|  |  |  |
| --- | --- | --- |
| **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 «Питон».

Аннотация

На основании здания курсовой работы было разработано игровое приложение «Питон» («S.N.A.K.E.»). 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](#_Toc474247574)

[Введение 5](#_Toc474247575)

[1. АНАЛИТИЧЕСКАЯ ЧАСТЬ 7](#_Toc474247576)

[1.1. Описание игрового процесса 7](#_Toc474247577)

[1.2. Представление и отображение объектов игры 7](#_Toc474247578)

[1.3. Взаимодействие игровых объектов 8](#_Toc474247579)

[2. КОНСТРУКТОРСКАЯ ЧАСТЬ 9](#_Toc474247580)

[2.1. Описание средств и инструментов разработки 9](#_Toc474247581)

[2.2. Описание модели программной системы 10](#_Toc474247582)

[2.3. Описание структур данных 14](#_Toc474247583)

[2.4. Описание модулей приложения 15](#_Toc474247584)

[3. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 17](#_Toc474247585)

[3.1. Комплект поставки и системные требования 17](#_Toc474247586)

[3.2. Установка и удаление программы 17](#_Toc474247587)

[3.3. Запуск программы 18](#_Toc474247588)

[3.4. Описание меню приложения 18](#_Toc474247589)

[3.5. Управление во время игры 18](#_Toc474247590)

[3.6. Выход из игры 19](#_Toc474247591)

[ЗАКЛЮЧЕНИЕ 20](#_Toc474247592)

[Список литературы 21](#_Toc474247593)

# Введение

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 DirectMedia Library (SDL 2.0) и среды разработки приложений Microsoft Visual Studio в процессе создания игрового приложения в соответствии с заданием.

В задачи работы входило изучение подсистем как самой библиотеки SDL, так и дополнительных библиотек (подпроектов SDL), создание и проработка алгоритмов и структур данных игры.

# АНАЛИТИЧЕСКАЯ ЧАСТЬ

## Описание игрового процесса

Существует большое количество вариантов игры «Питон» («Змейка», Snake и т.д.). Для целей настоящей курсовой работы выбираем однопользовательский вариант классической игры в графическом варианте с некоторыми изменениями:

* персонаж движется по игровому полю в произвольном направлении в сторону курсора мыши;
* персонаж движется по игровому полю, поедая корм, при этом увеличивается его длина и текущий счет;
* персонаж движется по игровому полю, избегая столкновения с препятствиями и с собственным телом;
* при столкновении с препятствием или с собственным телом, игра заканчивается и пользователю предлагается ввести свое имя для таблицы результатов;
* в игре должно присутствовать меню, с помощью которого можно продолжить текущую игру, начать новую, просмотреть таблицу рекордов, завершить игру.

## Представление и отображение объектов игры

Так как игровое приложение будет работать в графическом режиме, то целесообразно главного персонажа игры представить в виде связанного набора сегментов круглой формы, перемещающихся друг за другом. Голова змеи будет представлена в виде круга, внутри которого находятся глаза, следящие за положением курсора мыши.

Объекты игры «Еда» и «Отрава» тоже будут представлены в виде круглых объектов с разными текстурами.

Текстуры сегментов змеи и других игровых объектов должны располагаться в папке приложения или подпапке с соответствующим именем.

## Взаимодействие игровых объектов

Взаимодействие главного персонажа игры сводится к проверке на каждом игровом шаге, пересекается ли голова змеи с каким-либо объектом или со своим сегментом тела (при этом нужно учитывать, что допускается пересечение с несколькими передними сегментами из-за того, что сегменты накладываются друг на друга). Перемещение змеи за единицу времени фиксированное и возрастание сложности игры достигается за счет удлинения тела змеи и увеличения количества несъедобных объектов на игровом поле.

# КОНСТРУКТОРСКАЯ ЧАСТЬ

## Описание средств и инструментов разработки

При разработке игрового приложения использовались интегрированные среды разработки Microsoft Visual Studio 2015 Community Edition и Code::Blocks 16.01 (MinGW 4.3.0, gcc gcc-4.9.3), библиотеки Simple DirectMedia Layer 2.0, SDL image 2.0.1, SDL ttf 2.0.14, SDL2 gfx 1.0.1.

