Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Программирование на языках высокого уровня

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту
на тему

Игра-платформер

БГУИР КП 1-40 02 01 217 ПЗ

Студент: гр. 250502 Коркотко А.Д.

Руководитель: Богдан Е. В.

МИНСК 2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В последние годы в мире все более актуальна разработка игр. Спрос на качественные игры только растет. Это служит не только развлекательным целям, но и помогает в образовании. Доказана эффективность геймификации в процессе обучения. Для максимальной производительности, игры часто пишут на языке C++.

C++ – компилируемый язык программирования общего назначения. Он поддерживает такие парадигмы программирования, как процедурное программирование, объектно-ориентированное программирование (ООП), обобщенное программирование. Язык программирования широко используется для разработки программного обеспечения: создание разнообразных прикладных программ, разработка операционных систем, драйверов устройств, а также видеоигр и многое другое.

Популярность языка обусловлена его эффективностью, гибкостью и широким спектром применений, для которых он может быть использован. C++ известен своей высокой производительностью, поскольку позволяет выполнять низкоуровневые манипуляции с оборудованием и памятью, что делает его подходящим для разработки ресурсоемких приложений.

ООП – подход к программированию как к моделированию информационных объектов, решающий на новом уровне основную задачу структурного программирования: структурирование информации с точки зрения управляемости, что существенно улучшает управляемость самим процессом моделирования, что, в свою очередь, особенно важно при реализации крупных проектов. Основные принципы структурирования в случае ООП связаны с различными аспектами базового понимания предметной задачи, которое требуется для оптимального управления соответствующей моделью:

- абстракция для выделения в моделируемом предмете важного для решения конкретной задачи по предмету, в конечном счёте – контекстное понимание предмета, формализуемое в виде класса;
- инкапсуляция для быстрой и безопасной организации собственно иерархической управляемости: чтобы было достаточно простой команды «что делать», без одновременного уточнения как именно делать, так как это уже другой уровень управления;
- наследование для быстрой и безопасной организации родственных понятий: чтобы было достаточно на каждом иерархическом шаге учитывать только изменения, не дублируя всё остальное, учтённое на предыдущих шагах;
- полиморфизм для определения точки, в которой единое управление лучше распараллелить или наоборот – собрать воедино.

1 ПОСТАНОВКА ЗАДАЧИ

Исследовать принцип работы ООП и библиотеки SFML и реализацию 2D-игр на языке C++. Реализовать графический интерфейс и взаимодействие пользователя с игрой.

Игра-платформер должна содержать классы, отражающие основные компоненты: игровой персонаж, противники, оружие, игровое меню.

Для обеспечения максимальной производительности и для того, чтобы избежать задержки при использовании графического интерфейса, будем использовать библиотеку SFML. Библиотека упрощает создание игр и работу с графическими материалами.

Необходимо добавить возможность создавать макет игрового уровня, изменять расположение объектов, противников и игрового персонажа, реализовать управление персонажем, атаку, и систему здоровья у персонажа и противников.

При запуске игры пользователю открывается меню с возможностью выбора дальнейших действий. Далее пользователь должен дойти до конца каждого из уровней, преодолевая препятствия и побеждая противников. При прохождении одного уровня появляется возможность пройти следующий. В любой момент пользователь может поставить игру на паузу и выбрать один из вариантов во вновь открывшемся игровом меню.

В случае смерти персонажа дается возможность пройти уровень заново.

Данные о текущем уровне сохраняются, а с каждым новым уровнем параметры здоровья персонажа возвращаются к своему начальному значению. При полном прохождении игры появляется возможность вернуться на любой из уровней и пройти его снова.

2 ОБЗОР ЛИТЕРАТУРЫ

2.1 Анализ существующих аналогов

В настоящее время существует огромное множество игр-платформеров. Все началось с игры-платформера Donkey King, созданная компанией Nintendo. В игре главным героем был маленький сантехник по имени Марио, который карабкаясь по лестницам и уклоняясь от бочек должен спасти свою возлюбленную от гориллы по имени Донки Конг, откуда и название. На рисунке 2.1 можно наблюдать один из уровней этого платформера. Он включает в себя графический интерфейс, где можно наблюдать персонажей, платформы и препятствия, количество здоровья персонажа, номер уровня, количество очков и бонус.

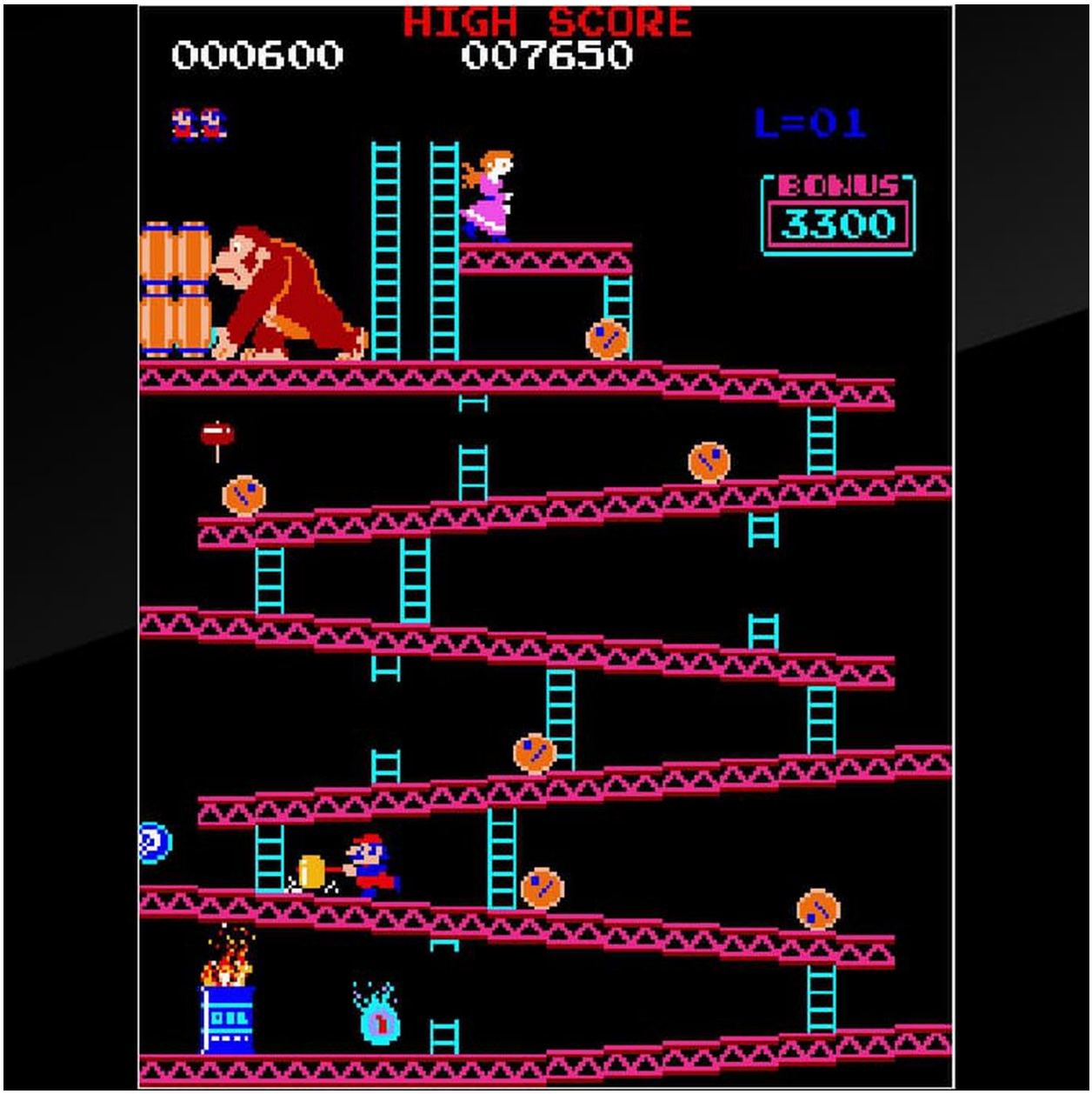

Рисунок 2.1 – Игра Donkey King

Игра была настолько успешной, что стала прародителем всего жанра платформеров и таких знаменитых примеров, как Super Mario Bros, Sonic the Hedgehog.

Рисунок 2.2 – Super Mario Bros

На рисунке 2.2 можно наблюдать сцену из игры Super Mario Bros, где видно заметное улучшение графики по сравнению с ее предшественником.Super Mario Bros видеоигра в жанре платформера, разработанная и выпущенная в 1985 году японской компанией Nintendo для платформы Famicom. Занесена в Книгу рекордов Гиннеса как самая продаваемая игра в истории.

В Super Mario Bros кроме Марио появляется новый персонаж Луиджи. Также добавляются несколько новых видов противников, препятствий и игровых механик.

На рисунке 2.3 можно наблюдать более современный аналог 2D платформера Ori and the Blind Forest.

Рисунок 2.3 – игра Ori and the Blind Forest

В игре рассказывается об Ори, маленьком светящемся лесном духе, который пытается восстановить жизнь в умирающем лесу Нибеля. История начинается с мощного шторма, который приводит к рождению Ори. Его находит и усыновляет Нару, нежное лесное существо. Однако случилась так, что лес поглотила тьма, жить в нем стало не комфортно, и Ори, ведомый духом по имени Сейн, должен восстановить былую благоустроенность и омолодить Нибеля.

Игра представляет собой платформер, взаимосвязанный картами и прогрессивным исследованием. Вы начинаете с базового набора навыков, который постепенно расширяется по мере того, как находите бонусы и способности. Они позволяют Ори перемещаться по миру прыжками от стен, рывками, двойными кульбитами, давая возможность проходить ранее недоступные области.

В «Ori and the Blind Forest» прекрасно организованная симфония сложного дизайна уровней, очаровательной графики и эмоционального повествования. Она оживляет мир Нибеля, заставляя каждую область чувствовать себя живой и изобилующей магией. Пример можно увидеть на рисунке 2.4

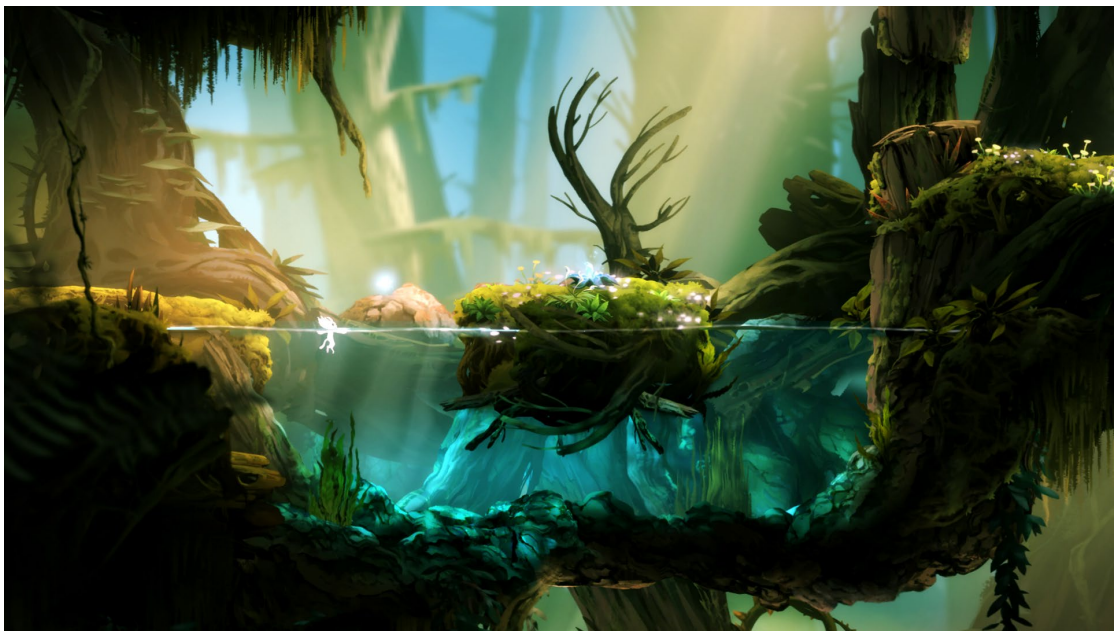

Рисунок 2.4 – сцена из игры Ori and the blind forest

Игровой процесс интуитивно понятен. От геймера лишь требуется умение точно рассчитывать время, быстро реагировать на происходящее и способность ориентироваться на местности. Ori and the Blind Forest считается одной из лучших игр-платформеров из-за эмоциональной глубины сюжета. Здесь продемонстрирована пронзительная история любви, самопожертвования и надежды. Волнующий саундтрек, написанный Гаретом Кокером, усиливает эту эмоциональную вовлеченность, делая путешествие игрока по Нибелю поистине незабываемым.
Количество альтернатив бесчисленное множество, но для реализации поставленной задачи был выбран лишь основной функционал, а именно: управление персонажем и прохождение уровней, преодолевая препятствия и побеждая противников. Весь остальной функционал это лишь надстройки над базовыми примитивами. Например, хорошая графика является результатом большей работы над текстурами и анимациями.

