**Final Group Project - 17 B**

**The Dungeon Manuel**

**Gary (The Elric) Schrienr**

**Miguel (The Thinker) Duarte**

**Adam (The Crafty) Blank**

**Jacari (The Sleeping) Brandon**

**Table Of Content:**

Summary.... pg 2 Flow chart....pg 49

Teams Jobs.... pg 2-3 Code Requirements....pg

Pseudo Code.... pg 3-4

Displayed Code.... pg 4-48

**Summary**

An RPG Rouge like game created by few members of the original group of 7 turned out to be more or less a changed for the team. The rouge like mini game adventure sets you in a dungeon where the user must navigate through the maze avoiding various enemy units and traps. The arrow directional key moves the user's hero sprite up, down, left, and right. Upon confronting an enemy a pop up window displays the dungeons combat system where you charge and attack in real time. Winning a combat battle rewards the user with experience points which is saved into the game. The final step to the game after navigating through the dungeon is defeating a final boss to lead you out of the dungeon and on forward to the next level. Next level mechanic has not been implemented.

**Assigned Jobs**

Project Leader Gary wrote most of the back end of the code. He structured the way the code and methods were to be utilized. Furthermore, he created the file save and load feature. He wrote the database SQL hookup; however, our version Qt 4.8 did not allow it run successfully.

Adam took charge in writing the time driven turned based combat system.

Miguel had the vision for the game and was the one to present his idea for this group project. He was in charge of editing and designing most of the front end elements in the game. He created the help screen and edited and implemented most of the resource files within the game. His prime directive was to make the game look good and populate the map with the appropriate variables. A lot of Qpixmap manipulation. Jacari handled minor parts of the game primarily assisting Miguel with the gathering character images and resource files. Jacari also helped Miguel with the general design vision of the game when providing resource files and doing the documentation.

**Concepts:**

**CLASSES**: We utilize multiple classes, including a class for each window (mainwindow.cpp, help\_screen.cpp, combat\_screen.cpp, help\_screen.cpp, lose\_screen.cpp, items\_screen.cpp, win\_screen.cpp, magic.cpp) and it's operations, a class for saved games (savedgame.cpp), and a class for loading the config file (loadfiles.cpp).

**POLYMORPHISM**: All of the windows extend QDialog or QMainWindow classes, which demonstrates polymorphism.

**DATABASE**: In savedgame.cpp (SavedGame class) we connect to an external database, retrieve the saved game data from the table and save it to a QStringList. When a game is saved, I INSERT the game into the table and update the QStringList.

**FILE OPERATIONS**: In savedgame.cpp, we open sqlconfig.cfg and load the connection details for the SQL server.

**OPEN EXTERNAL** APPLICATION: in mainwindow.cpp, when a user clicks the "SQL CONFIG" button, the game opens notepad2.exe in the CONFIG folder, with the sqlconfig.cfg as a parameter, giving the user the ability to modify the file.  
**GUI DESIGN**: We utilize QT gui design utilities to design a GUI that includes a menu, a gameboard, and several child windows to conduct operations within the game.

**Pseudo Code:**

The Game Opens up into main giving the user three options New Game, Load, and Help the user inputs whichever option best suits them. Selecting Help will pop up a second window which explains the point of the game and rules, while also informing the user of the controls needed to operate the game. Selecting load will upload the previous saved game of any user prior to its use saving the data of all monsters and previous character placement in the map. Selecting new game opens the game “Dungeons” a window will display and the program will run off of mainwindow.cpp, placing the user in a fixed location on the map which is 98. Using the array 45 by 45 “Dungeon ” is a maze which 1, 4, 6, 8, and 9s represent objects the user can not walk through such as walls and 0s represent the area of free movement allowed to the user. Placing enemy throughout the map in the number of 91 and the final boss 90.

The user then has the ability to move through the map using the directional keys using the keyPressEvent and checkmove algorithm also ran in the mainwindow.cpp. Once the user encounters and Enemy a separate window appears. Run from the Combat\_screen files the user has the ability to have a battle with the time driven turned based combat system. Using the ability provided to the hero's class the user will have a limited amount of time to defeat the enemy. Picking attack, magic, or defend the hero character will run through a random damage generator of that specific action to affect the enemy characters. While attack charging your attacks and defend the Enemy A.I will attack the user periodically with a random generated damage. After defeating the Enemy the user is rewarded with experience and the combat screen closes to let the user continue on the main map.

Once the user arrives to the final boss the combat screen opens and the hero character begins his final battle to win the game. Unlike other enemy classes the final boss class is far stronger then the minions. Once defeating the final boss the Win\_screen files will run popping up a window stating to the user congratulations you've won the game (exit stage right). If being defeated by the boss or any other minion occurs the Lose\_screen file will run displaying the “You've been Defeated” screen. Throughout anytime in game you can click on the save game button which will record and save all data that is currently available on the map design recording the placement of the hero and any defeated enemies on the map.

***Header Files:***

Boss.h :

#ifndef BOSS\_H

#define BOSS\_H

#include <QString>

class Boss{

public:

void setBossHealth(){bossHealth=100;}

void setMassiveAttack(){massiveDamage=20;};

void setMediumAttack(){mediumDamage=10;};

void setSmallAttack(){smallDamage=5;};

int getBossHealth(){return bossHealth;}

int getMassiveAttack(){return massiveDamage;};

int getMediumAttack(){return mediumDamage;};

int getSmallAttack(){return smallDamage;};

void setBossName(QString n){bossName.append(n);};

QString getBossName(){return bossName;};

private:

int massiveDamage;

int mediumDamage;

int smallDamage;

int bossHealth;

QString bossName;

};

#endif // BOSS\_H

***Combat\_Screen.h :***

#ifndef COMBAT\_SCREEN\_H

#define COMBAT\_SCREEN\_H

#include <QtGui>

#include "hero.h"

#include <QProgressBar>

#include <QHBoxLayout>

#include <QDialog>

using namespace std;

namespace Ui {

class Combat\_Screen;

}

class Combat\_Screen : public QDialog,public Hero

{

Q\_OBJECT

public:

unsigned short atkValue;

unsigned short dfnValue;

unsigned short mgcValue;

void setStandardEnemy(bool n){standardEnemy = n;};

void setBossEnemy(bool n){bossEnemy = n;};

bool getStandardEnemy(){return standardEnemy;};

bool getBossEnemy(){return bossEnemy;};

QProgressBar \*hero\_health\_bar;

QLabel \*enemy\_health\_display;

explicit Combat\_Screen(QWidget \*parent = 0);

~*Combat\_Screen*();

private:

Ui::Combat\_Screen \*ui;

bool standardEnemy;

bool bossEnemy;

QPushButton \*attack\_button;

QPushButton \*defend\_button;

QPushButton \*magic\_button;

QPushButton \*item\_button;

QLabel \*hero\_action\_label;

QLabel \*enemy\_action\_label;

QProgressBar \*attack\_bar;

QProgressBar \*defend\_bar;

QProgressBar \*magic\_bar;

QTimer \*attack\_timer;

QTimer \*standardAttackTimer;

QTimer \*magic\_timer;

QTimer \*defend\_timer;

QTimer \*enemy\_timer;

QTimer \*enemy\_display\_timer;

QTimer \*hero\_display\_timer;

QTimer \*attack\_bar\_timer;

QTimer \*defend\_bar\_timer;

QTimer \*magic\_bar\_timer;

QTimer \*bossTimer;

void win();

void lose();

private slots:

void calculate\_attack();

void enable\_attack();

void enable\_magic();

void enable\_defend();

void calculate\_enemy\_attack();

void calculate\_standard\_attack();

void set\_enemy\_action\_display();

void set\_hero\_action\_display();

void execute\_defend();

void execute\_magic();

void execute\_item();

void update\_attack\_bar();

void update\_defend\_bar();

void update\_magic\_bar();

void bossAttacks();

};

#endif // COMBAT\_SCREEN\_H

***Declarations.h :***

#ifndef DECLARATIONS\_H

#define DECLARATIONS\_H

#define MAP\_EMPTY 0

#define MAP\_WALL 1

#define MOVE\_LEFT 50

#define MOVE\_RIGHT 51

#define MOVE\_UP 52

#define MOVE\_DOWN 53

#define BOARD\_CANT\_MOVE 54

#define BOARD\_CAN\_MOVE 55

#define CHAR\_ISENEMY 56

#define CHAR\_ISFRIENDLY 57

#define MOVE\_AUTHORIZED 58

#define MOVE\_ERROR\_OUTOFBOUNDS 59

#define MOVE\_ERROR\_BLOCKED 60

#define MOVE\_ERROR\_UNKNOWN 61

#define CHARACTER\_BOSS 90

#define CHARACTER\_ENEMY 91

#define CHARACTER\_ITEM 92

#define CHARACTER\_SPAWN 98

#define CHARACTER\_EXIT 99

#endif // DECLARATIONS\_H

***Enemy.h :***

#ifndef ENEMY\_H

#define ENEMY\_H

#include <QtGui>

#include "Boss.h"

