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:

студент группы 21412

Н. А. Сидоров

«21» декабря 2022 г.

Принял:

канд. физ.-мат. наук, доцент

А. В. Бульба

« » декабря 2022 г.

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, 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 UML-диаграмма классов (class diagram) и диаграмма вариантов использования (use case diagram) разработанной программы. Промежуточные результаты работы команды сохранялись на GitHub.

История коммитов на GitHab

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

Ниже приведён список моих коммитов:

Author: Nikita Sidorov <eternium.save@gmail.com>
Date: Wed Dec 21 02:28:31 2022 +0300
Исправление багов отображения сердечек игрока и полосок здоровья врагов + условия проверки статусов робота. Небольшая обработка картинок для улучшения визуального восприятия

Author: Nikita Sidorov <eternium.save@gmail.com>
Date: Tue Dec 20 01:31:49 2022 +0300
Поправил отображение и обработку снарядов босса, уменьшил скорость врагов

Author: Nikita Sidorov <eternium.save@gmail.com>
Date: Sun Dec 18 03:46:52 2022 +0300
Замерджили ветку enemy в develop

Author: Nikita Sidorov <eternium.save@gmail.com>
Date: Sun Dec 18 03:30:10 2022 +0300
Дописал метод animation для босса, добавил метод update

Author: Nikita Sidorov <eternium.save@gmail.com>
Date: Sun Dec 18 01:44:55 2022 +0300
Добавление методов setdirection и action. Враги движутся к цели

Author: Nikita Sidorov <eternium.save@gmail.com>
Date: Sun Dec 18 00:50:42 2022 +0300
Добавлен метод получения координат цели (для движения врагов)

Author: Nikita Sidorov <eternium.save@gmail.com>
Date: Sun Dec 18 00:31:42 2022 +0300
Дописал конструктор для босса

Author: Nikita Sidorov <eternium.save@gmail.com>
Date: Sat Dec 17 21:28:14 2022 +0300
Создан скелет и конструктор для класса врагов

Перечень индивидуальных работ

В течение проекта принимал участие в обсуждении идей. По решению Лидера команды (Дьяченко А. Е.), работал с ней над классом Enemy. Данный класс содержит в себе методы, описывающие поведение врагов и реализующие их взаимодействие с игроком.

На основании общих идей и предложений участников команды мною была составлена диаграмма классов.

