**Федеральное государственное бюджетное образовательное учреждение высшего образования**

«Петрозаводский государственный университет»

Физико-технический институт

КУРСОВОЙ ПРОЕКТ

по дисциплине

”Технология программирования”

Выполнила:

Студентка группы 21318

И.О.Евтихова

« » 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 19:05:12 – Создана ветка feature\_Player

27.12.2017 19.08:50 – Перешли на ветку feature\_Player. Вынесли класс Player в отдельный файл (Player.h и Player.cpp)

27.12.2017 19:10:32 – Закомментирована старая реализация класса Player в файле main.cpp

27.12.2017 19:13:32 – Влили ветку feature\_Player в develop

27.12.2017 19:41:19 – Создали ветку hot\_Fix

27.12.2017 19:44:07 –Изменили main.cpp

27.12.2017 20:45:18 – Влили ветку hot\_Fix в develop

27.12.2017 23:37:25 – Увеличили количество врагов, увеличили размер карты, изменили условия столкновения, изменили переменную health

28.12.2017 – 01:37:39 – Подготовка файлов к релизу

28.12.2017 01:43:22 – Влили ветку release-v1.5 в ветку develop

29.12.2017 02:26:36 – Создали ветку release\_v2.0

29.12.2017 02:31:27 – Влили ветку release\_v2.0 в master и develop

# Фрагменты UML диаграммы, реализованные мной.

**Диаграмма прецедентов**

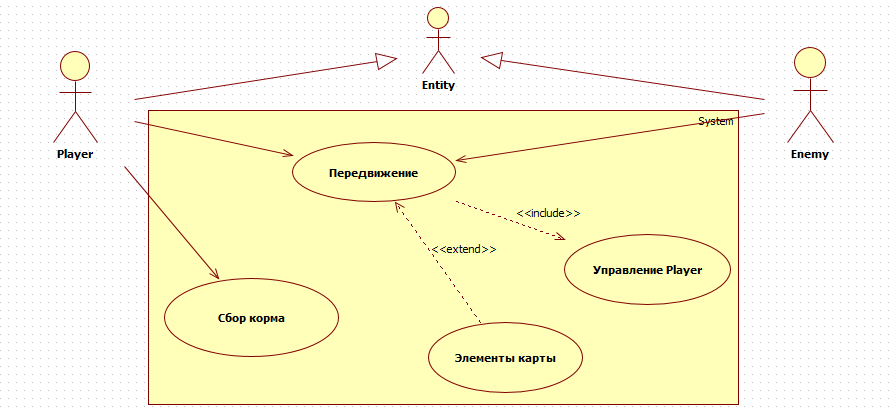
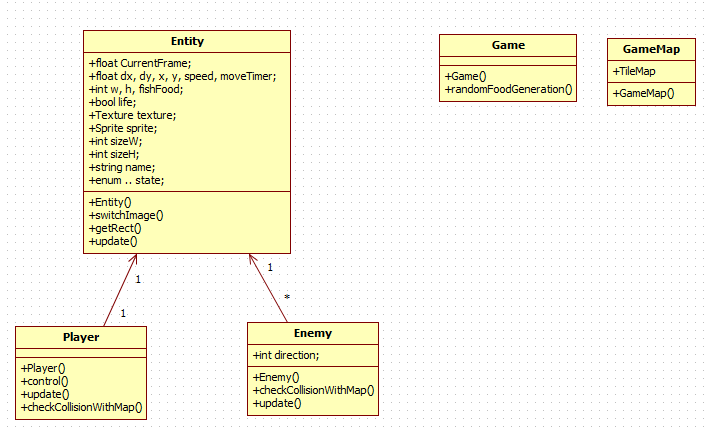


Диаграмма классов:



# Фрагменты исходного кода, над которыми я работала.

Player.cpp:

#include "Player.h"

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

{

state = stay;

if (name == "Player1")

{

//Создаем спрайту один прямоугольник для

//Вывода одного игрока. IntRect для приведения типов

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

}

}

void Player::control()

{

if (Keyboard::isKeyPressed(Keyboard::Left)) {

state = left;

speed = 0.1;

}

if (Keyboard::isKeyPressed(Keyboard::Right)) {

state = right;

speed = 0.1;

}

if (Keyboard::isKeyPressed(Keyboard::Up)) {

state = up;

speed = 0.1;

}

if (Keyboard::isKeyPressed(Keyboard::Down)) {

state = down;

speed = 0.1;

}

}

void Player::checkCollisionWithMap(GameMap &gameMap, float Dx, float Dy) //Метод проверки столкновений с элементами карты

{

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

{

if (Dy > 0) { y = i \* 32 - h; dy = 0; } //По Y

if (Dy < 0) { y = i \* 32 + 32; dy = 0; } //Столкновение с верхними краями

if (Dx > 0) { x = j \* 32 - w; dx = 0; } //С правым краем карты

if (Dx < 0) { x = j \* 32 + 32; dx = 0; } //С левым краем карты

}

if (gameMap.TileMap[i][j] == 'f') {

fishFood += 1; //Берём еду

gameMap.TileMap[i][j] = ' '; //Замена еды на пробел

}

}

}

void Player::update(GameMap &gameMap, float time) //Метод "оживления/обновления" объекта класса.

{

if (life) { //Проверяем, жив ли герой

control(); //Функция управления персонажем

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

{

case right: { //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 left: { //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 up: { //Идти вверх

dy = -speed;

CurrentFrame += 0.005\*time;

if (CurrentFrame > 3) CurrentFrame -= 3;

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

break;

}

case down: { //Идти вниз

dy = speed;

CurrentFrame += 0.005\*time;

if (CurrentFrame > 3) CurrentFrame -= 3;

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

break;

}

case stay: { //Стоим

dy = speed;

dx = speed;

break;

}

}

x += dx\*time; //Движение по "X"

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

y += dy\*time; //Движение по "Y"

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

speed = 0; //ОбнулЯем скорость, чтобы персонаж остановился

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

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

}

};

Game.cpp

#include "Game.h"

Game::Game()

{

//Пустой конструктор

}

void Game::randomFoodGenerate(std::string TileMap[], int width, int height) { //Рандомно расставляем еду

int randomElementX = 0; //Переменная для хранения случайного элемента по горизонтали

int randomElementY = 0; //Переменная для хранения случайного элемента по вертикали

int countFood = 2; //Количество еды

while (countFood > 0) {

randomElementX = 1 + rand() % (width - 1); //Псевдослучайное значение по "x" от 1 до \*ширина карты-1\*. Ограничение введено чтобы не получать числа бордюра карты

randomElementY = 1 + rand() % (height - 1); //По "y"

if (TileMap[randomElementY][randomElementX] == ' ') //Если встретили символ пробел

{

TileMap[randomElementY][randomElementX] = 'f'; //То ставим туда корм

countFood--;

}

}

}

# Entity.cpp

#include "Entity.h"

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

{

x = X; y = Y; //Координата появления спрайта

w = W; h = H;

name = Name;

moveTimer = 0;

dx = 0; dy = 0;

speed = 0;

CurrentFrame = 0;

fishFood = 1;

life = true; //Инициализировали логическую переменную жизни, герой жив

texture.loadFromImage(image); //Заносим наше изображение в текстуру

sprite.setTexture(texture); //Заливаем спрайт текстурой

}

void Entity::switchImage(Image &image)

{

texture.loadFromImage(image); //Заносим наше изображение в текстуру

sprite.setTexture(texture); //Заливаем спрайт текстурой

}

# Приложение 1

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