**Федеральное государственное бюджетное образовательное учреждение высшего образования**

«Петрозаводский государственный университет»

Физико-технический институт

КУРСОВОЙ ПРОЕКТ

по дисциплине

”Технология программирования”

Выполнила:

Студентка группы 21318

О.Л. Янишена

« » 2017г.

Научный руководитель:

канд. физ.-мат. наук, доцент

А.В.Бульба

« » 2017г.

# Оглавление

# Перечисление (дата, время) и цель сделанных коммитов 3

1. Фрагменты UML диаграммы, реализованные мной 3
2. Фрагменты исходного кода, над которым я работала 4
3. Приложение1 7

**Перечисление (дата, время) и цель сделанных коммитов**

27.12.2017 18:36:42 – Создана ветка feature\_Enemy

27.12.2017 18:45:56 – Перешли на ветку feature\_Enemy. Вынесли класс Enemy в отдельный файл (Enemy.h и Enemy.cpp)

27.12.2017 18:50:43 – Закомментирована старая реализация класса Enemy в файле main.cpp

27.12.2017 19:52:57 – Добавили тайлсеты с рыбками

27.12.2017 20:41:43 – Из-за неправильной кодировки символы в комментариях некоторых файлов пострадали. Пришлось восстанавливать (Изменили map.h)

27.12.2017 20:45:18 – (Изменили Enemy.cpp)

27.12.2017 20:54:24 – (Изменили Game.cpp)

27.12.2017 23:55:24 – Исправление ошибок

27.12.2017 23:58:19 – Создание ветки release-v1.0

28.12.2017 00:02:17 – Подготовка файлов к релизу

28.12.2017 00:08:21 – Влили ветку release-v1.0 в ветку develop

29.12.2017 00:29:01 – Разделение файла map.h и изменение текстуры, когда можем есть врагов

# Фрагменты UML диаграммы, реализованные мной.

Диаграмма прецедентов:

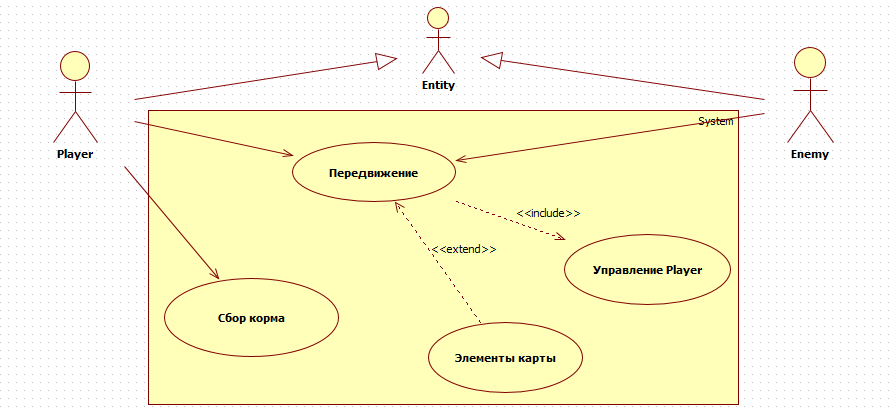
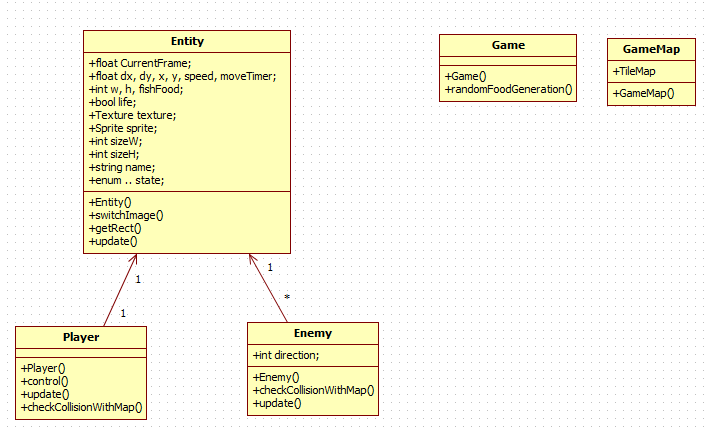


Диаграмма классов:



# Фрагменты исходного кода, над которыми я работала.

Enemy.h:

#ifndef \_Enemy\_H\_

#define \_Enemy\_H\_

//--Начало--//

#include "Entity.h"

class Enemy :public Entity

{

public:

int direction; //Направление движения врага

Enemy(Image &image, float X, float Y, int W, int H, std::string Name);

void checkCollisionWithMap(GameMap &gameMap, float Dx, float Dy); //Ф-ция проверки столкновений с картой

void update(GameMap &gameMap, float time);

