Министерство науки и высшего образования

ПЕТРОЗАВОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Физико-технический институт

Индивидуальный отчет по курсовому проекту
по дисциплине «Технология программирования»

Выполнили:

студент гр. 21312

Шатунов А.Н.

Преподаватель:

канд. физ.-мат. наук

Бульба А.В.

Петрозаводск 2020

Цель работы: разработать игру на языке C++ с использованием библиотеки SFML.

Программная реализация:

Среда разработки: Visual Studio Express 2013;

Язык: C++;

Заголовочные файлы:

Entity.h – Содержит класс сущность. Данный класс является предком класса Игрок и Враг.

Объявляет координаты и размеры сущности, а также логические переменные для управления стартом игры

Player.h - Содержит класс Игрока (PacMan). Данный класс содержит поле очков, и выбранное направление. А также объявляет методы управления игроком, проверки на препятствия и обновления состояния.

Enemy.h - Содержит класс Врага (Призрака). Объявляет буферные поля для памяти предыдущих координат, поле направления, методы проверки на препятствие и обновления состояния.

Interface.h - Содержит класс самого игрового поля. Объявляет поля для текстур, спрайтов, шрифтов, таймеров, карты и список врагов, а также метод взаимодействия с игрой.

Map.h - Содержит класс карты. Объявляет поле карты.

Файлы реализации (.cpp):

Entity.cpp – Содержит конструктор, инициализирующий поля данного класса.

Player.cpp – Содержит реализацию методов получения полей, управления игроком, проверкой на препятствия и обновлением, так же содержит конструктор и деструктор

Enemy.cpp - Содержит реализацию конструктора, методов проверки на препятствия и обновления.

Interface.cpp – Содержит конструктор и деструктор, а так же реализацию метода взаимодействия с полем

Map.cpp – содержит в себе реализацию карты.

Процесс разработки:

Краткое словесное описание сюжета

К нам обратился владелец клуба со старыми игровыми аппаратами. Заказчик просит создать игру, похожую на классическую PacMan. В игре должно быть реализовано:
игровое поле в виде лабиринта, в которых стенки являются препятствиями;
точки, которые поедает главный персонаж и тем самым зарабатывает очки;
3 противника в виде приведений, при прикосновении с которыми заканчивается игра.
Цель игры собрать все точки и не попасться призракам. Игра должна отображать текущий счет очков и возможность на подготовку к запуску игры и выход из нее по завершению»

Список классов

- Экран – Данный класс будет содержать реализацию самой игры. Он будет отвечать за содержание игрового поля.
- Игрок – Данный класс будет содержать в методы и атрибуты игрока. Он будет отвечать за поведение игрока и всё, что с ним происходит во время игры.

Код заголовочных файлов:

Player.h

```
#ifndef __PLAYER_H__
#define __PLAYER_H__
#include "stdafx.h"
#include "Entity.h"

using namespace sf;

class Player :public Entity {
private:
	int  playerScore; //Очки игрока
	int dir; //Направление
public:
	Player(Image &image, float X, float Y, int W, int H);  //конструктор класса image - изображение игрока, X Y координаты игрока, W H высота и ширина модели игрока
	int getScore(); //геттер для playerScore
	void setScore(int);  //сеттер для playerScore
	void control(); // метод  управления игроком
```

```
	void checkCollisionWithMap(float Dx, float Dy); //Проверка столкновение с препядствиями на карте
	void update(float time); //Обновление состояния игрока
};
#endif
```

Interface.h

```
#ifndef INTERFACE_H
#define INTERFACE_H
#include "Enemy.h"
#include "Player.h"
#include "Entity.h"
#include "Player.h"
#include "stdafx.h"

using namespace sf;
class Interface
{
private:
	Player* p;
	Font font;//Шрифт для строк
	Texture map;//текстура карты
	Sprite s_map;//спрайт карты
	Clock clock, gameTimeClock;  //таймеры
	int gameTime; //игровое время
	Map inter;
	Event event;
	Image map_image, heroImage, enemy1, enemy2, enemy3;
	std::list<Entity*>  enemy; //список врагов
	std::list<Entity*>::iterator it; //итератор для списка врагов
public:
	Interface();
	~Interface();
	void interact(); //интерфейс программы
};

#endif
```