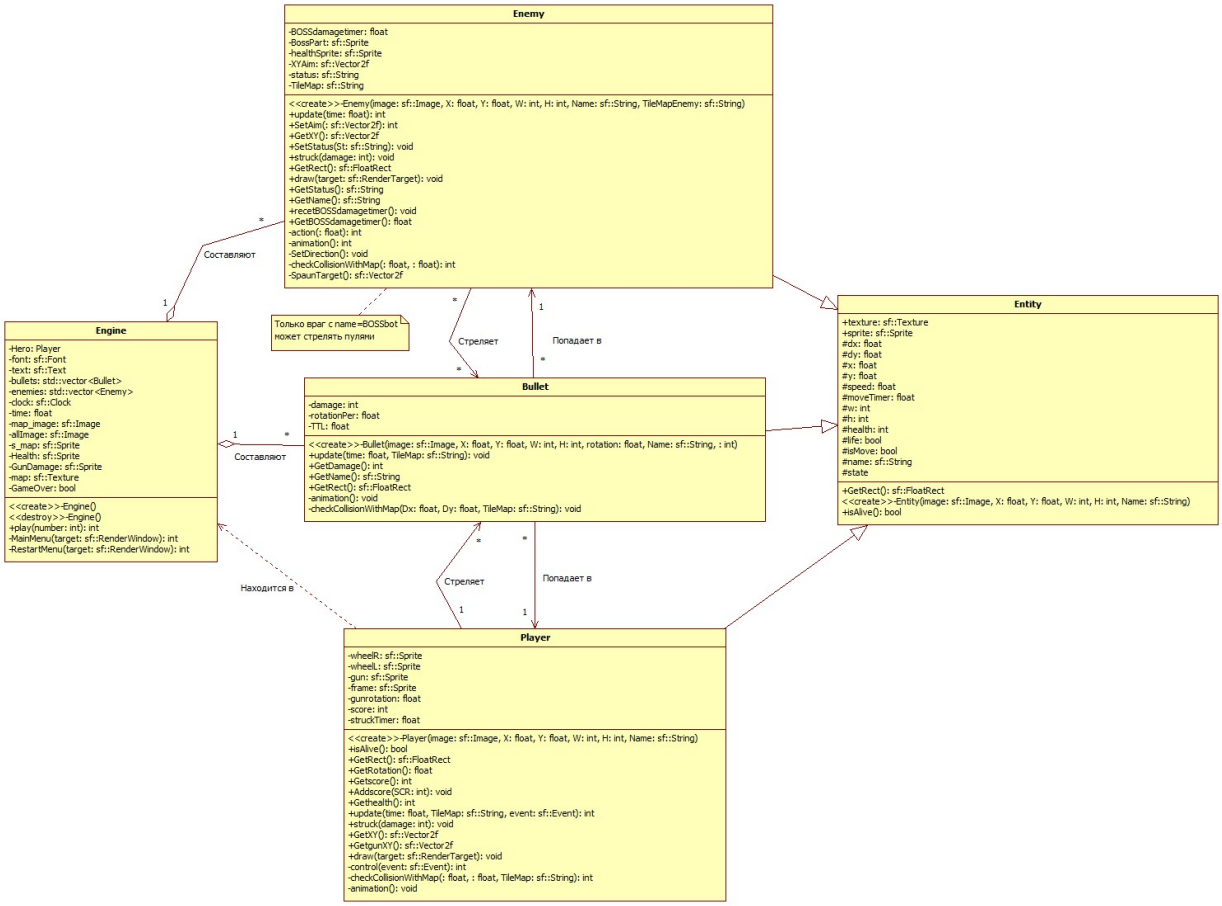

Рисунок 1 – диаграмма классов.

В рамках совместной работы над классом Enemy производился поиск изображений врагов и написание кода программы. Экземпляры данного класса представляют собой врагов двух видов: flybot и BOSSbot. Они появляются на спаунах, расположенных на карте. BOSSbot перемещается между спаунами, flybot всегда летит в сторону персонажа-танка. BOSSbot раз в 2 секунды выпускает острые пилы прямо в танк, а при пересечении с танком – наносит ему урон 2 раза за секунду; flybot не может стрелять, так что наносит урон персонажу-танку только касанием (при достижении игрока наносит урон и погибает). Враги имеют уровень здоровья, при исчерпывании которого (танк стреляет в них пулями

разного калибра) считаются уничтоженными. Во время уничтожения врагов происходит анимация взрыва. Враги могут летать над ящиками, но не могут выходить за пределы игрового поля.

Класс Enemy наследуется от класса Entity. Экземпляры Enemy создаются внутри класса Engine и уничтожаются там же, хранятся в STL-контейнере.

В приложениях А и Б приведены тексты файлов, над которыми я работал.

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. К отчету приложены диаграммы классов и вариантов использования.

Мною был получен опыт совместной разработки крупного проекта. Благодаря координированию действий от лидера команды и совместной, почти всегда слаженной работе участников команды, несмотря на возникающие трудности в ходе работы над проектом, поставленная цель была достигнута.

