содержание

ВВЕДЕНИЕ.

1 СИСТЕМНЫЙ АНАЛИЗ И ПОСТАНОВКА ЗАДАЧ 5

2 ПРОЕКТИРОВАНИЕ СИСТЕМЫ 8

2.1 Модель проектирования 8

2.2 Разработка графического интерфейса пользователя 9

3 РЕАЛИЗАЦИЯ СИСТЕМЫ 10

3.1 Определение требований к техническим средствам 10

3.2 Определение формы представления входных и выходных данных 11

3.3 Проектирование программного обеспечения 12

3.4 Спецификация программы 14

4 ТЕСТИРОВАНИЕ СИСТЕМЫ 18

ЗАКЛЮЧЕНИЕ 23

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 24

ПРИЛОЖЕНИЕ А СХЕМА ПРОГРАММЫ

­­

# ВВЕДЕНИЕ.

Будние дни в большом городе. Шум на улицах, работа, вечные проблемы и заботы... И все это изо дня в день. Всем это рано или поздно надоедает, и у каждого возникает желание почувствовать себя свободным от все забот и проблем, хотя бы на несколько минут. Но не всегда получается выделить для отдыха большое количество времени. И тогда, на помощь "уставшим" приходят современные компьютеры. Они дают возможность "уйти" от "серых" будней и погрузиться в игровую реальность, которая дает возможность почувствовать себя свободным и не обремененным заботами, при этом не заставляя тратить время или деньги на дорогу, и всегда оставляя возможность максимально быстро вернуться к работе.

Каждому интересны определенные жанры игр, но никого не оставит равнодушным старая добрая игра "Battle city", так полюбившаяся как детям, так и взрослым еще в 90x годах. Но кому то мог надоесть старый режим игры. Или просто захотелось чего то нового. Тогда стоит попробовать создать свою игру, со старой графикой, но с новым смыслом. Такой игрой может стать игра "Танки 2D".

# СИСТЕМНЫЙ АНАЛИЗ И ПОСТАНОВКА ЗАДАЧИ.

Для реализации этой задачи использовался язык программирования С++ и среда разработки Qt Creator. В основном Qt используют для создания очень быстрых и высокопроизводительных приложений. Это мессенджеры, игры или сложные ресурсоемкие программы. Он популярен в сферах, имеющих повышенные требования к безопасности ПО. Среди них:

1. Промышленность и транспортная отрасль. На C++ и на Qt пишут программы для роботов, которые используются на производствах, в перевозке грузов и других похожих отраслях. Qt используют при написании программного обеспечения для автомобилей, кораблей и других видов транспорта;

2. MedTech. Фреймворк применяют при создании программных систем и интерфейсов для медицинского оборудования;

3. IoT. На C++ с Qt пишут логику для «умных» приборов, которые подключаются к интернету вещей.

С использованием Qt написаны мессенджер Telegram, продукты Autodesk, окружение рабочего стола для многих систем под ядром Linux и пр.

Танки 2D - одиночная игра между игроком и ботом на случайно сгенерированной карте, которая имеет стены и базу. Игрок управляет мирным танком.

Задача разработать небольшую программу игры в танки 2D на случайно сгенерированной карте с применением графики, в которой нет ожиданием хода от игрока или бота.

Игра ведется на случайно сгенерированной карте, разделенной стенами, рассчитана на одного игрока. В ходе игры танк игрока и танк противника двигаются по карте, уничтожая преграды и стреляют из орудий.

Цель игры – уничтожить как можно больше противников для получения максимального количества очков. Игра заканчивается при получении урона от противника, или когда противник уничтожит базу.

Максимального предела очкам нет.

Компоненты, которые должны входить в приложение можно стандартно разделить на две составляющие: интерфейс и программная часть.

В интерфейсе должно присутствовать игровое поле, возможность стрелять или двигаться, при нажатии на соответствующую клавишу, выводиться информация о ходе игры.

Программно игра должна быть реализована с использованием двумерного массива, представляющий игровое поле. Каждый элемент массива представляет клетку поля. Индексы массива соответствуют положению танка игрока, танка противника и стен.

При стрельбе сверяются положения снаряда и координат танка. Если координаты совпадают, танк уничтожается.

Входные данные:

1) Код нажатых клавиш клавиатуры.

