**Final Project**

**<Room 3 RPG style game>**

**CSC-17B-48982**

**Najera, Enrique**

**16 December 2015**

**Introduction**

Title: Room 3

*Room 3* is an RPG puzzle game. The main purpose is to make sure both sides of the room are “mirrored” by flipping the furniture.

I am doing this because I wanted to try out basic game logic seen in more advanced games and have a feel of what it is really like.

This is important because it helps me gain a better understanding of how computer applications work and interact.

**Summary**

Project Size: ~1753 lines

It meets the criteria for a project because it includes everything we have learned thus far plus application functionality that hasn’t been taught.

Variables: ~27

Constructs (from CSC17B): ~14

It was a bit challenging to do, but after a while it was easier to get what I wanted. The biggest challenge, however, was getting MySQL Workbench to export a sql file. My error was the mysqldump executable was an earlier version than the SQL on my machine but that mysqldump executable was the latest version. In the end, I copied the .sql from the group project and coded in what I needed.

This project took me about 10 days, including the music and art. I gave up on my first two projects because they couldn’t really be expanded.

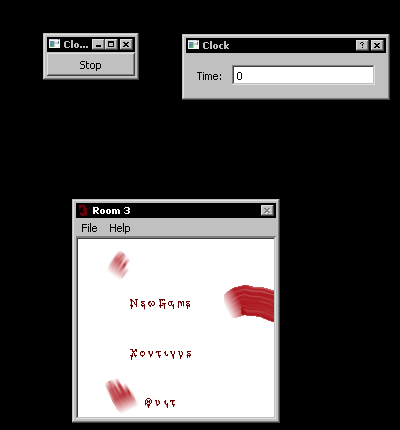
Other aspects I used not covered in class are 2D images and implementing sound.

**Description**

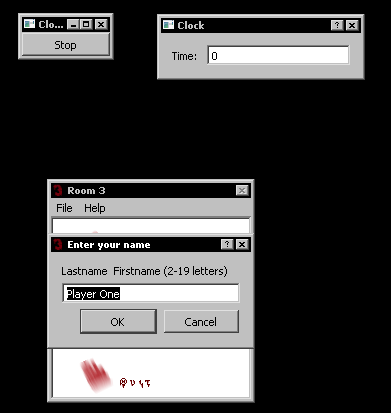
I took your advice on moving everything else except for the player and I liked the idea then moved on from there. I moved a background image based on the player’s “position,” added objects and moved those objects based on the background image’s position, created a dummy collision detector, by stopping the player when they get to a certain position, to prevent the player from crossing objects and walls, and finally made a collision interact-able.

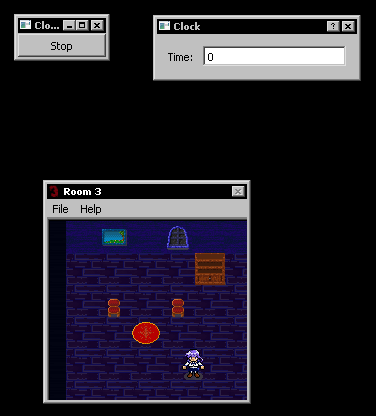
Sample Input/Output

Main program



Click NEW GAME button



Click ok with right input (wrong input resets dialog as above)

UP arrow key (hold shows animation that can’t be shown in still image)



DOWN arrow key (hold shows animation that can’t be shown in still image)



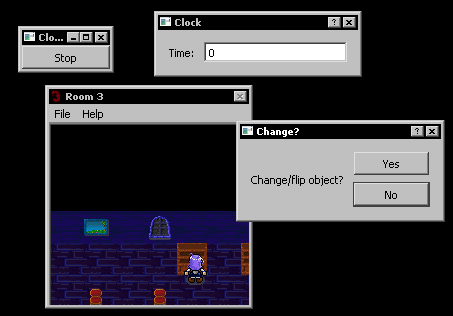
LEFT arrow key (hold shows animation that can’t be shown in still image)



RIGHT arrow key (hold shows animation that can’t be shown in still image)



‘z’ key while in front of flip-able object (right cupboard, table, plant)



Clicking ‘no’ on “Change?” dialog



Clicking ‘yes’ on “Change?” dialog (changes image; I moved player backnges image; I moved pnt)e mation that can'in positionhe player from crossing objects and walls, rsion. Towever, was for visibility)



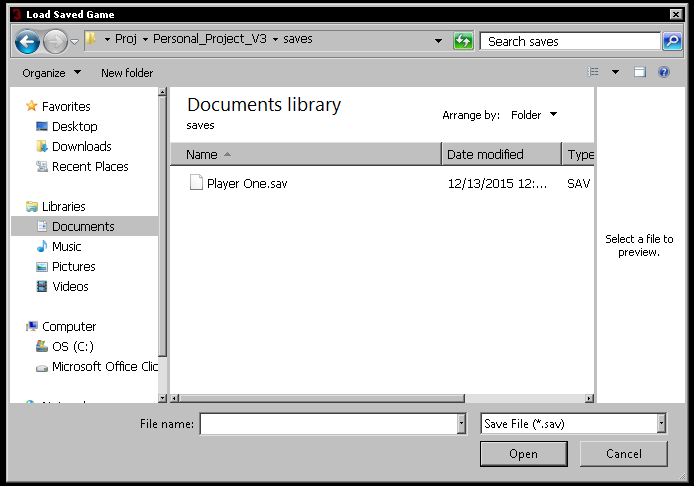
Clicking “File” menu



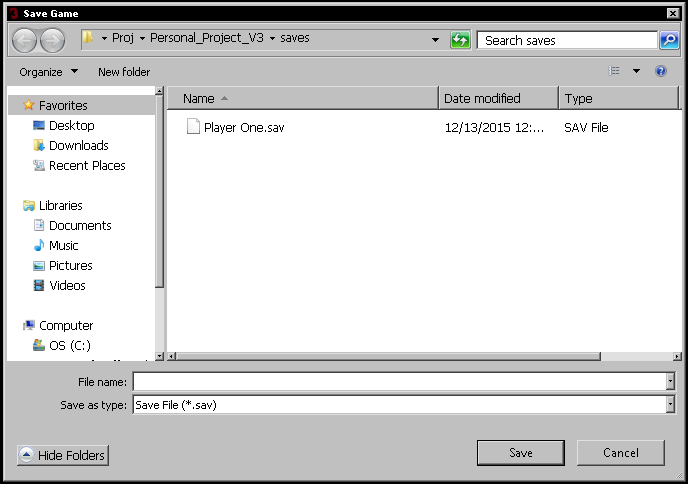
Clicking File>New Game (not functioning as intended)



Clicking File>Load Game



Clicking File>Save Game



Clicking File>Multiplayer (not functioning as intended, qDebug a string)





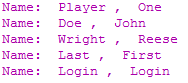
Clicking File>Exit (just terminates program)

Clicking “Help” menu



Clicking Help>Players List (qDebug database table)





Clicking Help>About Game



Flowchart

Pseudo-Code

Major Variables

Language Constructs

**References**

Used <http://doc.qt.io/qt-5/> a lot to help me understand what the functions take in and what the objects had e.g. QGraphicsView has setFixedSize etc.