Код исходных файлов:

Player.cpp

```
##include "stdafx.h"
#include "Player.h"

using namespace sf;

Player::Player(Image &image, float X, float Y, int W, int H) :Entity(image, X, Y, W, H)
```

```
{
	playerScore = 0;
	dir = 1;
	sprite.setTextureRect(IntRect(0, 0, w, h));
}

int Player::getScore()
{
	return(playerScore);
}

void Player::setScore(int score)
{
	playerScore = score;
}

void Player::control()
{ //При нажатии на одну из клавиш, меняется направление движения по
координате
	if (Keyboard::isKeyPressed(Keyboard::Left))
	{
		dx = -0.1;
	}
	if (Keyboard::isKeyPressed(Keyboard::Right))
	{
		dx = 0.1;
	}
	if (Keyboard::isKeyPressed(Keyboard::Up))
	{
		dy = -0.1;
	}
	if (Keyboard::isKeyPressed(Keyboard::Down))
	{
		dy = 0.1;
	}
}

void Player::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 (mp.TileMap[i][j] == '0')//если есть стена
		{
			if ((Dy > 0) && (dir == 2)) { y = i * 32 - h;  dy = 0;
}//столкновение снизу
			if ((Dy < 0) && (dir == 2)){ y = i * 32 + 32; dy = 0;
}//столкновение сверху
			if ((Dx > 0) && (dir == 1)) { x = j * 32 - w; dx = 0;
}//столкновение справа
			if ((Dx < 0) && (dir == 1)) { x = j * 32 + 32; dx = 0;
}//столкновение слева
		}
		if (mp.TileMap[i][j] == 's')
		{
```

```
                setScore(++playerScore); //Добавлени очки игроку
                mp.TileMap[i][j] = ' ';
            }
        }
}

void Player::update(float time)//обновление объекта класса.
{
      if (life)
      {//проверяем, жив ли герой
            control();
            x += dx*time; //движение по “X”
            dir = 1;
            checkCollisionWithMap(dx, 0);//обрабатываем столкновение по X
            y += dy*time; //движение по “Y”
            dir = 2;
            checkCollisionWithMap(0, dy);//обрабатываем столкновение по Y

            if (dx > 0) {//состояние идти вправо

                  dx = speed;
                  dy = 0;
                  CurrentFrame += 0.005*time;
                  if (CurrentFrame > 3) CurrentFrame -= 3;
                  sprite.setTextureRect(IntRect(32 * int(CurrentFrame), 0,
32, 32));
            }
            if (dx < 0)
            {//состояние идти влево
                  dx = -speed;
                  dy = 0;
                  CurrentFrame += 0.005*time;
                  if (CurrentFrame > 3) CurrentFrame -= 3;
                  sprite.setTextureRect(IntRect(32 * int(CurrentFrame) + 32,
0, -32, 32));
            }
            if (dy < 0)
            {//идти вверх
                  dy = -speed;
                  dx = 0;
                  CurrentFrame += 0.005*time;
                  if (CurrentFrame > 3) CurrentFrame -= 3;
                  sprite.setTextureRect(IntRect(32 * int(CurrentFrame), 32,
32, 32));
            }
            if (dy > 0)
            {//идти вниз
                  dy = speed;
                  dx = 0;
                  CurrentFrame += 0.005*time;
                  if (CurrentFrame > 3) CurrentFrame -= 3;
                  sprite.setTextureRect(IntRect(32 * int(CurrentFrame), 64,
32, 32));
            }
      }
      sprite.setPosition(x, y); //спрайт в позиции (x, y).
```

