

Министерство науки и высшего образования Российской Федерации
федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕТРОЗАВОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ПетрГУ)

РАЗРАБОТКА КОМПЬЮТЕРНОЙ ИГРЫ НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ C++
С ИСПОЛЬЗОВАНИЕМ БИБЛИОТЕКИ SFML

курсовая работа

Автор работы:
студентка группы
21412
_____ Сулакова С.В.
«___» _____ 2022 г.

Научный
руководитель: канд.
физ-мат. наук,
доцент
_____ Бульба А.В.
«___» _____ 2022 г.

Петрозаводск, 2022 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	3
ПРОЦЕСС РАЗРАБОТКИ	4
ИСТОРИЯ КОММИТОВ	5
ЗАКЛЮЧЕНИЕ.....	7
ПРИЛОЖЕНИЕ А	8
ПРИЛОЖЕНИЕ Б	9
ПРИЛОЖЕНИЕ В.....	20

ВВЕДЕНИЕ

Целью выполнения данной курсовой работы является реализация простой 2D-игры на языке программирования C++ с использованием библиотеки SFML (Simple and Fast Multimedia Library — простая и быстрая мультимедийная библиотека).

Было решено написать современную версию игры “Tank 1990”.

В программе должен быть реализован класс предок, класс-игрок, класс-враг, класс-пуля. В игре объект-игрок в одном экземпляре, объекты-враги создаются и уничтожаются по ходу игры, объекты-пули создаются и уничтожаются по ходу игры. Пересечение участников игры постоянно проверяется возможностями SFML.

При разработке программы использовано наследование, контейнеры, итераторы, отдельная компиляция. В отчете присутствуют UML-диаграмма классов (class diagram) и диаграмма вариантов использования (use case diagram) разработанной программы. Промежуточные результаты работы команды сохранялись на GitHub.

ПРОЦЕСС РАЗРАБОТКИ

Процесс разработки нельзя четко ограничить только одной сферой, так как все участники активно работали над всеми пунктами, предлагали и обсуждали идеи, вносили правки.

Однако, если выделять конкретные пункты, то можно отметить:

- Работу над файлами *interaction.cpp* и *interaction.h* (над визуальной частью проекта)
 - *interaction.h* – содержит описание класса Engine. Предназначен для описания необходимых атрибутов и методов, благодаря которым все остальные классы смогут взаимодействовать друг с другом. По сути, содержит в себе основную логику игры. Кроме того, содержит описания игровых меню
 - *interaction.cpp* – содержит реализацию методов класса Engine.
- Разработку руководства пользователя (где были описаны назначение игры, требования к ее программному и аппаратному обеспечению, работа игры, а также решение технических проблем)

Оба этих пункта велись в паре: в Приложении А и Б выделана та часть кода, которая была написана лично мной, а руководство пользователя можно посмотреть в репозитории проекта в папке reports (т.к. в основном отчете приложена его сокращенная версия, которую я прикладываю в Приложении В).

ИСТОРИЯ КОММИТОВ

Ниже приведена история коммитов в хронологическом порядке:

commit 37591763b140d18f281a2861c52cdefd4278e3ba

Author: Sentyabrina Sulakova <sulakova.sentyabrina@yandex.ru>

Date: Sun Dec 18 15:07:56 2022 +0300

Добавила изображения, в которые будем подгружать файлы, тестуру и спрайт для карты

commit 4930243af4866ffbf7295435b859326a4770691d

Author: Santa1905 <sulakova.sentyabrina@yandex.ru>

Date: Sun Dec 18 17:53:33 2022 +0300

Добавила отрисовку карты и танка

commit 9167ad6945294ab845e9652de4c7ca90efcb49ee

Author: Santa1905 <sulakova.sentyabrina@yandex.ru>

Date: Sun Dec 18 21:35:34 2022 +0300

Добавила меню в начало игры

commit 1bdce3d9d4177af8c95ba5e41a7f7e206a9854ef

Author: Santa1905 <71601224+Santa1905@users.noreply.github.com>

Date: Sun Dec 18 21:43:39 2022 +0300

Delete .DS_Store

случайно добавила временный файл, извините!

