|  |  |  |
| --- | --- | --- |
| **МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ**  **РОССИЙСКОЙ ФЕДЕРАЦИИ** | | |
| ФГБОУ ВПО  **«БРЯНСКИЙ ГОСУДАРСТВЕННЫЙ**  **ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»** | | |
| **Кафедра «**Информатика и программное обеспечение**»** | | |
|  | |  |
| **КУРСОВАЯ РАБОТА** | | |
| РАЗРАБОТКА ИГРОВОЙ ПРОГРАММЫ «КРАСНЫЙ КВАДРАТ» | | |
| Вариант № 5 | | |
| Всего листов | | |
|  | Выполнил студент гр. З16-ИВТ | |
|  | Мамаков А.В.  зачетная книжка № 16.0649 | |
|  | «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2017 г. | |
|  | Руководитель | |
|  | доц. Булатицкий Д.И. | |
|  | «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2017 г. | |
| Брянск 2017 | | |

ЗАДАНИЕ

Разработка игровой программы «Красный квадрат» с использованием методологии объектно-ориентированного программирования.

Аннотация

На основании здания курсовой работы было разработано игровое приложение «Красный квадрат». В аналитической части были определены принципы игрового процесса, представления объектов игры и их взаимодействия.

Был произведен выбор среды разработки приложения, мультимедийной библиотеки для разработки графической части игры и интерфейса пользователя. В процессе разработки была изучена литература и документация по выбранным средствам разработки. Построены модели предметной области и программной системы, разработана архитектура программы и проведено кодирование на языке С++.

В конструкторской части было приведено описание примененных средств разработки, структур данных, использованных для реализации игровых объектов и процессов, а также краткое описание программных модулей.

В соответствующем разделе было описано руководство пользователя.

В результате разработки было создано устойчиво функционирующее игровое приложение в соответствии с заданием.

# Содержание

[Содержание 4](#_Toc501375716)

[Введение 5](#_Toc501375717)

[1. АНАЛИТИЧЕСКАЯ ЧАСТЬ 6](#_Toc501375718)

[1.1. Описание игрового процесса 6](#_Toc501375719)

[1.2. Представление и отображение объектов игры 6](#_Toc501375720)

[1.3. Взаимодействие игровых объектов 6](#_Toc501375721)

[2. КОНСТРУКТОРСКАЯ ЧАСТЬ 7](#_Toc501375722)

[2.1. Описание средств и инструментов разработки 7](#_Toc501375723)

[2.2. Описание объектной модели программной системы 8](#_Toc501375724)

[2.3. Описание модулей приложения 12](#_Toc501375725)

[3. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 14](#_Toc501375726)

[3.1. Комплект поставки и системные требования 14](#_Toc501375727)

[3.2. Установка и удаление программы 14](#_Toc501375728)

[3.3. Запуск программы 14](#_Toc501375729)

[3.4. Описание меню приложения 14](#_Toc501375730)

[3.5. Управление во время игры 16](#_Toc501375731)

[3.6. Выход из игры 16](#_Toc501375732)

[ЗАКЛЮЧЕНИЕ 17](#_Toc501375733)

[Список литературы 18](#_Toc501375734)

# Введение

Игра «Красный квадрат» представляет собой игру – тренажер реакции, в которой пользователь, управляя красным квадратом, должен избегать столкновения с синими квадратами разного размера, движущимися в игровом поле хаотично и с скорением.

Целью данной курсовой работы было изучение средств языка программирования C++, графической библиотеки Simple and Fast Multimedia Library (SFML 2.4) и среды разработки приложений Microsoft Visual Studio в процессе создания игрового приложения в соответствии с заданием.

В задачи курсовой работы входило изучение подсистем как самой библиотеки SFML, так и дополнительных библиотек (SFGUI - Simple and Fast Graphical User Interface, Thor C++ Library), создание и проработка алгоритмов и объектной структуры приложения игры.

# АНАЛИТИЧЕСКАЯ ЧАСТЬ

## Описание игрового процесса

Игровой процесс игры «Красный квадрат» довольно прост – пользователь, управляя с помощью манипулятора «мышь» красным квадратом должен избегать столкновения с синими квадратами, которые хаотично движутся в игровом поле, и не выйти за границы своей игровой зоны, обозначенной желтой рамкой.

