Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Петрозаводский государственный университет»

Физико-технический институт

Кафедра информационно-измерительных систем и физической электроники

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

курсовая работа

Автор работы:

студенты группы 21412

А. О. Лайкачев

Принял:

канд. физ.-мат. наук, доцент

А. В. Бульба

Петрозаводск 2022

Содержание

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, с помощью объекты-пули создаются и уничтожаются по ходу игры. Пересечение участников игры постоянно проверяется возможностями SFML.

При разработке программы использовано наследование, контейнеры, итераторы, раздельная компиляция. В отчете присутствуют сценарии вариантов использования разработанной программы и список существительных, классы диаграмм. Промежуточные результаты работы команды сохранялись на GitHub.

История коммитов на GitHab

Адрес проекта на GitHab: https://github.com/DyachenkoAnna/game.git

Ниже приведен список моих коммитов:

Author: aleksey laikachev <super.games29@yandex.ru>
Date: Mon Dec 19 22:52:20 2022 +0300
Добавил комментарии в файлы interaction.h и interaction.cpp

Author: aleksey laikachev <super.games29@yandex.ru>
Date: Mon Dec 19 22:24:48 2022 +0300
Враги начали получать урон, также Flybot убивается об героя, босс почему то не стреляет и не наносит урон при столкновении

Author: aleksey laikachev <super.games29@yandex.ru>
Date: Mon Dec 19 22:12:11 2022 +0300
подгрузил обновленные файлы

Author: aleksey laikachev <super.games29@yandex.ru>
Date: Mon Dec 19 17:37:42 2022 +0300
Враги начали передвигаться и преследовать игрока, но по прежнему не получают урон

Author: aleksey laikachev <super.games29@yandex.ru>
Date: Mon Dec 19 16:57:17 2022 +0300
Добавил отображение пули,но пока что урон не наносится

Author: aleksey laikachev <super.games29@yandex.ru>
Date: Mon Dec 19 16:47:01 2022 +0300
в файле interaction.h, подключил Bullet.h и объявил ее вектор

Author: aleksey laikachev <super.games29@yandex.ru>
Date: Mon Dec 19 16:41:39 2022 +0300
Обновил main, теперь при нажатие на кнопку рестарт, игра перезагружается

Author: aleksey laikachev <super.games29@yandex.ru>
Date: Mon Dec 19 15:56:11 2022 +0300
Подгрузил картинку пули

Author: aleksey laikachev <super.games29@yandex.ru>
Date: Mon Dec 19 15:45:53 2022 +0300
подгрузил файлы с Bullet

Author: aleksey laikachev <super.games29@yandex.ru>
Date: Sun Dec 18 23:44:41 2022 +0300
Сделал появление противников(пока что просто стоят на месте), а также добавил паузу в игру(нужно будет сделать кнопку рестарт)

Author: aleksey laikachev <super.games29@yandex.ru>
Date: Sun Dec 18 23:20:39 2022 +0300
Добавил в файл interaction.h переменные времени и меню паузы

Author: aleksey laikachev <super.games29@yandex.ru>
Date: Sun Dec 18 22:58:02 2022 +0300
Файл с возможным дальнейшим развитием интерфейса

Author: aleksey laikachev <super.games29@yandex.ru>
Date: Sun Dec 18 17:43:51 2022 +0300
Добавил файл interaction.cpp - файл в котором будет отрисовываться игра и ее игровая логика

Author: aleksey laikachev <super.games29@yandex.ru>
Date: Sun Dec 18 17:24:55 2022 +0300
подгрузили файлы с develop

Author: aleksey laikachev <super.games29@yandex.ru>
Date: Sun Dec 18 14:28:29 2022 +0300
Добавил файл interaction.h - файл объявляющий класс Engine

Перечень индивидуальных работ

1. Процесс разработки

В течение всего проекта принимал участие в обсуждении идей. По решению Team Leader, совместно с Сулаковой С. В. работал над классом Play. Данный класс связывает все остальные классы, реализованные в рамках разработки игры, и содержит в себе методы, реализующие внутриигровую логику и взаимодействие пользователя с системой.

2.3. Сценарии вариантов использования

Игра. При нажатии кнопки PLAY в главном меню после запуска игры на экране загружается карта игры, по которой будет перемещаться персонаж игрока (танк).

