**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**ПЕТРОЗАВОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Физико-технический институт**

**Командное задание в рамках курсового проекта «Разработка игры «Pacman++»»**

**Индивидуальный отчет**

**Выполнил:**

Студент

Физико-технического Института

группы 21418

Красавцев М.Ю

**Проверил:**

Преподаватель:

Бульба Артем Владимирович

**Петрозаводск 2023**

# Цель работы

Разработать аналог игры «Pacman», добавив в не функционал создания призраками бомбы.. Вести разработку с помощью системы контроля версий Git через веб-сервис GitHub. Закрепить полученные знания использования языков С++ с использованием библиотеки SFML и UML.

# Задачи

1. Познакомиться с принципами работы в команде.
2. Научиться использовать при работе систему контроля версии Git.
3. Научиться строить UML-диаграммы.
4. Научиться использовать SFML библиотеку в программной среде С++.
5. Написание отчёта о проделанной работе.

# Кратко о программной реализации

Программная реализация выполнена на языке c++ с использованием библиотеки SFML, среда разработки - IDE Qt Creator (Qt 5.4.2)

# Единицы компиляции:

score.h

score.cpp

Класс работает с игровым счетом: сохранение, загрузка и обновление в ходе игры.

# Описание сюжета игры

Игрок управляет пакманом, передвигаясь по карте и избегая призраков. Целью игры является избегание контакта с призраком и сбор всей еды(точек). При съедении большой точки, у призраков включается состояние “испуг”, в котором его можно скушать, за что даются очки, а сам призрак начнет свое движение со своего домика. При сборе всех точек, начинается новый уровень.

# Заголовочные файлы

## score.h

#ifndef SCORE\_H

#define SCORE\_H

#include "game\_types.h"

class score

{

public:

int points;

int points\_record;

void save();

void load();

void add(int points, int level);

};

#endif // SCORE\_H

# Исполнительные файлы

## main.cpp

#include <array>

#include <vector>

#include <chrono>

#include <ctime>

#include "SFML/Graphics.hpp"

#include "game\_types.h"

#include "ghost\_manager.h"

#include "tools.h"

#include "graphics.h"

int main()

{

bool game\_won = 0;

unsigned lag = 0;

byte level = 0;

std::chrono::time\_point<std::chrono::steady\_clock> previous\_time;

std::array<std::string, MAP\_HEIGHT> map\_sketch = {

" ################### ",

" #........#........# ",

" #o##.###.#.###.##o# ",

" #.................# ",

" #.##.#.#####.#.##.# ",

" #....#...#...#....# ",

" ####.### # ###.#### ",

" #.# 0 #.# ",

"#####.# ##=## #.#####",

" . #123# . ",

"#####.# ##### #.#####",

" #.# #.# ",

" ####.# ##### #.#### ",

" #........#........# ",

" #.##.###.#.###.##.# ",

" #o.#.....P.....#.o# ",

" ##.#.#.#####.#.#.## ",

" #....#...#...#....# ",

" #.######.#.######.# ",

" #.................# ",

" ################### "};

game\_map map;

std::array<position, 4> ghost\_positions;

std::vector<bomb> bombs;

sf::Event event;

sf::RenderWindow window(sf::VideoMode(CELL\_SIZE \* MAP\_WIDTH \* SCREEN\_RESIZE, (FONT\_HEIGHT + CELL\_SIZE \* MAP\_HEIGHT) \* SCREEN\_RESIZE), "Pac-Man", sf::Style::Close);

window.setView(sf::View(sf::FloatRect(0, 0, CELL\_SIZE \* MAP\_WIDTH, FONT\_HEIGHT + CELL\_SIZE \* MAP\_HEIGHT)));

ghost\_manager ghost\_manager;

pacman pacman;

srand(time(0));

map = make\_game\_map(map\_sketch, ghost\_positions, pacman);

ghost\_manager.reset(level, ghost\_positions);

previous\_time = std::chrono::steady\_clock::now();

pacman.score.load();

while (window.isOpen())

{

unsigned delta\_time = std::chrono::duration\_cast<std::chrono::microseconds>(std::chrono::steady\_clock::now() - previous\_time).count();

lag += delta\_time;

previous\_time += std::chrono::microseconds(delta\_time);

while (FRAME\_DURATION <= lag)

{

lag -= FRAME\_DURATION;

while (window.pollEvent(event))

{

switch (event.type)

{

case sf::Event::Closed:

pacman.score.save();

window.*close*();

default:

break;

}

}

for (long unsigned int i = 0; i < bombs.size(); i++)

if (bombs[i].exploded)

bombs.erase(bombs.begin() + i);

for (bomb &b : bombs)

if (b.pacman\_collision(pacman.cur\_position))

pacman.is\_dead = true;

if (!game\_won && !pacman.is\_dead)

{

game\_won = 1;

pacman.update(level, map);

ghost\_manager.update(level, map, pacman, bombs);

for (int i = 0; i < MAP\_WIDTH; i++)

{

for (int j = 0; j < MAP\_HEIGHT; j++)

if (map[i][j] == cell::pellet)

{

game\_won = 0;

break;

}

if (!game\_won)

break;

}

if (game\_won)

pacman.animation\_timer = 0;