Used <http://www.qtcentre.org/forum.php> for fixes to Qt errors I was encountering.

Used the book for examples and explanations.

**Program**

CLOCK SERVER

clockgears.h

/\*

\* File: clockgears.h

\* Author: Enrique Najera

\* Purpose: Return time to client

\* 16 December 2015

\*/

#ifndef CLOCKGEARS\_H

#define CLOCKGEARS\_H

// Q\_LIBS

#include <QPushButton>

#include <QUdpSocket>

#include <QTimer>

#include <QDateTime>

// Start class ClockGears

class ClockGears : public QPushButton

{

Q\_OBJECT

public:

ClockGears(QWidget \*parent=0);

int monsterResponse() const; // Send number as monster response

// to player noise

private slots:

void sendDatagram();

private:

QUdpSocket udpSocket;

QTimer timer;

};// End class ClockGears

#endif // CLOCKGEARS\_H

clockgears.cpp

/\*

\* File: clockgears.cpp

\* Author: Enrique Najera

\* Purpose: Return time to client

\* 16 December 2015

\*/

// USER\_LIBS

#include "clockgears.h"

// Start constructor ClockGears

ClockGears::ClockGears(QWidget \*parent) : QPushButton(tr("Stop"), parent)

{

connect(this, SIGNAL(clicked()), this, SLOT(close()));

connect(&timer, SIGNAL(timeout()), this, SLOT(sendDatagram()));

timer.start(1000);

setWindowTitle(tr("Clock"));

}// End constructor ClockGears

// Start method monster response

int ClockGears::monsterResponse() const

{

return 240;

}// End method monsterResponse

// Start method sendDatagram

void ClockGears::sendDatagram()

{

QByteArray datagram;

QDataStream out(&datagram, QIODevice::WriteOnly);

out.setVersion(QDataStream::Qt\_5\_0);

out << QDateTime::currentDateTime() << monsterResponse();

udpSocket.writeDatagram(datagram, QHostAddress::LocalHost, 5824);

}// End method sendDatagram

main.cpp

/\*

\* File: main.cpp

\* Author: Enrique Najera

\* Purpose: Create ClockGears obj

\* 16 December 2015

\*/

// Q\_LIBS

#include <QApplication>

// USER\_LIBS

#include "clockgears.h"

// Start method main

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

{

QApplication a(argc, argv);

ClockGears cGears;

cGears.show();

return a.exec();

}// End method main

RPGtest a.k.a main project

main.cpp

/\*

\* File: main.cpp

\* Author: Enrique Najera

\* Purpose: Create splashscreen, game window,

\* Game object, connections

\* 16 December 2015

\*/

// Q\_LIBS

#include <QApplication>

#include <QSplashScreen>

// USER\_LIBS

#include "database.h"

#include "gamewindow.h"

#include "game.h"

#include "clock.h"

// Start method main

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

{

QApplication a(argc, argv);

// Create splashscreen object

QSplashScreen \*splash = new QSplashScreen;

splash->setPixmap(QPixmap(":/img/images/splash.png"));

splash->show();

// Create database object

Database \*db = new Database();

Qt::Alignment topRight = Qt::AlignRight | Qt::AlignTop;

splash->showMessage(QObject::tr("Establishing connections..."),

topRight, Qt::white);

// Database connection

if(!db->createConnection())

return 1;

// Client connection

Clock clock;

clock.show();

splash->showMessage(QObject::tr("Setting up main window..."),

topRight, Qt::white);

// Create game window

GameWindow gameWindow;

splash->showMessage(QObject::tr("Setting up game..."),

topRight, Qt::white);

// create game

Game \*game = new Game();

gameWindow.setCentralWidget(game);

gameWindow.show();

splash->finish(&gameWindow);

delete splash;

return a.exec();

}// End method main

button.h

/\*

\* File: button.h

\* Author: Enrique Najera

\* Purpose: Create buttons for main menu

\* 16 December 2015

\*/

#ifndef BUTTON\_H

#define BUTTON\_H

// Q\_LIBS

#include <QGraphicsPixmapItem>

// Start class Button

class Button : public QObject, public QGraphicsPixmapItem

{

Q\_OBJECT

public:

Button(int y);

void *mousePressEvent*(QGraphicsSceneMouseEvent \*event);

private:

QPixmap \*sheet\_button;

signals:

void click();

};// End class Button

#endif // BUTTON\_H

button.cpp

/\*

\* File: button.cpp

\* Author: Enrique Najera

\* Purpose: Create buttons for main menu

\* 16 December 2015

\*/

// USER\_LIBS

#include "button.h"

// Start constructor Button

Button::Button(int y)

{

sheet\_button = new QPixmap(":/img/images/buttons.png");

setPixmap(sheet\_button->copy(0,y,64,32));

}// End constructor Button

// Start method mousePressEvent

void Button::*mousePressEvent*(QGraphicsSceneMouseEvent \*event)

{

emit click();

}// End method mousePressEvent

clock.h

/\*

\* File: clock.h

\* Author: Enrique Najera

\* Purpose: Get time from server

\* 16 December 2015

\*/

#ifndef CLOCK\_H

#define CLOCK\_H

// Q\_LIBS

#include <QDialog>

#include <QUdpSocket>

#include <QDateTime>

#include <QLabel>

#include <QLineEdit>

#include <QGridLayout>

#include <QtGui>

#include <QtNetwork>

// Start class Clock

class Clock : public QDialog

{

Q\_OBJECT

public:

Clock(QWidget \*parent=0);

private slots:

void processPendingDatagrams();

private:

QUdpSocket udpSocket;

QLabel \*lbl\_time;

QLineEdit \*line\_time;

QGridLayout \*layout\_main;

};// End class Clock

#endif // CLOCK\_H

clock.cpp

/\*

\* File: clock.cpp

\* Author: Enrique Najera

\* Purpose: Get time from server

\* 16 December 2015

\*/

// USER\_LIBS

#include "clock.h"

// Start constructor Clock

Clock::Clock(QWidget \*parent) : QDialog(parent)