Движение вверх. Перемещение танка вверх осуществляется по нажатию клавиши w на клавиатуре.

Движение вниз. Перемещение танка вниз осуществляется по нажатию клавиши s на клавиатуре.

Движение влево. Перемещение танка влево осуществляется по нажатию клавиши a на клавиатуре.

Движение вправо. Перемещение танка вправо осуществляется по нажатию клавиши d на клавиатуре.

Навести прицел. Наводка прицела осуществляется при помощи курсора-крестика.

Выстрелить. Выстрел из дула танка осуществляется по нажатию клавиши “Space’’ на клавиатуре. После выстрела требуется восстановление уровня урона пуль (время – до 1 секунды). Размер пули зависит от уровня наносимого ею урона (20–100 ед. урона). Если пуля достигает врага, то у него снижается уровень здоровья. Враг умирает, если уровень его здоровья исчерпывается; игроку добавляются очки (+100 – за уничтоженного flybot-a, +500 – за BOSSbot-a). Если пуля сталкивается с препятствием, то дальше она лететь не может.

Столкновение с врагом. Если flybot сталкивается с танком, то он (враг) умирает, а у персонажа-танка снижается уровень здоровья. Если BOSSbot сталкивается с танком, то с ним (с врагом) ничего не происходит, а у персонажа-танка снижается уровень здоровья.

Пополнить жизни. Если персонаж-танк встанет на сердечко (точка хила), то уровень его здоровья пополнится через 3 секунды без движения.

Пауза. Приостановка игры осуществляется по нажатию клавиши Escape на клавиатуре. На экране в меню паузы появляется 3 кнопки: Restart, PLAY, Quit.

Перезапуск. При нажатии кнопки Restart в любом меню сражение начнется заново.

Продолжить. При нажатии кнопки PLAY в меню паузы сражение продолжится.

Выход. При нажатии кнопки Quit в любом меню программа завершает свою работу.

Проигрыш. В момент, когда уровень жизни персонажа-танка исчерпается, сражение будет окончено. На экране появится меню, в котором содержится информация о количестве набранных очков за сражение и 2 кнопки: Restart и Quit.

2.4. Список существительных, классы программы

Список всех существительных из вариантов использования:

1. Кнопка PLAY
2. Кнопка Restart
3. Кнопка Quit

4. Главное меню
5. Меню паузы
6. Меню
7. Игра
8. **Экран**
9. Карта
10. Персонаж
11. Игрок
12. **Танк**
13. Клавиша
14. Перемещение
15. Прицел
16. Выстрел
17. Курсор-крестик
18. Дуло
19. **Пуля**
20. flybot
21. BOSSbot
22. **Враг**
23. Уровень здоровья
24. Сердечко (Точка хила)
25. Жизни
26. Приостановка
27. Сражение
28. Работа программы
29. Очки

Внимательно изучив список существительных, в качестве кандидатов на роль классов было отобрано 4 существительных: Танк, Пуля, Враг, Экран.

Большинство из оставшихся существительных станут атрибутами классов, а многие глаголы – методами.

В ходе обсуждения было выявлено, что у классов Танк, Пуля и Враг достаточно общего, чтобы создать для них общий родительских класс.

Таким образом, получаем список классов:

1. Entity – базовый класс. Является родительским для классов Player, Bullet, Enemy
2. Player – тип данных персонажа-танка, наследуется от класса Entity.
3. Bullet – тип данных экземпляров пуль, наследуется от класса Entity.
4. Enemy – тип данных экземпляров врагов, наследуется от класса Entity.
5. interaction – главный класс для запуска игры. Этот класс определяет взаимодействие пользователя с ней.

В приложениях А, Б и В приведены тексты файлов, над которыми мой велась работа, однако, так как, в данных файлах велась разработка одновременно 2 людьми, для лучшей наглядности части своего кода я выделю зеленым фоном.

ЗАКЛЮЧЕНИЕ

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

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 раздельной 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. Поставленная цель достигнута.

Мной был получен опыт совместной разработкой крупного проекта. Работа в группе вызывала немало трудностей. Несмотря на то, что в процессе разработки принимало участие 6 человек, что должно было сказаться наилучшим образом на эффективности, ведь 6 голов больше, чем 1, однако не всегда лучше, в реальности это наложило свои сложности, с которыми нужно было справляться в процессе разработки. И тем не менее это был очень важный опыт, который продемонстрировал все трудности разработки в командной среде.