}

else if (sf::Keyboard::isKeyPressed(sf::Keyboard::Return))

{

game\_won = 0;

if (pacman.is\_dead)

{

level = 0;

pacman.score.points = 0;

pacman.score.save();

}

else

level++;

map = make\_game\_map(map\_sketch, ghost\_positions, pacman);

ghost\_manager.reset(level, ghost\_positions);

pacman.reset();

bombs.clear();

}

if (FRAME\_DURATION > lag)

{

window.clear();

if (!game\_won && !pacman.is\_dead)

{

draw\_map(map, window);

for (bomb &b : bombs)

b.draw(window);

ghost\_manager.draw(GHOST\_FLASH\_START >= pacman.energizer\_timer, window);

draw\_text(0, {0, CELL\_SIZE \* MAP\_HEIGHT}, "Level: " + std::to\_string(1 + level), window);

draw\_text(0, {CELL\_SIZE \* MAP\_WIDTH - CELL\_SIZE \* 16, CELL\_SIZE \* MAP\_HEIGHT}, "Score: " + std::to\_string(pacman.score.points), window);

draw\_text(0, {CELL\_SIZE \* MAP\_WIDTH - CELL\_SIZE \* 8, CELL\_SIZE \* MAP\_HEIGHT}, "Record: " + std::to\_string(pacman.score.points\_record), window);

}

pacman.draw(game\_won, window);

if (pacman.is\_animation\_over)

{

if (game\_won)

draw\_text(1, {0, 0}, "Next level!", window);

else

draw\_text(1, {0, 0}, "Game over!", window);

}

window.display();

}

}

}

}

## score.cpp

#include "score.h"

#include <fstream>

void score::save()

{

std::ofstream file("max\_score.txt", std::ofstream::out);

file << points\_record;

file.close();

}

void score::load()

{

std::ifstream file("max\_score.txt", std::ifstream::in);

if (!file.is\_open())

return;

file >> points\_record;

file.close();

}

void score::add(int points, int level)

{

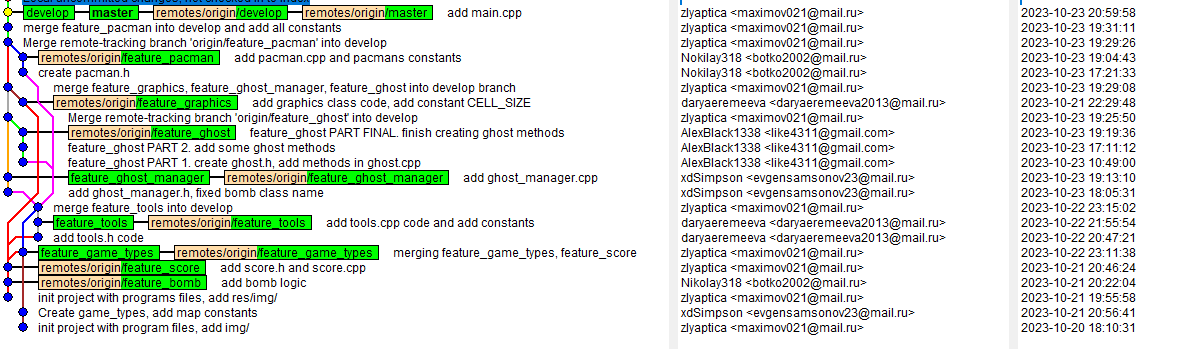
this->points += points \* (level + 1);

if (this->points > points\_record)

points\_record = this->points;

}

# История проекта Github



**Мои коммиты:**

1. "init project with program files, add img/"

Инициализация проекта, создание всех файлов, которые необходимо для работы игры. Добавлена папка с изображениями.

1. “add score.h and score.cpp”.

Добавлен код для файлов score.h иscore.cpp.

1. “merging feature\_game\_types, feature\_score”.

Объединение веток feature\_game\_types, feature\_score с develop.

1. “merge feature\_tools into develop”.

Объединение ветки feature\_tools с develop.

1. “Merge remote-tracking branch 'origin/feature\_ghost' into develop”.

Объединение ветки feature\_ghost с develop.

1. “merge feature\_graphics, feature\_ghost\_manager, feature\_ghost into develop branch”.

Объединены ветки feature\_graphics, feature\_ghost\_manager, feature\_ghost с develop.

1. “Merge remote-tracking branch 'origin/feature\_pacman' into develop”.

Объединение ветки feature\_pacman' с develop.

1. “merge feature\_pacman into develop and add all constants”.

Объединение ветки feature\_pacman с develop, добавлены все константы.

1. “add main.cpp”.

Добавлен код main.cpp, слияние с веткой master.

# Вывод

В ходе выполнения командного задания в рамках курсового проекта использовалась среда разработки Qt язык C++ с использованием библиотеки SFML. Применялась система контроля версий git. Реализованы все прецеденты. В ходе тестирования игры зависания или сбои не были обнаружены. В алгоритмы методов старались не добавлять не нужных циклов, массивов и др. Был разработан класс score, который содержит в себе реализацию сохранения, чтения счета игры, а также увеличение его в процессе игры. Также мною был создан код для main.cpp, который управляет игрой.