commit b3824c140adc471e92fb3fe731874c8c71887dca

Author: Santa1905 <sulakova.sentyabrina@yandex.ru>

Date: Sun Dec 18 23:03:03 2022 +0300

Украсила меню

commit 5e244aeaf9358b2a60db06a540a8a9513ce9d792

Author: Santa1905 <sulakova.sentyabrina@yandex.ru>

Date: Mon Dec 19 16:11:41 2022 +0300

Подгрузила restart для меню при нажати на ESC

commit aef2db29ad747198da07e2fc7d04449df39f6b6c

Author: Santa1905 <sulakova.sentyabrina@yandex.ru>

Date: Mon Dec 19 16:32:58 2022 +0300

Подгрузила наш main + добавила картинку

commit ed85e8429160b04c01ab5204a625ec84939f3a0c

Author: Santa1905 <sulakova.sentyabrina@yandex.ru>

Date: Mon Dec 19 17:16:39 2022 +0300

Добавила индикаторы жизни и пуль на экране (слева наверху)

commit 377239779f31ce844b72061ee31d85f89995a238

Author: Santa1905 <sulakova.sentyabrina@yandex.ru>

Date: Mon Dec 19 23:09:42 2022 +0300

Добавила счет игрока

commit 1774a5384b479e89d5d583a1edfa65f8b37bc9aa

Author: Santa1905 <sulakova.sentyabrina@yandex.ru>

Date: Mon Dec 19 23:24:03 2022 +0300

Добавили более ламповый шрифт (CyrilicOld)

commit 2b63500c6cffa45c3668d00b55f0bf9d3544121b

Merge: 24e4777 1774a53

Author: Santa1905 <sulakova.sentyabrina@yandex.ru>

Date: Mon Dec 19 23:43:59 2022 +0300

Влили ветку interaction в ветку develop

commit 5d70b9ebb55b126266fbd86919acbb0b53560f3c

Author: Santa1905 <sulakova.sentyabrina@yandex.ru>

Date: Tue Dec 20 19:51:04 2022 +0300

Пофиксили проблему с паузой (зависала при нажатии)

ЗАКЛЮЧЕНИЕ

В результате выполнения данной курсовой работы создана игра на языке программирования C++, которую можно считать современной версией игры “Tank 1990”.

Достигнуто понимание стадий разработки, создания и написания руководства пользователя игры. В процессе работы применялась система контроля версий Git. Реализованы все прецеденты. Сбои/зависания программы во время тестирования и использования не наблюдаются. Программа написана с учетом принципа отдельной компиляции. Реализовано освобождение памяти для класса Engine (в нем освобождается вся используемая память). Нет неиспользуемых методов, атрибутов и переменных. Все конструкции в программе используются для работы с классами. Нет конструкций, без которых можно было обойтись. К основному отчету приложены диаграммы классов и вариантов использования. Поставленная цель достигнута.

ПРИЛОЖЕНИЕ А

КОД ПРОГРАММЫ

Файл *interaction.h*

```
#pragma once
#include "Enemy.h" //файл с врагами
#include "Player.h" //файл с игроком
#include "Bullet.h" //Файл с пулей

class Engine {
public:
    Engine(); //конструктор
    ~Engine(); //деструктор
    int play(int number); //метод игры

private:
    Player* Hero; //объект игры

    sf::Font font; // шрифт для текста
    sf::Text text; // текст

    std::vector<Bullet> bullets; // вектор пуль
    std::vector<Enemy> enemies; // вектор врагов
    sf::Clock clock; // аппаратный таймер, время процесса игры
    float time; // основное время, отвечает за скорость игры
    sf::Image map_image; //объект изображения для карты
    sf::Image allImage; //Все изображения, которые используются поверх
карты
    sf::Sprite s_map; //спрайт для карты
    sf::Sprite Health; //для здоровья
    sf::Sprite GunDamage; //для уровня заряда пушки
    sf::Texture map; //текстура карты
    bool GameOver; //триггер для окончания игры
    int MainMenu(sf::RenderWindow& target); //меню при старте
    int RestartMenu(sf::RenderWindow& target); //меню внутри игры
};
```