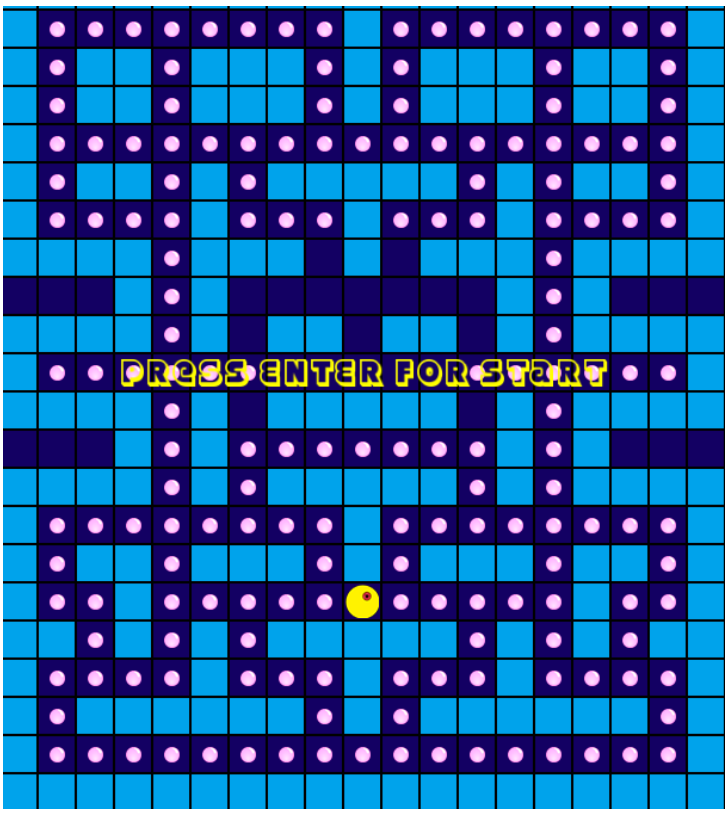
```
}
```
Interface.cpp
```
#include "stdafx.h"
#include "Interface.h"
#include "Player.h"
#include "Entity.h"
#include "Enemy.h"
using namespace sf;


Interface::Interface(){
	font.loadFromFile("fonts/Strenuous.ttf");//установка шрифта
	map_image.loadFromFile("images/map.png");//загружаем файл с текстурой
карты
	map.loadFromImage(map_image);//заряжаем текстуру карты из картинки
	s_map.setTexture(map);//устанавливаем текстуру карты
	gameTime = 0;//начало игрового времени
	heroImage.loadFromFile("images/hero.png"); // загружаем изображение
ПакМэна
	enemy1.loadFromFile("images/enemy1.png"); // загружаем изображение
призрака
	enemy2.loadFromFile("images/enemy2.png");
	enemy3.loadFromFile("images/enemy3.png");
	p = new Player(heroImage, 288, 512, 30, 30);
}

Interface::~Interface()
{
	delete p;
	while (!enemy.empty())
	{
		it = enemy.begin();
		delete *it;
		enemy.erase(it);
	}

};


void Interface::interact(){

	sf::VideoMode desktop = sf::VideoMode::getDesktopMode();
	sf::RenderWindow window(sf::VideoMode(608, 704, desktop.bitsPerPixel),
"PacMan");

	Text text("", font, 20), menu("", font, 30);//создаем объект текст
	text.setColor(Color::Yellow);//покрасили текст в красный
	text.setStyle(Text::Bold);//жирный текст.
	menu.setColor(Color::Yellow);//покрасили текст в красный
	menu.setStyle(Text::Bold);//жирный текст.
	//Player p(heroImage, 288, 512, 30, 30);//объект класса игрока
	srand(time(0));
	enemy.push_back(new Enemy(enemy1, 288, 288, 32, 32)); //создаем врагов
и помещаем в список
	enemy.push_back(new Enemy(enemy2, 256, 320, 32, 32));
	enemy.push_back(new Enemy(enemy3, 288, 320, 32, 32));
```

```
      while (window.isOpen())  { //пока открыто
            if (p->getGame() == true)
            if (Keyboard::isKeyPressed(Keyboard::Enter)){ p->setLife(true);
p->setGame(false); }
            float time = clock.getElapsedTime().asMicroseconds(); //таймер
логики
            if (p->getLife()) gameTime =
gameTimeClock.getElapsedTime().asSeconds();//игровое время
            clock.restart(); //перезапуск таймера
            time = time / 800;

            while (window.pollEvent(event))  //обработчик событий на закрытие
            {
                  if (event.type == sf::Event::Closed)
                  {
                        window.close();

                  }
                        if (Keyboard::isKeyPressed(Keyboard::Q)) {

                              window.close();
                        }
            }

            p->update(time); //обновление игрока

            if (p->getLife())
            for (it = enemy.begin(); it != enemy.end(); it++)
            {
                  (*it)->update(time); //запускаем метод update()
            }

            if (p->getLife() == true)
            {//если игрок жив
                  for (it = enemy.begin(); it != enemy.end(); it++)
                  {//бежим по списку врагов
                        if ((p->getRect().intersects((*it)->getRect())))
                        {
                              p->setLife(false);
                              std::cout << "Game over";
                        }
                  }
            }

            window.clear();

            for (int i = 0; i < 22; i++)//отрисовка карты по шаблону из
map.cpp
            for (int j = 0; j < 19; j++)
            {
                  if (inter.TileMap[i][j] == ' ')
s_map.setTextureRect(IntRect(0, 0, 32, 32));   //если пусто, то рисовать блок
земли
```