2.2 Обзор методов и алгоритмов решения поставленной задачи

Разработанная игра-платформер является проектом на языке C++ с активно используемой графической частью. Для упрощения процесса работы применяется кроссплатформенная библиотека SFML.
SFML (Simple and Fast Multimedia Library) — свободная кроссплатформенная мультимедийная библиотека. Написана на C++, но доступна также для C, C#, .Net, D, Java, Python, Ruby, OCaml, Go и Rust.
Представляет собой объектно-ориентированный аналог SDL. SFML содержит ряд модулей для простого программирования игр и мультимедиа приложений. Исходный код библиотеки предоставляется под лицензией zlib/png license. Состоит библиотека из пяти модулей: system, window, graphics, audio и

network. Среди них наиболее важными для создания курсового проекта являлись System, Window и Graphics.

Модуль Graphics делает простым отображение графических примитивов и изображений, без чего в случае создания игры-платформера практически невозможно обойтись.

Модуль Window поддерживает рисование с помощью OpenGL, позволяет управлять окнами, создавать окна необходимых размеров и так далее.

Window отвечает за взаимодействие с пользователем, отслеживание пользовательского ввода, отслеживание и обработку событий.

Модуль System является обязательным, так как все модули зависят от него. System отвечает за классы векторов, управление потоками и временем. Благодаря возможности управления временем и его отслеживанию возможна привязка анимаций не к счетчику кадров, а к вышеупомянутому времени, что позволяет запускать игру и получать более четкие анимации даже на слабых устройствах.

В основе создания всего проекта лежит принцип представления всех объектов игры в виде прямоугольников. Каждому из таких прямоугольников задаются 4 параметра: координата X левого верхнего угла, координата Y левого верхнего угла, ширина и высота. При открытии игрового окна по заданным координатам создаются прямоугольники заданных размеров, им соответственно задаются различные тестуры или же объекты просто окрашиваются в определенный цвет. Для пользователя это показывается как игровые уровни, персонажи, объекты.

Для взаимодействия персонажа и противников друг с другом и с объектами окружения используется коллизия. Коллизией называют событие, когда по одним и тем же координатам находятся два прямоугольника. В таком случае над одним из прямоугольников или над обоими одновременно выполняются заранее заданные действия. Пользователь может видеть это как столкновение персонажа со стеной, попадание в ловушку, при котором отнимаются единицы здоровья, или попросту возможность ходить по определенной поверхности, не проваливаясь в пустоту.

Для формирования игрового уровня в коде программы заранее создается карта статичных объектов, обозначенных различными буквами. Это можно увидеть на рисунке 2.9:

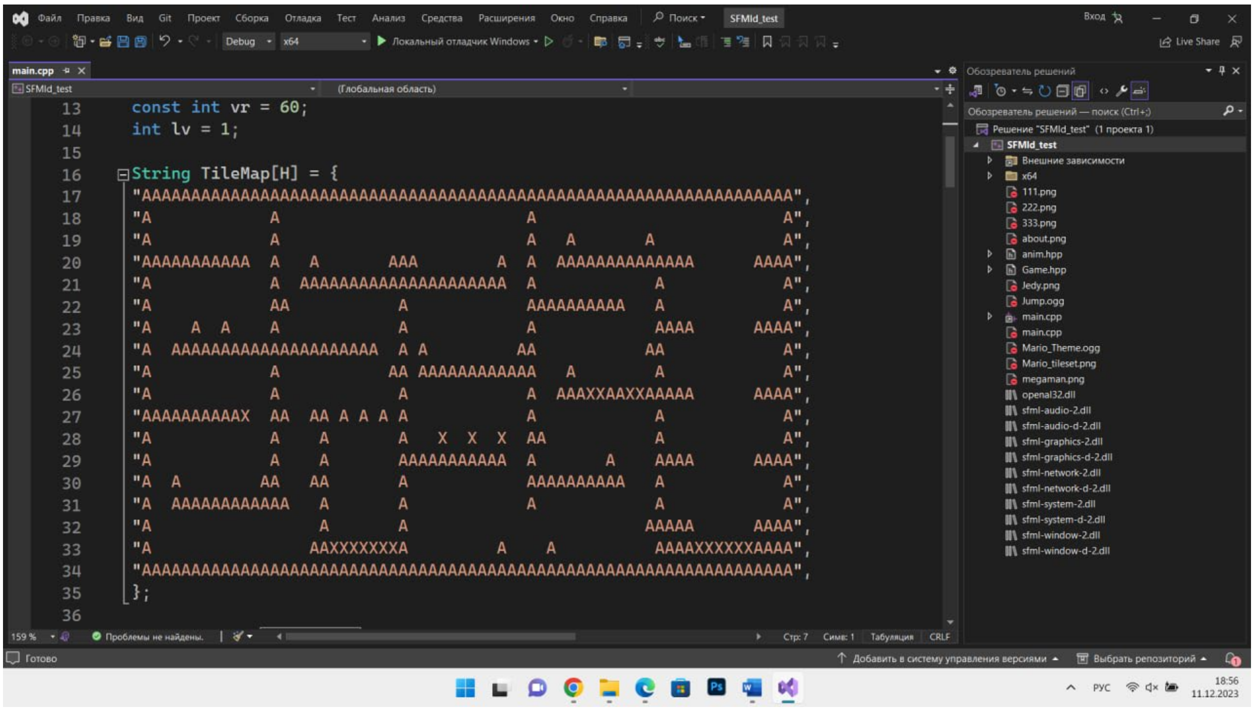

Рисунок 2.9 – карта статичных объектов уровня

Здесь буквами А обозначены объекты твердых поверхностей, а буквами Х объекты ловушек, при взаимодействии с которыми персонаж будет терять очки здоровья.

Библиотека SFML открывает возможность упрощенного создания игры, но главным инструментом для создания проекта является Visual Studio 2022. Visual Studio — это мощное средство разработчика, которое можно применять выполнения всего цикла разработки в одном месте. Это комплексная интегрированная среда разработки (IDE), которую можно использовать для записи, редактирования, отладки и сборки кода, а затем развертывания приложения. Помимо редактирования и отладки кода Visual Studio включает компиляторы, средства завершения кода, управление версиями, расширения и многое другое, чтобы улучшить каждый этап процессе разработки программного обеспечения.

3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

В данном разделе описываются входные и выходные данные программы, диаграмма классов, а также приводится описание используемых классов и их методов.

3.1 Структура входных и выходных данных

Блок кода отвечающий за входные параметры и генерацию уровня:

```
float offsetX = 0, offsetY = 0;
const int H = 18;
const int W = 66;
```

В этот блок кода также входит заранее введенная карта статичных объектов уровня, пример которой можно увидеть на рисунке 2.9
Все анимации и текстуры находятся в заранее заготовленных файлах проекта.

3.2 Разработка диаграммы классов

Диаграмма классов для курсового проекта приведена в Приложении А.

3.3 Описание классов

3.3.1 Класс задания меню

Класс `setMenu` представляет собой информацию о пройденных уровнях и о текущем уровне персонажа.

Поля класса:

- `bool pusklever[5]` – поле с информацией о пройденных уровнях.
- `bool blv[5]`– поле с информацией о текущем уровне персонажа.

Методы:

- `setMenu()` – настраивает информацию об уровнях в начале игры.

3.3.2 Класс игрового персонажа

Класс `Player` является классом игрового персонажа и содержит основную информацию об игроке и необходимые методы

Поля класса:

- `float dx` – скорость движения игрока по оси X.
- `float dy` – скорость движения игрока по оси У.
- `FloatRect rect` – объект, которому принадлежит игрок.
- `bool onGround` – поле с информацией, находится ли персонаж на земле.

- `bool rig` – поле с информацией, в какую сторону повернут персонаж.
- `Sprite sprite` – текстура персонажа, которую видит пользователь.
- `float curFrame` – поле с информацией о текущем кадре.
- `int l` – количество очков здоровья персонажа.
- `bool life` – поле с информацией, жив ли персонаж.

Методы:

- `update (float time)` – контролирует состояние персонажа в любой момент времени.
- `Collision(int dir)` –определяет состояние персонажа при коллизии с любым из объектов

Класс `Player` работает до момента смерти игрока.

3.3.3 Класс оружия игрока

Класс `Pula` является классом оружия игрока, который способен наносить урон противникам.

Он содержит следующие поля:

- `float dx` – координата по оси X;
- `float dy` – координата по оси У;
- `FloatRect rect` –объект, которому принадлежит пуля;
- `int go` – триггер для запуска пули;
- `Sprite sprite` – поле с текстурой для пули.

Методы:

- `update(float time)` – отвечает за состояние пули в любой момент времени.
- `Collision(int dir)` – изменяет состояние пули при коллизии с другими объектами

3.3.4 Класс противников

Класс `Vrag` содержит поля, наследуемые классами `Enemy` и `Boss`

Поля класса:

- `float dx` – координата по оси X.
- `float dy` – координата по оси У.
- `FloatRect rect` – объект, которому принадлежит противник.
- `Sprite sprite` – поле с текстурой противника.
- `float curFrame` – поле с информацией о текущем кадре.
- `bool life` – поле с информацией, жив ли противник.

3.3.5 Класс обычного противника

Класс содержит необходимые методы для функционирования противника

Методы:

- `set(Texture& image,int x, int y)` – создает противника по заданным координатам.
- `update(float time)` – управляет состоянием противника в любой момент времени.
- `Collision()` – управляет состоянием противника при коллизии с другим объектом.

3.3.6 Класс босса

Класс содержит поля и методы, управляющие состоянием босса.

Поля класса:

- `int l` – поле, содержащее количество единиц здоровья босса.
- `bool rig` – поле с информацией, в какую сторону повернут босс.

Методы:

- `set(Texture& image,int x, int y)` – создает босса по заданным координатам
- `update(float time)` – управляет состоянием босса в любой момент времени.
- `Collision()` – управляет состоянием босса при коллизии с другим объектом.

3.3.7 Класс оружия босса

Класс `PulaBoss` является классом оружия босса, который способен наносить урон игроку.

Он содержит следующие поля:

- `float dx` – координата по оси X;
- `float dy` – координата по оси У;
- `FloatRect rect` –объект, которому принадлежит пуля босса;
- `int go` – триггер для запуска пули босса;
- `Sprite sprite` – поле с текстурой для пули босса.

Методы:

- `update(float time)` – отвечает за состояние пули в любой момент времени.
- `Collision()` – изменяет состояние пули при коллизии с другими объектами.
- `set(Texture& image,int x, int y)` – создает пулю босса по заданным координатам.

4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

4.1 Разработка схем алгоритмов

int main() – функция предназначена для организации работы всего проекта

Алгоритм по шагам:

1. Начало.
2. Создание окна игры.
3. Загрузка текстур.
4. Определение игровых объектов.
5. Задание позиций игровым объектам.
6. Пока окно открыто выполнять шаги 7–13.
7. Если выполняются необходимые условия, запустить игру и игровое меню.
8. Загрузка статичных объектов уровня.
9. При выполнении необходимых условий, выполнять действия над игровым персонажем
10. При выполнении необходимых условий, выполнять действия над противниками.
11. При выполнении необходимых условий, выполнять действия над боссом.
12. Вызов методов обновления у персонажа, противников, босса, снарядов персонажа и босса, зависящих от времени.
13. Вызов игрового меню в случае выполнения необходимых для этого условий.
14. Конец.

void update(float time) – функция предназначена для изменения состояния персонажа в любой момент времени.