{

udpSocket.bind(5824);

connect(&udpSocket, SIGNAL(readyRead()), this, SLOT(processPendingDatagrams()));

lbl\_time = new QLabel(tr("Time: "));

line\_time = new QLineEdit;

line\_time->setReadOnly(true);

layout\_main = new QGridLayout;

layout\_main->addWidget(lbl\_time, 0, 0);

layout\_main->addWidget(line\_time, 0, 1);

setLayout(layout\_main);

setWindowTitle("Clock");

}// End constructor Clock

// Start method processPendingDatagrams

void Clock::processPendingDatagrams()

{

QByteArray datagram;

do

{

datagram.resize(udpSocket.pendingDatagramSize());

udpSocket.readDatagram(datagram.data(), datagram.size());

}while(udpSocket.hasPendingDatagrams());

QDateTime dateTime;

int monsterResp;

QDataStream in(&datagram, QIODevice::ReadOnly);

in.setVersion(QDataStream::Qt\_5\_0);

in >> monsterResp;

line\_time->setText(tr("%1").arg(monsterResp));

}// End method processPendingDatagrams

database.h

/\*

\* File: database.cpp

\* Author: Enrique Najera

\* Purpose: Connect to database to retrieve names

\* 16 December 2015

\*/

#ifndef DATABASE\_H

#define DATABASE\_H

// Q\_LIBS

#include <QSqlDatabase>

#include <QSqlError>

#include <QMessageBox>

// Start class Database

class Database

{

public:

Database();

bool createConnection();

};// End class Database

#endif // DATABASE\_H

database.cpp

/\*

\* File: database.cpp

\* Author: Enrique Najera

\* Purpose: Connect to database to retrieve names

\* 16 December 2015

\*/

// USER\_LIBS

#include "database.h"

// Start constructor Database

Database::Database()

{

}// End constructor Database

// Start method createConnection

bool Database::createConnection()

{

QSqlDatabase db = QSqlDatabase::addDatabase("QMYSQL");

db.setHostName("localhost");

db.setDatabaseName("db\_room3");

db.setUserName("root");

db.setPassword("");

if (!db.open())

{

QMessageBox::critical(0, QObject::tr("Database Error"),

db.lastError().text());

return false;

}

return true;

}// End method createConnections

game.h

/\*

\* File: game.h

\* Author: Enrique Najera

\* Purpose: Holds main game window & logic

\* 16 December 2015

\*/

#ifndef GAME\_H

#define GAME\_H

// Q\_LIBS

#include <QGraphicsPixmapItem>

#include <QGraphicsScene>

#include <QGraphicsView>

#include <QObject>

#include <QTimer>

#include <QMediaPlayer>

#include <QIcon>

#include <QInputDialog>

#include <QRegExp>

#include <QRegularExpression>

#include <QRegularExpressionMatch>

// USER\_LIBS

#include "gamewindow.h"

#include "button.h"

#include "switchdialog.h"

#include "player.h"

#include "room.h"

#include "things.h"

#include "thread.h"

//#include "collisions.h"

// Start class Game

class Game : public QGraphicsView

{

Q\_OBJECT

public:

Game();

QString name; // Public for I/O

// No mutator/Accessor for speed

private:

int frame; // Holds frame number

bool isGhostExist;

// Create Objects

QGraphicsScene \*scene;

QTimer \*timer;

QInputDialog \*dialog\_form;

QMediaPlayer \*snd\_bkgdMusic;

QMediaPlayer \*snd\_flip;

Button \*btn\_new;

Button \*btn\_cont;

Button \*btn\_quit;

SwitchDialog \*switchDialog;

Player \*player;

Room \*room;

Thread \*td\_name;

Things \*table1\_L;

Things \*table2\_L;

Things \*table1\_R;

Things \*table2\_R;

Things \*flower1\_L;

Things \*flower2\_L;

Things \*flower1\_R;

Things \*flower2\_R;

Things \*window1\_L;

Things \*window2\_L;

Things \*window1\_R;

Things \*window2\_R;

Things \*chair1\_L;

Things \*chair2\_L;

Things \*chair1\_R;

Things \*chair2\_R;

Things \*picture1\_L;

Things \*picture2\_L;

Things \*picture1\_R;

Things \*picture2\_R;

Things \*carpet\_L;

Things \*carpet\_R;

Things \*cupboard1\_L;

Things \*cupboard2\_L;

Things \*cupboard3\_L;

Things \*cupboard4\_L;

Things \*cupboard1\_R;

Things \*cupboard2\_R;

Things \*cupboard3\_R;

Things \*cupboard4\_R;

Things \*couch1\_L;

Things \*couch2\_L;

Things \*couch3\_L;

Things \*couch1\_R;

Things \*couch2\_R;

Things \*couch3\_R;

public slots:

void update(); // Updates graphics

void newGame(); // Creates all game obj

void form(); // Takes in user's info

};// End class Game

#endif // GAME\_H

game.cpp

/\*

\* File: game.cpp

\* Author: Enrique Najera

\* Purpose: Holds main game window & logic

\* 16 December 2015

\*/

// USER\_LIBS

#include "game.h"

// Start constructor Game

Game::Game()

{

// Create scene

scene = new QGraphicsScene();

scene->setSceneRect(0,0,384,384); // 256 x 256

setScene(scene);

setHorizontalScrollBarPolicy(Qt::ScrollBarAlwaysOff);

setVerticalScrollBarPolicy(Qt::ScrollBarAlwaysOff);

setBackgroundBrush(QPixmap(":/img/images/introBKGD.png"));

// Create buttons

btn\_new = new Button(0); // New Game

connect(btn\_new, SIGNAL(click()), this, SLOT(form()));

btn\_cont = new Button(32); // Continue

connect(btn\_cont, SIGNAL(click()), this, SLOT(newGame()));

btn\_quit = new Button(64); // Quit

//connect(btn\_quit, SIGNAL(click()), , SLOT());

// Set toolTips for buttons

btn\_new->setToolTip("Start a new game");

btn\_cont->setToolTip("Continue from last save file");

btn\_quit->setToolTip("Quits application");

// Add buttons

scene->addItem(btn\_new);

scene->addItem(btn\_cont);

scene->addItem(btn\_quit);

// Layout buttons

btn\_new->setPos(50,50);

btn\_cont->setPos(50,100);

btn\_quit->setPos(50,150);

}// End constructor Game

// Start method form

void Game::form()

{

// Declare Variables

bool dispForm = true,

isMatch;

// Must match: [L][astname]{1-19} + 'space' + [F][irstname]{1-19}

QRegularExpression re("^[A-Z][a-z]+(\\s|,)[A-Z][a-z]{1,19}$");

dialog\_form = new QInputDialog();