};

//--Конец--//

#endif

Enemy.cpp:

#include "Enemy.h"

Enemy::Enemy(Image &image, float X, float Y, int W, int H, std::string Name) :Entity(image, X, Y, W, H, Name)

{

fishFood = 7;

if (name == "EasyEnemy")

{

//Задаем спрайту один прямоугольник для вывода одного игрока. IntRect – для приведения типов

sprite.setTextureRect(IntRect(0, 0, w, h));

direction = rand() % (3); //Направление движения врага задаём случайным образом через генератор случайных чисел

speed = 0.05; //Даем скорость. Этот объект всегда двигается

dx = speed;

}

};

void Enemy::checkCollisionWithMap(GameMap &gameMap, float Dx, float Dy) //Ф-ция проверки столкновений с картой

{

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 (j < 0) j == 0;

else if (j > WIDTH\_MAP - 1) j = WIDTH\_MAP - 1;

if (i < 0) i == 0;

else if (i > HEIGHT\_MAP - 1) i = HEIGHT\_MAP - 1;

if (gameMap.TileMap[i][j] == '0') //Если элемент - тайлик земли

{

if (Dy > 0)

{

y = i \* 32 - h; dy = -0.1;

direction = rand() % (3); //Направление движения врага

} //По Y

if (Dy < 0)

{

y = i \* 32 + 32; dy = 0.1;

direction = rand() % (3); //Направление движения врага

} //Столкновение с верхними краями

if (Dx > 0)

{

x = j \* 32 - w; dx = -0.1;

direction = rand() % (3); //Направление движения врага

} //С правым краем карты

if (Dx < 0)

{

x = j \* 32 + 32; dx = 0.1;

direction = rand() % (3); //Направление движения врага

} //С левым краем карты

}

}

};

void Enemy::update(GameMap &gameMap, float time)

{

if (name == "EasyEnemy") //Для персонажа с таким именем логика будет такой

{

if (life) { //Проверяем, жив ли герой

switch (direction) //Делаются различные действия в зависимости от состояния

{

case 0: { //Состояние идти вправо

dx = speed;

CurrentFrame += 0.005\*time;

if (CurrentFrame > 3) CurrentFrame -= 3;

sprite.setTextureRect(IntRect(96 \* int(CurrentFrame), 192, 96, 96));

break;

}

case 1: { //Состояние идти влево

dx = -speed;

CurrentFrame += 0.005\*time;

if (CurrentFrame > 3) CurrentFrame -= 3;

sprite.setTextureRect(IntRect(96 \* int(CurrentFrame), 96, 96, 96));

break;

}

case 2: { //Идти вверх

dy = -speed;

CurrentFrame += 0.005\*time;

if (CurrentFrame > 3) CurrentFrame -= 3;

sprite.setTextureRect(IntRect(96 \* int(CurrentFrame), 288, 96, 96));

break;

}

case 3: { //Идти вниз

dy = speed;

CurrentFrame += 0.005\*time;

if (CurrentFrame > 3) CurrentFrame -= 3;

sprite.setTextureRect(IntRect(96 \* int(CurrentFrame), 0, 96, 96));

break;

}

}

x += dx\*time; //Движение по "X"

checkCollisionWithMap(gameMap, dx, 0); //Обрабатываем столкновение по Х

y += dy\*time; //Движение по "Y"

checkCollisionWithMap(gameMap, 0, dy); //Обрабатываем столкновение по Y

sprite.setPosition(x, y); //Спрайт в позиции (x, y).

if (fishFood <= 0) { life = false; } //Если жизней меньше 0, либо равно 0, то умираем

}

}

};

GameMap.h:

#ifndef \_GameMap\_H\_

#define \_GameMap\_H\_

//--Начало--//

#include <SFML\Graphics.hpp>

const int HEIGHT\_MAP = 25; //Размер карты высота

const int WIDTH\_MAP = 53; //Размер карты ширина

class GameMap

{

public:

GameMap();

std::string TileMap[HEIGHT\_MAP];

};

//--Конец--//

#endif

GameMap.cpp

#include "GameMap.h"

GameMap::GameMap()

{

int i = 0;

while (i < HEIGHT\_MAP)

{

if (i == 0 || i == HEIGHT\_MAP - 1) TileMap[i] = "00000000000000000000000000000000000000000000000000000";

else TileMap[i] = "0 0";

i++;

}

}

# Приложение 1

Ссылка на репозиторий: https://github.com/PicTurax/Game