Алгоритм по шагам:

1. Начало.
2. Координаты объекта персонажа определяются в зависимости от скорости по оси X и времени.
3. Если персонаж находится не на земле, задаем ему ускорение прыжка и затем свободного падения.
4. Если выполняется условие жизни персонажа, выполнять шаги 5-7.
5. Определение текущего кадра персонажа в зависимости от необходимых условий.
6. Если скорость по оси X положительная, изменяем текстуру персонажа соответствующим образом.
7. Если скорость по оси X отрицательная, изменяем текстуру персонажа соответствующим образом.
8. В случае смерти персонажа изменяем текстуру персонажа в соответствие с тем, в какую сторону направлен персонаж.

9. В случае смерти персонажа делаем текстуру статичной.
10. При выходе игрока в меню делаем текстуру персонажа статичной.
11. Конец.

4.2 Разработка алгоритмов

Схема алгоритма метода int main() приведена в приложении Б. – эта функция является главной частью кода проекта. Она выполняется в первую очередь. Функция оперирует с объектами заранее написанных классов, определяет главные параметры игрового окна и самого игрового процесса.

Схема алгоритма метода Player:: update(float time) приведена в приложении В. – эта функция использует платформо-зависимую функцию получения скриншота экрана, а далее использует OpenCV для его преобразования перед отправкой. Важно отправить начальный чанк с длиной буфера для избежания ошибок переполнения буфера.

При разработке алгоритмов были использованы библиотека SFML и интегрированная среда разработки Visual Studio 2022.

Данные алгоритмы являются основными во всем приложении. Алгоритм main() запускается в первую очередь и является основой программы, а алгоритм update как для класса Player, так и для других классов является определяющим и контролирует состояние и взаимодействие всех объектов в любой момент времени.

При разработке функций часто бывает полезно сначала продумать общий алгоритм действий, составить схему алгоритма, а потом реализация становится очень простой.

При разработке алгоритма по шагам и схемы алгоритма важно не включать детали реализации, которые никак не помогают понять суть алгоритма. Например, какие-то специфические особенности разных операционных систем, языков программирования и не только вряд ли будут указаны на схеме алгоритма. Схему алгоритма можно описать простыми словами, иногда с указанием сторонних функций.

В данном случае не были включены особенности того, как загружаются текстуры объектов, как создаются объекты, как работает графическая часть проекта и как создается карта уровней.

6 РЕЗУЛЬТАТЫ РАБОТЫ

При запуске программы нас приветствует стартовое меню программы (рисунок 6.1). Можно наблюдать часть уровня игры, название игры и имеется возможность выбрать одну из двух опций: Играть или Выход. При нажатии на кнопку Выход происходит выход из программы и завершение всех процессов. Вернуться в игру можно в любой момент, заново запустив код программы. При нажатии кнопки Играть открывается меню доступных уровней.

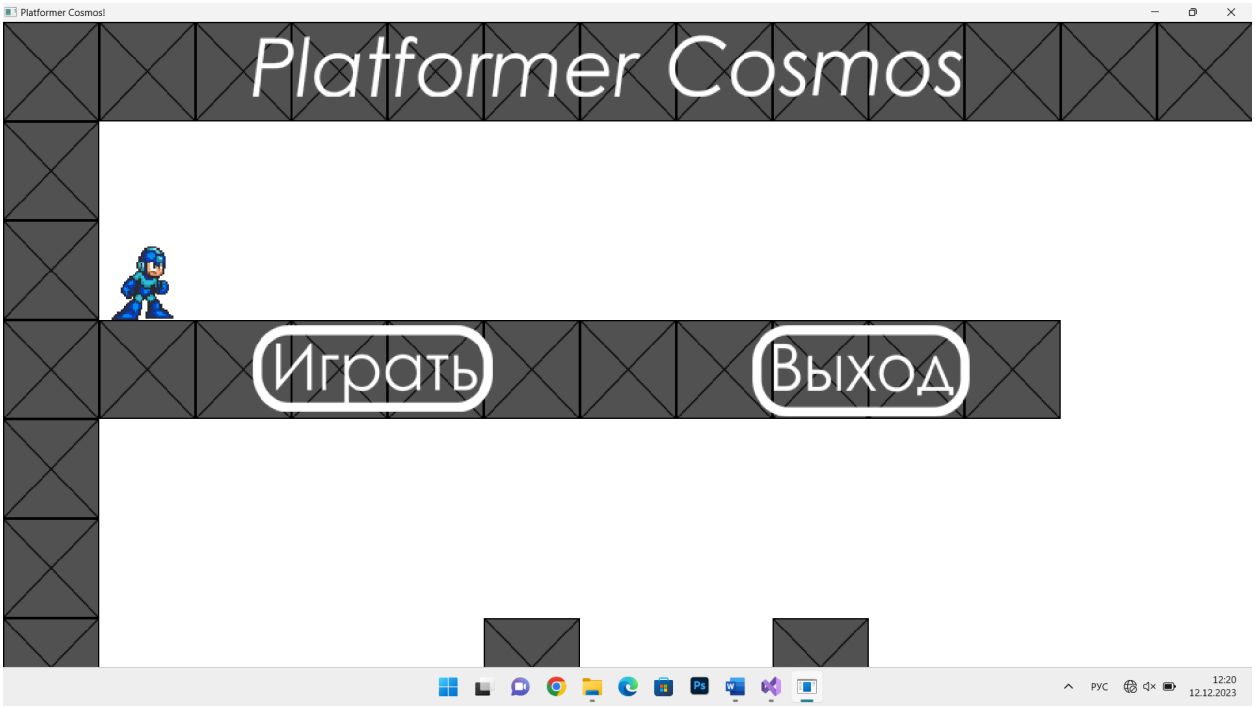

Рисунок 6.1 – Стартовое меню

Меню доступных уровней показано на рисунке 6.2.

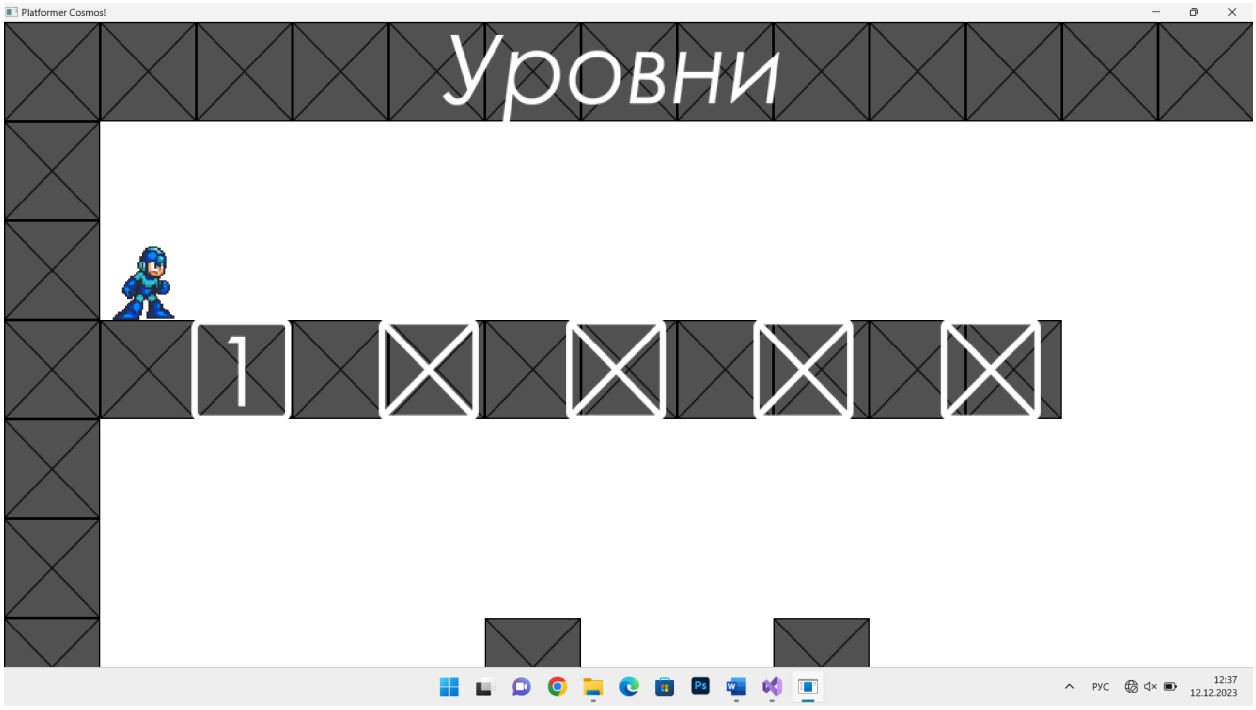

Рисунок 6.2 – меню выбора доступных уровней

Здесь наглядно видно какие уровни доступны для прохождения. В любой момент игры можно перейти с текущего уровня на любой из доступных.

При выборе первого доступного уровня пользователь увидит игровой уровень, отображенный на рисунке 6.3.

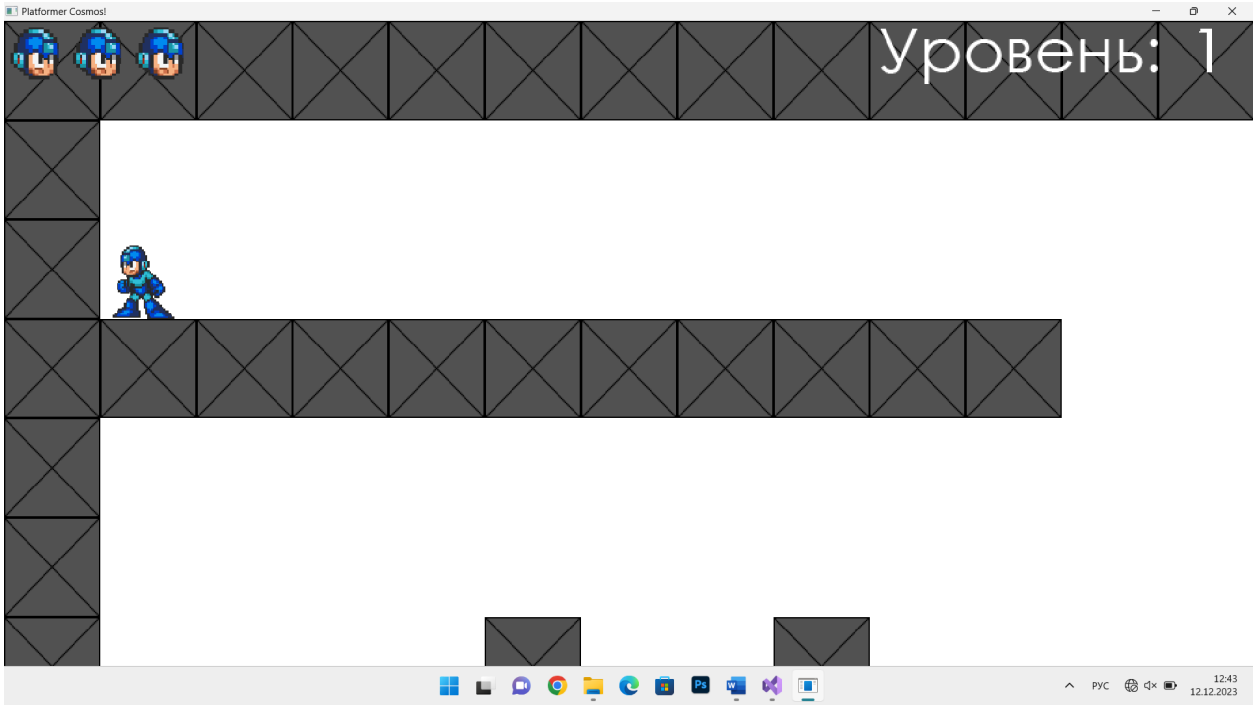

Рисунок 6.3 – начальный уровень

В левом верхнем углу окна отображается количество очков здоровья персонажа. В правом верхнем углу отображается текущий уровень, на котором находится игрок. Блок твердых поверхностей отображаются как черные квадраты.