ПРИЛОЖЕНИЕ Б

КОД ПРОГРАММЫ

Файл *interaction.cpp*

```
#include "interaction.h"           //заголовочный файл
#include "map.h"                   //файл с заранее прописанной картой

Engine::Engine()
{
    ap_image.loadFromFile("images/map.png");           //загружаем файл для карты
    map.loadFromImage(map_image);                       //заряжаем текстуру
картинкой
    s_map.setTexture(map);                             //заливаем текстуру
спрайтом
    allImage.loadFromFile("images/robots.png");        //загрузили изображения
объектов
    clock.restart();                                   //перезагружает время
    Hero = new Player(allImage, 500, 500, 70, 80, "hero"); //Создаем объект
героя
    GameOver = false;                                //игра "не окончена"

    font.loadFromFile("images\\CyrilicOld.ttf");        //шрифт загрузили
    text.setFont(font);
    text.setCharacterSize(24);
    text.setStyle(sf::Text::Bold);
    text.setPosition(1000, 5);
}

Engine::~Engine()
{
    delete Hero; //удаляет объект игрок
    enemies.clear(); //удаляет вектора врагов
    bullets.clear(); //удаляет вектора пули
    //очищаем память
}

int Engine::play(int number)
{
    sf::RenderWindow window(sf::VideoMode(1280, 800), "Game",
sf::Style::Fullscreen); //окно сформировали, сделали на полный экран
    sf::Cursor cursor; //устанавливаем курсор-крестик (типа прицел)
    if (cursor.loadFromSystem(sf::Cursor::Cross))
        window.setMouseCursor(cursor);
```

```

if (number != 1 && !MainMenu(window))//возвращает 0, если нажат выход
{
    return 0;
}

float timerspaun = 0;//таймер для появления врагов
float gunTimer = 0;//для контроля выстрелов гг
float spaunlvl = 5000;//время нужное спауна
float timerLVUp = 0;//повышаем уровень, со временем
sf::Image healthImg;
healthImg.loadFromFile("images/Health.png");
sf::Texture healthTexture;
healthTexture.loadFromImage(healthImg);
healthTexture.setRepeated(true);//Чтоб не рисовать - повторяем один и тот же
texture в спрайте
Health.setTexture(healthTexture);

sf::Image gunDamageImg;
gunDamageImg.loadFromFile("images/Bullet.png");//загрузили изображения пуля
sf::Texture gunDamageTexture;
gunDamageTexture.loadFromImage(gunDamageImg);
gunDamageTexture.setRepeated(true);
GunDamage.setTexture(gunDamageTexture);
Health.setTextureRect(sf::IntRect(0, 0, 32, 32));//Поставили картинку здоровья
Health.setScale(0.5, 0.5);
Health.setPosition(10, 10);
GunDamage.setTextureRect(sf::IntRect(0, 0, 70, 348));//Поставили картинку заряда
пушки
GunDamage.setScale(0.1, 0.1);
GunDamage.setPosition(10, 30);
while (window.isOpen() && !GameOver)
{
    //Пока окно открыто и игра не закончена
    sf::Event event;//если нажата клавиша
    while (window.pollEvent(event))
    {
        if (sf::Keyboard::isKeyPressed(sf::Keyboard::Escape))
        {
            //нажали esc = открыли меню, игра на паузе
            switch (RestartMenu(window))
            {
                case 0://если 0, то закрывает игру
                window.close();
                return 0;
                case 1://если 1, то запуск игры
                window.close();
                return 1;
                default:
            }
        }
    }
}

```

```

clock.restart(); //пофиксили баг - теперь работает пауза без
КОСЯКОВ
break;
}
}
}
time = clock.getElapsedTime().asMicroseconds(); //дать прошедшее время в
микросекундах
clock.restart(); //перезагружает время
time = time / 800; //скорость игры
timerspaun += time;
gunTimer += time;
timerLVLup += time;
if (timerLVLup > 30000)
{
    //30 сек и повысили сложность + появился босс
    Enemy* anotherEnemy = new Enemy(allImage, 200, 200, 120, 90,
"BOSSbot", TileMapMy);
    enemies.push_back(*anotherEnemy); //указатель, чтоб подгрузить
картинку
    spawnlvl -= 500;
    if (spawnlvl < 2000) //максимальная сложность, враг появляется раз в 2
секунды
    {
        spawnlvl = 2000;
    }
    timerLVLup = 0;
    timerspaun = 0;
}
//
if (timerspaun > spawnlvl && enemies.size() < 10)
{
    //пусть будет не больше 10 врагов
    Enemy* anotherEnemy = new Enemy(allImage, 200, 200, 45, 65, "flybot",
TileMapMy);
    enemies.push_back(*anotherEnemy); //указатель, чтобы подгрузить
картинку
    timerspaun = 0;
}

Hero->update(time, TileMapMy, event); //Герой сделал свой ход
Health.setTextureRect(sf::IntRect(0, 0, 32 * (Hero->Gethealth() / 20), 32));
if (gunTimer > 1000) //максимальный урон = большая полоска
{
    GunDamage.setTextureRect(sf::IntRect(0, 0, 70 * 10, 348));
}

