**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**

**Выполнила:**

Студентка

Физико-технического

Института группы 21318

Еремеева Дарья

**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:**

constants.h

Файл хранит константы, используемые в проекте.

bomb.h

bomb.cpp

Класс описывает сущность бомбы и ее поведение.

game\_types.h

game\_types.cpp

Класс содержит перечисляемые типа и структуру, используемые в проекте, например, для хранения позиции.

ghost.h

ghost.cpp

Описание класса призрака. Содержит в себе поведение призрака

ghost\_manager.h

ghost\_manager.cpp

Управляет поведением всех призраков.

graphics.h

graphics.cpp

Класс отвечает за отрисовку карты и текста.

pacman.h

pacman.cpp

Класс описывает сущность пакмана и его поведение.

score.h

score.cpp

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

tools.h

tools.cpp

Класс содержит инструменты, такие как: установление соответствия между картой и тем, что нарисовано в единице карты(в т.ч. начальные позиции призраков, пакмана, еды), проверка на коллизию и подсчет координат для нее, а также логику сбора еды.

**Заголовочные файлы:**

## **graphics.h**

#ifndef GRAPHICS\_H

#define GRAPHICS\_H

#include "game\_types.h"

#include "pacman.h"

void draw\_map(game\_map &map, sf::RenderWindow &window);

void draw\_text(bool center, position pos, std::string text, sf::RenderWindow &window);

#endif // GRAPHICS\_H

## **tools.h**

#ifndef TOOLS\_H

#define TOOLS\_H

#include "game\_types.h"

#include "pacman.h"

#include "score.h"

game\_map make\_game\_map(const std::array<std::string, MAP\_HEIGHT> &map\_from\_strings, std::array<position, 4> &ghost\_positions, pacman &pacman);

void free\_map(game\_map p);

bool map\_collision(bool use\_door, position point, game\_map& map);

bool collect\_pellets(position point, game\_map &map, score &score, int level);

#endif // TOOLS\_H

**Исполнительные файлы:**

## **graphics.cpp**

#include "graphics.h"

#include "SFML/Graphics.hpp"

#include <math.h>

void draw\_map(game\_map &map, sf::RenderWindow &window)

{

   sf::Sprite sprite;

   sf::Texture texture;

   texture.loadFromFile("img/map" + std::to\_string(CELL\_SIZE) + ".png");

   sprite.setTexture(texture);

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

   {

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

       {

           sprite.setPosition((float)(CELL\_SIZE \* a), (float)(CELL\_SIZE \* b));

           switch (map[a][b])

           {

           case cell::empty:

               break;

           case cell::door:

               sprite.setTextureRect(sf::IntRect(2 \* CELL\_SIZE, CELL\_SIZE, CELL\_SIZE, CELL\_SIZE));

               window.draw(sprite);

               break;

           case cell::energizer:

               sprite.setTextureRect(sf::IntRect(CELL\_SIZE, CELL\_SIZE, CELL\_SIZE, CELL\_SIZE));

               window.draw(sprite);

               break;

           case cell::pellet:

               sprite.setTextureRect(sf::IntRect(0, CELL\_SIZE, CELL\_SIZE, CELL\_SIZE));

               window.draw(sprite);

               break;

           case cell::wall:

               bool down = 0;

               bool left = 0;

               bool right = 0;

               bool up = 0;

               if (b < MAP\_HEIGHT - 1)

                   if (map[a][1 + b] == cell::wall)

                       down = 1;

               if (a > 0)

               {

                   if (map[a - 1][b] == cell::wall)

                       left = 1;

               }

               else

                   left = 1;

               if (a < MAP\_WIDTH - 1)

               {

                   if (map[1 + a][b] == cell::wall)

                       right = 1;

               }

               else

                   right = 1;

               if (b > 0)

                   if (map[a][b - 1] == cell::wall)

                       up = 1;

               sprite.setTextureRect(sf::IntRect(CELL\_SIZE \* (down + 2 \* (left + 2 \* (right + 2 \* up))), 0, CELL\_SIZE, CELL\_SIZE));

               window.draw(sprite);

           }

       }

   }

}

void draw\_text(bool center, position pos, std::string text, sf::RenderWindow &window)

{

   short character\_x = pos.x;

   short character\_y = pos.y;

   byte character\_width;

   sf::Sprite character\_sprite;

   sf::Texture font\_texture;

   font\_texture.loadFromFile("img/font.png");

   character\_width = font\_texture.getSize().x / 96;

   character\_sprite.setTexture(font\_texture);

   if (center)