Кроме элементов интерфейса и твердых поверхностей в игре присутствуют динамичных объекты, про коллизии с которыми происходят определенные действия .

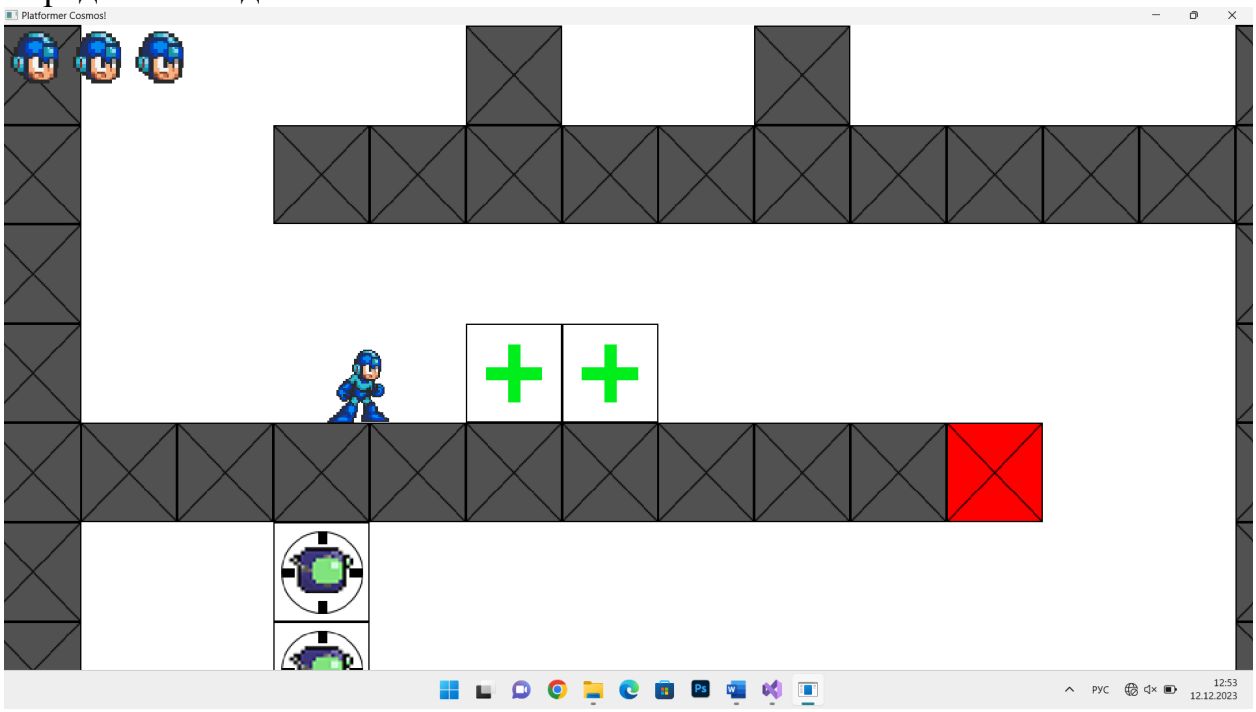

Рисунок 6.4 – уровень и динамичные элементы

Как можно наблюдать на рисунке 6.4, имеются блоки, восполняющие запас боеприпасов персонажа, блоки, наносящие персонажу урон, восполняющие здоровье блоки и блоки, закрывающие путь для дальнейшего прохождения уровня, которые исчезают, как только погибают привязанные к ним противники.

При этом у игрока имеется возможность стрельбы. В процессе стрельбы создаются снаряды игрока, которые двигаются в заданном направлении стрельбы до коллизии с каким-либо объектом. При коллизии с противником, снаряд уничтожается, а у противника отнимается одна из единиц здоровья. Про коллизии с другим объектом снаряд просто уничтожается.

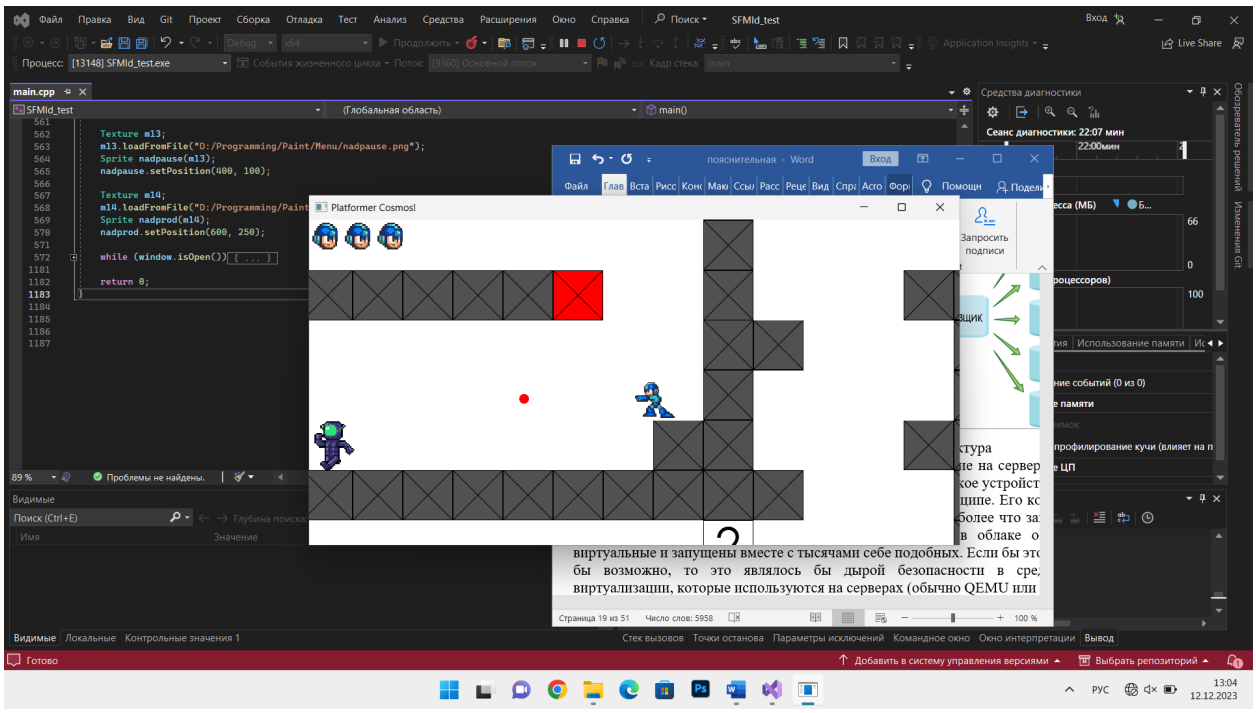

Рисунок 6.5 – возможность стрельбы

Для пользователя снаряд игрока отображается как красный шар.

При любом взаимодействии игрового персонажа с противником, а именно при их коллизии, игроку наносится урон в размере одной единицы здоровья. Это можно наблюдать на рисунке 6.6

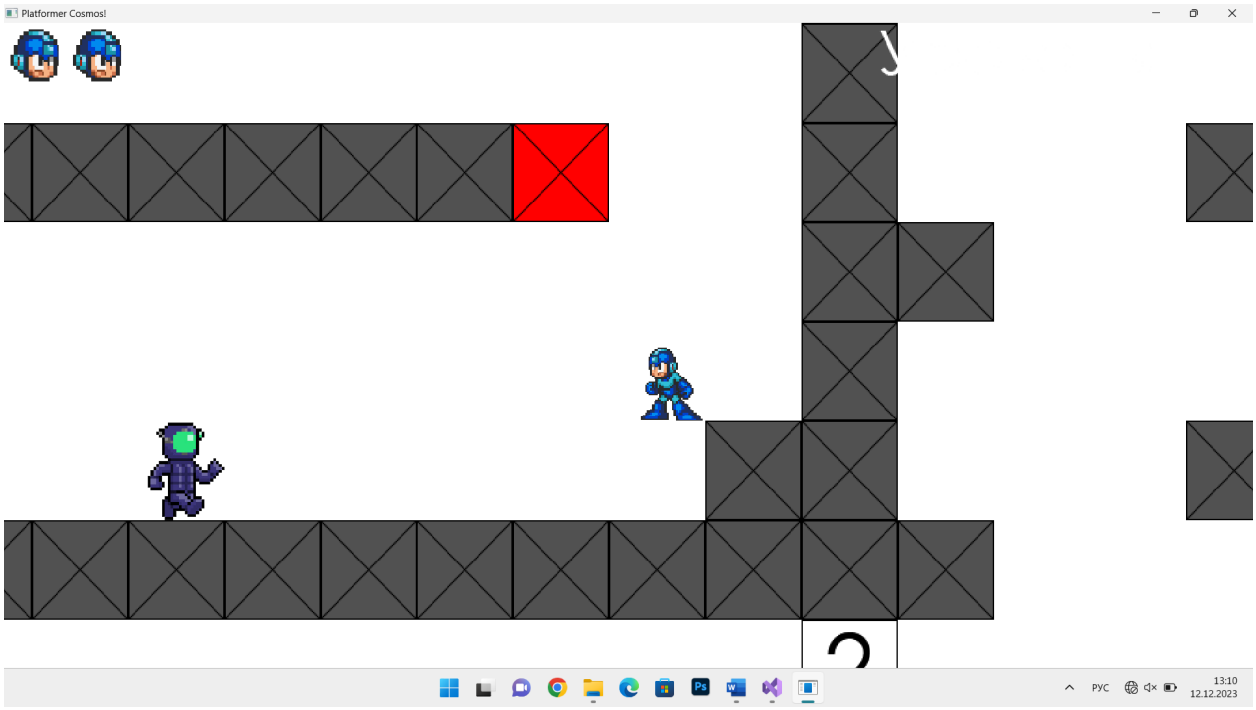

Рисунок 6.6 – получение игроком урона

Как видно, после взаимодействия с противником у игрока вместо трех единиц здоровья остается две. А сам игрок переносится в ближайшее безопасное место.

В случае смерти перед игроком открывается меню, в котором можно попробовать пройти уровень заново или же выйти в главное меню. Это можно наблюдать на рисунке 6.7

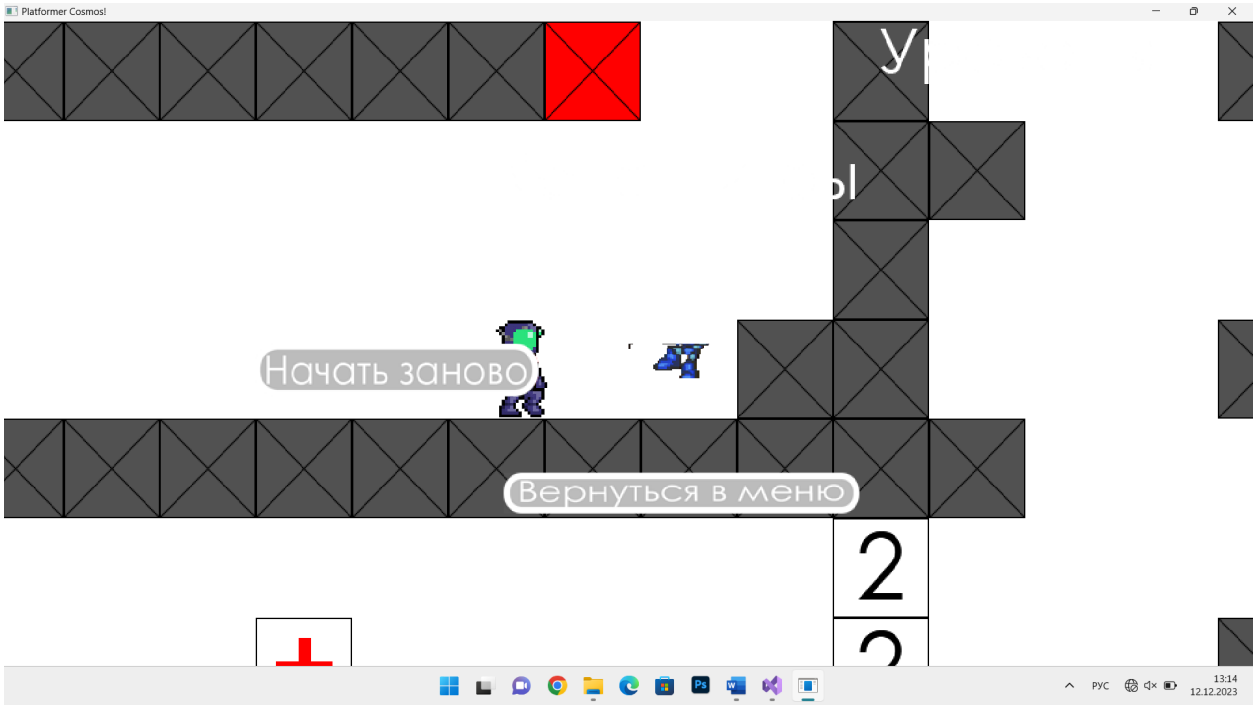

Рисунок 6.7 – меню, появляющееся в случае смерти игрока

Кроме обычных противников в игре присутствует разновидность врага, которую намного труднее победить. Это так называемый Босс. В отличие от обычных противников у Босса намного большее количество очков здоровья, которые видны пользователю, а также Босс способен открывать огонь по персонажу, что может привести к смерти игрока. Во всем вышесказанном можно убедиться благодаря рисунку 6.8

