Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Основы алгоритмизации и программирования (ОАИП)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему:

**ПРОГРАММНОЕ ИГРОВОЕ СРЕДСТВО**

**«ЛАБИРИНТ»**

БГУИР КП 1-40 01 01 028 ПЗ

Студент Прокопчук Д.М.

Руководитель Шостак Е.В.

Минск 2022

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 5](#_Toc121684149)

[1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 6](#_Toc121684150)

[1.1 Обзор аналогов 6](#_Toc121684151)

[1.2 Постановка задачи 7](#_Toc121684152)

[2 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА 8](#_Toc121684153)

[2.1 Структура программы 8](#_Toc121684154)

[2.2 Проектирование интерфейса программного средства 8](#_Toc121684155)

[2.3 Проектирование функционала программного средства 10](#_Toc121684156)

[3 РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА 17](#_Toc121684157)

[3.1 Передвижение персонажа 17](#_Toc121684158)

[4 ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА 24](#_Toc121684159)

[5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 27](#_Toc121684160)

[5.1 Интерфейс программного средства 27](#_Toc121684161)

[5.2 Управление программным средством 30](#_Toc121684162)

[ЗАКЛЮЧЕНИЕ 31](#_Toc121684163)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 32](#_Toc121684164)

[ПРИЛОЖЕНИЕ А 33](#_Toc121684165)

# ВВЕДЕНИЕ

Видеоигра (от [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) video game) – [игра](https://ru.wikipedia.org/wiki/%D0%98%D0%B3%D1%80%D0%B0) с использованием изображений, сгенерированных электронной аппаратурой. Другими словами, видеоигра является [электронной игрой](https://en.wikipedia.org/wiki/electronic_game), которая базируется на взаимодействии человека и устройства посредством визуального [интерфейса](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81), например [телевизора](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BB%D0%B5%D0%B2%D0%B8%D0%B7%D0%BE%D1%80), [монитора](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%BD%D0%B8%D1%82%D0%BE%D1%80_(%D1%83%D1%81%D1%82%D1%80%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE)) компьютера или телефона. Исторически видеоигры появились раньше, чем был создан персональный компьютер. Это обусловлено тем, что в 1960-х годах компьютерные технологии не были массовыми и не рассматривались как серьёзный инструмент в гражданской сфере. Создаваемые видеоигры на электронной базе (электронные игры) сыграли важную роль в мирной социализации компьютерной техники и её интеграции в популярную культуру. В 1970-х видеоигры стали одним из самых популярных развлечений, когда устанавливались игровые автоматы в барах, кафе и других заведениях. Компьютерные игры оказали столь существенное влияние на общество, что в [информационных технологиях](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5_%D1%82%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D0%B8) отмечена устойчивая тенденция к [геймификации](https://ru.wikipedia.org/wiki/%D0%93%D0%B5%D0%B9%D0%BC%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F) для неигрового [прикладного программного обеспечения](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5).

Сейчас существует большое количество различных жанров видеоигр:

симуляторы, стратегии, RPG, файтинги, стелс, текстовые, рогалики, визуальные новеллы и многие другие. Популярностью пользуются и различного рода лабиринты. Лабиринт — жанр [компьютерных игр](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%B8%D0%B3%D1%80%D0%B0); игры данного жанра характеризуются тем, что успех игрока определяется в основном навигацией и ориентацией в лабиринте. Ответ на вопрос, является ли что-то лабиринтом, относителен, но как правило имеется возможность различить, ставит ли предлагаемая комбинация комнат и коридоров необходимость решения задачи навигации в лабиринте. Лабиринт является довольно распространённым решением в построении современной видеоигры, чаще всего лабиринт появляется в виде побочного задания или головоломки, однако бывают и целые игры посвящённые теме лабиринтов.Существует также огромная вариативность в условиях игры. Например, добавление различных комнат или ключей, добавление нескольких лабиринтов. ­­ Одной из самых старых игр в жанре лабиринт является игра “Gotcha”. В данной игре два игрока по очереди пытаются догнать друг друга, а после этого меняются ролями.

Целью данной курсовой работы является разработка игрового средства «Лабиринт».

# АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

Идея создания лабиринтов зарождалась ещё в древние времена. До нас дошли легенды о Кносском лабиринте, созданном великим учёным и изобретателем Дедалом на острове Крит. Первые игры о лабиринтах пользовались популярностью, однако игрокам уже приелись простенькие лабиринты, в которых видна вся карта. Тогда разработчики игр начали придумывать новые идеи, такие как закрытие всей области кроме небольшого круга вокруг игрока, многократное увеличение лабиринта, добавление врагов и различного рода активностей.

## Обзор аналогов

1.1.1 “Али-Баба и 40 воров”

“Али-Баба и 40 воров” — игра, выпущенная в 1982 году для платформы Sega, представляет собой небольшой лабиринт, где игроку предстоит отыгрывать роль Али-Бабы из одноимённой народной сказки, а также убить 40 воров, которые пытаются украсть его деньги. На рисунке 1.1 предоставлен внешний вид программы.

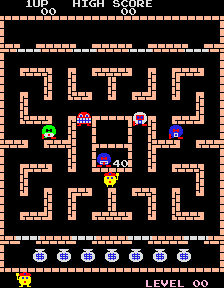


Рисунок 1.1 – Интерфейс игры Али-Баба и 40 воров

1.1.2 “Robot Rescue”

“Robot Rescue” – игра-головоломка, представляет собой около 45 различных лабиринтов, включающих в себя ловушки, усложняющиеся с каждым новым уровнем. Целью игры является спасти всех заключенных роботов. Пример интерфейса можно увидеть на рисунке 1.2.

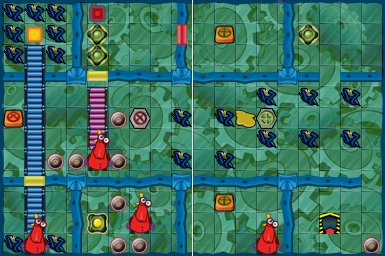


Рисунок 1.2 – Интерфейс игры “Robot Rescue”

## Постановка задачи

В данном игровом средстве планируется реализовать следующие функции:

* движение модели игрока;
* случайная генерация лабиринта;
* показ победного окна;
* скрытие карты кроме небольшой области вокруг персонажа;
* выбор аватара на своё усмотрение;
* добавление аватарам способностей;
* добавление активностей с картой(например, ключи);

Для разработки программного средства будет использоваться язык программирования C++ и среда разработки CLion, а также библиотека SFML.

# ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

## 2.1 Структура программы

При разработке приложения будет использовано 6 модулей:

* MainMenu – модуль для отображения главного меню;
* Game – модуль, отвечающий за окно игры и все взаимодействия с

ним;

* ChooseMenu – отображение окна выбора аватара;
* Hero – реализация класса, описывающего главного персонажа;
* Door – реализация вспомогательного класса, отвечающего за двери;
* Instruction– отображение инструкции;

## 2.2 Проектирование интерфейса программного средства

2.2.1 Главное меню

Главное меню приложения должно состоять из простенького, содержащего надписи-кнопки окна. Главное должно содержать следующие надписи: “Начать игру”, “О разбработчике”, “Выйти”, “Выбрать персонажа”, “Инструкция”. Макет главного окна приложения в Delphi-форме представлен на рисунке 2.1.

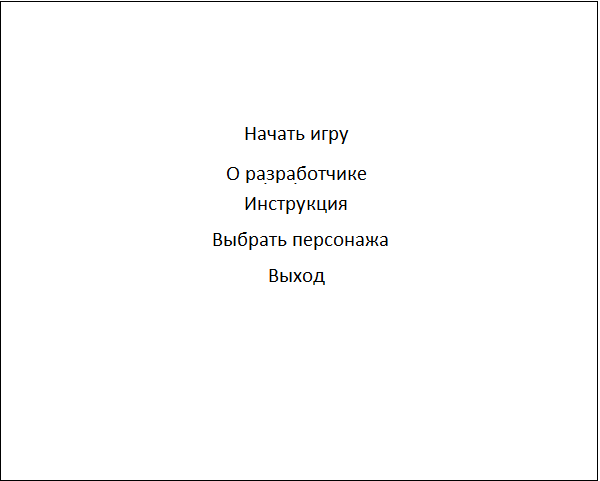


Рисунок 2.1 – Главное меню приложения

2.2.2 Окно игры

Окно игры должно представлять из себя большое окно, содержащее игровое поле внутри себя, а также персонажа.Увидеть можно на рисунке 2.2.



Рисунок 2.2 – Окно игры

2.2.3 Окно выбора аватара

Окно выбора аватара должно содержать значок для выхода из окна, а также изображения всех доступных аватаров, а по нажатию на изображение должна начинаться игра с выбранным аватаром. Макет окна выбора аватара представлен на рисунке 2.3.

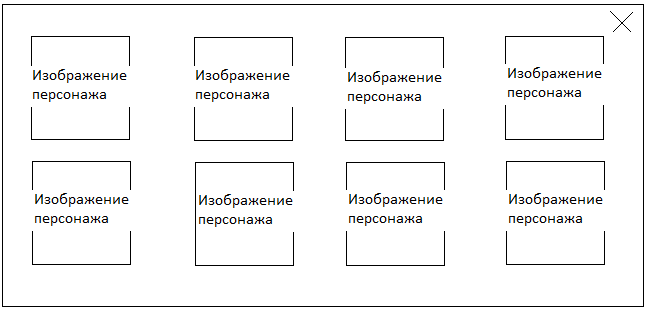


Рисунок 2.3 – Окно выбора аватара

2.2.4 Окно инструкции

Окно инструкции должно иметь текст, освещающий некоторые вопросы, касающиеся управления персонажем и его способностями, а также описание этих способностей. Макет окна инструкции представлен на рисунке 2.4.

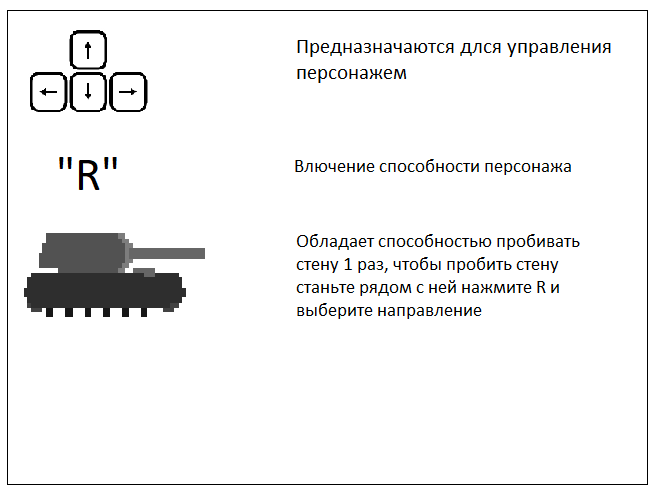


Рисунок 2.4 – Окно инструкции

## 2.3 Проектирование функционала программного средства

При создании игры очень важно сразу определить задачу и цели, а также хорошо составить рабочие алгоритмы. Искать ошибки в коде придётся в любом случае, но хорошо написанный алгоритм упрощает это в разы. В игре должны быть использованы следующие методы:

* автоматическое создание случайного лабиринта;
* отслеживание победы персонажа;
* открытие двери при наличии ключа;
* подбор ключа в лабиринте;
* осуществление способностей автаров.

2.3.1 Автоматическое создание случайного лабиринта

Автоматическое создание случайного лабиринта необходимо, чтобы сделать каждую партию разнообразной и интересной. При создании будет использоваться основная процедура mazeMake. Блок-схема алгоритма данной функции приведена на рисунке 2.5.



Рисунок 2.5 – Блок-схема процедуры mazeMake.

2.3.2 Отслеживание победы персонажа

Отслеживание победы является обязательным атрибутом данного программного средства. На протяжении всей игры после каждого хода персонажа будет осуществляться проверка победы. Блок-схема алгоритма функции checkWinCondition приведена на рисунке 2.6.



Рисунок 2.6 – Блок-схема процедуры checkWinCondition

2.3.3 Открытие двери при наличии ключа

Открытие двери в комнату необходимая механика, так как в одной из 2 комнат лежит золотой ключ, а он является непосредственным условием победы, так как без него нельзя пройти на выход. Блок-схема алгоритма процедуры openDoor приведена на рисунке 2.7.