Выходные данные:

1) Игровое поле, заполненное преградами, отображаемое на экране.

2) В случае поражения игрока, вывод соответствующего сообщения.

Для всех операций были использованы стандартные возможности Qt Creator, так что не понадобилось даже использования сторонних библиотек.

Целью курсовой работы является создание компьютерной игры «Танки 2D».

В данной курсовой работе я ставлю перед собой следующие задачи:

1. В программе должно присутствовать меню, в котором можно просматривать данные о разработчике и игре, и непосредственно переход к самой игре.

2. Должна быть база игрока при уничтожении которой, игра заканчивается.

3. Игрок должен уничтожать противников. Противник должны уничтожить базу или танк.

4. Должна присутствовать возможность уничтожения объектов карты.

Необходимо реализовать следующие запросы:

1. Создание карты.
2. Добавление объектов игры.
3. Взаимодействие объектов игры.
4. Реализация меню.

# 2 ПРОЕКТИРОВАНИЕ СИСТЕМЫ

**2.1 Модель проектирования**

Для выполнения задачи потребовалось подключение стандартных библиотек QMainWindow, QPushButton, QVBoxLayout, QMediaPlayer, QKeyEvent, QGraphicsItem, QGraphicsPixmapItem, QGraphicsRectItem, QTimer, QObject:

QMainWindow – обеспечивает основу для создания пользовательского интерфейса приложения;

QPushButton – представляет собой командную кнопку;

QVBoxLayout – выстраивает виджеты в вертикальную линию;

QMediaPlayer – подключение музыки к проекту;

QKeyEvent – описывает нажатие клавиш;

QGraphicsItem – является базовым классом для всех графических элементов;

QGraphicsRecItem – предоставляет прямоугольный элемент;

При запуске игры создается карта. Далее происходит генерация стен и базы. Случайно выбирается положение танка игрока и танка противника. Отображается количество очков и управление в игре.

В ходе объектно-ориентированного анализа в соответствие с каждым прецедентом были поставлены классы, отвечающие за представленный в прецеденте функционал.

Класс Gamemenu – отображает окно с кнопками: запуск игры, о разработчике, настройки.

Класс Player – создает танк игрока. Хранит изображение танка, функции передвижения, стрельбы и любого другого взаимодействия с игровым полем.

Класс Enemy – создает танк – бот. Так же, как и класс Player, хранит в себе изображение танка – бота, функции стрельбы, передвижения и любого другого взаимодействия с игровым полем.

Класс Bullet – создает снаряд. Хранит изображение снаряда, функцию перемещения и сравнения координат с другими объектами игрового поля.

Класс Score – отображает очки, заработанные игроком за время игры.

Класс Gameloop – хранит все процессы, происходящие в игре такие как: создание и генерация игрового поля, выбор позиции танка и танка – бота, уничтожение объектов, цикл игры.

Визуализация игровой логики реализована с помощью диаграммы классов. Данная диаграмма состоит из двух частей: первая часть - это классы описывающие отображение игровых объектов, вторая часть классов описывает игровую логику. Данные классы связаны при помощи класса Gamemenu отвечающего за игровой процесс и перехватывающий события с остальных классов системы.

**2.2 Разработка графического интерфейса пользователя**

На основе данных полученных в результате анализа и проектирования была разработана структура графического интерфейса пользователя, которая содержит связи между основными экранами игры и модальными окнами, вызываемыми на этих экранах.

Главное меню игры состоит из названия игры и 3х кнопок: «Старт», «О программе» и «Настройки». При нажатии на кнопку «Старт» открывается модальное окно игры. Кнопка «О разработчике» выводит виджет о создателе проекта. Кнопка «Настройки» изменяет размер шрифта и кнопок.

Экран игры состоит из области игрового поля и очков. Область игрового поля показывает все происходящее в игре. Область очков показывает количество набранных очков и управление в игре.

Для реализации игры и главного меню, в Qt Creator используется виджеты и окна. Виджеты в контексте Qt своеобразный контейнер, в котором могут размещаться различные игровые объекты, звуки, освещение, а также элементы пользовательского интерфейса.