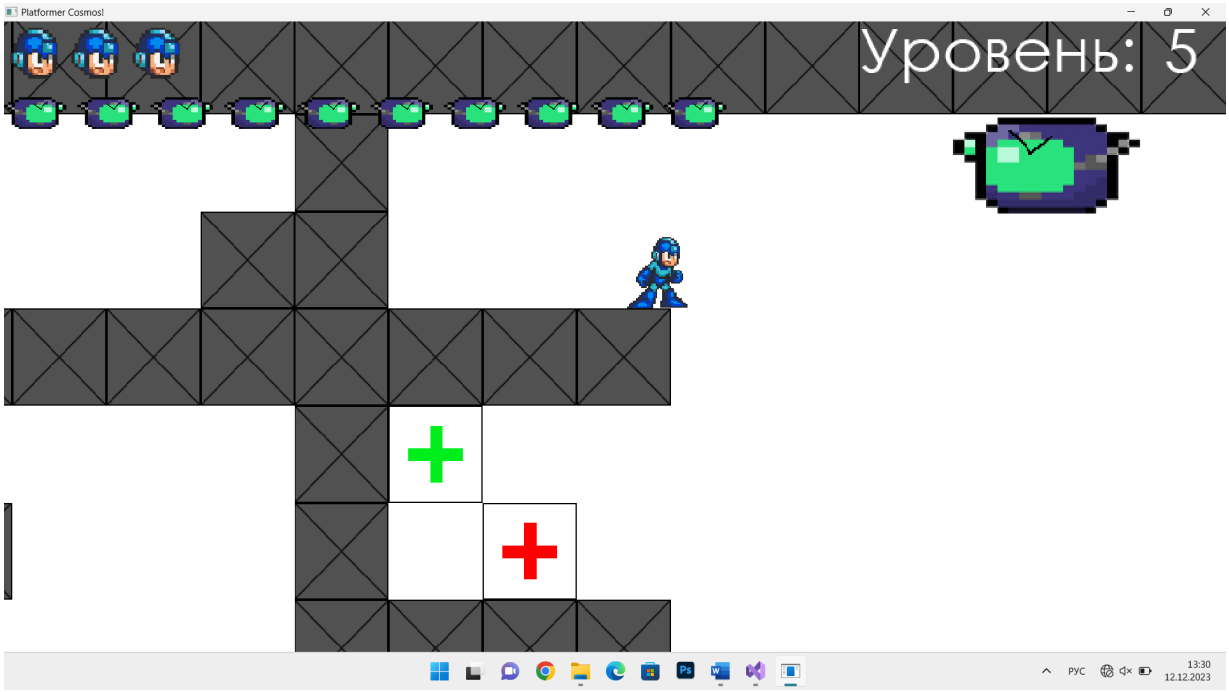

Рисунок 6.8 – противник Босс

Программа разрабатывалась и запускалась на OS Windows. Также проводились тесты на OS Linux.

Разработка производилась в интегрированной среде разработки Visual Studio с подключенной библиотекой SFML. Это позволяло комфортно, быстро и эффективно собирать и разрабатывать сложный проект. На рисунке 5.1 виден интерфейс интегрированной среды разработки Visual Studio. При нажатии кнопки локального отладчика Windows запускается разработанный проект.

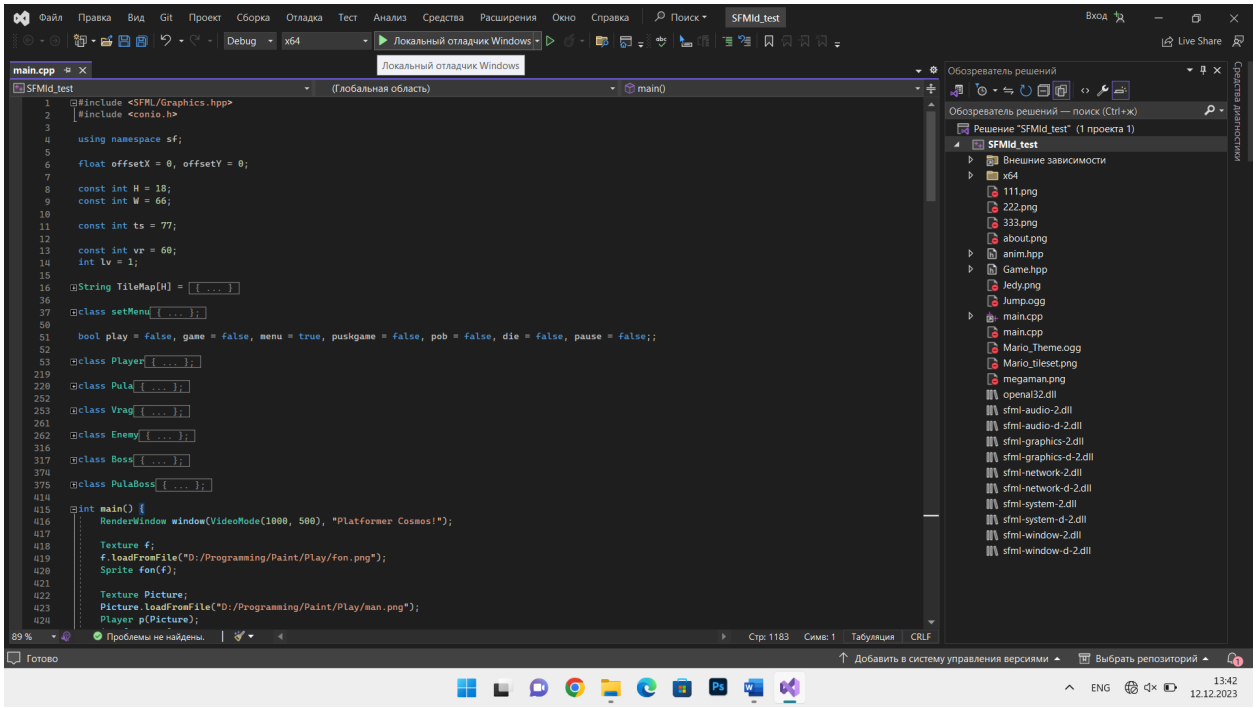

Рисунок 6.9 – интерфейс Visual Studio

Для корректной работы проекта необходимо установить библиотеку SFML. Ссылку на скачивание можно найти в Гугл Поиске. После загрузки проекта перед вами откроется меню с дальнейшими опциями. Если вы выбрали начали игру и открыли доступный уровень, то вам доступно управление персонажем с помощью клавиш клавиатуры. Чтобы двигаться влево или вправо, необходимо нажать клавишу левой или правой стрелки на клавиатуре соответственно. Для того, чтобы совершить прыжок, необходимо нажать на клавишу стрелки вверх. Для стрельбы необходимо нажать на клавишу Пробел. Чтобы поставить игру на паузу и открыть меню, необходимо нажать на клавишу Esc. Главной целью игры для вас является пройти все уровни, при этом одолев всех противников и босса. При победе над боссом игра считается завершенной, но вы можете вернуться на любой из уровней и пройти его заново.

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы была создана 2D игра-платформер. Для этого использовалась интегрированная среда разработки Visual Studio для разработки и запуска кода проекта, библиотека SFML для упрощенного создания игры, работы с графическими элементами и со временем.

В результате игра представляет собой полноценный платформер, со всеми необходимыми механиками. Реализовано построение уровней, добавление объектам любых текстур и анимаций. В игре присутствуют статические объеткы, объекты для восполнения единиц здоровья, игровое меню с возможностью выбора необходимых опций, противники, которые наносят урон игроку и которых игрок может уничтожить. Также реализован персонаж, который является центральным объектом игры и которым игрок может управлять. В дальнейшем проект можно только совершенствовать, усложняя уровни, добавляя новых противников, механики, анимации и так далее.

Игры-платформеры это целый жанр компьютерных игр, в которых основу игрового процесса составляют прыжки по платформам, лазанье по лестницам, сбор предметов, необходимых для победы над врагами или завершения уровня. Многие игры подобного жанра характеризуются нереалистичностью, рисованной мультяшной графикой..

Данная работа показала, что написать собственную игру-платформер не так сложно, как может показаться на первый взгляд. Самыми сложными частями проекта были реализация графической части, взаимодействие объектов друг с другом при коллизии и функция отслеживания состояния объектов в любой момент времени.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Объектно-ориентированное программирование на языке C++: учеб. пособие / Ю. А. Луцик, В. Н. Комличенко. – Минск : БГУИР, 2008.
2. The C++ Programming Language, – Bjarne Stroustrup, 1985
3. Habr[Электронный ресурс]. -Электронные данные. -Режим доступа: https://habr.com/ru/articles/703500/ - Дата доступа: 22.11.2023
4. Kuchka-pc[Электронный ресурс]. -Электронные данные. -Режим доступа: https://kychka-pc.ru/- Дата доступа: 22.11.2023
5. Dzen[Электронный ресурс]. -Электронные данные. -Режим доступа: https://dzen.ru/list/games/kak-napisat-2d-igru-na-s- Дата доступа: 22.11.2023

ПРИЛОЖЕНИЕ А
(обязательное)

Диаграмма классов

ПРИЛОЖЕНИЕ Б
(обязательное)

Схема метода int main()

ПРИЛОЖЕНИЕ В
(обязательное)

Схема метода Player::update()

ПРИЛОЖЕНИЕ Г
(обязательное)