## Представление и отображение объектов игры

Приложение должно работать в графическом режиме, использовать для управления игровым персонажем манипулятор «мышь». Игровые объекты должны быть представлены в виде прямоугольников красного и синего цвета. После запуска приложения игра переходит в режим заставки, после которого должно появиться меню игры, одним из пунктов которого должен запускать игровой процесс. После окончания игры (при условии столкновения красного квадрата с синим или границами игровой зоны) должно появляться окно с текущим и лучшим за сеанс работы игры результатом.

## Взаимодействие игровых объектов

Взаимодействие игровых персонажей сводится к проверке на каждом игровом шаге пересечений синих квадратов, красного квадрата, управляемого пользователем с границей игровой зоны и окна. Пересечение границ красного квадрата с любым из синих квадратов или с границей игровой зоны приводит к завершению игры и переходу к окну отображения результатов. При столкновении синих квадратов друг с другом или с границами окна они должны «отскочить» - поменять вектор направления своего движения. Скорость движения синих квадратов возрастает с каждым игровым шагом.

# КОНСТРУКТОРСКАЯ ЧАСТЬ

## Описание средств и инструментов разработки

При разработке игрового приложения использовались интегрированные среды разработки Microsoft Visual Studio 2015 (2017) Community Edition, библиотеки Simple and Fast Multimedia Library (SFML) 4.2, Simple and Fast Graphical User Interface (SFGUI) 0.3.1, Thor C++ Library v.2.0.

Simple and Fast Multimedia Library (SFML) – объектно-ориентированная мультимедийная библиотека C++ с открытым исходным кодом, представляющая собой «прослойку» между аппаратным обеспечением и операционной системой с одной стороны и прикладной программой с другой стороны. SFML разбита на несколько модулей:

- System: ядро библиотеки, вокруг которого построены остальные модули. В нем описаны такие классы как двух- и трех-мерные вектора, таймеры, нити, строки Юникода и другие;

- Window: этот модуль позволяет создать окно приложения и перехватывать пользовательский ввод – события от «мыши» и клавиатуры;

- Graphics: предоставляет всю функциональность для рендеринга объектов, таких как изображения, текст, шейпы (shapes);

- Audio: SFML также имеет в своем составе модуль для работы со звуком, позволяющем загружать и воспроизводить звуковые файлы;

- Network: этот модуль позволяет обмениваться с другими приложениями через локальную сеть или Internet, используя протоколы HTTP или FTP.

Simple and Fast Graphical User Interface (SFGUI) – библиотека с открытым исходным кодом, которая позволяет реализовывать графический интерфейс пользователя на базе графической библиотеки SFML. Она содержит богатый набор виджетов, легко расширяема и конфигурируема при помощи поддержки графических тем.

## Описание объектной модели программной системы

Общая диаграмма классов приложения приведена на рис. 1. Объектная модель приложения построена таким образом, чтобы позволить простое расширение функционала приложения с минимальным внесением изменений в базовые классы приложения. Ядром приложения является экземпляр класса *Application*, который имеет члены-данные:

- *Window* класса *RenderWindow* – главного окна приложения SFML;

- *settings* класса *ContextSettings* – структуры с текущими графическими настройками рендеринга главного окна;

- *mGUI* класса *GUI* – представляющего интерфейс к библиотеке SFGUI для состояний приложения *StateMenu* и *StateScore*;

- *mGUD* класса *HUD* – для отображения информации о скорости рендеринга изображения;

- *Textures* класса *TextureHolder*, определенного через шаблон *ResourceHolder<sf::Texture, Textures::ID>* - для хранения текстур, используемых приложением;

- *Fonts* класса *FontHolder* определенного через шаблон *ResourceHolder<sf::Font, Fonts::ID>* - для загрузки и хранения шрифтов;

- *mStateStack* – класса *StateStack*, содержащего ссылки на состояния приложения и представляющего текущий стек состояний для обработки сообщений и рендеринга. Состояния приложения порождаются от базового класса *State* и переопределяют свое поведение в член-функциях: draw() – рендеринга изображения состояния, update(sf::Time dt) – обновления состояния, handleEvent(const sf::Event& event) – обработки системных сообщений.

