Министерство образования Республики Беларусь
Учреждение Образования
«Брестский Государственный Технический Университет»
Кафедра ИИТ

Лабораторная работа №3
По дисциплине ОСиСП за 5 семестр
Тема: Разработка приложения: «Игра Lines»

Выполнил:
Студент 3-го курса
Группы ПО-5
Крощук В.В.
Проверила:
Дряпко А.В.

Брест 2021

Лабораторная работа №3
Вариант 11

Цель работы:
Научиться разрабатывать и использовать динамические библиотеки (DLL) с использованием Qt.

Задания и выполненные решения:

Доработать программу, разработанную в лабораторной работе No1-2, внеся следующие изменения:

1) Выбрать 3 вспомогательные функции и вынести их описание и реализацию в динамическую библиотеку helper.dll. В основном приложении осуществить загрузку реализованных функций во время работы программы (at run-time, с использованием объекта QLibrary) и их вызов.

2) Выбрать вспомогательный класс и вынести его описание и реализацию в динамическую библиотеку helper_class.dll. В основном приложении осуществить загрузку реализованного класса во время компиляциии (at compile-time).

3) Реализовать окно «О программе» в виде объекта динамической библиотеки about.dll с указанием автора программы, группы, курса и краткого описания разработанного приложения.
Осуществить импорт указанной библиотеки и отображение соответствующего окна при выборе пункта меню «О программе».

4) Реализовать расширения для приложения, позволяющие изменять оформление пунктов меню (шрифт, размер, начертание и т.д.). Соответствующие изменения должны происходить при выборе специального пункта меню. Создать как минимум три расширения такого типа.

Код программы:

main.cpp:
```
#include "mainwindow.h"
#include "Definitions.h"
#include <QtCore>
#include <QApplication>
#include <iostream>
#include "GameLogic.h"

using namespace std;

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.startScreen();
    return a.exec();
}
```