```

```

    }
    else//чем меньше полоска, тем меньше урон
    {
        GunDamage.setTextureRect(sf::IntRect(0, 0, 70 * (gunTimer / 100), 348));
    }

    if (gunTimer > 150 && sf::Keyboard::isKeyPressed(sf::Keyboard::Space))
    {
        int damage = gunTimer / 10;
        if (damage > 100)
        {
            damage = 100;
        }
        if (damage > 20)
        {
            sf::Vector2f HeroXY = Hero->GetgunXY();//если пуля большая, то
выстреливает из дула
            Bullet* anotherBullet = new Bullet(allImage, HeroXY.x, HeroXY.y,
25, 25, Hero->GetRotation(), "HeroBullet", damage);
            bullets.push_back(*anotherBullet);//указатель, чтоб подгрузить
картинку
        }
        else
        {
            sf::Vector2f HeroXY = Hero->GetXY();//если пулька маленькая, то
выстреливает из центра героя, т.к. иначе проблема с хитбоксами
            Bullet* anotherBullet = new Bullet(allImage, HeroXY.x, HeroXY.y,
10, 10, Hero->GetRotation(), "HeroBullet", damage);
            bullets.push_back(*anotherBullet);//указатель, чтоб подгрузить
картинку
        }
        gunTimer = 0;
    }
    std::vector<Enemy>::iterator iterEnemies = enemies.begin();//итераторы для
врагов в начало + создаем их
    std::vector<Bullet>::iterator iterBullet = bullets.begin();//итераторы для пуль в
начало + создаем их
    while (iterBullet != bullets.end())
    {
        if (iterBullet->isAlive())
        {
            iterBullet->update(time, TileMapMy);
            ++iterBullet;//проходим по каждому объекту в векторе
        }
        else

```

```

        {
            iterBullet = bullets.erase(iterBullet); //стереть из списка
        }
    }
    while (iterEnemies != enemies.end())
    {
        if (iterEnemies->isAlive())
        {
            sf::Vector2f BufXYHero = Hero->GetX();
            iterEnemies->SetAim(BufXYHero); //цель - герой
            iterEnemies->update(time); //враг сходил
            if (iterEnemies->GetName() == "BOSSbot" && iterEnemies-
>GetBOSSdamagetime() > 2000) //босс стреляет раз в 2 секунды
            {
                sf::Vector2f BufXYEnemy = iterEnemies->GetX(); //берем
координаты
                float rotation = atan2(BufXYHero.y -
BufXYEnemy.y, BufXYHero.x - BufXYEnemy.x) * 180 / 3.14159265 + 90;
                //вычисляем угол поворота
                Bullet* anotherBullet = new Bullet(allImage, BufXYEnemy.x,
BufXYEnemy.y, 30, 30, rotation, "BossBullet", 25);
                bullets.push_back(*anotherBullet); //создаем и добавляем в
вектор пулю
                iterEnemies->recetBOSSdamagetime();
            }

            iterBullet = bullets.begin();
            while (iterBullet != bullets.end())
            { //Находим пересечение хитбоксов пули и врагов с героем, далее
уничтожается пуля, а остальные получают урон
                if (iterBullet->isAlive())
                {
                    if (iterBullet->GetName() != "BossBullet" && iterBullet-
>GetRect().intersects(iterEnemies->GetRect()))
                    {
                        iterEnemies->struck(iterBullet->GetDamage());
                    }
                    if (iterBullet->GetName() != "HeroBullet" && iterBullet-
>GetRect().intersects(Hero->GetRect()))
                    {
                        Hero->struck(iterBullet->GetDamage());
                    }
                    ++iterBullet;
                }
            }
        }
    }