#endif // ENEMY\_H

class Enemy : public Boss{

public:

void set\_enemy\_health(int n){enemy\_health=n;}

void set\_enemy\_attack(int n){enemy\_attack=n;};

int get\_enemy\_health(){return enemy\_health;}

int get\_enemy\_attack(){return enemy\_attack;};

void set\_enemy\_name(QString n){enemy\_name.append(n);};

QString get\_enemy\_name(){return enemy\_name;};

private:

int enemy\_health;

QString enemy\_name;

int enemy\_attack;

};

***Help\_Screen.h :***

#ifndef HELP\_SCREEN\_H

#define HELP\_SCREEN\_H

#include <QMainWindow>

namespace Ui {

class Help\_Screen;

}

class Help\_Screen : public QMainWindow

{

Q\_OBJECT

public:

explicit Help\_Screen(QWidget \*parent = 0);

~*Help\_Screen*();

private slots:

void on\_ReturnToMain\_clicked();

void on\_ReturnToMain\_pressed();

private:

Ui::Help\_Screen \*ui;

};

#endif // HELP\_SCREEN\_H

***Hero.h :***

#ifndef HERO\_H

#define HERO\_H

#endif // HERO\_H

#include <QLabel>

#include "enemy.h"

class Hero : public Enemy{

public:

QLabel \*hero\_pic;

void set\_attacks(int n,int m){attack=5;

ice=n;

fire=m;};

void set\_lives(int n){lives=n;};

int get\_lives(){return lives;};

int get\_hero\_lives(){return lives;};

void set\_hero\_health(int n){hero\_health=n;};

void calculate\_hero\_damage(int damage){hero\_health-=damage;};

int get\_hero\_health(){return hero\_health;};

int get\_hero\_attack(){return attack;};

int get\_hero\_fire(){return fire;};

int get\_hero\_ice(){return ice;};

bool get\_defend(){return defend;};

void set\_defend(bool n){defend=n;};

void set\_enemy\_health(int n){enemy\_health=n;};

int get\_enemy\_health(){return enemy\_health;};

void calculate\_enemy\_damage(int damage){enemy\_health-=damage;};

private:

int attack;

int fire;

int ice;

bool defend;

int hero\_health;

int lives;

int enemy\_health;

};

***Screen\_item.h :***

#ifndef ITEM\_SCREEN\_H

#define ITEM\_SCREEN\_H

#include <QDialog>

namespace Ui {

class Item\_screen;

}

class Item\_screen : public QDialog

{

Q\_OBJECT

public:

explicit Item\_screen(QWidget \*parent = 0);

~*Item\_screen*();

private slots:

void on\_pushButton\_clicked();

void on\_pushButton\_2\_clicked();

void on\_pushButton\_3\_clicked();

void on\_pushButton\_4\_clicked();

private:

Ui::Item\_screen \*ui;

};

#endif // ITEM\_SCREEN\_H

***Loadfiles.h :***

#ifndef LOADFILES\_H

#define LOADFILES\_H

#include <QDialog>

#include <QStringList>

#include <QListWidgetItem>

namespace Ui {

class LoadFiles;

}

class LoadFiles : public QDialog

{

Q\_OBJECT

public:

explicit LoadFiles(int savedGameCount, QWidget \*parent = 0);

~*LoadFiles*();

private slots:

void on\_listWidget\_itemDoubleClicked(QListWidgetItem \*item);

private:

Ui::LoadFiles \*ui;

};

#endif // LOADFILES\_H

***Lose\_screen.h :***

#ifndef LOSE\_SCREEN\_H

#define LOSE\_SCREEN\_H

#include <QDialog>

namespace Ui {

class lose\_screen;

}

class lose\_screen : public QDialog

{

Q\_OBJECT

public:

explicit lose\_screen(QWidget \*parent = 0);

~*lose\_screen*();

private slots:

void on\_close\_button\_clicked();

private:

Ui::lose\_screen \*ui;

};

#endif // LOSE\_SCREEN\_H

Magic.h :

#ifndef MAGIC\_H

#define MAGIC\_H

#include <QDialog>

namespace Ui {

class Magic;

}

class Magic : public QDialog

{

Q\_OBJECT

public:

explicit Magic(QWidget \*parent = 0);

~*Magic*();

private:

Ui::Magic \*ui;

QPushButton \*fire\_button;

QPushButton \*ice\_button;

QPushButton \*close\_button;

QTimer \*magic\_timer;

private slots:

void execute\_fire();

void execute\_ice();

};

#endif // MAGIC\_H

***Mainwindow.h :***

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <iostream>

#include <sstream>

#include <QMainWindow>

#include <QGridLayout>

#include <QLabel>

#include <QString>

#include <QPushButton>

//#include "declarations.h"

#include <vector>

#include "combat\_screen.h"

#include "help\_screen.h"

#include "savedgame.h"

namespace Ui {

class MainWindow;

}

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit MainWindow(QWidget \*parent = 0);

~*MainWindow*();

int boardBoundX;

int boardBoundY;

static const int boardMatrix[45][45];

int checkMove(int direction);

int getX() {return this->\_curX;}

int getY() {return this->\_curY;}

void centerWidget(bool useSizeHint);

private:

Ui::MainWindow \*ui;

void *keyPressEvent*(QKeyEvent \*event);

int currentOption;

int numOptions;

bool *eventFilter*(QObject \*obj, QEvent \*event);

void updateMainMenu(int &oldOption, int &newOption, bool DirectionUp);

QPushButton mainButtons[4];

QWidget \*gameBoardWidget;

int \*GetSavedGames();

void SetupGameBoard();

void LoadGame(int savedGamNumber);

void *closeEvent*(QCloseEvent \*event);

void updateHealth(int health);

QLabel \*healthStatus;

QWidget \*menuWidget;

SavedGame savedGames;

bool inGame;

int \_curX;

int \_curY;

int \_curHealth;

QGridLayout \*gridLayout;

QImage \_path;

QImage \_wall;

QImage \_character;

QImage \_coffin; //Cosmetic wall To be represented by #2 in array

QImage \_topLeft;

QImage \_topMiddle;

QImage \_topRight;

QImage \_botLeft;

QImage \_botRight;

QImage \_midLeft;

QImage \_midRight;

QImage \_enemyOne;

QImage \_enemyTwo;

QImage \_enemyBoss;

Combat\_Screen \*mainChar;

Help\_Screen \*helpScreenMain;

private slots:

int on\_pushButton\_clicked();

int on\_pushButton\_2\_clicked();

int on\_pushButton\_3\_clicked();

int on\_pushButton\_4\_clicked();

void moveCharacter(int moveType);

int on\_back\_clicked();

int on\_save\_clicked();

};

#endif // MAINWINDOW\_H

***SavedGame.h :***

#ifndef SAVEDGAME\_H

#define SAVEDGAME\_H

#include <QString>

#include <QFile>

#include <QStringList>

class SavedGame

{

private:

QFile \*configFile;

QStringList savedGames;

bool SaveGamesToFile();

bool IsLoaded;

public:

SavedGame();

bool Load(QString const& configPath, QString const& configFileName);

QStringList GetSavedGameList();

int GetSavedGameCount();

QString GetSavedGame(int gameNumber);

bool GetIsLoaded();

void AddGame(int x, int y, int experience);

bool SaveGame();

bool DeleteGame();

int\* getGame(int gameNumber);

};

Win\_screen.h :

#ifndef WIN\_SCREEN\_H

#define WIN\_SCREEN\_H

#include <QDialog>

#include <QPushButton>

#include "combat\_screen.h"

namespace Ui {

class Win\_Screen;

}

class Win\_Screen : public QDialog

{

Q\_OBJECT

public:

explicit Win\_Screen(QWidget \*parent = 0);

~*Win\_Screen*();

private:

Ui::Win\_Screen \*ui;

QPushButton \*close\_button;

private slots:

void finish();

};

#endif // WIN\_SCREEN\_H

**Source Files:**

***Combat\_screen :***

#include <QtGui>

#include <iostream>

#include "combat\_screen.h"

#include "win\_screen.h"

#include "lose\_screen.h"

#include "magic.h"

#include "item\_screen.h"

#include "ui\_combat\_screen.h"

#include "winscreenendgame.h"

#include <QDebug>

Combat\_Screen::Combat\_Screen(QWidget \*parent) :