mainwindow.cpp:
```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "about.h"
```

```
#include "helper.h"

typedef QString (*Helper_set_name_window)();
typedef QString (*Helper_set_first_name)();
typedef QString (*Helper_set_second_name)();

QImage createImageWithOverlay(const QImage& baseImage, const QImage& overlayImage);

MainWindow::MainWindow(QWidget *parent): QMainWindow(parent), ui(new
Ui::MainWindow)
{
    ui->setupUi(this);

    // установка имени окна приложения
    QLibrary *helper_set = new QLibrary("Helper");
    Helper_set_name_window set_window_name_ = (Helper_set_name_window)helper_set-
>resolve("set_window_name");
    QString windowTitle = set_window_name_();
    delete helper_set;
    this->setWindowTitle(windowTitle);

    // установка имени текста 'играть'
    QLibrary *helper_set_f_n = new QLibrary("Helper");
    Helper_set_first_name set_window_name_first =
(Helper_set_first_name)helper_set_f_n->resolve("set_first_name_HeaderLabels");
    set_name_first = set_window_name_first();
    delete helper_set_f_n;


    // установка имени текста 'выход'
    QLibrary *helper_set_s_n = new QLibrary("Helper");
    Helper_set_second_name set_window_name_second =
(Helper_set_second_name)helper_set_s_n->resolve("set_second_name_HeaderLabels");
    set_name_second = set_window_name_second();
    delete helper_set_s_n;

    QPixmap bkgnd(":/Assets/background.png");
    QIcon icon(":/Assets/ghost_green.png");
    this->setWindowIcon(icon);
    QPalette palette;
    palette.setBrush(QPalette::Background, bkgnd);
    this->setPalette(palette);
    this->setFixedHeight(sWidth + 100);
    this->setFixedWidth(sWidth);
}

void MainWindow::closeEvent(QCloseEvent *event) {
    QMessageBox::StandardButton resBtn = QMessageBox::question( this, "Выход?",
                                                                tr("Вы уверены, что
хотите выйти?"),
                                                                QMessageBox::No |
QMessageBox::Yes,
                                                                QMessageBox::No);
    if (resBtn != QMessageBox::Yes) {
        event->ignore();
    } else {
        event->accept();
    }
}

void MainWindow::aboutButtonWasClicked(bool) {
    info();
}

void MainWindow::playButtonWasClicked(bool) {
    startGame();
}
```

```
void MainWindow::settingsButtonWasClicked(bool) {
    QMessageBox::StandardButton reply;
    reply = QMessageBox::question(this, "Выход?", "Вы уверены, что хотите выйти?",
                                  QMessageBox::Yes|QMessageBox::No,
QMessageBox::No);
    if (reply == QMessageBox::Yes) {
        qDebug() << "Yes was clicked";
        QApplication::exit();
    } else {
        qDebug() << "Yes was *not* clicked";
    }

}

void MainWindow::homeButtonWasClicked(bool) {
    QMessageBox::StandardButton reply;
    reply = QMessageBox::question(this, "Назад", "Вы покинете игру. Уверены?",
                                  QMessageBox::Yes|QMessageBox::No);
    if (reply == QMessageBox::Yes) {
        qDebug() << "Yes was clicked";
        startScreen();
    } else {
        qDebug() << "Yes was *not* clicked";
    }
}

void MainWindow::buttonWasPressed(QWidget* buttonW) {
    QPushButton* button = (QPushButton*)(buttonW);
    button->setIconSize(QSize(button->iconSize().width() / 2, button-
>iconSize().height() / 2));
}

void MainWindow::buttonWasReleased(QWidget* buttonW) {
    QPushButton* button = (QPushButton*)(buttonW);
    button->setIconSize(QSize(button->iconSize().width() * 2, button-
>iconSize().height() * 2));
}

void MainWindow::startScreen() {
    GameLogic::m_pInstance = NULL;
    GameLogic::window = this;

    QFont newFont("Ink Free", 14, QFont::Bold, true);

    QWidget *parent = new QWidget();
    parent->resize(sWidth, sWidth);

    list = new QComboBox(parent);
    list->setFont(newFont);
    list->setStyleSheet("border: 0px;");
    setPlugins();

    QPushButton* playGame = new QPushButton(QIcon("://Assets/button_play.png"),
set_name_first, parent);
    playGame->move(100, 110);
    playGame->setStyleSheet("border: 0px; color: rgb(255, 255, 255)");
    playGame->setIconSize(QSize(200, 200));
    connect(playGame, SIGNAL(clicked(bool)), this,
SLOT(playButtonWasClicked(bool)));


    QPushButton* load = new QPushButton("Обновить", parent);
    load->move(380, 110);
    load->setStyleSheet("border: 0px; color: rgb(255, 255, 255)");
    connect(load, SIGNAL(clicked(bool)), this, SLOT(playButtonWasClicked(bool)));


    QPushButton* About = new QPushButton("О программе", parent);
```

```
    About->move(420, 310);
    About->setStyleSheet("border: 0px; color: rgb(255, 255, 255)");
    connect(About, SIGNAL(clicked(bool)), this, SLOT(aboutButtonWasClicked(bool)));


    QPushButton* settings = new QPushButton(QIcon("://Assets/button_home.png"),
set_name_second, parent);
    settings->move(200, 400);
    settings->setStyleSheet("border: 0px; color: rgb(255, 255, 255)");
    settings->setIconSize(QSize(200, 200));
    connect(settings, SIGNAL(clicked(bool)), this,
SLOT(settingsButtonWasClicked(bool)));

    this->setCentralWidget(parent);
    this->show();
}

void MainWindow::startGame() {
    board = new QWidget();
    cells = new Cell* *[boardRow];
    for(int i = 0; i < boardRow; i++) {
        cells[i] = new Cell*[boardColumn];
    }
    QGridLayout* layout = new QGridLayout();

    this->setCentralWidget(board);
    board->setLayout(layout);
    for(int i = 0; i < boardRow; i++)
        for(int j = 0; j < boardColumn; j++){
            Cell* temp = new Cell();
            temp->place = MatrixPoint(i,j);
            QImage cell;
            (i + j) % 2 == 0 ? cell = QImage("://Assets/cell_light.png") : cell =
QImage("://Assets/cell_dark.png");
            temp->setIcon(QIcon(QPixmap::fromImage(cell)));
            temp->setStyleSheet("border: 0px");
            temp->setIconSize(QSize(cellSize, cellSize));


            connect(temp, SIGNAL(wasPressed(MatrixPoint)), GameLogic::Instance(),
SLOT(cellWasPressed(MatrixPoint)));
            connect(temp, SIGNAL(clicked(bool)), temp, SLOT(idiotClick(bool)));

            temp->setMinimumSize(cellSize, cellSize);
            layout->addWidget(temp, i, j);
            cells[i][j] = temp;
        }