# 3 РЕАЛИЗАЦИЯ СИСТЕМЫ

**3.1 Определение требований к техническим средствам**

Разрабатываемая программа должна функционировать на персональных компьютерах с минимальной конфигурацией: CPU - 100MHz, RAM - 4Mb, HDD 100 Mb, keyboard, mouse, работающими под управлением операционной системы Windows XP, MacOS или более поздней версии.

Программное обеспечение включает в себя комплекс системных, прикладных и диагностических программных средств, необходимых для разработки и эксплуатации проектируемой игры.

К программному обеспечению диагностики и контроля относятся вспомогательные программы (утилиты) для выполнения различных сервисных функций, таких как проверка системы на вирусы, дефрагментация дискового пространства и т.д.

**3.2 Определение формы представления входных и выходных данных**

От правильной реализации диалога пользователя с системой во многом зависит успех системы, а также эффективность ее использования. При разработке системы к диалогу предъявляются следующие требования: требование гибкости диалога. Гибкость диалога — это мера того, насколько хорошо он соответствует различным уровням подготовки и производительности труда пользователя требование естественности диалога. Естественность — это свойство диалога, при наличии которого пользователь не изменяет свои традиционные способы решения. Немаловажен также порядок диалога. Всегда следует придерживаться такого порядка, в котором пользователь обычно обрабатывает информацию. Требование к последовательности диалога. Последовательность диалога гарантирует, что пользователь, освоивший работу с одной частью системы, освоит работу с другими ее частями. Данное требование к диалогу состоит из последовательности использования форматов данных, последовательности в размещении данных на экране и последовательности при выделении информации.

**3.3 Проектирование программного обеспечения**

**Реализованные пользовательские классы:­­**

Gamemenu.h– класс меню программы;

Client.h – класс обновления игры;

Gameloop.h – класс для создания всех объектов и старта игры;

Player.h – класс игрока;

Enemy.h – класс противника;

Bullet.h – класс снарядов;

Score.h – класс подсчета и вывода очков;

**Реализованные пользовательские процедуры:**

void startGame();

void about();

void setMenu();

int \*\*ft\_get\_map();

void ft\_update\_map(int \*\*map);

void ft\_print\_map();

void ft\_set\_headquarters();

void ft\_add\_obstacle();

void ft\_make\_map();

void ft\_destroy\_map();

void keyPressEvent(QKeyEvent \*event);

int ft\_get\_x();

int ft\_get\_y();

void ft\_set\_x(int x);

void ft\_set\_y(int y);

void ft\_spanw();

void ft\_create\_map(int \*\*map);

void ft\_update\_map(int \*\*map);

int \*\*ft\_player\_map();

int ft\_rotate\_left();

int ft\_rotate\_right();

int ft\_rotate\_up();

int ft\_rotate\_down();

void ft\_set\_grid(QGraphicsPixmapItem \*\*grid);

QGraphicsPixmapItem \*\*ft\_get\_grid();

void ft\_set\_enemy(Enemy \*e);

Enemy \*ft\_get\_enemy();

void ft\_incr\_score();

Score \*ft\_get\_score();

void ft\_set\_score(Score \*score);

void ft\_increase(int flag);

void move();

void shoot();

int ft\_get\_x();

int ft\_get\_y();

void ft\_spawn();

void ft\_set\_pl(Player \*pl);

Player \*ft\_get\_pl();

Int ft\_get\_direct();

Bullet(int direct, int \*\*map, Player \*pl, int flag);

int \*\*ft\_get\_map();

int ft\_check\_hit(int x, int y);



Рисунок 3.1 – Меню игры.

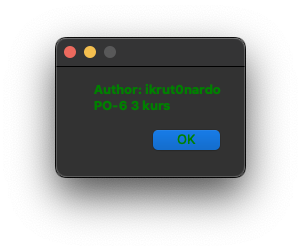


Рисунок 3.2 – О приложении.



Рисунка 3.3 – Окно игры.

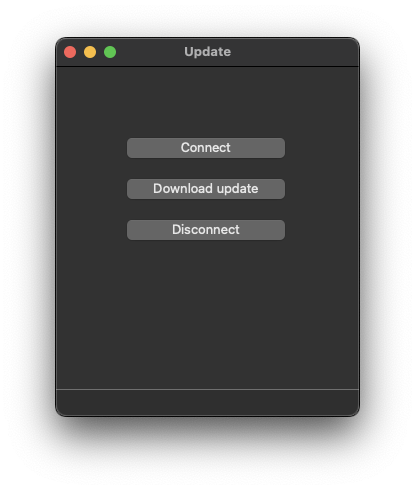


Рисунок 3.4 – Обновление игры.