```

```

        {
            iterBullet = bullets.erase(iterBullet); //стереть из списка
        }
    }

    if (Hero->GetRect().intersects(iterEnemies->GetRect()))
    { //При столкновении героя и врагов
        if (iterEnemies->GetName() == "BOSSbot")
        {
            if (iterEnemies->GetStatus() != "anikilled")
            {
                if (iterEnemies->GetBOSSdamagetime() > 500)
                {
                    //При пересечении хитбоксов героя и босса,
                    босс бьет раз в 500 мкс

                    Hero->struck(20);
                    iterEnemies->recetBOSSdamagetime();
                }
            }
        }
        else
        {
            iterEnemies->struck(100); //Другие же получают урон,
            несовместимый с жизнью

            if (iterEnemies->GetStatus() != "anikilled")
            {
                Hero->struck(20);
            }
        }
    }
    ++iterEnemies;
}
else
{
    if (iterEnemies->GetName() == "BOSSbot")
    {
        Hero->Addscore(500); //Босс убит - получили 500 очков
    }
    else if (iterEnemies->GetName() == "flybot")
    {
        Hero->Addscore(100); //flybot убит - получили 100 очков
    }
    iterEnemies = enemies.erase(iterEnemies); //стереть из списка
}

```

```

    }

    }
    GameOver = !Hero->isAlive();
    //drawing ->
    window.clear();
    //////////////////////////////////Рисуем карту////////////////////////////////////
    for (int i = 0; i < HEIGHT_MAP; i++)
        for (int j = 0; j < WIDTH_MAP; j++)
        {
            if (TileMapMy[i][j] == ' ') s_map.setTextureRect(sf::IntRect(0, 0, 32,
32)); //если встретили символ пробел, то рисуем 1й квадратик
            else if (TileMapMy[i][j] == '0')
s_map.setTextureRect(sf::IntRect(32, 0, 32, 32)); //если встретили символ 0, то рисуем 2й
квадратик
            else if (TileMapMy[i][j] == 'b') s_map.setTextureRect(sf::IntRect(64,
0, 32, 32)); //если встретили символ b, то рисуем 3й квадратик
            else if (TileMapMy[i][j] == 'p') s_map.setTextureRect(sf::IntRect(96,
0, 32, 32)); //если встретили символ p, то рисуем 4й квадратик
            else if (TileMapMy[i][j] == 'h')
s_map.setTextureRect(sf::IntRect(128, 0, 32, 32)); //если встретили символ h, то рисуем
5й квадратик
            else s_map.setTextureRect(sf::IntRect(160, 0, 32, 32));
            s_map.setPosition(j * 32, i * 32); //по сути раскидывает квадратики,
превращая в карту.
            //то есть задает каждому из них позицию.
            window.draw(s_map); //рисуем квадратики на экран
        }
    Hero->draw(window); //рисуются герой-танк
    iterEnemies = enemies.begin();
    while (iterEnemies != enemies.end()) {
        if (iterEnemies->GetName() != "BOSSbot")
        {
            iterEnemies->draw(window);
            //босс выше всех летает
        }
        ++iterEnemies;
    }
    iterBullet = bullets.begin();
    while (iterBullet != bullets.end()) {
        window.draw(iterBullet->sprite);
        ++iterBullet;
    }
    iterEnemies = enemies.begin();
    while (iterEnemies != enemies.end()) {

```

```

        if (iterEnemies->GetName() == "BOSSbot")
        {
            iterEnemies->draw(window);
        }
        ++iterEnemies;
    }

    window.draw(GunDamage);
    window.draw(Health);
    text.setString("Score: " + std::to_string(Hero->Getscore())); //преобразовали
цифру в текст и показали
    window.draw(text);

    window.display();
}
if (RestartMenu(window))
{
    window.close();
    return 1;
}

window.close();
return 0;
}

```

```

int Engine::RestartMenu(sf::RenderWindow& target)
{
    sf::Texture menuTexturePlay, menuTextureQuit, menuTextureRestart;
    menuTexturePlay.loadFromFile("images/Play.png");
    menuTextureQuit.loadFromFile("images/Quit.png");
    menuTextureRestart.loadFromFile("images/Restart.png");
    sf::Sprite menuPlay(menuTexturePlay), menuQuit(menuTextureQuit),
menuRestart(menuTextureRestart);
    bool isMenu = 1;
    int menuNum = 0;
    if (!GameOver)
    {
        menuRestart.setPosition(WIDTH_MAP * 16 - 155, HEIGHT_MAP * 16 - 120);
        menuPlay.setPosition(WIDTH_MAP * 16 - 155, HEIGHT_MAP * 16);
    }
    else
    {
        menuPlay.setPosition(WIDTH_MAP * 16 - 155, HEIGHT_MAP * 16 - 120);
        menuRestart.setPosition(WIDTH_MAP * 16 - 155, HEIGHT_MAP * 16);
    }
    menuQuit.setPosition(WIDTH_MAP * 16 - 155, HEIGHT_MAP * 16 + 120);
}