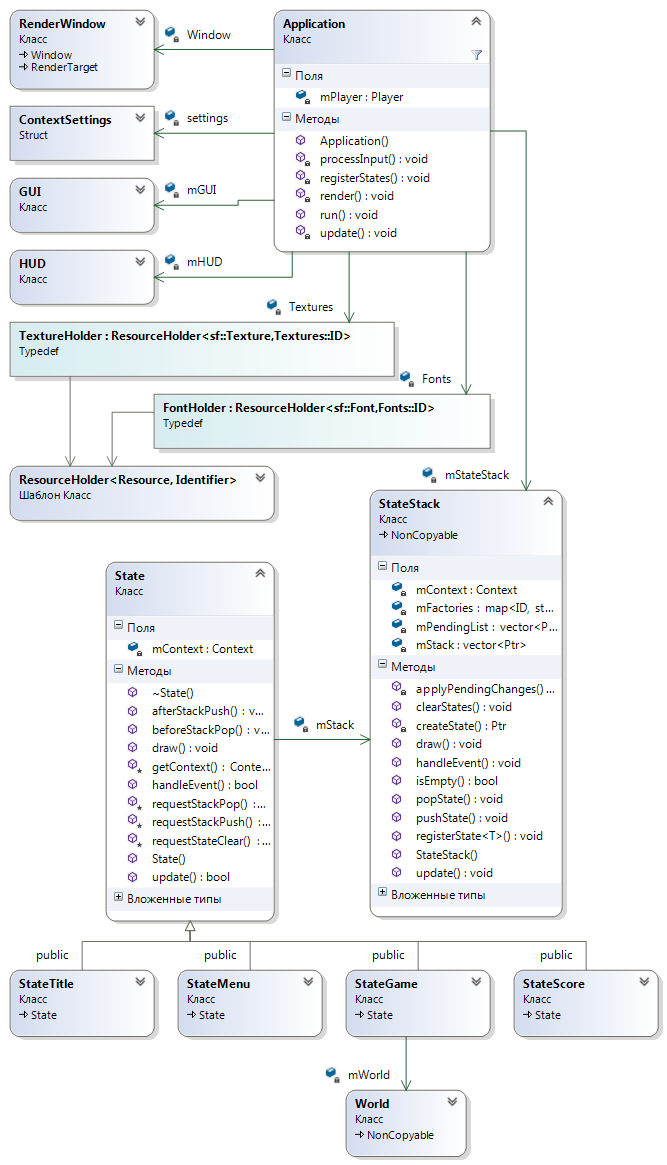


Рис. 1

Классы *StateTitle*, *StateMenu*, *StateGame*, *StateScore* (порожденные от базового класса *State*) реализуют поведение приложения в соответствующих состояниях: в заставке приложения, главном меню, состоянии игры и отображения результатов игры.

В качестве примера рассмотрим реализацию класса *StateMenu* (Листинг 1):

#include "StateMenu.h"

#include <SFML/Graphics/Font.hpp>

#include "ResourceManager.h"

#include <SFML/Graphics/RenderWindow.hpp>

StateMenu::StateMenu(StateStack & stack, Context context)

: State(stack, context) {

btnPlay = sfg::Button::Create(" P L A Y ");

btnPlay->GetSignal(sfg::Widget::OnLeftClick).Connect([this]

{ requestStackPop(); requestStackPush(States::Game); });

btnPlay->SetRequisition(sf::Vector2f(0.f, 64.f));

btnExit = sfg::Button::Create(" E X I T ");

btnExit->GetSignal(sfg::Widget::OnLeftClick).Connect([this]

{ requestStackPop(); });

btnReloadTheme = sfg::Button::Create(" reload theme ");

btnReloadTheme->GetSignal(sfg::Widget::OnLeftClick).Connect(

[this] {

if (std::ifstream("sfgui.theme"))

getContext().gui->sfgDesktop.LoadThemeFromFile("sfgui.theme");

});

menuBox = sfg::Box::Create(sfg::Box::Orientation::VERTICAL);

menuBox->SetSpacing(20.f);

menuBox->Pack(btnPlay, false);

menuBox->Pack(btnExit, false);

menuBox->Pack(sfg::Separator::Create());

menuBox->Pack(btnReloadTheme, false);

sfgWindow = sfg::Window::Create();

sfgWindow->SetTitle("Menu");

sfgWindow->SetStyle(sfg::Window::BACKGROUND);

sfgWindow->Add(menuBox);

auto wSize = sf::Vector2f(getContext().window->getSize()) / 2.f;

auto wPos = (sf::Vector2f(getContext().window->getSize())-wSize) / 2.f;

sfgWindow->SetAllocation(sf::FloatRect(wPos, wSize));

sfgWindow->SetId("main\_window");

menuBox->SetId("menu\_box");

btnPlay->SetId("btn\_play");

btnExit->SetId("btn\_exit");

btnReloadTheme->SetId("btn\_reloadtheme");

}

void StateMenu::draw() {

sf::RenderWindow& window = \*getContext().window;

getContext().gui->sfGUI.Display(window);

}

bool StateMenu::update(sf::Time dt) {

getContext().gui->sfgDesktop.Update(dt.asSeconds());

return false;

}

bool StateMenu::handleEvent(const sf::Event & event) {

getContext().gui->sfgDesktop.HandleEvent(event);

return false;

}

void StateMenu::beforeStackPop() {

getContext().gui->sfgDesktop.Remove(sfgWindow);

}

void StateMenu::afterStackPush() {

getContext().gui->sfgDesktop.Add(sfgWindow);

}

Листинг 1.