ПРИЛОЖЕНИЕ А

Enemy.h

```
#pragma once
#include "Entity.h";
#include "Constants.h"

class Enemy :public Entity {
public:
	//Передаем так же строку карты, чтоб все время не создавалась память при вызовах функций
	Enemy(sf::Image& image, float X, float Y, int W, int H, sf::String Name, sf::String TileMapEnemy[HEIGHT_MAP]);
	int update(float time);			//Жизнь объекта, функция вызывается в основной программе
	int SetAim(sf::Vector2f);	//Устанавливает положение движения
	sf::Vector2f GetXY() { return sf::Vector2f(x, y); };	//Возвращает позицию спрайта Enemy
	void SetStatus(sf::String St) { status = St; };
	void struck(int damage);
	sf::FloatRect GetRect();	//Переопределили функцию, чтобы подвинуть хитбокс
	void draw(sf::RenderTarget& target);	//Так как нужно много за раз нарисовать, то вынесем в отдельный метод
	sf::String GetStatus() { return status; }; // статус объекта
	sf::String GetName() { return name; }; // босс или флайбот
	void recetBOSSdamagetimer() { BOSSdamagetimer = 0; };// для сброса таймера атаки босса
	float GetBOSSdamagetimer() { return BOSSdamagetimer; };// вернули таймер
private:
	//Будет вызываться внутри, так что инкапсулирую функции
	float BOSSdamagetimer;
	sf::Sprite BossPart;		//босс состоит из 2х частей. Это - голова, которая поставится на движущуюся часть (пилы)
	sf::Sprite healthSprite;	//спрайт для полоски HP
	sf::Vector2f XYAim;			//координата спрайта игрока
	sf::String status;
	int action(float);			//действия врагов
	int animation();
	void SetDirection();		//вычисление направления по координатам, взятых с SetAim
	int checkCollisionWithMap(float, float);
	sf::Vector2f SpaunTarget();		//смотрим где наши спауны на карте
	sf::String TileMap[HEIGHT_MAP];	//карта
};
```