QDialog(parent),

ui(new Ui::Combat\_Screen)

{

ui->setupUi(this);

//Setting Background image for TitleScreen Ui - Miguel

QPixmap bg(":/BGCombat.png");

bg = bg.scaled(this->size(), Qt::IgnoreAspectRatio);

QPalette palette;

palette.setBrush(QPalette::Background, bg);

this->setPalette(palette);

//set\_lives(1);

//set\_hero\_health(100);

//set\_attacks(8,20);

//set\_defend(false);

atkValue=0;

dfnValue=0;

mgcValue=0;

QImage hero;

hero = QImage(":/Combathero.png");

hero\_pic = new QLabel;

hero\_pic->setPixmap(QPixmap::fromImage(hero));

QLabel \*enemy\_title = new QLabel;

QLabel \*hp\_label = new QLabel;

enemy\_title->setText(get\_enemy\_name());

hp\_label->setText(" HP");

hp\_label->setStyleSheet("font: 15pt; color: red;");

enemy\_display\_timer = new QTimer;

hero\_display\_timer = new QTimer;

defend\_timer = new QTimer;

magic\_timer = new QTimer;

attack\_timer = new QTimer;

enemy\_timer = new QTimer;

standardAttackTimer = new QTimer;

bossTimer = new QTimer;

hero\_action\_label = new QLabel;

hero\_action\_label->setText(" ");

enemy\_action\_label = new QLabel;

enemy\_action\_label->setText(" ");

enemy\_health\_display = new QLabel;

enemy\_health\_display->setNum(get\_enemy\_health());

enemy\_health\_display->setStyleSheet("font: 15pt; color: red;");

attack\_button = new QPushButton;

attack\_button->setText("ATTACK");

attack\_button->setDefault(true);

defend\_button = new QPushButton;

defend\_button->setText("DEFEND");

magic\_button = new QPushButton;

magic\_button->setText("MAGIC");

item\_button = new QPushButton;

item\_button->setText("ITEM");

hero\_health\_bar = new QProgressBar;

hero\_health\_bar->setValue(get\_hero\_health());

QString mainStyleSheet = QString("QProgressBar::chunk { background: green;}");

mainStyleSheet.append("QProgressBar{ color: black;}");

mainStyleSheet.append("QProgressBar {border: 2px solid grey; text-align: center; background-color: grey;}");

hero\_health\_bar->setStyleSheet(mainStyleSheet);

attack\_bar = new QProgressBar;

QString myStyleSheet1 = QString("QProgressBar::chunk { background: purple;}");

myStyleSheet1.append("QProgressBar{ color: black;}");

myStyleSheet1.append("QProgressBar {border: 1px solid grey; text-align: center; background-color: grey;}");

attack\_bar->setStyleSheet(myStyleSheet1);

defend\_bar = new QProgressBar;

QString myStyleSheet2 = QString("QProgressBar::chunk { background: yellow;}");

myStyleSheet2.append("QProgressBar{ color: black;}");

myStyleSheet2.append("QProgressBar {border: 1px solid grey; text-align: center; background-color: grey;}");

defend\_bar->setStyleSheet(myStyleSheet2);

magic\_bar = new QProgressBar;

QString myStyleSheet3 = QString("QProgressBar::chunk { background: blue;}");

myStyleSheet3.append("QProgressBar{ color: black;}");

myStyleSheet3.append("QProgressBar {border: 1px solid grey; text-align: center; background-color: grey;}");

magic\_bar->setStyleSheet(myStyleSheet3);

attack\_bar->setValue(atkValue);

defend\_bar->setValue(dfnValue);

magic\_bar->setValue(mgcValue);

QHBoxLayout \*attack\_button\_layout = new QHBoxLayout;

attack\_button\_layout->addWidget(attack\_button);

attack\_button\_layout->addWidget(attack\_bar);

QHBoxLayout \*defend\_button\_layout = new QHBoxLayout;

defend\_button\_layout->addWidget(defend\_button);

defend\_button\_layout->addWidget(defend\_bar);

QHBoxLayout \*magic\_button\_layout = new QHBoxLayout;

magic\_button\_layout->addWidget(magic\_button);

magic\_button\_layout->addWidget(magic\_bar);

QHBoxLayout \*item\_button\_layout = new QHBoxLayout;

item\_button\_layout->addWidget(item\_button);

QVBoxLayout \*left\_layout = new QVBoxLayout;

left\_layout->addWidget(hero\_health\_bar);

left\_layout->addWidget(hero\_pic);

left\_layout->addLayout(attack\_button\_layout);

left\_layout->addLayout(defend\_button\_layout);

left\_layout->addLayout(magic\_button\_layout);

left\_layout->addLayout(item\_button\_layout);

QHBoxLayout \*top\_right\_layout = new QHBoxLayout;

top\_right\_layout->addWidget(enemy\_title);

top\_right\_layout->addWidget(hp\_label);

top\_right\_layout->addWidget(enemy\_health\_display);

QVBoxLayout \*right\_layout = new QVBoxLayout;

right\_layout->addLayout(top\_right\_layout);

right\_layout->addStretch();

QHBoxLayout \*top\_layout = new QHBoxLayout;

top\_layout->addLayout(left\_layout);

top\_layout->addSpacing(100);

top\_layout->addLayout(right\_layout);

QVBoxLayout \*bottom\_layout = new QVBoxLayout;

bottom\_layout->addWidget(hero\_action\_label);

bottom\_layout->addWidget(enemy\_action\_label);

QVBoxLayout \*main\_layout = new QVBoxLayout;

main\_layout->addLayout(top\_layout);

main\_layout->addLayout(bottom\_layout);

setLayout(main\_layout);

QObject::connect(attack\_button,SIGNAL(clicked()),

this,SLOT(calculate\_attack()));

QObject::connect(enemy\_timer,SIGNAL(timeout()),

this,SLOT(calculate\_enemy\_attack()));

QObject::connect(standardAttackTimer,SIGNAL(timeout()),

this,SLOT(calculate\_standard\_attack()));

QObject::connect(defend\_button,SIGNAL(clicked()),

this,SLOT(execute\_defend()));

QObject::connect(magic\_button,SIGNAL(clicked()),

this,SLOT(execute\_magic()));

QObject::connect(item\_button,SIGNAL(clicked()),

this,SLOT(execute\_item()));

QObject::connect(bossTimer,SIGNAL(timeout()),

this,SLOT(bossAttacks()));

hero\_action\_label->show();

enemy\_action\_label->show();

}

Combat\_Screen::~*Combat\_Screen*()

{

delete ui;

}

void Combat\_Screen::calculate\_attack(){

calculate\_enemy\_damage(get\_hero\_attack());

enemy\_health\_display->setNum(get\_enemy\_health());

if(get\_enemy\_health()<=0){

win();

Win\_Screen dialog(this);

if(dialog.*exec*()==1){

*accept*();

}

}

attack\_bar\_timer = new QTimer;

QObject::connect(attack\_bar\_timer,SIGNAL(timeout()),

this,SLOT(update\_attack\_bar()));

QObject::connect(hero\_display\_timer,SIGNAL(timeout()),

this,SLOT(set\_hero\_action\_display()));

QObject::connect(attack\_timer,SIGNAL(timeout()),

this,SLOT(enable\_attack()));

attack\_button->setEnabled(false);

attack\_timer->start(4000);

attack\_bar\_timer->start(30);

hero\_display\_timer->start(3000);

hero\_action\_label->setText("You used ATTACK! -5");

hero\_action\_label->setStyleSheet("font: 8pt; color: red;");

hero\_action\_label->show();

if(getStandardEnemy() == true){

if(!enemy\_timer->isActive()){

enemy\_timer->start(5000);

standardAttackTimer->start(3000);

}

}

if(getBossEnemy() == true){

if(!bossTimer->isActive()){

bossTimer->start(6000);

qDebug() <<"supposed to do this";

}

}

}

void Combat\_Screen::execute\_magic(){

//Create and execute the magic menu dialog and set an integer to hold the value returned

Magic \*magic\_dialog = new Magic;

int n=magic\_dialog->*exec*();

//if the fire button is cliced a 1 is returned

if(n==1){

calculate\_enemy\_damage(get\_hero\_fire());

hero\_action\_label->setText("You used FIRE -20");

hero\_action\_label->setStyleSheet("font: 8pt; color: red;");

enemy\_health\_display->setNum(get\_enemy\_health());

QObject::connect(hero\_display\_timer,SIGNAL(timeout()),

this,SLOT(set\_hero\_action\_display()));

magic\_button->setEnabled(false);

magic\_timer->start(15000);

hero\_display\_timer->start(3000);

if(getStandardEnemy() == true){

if(!enemy\_timer->isActive()){

enemy\_timer->start(5000);

standardAttackTimer->start(3000);