В конструкторе класса создаются экземпляры кнопок главного меню (*sfg::Button::Create*), устанавливаются их размеры и расположение и к их событиям привязываются обработчики событий (пример - *btnPlay->GetSignal(sfg::Widget::OnLeftClick).Connect([this] {requestStackPop(); requestStackPush(States::Game); });*). В последнем примере к событию *OnLeftClick*  виджета *btnPlay* присоединяется лямбда выражение с вызовом двух методов: *requestStackPop()* и *requestStackPush()*, которые «выталкивают» последнее состояние из стека состояний и помещают в него состояние *States::Game.* Затем созданные виджеты-кнопки компонуются вместе в виджет-контейнер *Box* вместе с разделителем и добавляются в окно SFGUI. Также виджетам устанавливаются соответствующие уникальные значения идентификаторов, которые можно использовать в файле настройки темы SFGUI для настройки визуальных свойств виджетов.

## Описание модулей приложения

oop-kp.cpp – основной модуль приложения, содержащий переменную *game* типа *Application* и точку входа приложения main, в которой происходит вызов *game.run()*;

* Application.h, Application.cpp – главные модули приложения;
* ResourceManager.h – модуль, содержащий описание шаблона *ResourceHolder* и классов, созданных на его основе *TextureHolder* и *FontHolder;*
* Entity.h, Entity.cpp – модуль с описанием базового класса, описывающего объект игры в общем смысле;
* BlueBox.h, BlueBox.cpp, RedBox.h, RedBox.cpp – модули, описывающие объекты игры красный квадрат и синие прямоугольники;
* World.h, World.cpp – модуль с описанием и реализацией класса *World* в котором реализована игровая механика;
* ClockHUD.h, FrameClock.h, HUD.h, HUD.cpp – вспомогательные модули для реализации диагностического счетчика кадров и пр.;
* GUI.h – модуль с базовым определением классов поддержки SFGUI;
* State.h , State.cpp – описание базового класса состояния приложения;
* StateGame.h, StateGame.cpp, StateTitle.h, StateTitle.cpp, StateMenu.h, StateMenu.cpp, StateScore.h , StateScore.cpp – содержат описания классов состояний приложения, порожденных от класса *State*;
* StateStack.h, StateStack.cpp – содержит реализацию стека состояний игры, отображаемых в каждую единицу времени.

# РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

## Комплект поставки и системные требования

Папка разработанного приложения содержит следующие файлы:

- исполняемый файл oop-kp.exe;

- библиотеки SFML и SFGUI: sfgui-d.dll, sfml-graphics-d-2.dl, sfml-system-d-2.dll, sfml-window-d-2.dll

- фай sfgui.theme, содержащий описание темы окон и виджетов SFGUI;

- подпапку **fonts**, содержащая TTF шрифты, используемые приложением;

- подпапку **textures**, содержащая текстуры объектов приложения.

## Установка и удаление программы

Для установки приложения необходимо скопировать папку, содержащую файлы приложения на жёсткий диск и установить распространяемый пакет Visual C++ для Visual Studio 2015 (vc\_redist.x86.exe). Программа не создаёт временных файлов и может быть запущена и непосредственно со съёмных носителей при условии наличия установленного распространяемого пакет Visual C++ для Visual Studio 2015.

Для удаления программы необходимо просто удалить папку приложения.

## Запуск программы

Запускаемым файлом является файл oop-kp.exe.

## Описание меню приложения

Меню приложения состоит из следующих пунктов (см. рис. 4):

- PLAY – запуск новой игры;

- EXIT – выход из приложения;

- RELOAD THEME – загрузка и применение темы SFGUI из файла sfgui.theme.



