Федеральное государственное бюджетное

образовательное учреждение высшего образования

«Саратовский государственный технический университет

имени Гагарина Ю.А.»

Институт прикладных информационных

технологий и коммуникаций

Кафедра Информационная безопасность

автоматизированных систем

Направление подготовки 10.03.01 Информационная безопасность

Расчётно-графическая работа

по дисциплине «Языки программирования»

**«Игра Digger»**

Выполнил: студент 1 курса

Учебной группы с-ИБС12

Очной формы обучения

Никонов С. Н.

Проверил: ассистент каф. ИБС

Романчук С. П.

Саратов 2020

**Аннотация**

**Краткое описание работы**

**Игра Digger:** на прямоугольной карте, представляющей собой «шахту» перемещается диггер. Он может как двигаться по уже существующим ходам, так и прокапывать новые. Цель игры – собрать все разбросанные по карте сокровища. Игроку активно мешают монстры, которые также перемещаются по карте. Столкновение с монстром означает «гибель» игрока. Монстры могут преследовать игрока.

**Возможные расширения:** несколько уровней, бонусы с различным эффектом, разные по силе и характеристикам противники, возможность «отбиваться» от монстров.

**Оглавление**

[**Введение 3**](#_Toc42100030)

[Тема 3](#_Toc42100031)

[Цель работы 3](#_Toc42100032)

[Задачи 3](#_Toc42100033)

[Описание приложения 3](#_Toc42100034)

[**Теоретическая часть 4**](#_Toc42100035)

[C++ 4](#_Toc42100036)

[SFML 5](#_Toc42100037)

[**Практическая часть 6**](#_Toc42100038)

[Библиотеки 6](#_Toc42100039)

[Классы 6](#_Toc42100040)

[Функции 7](#_Toc42100041)

[**Заключение 9**](#_Toc42100042)

[**Код программы 10**](#_Toc42100043)

[**Литература 18**](#_Toc42100044)

# Введение

# ****Тема****

«Игра Digger»

# ****Цель работы****

Разработать игру Digger, используя полученные в ходе обучения знания о языке C++, в частности об объектно-ориентированном программировании.

# ****Задачи****

1. Ознакомиться с приложенной к работе статьёй об игре Digger и сформировать своё представление о будущем проекте.
2. Разработать игру Digger, опираясь на техническое задание и сформированное представление.
3. Проверить и, в случае необходимости, подкорректировать код, исправляя ошибки, убирая странные и нелогичные конструкции, и улучшая дизайн приложения и удобство работы с ним.

# Описание приложения

Приложение предусматривает меню с выбором нескольких уровней. У пользователя есть возможность начать уровень с начала, или переключиться на другой. Так же он может в любой момент выйти из игры.

Интерфейс во время прохождения показывает управление, цель игры и прогресс выполнения цели.

Карта – это пространство, ограждённое по периметру неразрушаемыми стенами. Оно заполняется тремя видами блоков: камнем, который игрок способен разрушить, золотой рудой, которую игроку необходимо добыть, и коренной породой, которую игрок не может преодолеть. Отдельные участки пространства могут оставаться пустыми.

На карту помещается персонаж игрока. Он может как двигаться по уже существующим ходам, так и прокапывать новые. Игроку активно мешают монстры, которые также заранее помещаются в разные части карты и перемещаются по пустому пространству. Столкновение с монстром означает смерть. Некоторые виды монстров могут преследовать игрока.

Игра заканчивается победой, если игрок собирает всю золотую руду, или поражением, если игрок погибает от рук монстров.

# Теоретическая часть

C++ — компилируемый, статически типизированный язык программирования общего назначения.

Поддерживает такие парадигмы программирования, как процедурное программирование, объектно-ориентированное программирование, обобщённое программирование. Язык имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности и другие возможности. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков. В сравнении с его предшественником — языком  
C, — наибольшее внимание уделено поддержке объектно-ориентированного и обобщённого программирования.

Стандарт C++ состоит из двух основных частей: описание ядра языка и описание стандартной библиотеки.

Первое время язык развивался вне формальных рамок, спонтанно, по мере встававших перед ним задач. Развитию языка сопутствовало развитие кросс-компилятора [cfront](https://ru.wikipedia.org/w/index.php?title=Cfront&action=edit&redlink=1). Новшества в языке отражались в изменении номера версии кросс-компилятора. Эти номера версий кросс-компилятора распространялись и на сам язык, но применительно к настоящему времени речь о версиях языка C++ не ведут. Лишь в 1998 году язык стал стандартизированным.

* C++ поддерживает как комментарии в стиле C (/\* комментарий \*/), так и однострочные (// вся оставшаяся часть строки является комментарием), где // обозначает начало комментария, а ближайший последующий символ новой строки, который не предварён символом \ (либо эквивалентным ему обозначением ??/), считается окончанием комментария. Плюс этого комментария в том, что его не обязательно заканчивать, то есть обозначать окончание комментария.
* Спецификатор inline для функций. Функция, определённая внутри тела класса, является inline по умолчанию. Данный спецификатор является подсказкой компилятору и может встроить тело функции в код вместо её непосредственного вызова.
* Квалификаторы const и volatile. В отличие от С, где const обозначает только доступ на чтение, в C++ переменная с квалификатором const должна быть инициализирована. volatile используется в описании переменных и информирует компилятор, что значение данной переменной может быть изменено способом, который компилятор не в состоянии отследить. Для переменных, объявленных volatile, компилятор не должен применять средства оптимизации, изменяющие положение переменной в памяти (например, помещающие её в регистр) или полагающиеся на неизменность значения переменной в промежутке между двумя присваиваниями ей значения. В многоядерной системе volatile помогает избегать барьеров памяти 2-го типа.
* Пространства имён (namespace).
* Для работы с памятью введены операторы new, new[], delete и delete[]. В отличие от библиотечных malloc и free, пришедших из C, данные операторы производят инициализацию объекта. Для классов это вызов конструктора, для POD типов инициализацию можно либо не проводить (new Pod;), либо провести инициализацию нулевыми значениями (new Pod(); new Pod{};).

SFML — 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. В настоящее время доступны следующие модули:

* System — управление временем и потоками, он является обязательным, так как все модули зависят от него.
* Window — управление окнами и взаимодействием с пользователем.
* Graphics — делает простым отображение графических примитивов и изображений.
* Audio — предоставляет интерфейс для управления звуком.
* Network — для сетевых приложений.