    int id = QFontDatabase::addApplicationFont("://Assets/ConcertOne-Regular.ttf");
    QString family = QFontDatabase::applicationFontFamilies(id).at(0);
    QFont monospace(family);
    monospace.setPointSize(30);

    QPushButton* score = new QPushButton(QIcon("://Assets/icon_path.png"), "0",
this);
    score->setFont(monospace);
    score->setStyleSheet("border: 0px; color: rgb(255, 255, 255)");
    score->setIconSize(QSize(100, 100));

    layout->addWidget(score, boardRow + 1, 0, 2, 5);
    this->score = score;

    QPushButton* home = new QPushButton(QIcon("://Assets/button_home.png"), "",
this);
    home ->setFont(monospace);
    home->setStyleSheet("border: 0px; color: rgb(255, 255, 255)");
    home->setIconSize(QSize(100, 100));
```

```
    connect(home, SIGNAL(clicked(bool)), this, SLOT(homeButtonWasClicked(bool)));

    layout->addWidget(home, boardRow + 1, 5, 2, 2);



    GameLogic::Instance()->generateGhosts();

    this->show();
}

void MainWindow::info()
{
    typedef void (*About)();
    QLibrary *aboutLib = new QLibrary("About");
    About showWindowAboutProgramm = (About)aboutLib->resolve("about");
    showWindowAboutProgramm();
    delete aboutLib;
}

void MainWindow::setPlugins()
{
    QDir dir(qApp->applicationDirPath());
    dir.cd("plugins");
    QStringList filter;
    filter << "*.dll";
    QPluginLoader loader;
    foreach(QFileInfo info, dir.entryInfoList(filter)) {
        loader.setFileName(info.absoluteFilePath());
        Interface* mode = qobject_cast<Interface*>(loader.instance());
        if(mode) {
            QString name =
loader.metaData().value("MetaData").toObject().value("Mode_name").toString();
            list->addItem(name);
            mPlugin.append(mode);
        }
    }
    connect(list, SIGNAL(activated(int)), this, SLOT(applyPlugin(int)));
}

void MainWindow::ghostWasMoved(std::vector<Node> road, Ghosts type) {
    QSequentialAnimationGroup* animationManager = new QSequentialAnimationGroup();
    connect(animationManager, SIGNAL(finished()), this, SLOT(finishedAnimating()));
    lastCellPlace = new MatrixPoint(road.back().y, road.back().x);
    lastCellType = type;
    QImage cell;
    (road[0].y + road[0].x) % 2 == 0 ? cell = QImage("://Assets/cell_light.png") :
cell = QImage("://Assets/cell_dark.png");
    cells[road[0].y][road[0].x]->setIcon(QIcon(QPixmap::fromImage(cell)));
    for(unsigned int i = 1; i < road.size(); i++) {
        Node nextCell = road[i];
        Cell* current = cells[nextCell.y][nextCell.x];
        QPropertyAnimation *animation = new QPropertyAnimation(current,
"iconSize");
        animation->setDuration(50);
        animation->setStartValue(QSize(0, 0));
        animation->setEndValue(QSize(cellSize, cellSize));

        animationManager->addAnimation(animation);
    }
    animationManager->start();
}

void MainWindow::finishedAnimating() {
    if(lastCellPlace != NULL) {
        cells[lastCellPlace->row][lastCellPlace->column]-
>setIcon(mergedIcon(lastCellType, *lastCellPlace));
        lastCellPlace = NULL;
```

```
        GameLogic::Instance()->nextMove();
    }
}

QIcon MainWindow::mergedIcon(Ghosts type, MatrixPoint place, bool select) {
    QImage icon;
    switch (type) {
    case yellow:
        icon = QImage("://Assets/ghost_yellow.png");
        break;
    case white:
        icon = QImage("://Assets/ghost_white.png");
        break;
    case green:
        icon = QImage("://Assets/ghost_green.png");
        break;
    case red:
        icon = QImage("://Assets/ghost_red.png");
        break;
    }
    QImage cellIcon;
    if(select)
        cellIcon = (place.row + place.column) % 2 == 0 ?
QImage("://Assets/cell_light.png") : QImage("://Assets/cell_dark.png");
    else
        cellIcon = QImage("://Assets/cell_selected.png");
    QImage merged = createImageWithOverlay(cellIcon, icon);
    return QIcon(QPixmap::fromImage(merged));

}

void MainWindow::ghostWasGenerated(Ghosts type, MatrixPoint place){
    Cell* current = cells[place.row][place.column];

    current->setIcon(mergedIcon(type, place));

    QPropertyAnimation *animation = new
QPropertyAnimation(cells[place.row][place.column], "iconSize");
    animation->setDuration(100);
    animation->setStartValue(QSize(0, 0));
    animation->setEndValue(QSize(cellSize, cellSize));
    animation->start();
}

void MainWindow::ghostWasDeleted(MatrixPoint place) {
    Cell* temp = cells[place.row][place.column];
    if((place.row + place.column) % 2 == 0)
        temp->setIcon(QIcon("://Assets/cell_light.png"));
    else
        temp->setIcon(QIcon("://Assets/cell_dark.png"));
    temp->setIconSize(QSize(cellSize, cellSize));
}

void MainWindow::ghostWasSelected(Ghosts type, MatrixPoint place) {
    if(GameLogic::Instance()->gameBoard[place.row][place.column])
        cells[place.row][place.column]->setIcon(mergedIcon(type, place, false));
}

void MainWindow::ghostWasDeselected(Ghosts type, MatrixPoint place) {
    if(GameLogic::Instance()->gameBoard[place.row][place.column])
        cells[place.row][place.column]->setIcon(mergedIcon(type, place, true));
}

void MainWindow::gameOver() {
    GameLogic::m_pInstance  = NULL;
    QMessageBox::StandardButton reply;
    reply = QMessageBox::question(this, "Game Over", "You have lost. Do you want to
try again?",
```

```
                                    QMessageBox::Yes|QMessageBox::No);
    if (reply == QMessageBox::Yes) {
        startGame();
    } else {
        startScreen();
    }
}

QImage createImageWithOverlay(const QImage& baseImage, const QImage& overlayImage)
{
    QImage imageWithOverlay = QImage(overlayImage.size(),
QImage::Format_ARGB32_Premultiplied);
    QPainter painter(&imageWithOverlay);

    painter.setCompositionMode(QPainter::CompositionMode_Source);
    painter.fillRect(imageWithOverlay.rect(), Qt::transparent);

    painter.setCompositionMode(QPainter::CompositionMode_SourceOver);
    painter.drawImage(0, 0, baseImage);

    painter.setCompositionMode(QPainter::CompositionMode_SourceOver);
    painter.drawImage(0, 0, overlayImage);

    painter.end();

    return imageWithOverlay;
}

MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::applyPlugin(int index)
{
    QFont palette = this->mPlugin[index]->changeView();
    QApplication::setFont(palette);
    startScreen();
}
```