Simple DirectMedia Layer (SDL) — свободная кроссплатформенная мультимедийная библиотека, которая реализует единый интерфейс к графической, звуковой подсистеме, и средствам ввода для широкого спектра платформ: Linux, Microsoft Windows, Mac OS X, iOS и Android. Программный интерфейс SDL доступен для многих языков программирования: C, C++, C#, VB.NET, D, Ada, Haskell, Java, Lisp, Lua, Pascal, Perl, Python, Ruby и других.

SDL можно рассматривать как некую прослойку, обеспечивающую поддержку для операций над пикселами, звука, доступа к файлам, обработки событий и т. п. SDL также можно использовать в дополнение к OpenGL, получая при этом простую поддержку мыши, клавиатуры и джойстиков. Библиотека состоит из нескольких подсистем, таких как Video, Audio, CD-ROM, Joystick и Timer. В дополнение к этой базовой низкоуровневой функциональности, существует ряд стандартных библиотек, предоставляющих дополнительную функциональность:

* SDL image — поддержка различных растровых форматов (в том числе JPEG, PNG), которая позволяет программисту загружать их в свое приложение без дополнительных усилий;
* SDL mixer — функции для организации сложного аудио, в основном, сведение звука из нескольких источников;
* SDL net — поддержка сетевых функций;
* SDL ttf — библиотека поддержки загрузки и рендеринга шрифтов TrueType, основанная на библиотеке freetype2 с поддержкой сглаживания;
* SDL rtf — отрисовка текста в формате RTF;
* SDL2 gfx - библиотека, предоставляющая возможности отрисовки базовых графических примитивов, таких как линии, круги, окружности, полигоны, прямоугольники и др. Также библиотека предоставляет функции для поворота и масштабирования текстур, применения различных фильтров к изображениям.

## Описание модели программной системы

Общая блок-схема алгоритма работы приложения представлена на рис. 1.

При запуске приложения управление передаётся функции инициализации initialization, в которой инициализируются подсистемы библиотеки SDL, создается окно приложения (SDL\_Window \*window), рендерер изображения (SDL\_Renderer \*renderer), загружаются текстуры для заставки и фона игры, затем вызываются процедуры инициализации модулей game\_menu, game\_scores, game\_snake, game\_stuff, game\_text. Затем создается основной таймер приложения, отвечающий за установку флага updateRequired через промежутки времени равные timer\_interval (по умолчания 20 мс).

В случае успешной инициализации приложения переменной game\_state присваивается значение gsInit и управление передается основному циклу, в теле которого происходит выборка сообщения из очереди сообщений SDL, его обработка (установка флагов и пр.), затем последовательный вызов обработчиков сообщений игровых модулей с передачей переменной event им в качестве параметра. Обработка событий от клавиатуры и мыши, а также рендеринг (формирование покадрового изображения игры) происходит в зависимости от значения глобальной переменной game\_state, которое, в свою очередь изменяется в зависимости от событий приложения.



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) {

case SDLK\_ESCAPE:

game\_state = gsRunning;

break;

case SDLK\_UP:

if (game\_menu.item\_current != 0)

game\_menu.item\_current--;

break;

case SDLK\_DOWN:

if (game\_menu.item\_current < (game\_menu.item\_count - 1) )

game\_menu.item\_current++;

break;

case SDLK\_RETURN:

if (game\_menu.items[game\_menu.item\_current].callbackFunc)

game\_menu.items[game\_menu.item\_current].callbackFunc(&game\_menu.items[game\_menu.item\_current]);

break;

}

}

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

if (event.button.button == SDL\_BUTTON\_LEFT) {

if ((event.button.x > game\_menu.x1) && (event.button.x < game\_menu.x2)) {

int i;

for (i = 0; i < game\_menu.item\_count; i++) {

if ((event.button.y > game\_menu.items[i].y1) && (event.button.y < game\_menu.items[i].y2)) {