# Практическая часть

Мой проект «Игра Digger» состоит из одиннадцати файлов, включающих в себя 4 класса: один базовый, три дочерних, и две функций: *Logic* и *Map*.

# Библиотеки

**SFML/Graphics.hpp** – заголовок модуля *Graphics*, автоматически подключающий заголовок модуля *Window*.

* *Graphics* – отображение графических примитивов и изображений.
* *Window* – управление окнами и взаимодействием с пользователем.

**Sstream –**  заголовочный файл с классами, функциями и переменными для организации работы со строками, через интерфейс потоков. Он включён в стандартную библиотеку C++.

# Классы

**Being** (рус. *Существо*) является базовым классом для игрока и обитающих на карте монстров.

Включает в себя шесть полей:

* *x* – поле типа *int*, хранящее в себе условную X-координату существа;
* *y* – поле типа *int*, хранящее в себе условную Y-координату существа;
* *Life –* поле типа *bool*, хранящее в себе состояние существа – живо или мертво;
* *T* – поле типа *Texture*, хранящее в себе текстуру существа;
* *S* – поле типа *Sprite*, хранящее в себе спрайт существа;
* *freezetime* – поле типа *Clock*, хранящее в себе таймер, использующийся в методах дочерних классов для контроля скорости перемещения.

Для полей *x*, *y* и *Life* реализованы «геттеры» и «сеттеры».

**Player** (рус. *Игрок*) является дочерним классом *Being* и описывает персонажа, за которого предстоит играть пользователю.

Включает в себя:

* *score* – поле типа *int*, отвечающее за фиксацию прогресса выполнения цели.
* конструктор по-умолчанию, помимо всего прочего, загружающий текстуру и присваивающий её спрайту;
* *Move –* метод, реализующий перемещение персонажа посредствам нажатия соответствующих клавиш на клавиатуре (*WASD*).

Для *score* реализованы «геттер» и «сеттер».

**Nobbin** (название монстров из оригинальной *Digger* 1983 года) – является дочерним классом *Being* и описывает один из видов монстров, мешающих игроку достичь поставленной цели.

Включает в себя:

* *soview* – поле типа *int*, отвечающее за хранение информации о направлении взгляда монстра (вверх, вниз, влево, вправо).
* конструктор по-умолчанию, помимо всего прочего, загружающий текстуру и присваивающий её спрайту;
* *Move* – метод, отвечающий за перемещение монстра по карте и присвоение соответствующей текстуры спрайту в зависимости от направления его взгляда.

**Death** (рус. *Смерть*) является дочерним классом *Being* и описывает один из видов монстров, мешающих игроку достичь поставленной цели.

Включает в себя:

* конструктор по-умолчанию, помимо всего прочего, загружающий текстуру и присваивающий её спрайту;
* *Move* – метод, отвечающий за перемещение монстра по карте и присвоение соответствующей текстуры спрайту в зависимости от направления его взгляда.

# Функции

**Map** (рус. *Карта*) – функция, отвечающая за отрисовку блоков на игровом поле. В соответствии с информацией о карте она создаёт спрайты блоков, добавляет им текстуры, выставляет им условные координаты и отрисоввывает на экране.

Принимает следующие значения:

* \**map* – указатель на двумерный массив типа *int*, содержащий информацию об игровом поле;
* *Blockt* – переменная типа *Texture*, хранящая текстуру блока камня.
* *Goldt* – переменная типа *Texture*, хранящая текстуру блока золотой руды.
* *Obst* – переменная типа *Texture*, хранящая текстуру блока коренной породы.

Не возвращает значений.

**Logic** (рус. *Логика*) – функция, отвечающая за работу игры. Она использует все вышеупомянутые классы и функции.

Её следует разбивать на три части:

* первоначальный ввод данных;
* меню;
* игровой процесс.

Первоначальный ввод данных отвечает за объявление и инициализацию большинства переменных: текстуры, спрайты, шрифты, объекты классов, таймеры и др.

Меню отвечает за показ экрана выбора уровня и сам выбор. После того, как пользователь выбирает уровень, этот участок Logic отстраивает его структуру и размещает на игровом поле существ: игрока и монстров.