   {

       character\_x = (short)(round(0.5f \* (CELL\_SIZE \* MAP\_WIDTH - character\_width \* text.substr(0, text.find\_first\_of('\n')).size())));

       character\_y = (short)(round(0.5f \* (CELL\_SIZE \* MAP\_HEIGHT - FONT\_HEIGHT \* (1 + std::count(text.begin(), text.end(), '\n')))));

   }

   for (std::string::const\_iterator a = text.begin(); a != text.end(); a++)

   {

       if ('\n' == \*a)

       {

           if (center)

               character\_x = (short)(round(0.5f \* (CELL\_SIZE \* MAP\_WIDTH - character\_width \* text.substr(1 + a - text.begin(), text.find\_first\_of('\n', 1 + a - text.begin()) - (1 + a - text.begin())).size())));

           else

               character\_x = pos.x;

           character\_y += FONT\_HEIGHT;

           continue;

       }

       character\_sprite.setPosition(character\_x, character\_y);

       character\_sprite.setTextureRect(sf::IntRect(character\_width \* (\*a - 32), 0, character\_width, FONT\_HEIGHT));

       character\_x += character\_width;

       window.draw(character\_sprite);

   }

}

## **tools.cpp**

#include "tools.h"

#include <math.h>

#include <array>

#include <string>

#include "SFML/Graphics.hpp"

game\_map make\_game\_map(const std::array<std::string, MAP\_HEIGHT> &map\_sketch, std::array<position, 4> &ghost\_positions, pacman &pacman)

{

   game\_map output\_map;

   for (byte a = 0; a < MAP\_HEIGHT; a++)

   {

       for (byte b = 0; b < MAP\_WIDTH; b++)

       {

           output\_map[b][a] = cell::empty;

           switch (map\_sketch[a][b])

           {

           case '#':

               output\_map[b][a] = cell::wall;

               break;

           case '=':

               output\_map[b][a] = cell::door;

               break;

           case '.':

               output\_map[b][a] = cell::pellet;

               break;

           case '0':

               ghost\_positions[0].x = CELL\_SIZE \* b;

               ghost\_positions[0].y = CELL\_SIZE \* a;

               break;

           case '1':

               ghost\_positions[1].x = CELL\_SIZE \* b;

               ghost\_positions[1].y = CELL\_SIZE \* a;

               break;

           case '2':

               ghost\_positions[2].x = CELL\_SIZE \* b;

               ghost\_positions[2].y = CELL\_SIZE \* a;

               break;

           case '3':

               ghost\_positions[3].x = CELL\_SIZE \* b;

               ghost\_positions[3].y = CELL\_SIZE \* a;

               break;

           case 'P':

               pacman.cur\_position = {CELL\_SIZE \* b, CELL\_SIZE \* a};

               break;

           case 'o':

               output\_map[b][a] = cell::energizer;

           }

       }

   }

   return output\_map;

}

std::array<position, 4> calculate\_map\_coordinates(position point, game\_map &map)

{

   std::array<position, 4> output;

   float cell\_x = point.x / (float)(CELL\_SIZE);

   float cell\_y = point.y / (float)(CELL\_SIZE);

   output[0] = {(int)floor(cell\_x), (int)floor(cell\_y)};

   output[1] = {(int)ceil(cell\_x), (int)floor(cell\_y)};

   output[2] = {(int)floor(cell\_x), (int)ceil(cell\_y)};

   output[3] = {(int)ceil(cell\_x), (int)ceil(cell\_y)};

   return output;

}

bool map\_collision(bool use\_door, position point, game\_map &map)

{

   bool output = 0;

   std::array<position, 4> corners = calculate\_map\_coordinates(point, map);

   for (position &corner : corners)

   {

       int x = corner.x;

       int y = corner.y;

       if (0 <= x && 0 <= y && MAP\_HEIGHT > y && MAP\_WIDTH > x)

       {

           if (map[x][y] == cell::wall)

               output = 1;

           else if (!use\_door && map[x][y] == cell::door)

               output = 1;

       }

   }

   return output;

}

bool collect\_pellets(position point, game\_map &map, score &score, int level)

{

   bool output = 0;

   std::array<position, 4> corners = calculate\_map\_coordinates(point, map);

   for (position &corner : corners)

   {

       int x = corner.x;

       int y = corner.y;

       if (map[x][y] == cell::energizer)

       {

           output = 1;

           map[x][y] = cell::empty;

       }

       else if (map[x][y] == cell::pellet)

       {

           map[x][y] = cell::empty;

           score.add(POINTS\_FOR\_PELLET, level);

       }

   }

   return output;

}

**Руководство пользователя:**

Запустить программу (игру).