}

}

}if(n==2){

calculate\_enemy\_damage(get\_hero\_ice());

hero\_action\_label->setText("You used ICE -8");

hero\_action\_label->setStyleSheet("font: 8pt; color: red;");

enemy\_health\_display->setNum(get\_enemy\_health());

QObject::connect(hero\_display\_timer,SIGNAL(timeout()),

this,SLOT(set\_hero\_action\_display()));

magic\_button->setEnabled(false);

magic\_timer->start(15000);

hero\_display\_timer->start(3000);

if(!enemy\_timer->isActive()){

enemy\_timer->start(6000);

}

}

magic\_bar\_timer = new QTimer;

QObject::connect(magic\_timer,SIGNAL(timeout()),

this,SLOT(enable\_magic()));

QObject::connect(magic\_bar\_timer,SIGNAL(timeout()),

this,SLOT(update\_magic\_bar()));

magic\_bar\_timer->start(130);

if(get\_enemy\_health()<=0){

win();

Win\_Screen dialog(this);

if(dialog.*exec*()==1){

*accept*();

}

}

}

void Combat\_Screen::execute\_defend(){

defend\_bar\_timer = new QTimer;

QObject::connect(defend\_timer,SIGNAL(timeout()),

this,SLOT(enable\_defend()));

QObject::connect(defend\_bar\_timer,SIGNAL(timeout()),

this,SLOT(update\_defend\_bar()));

defend\_bar\_timer->start(180);

defend\_timer->start(20000);

set\_defend(true);

defend\_button->setEnabled(false);

}

void Combat\_Screen::execute\_item(){

Item\_screen \*dialog = new Item\_screen;

dialog->*exec*();

}

void Combat\_Screen::calculate\_enemy\_attack(){

QObject::connect(enemy\_display\_timer,SIGNAL(timeout()),

this,SLOT(set\_enemy\_action\_display()));

QObject::connect(hero\_display\_timer,SIGNAL(timeout()),

this,SLOT(set\_hero\_action\_display()));

if(get\_defend()==true){

enemy\_action\_label->setText("Enemy defended! -0");

enemy\_action\_label->setStyleSheet("font: 8pt; color: red;");

hero\_action\_label->setText("You defended the attack!");

hero\_action\_label->setStyleSheet("font: 8pt; color: red;");

enemy\_display\_timer->start(3000);

hero\_display\_timer->start(3000);

set\_defend(false);

}else{

calculate\_hero\_damage(get\_enemy\_attack());

hero\_health\_bar->setValue(get\_hero\_health());

enemy\_action\_label->setText("Enemy Used Random Attack");

enemy\_action\_label->setStyleSheet("font: 8pt; color: red;");

enemy\_display\_timer->start(3000);

if(get\_hero\_health()<=0){

lose\_screen dialog(this);

set\_lives(get\_lives()-1);

if(dialog.*exec*()==1){

if(get\_lives()==0){

std::cout<<"YOU LOSE!"<<std::endl;

}

close();

}

}

}

}

void Combat\_Screen::calculate\_standard\_attack(){

QObject::connect(enemy\_display\_timer,SIGNAL(timeout()),

this,SLOT(set\_enemy\_action\_display()));

QObject::connect(hero\_display\_timer,SIGNAL(timeout()),

this,SLOT(set\_hero\_action\_display()));

if(get\_defend()==true){

enemy\_action\_label->setText("Enemy attack defended!");

enemy\_action\_label->setStyleSheet("font: 8pt; color: red;");

hero\_action\_label->setText("you defended the attack!");

hero\_action\_label->setStyleSheet("font: 8pt; color: red;");

enemy\_display\_timer->start(3000);

hero\_display\_timer->start(3000);

set\_defend(false);

}else{

calculate\_hero\_damage(3);

hero\_health\_bar->setValue(get\_hero\_health());

enemy\_action\_label->setText("Enemy Used small attack! -3");

enemy\_action\_label->setStyleSheet("font: 8pt; color: red;");

enemy\_display\_timer->start(3000);

if(get\_hero\_health()<=0){

lose\_screen dialog(this);

set\_lives(get\_lives()-1);

if(dialog.*exec*()==1){

if(get\_lives()==0){

std::cout<<"YOU LOSE!"<<std::endl;

}

close();

}

}

}

}

void Combat\_Screen::bossAttacks(){

QObject::connect(enemy\_display\_timer,SIGNAL(timeout()),

this,SLOT(set\_enemy\_action\_display()));

QObject::connect(hero\_display\_timer,SIGNAL(timeout()),

this,SLOT(set\_hero\_action\_display()));

if(get\_defend()==true){

enemy\_action\_label->setText("Boss was defended! -0");

enemy\_action\_label->setStyleSheet("font: 8pt; color: red;");

hero\_action\_label->setText("you defended the attack!");

hero\_action\_label->setStyleSheet("font: 8pt; color: red;");

enemy\_display\_timer->start(3000);

hero\_display\_timer->start(3000);

set\_defend(false);

}else{

int random;

random = rand()&100+1;

if(random <=20){

calculate\_hero\_damage(getMassiveAttack());

hero\_health\_bar->setValue(get\_hero\_health());

enemy\_action\_label->setText("Boss Used Massive Attack! -20");

enemy\_action\_label->setStyleSheet("font: 8pt; color: red;");

enemy\_display\_timer->start(3000);

}else if(random >20 && random <= 60){

calculate\_hero\_damage(getMediumAttack());

hero\_health\_bar->setValue(get\_hero\_health());

enemy\_action\_label->setText("Boss Used Medium Attack! -10");

enemy\_action\_label->setStyleSheet("font: 8pt; color: red;");

enemy\_display\_timer->start(3000);

}else if(random >60){

calculate\_hero\_damage(getSmallAttack());

hero\_health\_bar->setValue(get\_hero\_health());

enemy\_action\_label->setText("Boss Used Small Attack! -5");

enemy\_action\_label->setStyleSheet("font: 8pt; color: red;");

enemy\_display\_timer->start(3000);

}

if(get\_hero\_health()<=0){

lose\_screen dialog(this);

set\_lives(get\_lives()-1);

if(dialog.*exec*()==1){

if(get\_lives()==0){

std::cout<<"YOU LOSE!"<<std::endl;

}

close();

}

}

}

}

void Combat\_Screen::enable\_attack(){

atkValue=0;

attack\_bar->setValue(atkValue);

delete attack\_bar\_timer;

attack\_bar\_timer = NULL;

attack\_button->setEnabled(true);

attack\_timer->stop();

}

void Combat\_Screen::enable\_magic(){

mgcValue=0;

magic\_bar->setValue(mgcValue);

delete magic\_bar\_timer;

magic\_bar\_timer = NULL;

magic\_button->setEnabled(true);

magic\_timer->stop();

}

void Combat\_Screen::enable\_defend(){

dfnValue=0;

defend\_bar->setValue(dfnValue);

delete defend\_bar\_timer;

defend\_bar\_timer = NULL;

defend\_button->setEnabled(true);

defend\_timer->stop();

}

void Combat\_Screen::set\_enemy\_action\_display(){

enemy\_action\_label->setText(" ");

enemy\_display\_timer->stop();

}

void Combat\_Screen::set\_hero\_action\_display(){

hero\_action\_label->setText(" ");

hero\_display\_timer->stop();

}

void Combat\_Screen::update\_attack\_bar(){

atkValue++;

attack\_bar->setValue(atkValue);

}

void Combat\_Screen::update\_defend\_bar(){

dfnValue++;

defend\_bar->setValue(dfnValue);

}

void Combat\_Screen::update\_magic\_bar(){

mgcValue++;

magic\_bar->setValue(mgcValue);

}

void Combat\_Screen::win(){

set\_enemy\_health(0);

enemy\_health\_display->setNum(get\_enemy\_health());

hero\_action\_label->setText(" ");

enemy\_action\_label->setText(" ");

attack\_button->setEnabled(true);

defend\_button->setEnabled(true);

magic\_button->setEnabled(true);

item\_button->setEnabled(true);

magic\_timer->stop();

attack\_timer->stop();

enemy\_timer->stop();

standardAttackTimer->stop();

defend\_timer->stop();

enemy\_display\_timer->stop();

hero\_display\_timer->stop();

delete attack\_bar\_timer;

attack\_bar\_timer = NULL;

delete defend\_bar\_timer;

defend\_bar\_timer = NULL;

delete magic\_bar\_timer;

magic\_bar\_timer = NULL;

attack\_bar->setValue(0);

defend\_bar->setValue(0);

magic\_bar->setValue(0);

set\_enemy\_health(40);