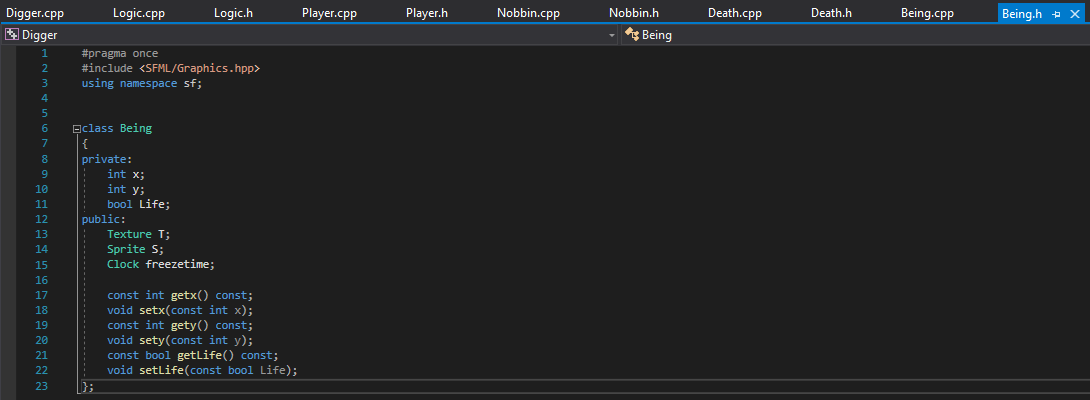
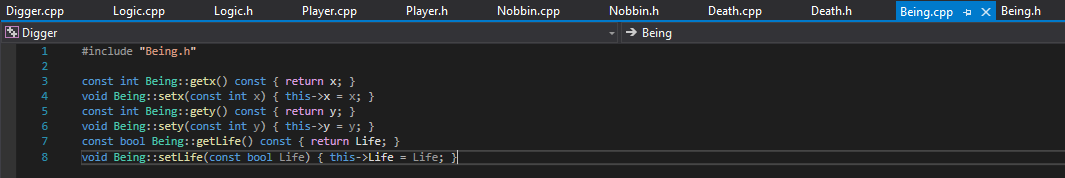
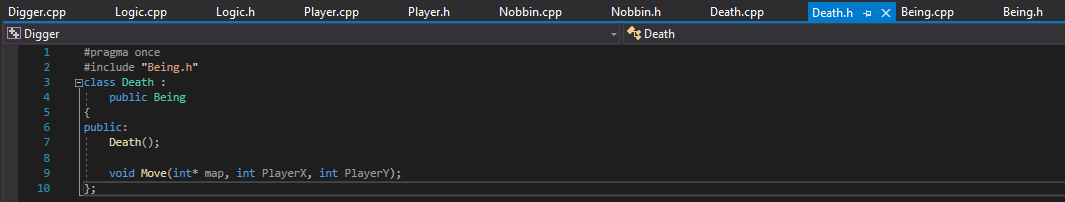
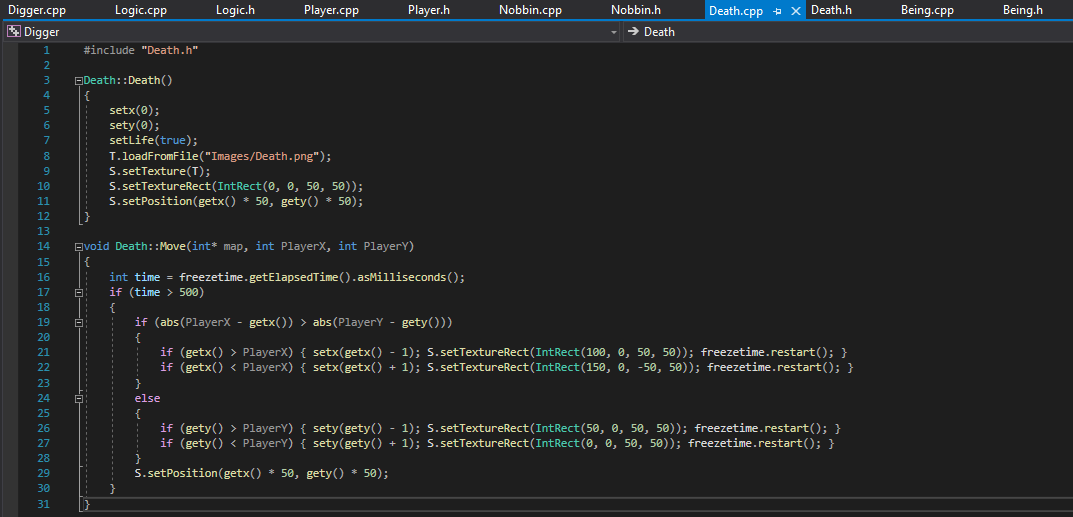
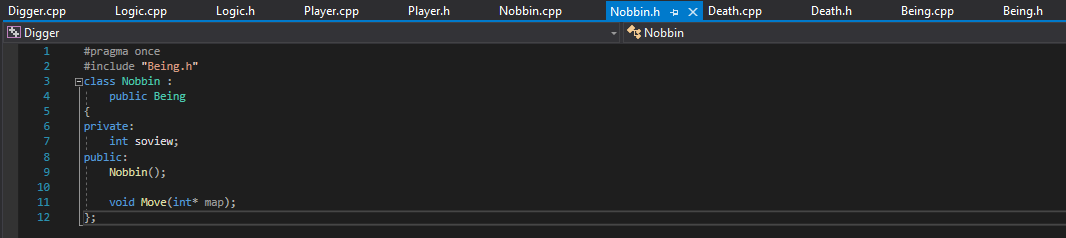
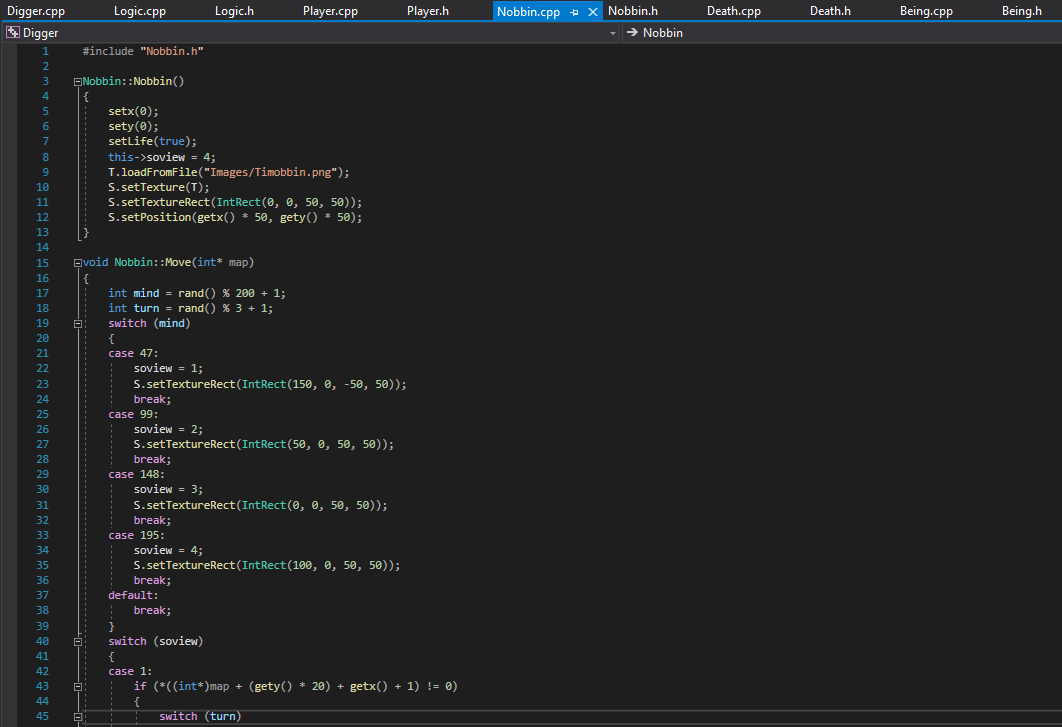
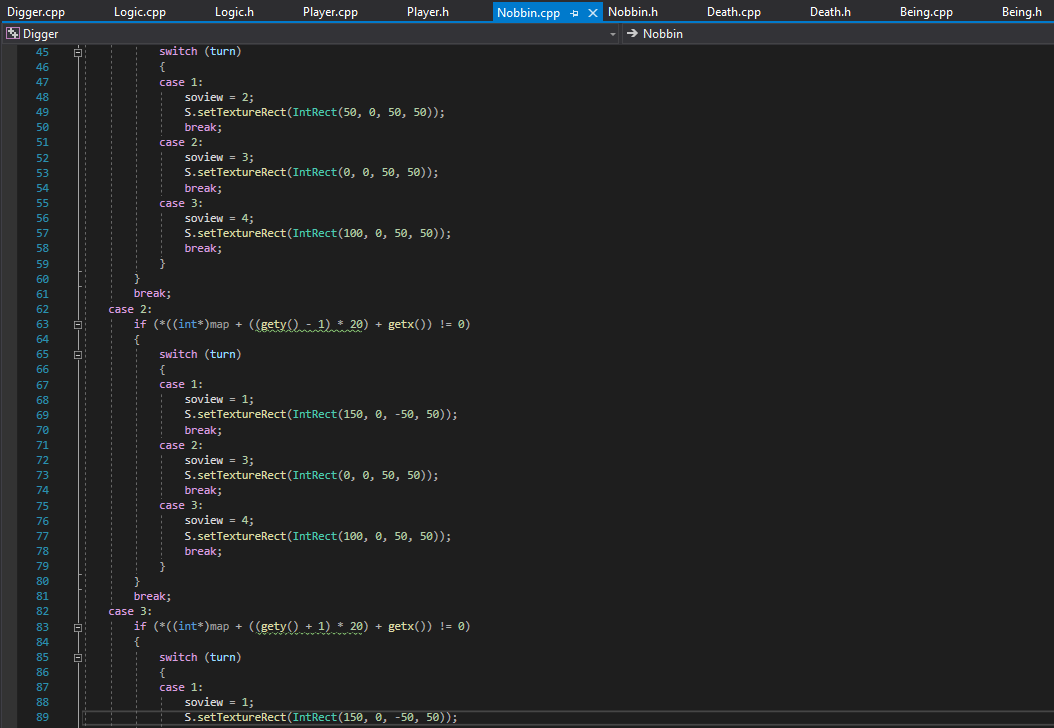
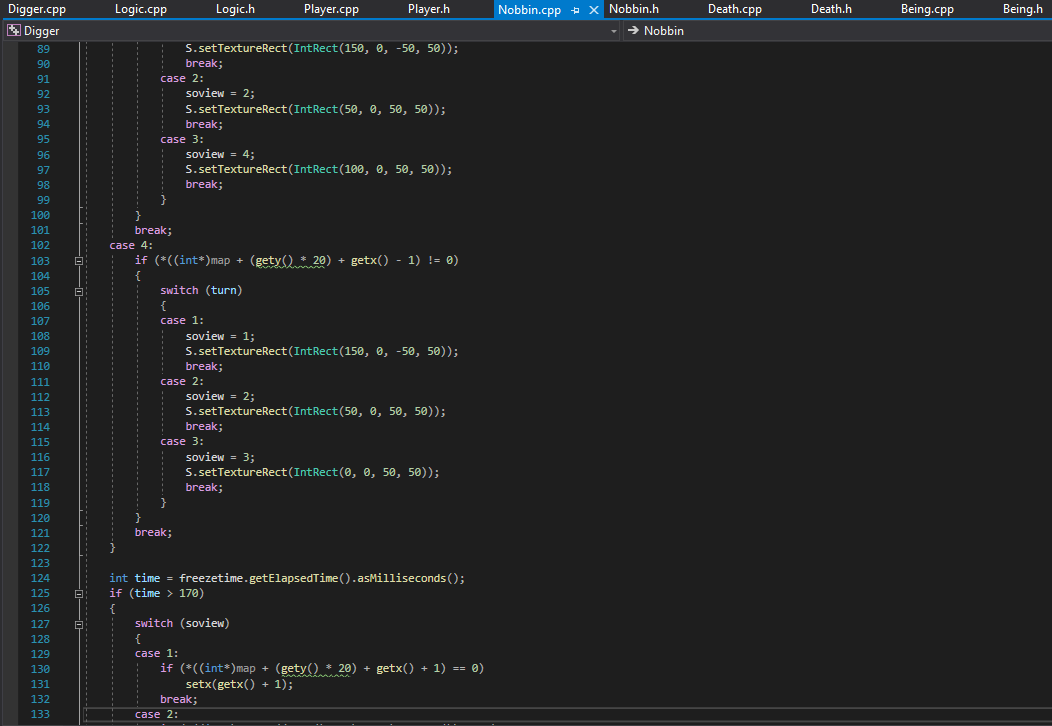