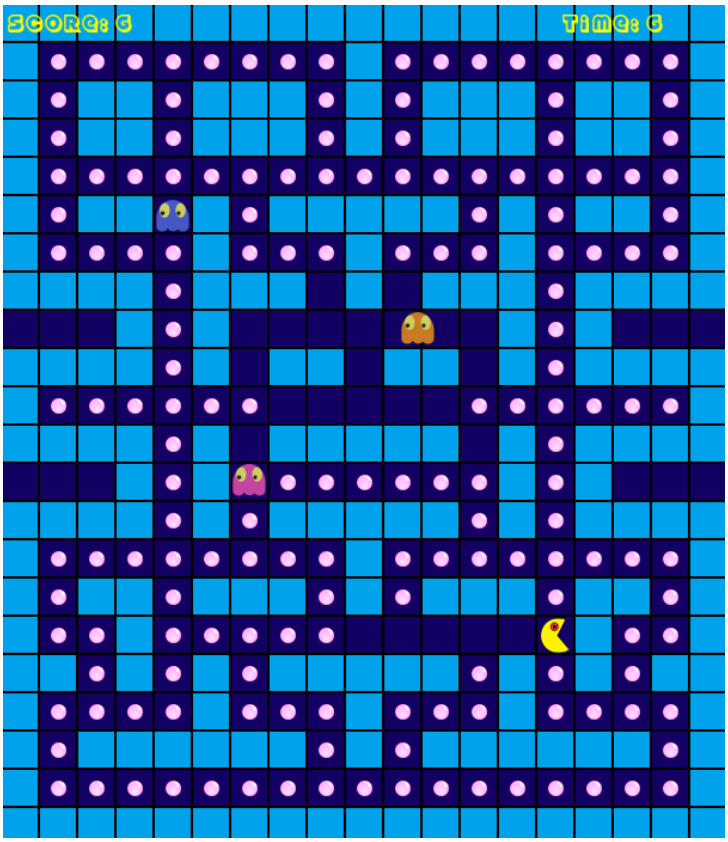
```
                if (inter.TileMap[i][j] == 's')
s_map.setTextureRect(IntRect(32, 0, 32, 32));  //если точка, то рисовать
кружок
                if ((inter.TileMap[i][j] == '0'))
s_map.setTextureRect(IntRect(64, 0, 32, 32)); //если препятствие, то рисовать
блок стены

                s_map.setPosition(j * 32, i * 32);
                window.draw(s_map);   //рисовать
          }

          std::ostringstream playerScoreString, gameTimeString;
          playerScoreString << p->getScore(); gameTimeString <<
gameTime;//Получаем счёт и время в игре
          text.setString("Score: " + playerScoreString.str() + "
Time: " + gameTimeString.str());//задаем строку тексту
          text.setPosition(5, 2);//задаем позицию текста
          window.draw(text);//рисуем этот текст

          if (p->getScore() == 176) p->setLife(false);
          if (p->getGame()){
                menu.setString("Press ENTER for start");
                menu.setPosition(100, 315);//задаем позицию текста
                window.draw(menu);//рисуем этот текст
          }
          if ((!p->getLife()) && (!p->getGame())){
                menu.setString("Press Q for exit");
                menu.setPosition(150, 315);//задаем позицию текста
                window.draw(menu);//рисуем этот текст
          }
          window.draw(p->sprite);//рисуем спрайт ПакМэна
          for (it = enemy.begin(); it != enemy.end(); it++)
          {
                window.draw((*it)->sprite); //рисуем призраков
          }
          window.display();
     }
}
```