Рисунок 2.7 – Блок-схема процедуры openDoor

2.3.4 Подбор ключа в лабиринте

Подбор ключа также является необходимой механикой, так как всё взаимодействие с комнатами завязано на ключах, работа метода подбора ключа будет осуществляться на каждом шаге персонажа, чтобы проверить попадает ли персонаж в область, в которой можно поднять ключ с пола. Блок-схема алгоритма процедуры pickUpKey приведена на рисунке 2.8.



Рисунок 2.8 – Блок-схема процедуры pickUpKey

2.3.5 Осуществление способностей аватаров

Способности аватаров являются механикой, которая разбравляет программное средство и добавляет интересных активностей связанных с конкретным выбранным аватаром. В главной процедуре будет проверка на выбранного персонажа, а затем передача управления подфункциям в зависимости от текущего аватара, а далее уже непосредственно осуществление способности. Блок-схема алгоритма процедуры doSpecialSpell приведена на рисунке 2.9.



Рисунок 2.9 – Блок-схема процедуры doSpecialSpell

Стоит также упомянуть про процедуру движения персонажа. Данная процедура по существа является главной во всем программном устройстве, однако ввиду простоты осуществления она не была упомянуто в данном разделе в качестве подтемы. Суть процедуры представляет из себя проверку желаемой точки передвижения по матрице лабиринта на возможность пройти туда и ,при положительном результате, перемещение игрового персонажа как в матрице лабиринта, так и на игровом окне на желаемую позицию.

# РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА

## 3.1 Передвижение персонажа

Основой данного программного средства является алгоритм передвижения персонажа по игровому полю.

Вызов необходимой процедуры осуществляется каждый раз, как игрок нажимает на клавиши управления, сперва мы обрабатываем направление движения и пункт назначения, далее проверяем возможность движения в заданную точку в заданном направлении. Если движение возможно, то мы перемещаем персонажа и вызываем некоторые процедуры проверяющие победу, подбор ключа, выполнение способности и т.д. Код процедуры game будет предоставлен ниже.

void Game::game(int\*\* &main\_maze, int width) {

int x = 0;

int y = width/2 - 1;

int direction = 0;

sameFont.loadFromFile("C:/maze/font/font.ttf");

while(gameWindow.isOpen()) {

sf::Event event{};

int newX = x;

int newY = y;

while (gameWindow.pollEvent(event)){

sf::Vector2f mp;

if(event.type == sf::Event::Closed) {

gameWindow.close();

mainMenu::start();

}

if(event.type == sf::Event::KeyPressed) {

newX = mainHero.getX();

newY = mainHero.getY();

if(event.key.code == sf::Keyboard::Right) {

newX+= 1;

direction = 0;

}

if (event.key.code == sf::Keyboard::Left) {

newX-= 1;

direction = 1;

}

if(event.key.code == sf::Keyboard::Up) {

newY-= 1;

direction = 2;

}

if (event.key.code == sf::Keyboard::Down) {

newY+= 1;

direction = 3;

}

if (event.key.code == sf::Keyboard::R) {

if (spellCounter == 0) {

printSpellConfirm("Click Yes to use\n your char spell");

} else {

printSpellConfirm("Your spell is \n already done");

}

}

}

}

if ((newX > -1 && newY > -1) && (newX < HEIGHT && newY < WIDTH) && main\_maze[newX][newY] == end\_door && !isWin) {

main\_maze[x][y] = pass;

main\_maze[newX][newY] = hero\_maze;

mainHero.setOldX(x);

mainHero.setOldY(y);

x = newX;

y = newY;

mainHero.setX(x);

mainHero.setY(y);

isWin = true;

if (checkWinCondition()) {

Game::win();

}

} else if ((newX > -1 && newY > -1) && (newX < HEIGHT && newY < WIDTH) && (main\_maze[newX][newY] == pass || main\_maze[newX][newY] == room ||

checkDoorState(newX,newY,door1) || checkDoorState(newX,newY,door2))) {

main\_maze[x][y] = pass;

main\_maze[newX][newY] = hero\_maze;

mainHero.setOldX(x);

mainHero.setOldY(y);

x = newX;

y = newY;

}

mainHero.setX(x);

mainHero.setY(y);

if (spellStart) {

doSpecialSpell(main\_maze);

x = mainHero.getX();

y = mainHero.getY();

}

if (!(silverFound && goldenFound))

pickUpKey();

printMaze(main\_maze, HEIGHT, WIDTH,direction);

}

}

Данная процедура относится к классу Game, который отвечает за игровое окно и все взаимодействия с ним. После проверки следующего хода идёт перестановка элементов матрицы. После идёт проверка на начало способности и при результате true идёт выполнение процедуры doSpecialSpell.

После идёт подбор ключа, в случае если не всё ключи подобраны идёт вызов процедуры pickUpKey, последним следует вызов процедуры, отрисовывающей всё игровое поле и все объекты на нём. Код процедур pickUpKey и doSpecialSpell будет предоставлен ниже.

void pickUpKey() {

auto x = (float)mainHero.getX();

auto y = (float)mainHero.getY();

sf::Vector2f hero\_vector(x,y);

if (!silverFound && Game::isHover(silver\_Rect, hero\_vector)) {

maze[silver.x][silver.y] = pass;

silverFound = true;

}

if (!goldenFound && Game::isHover(gold\_Rect, hero\_vector)) {

maze[gold.x][gold.y] = pass;

goldenFound = true;

}

}

void doSpecialSpell(int\*\* &main\_maze) {

int hero\_num = mainHero.getCharNumber();

switch(hero\_num) {

case 7:

doTankSpell(main\_maze);

break;

case 8:

doBeatleSpell();

break;

default:

doRestSpell(main\_maze);

break;

}

}

Процедура pickUpKey выполняет подбор ключа, путём сверки позиции персонажа с позицией ключей. Процедура doSpecialSpell выполняет обработку менеджмента способностей, данная процедура перенаправляет программу в нужную процедуру в зависимости от характеристик выбранного аватара.

Рассмотрим одну из процедур, отвечающих за способности. За пример будет взята процеда, которая позволяет телепортировать персонажа в свободноую клетку, расстояния до которой не превышает максимально допустимое. Данная процедура называется porting. Вызов происходит через пару шагов после вызова doSpecialSpell.

void porting(float mouseX, float mouseY, int\*\* &main\_maze) {

sf::Vector2f mousePos(mouseX, mouseY);

double x = mainHero.getX() \* mltp;

double y = mainHero.getY() \* mltp;

sf::Vector2f temp2 = {(HEIGHT - 3)\*50, ((float)(WIDTH) / 2 - 6)\*50};

sf::Vector2f temp3 = {150, 300};

sf::Rect<float> rect1 = {temp2, temp3};

sf::Vector2f heroPos((float) x, (float) y);

if (abs(heroPos.x - mousePos.x) < MAX\_PORT\_DISTANCE) {

if (!Game::isHover(rect1, mousePos)) {

if (abs(heroPos.y - mousePos.y) < MAX\_PORT\_DISTANCE) {

sf::Vector2i tpCoords = findFreePlace(mousePos,main\_maze);

main\_maze[mainHero.getX()][mainHero.getY()] = pass;

mainHero.setX(tpCoords.x);

mainHero.setY(tpCoords.y);

mainHero.setOldX(tpCoords.x);

mainHero.setOldY(tpCoords.y);

main\_maze[tpCoords.x][tpCoords.y] = hero\_maze;

if (main\_maze[tpCoords.x-1][tpCoords.y-1] == room || main\_maze[tpCoords.x-1][tpCoords.y] == room

|| main\_maze[tpCoords.x][tpCoords.y-1] == room || main\_maze[tpCoords.x+1][tpCoords.y-1] == room

|| main\_maze[tpCoords.x-1][tpCoords.y+1] == room || main\_maze[tpCoords.x+1][tpCoords.y+1] == room

|| main\_maze[tpCoords.x+1][tpCoords.y] == room || main\_maze[tpCoords.x][tpCoords.y+1] == room) {

sf::Vector2i doorPos = findNearestDoor(tpCoords.x, tpCoords.y);

openDoor(main\_maze,doorPos.x, doorPos.y);

}

spellCounter = 1;

spellStart = false;

}

}

}

}

**3.2** Создание поля

При запуске игрового окна происходит рандомная генерация лабиринта. Данная процедура вызывается всего лишь раз при запуске игрового поля при его открытии. Однако данная процедура является одной из важнейших, так как она реализует разный лабиринт при каждом запуске игры, что привносит разнообразие в игровой процесс. Часть кода процедуры mazemake будет показана ниже.

x = 3; y = 3; a = 0;

while(a < 10000){

main\_maze[y][x] = pass; a++;

while(true){

c = rand()%4;

switch(c){

case 0: if(y != 1)

if(main\_maze[y-2][x] == wall){

main\_maze[y-1][x] = pass;

main\_maze[y-2][x] = pass;

y-=2;

}

case 1: if(y != height-2)

if(main\_maze[y+2][x] == wall){

main\_maze[y+1][x] = pass;

main\_maze[y+2][x] = pass;

y+=2;

}

case 2: if(x != 1)

if(main\_maze[y][x-2] == wall){

main\_maze[y][x-1] = pass;

main\_maze[y][x-2] = pass

x-=2;

}

case 3: if(x != width-2)

if(main\_maze[y][x+2] == wall){

main\_maze[y][x+1] = pass;

main\_maze[y][x+2] = pass;

x+=2;

}

default:

break;

}

if(deadEnd(x,y,main\_maze,height,width))

break;

}

if(deadEnd(x,y,main\_maze,height,width))

do{

x = 2\*(rand()%((width-1)/2))+1;

y = 2\*(rand()%((height-1)/2))+1;

} while(main\_maze[y][x] != pass);

}

Имеет место указать код процедуры deadEnd, которые позволяет делать лабиринт целостным и заполнять его проходами и тупиками.

bool deadEnd(int x, int y, int\*\* main\_maze, int height, int width){

int a = 0;

if(x != 1){

if(main\_maze[y][x-2] == pass||

main\_maze[y][x-2] == room)

a+=1;

}

else a+=1;

if(y != 1){

if(main\_maze[y-2][x] == pass||

main\_maze[y-2][x] == room)

a+=1;

}

else a+=1;

if(x != width-2){

if(main\_maze[y][x+2] == pass||

main\_maze[y][x+2] == room)

a+=1;

}

else a+=1;

if(y != height-2){

if(main\_maze[y+2][x] == pass||

main\_maze[y+2][x] == room)

a+=1;

}

else a+=1;

if(a == 4)

return true;

else

return false;

}

**3.3** Победа

Окно победы появляется при соблюдении условий победы. Окно состоит из заднего фона и единственный кнопки “OK”, при нажатии на которую окно победы закрывается, следом за ним также закрывается и игровое окно. Код процедуры win будет приведён ниже.

void Game::win() {

sf::RenderWindow winWindow;

winWindow.create(sf::VideoMode(700,300), "You WIN!!!", sf::Style::None);

printWinBg(&winWindow);

showOkBtn(&winWindow);

winWindow.display();

while (winWindow.isOpen()) {

sf::Event event{};

while (winWindow.pollEvent(event)) {

if (event.type == sf::Event::MouseButtonPressed) {

if (event.mouseButton.button == sf::Mouse::Left) {

winWindow.close();

gameWindow.close();

mainMenu::start();

}

}

}

}

}

Следует отметить, что большая часть всех процедур и функций в данном программном средстве представляют собой вывод на экран определённых компонентов библиотеки SFML. Данные методы являются небольшими в размере и представляют собой в общем случае инициализацию спрайта или текста поддерживаемых SFML, а затем рисование данных компонент на окно, передаваемое в данную процедуру как указатель на объект. Также можно отметить наличие вспомогательных функций. Их назначение может быть разным, например: создание пути для файла персонажа, нахождение ближайшего свободного пути, нахождение ближайшей двери, проверка победных условий, проверка попадания курсора на передаваемый объект, выбор направления движения.

# ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

В ходе тестирования приложения были выявлены некоторые ошибки и недочеты в работе программного средства.

Первый недочёт. При запуске игры на небольшой промежуток времени можно было увидеть всю карту, не закрытую туманом войны, данный недочёт является довольно критическим, так как игрок сразу мог заметить, где находится серебряный ключ и в какой из комнат лежит золотой ключ. Изображение проблемы будет представлено на рисунке 4.1.

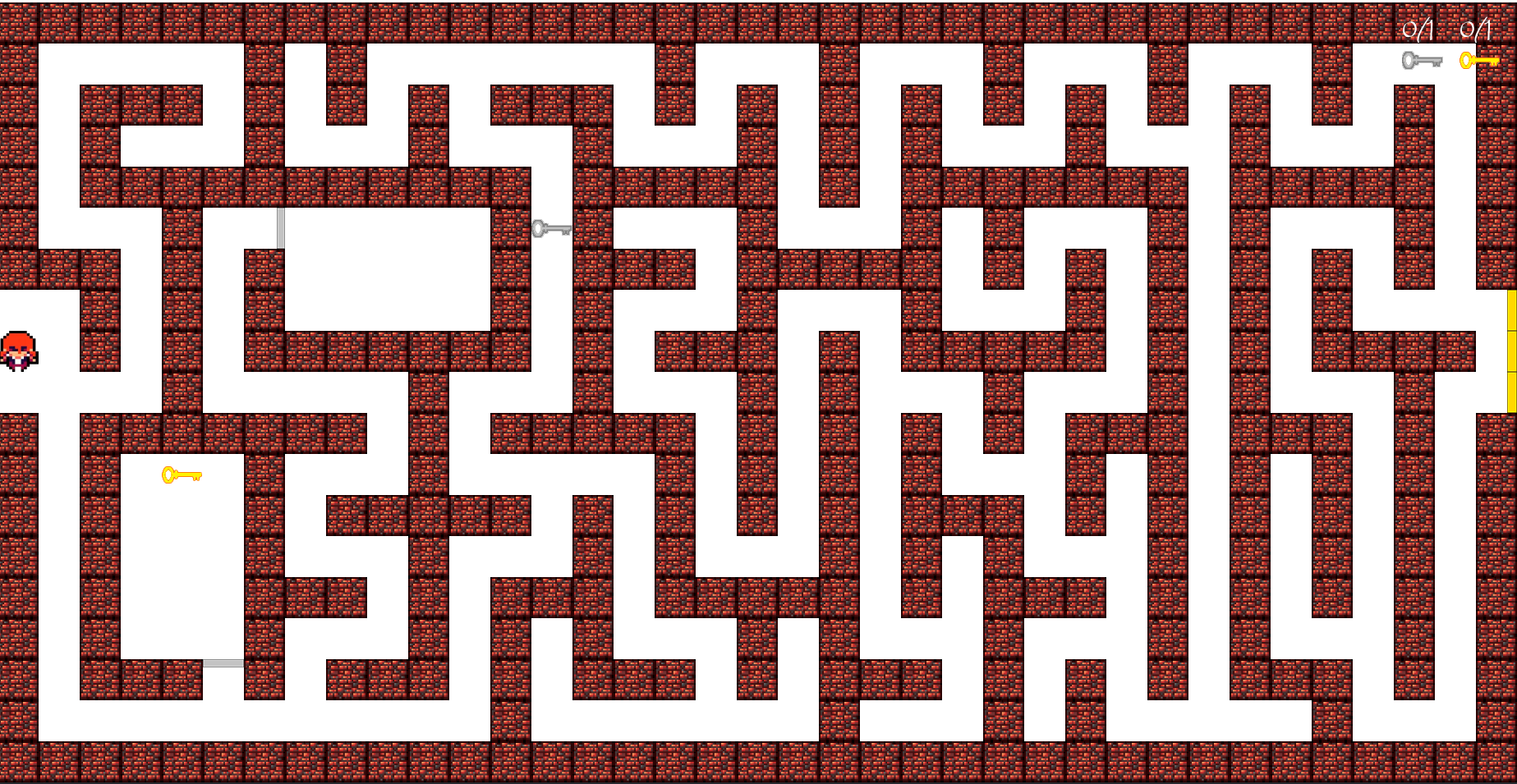


Рисунок 4.1 – Проблема появления открытого поля

Происходит это из-за того, что при запуске игрового поля используется другая функция вывода лабиринта на экран, которая ещё и создаёт рандомный лабиринт. Однако в программном средстве существует подобная процедура, которая не создаёт лабиринт и вдобавок выводит на экран туман войны. Решением стало то, что старая процедура с генерацией была удалена, а инициализация лабиринта была добавлена в метод Game::print, который вызывается при запуске окна игры.

maze = new int\*[HEIGHT];

for (int i = 0; i < HEIGHT; ++i) {

maze[i] = new int[WIDTH];

}

mazemake(maze, HEIGHT, WIDTH);

Существовала также ещё одна довольно неприятная проблема. Суть её заключается в том, что некоторые персонажи обладают способностью телепортироваться в место клика мышью. Данная способность также поддерживает возможность телепортироваться в комнату без наличия ключа, однако тогда возникает проблема. И тогда возникает вопрос того, как выйти из комнаты, если дверь закрыта, а ключа у персонажа нету. Проблема представлена на рисунке 4.2.

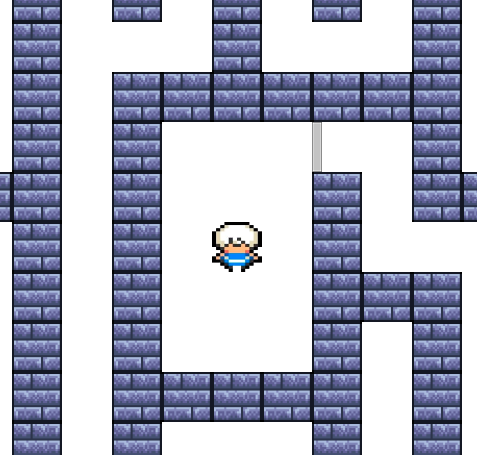


Рисунок 4.2 – Персонаж заперт в комнате

Данная проблема была решена довольно тривиальным способом. В код программы были внесены коррективы, которые при телепортации проверяют положение персонажа на поле, и если персонаж находится в комнате, то сперва вызывается процедура, находящая ближайшую к персонажу дверь из комнаты, а потом координаты найденной двери передаются в процедуру, открывающую дверь по её координатам. Корректирующий код будет приведён ниже.

void openDoor(int\*\* &main\_maze, int doorI, int doorJ) {

if (door1.i == doorI && door1.j == doorJ) {

door1.isOpen = true;

} else {

door2.isOpen = true;

}

main\_maze[doorI][doorJ] = pass;

}

sf::Vector2i findNearestDoor(int i, int j) {

int n = 0;

sf::Vector2i result;

do{

n++;

for (int k=0; k<n; k++) {

if (maze[i][j] == room\_door) {

result.x = i;

result.y = j;

return result;

}

i--;

}

for (int k=0; k<n; k++) {

if (maze[i][j] == room\_door) {

result.x = i;

result.y = j;

return result;

}

j++;

}

n++;

for (int k=0; k<n; k++) {

if (maze[i][j] == room\_door) {

result.x = i;

result.y = j;

return result;

}

i++;

}

for (int k=0; k<n; k++) {

if (maze[i][j] == room\_door) {

result.x = i;

result.y = j;

return result;

}

j--;

}

}

while (true);

}

Самое главное в тестировании, это исправить все проблемы и ошибки в программе до того, как она выйдет в свет. Можно считать, что в данном случае задача тестирования была выполнена успешно.

# РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

## 5.1 Интерфейс программного средства

5.1.1 Главное окно

Главное окно представляет собой меню, в котором пользователь может осуществить выбор одной из опций. “Start Game” непосредственно запускает игру с рандомным персонажем, “About” выведет информацию о создателе, “Choose Character” откроет новое окно, в котором необходимо выбрать аватара на свой вкус, “Instruction” содержит данные об управлении и сведения о способностях, “Exit Game” позволяет выйти из приложения. Внешний вид главного окна приложения представлен на рисунке 5.1.



Рисунок 5.1 – Главное окно приложения

5.1.2 Инструкция

Элемент меню главного окна “Instruction” можно вызвать нажатием на соответствующий текст левой кнопкой мыши. После нажатия перед пользователем откроется окно, отображающее информацию об управлении и способностях. Внешний вид окна инструкции представлен на рисунке 5.2.



Рисунок 5.2 – Инструкция приложения

5.1.3 Выбор персонажа

Пункт меню “Choose Character” открывается при нажатии на соответствующий текст в главном меню. При нажатии перед пользователем всплывает новое окно, в котором изображены все 8 персонажей. При нажатии на любого из них начнётся игра с выбранным персонажем. Также есть иконка вопроса, при нажатии на которую открывается окно “Instruction”. Внешний вид окна выбора персонажа представлен на рисунке 5.3.



Рисунок 5.3 – Таблица рекордов

5.1.4 О разработчике

Элемент меню “About” открывает новое окно, в котором показана информация о разработчике данной игры. Внешний вид окна информации о разработчике представлен на рисунке 5.4.



Рисунок 5.4 – Информация о разработчике

**5.1.5** Игровое поле

Окно игрового поля представляет собой лабиринт, закрытый черной пеленой. Видимость есть только вокруг персонажа на небольшом расстоянии, вход в лабиринт всегда находится с левой части экрана, а выход всегда находится в правой части экрана. Внешний вид игрового поля можно посмотреть на рисунке 5.5.