//SDL\_Log("Clicked menu!\n");

if (game\_menu.items[i].callbackFunc) {

game\_menu.items[i].callbackFunc(&game\_menu.items[i]);

//SDL\_Log("Callback\n");

}

}

} // for i

}

}

}

}

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

При 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

* game\_main.c – модуль содержит функции инициализации и завершения приложения, функцию рендеринга изображения, таймер обновления экрана. В теле основной функции main содержится главный цикл обработки сообщений библиотеки SDL.
* game\_main.h – заголовочный файл содержит включения основных заголовочных файлов, определения путей к файлам текстур и шрифтов, глобальные переменные ширины и высоты экрана, переменную состояния игры;
* game\_menu.c, game\_menu.h – модуль содержит определения типов «Пункт меню», «Меню», функции инициализации, уничтожения, рендеринга и обработки сообщений меню;
* game\_scores.c, game\_scores.h – модуль содержит определения типов «Результат», переменной score\_current, содержащей текущий результат, массива Scores – таблицы результатов и функций инициализации модуля, сохранения результатов в файл, загрузки таблицы из файла ,а также рендеринга таблицы результатов и пр.
* game\_snake.c, game\_snake.h – модуль содержит определения типов «Сегмент змеи», «Змея», функции инициализации, уничтожения, рендеринга и обработки сообщений персонажем игры, а также переменную Python – главного персонажа;
* game\_stuff.c, game\_stuff.h – модуль содержит определения типов «Тип объекта», «Экземпляр объекта», функции инициализации, уничтожения, рендеринга, массив типов объектов, двухсвязный список объектов игры;
* game\_text.c, game\_text.h – модуль содержит определения переменных шрифтов, используемых в приложении, функции инициализации, уничтожения, рендеринга текстовых строк и поддержки текстового ввода с помощью SDL.

# РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

## Комплект поставки и системные требования

Папка разработанного приложения содержит следующие файлы:

* библиотеки SDL и др.: libFLAC-8.dll, libfreetype-6.dll, libjpeg-9.dll, libmodplug-1.dll, libogg-0.dll, libpng16-16.dll, libtiff-5.dll, libvorbis-0.dll, libvorbisfile-3.dll, libwebp-4.dll, SDL2.dll, SDL2\_image.dll, SDL2\_mixer.dll, SDL2\_ttf.dll, smpeg2.dll, zlib1.dll;
* запускаемый файл приложения sdl\_snake.exe;
* файл таблицы результатов scores.dat (при его отсутствии создается при запуске приложения);
* подпапка **fonts**, содержащая TTF шрифты, используемые приложением:
  + font\_menu.ttf – шрифт меню приложения,
  + font\_scores.ttf – шрифт таблицы результатов,
  + font\_sys.ttf – системный шрифт,
* подпапка **textures**, содержащая текстуры объектов приложения:
  + backgrnd.jpg – фон игры,
  + intro.jpg – заставка,
  + snake\_seg.png – текстура сегмента змеи,
  + stuff00.png – текстура объекта «отрава»,
  + stuff01.png – текстура объекта «еда».

## 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 sdl\_snake.exe.

## Описание меню приложения

Меню приложения состоит из следующих пунктов (см. рис. 3):

* Resume – Возврат к текущей игре,
* New game – начало новой игры,
* Scores – вывод таблицы результатов,
* EXIT – выход из приложения.

Выбор пунктов меню осуществляется мышью или при помощи клавиш Up, Down и Enter.