**3.4 Спецификация программы.**

Создание меню игры, добавление и подключение кнопок меню:

gameMenu::gameMenu()

{

this->window = new QWidget();

button1 = new QPushButton("start");

button2 = new QPushButton("about");

button3 = new QPushButton("check for update");

QVBoxLayout \*layout = new QVBoxLayout(*window*);

layout->addWidget(*button1*);

layout->addWidget(*button2*);

layout->addWidget(*button3*);

QFont font("Times New Roman");

this->window->setFont(font);

this->window->resize(400,500);

this->window->show();

connect(button1,SIGNAL(released()),this,SLOT(startGame()));

connect(button2,SIGNAL(released()),this,SLOT(about()));

connect(button3,SIGNAL(released()),this,SLOT(setMenu()));

}

Запуск игры:

void gameMenu::startGame()

{

GameLoop \*gm;

gm = new GameLoop;

}

Окно «О игре»:

void gameMenu::about()

{

About \*about = new About();

}

Запуск средства обновления игры:

void gameMenu::setMenu()

{

window->close();

Client \*w = new Client();

w->setWindowTitle("Update");

w->show();

}

Создание поля игры:

void GameLoop::ft\_make\_map()

{

int i;

int j;

i = 0;

this->map = new int\*[MAP\_SIZE];

while (i < MAP\_SIZE)

{

this->map[i] = new int[MAP\_SIZE];

i++;

}

i = 0;

while (i < MAP\_SIZE)

{

j = 0;

while (j < MAP\_SIZE)

{

if (i == 0 || i == MAP\_SIZE - 1

|| j == 0 || j == MAP\_SIZE - 1)

this->map[i][j] = 3;

else

this->map[i][j] = 0;

j++;

}

i++;

}

ft\_set\_headquarters();

ft\_add\_obstacle();

this->map[1][1] = 0;

this->map[1][MAP\_SIZE - 2] = 0;

this->map[MAP\_SIZE - 2][1] = 0;

this->map[MAP\_SIZE - 2][MAP\_SIZE - 2] = 0;

}

Функции создание танка игрока и расположение на игровом поле:

Player::Player()

void Player::ft\_spanw()

Создание противника, перемещение противника, стрельба противника:

Enemy::Enemy(int \*\*map):QObject(),QGraphicsPixmapItem()

void Enemy::move()

void Enemy::shoot()

Создание и перемещение пуль:

Bullet::Bullet(int direct, int \*\*map, Player \*pl, int flag) : QObject() , QGraphicsPixmapItem()

void Bullet::move()

Создание и изменение очков в игре:

Score::Score(QGraphicsItem\*parent):QGraphicsTextItem(parent)

{

// set the score

this->score = 0;

// draw text

setPlainText("SCORE: " + QString::number(this->score) + "\nSPACEBAR TO SHOOT\nARROWS TO MOVE\n");

setDefaultTextColor(QColor("white"));

setFont(QFont("Helvetica", 15));

}

void Score::ft\_increase(int flag)

{

if (flag == 1)

{

this->score += 10;

setPlainText("SCORE: " + QString::number(this->score) + "\nSPACEBAR TO SHOOT\nARROWS TO MOVE\n");

}

if (flag == 2)

{

setPlainText("YOU LOSE\nYour score = "+QString::number(this->score));

setFont(QFont("Helvetica", 60));

setPos(300, 400);

}

}

# 4 ТЕСТИРОВАНИЕ СИСТЕМЫ

Тестирование программы проводилось на удобство пользования и функционирование программного продукта. При тестировании на удобство пользования пользователи, работая с программой, выполняли различные операции. Примером вариантов заданий для выполнения были: запуск различных меню игры, работоспособность самой игры, перемещение в игре, стрельба, вывод окончания игры.

При проведении тестов пользователь успешно справился с предложенными заданиями, что позволяет сказать о том, что разработанная игра обладает интуитивно понятным интерфейсом.

При функциональном тестировании программы проводилась проверка каждой команды меню. Как выяснилось, каждый пункт меню вызывает соответствующее окно, отображающее информацию. Тестирование на скорость позволяет определить время выполнения задачи. Выборка необходимой информации проводилась достаточно быстро, что является довольно приемлемой скоростью для такой игры.