}

void Combat\_Screen::lose(){

set\_hero\_health(0);

hero\_health\_bar->setValue(get\_hero\_health());

hero\_action\_label->setText(" ");

enemy\_action\_label->setText(" ");

attack\_button->setEnabled(true);

defend\_button->setEnabled(true);

magic\_button->setEnabled(true);

item\_button->setEnabled(true);

magic\_timer->stop();

attack\_timer->stop();

attack\_timer->stop();

defend\_timer->stop();

enemy\_display\_timer->stop();

hero\_display\_timer->stop();

delete attack\_bar\_timer;

attack\_bar\_timer = NULL;

delete defend\_bar\_timer;

defend\_bar\_timer = NULL;

delete magic\_bar\_timer;

magic\_bar\_timer = NULL;

attack\_bar->setValue(0);

defend\_bar->setValue(0);

magic\_bar->setValue(0);

WinScreenEndGame \*endGame = new WinScreenEndGame();

}

***Help\_Screen :***

#include "help\_screen.h"

#include "ui\_help\_screen.h"

#include <QDialog>

Help\_Screen::Help\_Screen(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::Help\_Screen)

{

ui->setupUi(this);

//Setting Background image for TitleScreen Ui - Miguel

QPixmap bg(":/Background.png");

bg = bg.scaled(this->size(), Qt::IgnoreAspectRatio);

QPalette palette;

palette.setBrush(QPalette::Background, bg);

this->setPalette(palette);

}

Help\_Screen::~*Help\_Screen*()

{

delete ui;

}

void Help\_Screen::on\_ReturnToMain\_clicked()

{

this->close();

}

void Help\_Screen::on\_ReturnToMain\_pressed()

{

this->close();

}

Load\_Screen :

#include "loadfiles.h"

#include "ui\_loadfiles.h"

#include "qDir"

#include <QProcess>

LoadFiles::LoadFiles(int savedGameCount, QWidget \*parent) :

QDialog(parent),

ui(new Ui::LoadFiles)

{

ui->setupUi(this);

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

QString item("SAVED GAME %1");

ui->listWidget->addItem(item.arg(i));

}

}

LoadFiles::~*LoadFiles*()

{

delete ui;

}

void LoadFiles::on\_listWidget\_itemDoubleClicked(QListWidgetItem \*item)

{

this->*done*(item->listWidget()->row(item));

}

***Lose\_Screen :***

#include "lose\_screen.h"

#include "ui\_lose\_screen.h"

lose\_screen::lose\_screen(QWidget \*parent) :

QDialog(parent),

ui(new Ui::lose\_screen)

{

ui->setupUi(this);

}

lose\_screen::~*lose\_screen*()

{

delete ui;

}

void lose\_screen::on\_close\_button\_clicked()

{

*done*(1);

}

***Magic :***

#include <QPushButton>

#include <QLayout>

#include "magic.h"

#include "ui\_magic.h"

Magic::Magic(QWidget \*parent) :

QDialog(parent),

ui(new Ui::Magic)

{

ui->setupUi(this);

fire\_button = new QPushButton;

fire\_button->setText("FIRE");

ice\_button = new QPushButton;

ice\_button->setText("ICE");

close\_button = new QPushButton;

close\_button->setText("CLOSE");

QHBoxLayout \*button\_layout = new QHBoxLayout;

button\_layout->addWidget(fire\_button);

button\_layout->addWidget(ice\_button);

QVBoxLayout \*main\_layout = new QVBoxLayout;

main\_layout->addStretch();

main\_layout->addLayout(button\_layout);

main\_layout->addWidget(close\_button);

setLayout(main\_layout);

QObject::connect(fire\_button,SIGNAL(clicked()),

this,SLOT(execute\_fire()));

QObject::connect(ice\_button,SIGNAL(clicked()),

this,SLOT(execute\_ice()));

QObject::connect(close\_button,SIGNAL(clicked()),

this,SLOT(close()));

}

Magic::~*Magic*()

{

delete ui;

}

void Magic::execute\_fire(){

QDialog::*done*(1);

}

void Magic::execute\_ice(){

QDialog::*done*(2);

}

Main :

#include "mainwindow.h"

#include <QApplication>

#include <QRect>

#include <QDebug>

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

MainWindow w;

QRect screenGeometry = QApplication::desktop()->screenGeometry();

int x = (screenGeometry.width()-w.width()) / 2;

int y = (screenGeometry.height()-w.height()) / 2;

qDebug() << x << ":" << y;

w.move(x, y);

w.show();

return a.exec();

}

***MainWindow :***

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include "QTextStream"

#include <iostream>

#include <sstream>

#include <QMainWindow>

#include <QGridLayout>

#include <QLabel>

#include <QString>

#include <QPushButton>

#include <QKeyEvent>

#include <QDialog>

#include <QDesktopWidget>

#include "declarations.h"

#include "combat\_screen.h"

#include "help\_screen.h"

#include <vector>

#include <QDir>

#include <QDebug>

#include <QtSql>

#include <QSqlQueryModel>

#include <QString>

#include "savedgame.h"

#include <QTextStream>

#include "loadfiles.h"

#include "winscreenendgame.h" //The end game screen

#include <QApplication>

MainWindow::MainWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

ui->setupUi(this);

this->\_curX = 2;

this->\_curY = 42;

this->\_curHealth = 100;

QDir qDir;

savedGames.Load(QString(qDir.absolutePath()+"/CONFIG/"), QString("sqlconfig.cfg"));

if (savedGames.GetIsLoaded()) {

if (savedGames.GetSavedGameCount() > 0) {

ui->pushButton\_4->setEnabled(true);

ui->pushButton\_4->setStyleSheet("color:yellow");

}

}

//Setting Background image for TitleScreen Ui - Miguel

QPixmap bg(":/Background.png");

bg = bg.scaled(this->size(), Qt::IgnoreAspectRatio);

QPalette palette;

palette.setBrush(QPalette::Background, bg);

this->setPalette(palette);

this->currentOption = 0;

this->numOptions = 4;

qApp->installEventFilter(this);

this->inGame = false;

//Play sound - Jacari

QSound::play(qDir.absolutePath() + "/sounds/8bitAudio1.wav");

this->setFixedSize(800,500);

this->centerWidget(true);

mainChar = new Combat\_Screen;

mainChar->set\_hero\_health(this->\_curHealth);

mainChar->set\_attacks(8,20);

mainChar->set\_defend(false);

mainChar->set\_lives(5);

mainChar->hero\_health\_bar->setValue(mainChar->get\_hero\_health());

// end old MainWindow.cpp

}

MainWindow::~*MainWindow*()

{

delete ui;

}

void MainWindow::SetupGameBoard() {

this->gridLayout = new QGridLayout;

this->gridLayout->setHorizontalSpacing(0);

this->gridLayout->setVerticalSpacing(0);

this->boardBoundX = 44;

this->boardBoundY = 44;

//Instantiating Qimages

this->\_path = QImage(":/PathRecent20x20.png");

this->\_wall = QImage(":/WallRecent20x20.png");

this->\_character = QImage(":/CharacterRecent20x20.png");

this->\_coffin = QImage(":/CoffinRecent20x20.png"); //Considering using this as a pickup item - Miguel

this->\_topLeft = QImage(":/TopLeft.png"); //Upper Tile

this->\_topMiddle = QImage(":/TopMiddle.png"); //Upper Tile

this->\_topRight = QImage(":/TopRight.png"); //Upper Tile

//Bottom Tiles

this->\_botLeft = QImage(":/BottomLeft.png");

this->\_botRight = QImage(":/BottomRight.png");

//Left Most Tiles

this->\_midLeft = QImage(":/LeftMiddle.png");

this->\_midRight = QImage(":/RightMiddle.png");

//Right Most Tiles

//Define Enemy Map Images

this->\_enemyOne = QImage(":/EnemyMap.png");

this->\_enemyTwo = QImage(":/EnemyMap2.png");

this->\_enemyBoss = QImage(":/EnemyBoss.png");

for (int y = 0; y <= this->boardBoundY; y++) {

for (int x = 0; x <= this->boardBoundX; x++) {

QLabel \*textBlock = new QLabel;

if (y == this->\_curY && x == this->\_curX) {

qDebug() << "Found characer!" << endl;

textBlock->setPixmap(QPixmap::fromImage(this->\_character));

}

else if (this->boardMatrix[y][x] == 1) {

textBlock->setPixmap(QPixmap::fromImage(this->\_wall));

}

else if (this->boardMatrix[y][x] == 2){

textBlock->setPixmap(QPixmap::fromImage(this->\_coffin));