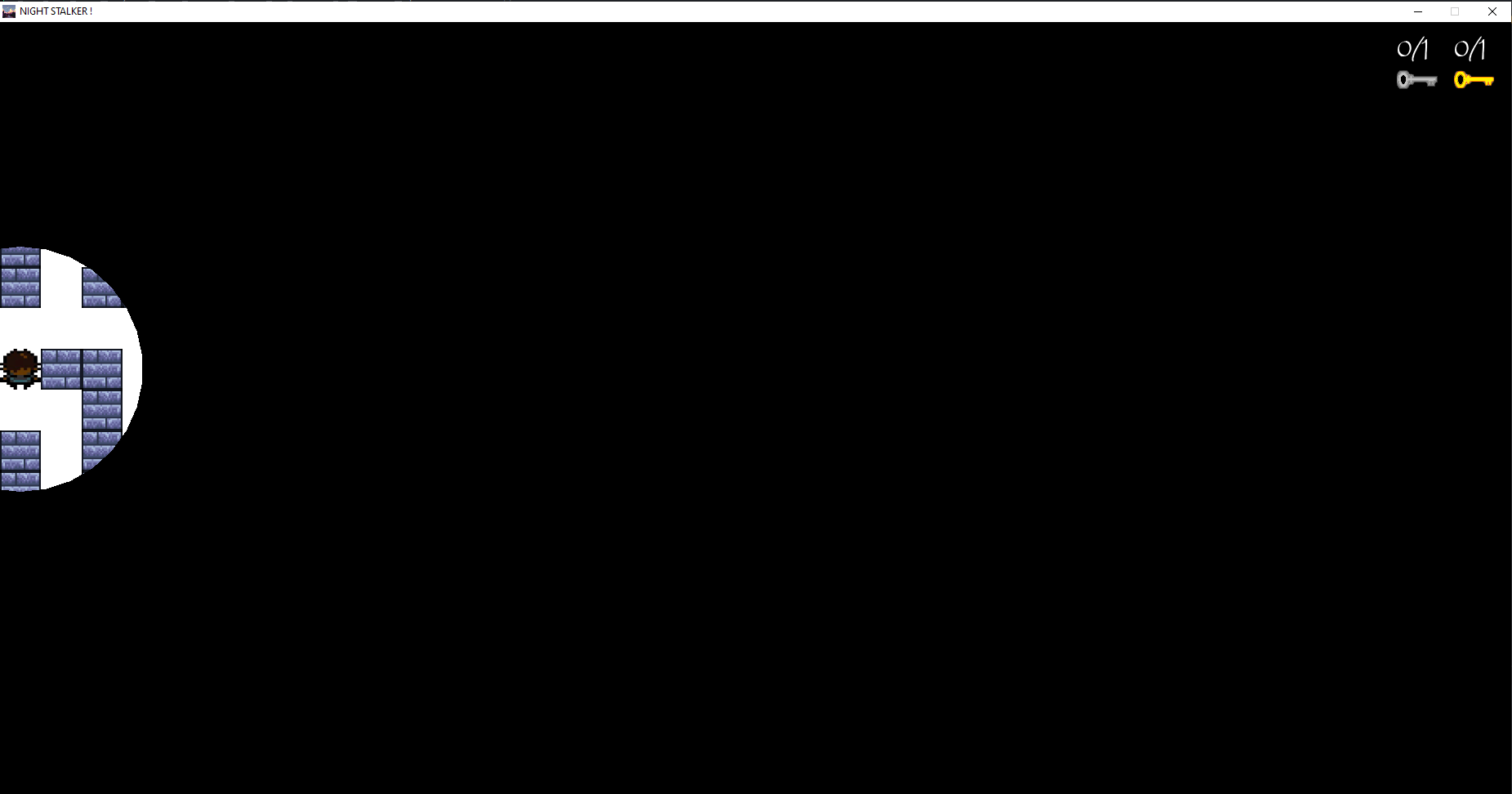


Рисунок 5.5 — Внешний вид игрового поля

## 5.2 Управление программным средством

Начать игру можно 2 путями: нажать на текст “Start Game” в главном меню, открыть окно “Choose Character” и нажать на интересующего аватара. После запуска игры перед пользователем предстанет игровая область, на которой расположен персонаж, небольшая видимая область вокруг него и невидимая область, занимающая всё остальное место. Также есть 2 индикатора ключей, расположенные справа сверху, они показывают сколько подобрано серебряных и золотых ключей. Управление персонажем осуществляется засчёт стрелочек, расположенных на клавиатуре, стрелочка вверх перемещает персонаж вверх и т.д. Использование особо умения осуществляется на кнопку “R”. Последующее управление способностью зависит от выбранного персонажа. Если выбран танк, то следует нажать одну из стрелок, чтобы задать направление выстрела в стену, однако так, чтобы стена была одиночной. Если выбран жук, то следует нажать левой кнопкой мыши на любую точку на игровом поле, после этого большая область вокруг точки станет видна на некоторое время, а после снова закроется туманом войны. Если выбран любой другой персонаж, то следует нажать мышкой на точку на игровом поле, куда вы желаете переместиться, после этого программа автоматически найдёт ближайшее к этой точке свободное место(комната/проход) и затем телепортирует персонажа в данную точку. Способность доступна лишь 1 раз за игру, чтобы обновить счётчик нужно начать новую игру. Прохождение достигается путём сбора золотого ключа, который расположен в одной из 2 комнат случайным образом. Доступ к комнатам осуществляется путём подбора серебряного ключа или использования способностей танка или других персонажей кроме жука. В результате победы пользователь увидит небольшое окно, оповещающее о победе, на котором имеется кнопка “OK”, по нажатии на которую игровое поле и само окно победы закроются, и на экране останется только главное меню игрового средства.

# ЗАКЛЮЧЕНИЕ

В заключении можно сказать, что была проделана хорошая работа по созданию данного программного средства, отдельно хочется выделить визуальную составляющую, а именно фоновые изображения различных окон. Также концепция тумана войны, когда часть или всё игровое поле скрыто от игрока, является довольно популярной и в то же время интересной, так как данная концепция требует от игрока затраты некоторых умственных сил на запоминание уже пройденного лабиринта. В рамках данного курсового проекта было разработано программное средство “Лабиринт”, которое позволит пользователям операционной системы Windows играть в данную игру. При разработке данного средства были успешно выполнены следующие поставленные задачи:

* реализация концепции тумана войны;
* случайная генерация лабиринта;
* показ победного окна;
* корректное исполнение способностей персонажей;
* выбор из нескольких персонажей;
* реализация системы ключей для усложнения;
* перезапуск игры.

Для успешного выполнения поставленных задач потребовалось изучить

основные принципы работы с директориями и файловыми компонентами, библиотеками ресурсов, динамическим созданием компонентов, использованием библиотеки SFML.

Существует множество способов улучшить данное программное средство: добавить более длинное поле, добавить звуковое сопровождение, добавить вражеских существ в лабиринт, добавить больше способностей.

Игровое средство “Лабиринт” позволяет людям весело и с интересом провести время за игрой.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Глухова, Л. А. Основы алгоритмизации и программирования: Лаб. практикум для студ. спец. I-40 01 01 «Программное обеспечение информационных технологий» дневной формы обуч. В 4 ч. / Л. А. Глухова, Е. Е. Фадеева, Е. П. Фадеева. – Минск: БГУИР, 2007. – Ч. 3. – 51 с.

[2] Артур М. SFML Game Development. – Birmingham, 2013. – 296c.

[3] Майкл Д. Изучаем C++ через программирование игр. – СПб.: Питер. 2015. – 450с.

[4] Орлов, С. А. Технологии разработки программного обеспечения: учеб. Пособие. – СПб, 2003. – 321 с.

[5] Уилсон, С. Принципы проектирования и разработки программного обеспечения, yчебн. курс. – СПб, 2003. – 570 с.

[6] https://www.sfml-dev.org/documentation/2.5.1/ – Документация к SFML.

[7] Петцольд 4. Programming Windows®, 1998. - СПб.: БХВ-Петербург, 2007. – 1448 c.

[8] Алгоритмы. Теория и практические применение / Род Стивенс. - Москва, 2016. – 544 с.

[9] Мартин Р. Чистый код: создание, анализ и рефакторинг. Библиотека программиста. - СПб.: Питер. 2018. – 464 с.

[10] Пол А. Объектно-ориентированное программирование на С++. – Москва, 2017. – 476с.

[11] Лафоре Р. Объектно-ориентированное программирование в C++. 4-е полное изд. – СПб.: Питер, 2018. – 928 с.

# ПРИЛОЖЕНИЕ А

**Исходный код программы**

#include "mainMenu.h"

#include <SFML/Graphics.hpp>

#include <windows.h>

#include "../choose/choose.h"

#include <random>

#include "../game/Game.h"

const int W = 40;

const int H = 28;

const int mltp = 30;

const int FONT\_SIZE = 50;

sf::RenderWindow startWindow;

sf::Font mainFont;

sf::Text ExitGame;

sf::Text startGame;

sf::Text AboutGame;

sf::Text ChooseChr;

sf::RenderWindow exitWindow;

sf::Text YesLbl;

sf::Text NoLbl;

sf::Text helpText;

sf::Sprite exitIco;

sf::RenderWindow aboutWindow;

bool isHover(sf::FloatRect sprite, sf::Vector2f mp){

if(sprite.contains(mp)) {

return true;

} else {

return false;

}

}

void printBgSprite () {

sf::Texture bg;

bg.loadFromFile("C:/maze/img/utility/phon.png");

sf::Sprite background(bg);

background.setPosition(0,0);

startWindow.draw(background);

}

void printStartText () {

startGame.setString("Start Game");

startGame.setFont(mainFont);

startGame.setCharacterSize(FONT\_SIZE);

startGame.setPosition((W \* mltp) / 2.0 - 90.0,220);

startWindow.draw(startGame);

}

void printAboutText () {

AboutGame.setString("About");

AboutGame.setFont(mainFont);

AboutGame.setCharacterSize(50);

AboutGame.setPosition((W \* mltp) / 2.0 - 50,220 + AboutGame.getCharacterSize()\*1.2);

startWindow.draw(AboutGame);

}

void printExitText () {

ExitGame.setString("Exit Game");

ExitGame.setFont(mainFont);

ExitGame.setCharacterSize(50);

ExitGame.setPosition((W \* mltp) / 2.0 - 76.0,220 + ExitGame.getCharacterSize()\*4.9);

startWindow.draw(ExitGame);

}

void printChooseText () {

ChooseChr.setString("Choose Character");

ChooseChr.setFont(mainFont);

ChooseChr.setCharacterSize(50);

ChooseChr.setPosition((W \* mltp) / 2.0 - 145,220 + ChooseChr.getCharacterSize()\*2.3);

startWindow.draw(ChooseChr);

}

void printInfoText () {

helpText.setString("Instruction");

helpText.setFont(mainFont);

helpText.setCharacterSize(50);

helpText.setPosition((W \* mltp) / 2.0 - 100,220 + helpText.getCharacterSize()\*3.6);

startWindow.draw(helpText);

}

void mainMenu::putIcoInWindow(sf::RenderWindow \*window) {

sf::Image img;

img.loadFromFile("C:/maze/img/newico.png");

window->setIcon(img.getSize().x, img.getSize().y, img.getPixelsPtr());

}

void mainMenu::printExitBtn(sf::RenderWindow \*window, int x, int y) {

sf::Image questionImg;

questionImg.loadFromFile("C:/maze/img/utility/close.png");

sf::Texture questionTexture;

questionTexture.loadFromImage(questionImg);

exitIco.setTexture(questionTexture);

exitIco.setScale(0.5,0.5);

exitIco.setPosition(x, y);

window->draw(exitIco);

}

void mainMenu::info() {

sf::RenderWindow infoWindow(sf::VideoMode(1000, 800), "Info", 0);

sf::Texture arrows;

sf::Text allText;

sf::Sprite chr\_sprite;

allText.setFont(mainFont);

allText.setFillColor(sf::Color(10,10,10));

allText.setCharacterSize(65);

sf::Texture bg;

bg.loadFromFile("C:/maze/img/utility/infoBg2.png");

sf::Sprite background(bg);

background.setPosition(0,0);

infoWindow.draw(background);

arrows.loadFromFile("C:/maze/img/utility/mvm.png");

sf::Sprite arrowSprite(arrows);

arrowSprite.setPosition(60,25);

arrowSprite.setScale(0.4,0.4);

allText.setString(" Use this keys to\n move your character");

allText.setPosition(500, 10);

infoWindow.draw(arrowSprite);

infoWindow.draw(allText);

mainMenu::printExitBtn(&infoWindow, 970, 10);

allText.setString("SPELLS");

allText.setCharacterSize(90);

allText.setPosition(400, 130);

infoWindow.draw(allText);

sf::Texture chr\_texture;

chr\_texture.loadFromFile("C:/maze/chars/char7/up.png");

chr\_sprite.setTexture(chr\_texture);

chr\_sprite.setScale(0.6,0.6);

chr\_sprite.setPosition(100, 250);

allText.setString(" Tank can destroy single wall with shot\nto use your spell you should stand near the\n wall "

"then press R and choose the direction.");

allText.setCharacterSize(42);

allText.setPosition(340, 250);

infoWindow.draw(chr\_sprite);

infoWindow.draw(allText);

sf::Sprite chr\_sprite2;

sf::Sprite chr\_sprite3;

chr\_texture.loadFromFile("C:/maze/chars/char8/up.png");

chr\_sprite2.setTexture(chr\_texture);

chr\_sprite2.setScale(0.75,0.70);

chr\_sprite2.setPosition(100, 430);

allText.setString(" The beatle can reveal wide circle\n area for three seconds to use your\nspell press R then click on the screen");

allText.setCharacterSize(42);

allText.setPosition(380, 430);

infoWindow.draw(chr\_sprite2);

infoWindow.draw(allText);

sf::Rect<float> exit{{970,10}, {30,30}};

std::string path;

for (int i = 1; i < 7; i++) {

path = "c:/maze/img/chars/char";

path += std::to\_string(i);

path+= ".png";

chr\_texture.loadFromFile(path);

chr\_sprite3.setTexture(chr\_texture);

chr\_sprite3.setScale(1.25,1.25);

if (i < 4)

chr\_sprite3.setPosition(50 + (i - 1) \* 70, 630);

else

chr\_sprite3.setPosition(50 + (i - 4) \* 70, 700);

infoWindow.draw(chr\_sprite3);

}

allText.setString(" The rest of chars can teleport\nto teleport click mouse in point and\n"