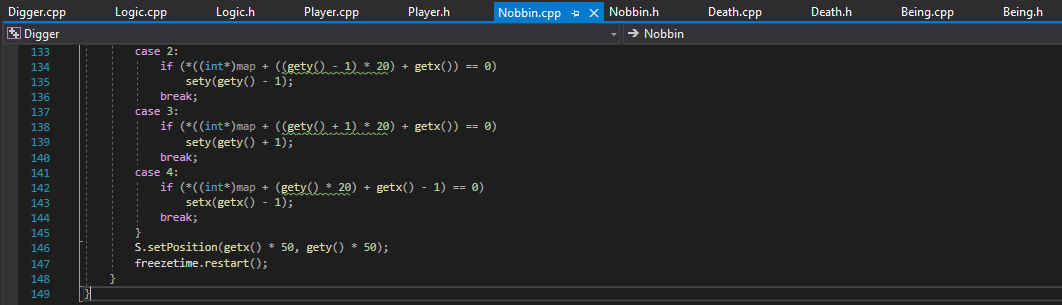
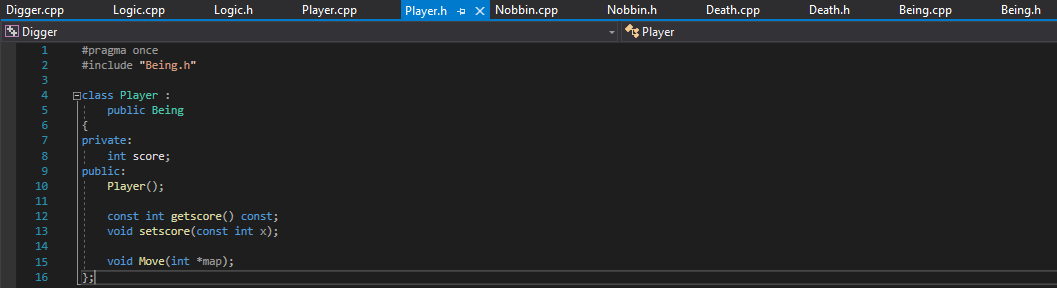
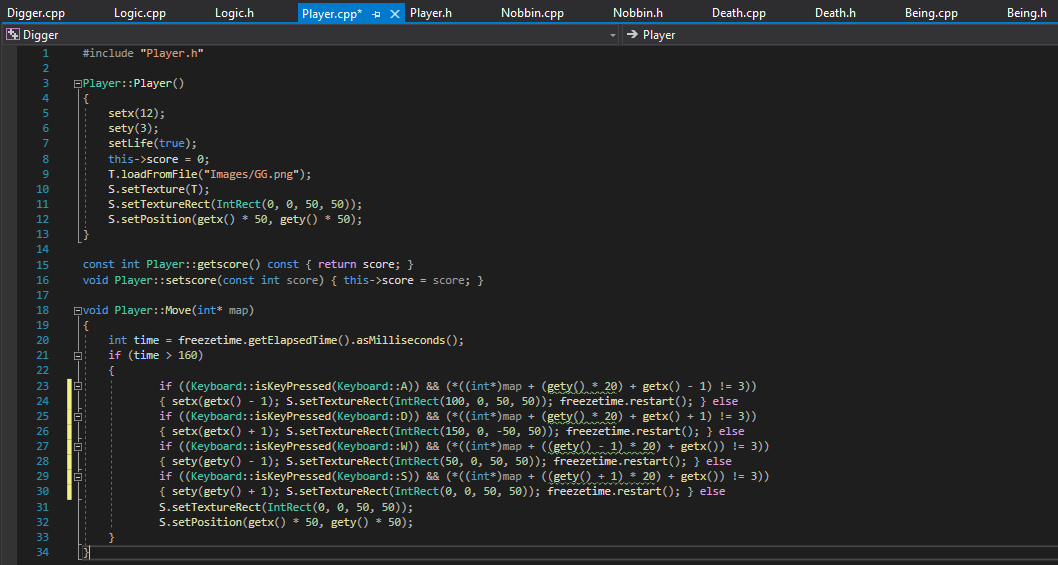
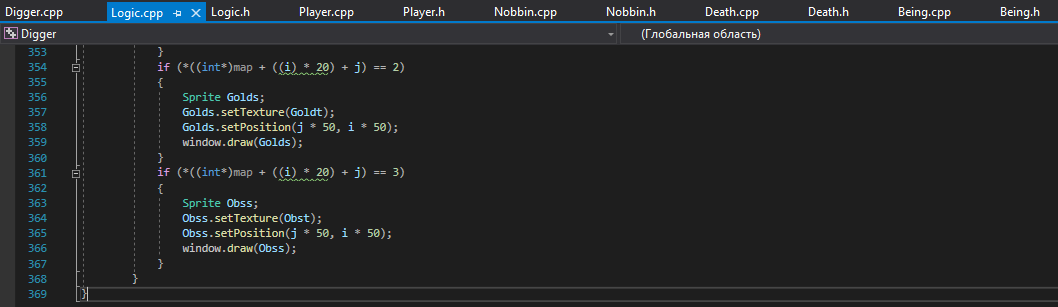
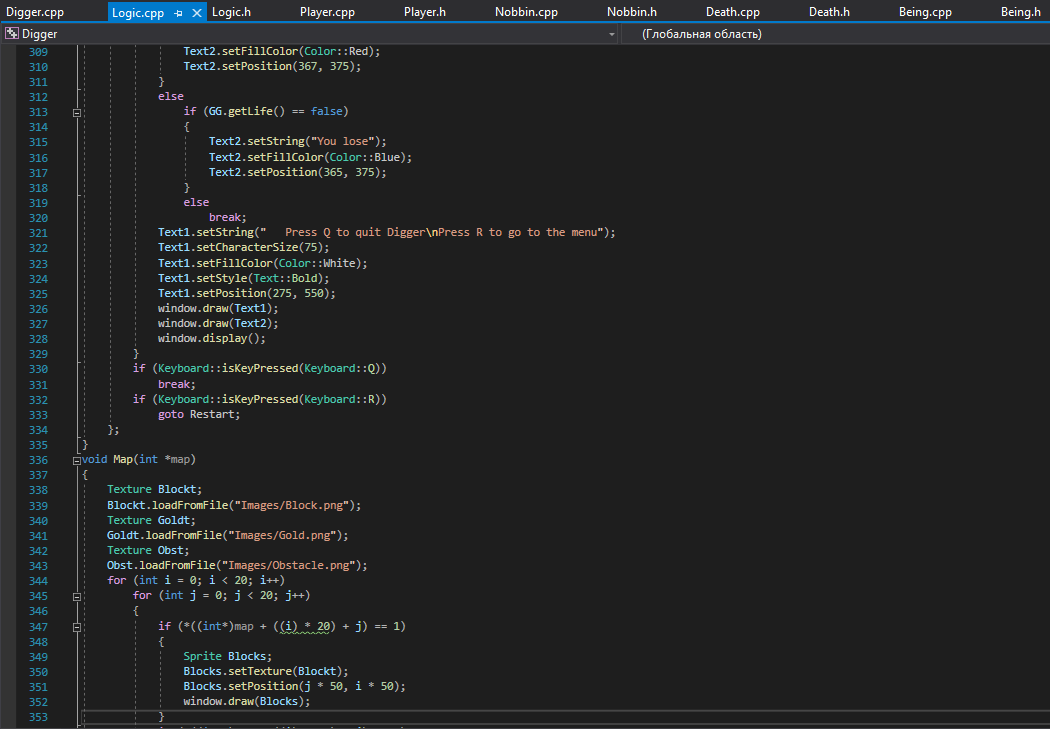
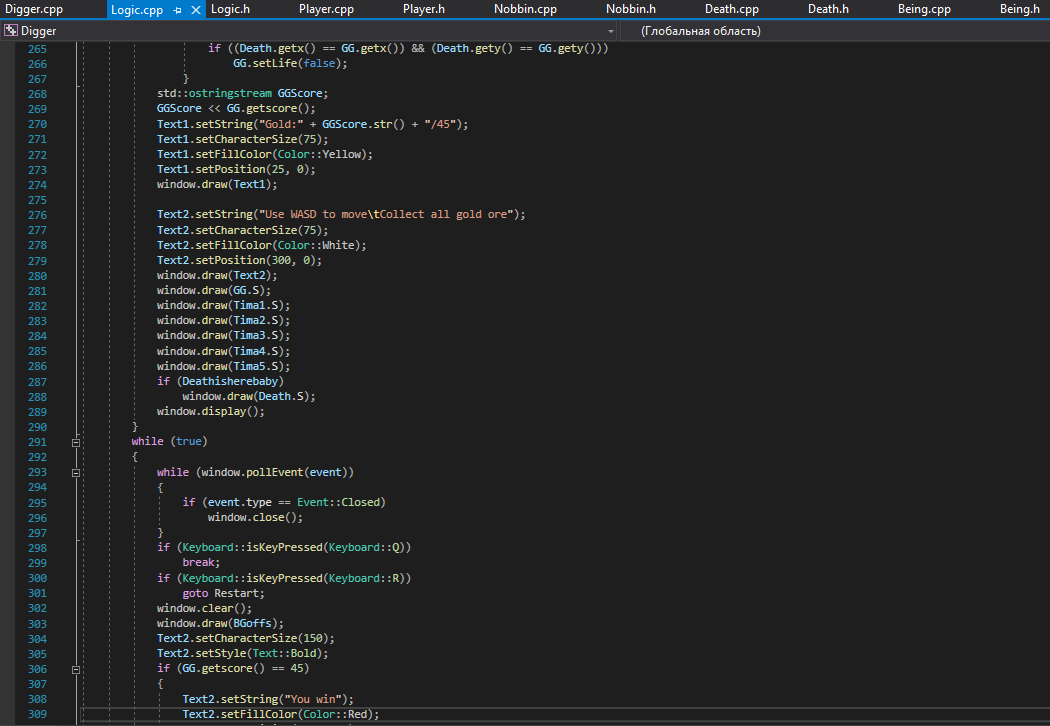
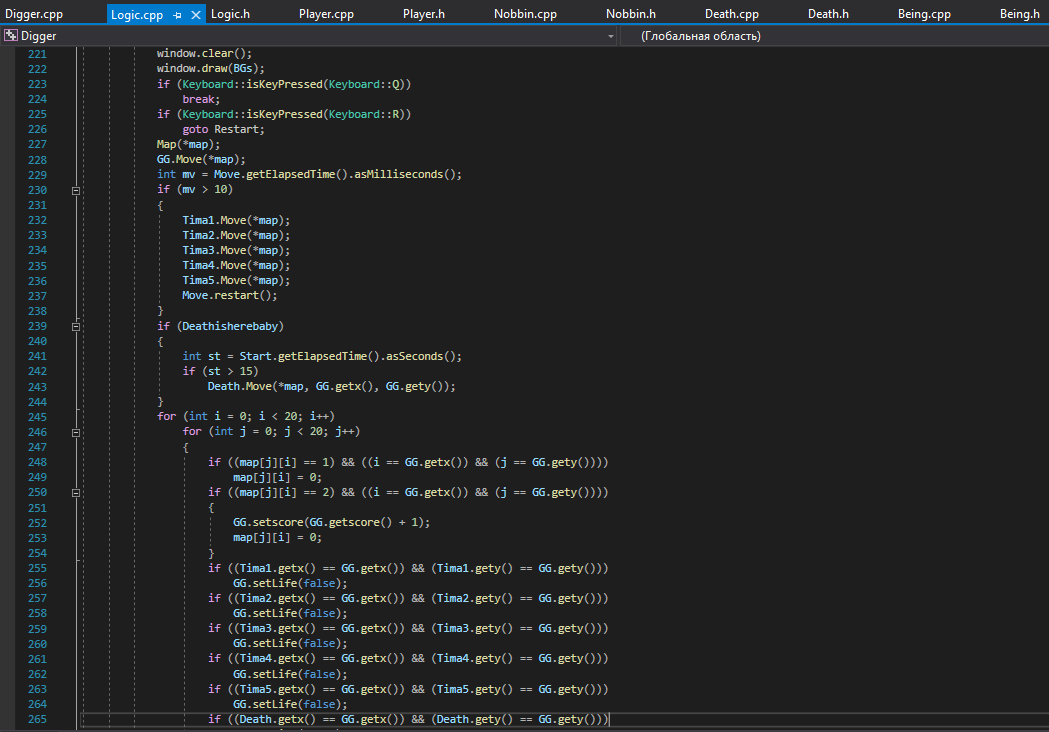
