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. Е. Дьяченко

«19» декабря 2022 г.

Принял:

канд. физ.-мат. наук, доцент

А. В. Бульба

«20» 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 приведен список моих коммитов:

Author: DyachenkoAnna <forannsstudy@gmail.com>
Date: Wed Dec 21 09:12:49 2022 +0300
Улила лишний файл, добавила папку для отчетов

Author: DyachenkoAnna <forannsstudy@gmail.com>
Date: Wed Dec 21 08:51:57 2022 +0300
Merge branch 'release-1.0'
В рамках курсового проекта написание игры окончено.

Author: DyachenkoAnna <forannsstudy@gmail.com>
Date: Tue Dec 20 23:29:47 2022 +0300
добавила версию программы

Author: DyachenkoAnna <forannsstudy@gmail.com>
Date: Tue Dec 20 22:37:57 2022 +0300
Обновила .gitignore

Author: DyachenkoAnna <forannsstudy@gmail.com>
Date: Tue Dec 20 21:18:02 2022 +0300
Merge branch 'Bullet' into develop

Author: DyachenkoAnna <forannsstudy@gmail.com>
Date: Tue Dec 20 21:17:55 2022 +0300
Удалила системные файлы VS

Author: DyachenkoAnna <forannsstudy@gmail.com>
Date: Tue Dec 20 21:13:04 2022 +0300
...

Author: DyachenkoAnna <forannsstudy@gmail.com>
Date: Mon Dec 19 21:47:40 2022 +0300
Добавила геттеры статуса врага, имени, таймера атаки босса (GetStatus(), GetName(), GetBOSSdamagetimer()) + сброс таймера атаки босса (recetBOSSdamagetimer())

Author: DyachenkoAnna <forannsstudy@gmail.com>
Date: Sun Dec 18 03:35:40 2022 +0300
Убрала отладочный код, ветка готова к вливанию в develop

Author: DyachenkoAnna <forannsstudy@gmail.com>
Date: Sun Dec 18 02:35:53 2022 +0300
Для флайбота добавлена анимация

Author: DyachenkoAnna <forannsstudy@gmail.com>
Date: Sun Dec 18 01:54:35 2022 +0300
Метод action() дописан полностью. Враги перемещаются к цели

Author: DyachenkoAnna <forannsstudy@gmail.com>
Date: Sun Dec 18 00:57:07 2022 +0300
Дописана функция выбора цели для флайбота

Author: DyachenkoAnna <forannsstudy@gmail.com>
Date: Sat Dec 17 23:47:21 2022 +0300
Добавлена о-о-очень простая функция, отнимающая здоровье

Author: DyachenkoAnna <forannsstudy@gmail.com>
Date: Sat Dec 17 23:38:11 2022 +0300
Добавлена проверка столкновения с картой. Обновлена функция получения координат GetRect(). Это повторный коммит, в прошлом не сохранила файл

Author: DyachenkoAnna <forannsstudy@gmail.com>
Date: Sat Dec 17 23:27:56 2022 +0300
Добавлена провека столкновения с картой. Обновлена функция получения координат GetRect()

Author: DyachenkoAnna <forannsstudy@gmail.com>
Date: Sat Dec 17 22:35:38 2022 +0300
Теперь флайботы появляются на рандомных спаунах

Author: DyachenkoAnna <forannsstudy@gmail.com>
Date: Sat Dec 17 22:26:00 2022 +0300
Забыла отрисовать сердечко

Author: DyachenkoAnna <forannsstudy@gmail.com>
Date: Sat Dec 17 22:19:31 2022 +0300
Флайбот теперь появляется на карте

Author: DyachenkoAnna <forannsstudy@gmail.com>
Date: Sat Dec 17 21:37:42 2022 +0300
Коммит ради коммита. Гит просит

Author: DyachenkoAnna <forannsstudy@gmail.com>
Date: Wed Dec 14 16:48:51 2022 +0300
Добавила файл map.h - содержит шаблон карты. Выглядит симпотично. В main.cpp отладочный код (выводит карту)

Author: DyachenkoAnna <forannsstudy@gmail.com>
Date: Wed Dec 14 16:32:03 2022 +0300
Добавила Constatnts.h - файл для описания констант. Задала размеры карты

Author: DyachenkoAnna <forannsstudy@gmail.com>
Date: Wed Dec 14 16:24:25 2022 +0300
initial commit, start a project

Перечень индивидуальных работ

1. Как Team Leader

Осуществляла руководство группой разработчиков:

- составила список задачи и разделила их между участниками проекта;
- назначала новые задачи, возникавшие в ходе работы над проектом;
- оказывала всестороннюю помощь участникам проекта;
- полностью контролировала процесс разработки, утверждала план действий.

2. Процесс разработки

Мнение каждого разработчика важно, поэтому я настояла, чтобы каждый из членов команды в течение всего процесса разработки активно участвовал в обсуждении, предлагал идеи, критиковал те или иные решения (аргументируя свое мнение, конечно же). После того, как мы приходили к соглашению, я обобщала всю информацию и назначала ответственных за оформление каждого пункта раздела 2 в общем отчете. Далее каждый разработчик в виде текста или диаграмм оформлял «принятое решение» и отправлял мне на утверждение. В процессе разработки приходилось возвращаться на предыдущие шаги, поэтому в отчетах представлены только финальные версии задокументированных шагов.

В результате составила список прецендентов и диаграмму использования.