Руководство пользователя

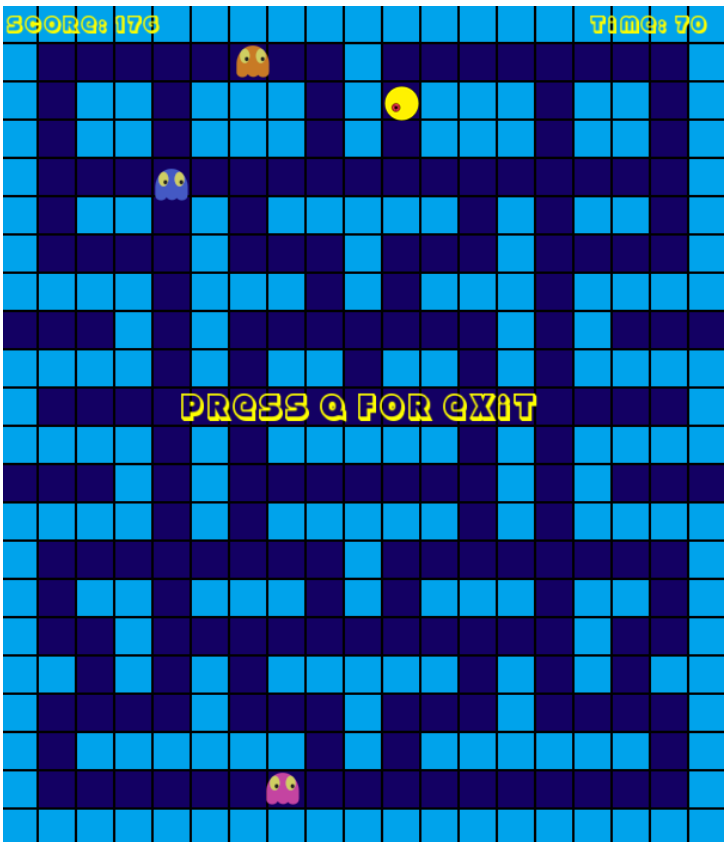
При запуске программы, первое что видит игрок это сообщение об запуске программы с помощью нажатия клавиши Enter:



После нажатия клавиши запускается игра и на экране обновляются действия героя и призраков:



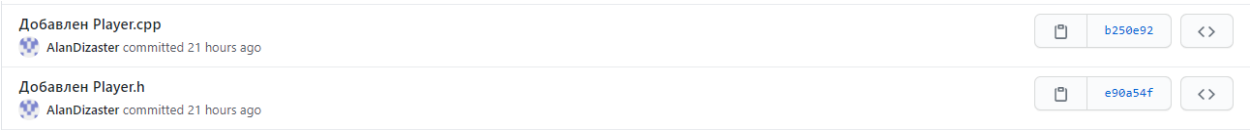
Собрав все точки либо погибнув от призрака, игра выдаст сообщения об выходе программы через кнопку Q:



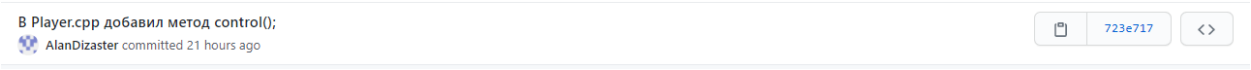
История проекта на GitHub.

