|  |  |  |
| --- | --- | --- |
| **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 21 | | |
|  | 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](#_Toc493254126)

[Введение 5](#_Toc493254127)

[1. АНАЛИТИЧЕСКАЯ ЧАСТЬ 7](#_Toc493254128)

[1.1. Описание игрового процесса 7](#_Toc493254129)

[1.2. Представление и отображение объектов игры 7](#_Toc493254130)

[1.3. Взаимодействие игровых объектов 8](#_Toc493254131)

[2. КОНСТРУКТОРСКАЯ ЧАСТЬ 9](#_Toc493254132)

[2.1. Описание средств и инструментов разработки 9](#_Toc493254133)

[2.2. Описание объектной модели программной системы 10](#_Toc493254134)

[2.3. Описание структур данных???????? 14](#_Toc493254135)

[2.4. Описание модулей приложения 15](#_Toc493254136)

[3. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 17](#_Toc493254137)

[3.1. Комплект поставки и системные требования 17](#_Toc493254138)

[3.2. Установка и удаление программы 17](#_Toc493254139)

[3.3. Запуск программы 18](#_Toc493254140)

[3.4. Описание меню приложения 18](#_Toc493254141)

[3.5. Управление во время игры 18](#_Toc493254142)

[3.6. Выход из игры 19](#_Toc493254143)

[ЗАКЛЮЧЕНИЕ 20](#_Toc493254144)

[Список литературы 21](#_Toc493254145)

# Введение

Игра «Красный квадрат»

Компьютерная игра Snake (Питон, Удав, Змейка и др.) появилась в середине-конце 1970-х. Игрок управляет длинным, тонким существом, напоминающим змею, которое ползает по плоскости (как правило, ограниченной стенками), собирая еду (или другие предметы), избегая столкновения с собственным хвостом и краями игрового поля. В некоторых вариантах на поле присутствуют дополнительные препятствия. Каждый раз, когда змея съедает кусок пищи, она становится длиннее, что постепенно усложняет игру. Игрок управляет направлением движения головы змеи (обычно 4 направления: вверх, вниз, влево, вправо), а хвост змеи движется следом. Игрок не может остановить движение змеи.

Первой игрой этого плана был игровой автомат Hustle, выпущенный фирмой Gremlin Industries в 1977 году, рассчитанный на одного или двух игроков, в которой нужно было управлять «змейками», направляя их на бессистемно появляющиеся цели. Для победы нужно было заполучить больше очков, чем у оппонента, преграждая по ходу игры ему путь к новым целям (в случае многопользовательской игры), или просто побить установленный на игровом автомате рекорд. Одна из первых реализаций на настольном микрокомпьютере была сделана в 1979 году на компьютере TRS-80, автор F. Seger (Германия). После неё вышла версия для Commodore VIC-20, под названием Worms. Микрокомпьютерная версия игры Hustle была выпущена Milton Bradley для компьютера TI-99/4A в 1980 году. Версии игры существовали и на многих советских компьютерах, например — на Радио 86РК. В число известных вариантов игры Snake входит игра Nibbles, некоторое время входившая в комплект MS-DOS. В настоящее время наиболее заметным представителем этого семейства является сетевая многопользовательская игра slither.io, в которой игроку необходимо «кормить» своего персонажа, избегать столкновения с другими игроками и охотиться на них, пресекая им путь.

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.