**Тестирование программы:**

1. Проверка кнопок «меню» (см. рисунок 4.1). «Меню» предлагает нам выбор:

«start», «about», «check for updates».

1. Работоспособность игры, перемещение танка игрока и танка противника.

(см. рисунок 4.2).

Для перемещения по карте используем клавиши «стрелки» и для стрельбы клавишу «space». (см. рисунок 4.3).

Уничтожение объектов игрового поля. (см. рисунок 4.4).

Вывод конца игры и очков на экран монитора. (см. рисунок 4.5).

Кнопка «about» - для демонстрации создателя игры. (см. рисунок 4.6).

Кнопка «check for updates» - для утилиты обновления игры. (см. рисунок 4.7).

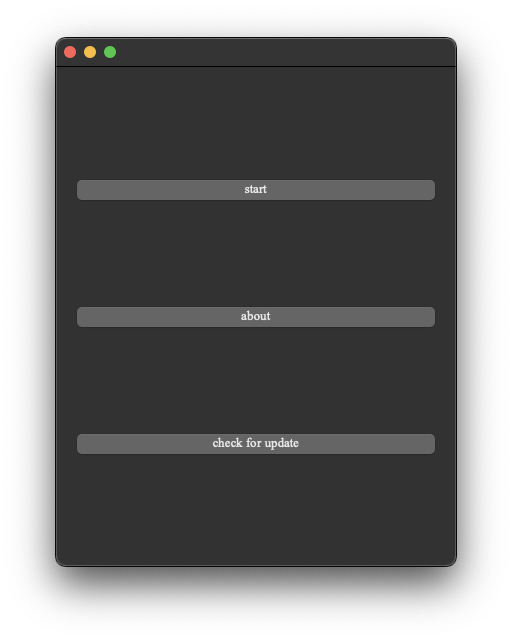


Рисунок 4.1 – Меню игры.



Рисунок 4.2 – Работоспособность игры.