// Display form while incorrect input

while(dispForm)

{

name = dialog\_form->getText(this, "Enter your name", "Lastname Firstname (2-19 letters) ",

QLineEdit::Normal, "Player One", &isMatch);

QRegularExpressionMatch reMatch = re.match(name);

if (reMatch.hasMatch() == true)

{

//player->setName(name); crashes

dispForm = false;

newGame();

}

}// End displaying form

}// End method form

// Start method newGame

void Game::newGame()

{

// Start name thread

td\_name = new Thread();

td\_name->setMessage(name);

td\_name->start();

bool isGhostExist = true; // Ghost in window

// Remove buttons

scene->removeItem(btn\_new);

scene->removeItem(btn\_cont);

scene->removeItem(btn\_quit);

delete btn\_new;

delete btn\_cont;

delete btn\_quit;

setBackgroundBrush(QBrush(Qt::black));

// Create room

// -Background

room = new Room();

scene->addItem(room);

// -Things

table1\_L = new Things(0, 0);

table1\_R = new Things(0, 0);

table2\_L = new Things(32, 0);

table2\_R = new Things(32, 0);

flower1\_L = new Things(0, 32);

flower1\_R = new Things(0, 32);

flower2\_L = new Things(32, 32);

flower2\_R = new Things(32, 32);

window1\_L = new Things(0, 64);

window1\_R = new Things(0, 64);

window2\_L = new Things(32, 64);

window2\_R = new Things(32, 64);

chair1\_L = new Things(0, 96);

chair2\_L = new Things(0, 96);

chair1\_R = new Things(0, 96);

chair2\_R = new Things(0, 96);

picture1\_L = new Things(0, 128);

picture1\_R = new Things(0, 128);

picture2\_L = new Things(32, 128);

picture2\_R = new Things(32, 128);

carpet\_L = new Things(0, 160);

carpet\_R = new Things(0, 160);

cupboard1\_L = new Things(0, 192);

cupboard1\_R = new Things(0, 192);

cupboard2\_L = new Things(32, 192);

cupboard2\_R = new Things(32, 192);

cupboard3\_L = new Things(64, 192);

cupboard3\_R = new Things(64, 192);

cupboard4\_L = new Things(96, 192);

cupboard4\_R = new Things(96, 192);

couch1\_L = new Things(0, 224);

couch2\_L = new Things(32, 224);

couch3\_L = new Things(64, 224);

couch1\_R = new Things(64, 224);

couch2\_R = new Things(32, 224);

couch3\_R = new Things(0, 224);

// Rotate right couch

couch1\_R->setTransformOriginPoint(couch1\_R->*boundingRect*().center());

couch2\_R->setTransformOriginPoint(couch2\_R->*boundingRect*().center());

couch3\_R->setTransformOriginPoint(couch3\_R->*boundingRect*().center());

couch1\_R->setRotation(180);

couch2\_R->setRotation(180);

couch3\_R->setRotation(180);

// Add all the things

// -left

scene->addItem(table1\_L);

scene->addItem(flower1\_L);

scene->addItem(window1\_L);

scene->addItem(chair1\_L);

scene->addItem(chair2\_L);

scene->addItem(picture1\_L);

scene->addItem(carpet\_L);

scene->addItem(cupboard1\_L);

scene->addItem(couch1\_L);

scene->addItem(couch2\_L);

scene->addItem(couch3\_L);

// -right

scene->addItem(table1\_R);

scene->addItem(flower2\_R);

scene->addItem(window2\_R);

scene->addItem(chair1\_R);

scene->addItem(chair2\_R);

scene->addItem(picture2\_R);

scene->addItem(carpet\_R);

scene->addItem(cupboard1\_R);

scene->addItem(couch1\_R);

scene->addItem(couch2\_R);

scene->addItem(couch3\_R);

// Create player

player = new Player();

player->setFlag(QGraphicsItem::ItemIsFocusable);

player->setFocus();

scene->addItem(player);

// Create dialogs

switchDialog = new SwitchDialog();

// Create and play background music

snd\_bkgdMusic = new QMediaPlayer();

snd\_bkgdMusic->setMedia(QUrl("qrc:/sound/sounds/Music/bkgd.mp3"));

snd\_bkgdMusic->play();

snd\_flip = new QMediaPlayer();

snd\_flip->setMedia(QUrl("qrc:/sound/sounds/SFX/flip.mp3"));

// Timer calls game update

timer = new QTimer();

connect(timer, SIGNAL(timeout()), this, SLOT(update()));

timer->start(10);

// Stop thread name

td\_name->stop();

td\_name->wait();

setFixedSize(384,384);

}// End method newGame

// Start method update

void Game::update()

{

// Hide cursor

setCursor(QCursor(QPixmap(":/img/images/cursor.png")));

// Restart frame every 4 + 4 frames

if (player->frame > 8)

player->frame = 0;

// Moves BKGND image based on player position

room->setPos(player->posX - 127, player->posY - 127);

// Move the things based on room position

// -left

table1\_L->setPos( room->x() + 80, room->y() + 352);

table2\_L->setPos( room->x() + 80, room->y() + 352);

flower1\_L->setPos( room->x() + 144, room->y() + 352);

flower2\_L->setPos( room->x() + 144, room->y() + 352);

window1\_L->setPos( room->x() + 112, room->y() + 0);

window2\_L->setPos( room->x() + 112, room->y() + 0);

chair1\_L->setPos( room->x() + 48, room->y() + 64);

chair2\_L->setPos( room->x() + 112, room->y() + 64);

picture1\_L->setPos( room->x() + 48, room->y() + 0);

picture2\_L->setPos( room->x() + 48, room->y() + 0);

carpet\_L->setPos( room->x() + 80, room->y() + 96);

cupboard1\_L->setPos(room->x() + 144, room->y() + 32);

cupboard2\_L->setPos(room->x() + 144, room->y() + 32);

cupboard3\_L->setPos(room->x() + 144, room->y() + 32);

couch1\_L->setPos( room->x() + 16, room->y() + 256);

couch2\_L->setPos( room->x() + 16, room->y() + 224);

couch3\_L->setPos( room->x() + 16, room->y() + 192);