Появится игровое поле, где вы управляете пакменом ( рис. 3, желтый круг) при помощи стрелок. Для прохождения уровня необходимо собрать все точки.

Призраки в это время будут пытаться догнать вас, если хотя бы один из них достигнет игрока - игра будет окончена.

Вы можете вернуть призраков в домик, если игрок подберет большую точку, то при соприкосновении с призраком, тот убежит домой, в центр карты, куда пакману проход запрещен.

Если вы проиграли, чтобы начать игру, снова нажмите ЭНТЕР.

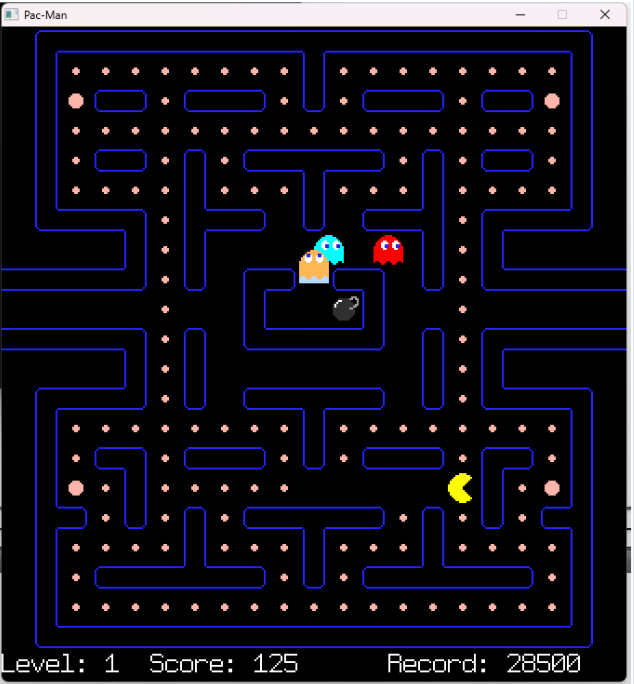
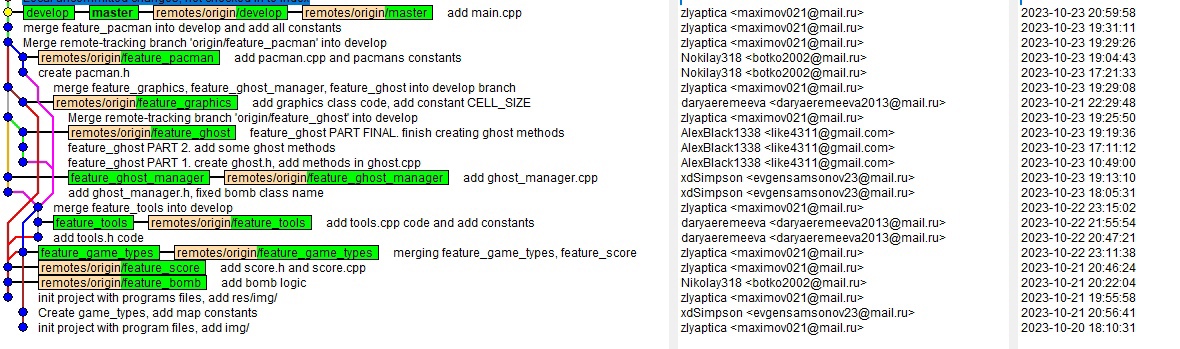


Рис. 1. Запущенная игра

**История проекта Github:**



(Еремеева Дарья)

“add tools.h code”.

Добавлен код для файла tools.h.

“add tools.cpp code and add constants”.

Добавлена реализация файла tools.cpp и добавлены константы.

“add graphics class code, add constant CELL\_SIZE”.

Добавлен код для класса graphics, добавлена константа CELL\_SIZE.

**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.Неиспользуемых переменных в коде нет. В алгоритмы методов старались не добавлять не нужных циклов, массивов и др. К отчету приложены диаграмма вариантов использования и диаграмма классов.