}

else if (this->boardMatrix[y][x] == 3){ //Top left tile

textBlock->setPixmap(QPixmap::fromImage(this->\_topLeft));

}

else if (this->boardMatrix[y][x] == 4){ //Top middle tile

textBlock->setPixmap(QPixmap::fromImage(this->\_topMiddle));

}

else if (this->boardMatrix[y][x] == 5){

textBlock->setPixmap(QPixmap::fromImage(this->\_topRight));

}

else if (this->boardMatrix[y][x] == 6){

textBlock->setPixmap(QPixmap::fromImage(this->\_botLeft));

}

else if (this->boardMatrix[y][x] == 7){

textBlock->setPixmap(QPixmap::fromImage(this->\_botRight));

}

else if (this->boardMatrix[y][x] == 8){

textBlock->setPixmap(QPixmap::fromImage(this->\_midLeft));

}

else if (this->boardMatrix[y][x] == 9){

textBlock->setPixmap(QPixmap::fromImage(this->\_midRight));

}

else if (this->boardMatrix[y][x] == 91){

textBlock->setPixmap(QPixmap::fromImage(this->\_enemyOne));

}

else if (this->boardMatrix[y][x] == 93){

textBlock->setPixmap(QPixmap::fromImage(this->\_enemyTwo));

}

else if (this->boardMatrix[y][x] == 90){

textBlock->setPixmap(QPixmap::fromImage(this->\_enemyBoss));

}

else

textBlock->setPixmap(QPixmap::fromImage(this->\_path));

gridLayout->addWidget(textBlock, y, x, 1, 1, Qt::AlignCenter);

}

QPushButton \*backButton = new QPushButton;

backButton->setText("EXIT");

backButton->setStyleSheet("background-color:blue; color:yellow; font-family: Terminal; font-size:20px;");

gridLayout->addWidget(backButton, 46, 0, 1, 5, Qt::AlignCenter | Qt::AlignVCenter);

QLabel \*healthStatusLabel = new QLabel("CURRENT HEALTH: ");

healthStatusLabel->setStyleSheet("color:yellow; font-family: Terminal; font-size:20px;");

gridLayout->addWidget(healthStatusLabel, 46, 5, 1, 20, Qt::AlignRight | Qt::AlignVCenter);

healthStatus = new QLabel("");

healthStatus->setStyleSheet("color:yellow; font-family: Terminal; font-size:20px;");

gridLayout->addWidget(healthStatus, 46, 25, 1, 14, Qt::AlignLeft | Qt::AlignVCenter);

QPushButton \*saveButton = new QPushButton("SAVE");

saveButton->setStyleSheet("background-color:blue; color:yellow; font-family: Terminal; font-size:20px;");

gridLayout->addWidget(saveButton, 46, 39, 1, 5, Qt::AlignCenter | Qt::AlignVCenter);

connect(backButton, SIGNAL(clicked()), this, SLOT(on\_back\_clicked()));

connect(saveButton, SIGNAL(clicked()), this, SLOT(on\_save\_clicked()));

}

gameBoardWidget = new QWidget;

this->updateHealth(this->mainChar->get\_hero\_health());

gameBoardWidget->setLayout(gridLayout);

gameBoardWidget->setWindowTitle("Dungeon");

}

void MainWindow::*closeEvent*(QCloseEvent \*event) {

if (this->inGame) {

menuWidget->setParent(this);

this->setCentralWidget(menuWidget);

this->inGame = false;

this->setFixedSize(800,500);

QRect screenGeometry = QApplication::desktop()->screenGeometry();

int x = (screenGeometry.width()-800) / 2;

int y = (screenGeometry.height()-500) / 2;

this->move(x, y);

event->ignore();

}

else

event->accept();

}

void MainWindow::centerWidget(bool useSizeHint)

{

if(this->centralWidget()->isFullScreen())

return;

QSize size;

if(useSizeHint)

size = this->centralWidget()->*sizeHint*();

else

size = this->centralWidget()->size();

QDesktopWidget \*d = QApplication::desktop();

int ws = d->width(); // returns screen width

int h = d->height(); // returns screen height

int mw = size.width();

int mh = size.height();

int cw = (ws/2) - (mw/2);

int ch = (h/2) - (mh/2);

this->move(cw,ch);

}

int MainWindow::on\_pushButton\_clicked()

{

this->SetupGameBoard();

this->inGame = true;

this->menuWidget = this->centralWidget();

if (this->centralWidget())

this->centralWidget()->setParent(NULL); // Preserve current central widget

this->setCentralWidget(this->gameBoardWidget);

this->setFixedSize(900,900);

this->centerWidget(true);

return 0;

}

//Push Button to open the loading dialog

int MainWindow::on\_pushButton\_2\_clicked()

{

QApplication::quit();

return 0;

}

//Push button to open the options dialog

int MainWindow::on\_pushButton\_3\_clicked()

{

//Open Title screen through modal approach - Miguel

//Help\_Screen helpScreen;

//helpScreen.setModal(true);

//helpScreen.exec();

//Modalus show screen with heap memory - Miguel

helpScreenMain = new Help\_Screen(this);

helpScreenMain->show();

//helpScreen.show();

return 0;

}

int MainWindow::on\_pushButton\_4\_clicked()

{

LoadFiles loadGameScreen(this->savedGames.GetSavedGameCount(), this);

loadGameScreen.setModal(true);

this->LoadGame(loadGameScreen.*exec*());

ui->pushButton->click();

return 0;

}

int MainWindow::on\_back\_clicked()

{

this->close();

return 0;

}

int MainWindow::on\_save\_clicked()

{

if (this->inGame) {

this->savedGames.AddGame(this->\_curX, this->\_curY, this->mainChar->get\_hero\_health());

this->close();

ui->pushButton\_4->setEnabled(true);

ui->pushButton\_4->setStyleSheet("color:yellow");

}

return 0;

}

void MainWindow::updateHealth(int health) {

this->healthStatus->setText(QString::number(health));

}

void MainWindow::*keyPressEvent*(QKeyEvent \*event) {

if (this->inGame) {

if (event->key() == Qt::Key\_Down)

MainWindow::moveCharacter(MOVE\_DOWN);

else if (event->key() == Qt::Key\_Left)

MainWindow::moveCharacter(MOVE\_LEFT);

else if (event->key() == Qt::Key\_Right)

MainWindow::moveCharacter(MOVE\_RIGHT);

else if (event->key() == Qt::Key\_Up)

MainWindow::moveCharacter(MOVE\_UP);

this->updateHealth(this->mainChar->get\_hero\_health());

} else {

if (event->key() == Qt::Key\_Down) {

int oldOption = this->currentOption;

this->currentOption++;

if (this->currentOption == this->numOptions)

this->currentOption = 0;

this->updateMainMenu(oldOption, this->currentOption, false);

//MainWindow::moveCharacter(MOVE\_DOWN);

}

else if (event->key() == Qt::Key\_Up) {

int oldOption = this->currentOption;

this->currentOption = this->currentOption - 1;

if (this->currentOption == -1) {

this->currentOption = this->numOptions-1;

}

this->updateMainMenu(oldOption, this->currentOption, true);

} else if (event->key() == Qt::Key\_Enter || event->key() == Qt::Key\_Return ) {

if (this->currentOption == 0) {

ui->pushButton->click();

} else if (this->currentOption == 1) {

ui->pushButton\_3->click();

} else if (this->currentOption == 2) {

ui->pushButton\_4->click();

} else if (this->currentOption == 3) {

ui->pushButton\_2->click();

}

}

}

}

void MainWindow::updateMainMenu(int &oldOption, int &newOption, bool DirectionUp) {

if (newOption == 2 && !ui->pushButton\_4->isEnabled())

{

newOption = (DirectionUp ? 1 : 3);

oldOption = (DirectionUp ? 3 : 1);

} else if (newOption == 3 && !ui->pushButton\_4->isEnabled() && !DirectionUp) {

oldOption = 1;

} else if (newOption == 1 && !ui->pushButton\_4->isEnabled() && DirectionUp) {

oldOption = 3;

}

qDebug() << (DirectionUp ? "UP" : "DOWN") << "::" << oldOption << ":" << newOption << endl;

QFont \*newFont = new QFont("Terminal", 20, QFont::Bold);

QFont \*oldFont = new QFont("Terminal", 18, QFont::Normal);

if (newOption == 0) {

ui->pushButton->setText("[ NEW GAME ]");

ui->pushButton->setFont(\*newFont);

//ui->pushButton->setStyleSheet("");

} else if (newOption == 1) {

ui->pushButton\_3->setText("[ HELP ]");

ui->pushButton\_3->setFont(\*newFont);

} else if (newOption == 2) {

ui->pushButton\_4->setText("[ LOAD ]");

ui->pushButton\_4->setFont(\*newFont);