```



```

//Расставили кнопки
//////////МЕНЮ//////////
while (isMenu)
{
    menuRestart.setColor(sf::Color::White);
    menuPlay.setColor(sf::Color::White);
    menuQuit.setColor(sf::Color::White);
    menuNum = 0;
    sf::RectangleShape rectangle(sf::Vector2f(20, 20));
    rectangle.setSize(sf::Vector2f(WIDTH_MAP * 32 - 40, HEIGHT_MAP * 32 -
40));
    if (!GameOver)
    {
        rectangle.setFillColor(sf::Color(255, 228, 200, 1));
        rectangle.setPosition(sf::Vector2f(20, 20));
        if (sf::IntRect(WIDTH_MAP * 16 - 155, HEIGHT_MAP * 16 - 120, 310,
110).contains(sf::Mouse::getPosition()))
        {
            menuRestart.setColor(sf::Color::Blue); menuNum = 1; // 1 = рестарт
        }
        if (sf::IntRect(WIDTH_MAP * 16 - 155, HEIGHT_MAP * 16, 310,
110).contains(sf::Mouse::getPosition()))
        {
            menuPlay.setColor(sf::Color::Blue); menuNum = 2; // 2 = играть
        }
        if (sf::IntRect(WIDTH_MAP * 16 - 155, HEIGHT_MAP * 16 + 120, 310,
110).contains(sf::Mouse::getPosition()))
        {
            menuQuit.setColor(sf::Color::Blue); menuNum = 0; // 0 = выйти
        }
    }
    else
    {
        //если игра окончена, то саму игру надо закрасить
        target.clear(sf::Color(255, 228, 200));
        text.setCharacterSize(64);
        text.setPosition(WIDTH_MAP * 16 - 155, HEIGHT_MAP * 16 - 120);
        if (sf::IntRect(WIDTH_MAP * 16 - 155, HEIGHT_MAP * 16, 310,
110).contains(sf::Mouse::getPosition()))
        {
            menuRestart.setColor(sf::Color::Blue); menuNum = 1;
        }
        if (sf::IntRect(WIDTH_MAP * 16 - 155, HEIGHT_MAP * 16 + 120, 310,
110).contains(sf::Mouse::getPosition()))
        {

```

```

        menuQuit.setColor(sf::Color::Blue); menuNum = 0;
    }
}

if (sf::Mouse::isButtonPressed(sf::Mouse::Left))
{

    return menuNum;

}
if (!GameOver)
{
    target.draw(rectangle);
    target.draw(menuPlay);
}
target.draw(menuRestart);
target.draw(menuQuit);

text.setString("Score: " + std::to_string(Hero->Getscore()));
target.draw(text);

target.display();
}
return 0;
}

int Engine::MainMenu(sf::RenderWindow& target)
{
    {
        sf::Texture menuTexturePlay, menuTextureQuit, menuBackground;
        menuTexturePlay.loadFromFile("images/Play.png");
        menuTextureQuit.loadFromFile("images/Quit.png");
        menuBackground.loadFromFile("images/jogaGame.png");
        sf::Sprite menuPlay(menuTexturePlay), menuQuit(menuTextureQuit),
menuBg(menuBackground);
        bool isMenu = 1;
        int menuNum = 0;
        menuPlay.setPosition(100, 200);
        menuQuit.setPosition(100, 500);
        menuBg.setPosition(0, 0);
        ////////////////////////////////////MEHIO////////////////////////////////////
        while (isMenu)
        {
            menuPlay.setColor(sf::Color::White);
            menuQuit.setColor(sf::Color::White);