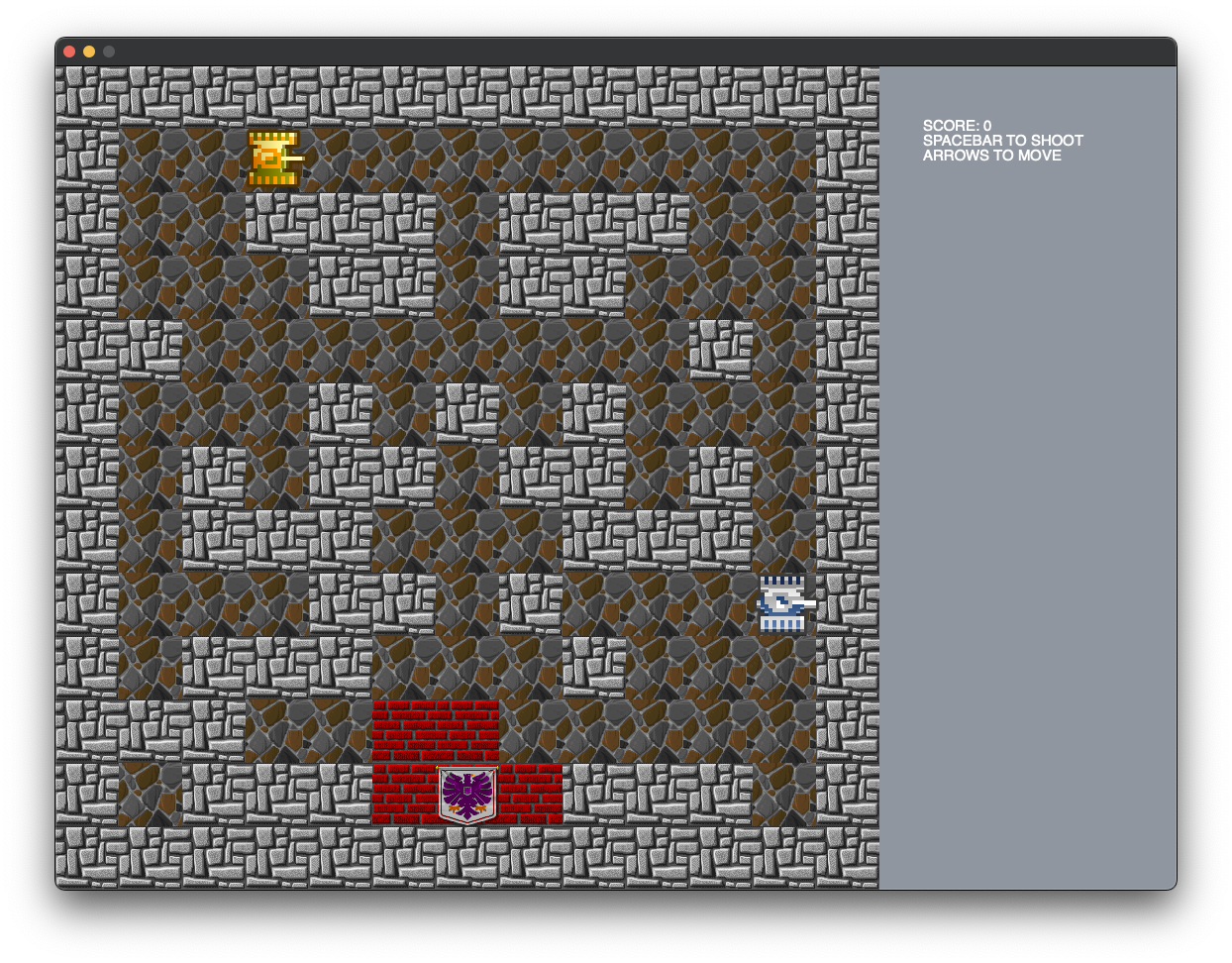


Рисунок 4.3 – Перемещение и стрельба.