"your char will tp in nearest free cell\n the max tp range is eight hundreds");

allText.setCharacterSize(40);

allText.setPosition(400, 600);

infoWindow.draw(allText);

infoWindow.display();

while (infoWindow.isOpen()) {

sf::Event event{};

while (infoWindow.pollEvent(event)) {

if (event.type == sf::Event::MouseButtonPressed) {

if (event.mouseButton.button == sf::Mouse::Left) {

if (Game::isHover(exit, static\_cast<sf::Vector2f>(sf::Mouse::getPosition(infoWindow)))) {

infoWindow.close();

}

}

}

}

}

}

void mainMenu::print() {

mainMenu::putIcoInWindow(&startWindow);

printBgSprite();

printStartText();

printAboutText();

printExitText();

printChooseText();

printInfoText();

}

int mainMenu::prerun() {

srand(time(NULL));

int num = rand()%7;

return num+1;

}

void mainMenu::start() {

startWindow.create(sf::VideoMode(W \* mltp, H \* mltp), "NIGHT STALKER !", sf::Style::None);

mainFont.loadFromFile("C:/maze/font/font.ttf");

print();

startWindow.display();

while(startWindow.isOpen()) {

sf::Event event{};

while (startWindow.pollEvent(event)){

sf::Vector2f mp;

mp.x = (float)sf::Mouse::getPosition(startWindow).x;

mp.y = (float)sf::Mouse::getPosition(startWindow).y;

if(event.type == sf::Event::Closed) {

mainMenu::exit();

}

if(event.type == sf::Event::MouseButtonPressed) {

if (event.mouseButton.button == sf::Mouse::Left) {

if(isHover(ExitGame.getGlobalBounds(), mp)) {

mainMenu::exit();

} else if (isHover(AboutGame.getGlobalBounds(), mp)) {

mainMenu::about();

} else if (isHover(ChooseChr.getGlobalBounds(),mp)) {

choose::print();

} else if(isHover(startGame.getGlobalBounds(), mp)) {

startWindow.setVisible(false);

Game::prerun(mainMenu::prerun());

} else if(isHover(helpText.getGlobalBounds(), mp)) {

mainMenu::info();

}

}

}

}

sf::Vector2f mp;

mp.x = (float)sf::Mouse::getPosition(startWindow).x;

mp.y = (float)sf::Mouse::getPosition(startWindow).y;

}

}

void printExitAsk() {

sf::Text exitAsk;

exitAsk.setString("Are you really\n want to exit");

exitAsk.setFont(mainFont);

exitAsk.setCharacterSize(45);

exitAsk.setPosition(95, 30);

exitWindow.draw(exitAsk);

sf::Image questionImg;

questionImg.loadFromFile("C:/maze/img/utility/question.png");

sf::Texture questionTexture;

questionTexture.loadFromImage(questionImg);

sf::Sprite question;

question.setTexture(questionTexture);

question.setPosition(219 + 85 , exitAsk.getGlobalBounds().top + exitAsk.getGlobalBounds().height - 23);

exitWindow.draw(question);

}

void printYesNo() {

YesLbl.setString("YES");

YesLbl.setFont(mainFont);

YesLbl.setCharacterSize(45);

YesLbl.setPosition(65, 140);

exitWindow.draw(YesLbl);

NoLbl.setString("NO");

NoLbl.setFont(mainFont);

NoLbl.setCharacterSize(45);

NoLbl.setPosition(335 -NoLbl.getGlobalBounds().width, 140);

exitWindow.draw(NoLbl);

}

void mainMenu::printExitBgSprite (sf::RenderWindow \*window) {

sf::Texture bg;

bg.loadFromFile("C:/maze/img/utility/exit\_bg.png");

sf::Sprite background(bg);

background.setPosition(0,0);

window->draw(background);

}

sf::Vector2i randomlyChoose(int width, int height) {

srand(time(nullptr));

float x = YesLbl.getGlobalBounds().left + 10;

float y = YesLbl.getGlobalBounds().top + 10;

sf::Vector2f tempVec = {x,y};

while (isHover(YesLbl.getGlobalBounds(), tempVec)) {

x = rand()%width;

y = rand()%height;

tempVec = {x,y};

}

int newX = floor(tempVec.x);

int newY = floor(tempVec.y);

sf::Vector2i newMouse = {newX,newY};

return newMouse;

}

void mainMenu::exit() {

exitWindow.create(sf::VideoMode(400,210), "Exit?",sf::Style::None);

printExitBgSprite(&exitWindow);

printExitAsk();

printYesNo();

mainMenu::putIcoInWindow(&exitWindow);

exitWindow.display();

srand(time(nullptr));

int repeatTimes = 0;

repeatTimes = rand()%2;

while(exitWindow.isOpen()) {

sf::Event event;

while(exitWindow.pollEvent(event)) {

if (event.type == sf::Event::Closed) {

exitWindow.close();

}

if(event.type == sf::Event::MouseButtonPressed) {

if (event.mouseButton.button == sf::Mouse::Left) {

sf::Vector2f mp;

mp.x = (float)sf::Mouse::getPosition(exitWindow).x;

mp.y = (float)sf::Mouse::getPosition(exitWindow).y;

if(isHover(YesLbl.getGlobalBounds(), mp)) {

if (repeatTimes != 0) {

sf::Vector2i newMouse = randomlyChoose(400, 210);

sf::Mouse::setPosition(newMouse, exitWindow);

repeatTimes--;

} else {

exitWindow.close();

startWindow.close();

}

}

if(isHover(NoLbl.getGlobalBounds(), mp)) {

exitWindow.close();

}

}

}

}

}

}

void printAboutInfo() {

sf::Text aboutInfo;

aboutInfo.setString("\t\tDeveloper\n BSUIR POIT student\n Prokopchuk Daniil");

aboutInfo.setFont(mainFont);

aboutInfo.setCharacterSize(45);

aboutInfo.setPosition(20, 22);

aboutWindow.draw(aboutInfo);

}

void mainMenu::about() {

aboutWindow.create(sf::VideoMode(400,210), "About",sf::Style::None);

printExitBgSprite(&aboutWindow);

mainMenu::printExitBtn(&aboutWindow, 370, 10);

mainMenu::putIcoInWindow(&aboutWindow);

printAboutInfo();

aboutWindow.display();

while(aboutWindow.isOpen()) {

sf::Event event{};

while (aboutWindow.pollEvent(event)) {

if (event.type == sf::Event::MouseButtonPressed) {

if (event.mouseButton.button == sf::Mouse::Left) {

sf::Vector2f mp((float)sf::Mouse::getPosition(aboutWindow).x, (float)sf::Mouse::getPosition(aboutWindow).y);

if (isHover(exitIco.getGlobalBounds(), mp)) {

aboutWindow.close();

}

}

}

}

}

}

#include "Game.h"

#include <SFML/Graphics.hpp>

#include "../hero/Hero.h"

#include "../main/mainMenu.h"

#include <cmath>

#include "../door/door.h"

typedef struct drct{

std::string left = "left";

std::string right = "right";

std::string up = "up";

std::string down = "down";

}Directions;

const int W = 37;

const int H = 19;

const int mltp = 50;

const int pass = 1;

const int room = 4;

const int wall = 0;

const int hero\_maze = 5;

const int room\_door = 3;

const int end\_door = 6;

const int silver\_key = 2;

const int gold\_key = 10;

const int HEIGHT = 37;

const int WIDTH = 19;

const int MAX\_PORT\_DISTANCE = 800;

int spellCounter = 0;

bool spellStart = false;

typedef struct key{

int x;

int y;

}key\_info;

key\_info silver;

key\_info gold;

bool isWin = false;

door door1(0,0);

door door2(0,0);

int\*\* maze;

sf::RenderWindow gameWindow;

sf::Font sameFont;

sf::Font digitsFont;

sf::Sprite exit\_spell;

sf::Text yesLbl;

sf::Sprite hero;

sf::Texture hero\_texture;

sf::Image hero\_Img;

sf::Sprite keySprite;

Hero mainHero(0,0,0,0,0);

std::string wall\_path;

sf::Rect<float> silver\_Rect;

sf::Rect<float> gold\_Rect;

sf::Text okText;

const std::string keyPos[] = {"0/1","1/1"};

bool silverFound = false;

bool goldenFound = false;

void printKeysData() {

sf::Text key;

if (silverFound)

key.setString(keyPos[1]);

else

key.setString(keyPos[0]);

key.setCharacterSize(30);

digitsFont.loadFromFile("C:/maze/font/font3.ttf");

key.setFont(digitsFont);

key.setPosition(1710, 10);

gameWindow.draw(key);

if (goldenFound)

key.setString(keyPos[1]);

else

key.setString(keyPos[0]);

key.setCharacterSize(30);

key.setPosition(1780, 10);

gameWindow.draw(key);

sf::Image keyImg;

keyImg.loadFromFile("C:/maze/img/utility/key2.png");

sf::Texture keyTexture;

keyTexture.loadFromImage(keyImg);

keySprite.setTexture(keyTexture);

keySprite.setPosition(1710,45);

gameWindow.draw(keySprite);

keyImg.loadFromFile("C:/maze/img/utility/key.png");

keyTexture.loadFromImage(keyImg);

keySprite.setTexture(keyTexture);

keySprite.setPosition(1780,45);

gameWindow.draw(keySprite);

}

std::string chooseWallImg() {

srand(time(nullptr));

std::string path = "C:/maze/img/walls/";

int num = rand()%13+1;

path += std::to\_string(num) + ".png";

return path;

}

bool deadEnd(int x, int y, int\*\* main\_maze, int height, int width){

int a = 0;

if(x != 1){

if(main\_maze[y][x-2] == pass||

main\_maze[y][x-2] == room)

a+=1;

}

else a+=1;

if(y != 1){

if(main\_maze[y-2][x] == pass||

main\_maze[y-2][x] == room)

a+=1;

}

else a+=1;

if(x != width-2){

if(main\_maze[y][x+2] == pass||

main\_maze[y][x+2] == room)

a+=1;

}

else a+=1;

if(y != height-2){

if(main\_maze[y+2][x] == pass||

main\_maze[y+2][x] == room)

a+=1;

}

else a+=1;

if(a == 4)

return true;

else

return false;

}

void pickUpKey() {

auto x = (float)mainHero.getX();

auto y = (float)mainHero.getY();

sf::Vector2f hero\_vector(x,y);

if (!silverFound && Game::isHover(silver\_Rect, hero\_vector)) {

maze[silver.x][silver.y] = pass;

silverFound = true;

}

if (!goldenFound && Game::isHover(gold\_Rect, hero\_vector)) {

maze[gold.x][gold.y] = pass;

goldenFound = true;

}

}

void mazemake(int\*\* main\_maze, int height, int width){

int x, y, c, a;

bool b;

for(int i = 0; i < height; i++)

for(int j = 0; j < width; j++)

main\_maze[i][j] = wall;

int rheight = 3, rwidth = 5;

rheight--;rwidth--;

int room\_num = 2;

bool swap = true;

int gold\_key\_room = rand()%2;

for(int l = 0; l < room\_num; l++){

b = true;

while(b){

do{

if(rwidth%4 == 0)

x = 2\*(rand()%(width/2))+1;

else

x = 2\*(rand()%(width/2))+2;

if(rheight%4 == 0)

y = 2\*(rand()%(height/2))+1;

else

y = 2\*(rand()%(height/2))+2;

}

while(x < (rwidth+2) || x > (width-rwidth-2) ||

y < (rheight+2) || y > (height-rheight-2));

b = false;

for(int i = (y-rheight-2); i < (y+rheight+2); i++)

for(int j = (x-rwidth-2); j < (x+rwidth+2); j++)

if(main\_maze[i][j] == room)

b = true;

if(b)

continue;

int temp\_x = rand()%rheight + (y - rheight/2);

int temp\_y = rand()%rwidth + (x - rwidth/2);

for(int i = (y-rheight/2); i < (y+rheight/2+1); i++)

for(int j = (x-rwidth/2); j < (x+rwidth/2+1); j++)

main\_maze[i][j] = room;

if (gold\_key\_room == l) {

main\_maze[temp\_x][temp\_y] = gold\_key;

gold.x = temp\_x; gold.y = temp\_y;

sf::Vector2f temp = {(float)temp\_x-1,(float)temp\_y-1};

sf::Vector2f size = {3,3};

gold\_Rect = sf::Rect<float>(temp,size);