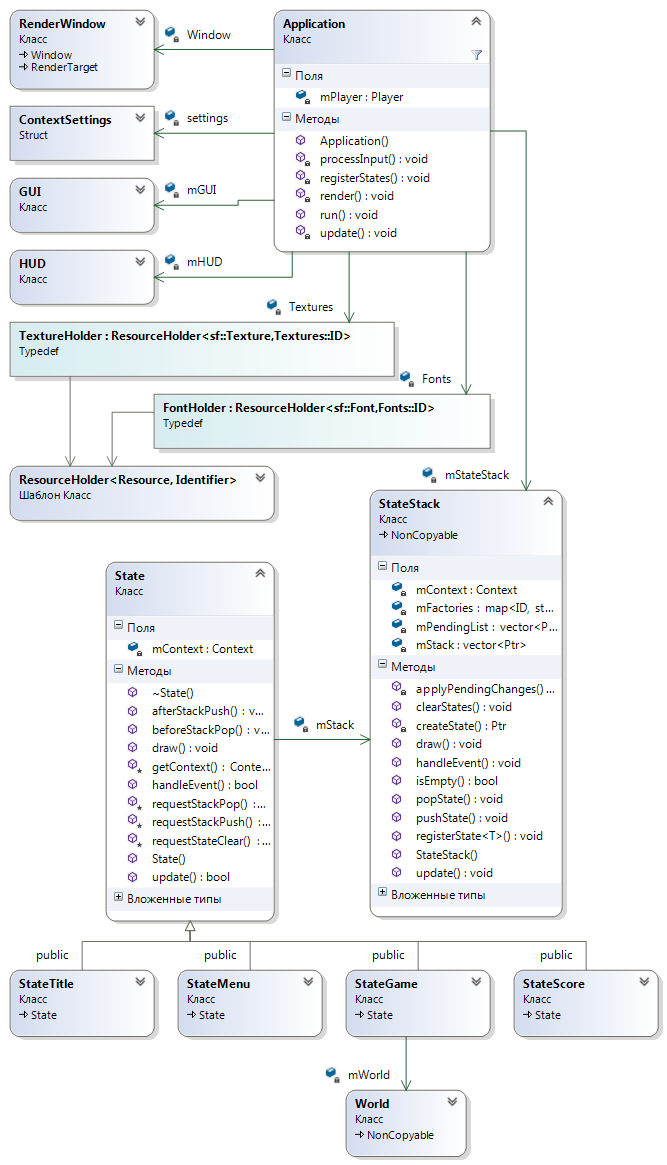


Рис. 1

Схема изменений состояния игры представлена на рис. 2. После вызова обработчиков событий управление передается функции рендеринга изображения do\_rendering, которая отрисовывает изображение фона (или заставки) и последовательно передает управление функциям отрисовки модулей приложения, которые в выводят изображение в рендерер в зависимости от значения флага game\_state.

void do\_rendering() {

if (!updateRequired)

return;

if (game\_state == gsInit)

SDL\_RenderCopy(renderer, intro\_texture, NULL, NULL); // draw intro

else

SDL\_RenderCopy(renderer, background\_texture, NULL, NULL);

// draw cursor circle....

filledCircleColor(renderer, mx,my, 12, 0xFF000080);

filledCircleColor(renderer, mx,my, 10, 0xF01010C8);

Stuffs\_Render();

Snake\_Render(&Python);

Score\_Render();

Menu\_Render();

Scores\_Render();

Text\_RenderInputText();

SDL\_RenderPresent(renderer);

…

}

Рис. 2

В качестве примера приведем обработчик события модуля game\_menu:

void Menu\_HandleEvent(const SDL\_Event event) {

if (game\_state != gsMenu)

return;

if (event.type == SDL\_KEYDOWN) {

SDL\_Keycode keyPressed = event.key.keysym.sym;

switch (keyPressed

}

Обработчик событий SDL данного модуля отрабатывает только тогда, когда приложение находится в состоянии gsMenu. При этом обрабатываются события от клавиатуры – при нажатии клавиш ВВЕРХ, ВНИЗ происходит перемещение активного пункта меню, при нажатии клавиши ВВОД – вызов функции, связанной с активным пунктом меню. При обработке событий от мыши (нажатия левой кнопки) определяется пункт меню, на котором находился курсор мыши и также происходит вызов функции, связанной с активным пунктом меню. В этих функциях происходит установка флага состояния игры.

## Описание структур данных????????

Пври реализации модели программной системы кроме стандартных типов (в т.ч. строк, массивов, структур) были использованы следующие виды структур данных:

Тип данных «Пункт меню»

typedef struct TMenuItem // Тип данных Пункт меню

{

char caption[MENUITEM\_CAPTION\_MAX\_LENGTH]; // Название пункта меню

TCallbackFunction callbackFunc; // (\*callbackFn)(sender);

SDL\_Texture \*captionTexture;

char visible;

Sint16 y1, y2;

} TMenuItem;

Тип данных «Меню»

typedef struct TMenu // Тип данных Меню

{

Uint8 item\_count;

Uint8 item\_current;

TMenuItem items[MENUITEM\_MAX\_COUNT];

Uint16 width;

Sint16 x1, x2;

Uint16 item\_height;

TTF\_Font\* menu\_font;

} TMenu;

Тип данных «Результат»

typedef struct TScore // Тип данных Результат

{

char name[ScoreNameLength];

Uint16 score;

SDL\_Texture \*scoreTexture;

} TScore;

Тип данных «Сегмент змеи»

typedef struct TSnakeSegment {

Sint16 x;

Sint16 y;

Uint16 n;

struct TSnakeSegment \*next;

struct TSnakeSegment \*prev;

} TSnakeSegment;