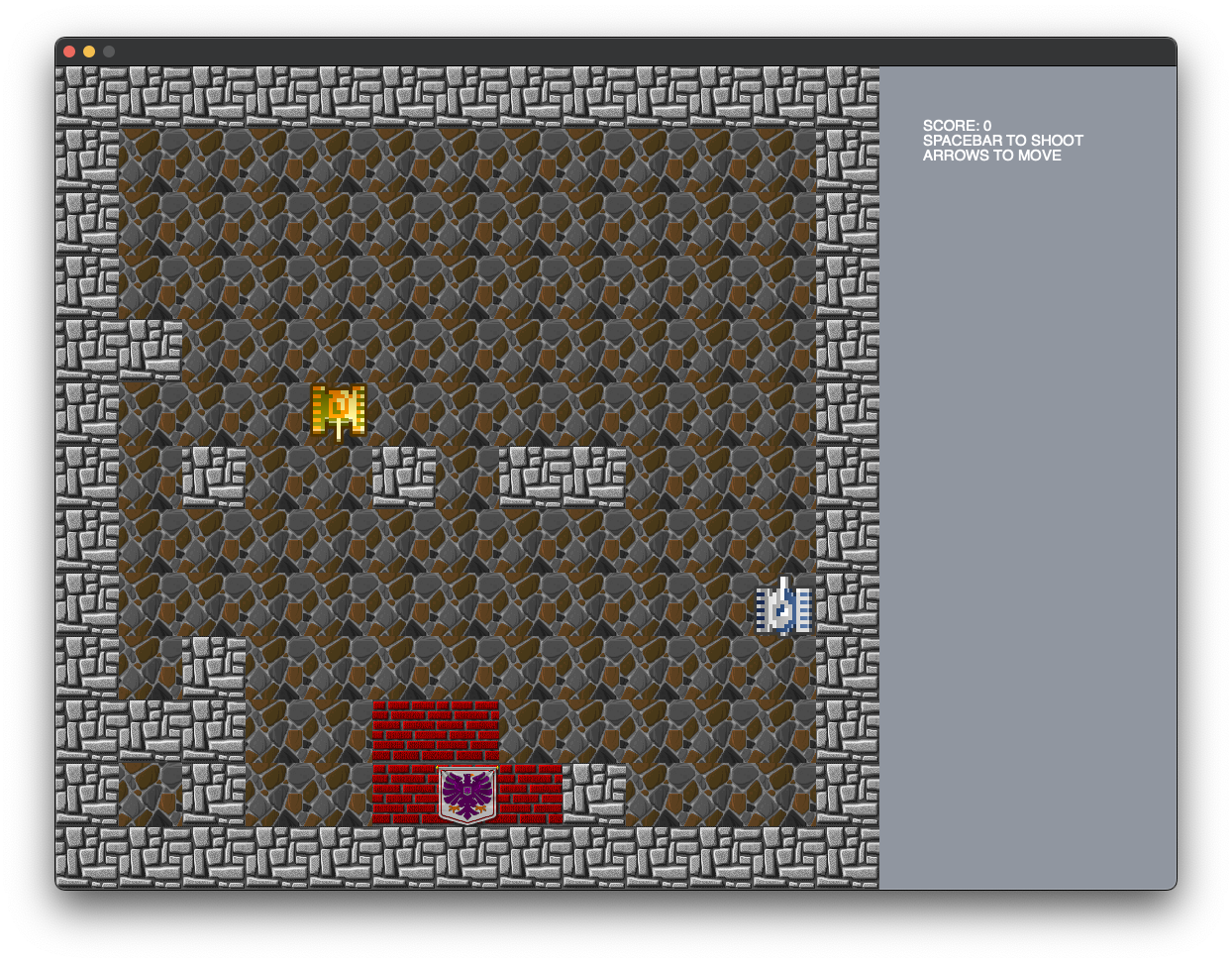


Рисунок 4.4 – Уничтожение объектов игры.



Рисунок 4.5 – Конец игры.

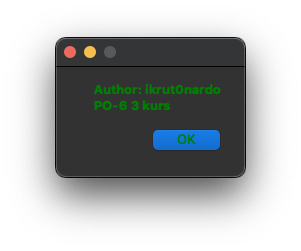


Рисунок 4.6 – Кнопка «about».

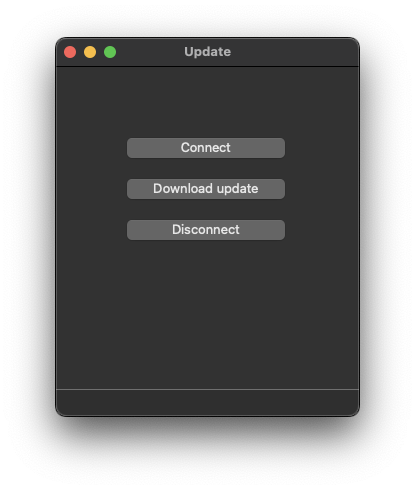


Рисунок 4.7 – Кнопка «check for update»

# Заключение

В ходе выполнения данной работы на основе различных источников были повышены знания в области ООП. И навыки программирования на языке Qt.

В результате выполнения проекта была полностью реализована игра «Танки 2D», используя объектно-ориентированное программирование.

В процессе тестирования приложения ошибок не обнаружено.

В ходе выполнения курсовой работы были более подробно изучены и изложены в пояснительной записке следующие вопросы:

- Основная информация по языку Qt.

- Ключевые алгоритмы и фрагменты программы, отвечающие за корректную работу программы.

- базовые элементы управления, их свойства и методы.

- управление игрой с помощью клавиатуры.

- Разработка игры "Танки 2D" на языке Qt.

Результатом работы является приложение-игра "Танки 2D", работающие на компьютере с любой ОС. (С установленным интерпретатором языка Qt)

# Список используемых источников

1 Макс Шлее. Qt 5.10. Профессиональное программирование на C++: наиболее полное руководство, 2022.- 1052с.

2 Роберт Мартин. Чистый код. Создание, анализ и рефакторинг / Роберт Мартин – СПБ.: БХВ - Санкт-Петербург. - 2020. — 425с.

3 Как написать игру на Qt - Урок 1. Управление объектом. [Электронный ресурс] – 2021. – Режим доступа: https://evileg.com/ru/post/98/- Дата доступа: 10.12.2022.

4 Как написать игру на C++. [Электронный ресурс] – 2021. – Режим доступа: https://habr.com/ru/post/537682/- Дата доступа: 10.12.2022.

5 ГОСТ 7.12-77 Реферат и аннотация

6 ГОСТ 2.108-68 ЕСКД Спецификация

7 ГОСТ 2.113-75 ЕСКД Групповые и базовые конструкторские документы

8 ГОСТ 21.101-93 Основные требования к рабочей документации