0; i < PlatformController.bullets.Count; i++)

{

if (Math.Abs(PlatformController.bullets[i].physics.transform.position.Y - player.physics.transform.position.Y) > 500)

{

PlatformController.RemoveBullet(i);

continue;

}

PlatformController.bullets[i].MoveUp();

}

}

if (PlatformController.enemies.Count > 0)

{

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

{

if (PlatformController.enemies[i].physics.StandartCollide())

{

PlatformController.RemoveEnemy(i);

break;

}

}

}

player.physics.ApplyPhysics();

FollowPlayer();

Invalidate();

}

public void FollowPlayer()

{

int offset = 400 - (int)player.physics.transform.position.Y;

player.physics.transform.position.Y += offset;

for (int i = 0; i < PlatformController.platforms.Count; i++)

{

var platform = PlatformController.platforms[i];

platform.transform.position.Y += offset;

}

for (int i = 0; i < PlatformController.bullets.Count; i++)

{

var bullet = PlatformController.bullets[i];

bullet.physics.transform.position.Y += offset;

}

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

{

var enemy = PlatformController.enemies[i];

enemy.physics.transform.position.Y += offset;

}

for (int i = 0; i < PlatformController.bonuses.Count; i++)

{

var bonus = PlatformController.bonuses[i];

bonus.physics.transform.position.Y += offset;

}

}

private void OnRepaint(object sender, PaintEventArgs e)

{

Graphics g = e.Graphics;

if (PlatformController.platforms.Count > 0)

{

for (int i = 0; i < PlatformController.platforms.Count; i++)

PlatformController.platforms[i].DrawSprite(g);

}

if (PlatformController.bullets.Count > 0)

{

for (int i = 0; i < PlatformController.bullets.Count; i++)

PlatformController.bullets[i].DrawSprite(g);

}

if (PlatformController.enemies.Count > 0)

{

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

PlatformController.enemies[i].DrawSprite(g);

}

if (PlatformController.bonuses.Count > 0)

{

for (int i = 0; i < PlatformController.bonuses.Count; i++)

PlatformController.bonuses[i].DrawSprite(g);

}

player.DrawSprite(g);

}

}

}

# ПРИЛОЖЕНИЕ 2. Листинг программы класс Bonus

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ДУДЛ.classes

{

public class Bonus

{

public Physics physics;

public Image sprite;

public int type;

public Bonus(PointF pos, int type)

{

switch (type)

{

case 1:

sprite = Properties.Resources.spring;

physics = new Physics(pos, new Size(15, 15));

break;

case 2:

sprite = Properties.Resources.jetpack;

physics = new Physics(pos, new Size(40, 40));

break;

}

this.type = type;

}

public void DrawSprite(Graphics g)

{

g.DrawImage(sprite, physics.transform.position.X, physics.transform.position.Y, physics.transform.size.Width, physics.transform.size.Height);

}

}

}

# ПРИЛОЖЕНИЕ 3. Листинг программы класс Bullet

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ДУДЛ.classes

{

public class Bullet

{

public Physics physics;

public Image sprite;

public Bullet(PointF pos)

{

sprite = Properties.Resources.bullet;

physics = new Physics(pos, new Size(15, 15));

}

public void MoveUp()

{

physics.transform.position.Y -= 15;

}

public void DrawSprite(Graphics g)

{

g.DrawImage(sprite, physics.transform.position.X, physics.transform.position.Y, physics.transform.size.Width, physics.transform.size.Height);

}

}

}

# ПРИЛОЖЕНИЕ 4. Листинг программы класс Enemy

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Security.Policy;

using System.Text;

using System.Threading.Tasks;

namespace ДУДЛ.classes

{

public class Enemy : Player

{

public Enemy(PointF pos, int type)

{

switch (type)

{

case 1:

sprite = Properties.Resources.enemy1r;

physics = new Physics(pos, new Size(45, 50));

break;

case 2:

sprite = Properties.Resources.enemy2r;

physics = new Physics(pos, new Size(53, 60));

break;

case 3:

sprite = Properties.Resources.enemy3r;

physics = new Physics(pos, new Size(77, 77));

break;

}

}

}

}

# ПРИЛОЖЕНИЕ 5. Листинг программы класс Physics

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ДУДЛ.classes

{

public class Physics

{

public Transform transform;

float gravity;

float a;

public float dx;

bool usedBonus = false;

public Physics(PointF position, Size size)

{

transform = new Transform(position, size);

gravity = 0;

a = 0.4f;

dx = 0;