// -right

table1\_R->setPos( room->x() + 272, room->y() + 352);

table2\_R->setPos( room->x() + 272, room->y() + 352);

flower1\_R->setPos( room->x() + 208, room->y() + 352);

flower2\_R->setPos( room->x() + 208, room->y() + 352);

window1\_R->setPos( room->x() + 240, room->y() + 0);

window2\_R->setPos( room->x() + 240, room->y() + 0);

chair1\_R->setPos( room->x() + 240, room->y() + 64);

chair2\_R->setPos( room->x() + 304, room->y() + 64);

picture1\_R->setPos( room->x() + 304, room->y() + 0);

picture2\_R->setPos( room->x() + 304, room->y() + 0);

carpet\_R->setPos( room->x() + 272, room->y() + 96);

cupboard1\_R->setPos(room->x() + 208, room->y() + 32);

cupboard2\_R->setPos(room->x() + 208, room->y() + 32);

cupboard3\_R->setPos(room->x() + 208, room->y() + 32);

couch1\_R->setPos( room->x() + 336, room->y() + 256);

couch2\_R->setPos( room->x() + 336, room->y() + 224);

couch3\_R->setPos( room->x() + 336, room->y() + 192);

// Ghost in the window

if (player->posX < 53 && player->posY > 128)

{

if (!isGhostExist)

{

scene->removeItem(window2\_R);

scene->removeItem(player);

scene->addItem(window1\_R);

scene->addItem(player);

player->setFocus();

isGhostExist = true;

}

}

// Check interactions

// -cupboard

if (player->pState == 'I' && player->facing == 'C')

{

switchDialog->show();

if (switchDialog->getYesClicked() == true)

{

// play flip sound

if (snd\_flip->state() == QMediaPlayer::PlayingState)

snd\_flip->setPosition(0);

else if (snd\_flip->state() == QMediaPlayer::StoppedState)

snd\_flip->play();

// Change item

scene->removeItem(cupboard1\_L);

scene->removeItem(player);

scene->addItem(cupboard2\_L);

scene->addItem(player);

player->setFocus();

}else{}

}

// -chair

else if (player->pState == 'I' && player->facing == 'c')

{

}

// -couch

else if (player->pState == 'I' && player->facing == 'Q')

{

}

// -window

else if (player->pState == 'I' && player->facing == 'W')

{

}

// -picture

else if (player->pState == 'I' && player->facing == 'P')

{

switchDialog->show();

}

// -table

else if (player->pState == 'I' && player->facing == 'T')

{

switchDialog->show();

if (switchDialog->getYesClicked() == true)

{

// play flip sound

if (snd\_flip->state() == QMediaPlayer::PlayingState)

snd\_flip->setPosition(0);

else if (snd\_flip->state() == QMediaPlayer::StoppedState)

snd\_flip->play();

// Change item

scene->removeItem(table1\_L);

scene->removeItem(player);

scene->addItem(table2\_L);

scene->addItem(player);

player->setFocus();

}else{}

}

// -plant

else if (player->pState == 'I' && player->facing == 'p')

{

switchDialog->show();

}

// end check interactions

// Sprite animation

// -if player is idle

if (player->pState == 'A')

player->setPixmap(player->sheet\_player->copy(0, player->left, 32, 32));

else if (player->pState == 'S')

player->setPixmap(player->sheet\_player->copy(0, player->forward, 32, 32));

else if (player->pState == 'F')

player->setPixmap(player->sheet\_player->copy(0, player->right, 32, 32));

else if (player->pState == 'W')

player->setPixmap(player->sheet\_player->copy(0, player->backward, 32, 32));

// If player is moving

// -frame 1

if (player->frame == 4 && player->pState != 'X')

{

if (player->pState == 'L')

player->setPixmap(player->sheet\_player->copy(32, player->left, 32, 32));

else if (player->pState == 'D')

player->setPixmap(player->sheet\_player->copy(32, player->forward, 32, 32));

else if (player->pState == 'R')

player->setPixmap(player->sheet\_player->copy(32, player->right, 32, 32));

else if (player->pState == 'U')

player->setPixmap(player->sheet\_player->copy(32, player->backward, 32, 32));

}

// -frame 2

else if (player->frame < 4 && player->pState != 'X')

{

if (player->pState == 'L')

player->setPixmap(player->sheet\_player->copy(64, player->left, 32, 32));

else if (player->pState == 'D')

player->setPixmap(player->sheet\_player->copy(64, player->forward, 32, 32));

else if (player->pState == 'R')

player->setPixmap(player->sheet\_player->copy(64, player->right, 32, 32));

else if (player->pState == 'U')

player->setPixmap(player->sheet\_player->copy(64, player->backward, 32, 32));

}// End sprite animation

// Check collisions

player->collisions();

}// End method update

gamewindow.h

/\*

\* File: gamewindow.h

\* Author: Enrique Najera

\* Purpose: Creates menu items/actions

\* 16 December 2015

\*/

#ifndef GAMEWINDOW\_H

#define GAMEWINDOW\_H

// Q\_LIBS

#include <QMainWindow>

#include <QAction>

#include <QLabel>

#include <QMenu>

#include <QMessageBox>

#include <QMenuBar>

#include <QFileDialog>

#include <QSqlQuery>

#include <QSqlError>

// USER\_LIBS

#include "game.h"

// Start class GameWindow

class GameWindow : public QMainWindow

{

Q\_OBJECT

public:

// Constructors

GameWindow();

protected:

void *closeEvent*(QCloseEvent \*event);

private slots:

void newGame();

void loadGame();

bool saveGame();

void multiplayer();

void players();

void about();

private:

void createActions();

void createMenus();

bool loadFile(const QString &fileName); // Takes in file name

bool saveFile(const QString &fileName); // Takes in file name

// Create objects

QMenu \*fileMenu;

QMenu \*helpMenu;

QAction \*newAction;

QAction \*loadAction;

QAction \*saveAction;

QAction \*multiplayerAction;

QAction \*exitAction;

QAction \*playersAction;

QAction \*aboutAction;

};// End class GameWindow

#endif // GAMEWINDOW\_H

gamewindow.cpp

/\*

\* File: gamewindow.cpp

\* Author: Enrique Najera

\* Purpose: Creates menu items/actions

\* 16 December 2015

\*/

// Q\_LIBS

#include <QtGui>

#include <QDebug> // Shows if connect worked --[[DELETE]]

// USER\_LIBS

#include "gamewindow.h"

// Start constructor GameWindow

GameWindow::GameWindow()

{

// Call creation methods

createActions();

createMenus();

setAttribute(Qt::WA\_DeleteOnClose);

// Window properties

setWindowIcon(QIcon(":/img/images/icon.png"));

setWindowTitle(tr("Room 3"));

setFixedSize(200,200); // 200 x 200

setWindowFlags(Qt::WindowTitleHint);

}// End method GameWindow

// Start method createActions creates menu actions

void GameWindow::createActions()