ПРИЛОЖЕНИЕ Б

Enemy.cpp

```
#include "Enemy.h";
/*
Есть два типа врагов:
- flybot летит к гг суециднуться
- BOSSbot летает от спауна к спауну, стреляет и пилит (когда пилит, не стреляет)
*/

Enemy::Enemy( sf::Image& image, float X, float Y, int W, int H, sf::String Name, sf::String TileMapEnemy[HEIGHT_MAP]) :Entity(image, X, Y, W, H, Name)
// ссылка на картинку, начальные координаты на общей картинке, ширина и высота спрайта,
{
	speed = 0.1; // задаем скорость движения
	BOSSdamagetimer = 0; //?
	for (int i = 0; i < HEIGHT_MAP; i++)
	{
		TileMap[i] = TileMapEnemy[i];
	}//Скопировали карту
	isMove = false;
	status = "WTFBOT";// инициировали статус
	state = fly;//летающее положение
	sf::Vector2f XY = SpaunTarget();// находим начальные координаты
	x = XY.x;
	y = XY.y;
	healthSprite.setTexture(texture);
	//каждому спрайту по текстуре!!!
	if (name == "flybot")
	{
		sprite.setTextureRect(sf::IntRect(12, 5, 57, 68));//w = 45; h = 63
		//Подрезали спрайт
		healthSprite.setTextureRect(sf::IntRect(0, 0, 34, 8));
		//Подрезали спрайт
		healthSprite.setPosition(XY.x + 10, XY.y - 10);
		//Поставили в точку
	}
	else if (name == "BOSSbot")
	{
		//w = 160; h = 60
		health = 500;
		BossPart.setTexture(texture);
		BossPart.setTextureRect(sf::IntRect(754, 216, 145, 128));
		sprite.setTextureRect(sf::IntRect(320, 0, 325, 292));

		healthSprite.setTextureRect(sf::IntRect(2, 252, 167, 25));
// здоровье
		healthSprite.setScale(0.5, 0.5); // уменьшили спрайт здоровья
		BossPart.setScale(0.5, 0.5); //

	 sprite.setScale(0.5, 0.5);// Уменьшили картинку в 2 раза
		BossPart.setOrigin(64, 72); // задали центр головы
```

```
                sprite.setOrigin(162, 146);// Нашли центр методом научного
тыка и поставили спрайту
                BossPart.setPosition(XY.x + W / 2 - 5, XY.y + H / 2 - 10);
// сместили голову
                // У боссов свои заморочки...
        }
        sprite.setPosition(XY.x + W / 2, XY.y + H / 2); // центр спрайта
}

int Enemy::checkCollisionWithMap(float Dx, float Dy)
{
        for (int i = y / 32; i < (y + h) / 32; i++)//проходимся по
элементам карты
                for (int j = x / 32; j < (x + w) / 32; j++)
                {
                        if (TileMap[i][j] == '0' )//если элемент наш тайлик
земли, то
                        {
                                isMove = false;
                                SetDirection(); // выбираем направление
движения
                        }
                }
        return 0;
}

void Enemy::SetDirection()
{
        float rotation = (atan2(XYAim.y - y, XYAim.x - x)); // вращение по
радианам
        // задаем угол поворота по направлению новой цели/спауна
        dx = cos(rotation) * 0.1;//
        dy = sin(rotation) * 0.08;//Изза разницы размеров по x & y
умножаем соответсвенно

}
sf::Vector2f Enemy::SpaunTarget()
{
        int countOfSpauns = 0;
        float Sy = 100;
        float Sx = 100;
        for (int i = 0; i < HEIGHT_MAP; i++)//проходимся по элементам
карты
                for (int j = 0; j < WIDTH_MAP; j++)
                {
                        if (TileMap[i][j] == 's')//если элемент наш тайлик
земли, то
                        {
                                countOfSpauns++;// Посмотрели сколько там
спаунов
                        }
                }
        if (countOfSpauns != 0)
        {
                int** SpaunsCoordinate = new int* [countOfSpauns];
                for (int i = 0; i < countOfSpauns; ++i)
                {
                        SpaunsCoordinate[i] = new int[2];// Создали
динамический массив координат
                }
```

```
                int randSpaun = rand() % countOfSpauns; // от 0 до 3
                for (int i = 0; i < HEIGHT_MAP; i++)//проходимся по
элементам карты
                        for (int j = 0; j < WIDTH_MAP; j++)
                        {
                                if (TileMap[i][j] == 's')//если элемент наш
тайлик земли, то
                                {
                                        countOfSpauns--;
                                        SpaunsCoordinate[countOfSpauns][0] =
i;
                                        SpaunsCoordinate[countOfSpauns][1] =
j;
                                        //Выбрали случайный спаун
                                }
                        }
                Sy = SpaunsCoordinate[randSpaun][0] * 32;//Волшебная цифра
- 32(x32). Именно такая площадь у одной координаты
                Sx = SpaunsCoordinate[randSpaun][1] * 32;

        }
        return sf::Vector2f(Sx, Sy);
}
void Enemy::draw(sf::RenderTarget& target)
{

        target.draw(sprite);
        if (name == "BOSSbot")
        {
                target.draw(BossPart);
        }
        target.draw(healthSprite);
}

sf::FloatRect Enemy::GetRect()
{
        // у босса чистый квадрат
        sf::FloatRect BufRect;
        if (name == "BOSSbot")
        {
                BufRect.left = x;
                BufRect.top = y;
                BufRect.width = w;
                BufRect.height = h;

        }
        // итбокс у нас плечи и начало огня - для красоты, чтобы не
сталкивался с пустым местом
        if (name == "flybot")
        {
                BufRect.left = x + 5;
                BufRect.top = y + 5;
                BufRect.width = w - 5;
                BufRect.height = h - h / 4;
        }
        return BufRect;
}
//Для определения действия врага
int Enemy::action(float time)
{
```

```
        if (!isMove)
        {
                isMove = true;
                SetDirection();//Если не двигаемся, то начинаем
                //а то че стоять тут
        }
        else
        {
                if (state == stay)
                {
                        return 0;
                        //Че? Все еще стоим?!
                }