Рис. Главное меню игры

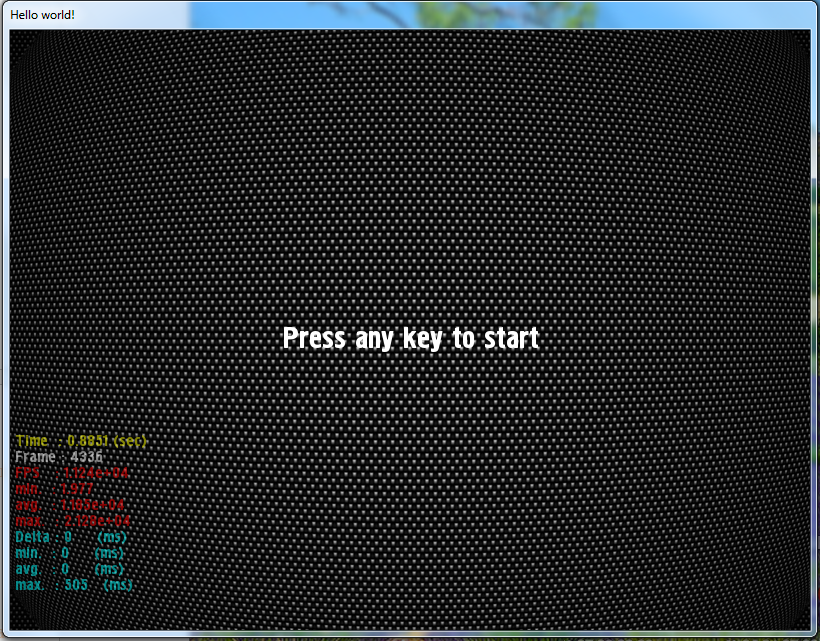


Рис. Начальная заставка

## Управление во время игры

Управление персонажем производится при помощи манипулятора «мышь». Красный квадрат движется в центральной позиции курсора.

При нажатии клавиши Escape игра останавливается и отображается меню программы.

По окончании игры появляется окно текущего результата (рис. 6).



Рис. 6 Окно с результатами игры

## Выход из игры

Для выхода из приложения нужно выбрать пункт меню EXIT.

# ЗАКЛЮЧЕНИЕ

В результате выполнения работы было разработано игровое приложение «Красный квадрат» в соответствии с заданием на курсовую работу.

Получены практические навыки создания приложений, использующих возможности кроссплатформенных мультимедийных библиотек с открытым исходным кодом SFML и SFGUI при помощи интегрированной среды разработки приложений Microsoft Visual Studio.

Разработанная программа соответствует полученному заданию, имеет простой интерфейс и модульную организацию, позволяющую расширять функциональность приложения.

# Список литературы

1. Джосаттис, Николаи М. Стандартная библиотека C++: справочное руководство, 2-е изд.: Пер. с англ. / Джосаттис, Николаи М. – М: ООО «И.Д. Вильямс», 2014. – 1136с.
2. Керниган, Брайан У. Язык программирования C / Брайан У. Керниган, Деннис М. Ритчи – М: Издательский дом «Вильямс», 2016. – 288с.
3. Мейерс, С. Эффективное использование STL / Скотт Мейерс – СПб: Питер, 2002 – 224с.
4. Мейерс, С. Эффективный и современный C++: 42 рекомендации по использованию C++11 и C++14.: Пер. с англ. / Скотт Мейерс – М.: ООО «И.Д. Вильямс», 2016. – 304с.
5. Страуструп, Б. Язык программирования C++ / Бьерн Страуструп – М: Бином, 2015. – 1136с.
6. Шилдт, Герберт. Полный справочник С++ / Герберт Шилдт – М: Издательский дом «Вильямс», 2006. – 800с.
7. Barbier, Maxime. SFML Blueprints / Maxime Barbier – Birmingham, UK: Packt Publishing, 2015. – 298p.
8. Documentation of SFML 2.4.2. Режим доступа: <https://www.sfml-dev.org/documentation/2.4.2/>
9. Documentation of the Thor Library. Режим доступа: <http://www.bromeon.ch/libraries/thor/documentation/v2.1/index.html>
10. Haller, Jan. SFML Game Development / Jan Haller, Henrik Vogelius Hannson, Artur Moreira – Birmingham, UK: Packt Publishing, 2013. – 296p.
11. Pupius, Raimondas. SFML Game Development By Example / Raimondas Pupius – Birmingham, UK: Packt Publishing, 2015. – 523p.
12. Simple and Fast Graphical User Interface. Режим доступа: <https://github.com/TankOs/SFGUI>
13. Tutorials for SFML 2.4. Режим доступа: [https://www.sfml-dev.org/  
    tutorials/2.4/](https://www.sfml-dev.org/tutorials/2.4/)