{

newAction = new QAction(tr("&New Game"), this);

newAction->setShortcut(tr("Ctrl+N"));

newAction->setStatusTip(tr("Restarts the game"));

connect(newAction, SIGNAL(triggered()),this,SLOT(newGame()));

loadAction = new QAction(tr("&Load Game"), this);

loadAction->setShortcut(tr("Ctrl+O"));

loadAction->setStatusTip(tr("Loads a saved game"));

connect(loadAction, SIGNAL(triggered()),this,SLOT(loadGame()));

saveAction = new QAction(tr("&Save Game"), this);

saveAction->setShortcut(tr("Ctrl+S"));

saveAction->setStatusTip(tr("Saves current game status"));

connect(saveAction, SIGNAL(triggered()),this,SLOT(saveGame()));

multiplayerAction = new QAction(tr("&Multiplayer"), this);

multiplayerAction->setShortcut(tr("Ctrl+M"));

multiplayerAction->setStatusTip(tr("Play with others on same network"));

connect(multiplayerAction, SIGNAL(triggered()),this,SLOT(multiplayer()));

exitAction = new QAction(tr("&Exit"), this);

exitAction->setShortcut(tr("Ctrl+Q"));

exitAction->setStatusTip("Quits current game");

connect(exitAction, SIGNAL(triggered()), qApp, SLOT(quit()));

playersAction = new QAction(tr("&Players List"), this);

playersAction->setStatusTip(tr("List of all players"));

connect(playersAction, SIGNAL(triggered()),this,SLOT(players()));

aboutAction = new QAction(tr("&About Game"), this);

aboutAction->setStatusTip(tr("About the game"));

connect(aboutAction, SIGNAL(triggered()),this,SLOT(about()));

}// End method createActions

// Start method createMenus

void GameWindow::createMenus()

{

fileMenu = menuBar()->addMenu(tr("&File"));

fileMenu->addAction(newAction);

fileMenu->addAction(loadAction);

fileMenu->addAction(saveAction);

fileMenu->addSeparator();

fileMenu->addAction(multiplayerAction);

fileMenu->addSeparator();

fileMenu->addAction(exitAction);

helpMenu = menuBar()->addMenu(tr("&Help"));

helpMenu->addAction(playersAction);

helpMenu->addAction(aboutAction);

}// End method createMenus

// Start method newGame INIT main window

void GameWindow::newGame()

{

Game \*game = new Game();

game->newGame();

} // End method newGame

// Start mehtod saveGame saves game state

bool GameWindow::saveGame()

{

Game \*game = new Game();

QString fileName = QFileDialog::getSaveFileName(NULL, "Save Game", QString(), "Save File (\*.sav);;All Files (\*)");

QFile file(fileName);

file.*open*(QIODevice::ReadWrite);

QTextStream out(&file);

file.setTextModeEnabled(true);

out << "Name " << game->name << "\n"; // Doesn't output name

out << "Time " << "\n";

file.*close*();

return true;

}// End mehtod saveGame

// Start mehtod loadGame loads saved file

void GameWindow::loadGame()

{

QString fileName = QFileDialog::getOpenFileName(this, "Load Saved Game", QString(), "Save File (\*.sav);;All Files (\*)");

QFile file(fileName);

file.*open*(QIODevice::ReadWrite);

QTextStream in(&file);

file.setTextModeEnabled(true);

QString inText;

newGame();

inText = in.readLine();

inText = in.readLine();

file.*close*();

}// End method saveGame

// Start mehtod mulitplayer

void GameWindow::multiplayer()

{

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* TODO: QTcp \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

qDebug() << "WAITING FOR OTHER PLAYERS...";

}// End method multiplayer

// Start mehtod closeEvent closes window

void GameWindow::*closeEvent*(QCloseEvent \*event)

{

qApp->quit();

}// End method closeEvent

// Start method players list player names

void GameWindow::players()

{

int i = 0;

QVector<QString> first(5);

QSqlQuery query;

query.exec("SELECT \* FROM names");

if(!query.isActive())

QMessageBox::warning(this, tr("Database Error"),

query.lastError().text());

while(query.next())

{

QString dbLastName = query.value(1).toString();

QString dbFirstName = query.value(2).toString();

qDebug() << "Name: " << qPrintable(dbLastName) << ", "

<< qPrintable(dbFirstName);

first.append(dbFirstName);

}

}// End method players

// Start method about

void GameWindow::about()

{

QMessageBox::about(this, tr("About Game"),

tr("<h2>Room 3</h2>"

"<h3>GAME DETAILS</h3>"

" "

"<p>Try to leave the room by solving <font color=#00F><i>puzzles</i></font> before the <font color=#F00>TIME </font>runs out!</p>"

" "

"<h3>GAME CONTROLS</h3>"

" "

"<ul style = 'list-style-image: url('icon.png')'> "

" <li><b>Right arrow key</b>: Move Right </li>"

" <li><b>Left arrow key</b>: Move Left </li>"

" <li><b>Up arrow key</b>: Move Up </li>"

" <li><b>Down arrow key</b>: Move Down </li>"

" <li><b>Z key</b>: Interact </li>"

" <li><b>X key</b>: In-Game Menu </li>"

""

"</ul>"));

}// End method about

player.h

/\*

\* File: player.h

\* Author: Enrique Najera

\* Purpose: Player properties

\* 16 December 2015

\*/

#ifndef PLAYER\_H

#define PLAYER\_H

// Q\_LIBS

#include <QGraphicsPixmapItem>

#include <QKeyEvent>

#include <QGraphicsScene>

#include <QPixmap>

#include <QMediaPlayer>

// USER\_LIBS

#include "room.h"

// Start class Player

class Player : public QGraphicsPixmapItem

{

public:

// Constructor

Player();

// Key events

void *keyPressEvent*(QKeyEvent \*event);

void *keyReleaseEvent*(QKeyEvent \*event);

// Function Prototypes

void collisions(); // Checks collisions with Things

// Mutator and Accessor functions

void setName(QString n){ name = n; }

QString getName() const{ return name; }

int posX;

int posY;

// Sprite position

int left,

forward,

right,

backward,

frame, // For animation

stopSpam; // Stops interaction dialog spamming

char pState; // State of player

char facing; // Holds what thing player is facing

QPixmap \*sheet\_player;

private:

QString name;

QMediaPlayer \*snd\_walk;

};// End class Player

#endif // PLAYER\_H

player.cpp

/\*

\* File: player.cpp

\* Author: Enrique Najera

\* Purpose: Player properties

\* 16 December 2015

\*/

// USER\_LIBS

#include "player.h"

// Start constructor Player

Player::Player()

{

// INIT sprite positions

left = 0;

forward = 32;

right = 64;

backward = 96;

frame = 0;

// Set player image

sheet\_player = new QPixmap(":/img/images/player.png");

setPixmap(sheet\_player->copy(0, forward, 32, 32));