Полный код программы

```
#include <SFML/Graphics.hpp>
#include <conio.h>

using namespace sf;

float offsetX = 0, offsetY = 0;

const int H = 18;
const int W = 66;

const int ts = 77;

const int vr = 60;
int lv = 1;

String TileMap[H] = {
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA",
"A            A                         A                         A",
"A            A                         A   A       A             A",
"AAAAAAAAAAA  A   A       AAA        A  A  AAAAAAAAAAAAAA      AAAA",
"A            A  AAAAAAAAAAAAAAAAAAAA  A           A            A",
"A            AA           A            AAAAAAAAAA   A            A",
"A    A  A    A            A            A            AAAA      AAAA",
"A  AAAAAAAAAAAAAAAAAAAA  A A         AA           AA            A",
"A            A           AA AAAAAAAAAAAA   A        A            A",
"A            A           A            A  AAAXXAAXXAAAAA      AAAA",
"AAAAAAAAAAX  A   AA A A A A            A           A            A",
"A            AA   A       A   X  X  X  AA          A            A",
"A            A    A       AAAAAAAAAAA  A       A    AAAA      AAAA",
"A  A        AA   AA       A            AAAAAAAAAA   A            A",
"A  AAAAAAAAAAA   A       A            A           A            A",
"A                A       A                       AAAAA      AAAA",
"A               AAXXXXXXXA         A    A         AAAAXXXXXXAAAA",
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA",
```

```
};

class setMenu {
public:
bool pusklever[5], blv[5];

setMenu() {
	for (int i = 0; i < 5; i++)
		pusklever[i] = false;

	blv[0] = true;
	for (int i = 1; i < 5; i++)
		blv[i] = false;
}
};

	bool play = false, game = false, menu = true, puskgame = false, pob = false,
die = false, pause = false;;

class Player {
public:
float dx, dy;
FloatRect rect;
bool onGround, rig;
Sprite sprite;
float curFrame;
int l;
bool life;

Player(Texture& image) {
	sprite.setTexture(image);
	rect = FloatRect(ts, ts * 2, 77 * 0.9, 65 * 0.9);
	sprite.setTextureRect(IntRect(0, 0, 77, 65));
	sprite.setScale(0.9, 0.9);

	dx = dy = 0;
	curFrame = 0;
```

```
		l = 3;
		life = true;
	}

	void update(float time) {
		rect.left += dx * time;
		Collision(0);

		if (!onGround)
			dy = dy + 0.0005 * time;

		rect.top += dy * time;
		onGround = false;
		Collision(1);

		if (life) {
			if (l <= 0)
				life = false;

			curFrame += 0.01 * time;
			if (curFrame > 8)
				curFrame -= 7;

			if (dx > 0) {
				if (dy == 0)
					sprite.setTextureRect(IntRect(77  *  int(curFrame),
0, 77, 65));
				else
					sprite.setTextureRect(IntRect(0, 81, 45, 65));

				rig = true;
			}
			if (dx < 0) {
				if (dy == 0)
					sprite.setTextureRect(IntRect(77 * int(curFrame) +
77, 0, -77, 65));
				else
```

```
                        sprite.setTextureRect(IntRect(45, 81, -45, 65));

                    rig = false;
                }
            }

            else {
                if (rig)
                    sprite.setTextureRect(IntRect(156, 104, 82, 30));
                else
                    sprite.setTextureRect(IntRect(238, 104, -82, 30));

                play = false;
                die = true;
            }

            if (pob)
                sprite.setTextureRect(IntRect(0, 0, 77, 65));

            sprite.setPosition(rect.left - offsetX, rect.top - offsetY);

            dx = 0;
        }

        void Collision(int dir) {
            for (int i = rect.top / ts; i < (rect.top + rect.height) / ts; i++)
                for (int j = rect.left / ts; j < (rect.left + rect.width) / ts;
j++) {
                    if (TileMap[i][j] == 'A' || TileMap[i][j] == 'K') {
                        if (dx > 0 && dir == 0)
                            rect.left = j * ts - rect.width;
                        if (dx < 0 && dir == 0)
                            rect.left = j * ts + ts;
                        if (dy > 0 && dir == 1) {
                            rect.top = i * ts - rect.height;
                            dy = 0;
                            onGround = true;
```

```
                        }
                        if (dy < 0 && dir == 1) {
                                rect.top = i * ts + ts;
                                dy = 0;
                        }
                }

                if (TileMap[i][j] == 'B')
                        TileMap[i][j] = ' ';

                if (TileMap[i][j] == 'C') {
                        TileMap[i][j] = ' ';

                        l++;
                }

                if (TileMap[i][j] == '2' || TileMap[i][j] == '3' ||
TileMap[i][j] == '4' || TileMap[i][j] == '5') {
                        rect.left += ts + ts;

                        for (int z = 0; z < H; z++)
                                TileMap[z][j] = 'A';

                        pob = true;
                        play = false;
                }

                if (TileMap[i][j] == 'X') {
                        if ((dx > 0 && dir == 0) || (dx < 0 && dir == 0) ||
                                (dy > 0 && dir == 1) || (dy < 0 && dir == 1))
{
                                l--;

                                if (l > 0) {
                                        if (i == 10 && j == 10) {
                                                if (dy < 0 && dir == 1) {
                                                        rect.left = 931.7;
```

```
                rect.top = 942.5;
        }

        else if (dy > 0 && dir == 1) {
                rect.left = 584.22;
                rect.top = 711.5;
        }
}

if (i == 16 && j >= 19 && j <= 25) {
        rect.left = 1362.86;
        rect.top = 711.5;
}

if (i == 11 && (j == 30 || j == 33 ||
j == 36)) {
        rect.left = 2173.41;
        rect.top = 865.5;
}

if (i == 9 && (j == 45 || j == 46 || j
== 49 || j == 50)) {
        if (dy < 0 && dir == 1) {
                rect.left = 3626.1;
                rect.top = 865.5;
        }

        else if (dy > 0 && dir == 1) {
                rect.left = 3311.82;
                rect.top = 557.5;
        }
}

if (i == 16 && j >= 56 && j <= 61) {
        rect.left = 4081;
        rect.top = 172.5;
        offsetY = 0;
```

```
                                                }
                                            }
                                        }
                                    }
                                }
        }
        };

        class Pula {
        public:
        float dx, dy;
        FloatRect rect;
        int go;
        Sprite sprite;

        Pula(Texture& image) {
              sprite.setTexture(image);
              dx = dy = 0;
              go = 0;
        }

        void update(float time) {
              rect.left += dx * time;
              Collision(0);

              rect.top += dy * time;
              Collision(1);

              sprite.setPosition(rect.left - offsetX, rect.top - offsetY);
        }

        void Collision(int dir) {
              for (int i = rect.top / ts; i < (rect.top + rect.height) / ts; i++)
                    for (int j = rect.left / ts; j < (rect.left + rect.width) / ts;
j++)
                          if (TileMap[i][j] == 'A' || TileMap[i][j] == 'K' ||
TileMap[i][j] == '2' ||
```

```
                                    TileMap[i][j] == '3' || TileMap[i][j] == '4' ||
TileMap[i][j] == '5')
                                    if ((dx > 0 && dir == 0) || (dx < 0 && dir == 0))
                                          go = 0;
      }
      };

      class Vrag {
      public:
      float dx, dy;
      FloatRect rect;
      Sprite sprite;
      float curFrame;
      bool life;
      };

      class Enemy: public Vrag {
      public:
      //float dx, dy;
      //FloatRect rect;
      //Sprite sprite;
      //float curFrame;
      //bool life;

      void set(Texture& image, int x, int y) {
            sprite.setTexture(image);
            sprite.setScale(0.38, 0.38);
            rect = FloatRect(x, y, ts, ts);

            dx = 0.1;
            curFrame = 0;
            life = false;
      }

      void update(float time) {
            if (!pause) {
                  rect.left += dx * time;
```

```
                    Collision();

                    curFrame += time * 0.005;
                    if (curFrame > 5)
                          curFrame -= 5;

                    if (dx > 0)
                          sprite.setTextureRect(IntRect(165  *  int(curFrame),  0,
165, 204));
                    else if (dx < 0)
                          sprite.setTextureRect(IntRect(165 * int(curFrame) + 165,
0, -165, 204));
                    if (!life)
                          sprite.setTextureRect(IntRect(0, 0, 0, 0));
             }

             sprite.setPosition(rect.left - offsetX, rect.top - offsetY);
      }

      void Collision() {
             for (int i = rect.top / ts; i < (rect.top + rect.height) / ts; i++)
                    for (int j = rect.left / ts; j < (rect.left + rect.width) / ts;
j++)
                          if (TileMap[i][j] == 'A') {
                                if (dx > 0) {
                                      rect.left = j * ts - rect.width;
                                      dx *= -1;
                                }
                                else if (dx < 0) {
                                      rect.left = j * ts + ts;
                                      dx *= -1;
                                }
                          }
      }
      };

      class Boss: public Vrag {
```

```
public:
//float dx, dy;
//FloatRect rect;
//Sprite sprite;
//float curFrame;
bool rig;
//bool life, rig;
int l;

void set(Texture& image, int x, int y) {
	sprite.setTexture(image);
	sprite.setScale(1.5, 1.5);
	rect = FloatRect(x, y, ts, ts);

	dy = 0.2;
	curFrame = 0;
	life = false;
	rig = true;
	l = 10;
}

void update(float time) {
	if (!pause) {
		rect.top += dy * time;

		Collision();
	}

	if (life && l == 0) {
		life = false;

		sprite.setTextureRect(IntRect(0, 0, 0, 0));

		pob = true;
		play = false;
	}
```

```
		sprite.setPosition(rect.left - offsetX, rect.top - offsetY);
	}

	void Collision() {
		for (int i = rect.top / ts; i < (rect.top + rect.height) / ts; i++)
			for (int j = rect.left / ts; j < (rect.left + rect.width) / ts;
j++)
				if (TileMap[i][j] == 'A')
				{
					if (dy > 0) {
						rect.top = i * ts - rect.height;
						dy *= -1;
					}
					else if (dy < 0) {
						rect.top = i * ts + ts;
						dy *= -1;
					}
				}
	}
	};

	class PulaBoss {
	public:
	float dx, dy;
	FloatRect rect;
	int go;
	Sprite sprite;

	void set(Texture& image, int x, int y) {
		sprite.setTexture(image);
		dx = dy = 0;
		go = 0;
	}

	void update(float time) {
		if (!pause) {
			rect.left += dx * time;
```

```
                Collision();

                rect.top += dy * time;
        }

        sprite.setPosition(rect.left - offsetX, rect.top - offsetY);
    }

    void Collision() {
        for (int i = rect.top / ts; i < (rect.top + rect.height) / ts; i++)
            for (int j = rect.left / ts; j < (rect.left + rect.width) / ts;
j++)
                if (TileMap[i][j] == 'A') {
                    if (dx > 0) {
                        rect.left = j * ts - rect.width;
                        go = 0;
                    }
                    else if (dx < 0) {
                        rect.left = j * ts + ts;
                        go = 0;
                    }
                }
    }
    };

    int main() {
    RenderWindow window(VideoMode(1000, 500), "Platformer Cosmos!");

    Texture f;
    f.loadFromFile("D:/Programming/Paint/Play/fon.png");
    Sprite fon(f);

    Texture Picture;
    Picture.loadFromFile("D:/Programming/Paint/Play/man.png");
    Player p(Picture);
    int frame = 0;
```

```
Clock clock;

Texture t2;
t2.loadFromFile("D:/Programming/Paint/Play/plat.png");
Sprite plat(t2);

Texture t1;
t1.loadFromFile("D:/Programming/Paint/Play/pula.png");
Pula pu(t1);

Texture t3;
t3.loadFromFile("D:/Programming/Paint/Play/enemy.png");
Enemy enemy[vr];
enemy[0].set(t3, 4 * ts, 13 * ts);
for (int i = 1; i < 4; i++)
      enemy[i].set(t3, (18 + (i - 1)) * ts, 3 * ts);
for (int i = 4; i < 9; i++)
      enemy[i].set(t3, (28 + (i - 4)) * ts, 3 * ts);
for (int i = 9; i < 16; i++)
      enemy[i].set(t3, (29 + (i - 9)) * ts, 7 * ts);
for (int i = 16; i < 26; i++)
      enemy[i].set(t3, (27 + (i - 16)) * ts, 16 * ts);
for (int i = 26; i < 36; i++)
      enemy[i].set(t3, (42 + (i - 26)) * ts, 16 * ts);
for (int i = 36; i < 44; i++)
      enemy[i].set(t3, (40 + (i - 36)) * ts, 12 * ts);
for (int i = 44; i < 53; i++)
      enemy[i].set(t3, (44 + (i - 44)) * ts, 8 * ts);
for (int i = 53; i < 60; i++)
      enemy[i].set(t3, (44 + (i - 53)) * ts, 2 * ts);

Texture t4;
t4.loadFromFile("D:/Programming/Paint/Play/Boss.png");
Boss boss;
boss.set(t4, 59 * ts, ts);

Texture t5;
```

```
t5.loadFromFile("D:/Programming/Paint/Play/pulaBoss.png");
PulaBoss puBos;
puBos.set(t5, 59 * ts, ts);

Sprite lifeBoss[10];
for (int i = 0; i < 10; i++) {
	lifeBoss[i].setTexture(t4);
	lifeBoss[i].setScale(0.5, 0.5);
	lifeBoss[i].setPosition(60 * i, 60);
}

Texture nl;
nl.loadFromFile("D:/Programming/Paint/Play/nadlev.png");
Sprite nadlev(nl);
nadlev.setPosition(700, 0);

Texture sc;
sc.loadFromFile("D:/Programming/Paint/Play/score.png");
Sprite score(sc);
score.setTextureRect(IntRect(0, 0, 27, 36));
score.setPosition(950, 5);

Texture tl;
tl.loadFromFile("D:/Programming/Paint/Play/life.png");
Sprite life[12];
for (int i = 0; i < 12; i++) {
	life[i].setTexture(tl);
	life[i].setScale(1.5, 1.5);
	life[i].setPosition(5 + 50 * i, 5);
}

setMenu sm;

Texture m1;
m1.loadFromFile("D:/Programming/Paint/Menu/naz.png");
Sprite naz(m1);
naz.setPosition(200, 10);
```

```
Texture m2;
m2.loadFromFile("D:/Programming/Paint/Menu/butplay.png");
Sprite butplay(m2);
butplay.setPosition(200, 235);

Texture m3;
m3.loadFromFile("D:/Programming/Paint/Menu/exit.png");
Sprite exit(m3);
exit.setPosition(600, 235);

Texture m4;
m4.loadFromFile("D:/Programming/Paint/Menu/nazurav.png");
Sprite nazurav(m4);
nazurav.setPosition(350, 10);

Texture m5;
m5.loadFromFile("D:/Programming/Paint/Menu/butlev.png");
Sprite butlev[5];
for (int i = 0; i < 5; i++) {
	butlev[i].setTexture(m5);
	butlev[i].setPosition(150 + 150 * i, 230);
	butlev[i].setTextureRect(IntRect(0, 0, 80, 80));
}
butlev[0].setTextureRect(IntRect(0, 80, 80, 80));

Texture m6;
m6.loadFromFile("D:/Programming/Paint/Menu/back.png");
Sprite back(m6);
back.setPosition(150, 400);

Texture m7;
m7.loadFromFile("D:/Programming/Paint/Menu/win.png");
Sprite win(m7);
win.setPosition(300, 100);

Texture m8;
```

```
m8.loadFromFile("D:/Programming/Paint/Menu/urpro.png");
Sprite urpro(m8);
urpro.setPosition(300, 100);

Texture m9;
m9.loadFromFile("D:/Programming/Paint/Menu/sledur.png");
Sprite sledur(m9);
sledur.setPosition(500, 250);

Texture m10;
m10.loadFromFile("D:/Programming/Paint/Menu/naczan.png");
Sprite naczan(m10);
naczan.setPosition(200, 250);

Texture m11;
m11.loadFromFile("D:/Programming/Paint/Menu/butmen.png");
Sprite butmen(m11);
butmen.setPosition(400, 350);

Texture m12;
m12.loadFromFile("D:/Programming/Paint/Menu/ki.png");
Sprite ki(m12);
ki.setPosition(400, 100);

Texture m13;
m13.loadFromFile("D:/Programming/Paint/Menu/nadpause.png");
Sprite nadpause(m13);
nadpause.setPosition(400, 100);

Texture m14;
m14.loadFromFile("D:/Programming/Paint/Menu/nadprod.png");
Sprite nadprod(m14);
nadprod.setPosition(600, 250);

while (window.isOpen())
{
	float time = clock.getElapsedTime().asMicroseconds();
```

```
            clock.restart();
            time = time / 800;

            Event event;
            while (window.pollEvent(event))
            {
                  if (event.type == Event::Closed)
                        window.close();

                  if (event.type == Event::MouseButtonPressed)
                        if (event.key.code == Mouse::Left) {
                              Vector2i pos = Mouse::getPosition(window);

                              if (menu) {
                                    if   (exit.getGlobalBounds().contains(pos.x,
pos.y))
                                          window.close();

                                    if
(butplay.getGlobalBounds().contains(pos.x, pos.y)) {
                                          menu = false;
                                          game = true;
                                    }
                              }

                              else if (game) {
                                    if   (back.getGlobalBounds().contains(pos.x,
pos.y)) {
                                          game = false;
                                          menu = true;
                                    }

                                    for (int i = 0; i < 5; i++)
                                          if
(butlev[i].getGlobalBounds().contains(pos.x, pos.y) && sm.blv[i]) {
                                                puskgame = true;
                                                sm.pusklever[i] = true;
                                          }
```

```
                                }

                                else if (pob || die || pause) {
                                      if (naczan.getGlobalBounds().contains(pos.x,
pos.y)) {
                                            pob = false;
                                            puskgame = true;
                                            pause = false;
                                      }

                                      if (sledur.getGlobalBounds().contains(pos.x,
pos.y) && lv < 5 && !die && !pause) {
                                            pob = false;
                                            puskgame = true;
                                            die = false;

                                            sm.pusklever[lv - 1] = false;
                                            sm.pusklever[lv] = true;
                                      }

                                      if
(nadprod.getGlobalBounds().contains(pos.x, pos.y) && pause)
                                            pause = false;

                                      if (butmen.getGlobalBounds().contains(pos.x,
pos.y)) {
                                            pob = false;
                                            menu = true;
                                            play = false;
                                            die = false;
                                            pause = false;

                                            p.l = 3;
                                            p.life = true;

                                            p.sprite.setTextureRect(IntRect(0,  0,
77, 65));

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