Рис. 3

## Управление во время 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).

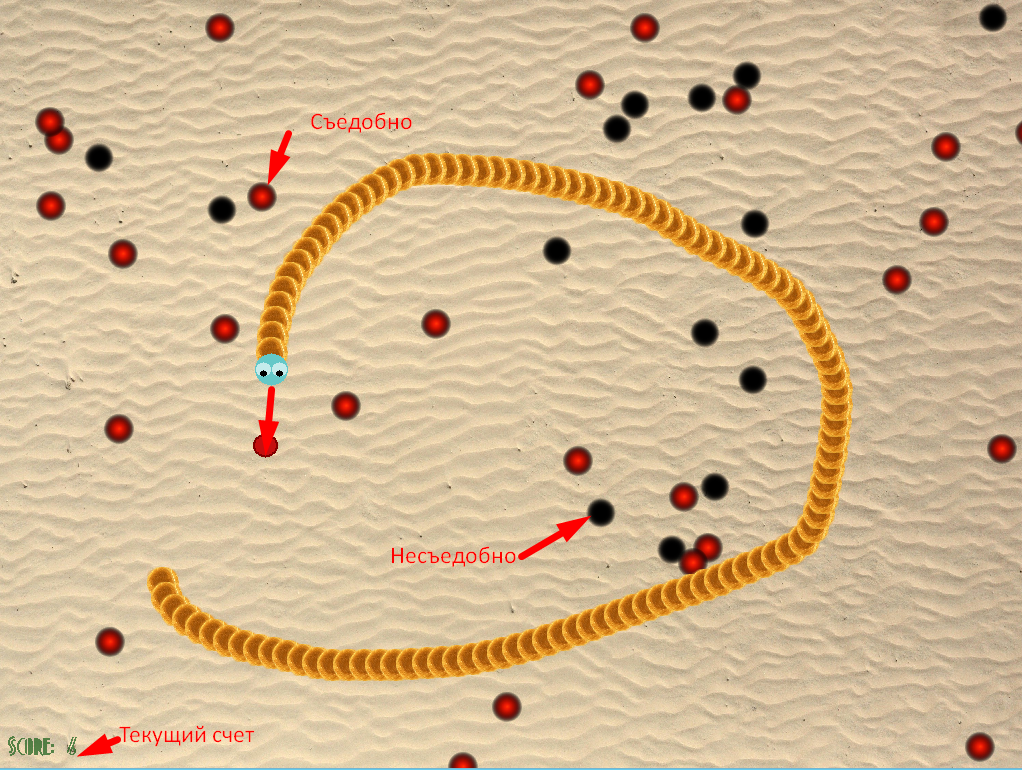
При нажатии клавиши Escape во время игры змея останавливается и отображается меню программы.

Рис. 4

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. 6

## Выход из игры

Для выхода из приложения нужно выбрать пункт меню EXIT.

# ЗАКЛЮЧЕНИЕ

В результате выполнения работы создано игровое приложение S.N.A.K.E., являющееся вариантом игры «Питон».

В ходе процесса выполнения работы была изучена история создания игр серии «Питон» («Змейка» и др.), основные методы и приемы программирования игровых приложений.

Получены практические навыки создания приложений, использующих возможности свободной кроссплатформенной мультимедийной библиотеки SDL 2.0 при помощи интегрированных сред разработки приложений Microsoft Visual Studio и Code::Blocks.

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

# Список литературы

1. Брайан У. Керниган. Язык программирования C / Брайан У. Керниган, Деннис М. Ритчи – М: Вильямс, 2016. – 288с.
2. Герберт Шилдт - Полный справочник С++. 2006
3. Джосаттис Н.М. - Стандартная библиотека C++. Справочное руководство – 2014
4. Страуструп Б. - Язык программирования C++ 4 изд.- 2013
5. Мейерс С. - Эффективный и современный C++. 42 рекомендации по использованию C++11 и C++14 – 2016
6. Мейерс С. - Эффективное использование STL
7. Documentation of SFML 2.4.2 <https://www.sfml-dev.org/documentation/2.4.2/>
8. Documentation of the Thor Library [http://www.bromeon.ch/libraries/thor/  
   documentation/v2.1/index.html](http://www.bromeon.ch/libraries/thor/documentation/v2.1/index.html)
9. SFML Blueprints. 2015
10. SFML Game Development
11. SFML Game Development By Example
12. Simple and Fast Graphical User Interface <https://github.com/TankOs/SFGUI>
13. Tutorials for SFML 2.4 <https://www.sfml-dev.org/tutorials/2.4/>