}

public void ApplyPhysics()

{

CalculatePhysics();

}

public void CalculatePhysics()

{

if (dx != 0)

{

transform.position.X += dx;

}

if (transform.position.Y < 700)

{

transform.position.Y += gravity;

gravity += a;

if (gravity > -25 && usedBonus)

{

PlatformController.GenerateRandomPlatform();

PlatformController.startPlatformPosY = -200;

PlatformController.GenerateStartSequence();

PlatformController.startPlatformPosY = 0;

usedBonus = false;

}

Collide();

}

}

public bool StandartCollidePlayerWithObjects(bool forMonsters, bool forBonuses)

{

if (forMonsters)

{

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

{

var enemy = PlatformController.enemies[i];

PointF delta = new PointF();

delta.X = (transform.position.X + transform.size.Width / 2) - (enemy.physics.transform.position.X + enemy.physics.transform.size.Width / 2);

delta.Y = (transform.position.Y + transform.size.Height / 2) - (enemy.physics.transform.position.Y + enemy.physics.transform.size.Height / 2);

if (Math.Abs(delta.X) <= transform.size.Width / 2 + enemy.physics.transform.size.Width / 2)

{

if (Math.Abs(delta.Y) <= transform.size.Height / 2 + enemy.physics.transform.size.Height / 2)

{

if (!usedBonus)

return true;

}

}

}

}

if (forBonuses)

{

for (int i = 0; i < PlatformController.bonuses.Count; i++)

{

var bonus = PlatformController.bonuses[i];

PointF delta = new PointF();

delta.X = (transform.position.X + transform.size.Width / 2) - (bonus.physics.transform.position.X + bonus.physics.transform.size.Width / 2);

delta.Y = (transform.position.Y + transform.size.Height / 2) - (bonus.physics.transform.position.Y + bonus.physics.transform.size.Height / 2);

if (Math.Abs(delta.X) <= transform.size.Width / 2 + bonus.physics.transform.size.Width / 2)

{

if (Math.Abs(delta.Y) <= transform.size.Height / 2 + bonus.physics.transform.size.Height / 2)

{

if (bonus.type == 1 && !usedBonus)

{

usedBonus = true;

AddForce(-30);

}

if (bonus.type == 2 && !usedBonus)

{

usedBonus = true;

AddForce(-60);

}

return true;

}

}

}

}

return false;

}

public bool StandartCollide()

{

for (int i = 0; i < PlatformController.bullets.Count; i++)

{

var bullet = PlatformController.bullets[i];

PointF delta = new PointF();

delta.X = (transform.position.X + transform.size.Width / 2) - (bullet.physics.transform.position.X + bullet.physics.transform.size.Width / 2);

delta.Y = (transform.position.Y + transform.size.Height / 2) - (bullet.physics.transform.position.Y + bullet.physics.transform.size.Height / 2);

if (Math.Abs(delta.X) <= transform.size.Width / 2 + bullet.physics.transform.size.Width / 2)

{

if (Math.Abs(delta.Y) <= transform.size.Height / 2 + bullet.physics.transform.size.Height / 2)

{

PlatformController.RemoveBullet(i);

return true;

}

}

}

return false;

}

public void Collide()

{

for (int i = 0; i < PlatformController.platforms.Count; i++)

{

var platform = PlatformController.platforms[i];

if (transform.position.X + transform.size.Width / 2 >= platform.transform.position.X && transform.position.X + transform.size.Width / 2 <= platform.transform.position.X + platform.transform.size.Width)

{

if (transform.position.Y + transform.size.Height >= platform.transform.position.Y && transform.position.Y + transform.size.Height <= platform.transform.position.Y + platform.transform.size.Height)

{

if (gravity > 0)

{

AddForce();

if (!platform.isTouchedByPlayer)

{

PlatformController.score += 20;

PlatformController.GenerateRandomPlatform();

platform.isTouchedByPlayer = true;

}

}

}

}

}

}

public void AddForce(int force = -10)

{

gravity = force;

}

}

}

# ПРИЛОЖЕНИЕ 6. Листинг программы класс Platform

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ДУДЛ.classes

{

public class Platform

{

Image sprite;

public Transform transform;

public int sizeX;

public int sizeY;

public bool isTouchedByPlayer;

public Platform(PointF pos)

{

sprite = Properties.Resources.platform;

sizeX = 60;

sizeY = 22;

transform = new Transform(pos, new Size(sizeX, sizeY));

isTouchedByPlayer = false;