Тип данных «Змея»

typedef struct TSnake{

SDL\_TimerID timer\_id;

Uint16 speed;

struct {Sint16 x,y,dist;} dir;

TSnakeSegment \*head;

TSnakeSegment \*tail;

} TSnake;

Тип данных «Тип объекта»

typedef struct TStuffType // Тип данных Тип объекта

{

Uint16 radius; // Радиус спрайта этого типа объекта для вычисления rect от x,y

SDL\_Surface \*image;

SDL\_Texture \*texture;

} TStuffType;

Тип данных «Экземпляр объекта»

typedef struct TStuffItem TStuffItem;

typedef struct TStuffItem // Тип данных Экземпляр объекта

{

Uint16 type; // Тип объекта

TStuffItem \*prev; // Указатель на следующий экземпляр

TStuffItem \*next; // Указатель на следующий экземпляр

Uint16 x,y; // Координаты объекта

SDL\_Rect rect; // Прямоугольник отрисовки обьъекта

} TStuffItem;

Тип данных «Список объектов»

typedef struct TStuffList // Тип данных Список Экземпляров объектов

{

TStuffItem \*first;

TStuffItem \*last;

Uint8 count;

} TStuffList;

## 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

ResourceManager.h

Entity.h, Entity.cpp

BlueBox.h, BlueBox.cpp

RedBox.h, RedBox.cpp

World.h, World.cpp

ClockHUD.h

HUD.h, HUD.cpp

Utility.h, Utility.cpp, Foreach.h, ColorConsole.h

FrameClock.h

GUI.h

Player.h

SceneNode.h, SceneNode.cpp

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

# 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. 4 Главное меню игры

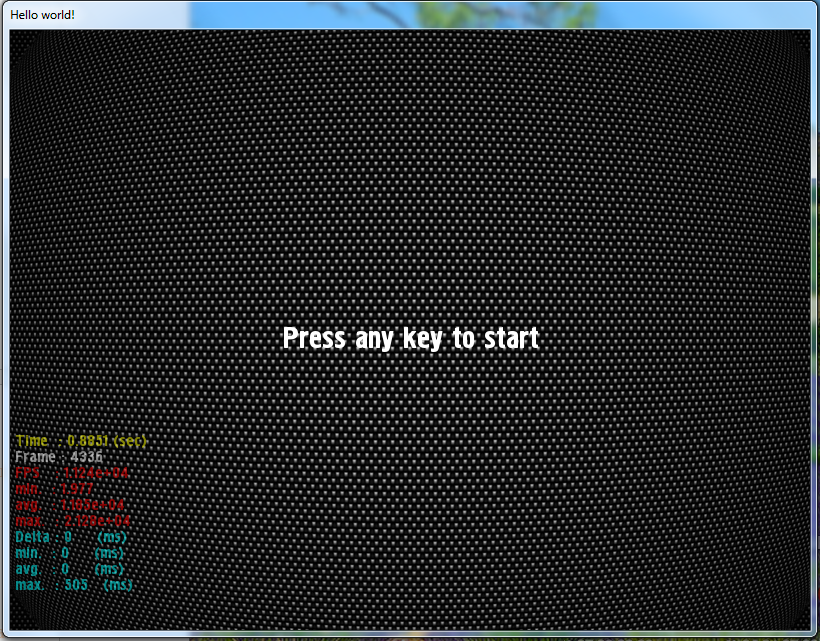


Рис. 3 Начальная заставка

Управление во время игры



Рис. 6 Окно с результатами игры

Управление персонажем производится при помощи манипулятора «мышь». Персонаж движется по игровому полю в направлении текущей позиции курсора. При этом необходимо избегать столкновения головы змеи с собственным телом и «несъедобными» объектами – это приводит к окончанию игры. При поедании «съедобных» объектов длина тела змеи увеличивается и на игровом поле в случайном месте появляется новый объект случайного вида. Текущий результат отображается в левом нижнем углу (см. рис. 4).

При нажатии клавиши Escape во время игры змея останавливается и отображается меню программы.

Рис. 7

По окончании игры появляется окно ввода имени пользователя для текущего результата (рис. 5).

При нажатии клавиши Enter после ввода имени появляется таблица результатов с введенным ранее именем (рис. 6).

Рис. 5

Из таблицы результатов можно вернуться в меню приложения для последующего выбора дальнейших действий.

Рис. 9

## Выход из игры

Для выхода из приложения нужно выбрать пункт меню EXIT.

# ЗАКЛЮЧЕНИЕ

Fsadfasdfasdf asdf

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/)