}

c = rand()%4;

if(c == 0) {

main\_maze[y+rheight/2+1][x-rwidth/2+2\*(rand()%(rwidth/2+1))] = room\_door;

}

if(c == 1) {

main\_maze[y - rheight / 2 - 1][x - rwidth / 2 + 2 \* (rand() % (rwidth / 2 + 1))] = room\_door;

}

if(c == 2) {

main\_maze[y - rheight / 2 + 2 \* (rand() % (rheight / 2 + 1))][x + rwidth / 2 + 1] = room\_door;

}

if(c == 3) {

main\_maze[y - rheight / 2 + 2 \* (rand() % (rheight / 2 + 1))][x - rwidth / 2 - 1] = room\_door;

}

if(swap){

rheight += rwidth;

rwidth = rheight - rwidth;

rheight -= rwidth;

}

}

}

int doors = 2;

for (int i = 0; i < height; i++) {

for (int j = 0; j < width; j++) {

if (main\_maze[i][j] == room\_door) {

if (doors > 1) {

door1.i = i;

door1.j = j;

} else {

door2.i = i;

door2.j = j;

}

doors--;

}

}

}

x = 3; y = 3; a = 0;

while(a < 10000){

main\_maze[y][x] = pass; a++;

while(true){

c = rand()%4;

switch(c){

case 0: if(y != 1)

if(main\_maze[y-2][x] == wall){

main\_maze[y-1][x] = pass;

main\_maze[y-2][x] = pass;

y-=2;

}

case 1: if(y != height-2)

if(main\_maze[y+2][x] == wall){

main\_maze[y+1][x] = pass;

main\_maze[y+2][x] = pass;

y+=2;

}

case 2: if(x != 1)

if(main\_maze[y][x-2] == wall){

main\_maze[y][x-1] = pass;

main\_maze[y][x-2] = pass;

x-=2;

}

case 3: if(x != width-2)

if(main\_maze[y][x+2] == wall){

main\_maze[y][x+1] = pass;

main\_maze[y][x+2] = pass;

x+=2;

}

default:

break;

}

if(deadEnd(x,y,main\_maze,height,width))

break;

}

if(deadEnd(x,y,main\_maze,height,width))

do{

x = 2\*(rand()%((width-1)/2))+1;

y = 2\*(rand()%((height-1)/2))+1;

} while(main\_maze[y][x] != pass);

}

bool notPlaced = true;

while(notPlaced) {

x = rand()%11+9;

y = rand()%16+1;

notPlaced = maze[x][y] != pass;

}

main\_maze[x][y] = silver\_key;

silver.x = x; silver.y = y;

sf::Vector2f temp = {(float)x-1,(float)y-1};

sf::Vector2f size = {3,3};

silver\_Rect = sf::Rect<float>(temp,size);

main\_maze[0][width/2] = pass;

main\_maze[0][width/2-2] = pass;

main\_maze[0][width/2-1] = hero\_maze;

main\_maze[height-1][width/2-1] = end\_door;

main\_maze[height-1][width/2-2] = end\_door;

main\_maze[height-1][width/2] = end\_door;

}

std::string getHeroImgPath(int direction) {

std::string hero\_path = "C:/maze/img/chars/";

hero\_path += "char";

if ( mainHero.getCharNumber() != 7 && mainHero.getCharNumber() != 8) {

hero\_path += std::to\_string(mainHero.getCharNumber());

hero\_path += ".png";

} else {

Directions drcts;

hero\_path += std::to\_string(mainHero.getCharNumber());

hero\_path += "/";

switch (direction) {

case 0:

hero\_path += drcts.right;

break;

case 1:

hero\_path += drcts.left;

break;

case 2:

hero\_path += drcts.up;

break;

case 3:

hero\_path += drcts.down;

break;

default:

break;

}

hero\_path += ".png";

}

return hero\_path;

}

void printHero(int i, int j, int direction = 0, int difference = 0, int dimension = 0) {

if (mainHero.getOldX() != mainHero.getX() || mainHero.getOldY() != mainHero.getY()) {

hero\_Img.loadFromFile(getHeroImgPath(direction));

hero\_texture.loadFromImage(hero\_Img);

hero.setTexture(hero\_texture);

if (dimension == 1)

hero.setPosition((float)(i \* mltp - difference), (float)j \* mltp);

if (dimension == 2)

hero.setPosition((float)(i \* mltp + difference), (float)j \* mltp);

if (dimension == 3)

hero.setPosition((float)i \* mltp, (float)(j \* mltp - difference));

if (dimension == 4)

hero.setPosition((float)i \* mltp, (float)(j \* mltp + difference));

gameWindow.draw(hero);

if (difference == 10) {

mainHero.setX(i);

mainHero.setY(j);

mainHero.setOldX(i);

mainHero.setOldY(j);

}

} else {

hero\_Img.loadFromFile(getHeroImgPath(direction));

hero\_texture.loadFromImage(hero\_Img);

hero.setTexture(hero\_texture);

hero.setPosition((float)i \* mltp, (float)j \* mltp);

mainHero.setX(i);

mainHero.setY(j);

mainHero.setOldX(i);

mainHero.setOldY(j);

gameWindow.draw(hero);

}

}

std::string chooseDoorDirection(int i, int j) {

if (door1.i == i && door1.j == j) {

if (!door1.isOpen) {

if (maze[i - 1][j] == room)

return "C:/maze/img/doors/left.png";

if (maze[i + 1][j] == room)

return "C:/maze/img/doors/right.png";

if (maze[i][j + 1] == room)

return "C:/maze/img/doors/up.png";

if (maze[i][j - 1] == room)

return "C:/maze/img/doors/down.png";

} else {

if (maze[i - 1][j] == room)

return "C:/maze/img/doors/up.png";

if (maze[i + 1][j] == room)

return "C:/maze/img/doors/down.png";

if (maze[i][j + 1] == room)

return "C:/maze/img/doors/left.png";

if (maze[i][j - 1] == room)

return "C:/maze/img/doors/right.png";

}

} else if (door2.i == i && door2.j == j){

if (!door2.isOpen) {

if (maze[i - 1][j] == room)

return "C:/maze/img/doors/left.png";

if (maze[i + 1][j] == room)

return "C:/maze/img/doors/right.png";

if (maze[i][j + 1] == room)

return "C:/maze/img/doors/up.png";

if (maze[i][j - 1] == room)

return "C:/maze/img/doors/down.png";

} else {

if (maze[i - 1][j] == room)

return "C:/maze/img/doors/up.png";

if (maze[i + 1][j] == room)

return "C:/maze/img/doors/down.png";

if (maze[i][j + 1] == room)

return "C:/maze/img/doors/left.png";

if (maze[i][j - 1] == room)

return "C:/maze/img/doors/right.png";

}

}

}

void printMaze(int\*\* &main\_maze, int height, int width, int direction, bool explosion = false) {

sf::Texture wallTexture;

wallTexture.loadFromFile(wall\_path);

sf::Texture passTexture;

passTexture.loadFromFile("C:/maze/img/utility/pass.png");

sf::Image door\_image;

sf::Texture door\_Texture;

sf::Sprite mazeBlock;

int difference = 0;

int diff\_dimension;

if (mainHero.getOldX() != mainHero.getX() || mainHero.getOldY() != mainHero.getY()) {

if (mainHero.getOldX() < mainHero.getX()) {

difference = mainHero.getX() - mainHero.getOldX();

diff\_dimension = 1;

} else if (mainHero.getOldX() > mainHero.getX()){

difference = mainHero.getOldX() - mainHero.getX();

diff\_dimension = 2;

} else if (mainHero.getOldY() < mainHero.getY()) {

difference = mainHero.getY() - mainHero.getOldY();

diff\_dimension = 3;

} else if (mainHero.getOldY() > mainHero.getY()){

difference = mainHero.getOldY() - mainHero.getY();

diff\_dimension = 4;

}

difference = difference \* 50 - 10;

}

do {

int hero\_i;

int hero\_j;

for (int i = 0; i < height; ++i) {

for (int j = 0; j < width; ++j) {

if (main\_maze[i][j] == silver\_key || main\_maze[i][j] == gold\_key) {

mazeBlock.setTexture(passTexture);

mazeBlock.setPosition((float)i \* mltp, (float)j \* mltp);

gameWindow.draw(mazeBlock);

if (main\_maze[i][j] == silver\_key) {

door\_image.loadFromFile("C:/maze/img/utility/key2.png");

} else {

door\_image.loadFromFile("C:/maze/img/utility/key.png");

}

door\_Texture.loadFromImage(door\_image);

mazeBlock.setTexture(door\_Texture);

}

if (main\_maze[i][j] == room\_door || main\_maze[i][j] == end\_door) {

mazeBlock.setTexture(passTexture);

mazeBlock.setPosition((float)i \* mltp, (float)j \* mltp);

gameWindow.draw(mazeBlock);

if (main\_maze[i][j] == room\_door)

door\_image.loadFromFile(chooseDoorDirection(i,j));

else

door\_image.loadFromFile("C:/maze/img/doors/end\_door.png");

door\_Texture.loadFromImage(door\_image);

mazeBlock.setTexture(door\_Texture);

}

if (main\_maze[i][j] == pass || main\_maze[i][j] == room) {

mazeBlock.setTexture(passTexture);

}

if (main\_maze[i][j] == wall) {

mazeBlock.setTexture(wallTexture);

}

mazeBlock.setPosition((float)i \* mltp, (float)j \* mltp);

gameWindow.draw(mazeBlock);

if (main\_maze[i][j] == hero\_maze) {

hero\_i = i;

hero\_j = j;

mazeBlock.setTexture(passTexture);

mazeBlock.setPosition((float)i \* mltp, (float)j \* mltp);

gameWindow.draw(mazeBlock);

}

}

}

if (mainHero.getOldX() != mainHero.getX() || mainHero.getOldY() != mainHero.getY()) {

printHero(hero\_i, hero\_j, direction,difference,diff\_dimension);

} else {

printHero(hero\_i, hero\_j, direction);

}

sf::RenderTexture tex;

tex.create(W \* mltp, H \* mltp);

{

sf::RectangleShape globalRect(sf::Vector2f{W \* mltp, H \* mltp});

globalRect.setFillColor(sf::Color::Black);

globalRect.setPosition(0, 0);

sf::CircleShape circle(150);

double x = mainHero.getX() \* mltp - mltp \* 2.5; double y = mainHero.getY() \* mltp - mltp \* 2.5;

if (diff\_dimension == 1) {

x = hero\_i \* mltp - difference - mltp \* 2.5;

y = hero\_j \* mltp - mltp \* 2.5;

}

if (diff\_dimension == 2) {

x = hero\_i \* mltp + difference - mltp \* 2.5;

y = hero\_j \* mltp - mltp \* 2.5;

}

if (diff\_dimension == 3) {

x = hero\_i \* mltp - mltp \* 2.5;

y = hero\_j \* mltp - difference - mltp \* 2.5;

}

if (diff\_dimension == 4) {

x = hero\_i \* mltp - mltp \* 2.5;

y = hero\_j \* mltp + difference - mltp \* 2.5;

}

circle.setFillColor(sf::Color::Transparent);

circle.setPosition((float)x,(float)y);

tex.draw(globalRect, sf::BlendNone);

tex.draw(circle, sf::BlendNone);

tex.display();

}

sf::Sprite eye(tex.getTexture());

gameWindow.draw(eye);

if (!explosion) {

printKeysData();

gameWindow.display();

}

difference-= 10;

sf::Time time = sf::milliseconds(23);

sleep(time);

} while (difference > 0);

}

void print2Eyes(int\*\* main\_maze, int height, int width, int direction, float visionX, float visionY) {

sf::Texture wallTexture;

wallTexture.loadFromFile(wall\_path);

sf::Texture passTexture;

passTexture.loadFromFile("C:/maze/img/utility/pass.png");

sf::Sprite mazeBlock;

int hero\_i;

int hero\_j;

for (int i = 0; i < height; ++i) {

for (int j = 0; j < width; ++j) {

if (main\_maze[i][j] == pass || main\_maze[i][j] == room) {

mazeBlock.setTexture(passTexture);