Адрес репозитория: https://github.com/VitKad/Pac-man.git

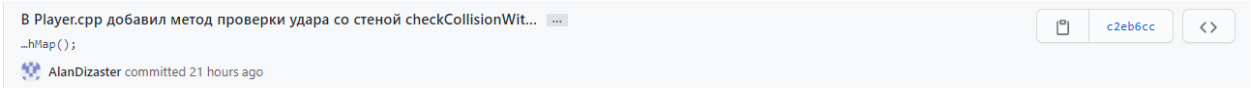
Шатунов Антон – Shatunov Anton



В коммите e90a54f был добавлен заголовочный файл для игрока
В коммите b250e92 был создан файл в котором будет содержаться реализация класса игрока.

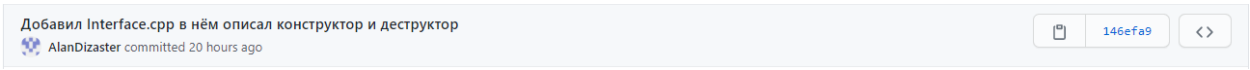


В коммите 723e717 в файл с реализацией игрока был добавлен метод control(), который отвечал за обработку нажатия клавиш пользователем.

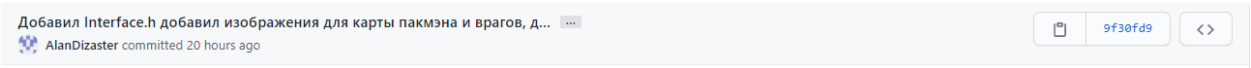


В коммите 2eb6cc в файл с реализацией игрока был добавлен метод checkCollisionWithMap(), который обрабатывает взаимодействие игрока с картой (стенами и предметами).

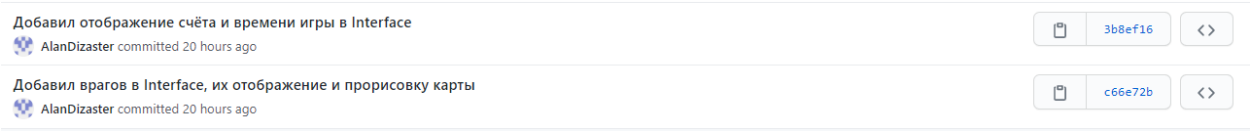
В коммите bd14692 в файл с реализацией игрока был добавлен метод update(), который обновляет положение игрока на карте.



В коммите 146efa9 добавлен файл в котором будет содержаться реализация интерфейса программы, в него добавлены конструктор и деструктор.

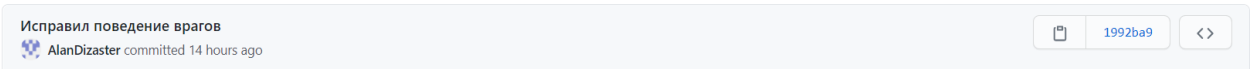


В коммите 9f30fd9 в заголовочный файл интерфейса были добавлены изображения для игрового поля, пакмэна и врагов, помещённых в список.