```
                                        sm.pusklever[i] = false;

                                    offsetX = 0;
                                    offsetY = 0;

                                    p.rect.left = 77;
                                    p.rect.top = 172.5;
                                }
                            }
                        }
                }

//std::cout << enemy[0].dx << "\n";

if (pob)
      sm.blv[lv] = true;

if (sm.blv[1])
      butlev[1].setTextureRect(IntRect(80, 80, 80, 80));
if (sm.blv[2])
      butlev[2].setTextureRect(IntRect(80 * 2, 80, 80, 80));
if (sm.blv[3])
      butlev[3].setTextureRect(IntRect(80 * 3, 80, 80, 80));
if (sm.blv[4])
      butlev[4].setTextureRect(IntRect(0, 80 * 2, 80, 80));

if (puskgame) {
      puskgame = false;
      game = false;
      pob = false;
      die = false;

      if (sm.pusklever[0]) {
            lv = 1;

            offsetX = 0;
            offsetY = 0;
```

```
		p.rect.left = 77;
		p.rect.top = 172.5;

		enemy[0].life = true;

		for (int i = 4; i < 7; i++)
			TileMap[9][i] = 'B';
		TileMap[16][7] = 'C';
		for (int i = 15; i < 17; i++)
			TileMap[i][13] = '2';
	}

	if (sm.pusklever[1]) {
		lv = 2;

		p.rect.left = 1098.39;
		p.rect.top = 1250.5;

		offsetX = p.rect.left;
		offsetY = p.rect.top - 365;

		for (int i = 1; i < 4; i++)
			enemy[i].life = true;

		for (int i = 15; i < 17; i++)
			TileMap[i][13] = 'A';
		TileMap[9][14] = 'B';
		for (int i = 17; i < 19; i++)
			TileMap[9][i] = 'B';
		TileMap[9][22] = 'B';
		for (int i = 19; i < 22; i++)
			TileMap[6][i] = 'C';
		TileMap[3][14] = 'B';
		for (int i = 1; i < 3; i++)
			TileMap[i][26] = '3';
	}
```

```
if (sm.pusklever[2]) {
	lv = 3;

	p.rect.left = 2079;
	p.rect.top = 172.5;

	offsetX = p.rect.left + 500;
	offsetY = 0;

	for (int i = 4; i < 9; i++)
		enemy[i].life = true;
	for (int i = 9; i < 16; i++)
		enemy[i].life = true;
	for (int i = 16; i < 26; i++)
		enemy[i].life = true;

	for (int i = 1; i < 3; i++)
		TileMap[i][26] = 'A';
	for (int i = 28; i < 30; i++)
		TileMap[2][i] = 'B';
	for (int i = 31; i < 33; i++)
		TileMap[2][i] = 'B';
	for (int i = 34; i < 36; i++)
		TileMap[2][i] = 'B';
	for (int i = 29; i < 32; i++)
		TileMap[6][i] = 'B';
	for (int i = 33; i < 36; i++)
		TileMap[6][i] = 'B';
	for (int i = 27; i < 29; i++)
		TileMap[11][i] = 'C';
	for (int i = 31; i < 33; i++)
		TileMap[11][i] = 'B';
	for (int i = 34; i < 36; i++)
		TileMap[11][i] = 'C';
	for (int i = 27; i < 31; i++)
		TileMap[14][i] = 'B';
```

```
	for (int i = 32; i < 36; i++)
		TileMap[14][i] = 'B';
	for (int i = 15; i < 17; i++)
		TileMap[i][39] = '4';
}

if (sm.pusklever[3]) {
	lv = 4;

	p.rect.left = 3080;
	p.rect.top = 1250.5;

	offsetX = p.rect.left;
	offsetY = p.rect.top - 365;

	for (int i = 26; i < 36; i++)
		enemy[i].life = true;
	for (int i = 36; i < 44; i++)
		enemy[i].life = true;
	for (int i = 44; i < 53; i++)
		enemy[i].life = true;
	for (int i = 53; i < 60; i++)
		enemy[i].life = true;

	for (int i = 15; i < 17; i++)
		TileMap[i][39] = 'A';
	TileMap[15][41] = 'C';
	for (int i = 42; i < 46; i++)
		TileMap[11][i] = 'B';
	TileMap[8][47] = 'C';
	TileMap[8][48] = 'B';
	for (int i = 43; i < 46; i++)
		TileMap[4][i] = 'B';
	for (int i = 1; i < 3; i++)
		TileMap[i][52] = '5';
}
```

```
                if (sm.pusklever[4]) {
                        lv = 5;

                        boss.life = true;
                        boss.l = 10;

                        p.rect.left = 4112.69;
                        p.rect.top = 172.5;

                        for (int i = 1; i < 3; i++)
                                TileMap[i][52] = 'A';
                        for (int i = 0; i < 4; i++) {
                                TileMap[4 + 3 * i][53] = 'B';
                                TileMap[5 + 3 * i][54] = 'C';
                        }
                        for (int i = 0; i < 5; i++) {
                                TileMap[2 + 3 * i][63] = 'C';
                                TileMap[1 + 3 * i][64] = 'B';
                        }
                }

                p.l = 3;
                p.life = true;

                play = true;
        }

        if (p.life && play && !pause) {
                if (!_kbhit())
                        if (p.dy == 0) {
                                if (p.rig)
                                        p.sprite.setTextureRect(IntRect(0,   0,   77,
65));
                                else if (!p.rig)
                                        p.sprite.setTextureRect(IntRect(77,  0,  -77,
65));
                        }
```

```
                if (Keyboard::isKeyPressed(Keyboard::Left))
                        p.dx = -0.3;
                if (Keyboard::isKeyPressed(Keyboard::Right))
                        p.dx = 0.3;
                if (Keyboard::isKeyPressed(Keyboard::Up))
                        if (p.onGround) {
                                p.dy = -0.5;

                                if (p.rig)
                                        p.sprite.setTextureRect(IntRect(0,  81,  45,
65));
                                else if (!p.rig)
                                        p.sprite.setTextureRect(IntRect(45, 81, -45,
65));

                                p.onGround = false;
                        }
                if (Keyboard::isKeyPressed(Keyboard::Space))
                        if (p.onGround && pu.go == 0) {
                                frame = 100;

                                if (p.rig) {
                                        pu.go = 1;

                                        pu.rect.left = p.rect.left + 67;
                                        pu.rect.top = p.rect.top + 20;
                                }
                                else {
                                        pu.go = 2;