} else if (newOption == 3) {

ui->pushButton\_2->setText("[ QUIT ]");

ui->pushButton\_2->setFont(\*newFont);

}

if (oldOption == 0) {

ui->pushButton->setText("NEW GAME");

ui->pushButton->setFont(\*oldFont);

} else if (oldOption == 1) {

ui->pushButton\_3->setText("HELP");

ui->pushButton\_3->setFont(\*oldFont);

} else if (oldOption == 2) {

ui->pushButton\_4->setText("LOAD");

ui->pushButton\_4->setFont(\*oldFont);

} else if (oldOption == 3) {

ui->pushButton\_2->setText("QUIT");

ui->pushButton\_2->setFont(\*oldFont);

}

}

bool MainWindow::*eventFilter*(QObject \*obj,

QEvent \*event)

{

QKeyEvent \*keyEvent = NULL;//event data, if this is a keystroke event

bool result = false;//return true to consume the keystroke

if (event->type() == QEvent::KeyPress)

{

//QTextStream(stdout) << "in eventFilter keyPress!";

keyEvent = dynamic\_cast<QKeyEvent\*>(event);

this->*keyPressEvent*(keyEvent);

result = true;

}//if type()

else if (event->type() == QEvent::KeyRelease)

{

keyEvent = dynamic\_cast<QKeyEvent\*>(event);

this->*keyReleaseEvent*(keyEvent);

result = true;

}//else if type()

//### Standard event processing ###

else

result = QObject::*eventFilter*(obj, event);

return result;

}//eventFilter

int MainWindow::checkMove(int direction) {

int moveStatus;

if (direction == MOVE\_LEFT) {

if (((\_curX - 1) >= 0) && (this->boardMatrix[\_curY][\_curX-1] != 1)

&& (this->boardMatrix[\_curY][\_curX-1] != 2)

&& (this->boardMatrix[\_curY][\_curX-1] != 3)

&& (this->boardMatrix[\_curY][\_curX-1] != 4)

&& (this->boardMatrix[\_curY][\_curX-1] != 5)

&& (this->boardMatrix[\_curY][\_curX-1] != 6)

&& (this->boardMatrix[\_curY][\_curX-1] != 7)

&& (this->boardMatrix[\_curY][\_curX-1] != 8)

&& (this->boardMatrix[\_curY][\_curX-1] != 9)

&& (this->boardMatrix[\_curY][\_curX-1] != 10)) {

\_curX--;

moveStatus = MOVE\_AUTHORIZED;

} else

moveStatus = \_curX - 1 >= 0 ? MOVE\_ERROR\_OUTOFBOUNDS : MOVE\_ERROR\_BLOCKED;

} else if (direction == MOVE\_RIGHT) {

if (((\_curX + 1) <= this->boardBoundX)

&& (this->boardMatrix[\_curY][\_curX+1] != 1)

&& (this->boardMatrix[\_curY][\_curX+1] != 2)

&& (this->boardMatrix[\_curY][\_curX+1] != 3)

&& (this->boardMatrix[\_curY][\_curX+1] != 4)

&& (this->boardMatrix[\_curY][\_curX+1] != 5)

&& (this->boardMatrix[\_curY][\_curX+1] != 6)

&& (this->boardMatrix[\_curY][\_curX+1] != 7)

&& (this->boardMatrix[\_curY][\_curX+1] != 8)

&& (this->boardMatrix[\_curY][\_curX+1] != 9)

&& (this->boardMatrix[\_curY][\_curX+1] != 10)) {

\_curX++;

moveStatus = MOVE\_AUTHORIZED;

} else

moveStatus = \_curX + 1 <= boardBoundX ? MOVE\_ERROR\_OUTOFBOUNDS : MOVE\_ERROR\_BLOCKED;

} else if (direction == MOVE\_UP) {

if (((\_curY - 1) >= 0)

&& (this->boardMatrix[\_curY-1][\_curX] != 1)

&& (this->boardMatrix[\_curY-1][\_curX] != 2)

&& (this->boardMatrix[\_curY-1][\_curX] != 3)

&& (this->boardMatrix[\_curY-1][\_curX] != 4)

&& (this->boardMatrix[\_curY-1][\_curX] != 5)

&& (this->boardMatrix[\_curY-1][\_curX] != 6)

&& (this->boardMatrix[\_curY-1][\_curX] != 7)

&& (this->boardMatrix[\_curY-1][\_curX] != 8)

&& (this->boardMatrix[\_curY-1][\_curX] != 9)

&& (this->boardMatrix[\_curY-1][\_curX] != 10)) {

\_curY--;

moveStatus = MOVE\_AUTHORIZED;

} else

moveStatus = \_curY - 1 >= 0 ? MOVE\_ERROR\_OUTOFBOUNDS : MOVE\_ERROR\_BLOCKED;

} else if (direction == MOVE\_DOWN) {

if (((\_curY + 1) <= this->boardBoundY)

&& (this->boardMatrix[\_curY+1][\_curX] != 1)

&& (this->boardMatrix[\_curY+1][\_curX] != 2)

&& (this->boardMatrix[\_curY+1][\_curX] != 3)

&& (this->boardMatrix[\_curY+1][\_curX] != 4)

&& (this->boardMatrix[\_curY+1][\_curX] != 5)

&& (this->boardMatrix[\_curY+1][\_curX] != 6)

&& (this->boardMatrix[\_curY+1][\_curX] != 7)

&& (this->boardMatrix[\_curY+1][\_curX] != 8)

&& (this->boardMatrix[\_curY+1][\_curX] != 9)

&& (this->boardMatrix[\_curY+1][\_curX] != 10)) {

\_curY++;

moveStatus = MOVE\_AUTHORIZED;

} else

moveStatus = \_curY + 1 <= boardBoundY ? MOVE\_ERROR\_OUTOFBOUNDS : MOVE\_ERROR\_BLOCKED;

} else

moveStatus = MOVE\_ERROR\_UNKNOWN;

if (moveStatus != MOVE\_AUTHORIZED)

return moveStatus;

switch(this->boardMatrix[\_curY][\_curX]) {

case CHARACTER\_ENEMY: QTextStream(stdout) << "ENEMY";

break;

case CHARACTER\_BOSS: QTextStream(stdout) << "BOSS";

break;

case CHARACTER\_ITEM: QTextStream(stdout) << "ITEM";

break;

default: return MOVE\_AUTHORIZED;

}

return MOVE\_AUTHORIZED;

}

void MainWindow::moveCharacter(int moveType) {

int prevX = this->getX();

int prevY = this->getY();

int result = this->checkMove(moveType);

if (result == MOVE\_AUTHORIZED) {

QLabel \*oldLabel = (QLabel \*)this->gridLayout->itemAtPosition(prevY, prevX)->*widget*();

QLabel \*newLabel = (QLabel \*)this->gridLayout->itemAtPosition(this->getY(),this->getX())->*widget*();

oldLabel->setPixmap(QPixmap::fromImage(this->\_path));

newLabel->setPixmap(QPixmap::fromImage(this->\_character));

if(this->boardMatrix[\_curY][\_curX]==CHARACTER\_ENEMY){

//set enemy traits

mainChar->setStandardEnemy(true);

mainChar->set\_enemy\_attack(rand()%10+1);

mainChar->set\_enemy\_health(40);

mainChar->enemy\_health\_display->setNum(mainChar->get\_enemy\_health());

mainChar->set\_enemy\_name("Spider");

mainChar->setModal(true);

mainChar->*exec*();

//save health and lives-Adam

this->\_curHealth = mainChar->get\_hero\_health();

int newLives = mainChar->get\_hero\_lives();

//delete previous object

delete mainChar;

//restart a new combat situation

mainChar = new Combat\_Screen;

mainChar->set\_hero\_health(this->\_curHealth);

mainChar->hero\_health\_bar->setValue(mainChar->get\_hero\_health());

mainChar->set\_attacks(8,20);

mainChar->set\_defend(false);

mainChar->set\_lives(newLives);

}

if(this->boardMatrix[\_curY][\_curX]==CHARACTER\_BOSS){

//set boss traits

std::cout<<mainChar->getStandardEnemy()<<std::endl;

mainChar->setStandardEnemy(false);

mainChar->setBossEnemy(true);

mainChar->setBossHealth();

mainChar->setMassiveAttack();

mainChar->setMediumAttack();

mainChar->setSmallAttack();

mainChar->set\_enemy\_health(mainChar->getBossHealth());

mainChar->setBossName("BOSS");

mainChar->enemy\_health\_display->setNum(mainChar->getBossHealth());

mainChar->setModal(true);

mainChar->*exec*();