ПРИЛОЖЕНИЕ А

main.cpp

```
// version 1.0

#include <ctime>
#include "interaction.h"

/*
Здесь вход в программу.
используем для вектора тип sf::Vector2f
*/

int main()
{
	srand(unsigned(time(0)));
	int way = 2;//для определения действия при завершении игры
	while (way != 0)
	 {
		Engine* level = new Engine();//Создали объект = запустили игру
		way = level->play(way);//начинаем игру
		delete level;// Удалили объект (игру)
	 }

	return 0;
}
```

ПРИЛОЖЕНИЕ Б

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 исполузуются 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;//тригер для окончания игры
    int MainMenu(sf::RenderWindow& target);//меню при старте
    int RestartMenu(sf::RenderWindow& target);//меню внутри игры
};
```

ПРИЛОЖЕНИЕ В

interaction.cpp

```
#include "interaction.h"//заголовочный файл
#include "map.h"//файл с заранее прописанной картой
Engine::Engine()
{
	map_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 герой
	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 вратов
	float gunTimer = 0;//для контроля выстрелов гг
	float spaunlvl = 5000;//время нужное спауна
	float timerLVLup = 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);
	//fix release 1.0 ++
	//Health.setTextureRect(sf::IntRect(0, 0, 32, 32));//Поставили картинку здоровья
```

```
	Health.setTextureRect(sf::IntRect(0, 0, 33, 32));//Поставили картинку здоровья //картинка одного сердечка на самом деле 33*32
	//fix release 1.0 --
	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
			spaunlvl -= 500;
			if (spaunlvl < 2000)//максимальная сложность, враг появляется раз в 2 секунды
			{
				spaunlvl = 2000;
			}
			timerLVLup = 0;
			timerspaun = 0;
		}
		//
		if (timerspaun > spaunlvl && 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
		//fix release 1.0 ++
```

```
            //Health.setTextureRect(sf::IntRect(0, 0, 32 * (Hero->Gethealth() /
20), 32));
            Health.setTextureRect(sf::IntRect(0, 0, 33 * ((Hero-
>Gethealth())/20.0), 32)); //картинка одного сердечка на самом деле 33*32
            //fix release 1.0 ++
            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 им
            std::vector<Bullet>::iterator iterBullet = bullets.begin();//итераторы
для пуль в начало + создаем им
            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->GetXY();
                        iterEnemies->SetAim(BufXYHero);//цель - герой
                        iterEnemies->update(time);//враг сходил
```

```
                        if (iterEnemies->GetName() == "BOSSbot" && iterEnemies->GetBOSSdamagetimer() > 2000)//босс стреляет раз в 2 секунды
                        {
                            sf::Vector2f BufXYEnemy = iterEnemies->GetXY();//берем координаты
                            float rotation = 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->recetBOSSdamagetimer();
                        }

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

                        if (Hero->GetRect().intersects(iterEnemies->GetRect()))
                        {//При стокновении героя и врагов
                            if (iterEnemies->GetName() == "BOSSbot")
                            {
                                //может произойти так, что вам нанесётся урон от взрыва после смерти босса. это нормально)
                                if (iterEnemies->GetStatus() != "anikilled")
                                {
                                    if (iterEnemies->GetBOSSdamagetimer() > 500)
                                    {
                                        //При пересечении хитбоксов героя и босса, босс бьет раз в 500 мкс
                                        Hero->struck(20);
                                        iterEnemies->recetBOSSdamagetimer();
                                    }

                                }
                            }
                            else
                            {
                                iterEnemies->struck(100);//Другие же получают урон, несовместимый с жизнью
                                //fix release 1.0 ++
                                //if (iterEnemies->GetStatus() != "anikilled")
```

```
                                        //а тут учтём условие смерти бота, чтобы он не
мог нанести урон после своей смерти:
                                        if ((iterEnemies->GetStatus() != "anikilled")
&& (iterEnemies->GetStatus() != "killed"))
                                        //fix release 1.0 --
                                        {
                                              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);//рисуется 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);
      //Расставил 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);
            ////////////////////////////////МЕНЮ//////////////////
            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 закрытили
игру
                        {
                              isMenu = false;
                              return 0;
                        }

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