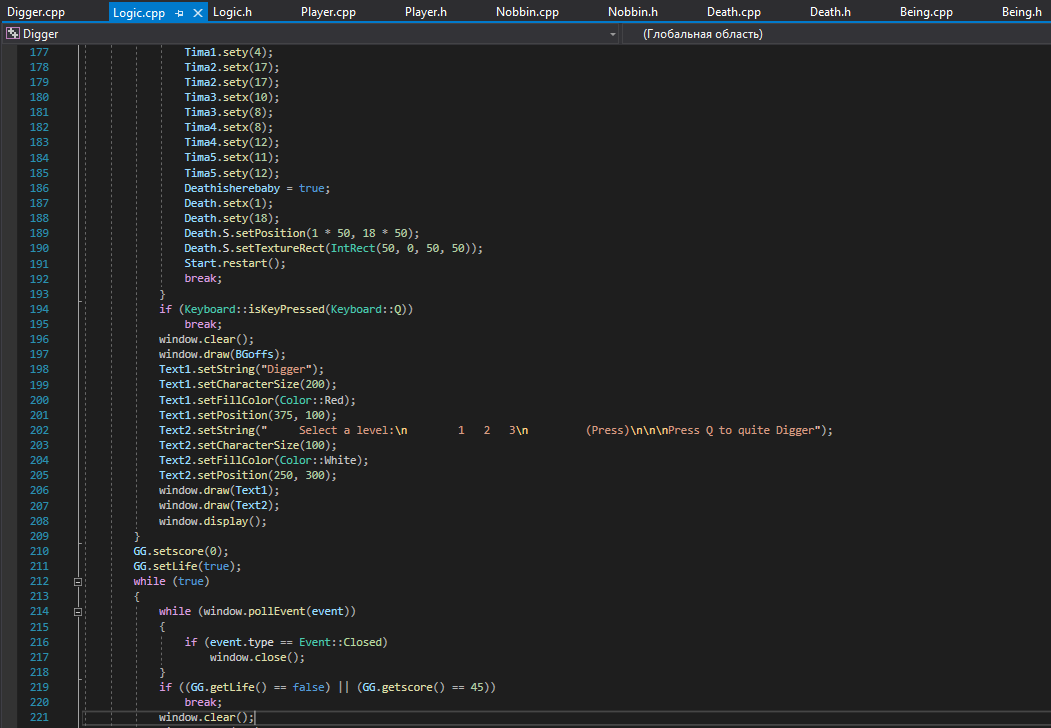
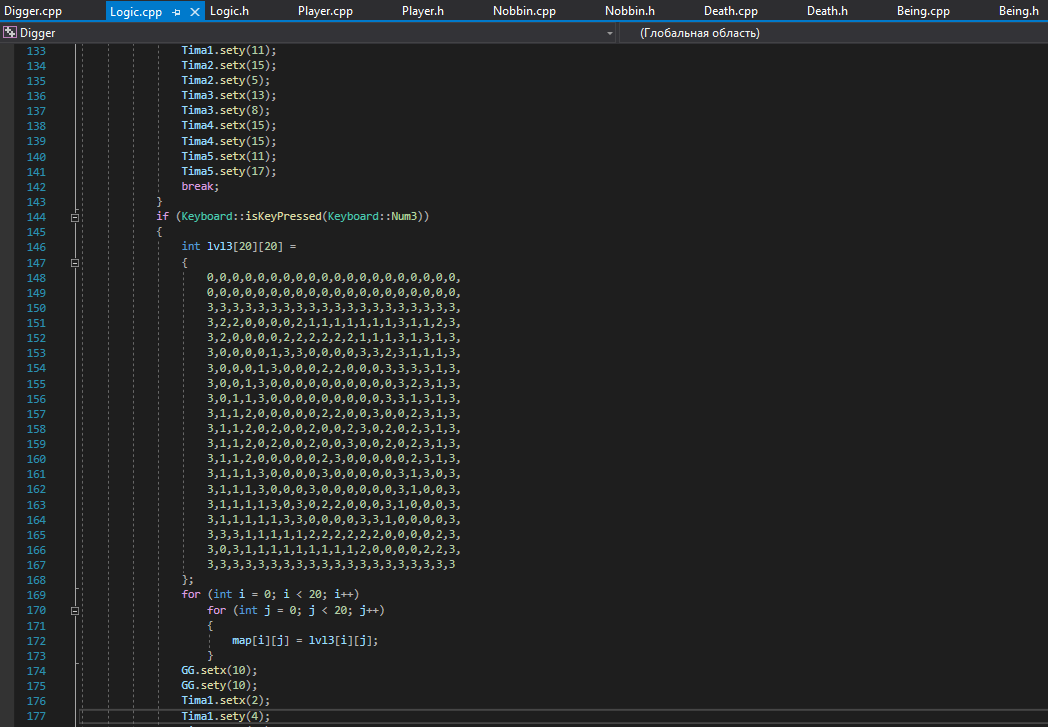
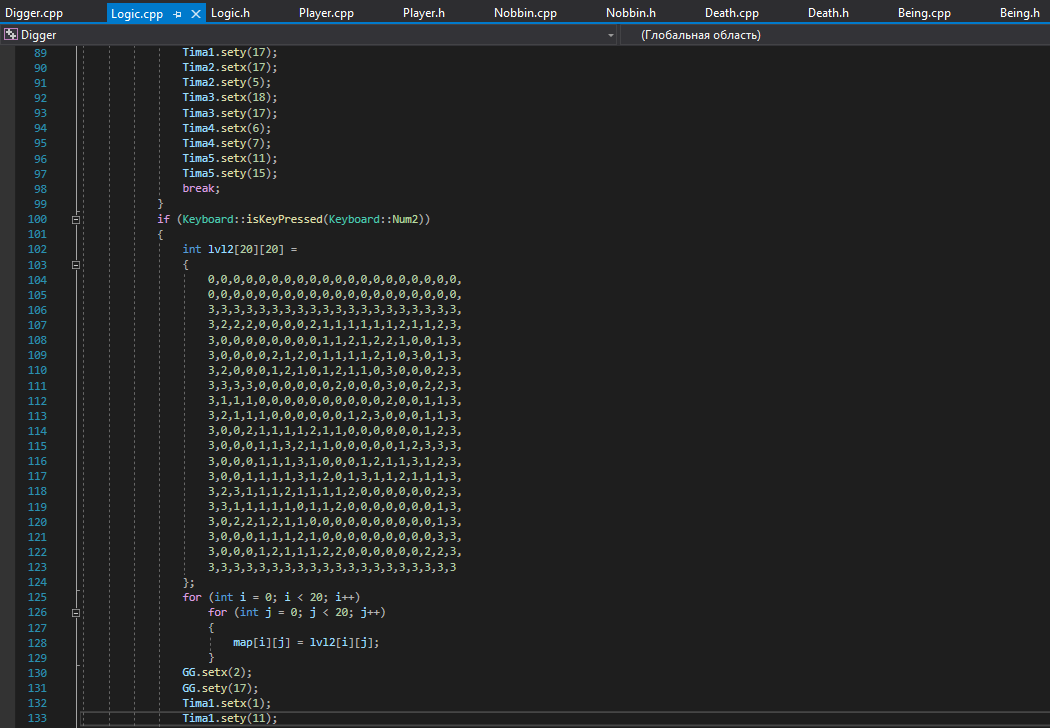
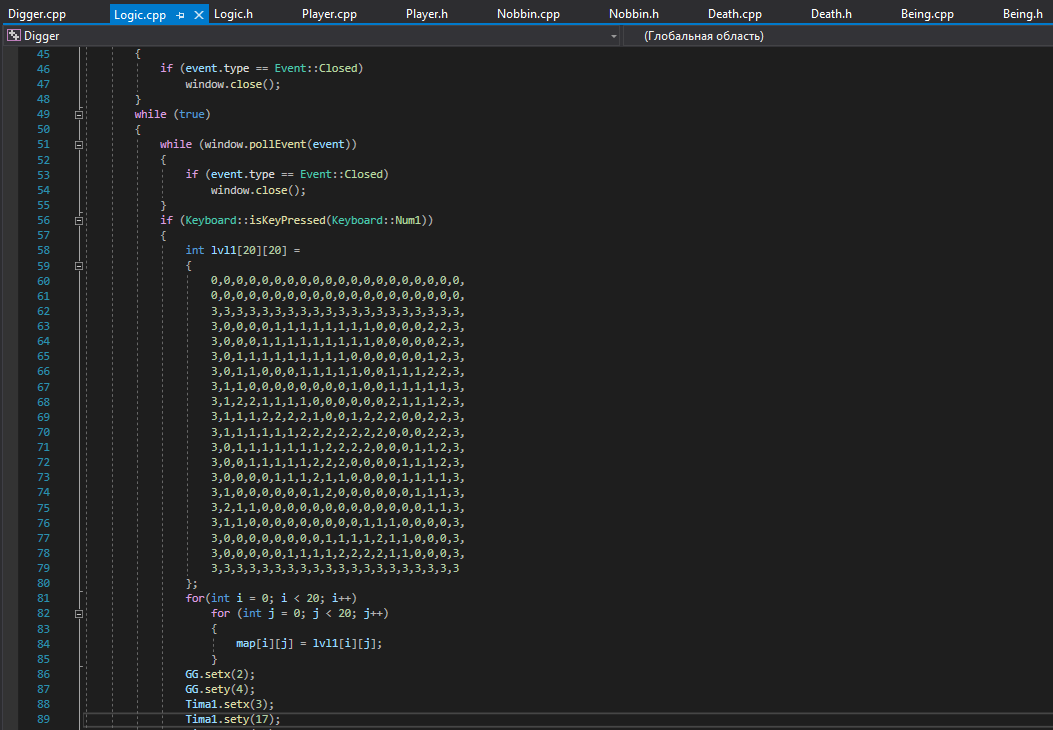
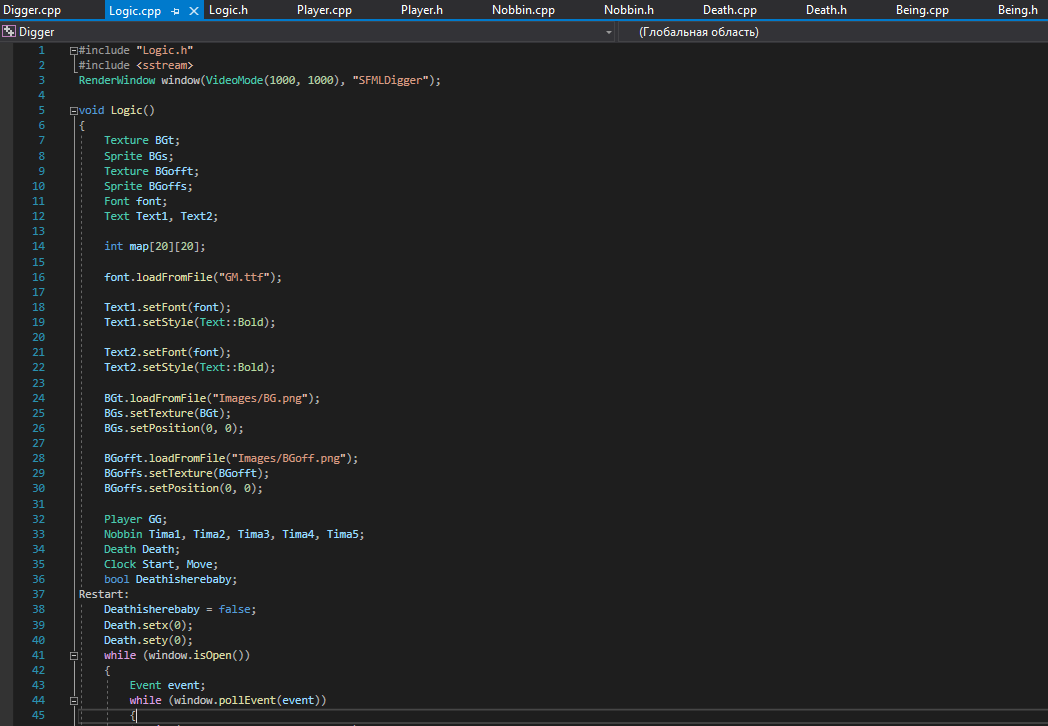
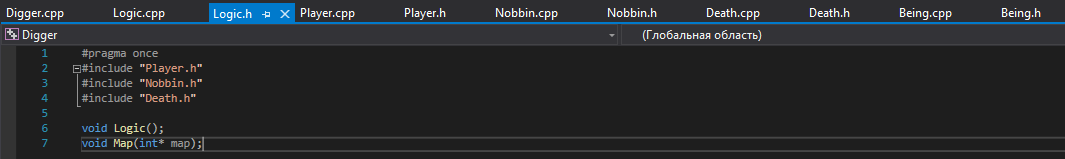
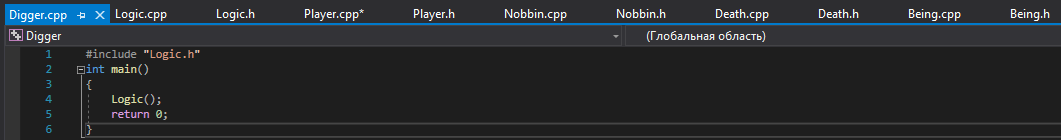
Игровой процесс в режиме реального времени изменяет карту в соответствии с игровой ситуацией, отрисоввывает интерфейс, спрайты, блоки и существ, а также проверяет условия победы и поражения.

Не принимает и не возвращает никаких значений.

# Заключение

На мой взгляд, я справился с поставленными задачами и достиг цели работы – а именно разработал игру Digger, используя полученные в ходе обучения знания о языке C++, в частности об объектно-ориентированном программировании. Приложение получилось качественным, удобным и приятным в использовании. В ходе работы я закрепил свои знания языка C++ и научился использовать мультимедийную библиотеку SFML, что несомненно поможет мне в будущем.

# Код программы

# Литература

* Статья об оригинальной игре Digger: <https://ru.wikipedia.org/wiki/Digger>;
* Плэйлист «Основы C++»: <https://www.youtube.com/playlist?list=PLQOaTSbfxUtCrKs0nicOg2npJQYSPGO9r>;
* Плэйлист «Уроки SFML»: <https://www.youtube.com/playlist?list=PLskEAOuS4gQMoAE68-71KRxQegr8rSw54>.