mainwindow.h:

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QComboBox>
#include "GameLogic.h"
#include <QGridLayout>
#include "cell.h"
#include "Definitions.h"
#include <QPainter>
#include <QAnimationDriver>
#include <QAnimationGroup>
#include <QGraphicsOpacityEffect>
#include <QButtonGroup>
#include <QFontDatabase>
#include <QMessageBox>
#include <QCloseEvent>
#include <QMainWindow>
#include <QPluginLoader>
#include <QLibrary>
#include <QFileDialog>
#include "interface.h"

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE
```

```
class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    QPushButton* score;
    QString set_name_second;
    QString set_name_first;
    void gameOver();
    void startScreen();
    void startGame();
    void info();
    void setPlugins();
    void nextMove();
    QIcon mergedIcon(Ghosts type, MatrixPoint place, bool select = true);
    ~MainWindow();

public slots:
    void applyPlugin(int index);
    void ghostWasSelected(Ghosts type, MatrixPoint place);
    void ghostWasDeselected(Ghosts type, MatrixPoint place);
    void ghostWasGenerated(Ghosts type, MatrixPoint place);
    void ghostWasMoved(std::vector<Node> road, Ghosts type);
    void ghostWasDeleted(MatrixPoint place);
    void finishedAnimating();

    void playButtonWasClicked(bool zrtik);
    void aboutButtonWasClicked(bool zrtik);
    void settingsButtonWasClicked(bool zrtik);
    void homeButtonWasClicked(bool zrtik);