// Position player in middle of screen

setPos(256 / 2, 256/2);

// INIT position

posX = 256 / 2;

posY = 256 / 2;

// Create sounds

snd\_walk = new QMediaPlayer;

snd\_walk->setMedia(QUrl("qrc:/sound/sounds/SFX/footstep.mp3"));

}// End constructor Player

// Start method keyPressEvent

void Player::*keyPressEvent*(QKeyEvent \*event)

{

// Player movement

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

{

// play walk sound

if (snd\_walk->state() == QMediaPlayer::PlayingState)

snd\_walk->setPosition(0);

else if (snd\_walk->state() == QMediaPlayer::StoppedState)

snd\_walk->play();

pState = 'R';

if (posX > -87)

{

frame++;

posX-=5;

}

else

pState = 'F'; // Idle

}

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

{

// play walk sound

if (snd\_walk->state() == QMediaPlayer::PlayingState)

snd\_walk->setPosition(0);

else if (snd\_walk->state() == QMediaPlayer::StoppedState)

snd\_walk->play();

pState = 'L';

if (posX < 243)

{

frame++;

posX+=5;

}

else

pState = 'A'; // Idle

}

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

{

// play walk sound

if (snd\_walk->state() == QMediaPlayer::PlayingState)

snd\_walk->setPosition(0);

else if (snd\_walk->state() == QMediaPlayer::StoppedState)

snd\_walk->play();

pState = 'U';

if (posY < 243)

{

frame++;

posY+=5;

}

else

pState = 'W'; // Idle

}

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

{

// play walk sound

if (snd\_walk->state() == QMediaPlayer::PlayingState)

snd\_walk->setPosition(0);

else if (snd\_walk->state() == QMediaPlayer::StoppedState)

snd\_walk->play();

pState = 'D';

if (posY > -97)

{

frame++;

posY-=5;

}

else

pState = 'S'; // Idle

}

// Player Interaction

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

{

pState = 'I';

}

// In-Game Menu

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

{

pState = 'M';

}

}// End method keyPressEvent

// Start method keyReleaseEvent

void Player::*keyReleaseEvent*(QKeyEvent \*event)

{

// Sets player as idle

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

{

pState = 'F';

}

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

{

pState = 'A';

}

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

{

pState = 'W';

}

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

{

pState = 'S';

}

}// End method keyReleaseEvent

// Start method collisions

void Player::collisions()

{

/\*\*\*\*\*RIGHT\*\*\*\*\*/

// -cupboard

// --facing up

if ((posX == 128 ||

posX == 123 ||

posX == 118 ||

posX == 113 ||

posX == 108 ||

posX == 103) && posY >= 213 && (pState == 'U' || pState == 'W'))

{

posY = 213;

facing = 'C';

}

// --facing right

else if ((posY == 223 ||

posY == 228 ||

posY == 233 ||

posY == 238 ||

posY == 243)&& posX <= 138 && posX >= 43 && (pState == 'R' || pState == 'F'))

{

posX = 138;

facing = 'X';

}

else

facing = 'X';

// end cupboard

// -chair 1

// --facing up

if ((posX == 193 ||

posX == 198 ||

posX == 203 ||

posX == 208 ||

posX == 213 ||

posX == 218)&& posY >= 183 && posY <=188 && (pState == 'U' || pState == 'W'))

{

posY = 183;

facing = 'c';

}

// --facing down

else if ((posX == 198 ||

posX == 203 ||

posX == 208 ||

posX == 213 ||

posX == 218)&& posY <= 208 && posY >=203 && (pState == 'D' || pState == 'S'))

{

posY = 208;

facing = 'X';

}

// --facing right

else if ((posY == 188 ||

posY == 193 ||

posY == 198 ||

posY == 203) && posX <= 223 && posX >= 218 && (pState == 'R' || pState == 'F'))

{

posX = 223;

facing = 'X';

}

// --facing left

else if ((posY == 188 ||

posY == 193 ||

posY == 198 ||

posY == 203) && posX >= 193 && posX <= 198 && (pState == 'L' || pState == 'A'))

{

posX = 193;

facing = 'X';

}

// end chair 1

// -chair 2

// --facing up

if ((posX == 128 ||

posX == 133 ||

posX == 138 ||

posX == 143 ||

posX == 148 ||

posX == 153)&& posY >= 183 && posY <=188 && (pState == 'U' || pState == 'W'))

{

posY = 183;

facing = 'c';

}

// --facing down

else if ((posX == 133 ||

posX == 138 ||

posX == 143 ||

posX == 148 ||

posX == 153)&& posY <= 208 && posY >=203 && (pState == 'D' || pState == 'S'))

{

posY = 208;

facing = 'X';

}

// --facing right

else if ((posY == 188 ||

posY == 193 ||

posY == 198 ||

posY == 203)&& posX <= 158 && posX >= 153 && (pState == 'R' || pState == 'F'))

{

posX = 158;

facing = 'X';

}

// --facing left

else if ((posY == 188 ||

posY == 193 ||

posY == 198 ||

posY == 203)&& posX >= 128 && posX <= 133 && (pState == 'L' || pState == 'A'))

{

posX = 128;

facing = 'X';

}

// end chair 2

// -couch

// --facing up

if ((posX == 228 ||

posX == 233 ||

posX == 238 ||

posX == 243)&& posY >= -2 && posY <= 3 && (pState == 'U' || pState == 'W'))

{

posY = -2;

facing = 'X';

}

// --facing down

else if ((posX == 228 ||

posX == 233 ||

posX == 238 ||

posX == 243)&& posY <= 88 && posY >= 83 && (pState == 'D' || pState == 'S'))

{

posY = 88;

facing = 'X';

}

// --facing left

else if ((posY == 3 ||

posY == 8 ||

posY == 13 ||

posY == 18 ||

posY == 23 ||

posY == 28 ||

posY == 33 ||

posY == 38 ||

posY == 43 ||

posY == 48 ||

posY == 53 ||

posY == 58 ||

posY == 63 ||

posY == 68 ||

posY == 73 ||

posY == 78 ||

posY == 83)&& posX >= 223 && posX <= 228 && (pState == 'L' || pState == 'A'))

{

posX = 223;

facing = 'Q';

}

// end couch

// -table

// --facing down

if ((posX == 153 ||

posX == 158 ||

posX == 163 ||

posX == 168 ||

posX == 173 ||

posX == 178 ||

posX == 183 ||

posX == 188 ||

posX == 193)&& posY <= -82 && posY >= -87 && (pState == 'D' || pState == 'S'))

{

posY = -82;

facing = 'T';