                                        pu.rect.left = p.rect.left - 12;
                                        pu.rect.top = p.rect.top + 18;
                                }
                        }

                if (Keyboard::isKeyPressed(Keyboard::Escape))
```

```
                pause = true;
        }

        if (pu.go == 0) {
              pu.rect.left = p.rect.left;
              pu.rect.top = p.rect.top;
        }
        else if (pu.go == 1)
              pu.dx = 2;
        else if (pu.go == 2)
              pu.dx = -2;

        if (frame != 0)
              if (p.dy == 0) {
                    if (p.rig)
                          p.sprite.setTextureRect(IntRect(68, 80, 69, 61));
                    else if (!p.rig)
                          p.sprite.setTextureRect(IntRect(68 + 69, 80, -69,
61));

                    frame--;
              }

        for (int i = 0; i < vr; i++)
              if (enemy[i].life) {
                    if (p.rect.intersects(enemy[i].rect)) {
                          p.l--;

                          if (p.l > 0) {
                                if (i == 0) {
                                      p.rect.left = 931.7;
                                      p.rect.top = 942.5;
                                }
                                else if (i >= 1 && i < 4) {
                                      p.rect.left = 1312.52;
                                      p.rect.top = 172.5;
                                }
```

```
                                else if (i >= 4 && i < 9) {
                                        p.rect.left = 2093.68;
                                        p.rect.top = 172.5;
                                }
                                else if (i >= 9 && i < 16) {
                                        p.rect.left = 2933.7;
                                        p.rect.top = 480.5;
                                }
                                else if (i >= 16 && i < 26) {
                                        p.rect.left = 2775.62;
                                        p.rect.top = 1173.5;
                                }
                                else if (i >= 26 && i < 36) {
                                        p.rect.left = 3176.79;
                                        p.rect.top = 1173.5;
                                }
                                else if (i >= 36 && i < 44) {
                                        p.rect.left = 3615.72;
                                        p.rect.top = 865.5;
                                }
                                else if (i >= 44 && i < 53) {
                                        p.rect.left = 3330.41;
                                        p.rect.top = 557.5;
                                }
                                else if (i >= 53 && i < 60) {
                                        p.rect.left = 3309.64;
                                        p.rect.top = 95.5;
                                }
                        }
                }

                        if      (enemy[i].rect.left     <       pu.rect.left      &&
enemy[i].rect.left + 10 > pu.rect.left &&
                                enemy[i].rect.top       <       pu.rect.top       &&
enemy[i].rect.top + 50 > pu.rect.top && pu.go != 0) {
                                enemy[i].life = false;
```

```
                        pu.go = 0;
                    }
                }

            if (!enemy[0].life) {
                TileMap[11][3] = ' ';
                TileMap[12][3] = ' ';
            }
            else if (enemy[0].life) {
                TileMap[11][3] = 'K';
                TileMap[12][3] = 'K';
            }
            if (!enemy[1].life && !enemy[2].life && !enemy[3].life) {
                TileMap[1][25] = ' ';
                TileMap[2][25] = ' ';
            }
            else if (enemy[1].life && enemy[2].life && enemy[3].life) {
                TileMap[1][25] = 'K';
                TileMap[2][25] = 'K';
            }
            if (!enemy[4].life && !enemy[5].life && !enemy[6].life &&
!enemy[7].life && !enemy[8].life) {
                TileMap[1][36] = ' ';
                TileMap[2][36] = ' ';
            }
            else if (enemy[4].life && enemy[5].life && enemy[6].life &&
enemy[7].life && enemy[8].life) {
                TileMap[1][36] = 'K';
                TileMap[2][36] = 'K';
            }
            if (!enemy[9].life && !enemy[10].life && !enemy[10].life &&
!enemy[11].life && !enemy[12].life &&
                !enemy[13].life && !enemy[14].life && !enemy[15].life) {
                TileMap[5][28] = ' ';
                TileMap[6][28] = ' ';
            }
            else if (enemy[9].life && enemy[10].life && enemy[10].life &&
enemy[11].life && enemy[12].life &&
```

```
                enemy[13].life && enemy[14].life && enemy[15].life) {
                TileMap[5][28] = 'K';
                TileMap[6][28] = 'K';
            }
            if (!enemy[16].life && !enemy[17].life && !enemy[18].life &&
!enemy[19].life && !enemy[20].life &&
                !enemy[21].life && !enemy[22].life && !enemy[23].life &&
!enemy[24].life && !enemy[25].life) {
                TileMap[15][38] = ' ';
                TileMap[16][38] = ' ';
            }
            else if (enemy[16].life && enemy[17].life && enemy[18].life &&
enemy[19].life && enemy[20].life &&
                enemy[21].life && enemy[22].life && enemy[23].life &&
enemy[24].life && enemy[25].life) {
                TileMap[15][38] = 'K';
                TileMap[16][38] = 'K';
            }
            if (!enemy[26].life && !enemy[27].life && !enemy[28].life &&
!enemy[29].life && !enemy[30].life &&
                !enemy[31].life && !enemy[32].life && !enemy[33].life &&
!enemy[34].life && !enemy[35].life) {
                TileMap[13][49] = ' ';
                TileMap[13][50] = ' ';
                TileMap[13][51] = ' ';
            }
            else if (enemy[26].life && enemy[27].life && enemy[28].life &&
enemy[29].life && enemy[30].life &&
                enemy[31].life && enemy[32].life && enemy[33].life &&
enemy[34].life && enemy[35].life) {
                TileMap[13][49] = 'K';
                TileMap[13][50] = 'K';
                TileMap[13][51] = 'K';
            }
            if (!enemy[36].life && !enemy[37].life && !enemy[38].life &&
!enemy[39].life && !enemy[40].life &&
                !enemy[41].life && !enemy[42].life && !enemy[43].life) {
                TileMap[9][40] = ' ';
                TileMap[9][41] = ' ';
            }
```

```
            else if (enemy[36].life && enemy[37].life && enemy[38].life &&
enemy[39].life && enemy[40].life &&
                  enemy[41].life && enemy[42].life && enemy[43].life) {
                  TileMap[9][40] = 'K';
                  TileMap[9][41] = 'K';
            }
            if (!enemy[44].life && !enemy[45].life && !enemy[46].life &&
!enemy[47].life &&
                  !enemy[48].life && !enemy[49].life && !enemy[50].life) {
                  TileMap[5][49] = ' ';
                  TileMap[5][50] = ' ';
                  TileMap[5][51] = ' ';
            }
            else if (enemy[44].life && enemy[45].life && enemy[46].life &&
enemy[47].life &&
                  enemy[48].life && enemy[49].life && enemy[50].life) {
                  TileMap[5][49] = 'K';
                  TileMap[5][50] = 'K';
                  TileMap[5][51] = 'K';
            }
            if (!enemy[53].life && !enemy[54].life && !enemy[55].life &&
!enemy[56].life && !enemy[57].life &&
                  !enemy[58].life && !enemy[59].life)
                  TileMap[1][51] = ' ';
            else if (enemy[53].life && enemy[54].life && enemy[55].life &&
enemy[56].life && enemy[57].life &&
                  enemy[58].life && enemy[59].life)
                  TileMap[1][51] = 'K';


            score.setTextureRect(IntRect(27 * (lv - 1), 0, 27, 36));


            if (boss.life) {
                  if (boss.rect.left < p.rect.left) {
                        boss.sprite.setTextureRect(IntRect(0, 0, 102, 51));
                        boss.rig = true;
                  }
                  else if (boss.rect.left > p.rect.left) {
                        boss.sprite.setTextureRect(IntRect(102, 0, -102, 51));
                        boss.rig = false;
```

```
		}

		if (boss.rect.top > p.rect.top && boss.rect.top < p.rect.top +
p.rect.height &&
			puBos.go == 0 && lv == 5) {
			if (boss.rig)
				puBos.go = 1;
			else
				puBos.go = 2;
		}

		if (puBos.go == 0) {
			if (boss.rig) {
				puBos.rect.left = boss.rect.left + 153;
				puBos.rect.top = boss.rect.top;
			}
			else if (!boss.rig) {
				puBos.rect.left = boss.rect.left;
				puBos.rect.top = boss.rect.top;
			}
		}
		else if (puBos.go == 1)
			puBos.dx = 1;
		else if (puBos.go == 2)
			puBos.dx = -1;

		if (boss.rect.left < pu.rect.left && boss.rect.left + 153 >
pu.rect.left && boss.life &&
			boss.rect.top < pu.rect.top && boss.rect.top + 77 >
pu.rect.top && pu.go != 0) {
			boss.l--;

			pu.go = 0;
		}

		if (p.rect.intersects(boss.rect)) {
			p.l--;
```

```
                        if (p.l > 0) {
                              p.rect.left = 4081;
                              p.rect.top = 172.5;
                              offsetY = 0;
                        }
                  }

                  if (p.rect.left < puBos.rect.left && p.rect.left + 77 > puBos.rect.left && p.life &&
                        p.rect.top < puBos.rect.top && p.rect.top + 65 > puBos.rect.top && puBos.go != 0) {
                        p.l--;

                        p.rect.left = 4081;
                        p.rect.top = 172.5;
                        offsetY = 0;
                  }
            }

            if (p.rect.left > 500 && p.rect.left < 4582)
                  offsetX = p.rect.left - 500;
            if (p.rect.top > 250 && p.rect.top < 1135)
                  offsetY = p.rect.top - 250;

            p.update(time);
            pu.update(time);
            boss.update(time);
            puBos.update(time);
            window.clear(Color::White);

            fon.setPosition(-offsetX, -offsetY);
            window.draw(fon);

            for (int i = 0; i < H; i++)
                  for (int j = 0; j < W; j++) {
                        if (TileMap[i][j] == 'A')
                              plat.setTextureRect(IntRect(0, 0, ts, ts));
```

```
            if (TileMap[i][j] == 'B')
                  plat.setTextureRect(IntRect(ts, 0, ts, ts));
            if (TileMap[i][j] == 'C')
                  plat.setTextureRect(IntRect(ts * 2, 0, ts, ts));
            if (TileMap[i][j] == 'X')
                  plat.setTextureRect(IntRect(ts * 3, 0, ts, ts));
            if (TileMap[i][j] == 'K')
                  plat.setTextureRect(IntRect(ts * 4, 0, ts, ts));
            if (TileMap[i][j] == '2')
                  plat.setTextureRect(IntRect(0, ts, ts, ts));
            if (TileMap[i][j] == '3')
                  plat.setTextureRect(IntRect(ts, ts, ts, ts));
            if (TileMap[i][j] == '4')
                  plat.setTextureRect(IntRect(ts * 2, ts, ts, ts));
            if (TileMap[i][j] == '5')
                  plat.setTextureRect(IntRect(ts * 3, ts, ts, ts));
            if (TileMap[i][j] == ' ')
                  continue;

            plat.setPosition(j * ts - offsetX, i * ts - offsetY);
            window.draw(plat);
      }

window.draw(p.sprite);
if (pu.go != 0)
      window.draw(pu.sprite);
for (int i = 0; i < vr; i++)
      enemy[i].update(time);
for (int i = 0; i < vr; i++)
      window.draw(enemy[i].sprite);
window.draw(boss.sprite);
if (puBos.go != 0)
      window.draw(puBos.sprite);
for (int i = 0; i < boss.l; i++)
      if (lv == 5)
            window.draw(lifeBoss[i]);
```

```
if (play || pob || die || pause) {
	window.draw(nadlev);
	window.draw(score);
	for (int i = 0; i < p.l; i++)
		window.draw(life[i]);

	if (pob || die || pause) {
		if (pob && !die && !pause) {
			if (lv < 5)
				window.draw(urpro);
			else
				window.draw(win);
		}
		else if (!pob && die && !pause)
			window.draw(ki);
		else if (!pob && !die && pause)
			window.draw(nadpause);

		if (lv < 5 && !die && !pause)
			window.draw(sledur);
		else if (pause)
			window.draw(nadprod);

		window.draw(naczan);
		window.draw(butmen);
	}
}

if (menu) {
	window.draw(naz);
	window.draw(butplay);
	window.draw(exit);
}

if (game) {
	window.draw(nazurav);
	for (int i = 0; i < 5; i++)
```

```
                    window.draw(butlev[i]);
                window.draw(back);
        }

        window.display();
}

return 0;
}
```

ПРИЛОЖЕНИЕ Д

(обязательное)

Ведомость документов