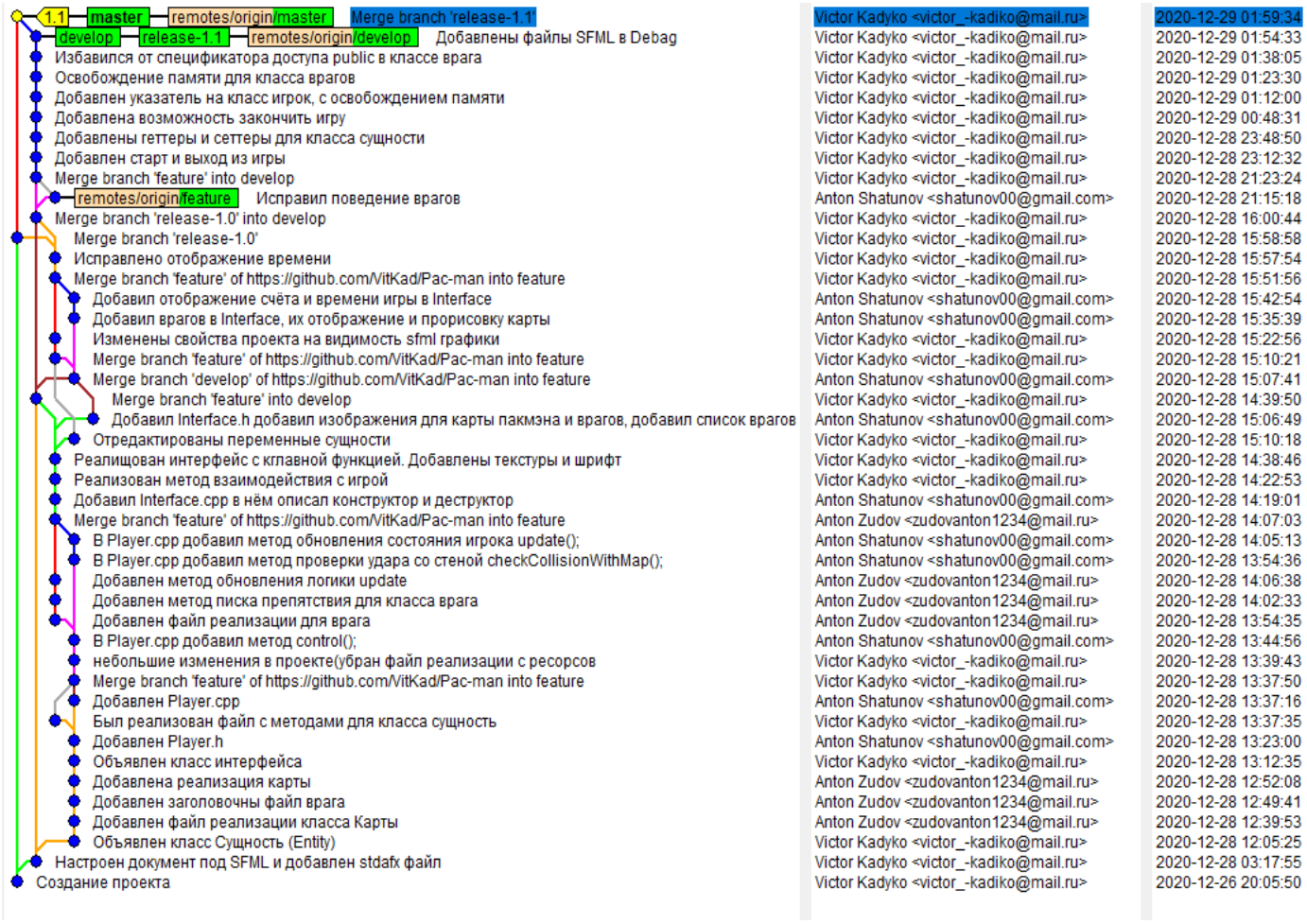
В коммите 3b8ef16 в файл с реализацией интерфейса было добавлено отображение счётчика и времени игры, настроены шрифты.

В коммите c66e72b были добавлены ещё враги и их отрисовка на крате.



В коммите 1992ba9 в файл с реализацией врагов была переработана система поведения врагов.

Графическое отображение:



Заключение.

Нашей командой была разработана игра, которая соответствует требованиям заказчика. Выбор среды разработки и языка программирования остался прежним. Была использована система контроля версия Git. Она применялась для совместной разработки программы. Сбоев и зависаний не наблюдается. Был использован принцип раздельной компиляции. Все классы разделены на отдельные заголовочные файлы, имеющие свою реализацию в соответствующих .cpp файлах. В программе реализована очистка динамической памяти. Неиспользованных переменных и избыточных алгоритмов не наблюдается. В отчете приведены диаграмма вариантов использования и диаграмма классов. Цель, поставленная заказчиком, выполнена.