Список вариантов использования:

1. Старт игры:
   - Навести прицел
   - Выстрелить
   - Движение вверх
   - Движение вниз
   - Движение влево
   - Движение вправо
   - Столкновение с врагом
   - Пополнить жизни
   - Сделать паузу
   - Проигрыш
2. Проигрыш:
   - Перезапуск игры
   - Выход
3. Пауза:
   - Перезапуск игры
   - Продолжить игру
   - Выход
4. Выход

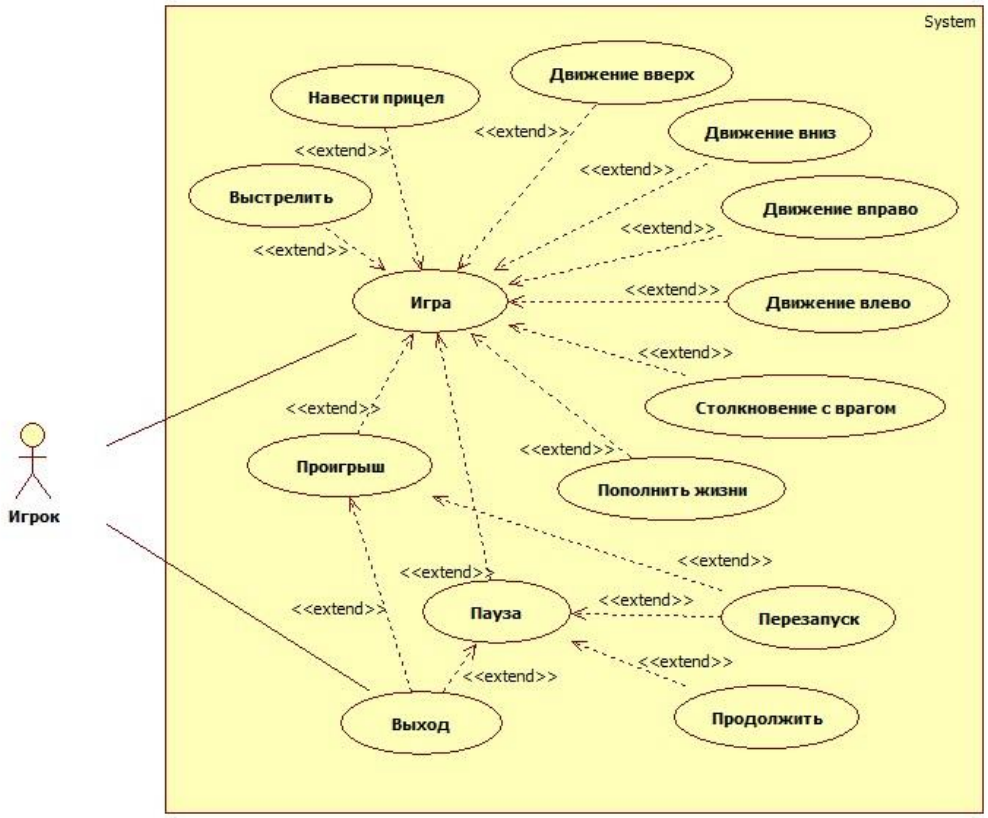

Рисунок 1 – диаграмма вариантов использования

Совместно с Сидоровым. Н. А. вела работу над классом Enemy (поиск изображений врагов, написание кода программы), экземпляры которого представляют собой врагов двух видов: flybot и BOSSbot. Они появляются в спаунах, расставленных на карте. flybot всегда летит в сторону персонажа-танка, а BOSSbot перемещается между спаунами. У flybot нет пуль, нанести урон персонажу-танку он может только врезавшись в него. BOSSbot раз в 2 секунды выпускает острые пилы прямо в танк, а при касании танка – наносит ему урон 2 раза за секунду. Враги имеют уровень здоровья, при исчерпании которого (танк стреляет в них пулями разного калибра) считаются уничтоженными. Во время их уничтожения происходит красивая 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.

Кроме того, мною был создан шаблон карты (map.h) и файл со значениями констант (Constants.h)

Приложении А, Б, В и Г приведен ы тексты файлов, над которыми я работала.

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. Поставленная цель достигнута.

Мою был получен опыт руководства группой разработчиков. Я поняла, что мне быть Team Leader-ом сложно, это слишком большая ответственность, требуется много терпения и концентрации. Но если судить по тому, что проект успешно реализован, то свои обязанности, как Team Leader-а, считаю выполненными.

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
}
```

ПРИЛОЖЕНИЕ В

map.h

```
#ifndef __map__
#define __map__

#include "Constants.h"
#include <SFML/Graphics.hpp>
/*
0 - границы
b - препятствие
p - растение)
' ' - земля/пол
s - спаун врага
h - точка хила
В дальнейшем рассматривается как массив.
*/
sf::String TileMapMy[HEIGHT_MAP] = {
      "0000000000000000000000000000000000000000",
      "0  p                                    0",
      "0   bp      p    bb             p      0",
      "0   bbbb         b                      0",
      "0   bb    p                  pp         0",
      "0                    p              b   0",
      "0         s                             0",
      "0                                       0",
      "0    p      p             bbb           0",
      "0                                   s   0",
      "0         b                             0",
      "0         b          p          p       0",
      "0      p    bbb                         0",
      "0                           b           0",
      "0                                       0",
      "0    b          p                  h    0",
      "0   bbb                        p         0",
      "0    b    p                             0",
      "0                    bbb           p    0",
      "0                    bbb     s           0",
      "0  p           p     bbb                 0",
      "0         s           p       p         0",
      "0                                  b    0",
      "0     p                                 0",
      "0000000000000000000000000000000000000000",
};

#endif
```

ПРИЛОЖЕНИЕ Г

Constants.h

```
#ifndef __Constants__
#define __Constants__

//Файл с константами

#include <SFML/Graphics.hpp>
const int HEIGHT_MAP = 25;//Высота карты
const int WIDTH_MAP = 40;//Ширина карты

#endif
```