}

if (main\_maze[i][j] == wall) {

mazeBlock.setTexture(wallTexture);

}

mazeBlock.setPosition((float)i \* mltp, (float)j \* mltp);

gameWindow.draw(mazeBlock);

if (main\_maze[i][j] == hero\_maze) {

hero\_i = i;

hero\_j = j;

mazeBlock.setTexture(passTexture);

mazeBlock.setPosition((float)i \* mltp, (float)j \* mltp);

gameWindow.draw(mazeBlock);

}

}

}

printHero(hero\_i, hero\_j, direction);

sf::RenderTexture tex;

tex.create(W \* mltp, H \* mltp);

{

sf::RectangleShape globalRect(sf::Vector2f{W \* mltp, H \* mltp});

globalRect.setFillColor(sf::Color::Black);

globalRect.setPosition(0, 0);

sf::CircleShape circle(150);

sf::CircleShape visionCircle(400);

visionCircle.setFillColor(sf::Color::Transparent);

visionCircle.setPosition(visionX-200\*2,visionY-200\*2);

double x = (double)mainHero.getX() \* mltp - mltp \* 2.5; double y = (double)mainHero.getY() \* mltp - mltp \* 2.5;

circle.setFillColor(sf::Color::Transparent);

circle.setPosition((float)x,(float)y);

tex.draw(globalRect, sf::BlendNone);

tex.draw(circle, sf::BlendNone);

tex.draw(visionCircle, sf::BlendNone);

tex.display();

}

sf::Sprite eye(tex.getTexture());

gameWindow.draw(eye);

gameWindow.display();

}

void Game::print() {

gameWindow.create(sf::VideoMode(W \* mltp, H \* mltp), "NIGHT STALKER !", sf::Style::Titlebar | sf::Style::Close);

gameWindow.setFramerateLimit(60);

gameWindow.setVerticalSyncEnabled(true);

gameWindow.setPosition({20,20});

mainMenu::putIcoInWindow(&gameWindow);

srand(time(nullptr));

maze = new int\*[HEIGHT];

for (int i = 0; i < HEIGHT; ++i) {

maze[i] = new int[WIDTH];

}

mazemake(maze, HEIGHT, WIDTH);

printMaze(maze, HEIGHT, WIDTH,0);

Game::game(maze, WIDTH);

}

void Game::prerun(int num) {

mainHero.setCharNumber(num);

wall\_path = chooseWallImg();

spellCounter = 0;

door1.isOpen = false;

door2.isOpen = false;

silverFound = false;

goldenFound = false;

Game::print();

}

bool Game::isHover(sf::FloatRect sprite, sf::Vector2f mp){

if(sprite.contains(mp)) {

return true;

} else {

return false;

}

}

bool checkWinCondition() {

sf::Vector2f temp2 = {HEIGHT - 1, (float)(WIDTH)/2 - 3};

sf::Vector2f temp3 = {1,3};

sf::Rect<float> rect1 = {temp2, temp3};

sf::Vector2f currPos = {static\_cast<float>(mainHero.getX()),static\_cast<float>(mainHero.getY())};

if (Game::isHover(rect1,currPos)) {

if (goldenFound) {

return true;

}

return false;

} else {

return false;

}

}

void printExitBtn(sf::RenderWindow \*window, int exit = 0, sf::Sprite \*sprite = &exit\_spell) {

sf::Image questionImg;

if (exit == 0) {

questionImg.loadFromFile("C:/maze/img/utility/close.png");

} else {

questionImg.loadFromFile("C:/maze/img/utility/about.png");

}

sf::Texture questionTexture;

questionTexture.loadFromImage(questionImg);

if (exit == 0) {

exit\_spell.setTexture(questionTexture);

exit\_spell.setScale(0.5,0.5);

exit\_spell.setPosition(370, 10);

} else {

sprite->setTexture(questionTexture);

sprite->setScale(0.5,0.5);

sprite->setPosition(343,10);

}

if (exit == 0) {

window->draw(exit\_spell);

} else {

window->draw(\*sprite);

}

}

void printSpellConfirmation(sf::RenderWindow \*window, const std::string& text) {

sf::Text exitAsk;

exitAsk.setString(text);

exitAsk.setFont(sameFont);

exitAsk.setCharacterSize(45);

exitAsk.setPosition(70, 45);

window->draw(exitAsk);

}

void printYes(sf::RenderWindow \*window) {

yesLbl.setString("YES");

yesLbl.setFont(sameFont);

yesLbl.setCharacterSize(45);

yesLbl.setPosition(180, 145);

window->draw(yesLbl);

}

void printExplosion(int newX, int newY) {

sf::Image gif\_img;

sf::Texture gif\_texture;

sf::Sprite gif\_sprite;

sf::Time time = sf::milliseconds(45);

int i = 0; std::string path; std::string mask = ".png";

while (i < 15) {

path = "C:/maze/img/gif/";

path += std::to\_string(i);

path+= mask;

gif\_img.loadFromFile(path);

gif\_texture.loadFromImage(gif\_img);

gif\_sprite.setTexture(gif\_texture);

gif\_sprite.setPosition((float)newX \* mltp, (float)newY\*mltp);

printMaze(maze, HEIGHT, WIDTH, 0,true);

gameWindow.draw(gif\_sprite);

gameWindow.display();

sleep(time);

i++;

}

}

bool checkNextWall(int newX, int newY, int direction, bool forDoor = false) {

if(!forDoor) {

switch (direction) {

case 0:

return (newX + 1 > 0 && newX + 1 < HEIGHT &&

(maze[newX + 1][newY] == pass || maze[newX + 1][newY] == room));

case 1:

return (newX - 1 > 0 && newX - 1 < HEIGHT &&

(maze[newX - 1][newY] == pass || maze[newX - 1][newY] == room));

case 2:

return (newY - 1 > 0 && newY - 1 < WIDTH &&

(maze[newX][newY - 1] == pass || maze[newX][newY - 1] == room));

case 3:

return (newY + 1 > 0 && newY + 1 < HEIGHT &&

(maze[newX][newY + 1] == pass || maze[newX][newY + 1] == room));

default:

return false;

}

} else {

switch (direction) {

case 0:

return (newX + 1 > 0 && newX + 1 < HEIGHT && maze[newX + 1][newY] == room);

case 1:

return (newX - 1 > 0 && newX - 1 < HEIGHT && maze[newX - 1][newY] == room);

case 2:

return (newY - 1 > 0 && newY - 1 < WIDTH && maze[newX][newY - 1] == room);

case 3:

return (newY + 1 > 0 && newY + 1 < HEIGHT && maze[newX][newY + 1] == room);

default:

return false;

}

}

}

void printBeatleSpell(int x, int y) {

sf::Time visionTime = sf::milliseconds(3000);

print2Eyes(maze, HEIGHT, WIDTH,0, (float)x,(float)y);

sleep(visionTime);

spellCounter = 1;

spellStart = false;

}

void doBeatleSpell() {

if (!spellStart) {

spellStart = true;

}

sf::Event event{};

while (spellCounter < 1) {

if (gameWindow.waitEvent(event)) {

if (event.type == sf::Event::MouseButtonPressed) {

if(event.mouseButton.button == sf::Mouse::Left) {

printBeatleSpell(sf::Mouse::getPosition(gameWindow).x,

sf::Mouse::getPosition(gameWindow).y);

}

}

}

}

}

void openDoor(int\*\* &main\_maze, int doorI, int doorJ) {

if (door1.i == doorI && door1.j == doorJ) {

door1.isOpen = true;

} else {

door2.isOpen = true;

}

main\_maze[doorI][doorJ] = pass;

}

sf::Vector2i findNearestDoor(int i, int j) {

int n = 0;

sf::Vector2i result;

do{

n++;

for (int k=0; k<n; k++) {

if (maze[i][j] == room\_door) {

result.x = i;

result.y = j;

return result;

}

i--;

}

for (int k=0; k<n; k++) {

if (maze[i][j] == room\_door) {

result.x = i;

result.y = j;

return result;

}

j++;

}

n++;

for (int k=0; k<n; k++) {

if (maze[i][j] == room\_door) {

result.x = i;

result.y = j;

return result;

}

i++;

}

for (int k=0; k<n; k++) {

if (maze[i][j] == room\_door) {

result.x = i;

result.y = j;

return result;

}

j--;

}

}

while (true);

}

void doTankSpell(int\*\* &main\_maze) {

if (!spellStart) {

spellStart = true;

}

sf::Event event{};

int direction = 0;

while (spellCounter < 1) {

int newX = mainHero.getX();

int newY = mainHero.getY();

do {

if (gameWindow.waitEvent(event)) {

if (event.type == sf::Event::KeyPressed) {

if (event.key.code == sf::Keyboard::Right) {

newX += 1;

direction = 0;

}

if (event.key.code == sf::Keyboard::Left) {

newX -= 1;

direction = 1;

}

if (event.key.code == sf::Keyboard::Up) {

newY -= 1;

direction = 2;

}

if (event.key.code == sf::Keyboard::Down) {

newY += 1;

direction = 3;

}

}

}

if ((newX > -1 && newY > -1) && ((main\_maze[newX][newY] == wall || main\_maze[newX][newY] == room\_door) && checkNextWall(newX, newY, direction))) {

printExplosion(newX, newY);

gameWindow.display();

main\_maze[newX][newY] = pass;

spellCounter++;

spellStart = false;

}

} while (event.key.code != sf::Keyboard::Right && event.key.code != sf::Keyboard::Left && event.key.code != sf::Keyboard::Up

&& event.key.code != sf::Keyboard::Down && (newX > -1 && newY > -1) && ((main\_maze[newX][newY] == wall || main\_maze[newX][newY] == room\_door)

&& checkNextWall(newX,newY, direction)));

}

}

sf::Vector2i findFreePlace(sf::Vector2f mousePos, int\*\* &main\_maze) {

int mouseI = (int)(mousePos.x)/mltp;

int mouseJ = (int)(mousePos.y)/mltp;

sf::Vector2i result(mainHero.getX(),mainHero.getY());

if (maze[mouseI][mouseJ] == pass) {

result.x = mouseI;

result.y = mouseJ;

return result;

} else {

int n = 0;

int i = mouseI;

int j = mouseJ;

do{

n++;

for (int k=0; k<n; k++) {

if (maze[i][j] == pass || maze[i][j] == room) {

result.x = i;

result.y = j;

return result;

}

i--;

}

for (int k=0; k<n; k++) {

if (maze[i][j] == pass || maze[i][j] == room) {

result.x = i;

result.y = j;

return result;

}

j++;

}

n++;

for (int k=0; k<n; k++) {

if (maze[i][j] == pass || maze[i][j] == room) {

result.x = i;

result.y = j;

return result;

}

i++;

}

for (int k=0; k<n; k++) {

if (maze[i][j] == pass || maze[i][j] == room) {

result.x = i;

result.y = j;

return result;

}

j--;

}

}

while (true);

}

}

void porting(float mouseX, float mouseY, int\*\* &main\_maze) {

sf::Vector2f mousePos(mouseX, mouseY);

double x = mainHero.getX() \* mltp;

double y = mainHero.getY() \* mltp;

sf::Vector2f temp2 = {(HEIGHT - 3)\*50, ((float)(WIDTH) / 2 - 6)\*50};

sf::Vector2f temp3 = {150, 300};

sf::Rect<float> rect1 = {temp2, temp3};

sf::Vector2f heroPos((float) x, (float) y);

if (abs(heroPos.x - mousePos.x) < MAX\_PORT\_DISTANCE) {

if (!Game::isHover(rect1, mousePos)) {

if (abs(heroPos.y - mousePos.y) < MAX\_PORT\_DISTANCE) {

sf::Vector2i tpCoords = findFreePlace(mousePos,main\_maze);

main\_maze[mainHero.getX()][mainHero.getY()] = pass;

mainHero.setX(tpCoords.x);

mainHero.setY(tpCoords.y);

mainHero.setOldX(tpCoords.x);

mainHero.setOldY(tpCoords.y);

main\_maze[tpCoords.x][tpCoords.y] = hero\_maze;

if (main\_maze[tpCoords.x-1][tpCoords.y-1] == room || main\_maze[tpCoords.x-1][tpCoords.y] == room

|| main\_maze[tpCoords.x][tpCoords.y-1] == room || main\_maze[tpCoords.x+1][tpCoords.y-1] == room

|| main\_maze[tpCoords.x-1][tpCoords.y+1] == room || main\_maze[tpCoords.x+1][tpCoords.y+1] == room

|| main\_maze[tpCoords.x+1][tpCoords.y] == room || main\_maze[tpCoords.x][tpCoords.y+1] == room) {

sf::Vector2i doorPos = findNearestDoor(tpCoords.x, tpCoords.y);

openDoor(main\_maze,doorPos.x, doorPos.y);