                x += dx * time * speed;
                checkCollisionWithMap(dx, 0);
                y += dy * time * speed;
                checkCollisionWithMap(0, dy);
                //подвинули и проверили на столкновение
                sprite.setPosition(x + w / 2, y + h / 2); //задаем позицию
спрайта

                if (name == "BOSSbot")
                {
                        BossPart.setPosition(x + w / 2 - 5, y + h / 2 - 10);
                        healthSprite.setPosition(x , y - 20);
                }
                else if (name == "flybot")
                {
                        healthSprite.setPosition(x + 10, y - 10);
                }
                if (abs(XYAim.x - x) < w && abs(XYAim.y - y) < h)
                {
                        isMove = false;
                        dx = 0;
                        dy = 0;
                        //Достигли точки назначения. Пора искать новую цель.
                }

        }
        return 0;
}

int Enemy::animation()
{
        if (name == "flybot")
        {
                healthSprite.setTextureRect(sf::IntRect(0, 0, 9 + health /
4, 8));//health ('max = 100') / 4 = 25
                //делим палочку по состоянию здоровья
                if (status == "anikilled")
                {
                        if (moveTimer < 100)
                        {
                                sprite.setTextureRect(sf::IntRect(240, 1, 80,
80)); // взрыв
                        }
                        else if (moveTimer < 200)
                        {
                                sprite.setColor(sf::Color::Red); // взрыв
закрашиваем красным
```

```
                        }
                        else
                        {
                                status = "killed";
                        }
                        //Уже убили? Надо уйти красиво
                }
                else if (moveTimer < 200)
                {
                        sprite.setTextureRect(sf::IntRect(12, 6, 57, 68));
                }
                else if (moveTimer < 400)
                {
                        sprite.setTextureRect(sf::IntRect(172, 5, 56, 69));
                }
                else if (moveTimer < 600)
                {
                        sprite.setTextureRect(sf::IntRect(92, 5, 58, 72));
                }
                else if (moveTimer < 800)
                {
                        sprite.setTextureRect(sf::IntRect(172, 5, 56, 69));
                }
                else
                {
                        moveTimer = 0;
                }
                //Пока живы надо двигаться красиво
        }
        else if (name == "BOSSbot")
        {
                healthSprite.setTextureRect(sf::IntRect(2, 252, 24 + health
/ 3.5 , 25));//health ('max = 500') / 3,5 = 142
                if (status == "anikilled")
                {

                        if (moveTimer < 200)
                        {
                                sprite.setTextureRect(sf::IntRect(240, 1, 80,
80));
                                sprite.setScale(4,4);
                        }
                        else if (moveTimer < 300)
                        {
                                sprite.setColor(sf::Color::Red);
                                BossPart.setColor(sf::Color::Red);
                        }
                        else if (moveTimer < 400)
                        {
                                sprite.setColor(sf::Color::Yellow);
                                BossPart.setColor(sf::Color::Black);
                        }
                        else if (moveTimer < 500)
                        {
                                status = "killed";
                        }
                        //БОССА ЗАВАЛИЛИ!!! ЩАС БОМБАНЕТ!!!
                }
                else if (moveTimer < 2000)
                {
```

```
                        sprite.rotate(0.5);
                }
                else if (moveTimer < 4000)
                {
                        sprite.rotate(-0.5);
                }
                else
                {
                        moveTimer = 0;
                        //you spin my head right round, right round...
                }

        }
        return 0;
}


int Enemy::update(float time)
{
        //
        if (!life)
        {
                return 0;
                //Точно сдох. Нечего тут брыкаться
        }
        if (speed < 1)
        {
                speed += 0.001;// Наращиваем скорость, чтоб сразу после
спауна не врезаться
        }

        moveTimer += time; // добавляем квант времени
        BOSSdamagetimer += time;
        animation();
        action(time);
        if (status == "killed")
        {
                life = false;
        }
        else if (health <= 0 && status != "anikilled") {
                status = "anikilled";
                state = stay;
                moveTimer = 0;
                if (name == "BOSSbot")
                {
                        sprite.setOrigin(40, 50); // центр взрыва
                        sprite.setRotation(0); // чтобы взрыв не вращался
                        //Далее меняем спрайт. Сделал чтоб тот не ерзал.
                }
        }
        return 0;
}

int Enemy::SetAim(sf::Vector2f XY)
{

        if (!isMove) {
                if (name == "flybot")
                {
                        XYAim = XY;
```

```
                }
                else if (name == "BOSSbot")
                {
                        XYAim = SpaunTarget();
                        //Пункт назначения - спаун
                }
        }
        return 0;
}

void Enemy::struck(int damage) // 100 для большой пули, 20 - для маленькой
{
        health -= damage;
        //АЙ
}
```