```

```
menuNum = 0;  
target.clear();
```

```
    if (sf::IntRect(100, 200, 310, 110).contains(sf::Mouse::getPosition()))  
{ menuPlay.setColor(sf::Color::Blue); menuNum = 1; }  
    if (sf::IntRect(100, 500, 310, 110).contains(sf::Mouse::getPosition()))  
{ menuQuit.setColor(sf::Color::Blue); menuNum = 2; }
```

```
    if (sf::Mouse::isButtonPressed(sf::Mouse::Left))  
    {  
        if (menuNum == 1)//если нажали кнопку играть, то запускаем  
игру  
        {  
            isMenu = false;  
            return 1;  
        }  
        if (menuNum == 2)//если нажали кнопку выйти, то закрыли  
игру  
        {  
            isMenu = false;  
            return 0;  
        }  
    }
```

```
    }  
    //target = window  
    target.draw(menuBg);  
    target.draw(menuPlay);  
    target.draw(menuQuit);  
    target.display();  
}  
return 0;  
}
```

ПРИЛОЖЕНИЕ В

2.7. Руководство пользователя

После запуска игры на экране появляется главное меню. Для того, чтобы начать сражение, нажмите кнопку PLAY. Для выхода из игры нажмите кнопку Quit.



Рисунок 3 – главное меню

После начала сражения перед игроком появляется карта местности. В правом верхнем углу отображается количество набранных очков, в левом – количество жизней и уровень заряда пушки.

Клавиши “w”, “a”, “s” и “d” перемещают танк вверх, влево, вниз и вправо. Прицел наводится при помощи курсора. Выстрел осуществляется по нажатию клавиши “Space”. Перед выстрелом требуется перезарядка. Если перезарядка не успела завершиться



полностью, то пуля будет меньшего размера. Встречаясь с препятствиями  , пуля дальше не летит, танк - не едет. Враги могут перелетать через ящики, но не могут выйти за края игрового поля.




Рисунок 4 – персонаж-Танк (один из видов)



Рисунок 5 – пуля (один из спрайтов)

Враги бывают двух типов: камикадзе-flybot и стреляющий пилами BOSSbot. Они

проявляются в 4-х спаунах  на карте. Враг-flybot летит прямо в вас, а враг-BOSSbot вращается и перемещается между спаунами.

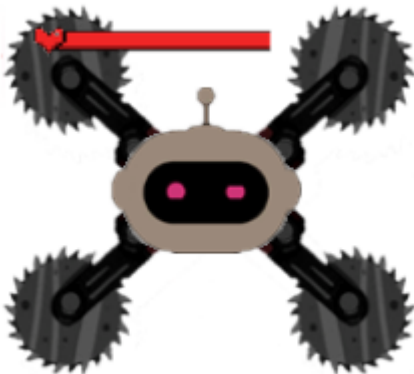


Рисунок 6 – BOSSbot



Рисунок 7 – flybot

Избегайте столкновений с врагами, иначе уровень здоровья будет снижаться (урон от flybot-а – 20 единиц, от BOSSbot-а – 40 единиц каждую секунду). Кроме того, опасайтесь оружия BOSSbota – летающих пил, которые он раз в 2 секунды запускает прямо в вас (урон – 25 единиц здоровья).



Рисунок 8 – оружие BOSSbot-a


Повысить уровень здоровья можно, встав на точку хила  в правой части карты. Пример анимации нанесения ущерб Танку:

Рисунок 9 – Танк получил урон



Пример анимации уничтожения flybot-a:



Рисунок 10 – Танк сделал выстрел, попал в цель, враг уничтожен Далее приведено несколько скриншотов экрана во время игры.