}

spellCounter = 1;

spellStart = false;

}

}

}

}

void doRestSpell(int\*\* &main\_maze) {

if (!spellStart) {

spellStart = true;

}

sf::Event event{};

while (spellCounter < 1) {

if (gameWindow.waitEvent(event)) {

if (event.type == sf::Event::MouseButtonPressed) {

if(event.mouseButton.button == sf::Mouse::Left) {

porting((float)sf::Mouse::getPosition(gameWindow).x,

(float)sf::Mouse::getPosition(gameWindow).y,

main\_maze);

}

}

}

}

}

void doSpecialSpell(int\*\* &main\_maze) {

int hero\_num = mainHero.getCharNumber();

switch(hero\_num) {

case 7:

doTankSpell(main\_maze);

break;

case 8:

doBeatleSpell();

break;

default:

doRestSpell(main\_maze);

break;

}

}

void printSpellConfirm(const std::string& text) {

sf::RenderWindow spellWindow;

spellWindow.create(sf::VideoMode(400,210), "Quick notification", sf::Style::None);

mainMenu::printExitBgSprite(&spellWindow);

printSpellConfirmation(&spellWindow, text);

printExitBtn(&spellWindow);

sf::Sprite about;

printExitBtn(&spellWindow, 1, &about);

printYes(&spellWindow);

spellWindow.display();

while (spellWindow.isOpen()) {

sf::Event event{};

while ( spellWindow.pollEvent(event) ) {

if (event.type == sf::Event::MouseButtonPressed) {

if ( event.mouseButton.button == sf::Mouse::Left ) {

sf::Vector2f mp;

mp.x = (float)sf::Mouse::getPosition(spellWindow).x;

mp.y = (float)sf::Mouse::getPosition(spellWindow).y;

if ( Game::isHover(exit\_spell.getGlobalBounds(), mp )) {

spellWindow.close();

}

if (Game::isHover(about.getGlobalBounds(), mp)) {

mainMenu::info();

}

if (Game::isHover(yesLbl.getGlobalBounds(), mp)){

spellWindow.close();

if (spellCounter == 0) {

if (!spellStart) {

spellStart = true;

}

}

}

}

}

}

}

}

bool checkDoorState(int x, int y, door temp\_door) {

return maze[x][y] == room\_door && x == temp\_door.i && y == temp\_door.j && silverFound;

}

void Game::game(int\*\* &main\_maze, int width) {

int x = 0;

int y = width/2 - 1;

int direction = 0;

sameFont.loadFromFile("C:/maze/font/font.ttf");

while(gameWindow.isOpen()) {

sf::Event event{};

int newX = x;

int newY = y;

while (gameWindow.pollEvent(event)){

sf::Vector2f mp;

if(event.type == sf::Event::Closed) {

gameWindow.close();

mainMenu::start();

}

if(event.type == sf::Event::KeyPressed) {

newX = mainHero.getX();

newY = mainHero.getY();

if(event.key.code == sf::Keyboard::Right) {

newX+= 1;

direction = 0;

}

if (event.key.code == sf::Keyboard::Left) {

newX-= 1;

direction = 1;

}

if(event.key.code == sf::Keyboard::Up) {

newY-= 1;

direction = 2;

}

if (event.key.code == sf::Keyboard::Down) {

newY+= 1;

direction = 3;

}

if (event.key.code == sf::Keyboard::R) {

if (spellCounter == 0) {

printSpellConfirm("Click Yes to use\n your char spell");

} else {

printSpellConfirm("Your spell is \n already done");

}

}

}

}

if ((newX > -1 && newY > -1) && (newX < HEIGHT && newY < WIDTH) && main\_maze[newX][newY] == end\_door && !isWin) {

main\_maze[x][y] = pass;

main\_maze[newX][newY] = hero\_maze;

mainHero.setOldX(x);

mainHero.setOldY(y);

x = newX;

y = newY;

mainHero.setX(x);

mainHero.setY(y);

isWin = true;

if (checkWinCondition()) {

Game::win();

}

} else if ((newX > -1 && newY > -1) && (newX < HEIGHT && newY < WIDTH) && (main\_maze[newX][newY] == pass || main\_maze[newX][newY] == room ||

checkDoorState(newX,newY,door1) || checkDoorState(newX,newY,door2))) {

main\_maze[x][y] = pass;

main\_maze[newX][newY] = hero\_maze;

mainHero.setOldX(x);

mainHero.setOldY(y);

x = newX;

y = newY;

}

mainHero.setX(x);

mainHero.setY(y);

if (spellStart) {

doSpecialSpell(main\_maze);

x = mainHero.getX();

y = mainHero.getY();

}

if (!(silverFound && goldenFound))

pickUpKey();

printMaze(main\_maze, HEIGHT, WIDTH,direction);

}

}

void printWinBg(sf::RenderWindow \*window) {

sf::Image bg;

bg.loadFromFile("C:/maze/img/utility/win.png");

sf::Texture bgTexture;

bgTexture.loadFromImage(bg);

sf::Sprite bgSprite(bgTexture);

window->draw(bgSprite);

}

void showOkBtn(sf::RenderWindow \*window) {

okText.setString("OK");

okText.setFillColor(sf::Color::White);

okText.setFont(sameFont);

okText.setCharacterSize(85);

okText.setPosition(320, 180);

window->draw(okText);

}

void Game::win() {

sf::RenderWindow winWindow;

winWindow.create(sf::VideoMode(700,300), "You WIN!!!", sf::Style::None);

printWinBg(&winWindow);

showOkBtn(&winWindow);

winWindow.display();

sf::Time time = sf::milliseconds(1000);

while (winWindow.isOpen()) {

sf::Event event{};

while (winWindow.pollEvent(event)) {

if (event.type == sf::Event::MouseButtonPressed) {

if (event.mouseButton.button == sf::Mouse::Left) {

winWindow.close();

gameWindow.close();

mainMenu::start();

}

}

}

}

}

//---------------------------------------------------------------------------

#include "Hero.h"

//---------------------------------------------------------------------------

Hero::Hero(int x, int y, int oldX, int oldY, int number)

{

this->x = x;

this->y = y;

this->oldX = oldX;

this->oldY = oldY;

this->char\_number = number;

}

int Hero::getX()

{

return this->x;

}

int Hero::getY()

{

return this->y;

}

int Hero::getCharNumber()

{

return this->char\_number;

}

void Hero::setX(int x)

{

this->x = x;

}

void Hero::setY(int y)

{

this->y = y;

}

void Hero::setCharNumber(int number)

{

this->char\_number = number;

}

int Hero::getOldX()

{

return this->oldX;

}

int Hero::getOldY()

{

return this->oldY;

}

void Hero::setOldX(int x)

{

this->oldX = x;

}

void Hero::setOldY(int y)

{

this->oldY = y;

}

#include "door.h"

door::door(int i, int j) {

this->i = i;

this->j = j;

}

#include "choose.h"

#include <SFML/Graphics.hpp>

#include "../main/mainMenu.h"

#include <windows.h>

#include "../game/Game.h"

sf::RenderWindow charsWindow;

const int W = 27;

const int H = 11;

const int mltp = 30;

const int BASE\_OFFSET = 25;

const int IMG\_OFFSET = 175;

const int WIDTH = 30;

const int IMG\_MLTPL = 3;

const int HEIGHT = 50;

std::string Dir = "C:/maze/chars/";

std::string mask = "png";

sf::Sprite exitMark;

sf::Sprite questMark;

bool isHover2(sf::FloatRect sprite, sf::Vector2f mp){

if(sprite.contains(mp)) {

return true;

} else {

return false;

}

}

void printQuestBtn(sf::RenderWindow \*window) {

sf::Image questionImg;

questionImg.loadFromFile("C:/maze/img/utility/about.png");

sf::Texture questionTexture;

questionTexture.loadFromImage(questionImg);

questMark.setTexture(questionTexture);

questMark.setScale(0.5,0.5);

questMark.setPosition(755,10);

window->draw(questMark);

}

void printBg () {

sf::Texture bg;

bg.loadFromFile("C:/maze/img/utility/phon3.png");

sf::Sprite background(bg);

background.setPosition(0,0);

charsWindow.draw(background);

}

int getCharNum() {

std::vector<std::string> s;

std::string path;

path+= Dir + "\*." + mask;

WIN32\_FIND\_DATA ffd;

HANDLE hl = FindFirstFileA(path.c\_str(),&ffd);

int counter = 0;

if (hl != INVALID\_HANDLE\_VALUE) {

do {

counter++;

} while (FindNextFileA(hl,&ffd) != 0);

} else {

MessageBoxA(nullptr, "FILES NOT FOUND !!!", "About", MB\_OK);

}

return counter+2;

}

void choose::printExitBtn(sf::RenderWindow \*window) {

sf::Image questionImg;

questionImg.loadFromFile("C:/maze/img/utility/close.png");

sf::Texture questionTexture;

questionTexture.loadFromImage(questionImg);

exitMark.setTexture(questionTexture);

exitMark.setScale(0.5,0.5);

exitMark.setPosition(780, 10);

window->draw(exitMark);

}

void choose::print() {

charsWindow.create(sf::VideoMode(W \* mltp, H \* mltp), "CHOOSE YOUR CHARACTER !" ,sf::Style::None);

sf::Image img;

img.loadFromFile("C:/maze/img/newico.png");

charsWindow.setIcon(img.getSize().x, img.getSize().y, img.getPixelsPtr());

printBg();

choose::printExitBtn(&charsWindow);

int char\_num = getCharNum();

sf::Sprite \*stArr[char\_num];

for (int i = 0; i < char\_num; ++i) {

std::string path;

path+=Dir + "char";

int num = i + 1;

if (num != 7 && num != 8) {

path+=std::to\_string(num);

} else {

path+=std::to\_string(num) + "/up";

}

path+="." + mask;

img.loadFromFile(path);

sf::Texture hero;

hero.loadFromImage(img);

stArr[i] = new sf::Sprite;

if (i < 4) {

stArr[i]->setPosition((float)BASE\_OFFSET + (float)i\*(WIDTH+IMG\_OFFSET),BASE\_OFFSET);

} else {

stArr[i]->setPosition((float)BASE\_OFFSET + (float)(i-4)\*(WIDTH+IMG\_OFFSET),BASE\_OFFSET + HEIGHT\*IMG\_MLTPL);

}

stArr[i]->setTexture(hero);

if (num == 7 || num == 8) {

stArr[i]->setScale(0.6, 0.6);

}

charsWindow.draw(\*stArr[i]);

}

printQuestBtn(&charsWindow);

charsWindow.display();

while(charsWindow.isOpen()) {

sf::Event event{};

while (charsWindow.pollEvent(event)){

if(event.type == sf::Event::MouseButtonPressed) {

if (event.mouseButton.button == sf::Mouse::Left) {

sf::Vector2f mp;

mp.x = (float)sf::Mouse::getPosition(charsWindow).x;

mp.y = (float)sf::Mouse::getPosition(charsWindow).y;

if (Game::isHover(exitMark.getGlobalBounds(), mp)) {

charsWindow.close();

}

if(Game::isHover(questMark.getGlobalBounds(), mp)) {

mainMenu::info();

}

for (int i = 0; i < char\_num; ++i) {

if(isHover2(stArr[i]->getGlobalBounds(), mp)) {

charsWindow.close();

Game::prerun(i+1);

}

}

}

}

}

}

}