//save health and lives-Adam

this->\_curHealth = mainChar->get\_hero\_health();

int newLives = mainChar->get\_hero\_lives();

//delete previous object

if(mainChar->Accepted){

//close();

//QApplication::quit();

//qDebug() << "Game has quit";

WinScreenEndGame endgame;

endgame.setModal(true);

endgame.*exec*();

}

delete mainChar;

}

}

}

void MainWindow::LoadGame(int savedGame) {

QString data = savedGames.GetSavedGame(savedGame);

QStringList datas = data.split("|");

this->\_curX = datas.at(0).toInt();

this->\_curY = datas.at(1).toInt();

this->\_curHealth = datas.at(2).toInt();

}

const int MainWindow::boardMatrix[45][45] = {{3,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,5},

{8,1,1,1,1,1,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,9},

{8,1,1,1,1,1,2,0,0,0,0,0,0,91,0,91,0,91,0,0,0,0,0,0,0,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,9},

{8,1,1,1,1,1,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,91,0,91,0,0,0,0,0,0,1,9},

{8,1,1,1,1,1,2,0,0,0,0,0,0,91,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,91,0,91,0,0,0,0,0,1,9},

{8,1,1,1,1,1,2,0,0,0,0,0,0,0,0,91,0,91,0,0,0,0,0,0,0,1,1,1,1,1,1,0,0,0,91,0,91,0,0,0,0,0,0,1,9},

{8,1,1,1,1,1,2,0,0,0,0,0,0,91,0,91,0,91,0,0,0,0,0,0,0,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,9},

{8,1,1,1,1,1,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,9},

{8,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,1,1,1,2,1,1,1,1,1,1,1,0,0,0,0,0,0,0,91,0,91,0,91,0,0,0,1,9},

{8,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,91,0,0,0,0,0,0,1,9},

{8,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,9},

{8,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,9},

{8,1,1,1,1,1,1,1,1,1,1,1,2,1,1,0,0,1,1,1,1,1,1,1,1,2,1,1,1,1,0,0,1,1,1,1,1,0,0,1,1,1,1,1,9},

{8,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,1,1,1,1,1,0,0,1,1,1,1,1,9},

{8,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,1,1,1,1,1,0,0,1,1,1,1,1,9},

{8,1,1,1,1,1,1,1,1,1,1,2,1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,2,1,1,91,0,1,1,1,1,1,91,0,1,1,1,1,1,9},

{8,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,0,0,1,1,1,1,1,0,0,1,1,1,1,1,9},

{8,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,0,91,1,1,1,1,1,91,0,1,1,1,1,1,9},

{8,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,0,0,1,1,1,1,1,0,0,1,1,1,1,1,9},

{8,1,1,1,0,0,0,0,91,0,91,0,91,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,0,0,1,1,1,1,1,0,0,1,1,1,1,1,9},

{8,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,9},

{8,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,9},

{8,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,9},

{8,1,1,1,2,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,9},

{8,1,1,1,2,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,9},

{8,1,1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,9},

{8,1,1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,9},

{8,1,1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,9},

{8,1,1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,9},

{8,1,1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,9},

{8,1,1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,9},

{8,1,1,1,1,91,91,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,9},

{8,1,1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,9},

{8,1,1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,9},

{8,1,1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,9},

{8,1,1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,9},

{8,1,1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,2,0,2,0,2,0,2,0,0,2,0,2,0,2,0,2,0,2,1,1,1,1,1,1,9},

{8,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,9},

{8,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,9},

{8,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,0},

{8,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,90,0},

{8,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,9},

{8,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,2,0,2,0,2,0,2,0,2,0,2,0,2,0,2,2,0,2,1,1,1,1,1,1,9},

{8,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,9},

{6,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,7}};

void MainWindow::on\_pushButton\_5\_clicked()

{

QDir qDir;

QString file = qDir.absolutePath().append("/config/notepad2.exe").append(" sqlconfig.cfg");

QProcess process;

qDebug() << "Opening: " << file << endl;

process.startDetached(file);

}

***SavedGame :***

#include "savedgame.h"

#include <QFile>

#include <QString>

#include <QTextStream>

#include <QDir>

#include <QDebug>

#include <QStringList>

#include <QSqlDatabase>

#include <QDebug>

#include <QSqlQuery>

#include <QSqlQueryModel>

SavedGame::SavedGame()

{

}

bool SavedGame::Load(QString const& configPath, QString const& configFileName)

{

QDir qDir;

IsLoaded = false;

configFile = new QFile(configPath + configFileName);

QStringList sqlValues;

if (!configFile->exists()) {

return false;

} else {

if (!configFile->*open*(QIODevice::ReadOnly)) {

qDebug() << "Cannot open file: " << qPrintable(configFile->errorString()) << endl;

} else {

QTextStream in(configFile);

if (!in.atEnd()) {

QString inLine = in.readLine();

sqlValues = inLine.split('|');

}

configFile->*close*();

if (sqlValues.count() == 4) {

QString host = sqlValues.at(0);

QString user = sqlValues.at(1);

QString password = sqlValues.at(2);

QString dbname = sqlValues.at(3);

this->db = QSqlDatabase::addDatabase("QMYSQL");

db.setHostName(host);

db.setDatabaseName(dbname);

db.setUserName(user);

db.setPassword(password);

if (db.open()) {

this->IsLoaded = true;

this->savedGames.clear();

QSqlQuery query;

query.exec("SELECT \* FROM saved\_games order by id desc");

while (query.next()) {

QString game("%1|%2|%3");

game = game.arg(query.value(1).toString()).arg(query.value(2).toString()).arg(query.value(3).toString());

this->savedGames.append(game);

}

db.close();

}

qDebug() << "Number Of Games Loaded: " << this->savedGames.count();

return true;

}

}

}

return false;

}

QStringList SavedGame::GetSavedGameList() {

return this->savedGames;

}

QString SavedGame::GetSavedGame(int gameNumber) {

return this->savedGames.at(gameNumber);

}

int SavedGame::GetSavedGameCount() {

return this->savedGames.count();

}

bool SavedGame::GetIsLoaded()

{

return this->IsLoaded;

}

void SavedGame::AddGame(int x, int y, int experience) {

QString game("%1|%2|%3");

game = game.arg(x).arg(y).arg(experience);

qDebug() << "Added Game: " << game;

try {

if (db.open()) {

qDebug() << "AddGame: Database opened!" << endl;

this->IsLoaded = true;

this->savedGames.clear();

QSqlQuery query;

query.prepare("INSERT INTO saved\_games (x, y, health) values (:x,:y,:health)");

query.bindValue(":x",x);

query.bindValue(":y",y);

query.bindValue(":health",experience);

if (query.exec()) {

qDebug() << "AddGame: Game added!" << endl;

}

db.close();

this->savedGames.append(game);

}

else

{

qDebug() << "AddGame: Unable to open database!" << endl;

}

} catch (std::exception exc) {

qDebug() << "Unable to save game: " << exc.*what*() << endl;

}

}

***Win\_Screen :***

#include "win\_screen.h"

#include "ui\_win\_screen.h"

#include <QLayout>

Win\_Screen::Win\_Screen(QWidget \*parent) :

QDialog(parent),

ui(new Ui::Win\_Screen)

{

ui->setupUi(this);

close\_button = new QPushButton;

close\_button->setText("CLOSE");

close\_button->setDefault(true);

QVBoxLayout \*main\_layout = new QVBoxLayout;

main\_layout->addStretch();

main\_layout->addWidget(close\_button);

setLayout(main\_layout);

QObject::connect(close\_button,SIGNAL(clicked()),

this,SLOT(finish()));

}

Win\_Screen::~*Win\_Screen*()

{

delete ui;

}

void Win\_Screen::finish(){

QDialog::*done*(1);

}

WinScreenEnd :

#include "winscreenendgame.h"

#include "ui\_winscreenendgame.h"

#include <QApplication>

WinScreenEndGame::WinScreenEndGame(QWidget \*parent) :

QDialog(parent),

ui(new Ui::WinScreenEndGame)

{

ui->setupUi(this);

//Setting Background image for TitleScreen Ui - Miguel

QPixmap bg(":/Background.png");

bg = bg.scaled(this->size(), Qt::IgnoreAspectRatio);

QPalette palette;

palette.setBrush(QPalette::Background, bg);

this->setPalette(palette);

}

WinScreenEndGame::~*WinScreenEndGame*()

{

delete ui;

}

void WinScreenEndGame::on\_ReturnToMain\_clicked()

{

//Exit the entire program

//WinScreenEndGame endgame;

//endgame.setModal(true);

//endgame.exec();

//End the game it is now working

QApplication::quit();

}

**Flow Chart:**