}

public void DrawSprite(Graphics g)

{

g.DrawImage(sprite, transform.position.X, transform.position.Y, transform.size.Width, transform.size.Height);

}

}

}

# ПРИЛОЖЕНИЕ 7. Листинг программы класс PlatformController

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ДУДЛ.classes

{

public static class PlatformController

{

public static List<Platform> platforms;

public static List<Bullet> bullets = new List<Bullet>();

public static List<Enemy> enemies = new List<Enemy>();

public static List<Bonus> bonuses = new List<Bonus>();

public static int startPlatformPosY = 400;

public static int score = 0;

public static void AddPlatform(PointF position)

{

Platform platform = new Platform(position);

platforms.Add(platform);

}

public static void CreateBullet(PointF pos)

{

var bullet = new Bullet(pos);

bullets.Add(bullet);

}

public static void GenerateStartSequence()

{

Random r = new Random();

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

{

int x = r.Next(0, 270);

int y = r.Next(50, 60);

startPlatformPosY -= y;

PointF position = new PointF(x, startPlatformPosY);

Platform platform = new Platform(position);

platforms.Add(platform);

}

}

public static void GenerateRandomPlatform()

{

ClearPlatforms();

Random r = new Random();

int x = r.Next(0, 270);

PointF position = new PointF(x, startPlatformPosY);

Platform platform = new Platform(position);

platforms.Add(platform);

var c = r.Next(1, 3);

switch (c)

{

case 1:

c = r.Next(1, 10);

if (c == 1)

{

CreateEnemy(platform);

}

break;

case 2:

c = r.Next(1, 10);

if (c == 1)

{

CreateBonus(platform);

}

break;

}

}

public static void CreateBonus(Platform platform)

{

Random r = new Random();

var bonusType = r.Next(1, 3);

switch (bonusType)

{

case 1:

var bonus = new Bonus(new PointF(platform.transform.position.X + (platform.sizeX / 2) - 7, platform.transform.position.Y - 15), bonusType);

bonuses.Add(bonus);

break;

case 2:

bonus = new Bonus(new PointF(platform.transform.position.X + (platform.sizeX / 2) - 15, platform.transform.position.Y - 30), bonusType);

bonuses.Add(bonus);

break;

}

}

public static void CreateEnemy(Platform platform)

{

Random r = new Random();

var enemyType = r.Next(1, 4);

switch (enemyType)

{

case 1:

var enemy = new Enemy(new PointF(platform.transform.position.X + (platform.sizeX / 2) - 20, platform.transform.position.Y - 40), enemyType);

enemies.Add(enemy);

break;

case 2:

enemy = new Enemy(new PointF(platform.transform.position.X + (platform.sizeX / 2) - 35, platform.transform.position.Y - 50), enemyType);

enemies.Add(enemy);

break;

case 3:

enemy = new Enemy(new PointF(platform.transform.position.X + (platform.sizeX / 2) - 35, platform.transform.position.Y - 60), enemyType);

enemies.Add(enemy);

break;

}

}

public static void RemoveEnemy(int i)

{

enemies.RemoveAt(i);

}

public static void RemoveBullet(int i)

{

bullets.RemoveAt(i);

}

public static void ClearPlatforms()

{

for (int i = 0; i < platforms.Count; i++)

{

if (platforms[i].transform.position.Y >= 700)

platforms.RemoveAt(i);

}

for (int i = 0; i < bonuses.Count; i++)

{

if (bonuses[i].physics.transform.position.Y >= 700)

bonuses.RemoveAt(i);

}

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

{

if (enemies[i].physics.transform.position.Y >= 700)

enemies.RemoveAt(i);

}

}

}

}

# ПРИЛОЖЕНИЕ 8. Листинг программы класс Player

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ДУДЛ.classes

{

public class Player

{

public Physics physics;

public Image sprite;

public Player()

{

sprite = Properties.Resources.man2;

physics = new Physics(new PointF(100, 350), new Size(40, 40));

}

public void DrawSprite(Graphics g)

{

g.DrawImage(sprite, physics.transform.position.X, physics.transform.position.Y, physics.transform.size.Width, physics.transform.size.Height);

}

}

}

# ПРИЛОЖЕНИЕ 9. Листинг программы класс Transform

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ДУДЛ.classes

{

public class Transform

{

public PointF position;

public Size size;

public Transform(PointF position, Size size)

{

this.position = position;

this.size = size;

}

}

}