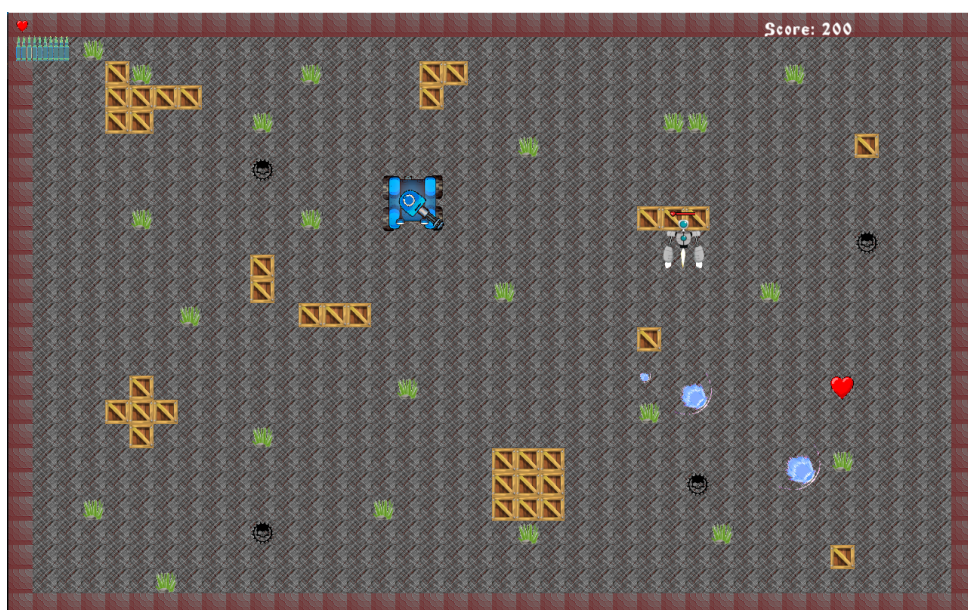


Рисунок 11 – процесс игры



Рисунок 12 – процесс игры

В процессе игры вы можете сделать паузу, нажав клавишу “Escape”. И далее выбрать одно из действий: начать сражение заново (кнопка Restart), продолжить текущий бой (кнопка PLAY) или выйти из игры (кнопка Quit).



Рисунок 13 – меню паузы

В случае проигрыша на экране появится набранное вами количество очков за бой. Кроме того, будет предложено начать новый бой (кнопка Restart) или выйти из игры (кнопка Quit).

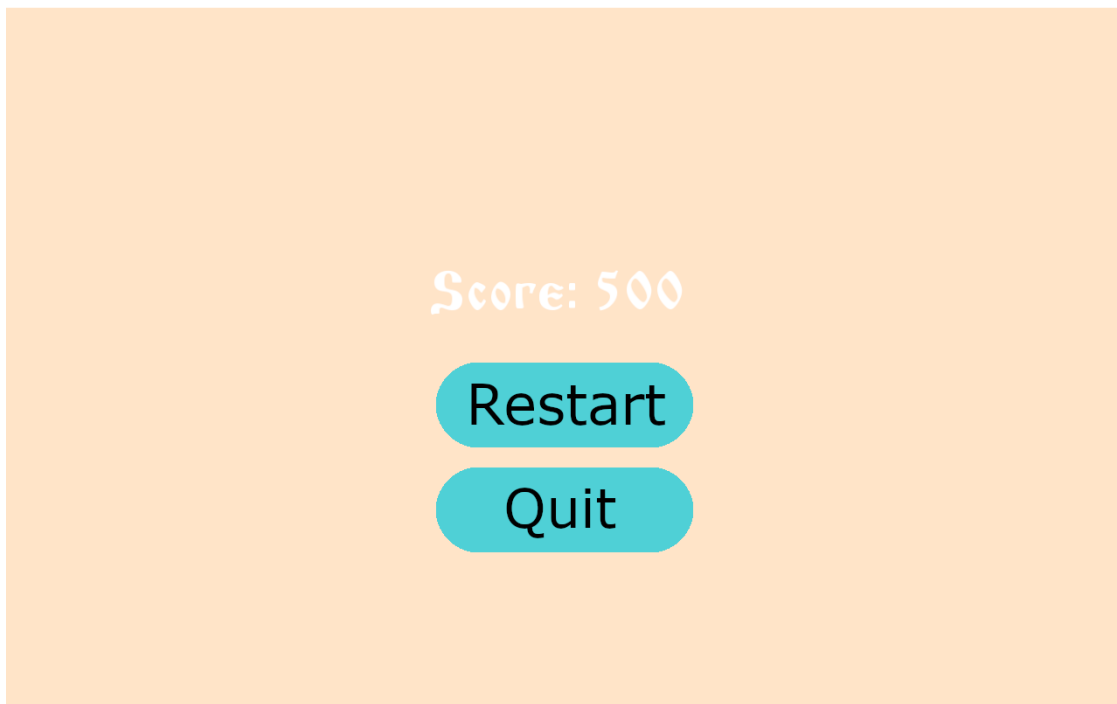


Рисунок 14 – меню после смерти игрока