    void buttonWasPressed(QWidget* button);
    void buttonWasReleased(QWidget* button);

signals:
    wasPressed(QWidget* button);
    wasReleased(QPushButton* button);

private:
    QWidget* board = new QWidget();
    Cell* **cells;
    Ghosts lastCellType;
    MatrixPoint* lastCellPlace;
    void closeEvent(QCloseEvent *bar);
    Ui::MainWindow *ui;
    QComboBox *list;
    QComboBox *list_theme;
    QVector <Interface*> mPlugin;
    QVector <Interface*> mThemes;
};
#endif // MAINWINDOW_H
```

helper.cpp:

```
#include "helper.h"

QString set_window_name() {
    return QString("Line Balls");
}

QString set_first_name_HeaderLabels() {
    return QString("Game");
}

QString set_second_name_HeaderLabels() {
    return QString("Quit the game");
}
```

helper.h:

```
#ifndef HELPER_H
#define HELPER_H

#include "helper_global.h"
#include <QString>

extern "C" HELPER_EXPORT QString set_window_name();
extern "C" HELPER_EXPORT QString set_first_name_HeaderLabels();
extern "C" HELPER_EXPORT QString set_second_name_HeaderLabels();

#endif // HELPER_H
```

helper_global.h:

```
#ifndef HELPER_GLOBAL_H
#define HELPER_GLOBAL_H

#include <QtCore/qglobal.h>

#if defined(HELPER_LIBRARY)
#  define HELPER_EXPORT Q_DECL_EXPORT
#else
#  define HELPER_EXPORT Q_DECL_IMPORT
#endif

#endif // HELPER_GLOBAL_H
```

about.cpp:

```
#include "about.h"
#include <QMessageBox>

void about()
{
    QMessageBox msgBox;
    msgBox.setText("Программу сделала студентка группы ПО-5 Нерода
Александра\nПрограмма является MP3 плеером");
    msgBox.exec();
}
```

about_global.h:

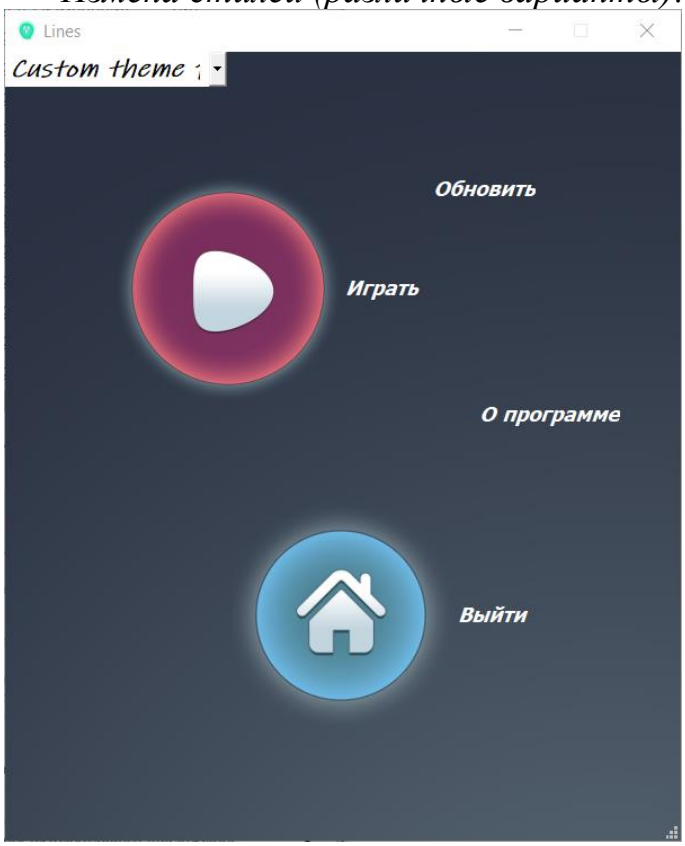
```
#ifndef ABOUT_H
#define ABOUT_H