}

// --facing left

else if ((posY == -87 ||

posY == -92 ||

posY == -97)&& posX >= 148 && posX <= 153 && (pState == 'L' || pState == 'A'))

{

posX = 148;

facing = 'X';

}

// --facing right

else if ((posY == -87 ||

posY == -92 ||

posY == -97)&& posX <= 198 && posX >= 193 && (pState == 'R' || pState == 'F'))

{

posX = 198;

facing = 'X';

}

// end table

// -plant

// --facing right

if ((posY == -72 ||

posY == -77 ||

posY == -82 ||

posY == -87 ||

posY == -92 ||

posY == -97)&& posX <= 133 && posX >= 128 && (pState == 'R' || pState == 'F'))

{

posX = 133;

facing = 'X';

}

// --facing down

else if ((posX == 103 ||

posX == 108 ||

posX == 113 ||

posX == 118 ||

posX == 123 ||

posX == 128)&& posY <= -67 && posY >= -72 && (pState == 'D' || pState == 'S'))

{

posY = -67;

facing = 'p';

}

// end plant

}// End method collisions

room.h

/\*

\* File: room.h

\* Author: Enrique Najera

\* Purpose: Room properties

\* 16 December 2015

\*/

#ifndef ROOM\_H

#define ROOM\_H

// Q\_LIBS

#include <QGraphicsPixmapItem>

#include <QKeyEvent>

// Start class Room

class Room : public QGraphicsPixmapItem

{

public:

Room();

};// End class Room

#endif // ROOM\_H

room.cpp

/\*

\* File: room.cpp

\* Author: Enrique Najera

\* Purpose: Room properties

\* 16 December 2015

\*/

// USER\_LIBS

#include "room.h"

// Start constructor Room

Room::Room()

{

setPixmap(QPixmap(":/img/images/bkgd.png"));

}// End constructor Room

switchdialog.h

/\*

\* File: switchdialog.h

\* Author: Enrique Najera

\* Purpose: Prompts if player wants to flip object

\* 16 December 2015

\*/

#ifndef SWITCHDIALOG\_H

#define SWITCHDIALOG\_H

// Q\_LIBS

#include <QDialog>

#include <QLabel>

#include <QPushButton>

#include <QHBoxLayout>

#include <QVBoxLayout>

// Start class SwitchDialog

class SwitchDialog : public QDialog

{

Q\_OBJECT

public:

SwitchDialog(QWidget \*parent=0);

void setYesClicked(bool c);

bool getYesClicked() const;

private:

bool isClicked;

QLabel \*label;

QPushButton \*btn\_yes;

QPushButton \*btn\_no;

QHBoxLayout \*layout\_text;

QVBoxLayout \*layout\_btn;

QHBoxLayout \*layout\_main;

private slots:

void yesClicked();

};// End class SwitchDialog

#endif // SWITCHDIALOG\_H

switchdialog.cpp

/\*

\* File: switchdialog.cpp

\* Author: Enrique Najera

\* Purpose: Prompts if player wants to flip object

\* 16 December 2015

\*/

// USER\_LIBS

#include "switchdialog.h"

// Start constructor SwitchDialog

SwitchDialog::SwitchDialog(QWidget \*parent)

{

// INIT vars

isClicked = false;

// Create buttons

label = new QLabel(tr("Change/flip object? "));

btn\_yes = new QPushButton(tr("Yes"));

btn\_no = new QPushButton(tr("No"));

btn\_no->setDefault(true);

// If clicked connections

connect(btn\_yes, SIGNAL(clicked()), this, SLOT(yesClicked()));

connect(btn\_no, SIGNAL(clicked()), this, SLOT(close()));

// Layout

layout\_text = new QHBoxLayout();

layout\_text->addWidget(label);

layout\_btn = new QVBoxLayout();

layout\_btn->addWidget(btn\_yes);

layout\_btn->addWidget(btn\_no);

layout\_btn->addStretch();

layout\_main = new QHBoxLayout();

layout\_main->addLayout(layout\_text);

layout\_main->addLayout(layout\_btn);

setLayout(layout\_main);

setWindowTitle("Change?");

setFixedHeight(*sizeHint*().height());

}// End constructor SwitchDialog

// Start method yesClicked

void SwitchDialog::yesClicked()

{

setYesClicked(true);

close();

}// End method yesClicked

// Start method setYesClicked

void SwitchDialog::setYesClicked(bool c)

{

isClicked = c;

}// End method setYesClicked

// Start method getYesClicked

bool SwitchDialog::getYesClicked() const

{

return isClicked;

}// End method getYesClicked

things.h

/\*

\* File: things.h

\* Author: Enrique Najera

\* Purpose: Room objects

\* 16 December 2015

\*/

#ifndef THINGS\_H

#define THINGS\_H

// Q\_LIBS

#include <QGraphicsPixmapItem>

// Start class Things

class Things : public QGraphicsPixmapItem

{

//Q\_OBJECT

public:

// (image's row, actual image)

Things(int row, int thing);

QPixmap \*sheet\_things;

};// End class Things

#endif // ITEMS\_H

things.cpp

/\*

\* File: things.cpp

\* Author: Enrique Najera

\* Purpose: Room objects

\* 16 December 2015

\*/

// USER\_LIBS

#include "things.h"

// Start constructor Things

Things::Things(int row, int thing)

{

// Set furniture image

sheet\_things = new QPixmap(":/img/images/things.png");

// Create item accordingly

// -table 0

// -flower 32

// -window 64

// -chair 96

// -picture 128

// -carpet 160

// -cupboard 192

// -couch 224

setPixmap(sheet\_things->copy(row, thing, 32, 32));

}// End constructor Things

thread.h

/\*

\* File: thread.h

\* Author: Enrique Najera

\* Purpose: Thread to print DB names

\* 16 December 2015

\*/

#ifndef THREAD\_H

#define THREAD\_H

#include <QThread>

#include <QDebug>

class Thread : public QThread

{

Q\_OBJECT

public:

Thread();

void setMessage(const QString &message)

{ messageStr = message; }

void stop();

protected:

void *run*();

private:

QString messageStr;

volatile bool stopped;

};

#endif // THREAD\_H

thread.cpp

/\*

\* File: thread.cpp

\* Author: Enrique Najera

\* Purpose: Thread to print DB names

\* 16 December 2015

\*/

// USER\_LIBS

#include "thread.h"

// Start constructor Thread

Thread::Thread()

{

stopped = false;

}// End constructor Thread

// Start method stop

void Thread::stop()

{

stopped = true;

}// End method stop

// Start method run

void Thread::*run*()

{

while (!stopped)

qDebug() << qPrintable(messageStr);

stopped = false;

}// End method run