#include "about_global.h"
extern "C" ABOUT_EXPORT void about();
#endif // ABOUT_H
```

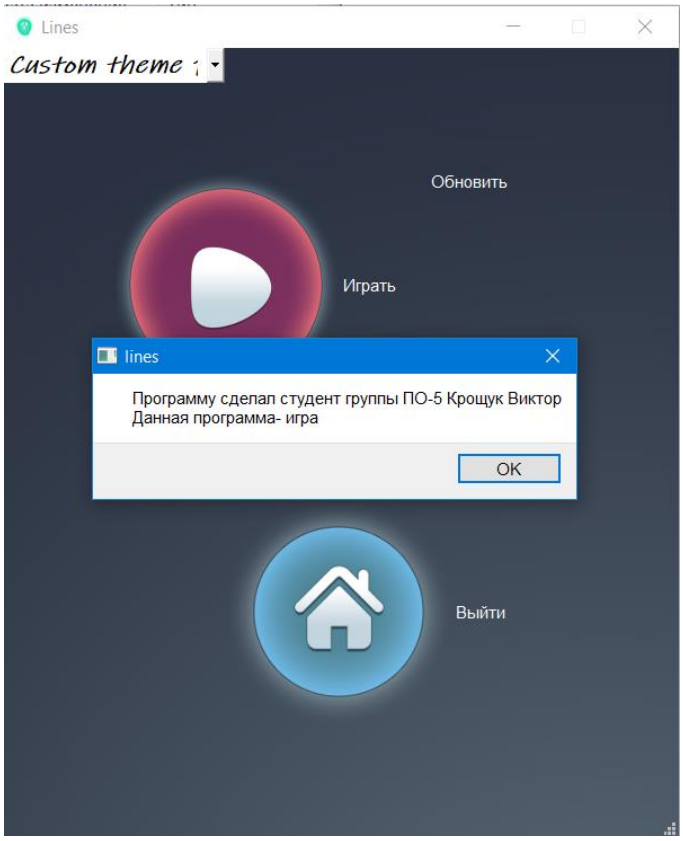
about_global.h:

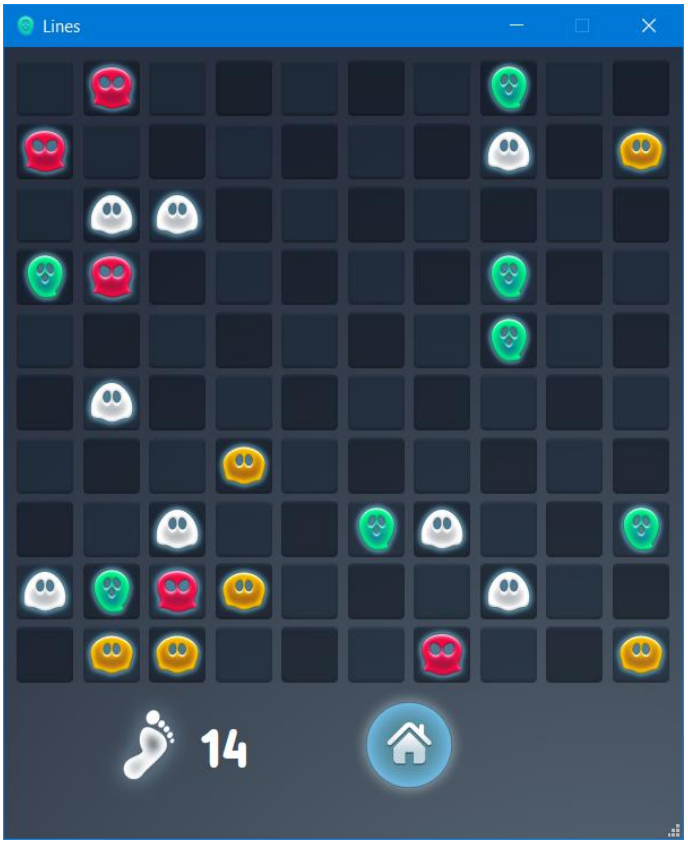
```
#ifndef ABOUT_GLOBAL_H
#define ABOUT_GLOBAL_H

#include <QtCore/qglobal.h>

#if defined(ABOUT_LIBRARY)
#  define ABOUT_EXPORT Q_DECL_EXPORT
#else
#  define ABOUT_EXPORT Q_DECL_IMPORT
#endif

#endif // ABOUT_GLOBAL_H
```

Работа программы:

Измена стилей (различные варианты):







Вывод: научился разрабатывать и использовать динамические библиотеки (DLL) с использованием Qt.