|  |  |  |
| --- | --- | --- |
| **МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ**  **РОССИЙСКОЙ ФЕДЕРАЦИИ** | | |
| ФГБОУ ВПО  **«БРЯНСКИЙ ГОСУДАРСТВЕННЫЙ**  **ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»** | | |
| **Кафедра «**Информатика и программное обеспечение**»** | | |
|  | |  |
| **КУРСОВАЯ РАБОТА** | | |
| РАЗРАБОТКА ИГРОВОЙ ПРОГРАММЫ «ПИТОН» | | |
| Вариант № 5 | | |
| Всего листов 19 | | |
|  | 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».

Сделать закладки на каждый номер рисунка!!!!!

Аннотация

В аналитической части рассмотрены основные понятия элек-тронной торговли, проведен анализ принципов построения интер-нет-магазина, показаны особенности работы интернет-магазинов,приведены наиболее распространенные интернет-магазины. Сфор-мулированы требования к разрабатываемому интернет-магазину.

В техническом задании определены основание для разработки и назначение разработки. Описаны основные требования к про-граммной системе и этапы ее разработки.

В исследовательской части рассмотрены различные вариантыоплаты услуг через интернет.

В конструкторской части рассмотрена архитектура программ-ной системы, выполнено обоснование выбора языка и средств раз-работки, проведено моделирование системы, базы данных и интер-фейса.

В части технической документации разработаны необходимые документы, сопровождающие программный продукт.

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

При разработке и кодировании использовался КОНЕЧНЫЙ АВТОМАТ.

При программировании используются основные принципы ООП: описана  
иерархия классов, задействован механизм виртуальных функций, организовано  
взаимодействие объектов программы посредством сообщений.  
В результате разработки получено устойчиво фукционирующее

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

# Содержание

[Содержание 5](#_Toc473817082)

[Введение 6](#_Toc473817083)

[1. АНАЛИТИЧЕСКАЯ ЧАСТЬ 8](#_Toc473817084)

[1.1. Описание игрового процесса 8](#_Toc473817085)

[1.2. Представление и отображение объектов игры 8](#_Toc473817086)

[1.3. Взаимодействие игровых объектов 8](#_Toc473817087)

[2. КОНСТРУКТОРСКАЯ ЧАСТЬ 9](#_Toc473817088)

[2.1. Описание средств и инструментов разработки 9](#_Toc473817089)

[2.2. **Описание Модель программной системы??????**. 10](#_Toc473817090)

[2.3. Описание структур данных 12](#_Toc473817091)

[2.4. **Описание модулей приложения**. 14](#_Toc473817092)

[3. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 15](#_Toc473817093)

[3.1. Комплект поставки и системные требования 15](#_Toc473817094)

[3.2. Установка и удаление программы 15](#_Toc473817095)

[3.3. Запуск программы 16](#_Toc473817096)

[3.4. Описание меню приложения 16](#_Toc473817097)

[3.5. Управление во время игры 16](#_Toc473817098)

[3.6. Выход из игры 17](#_Toc473817099)

[ЗАКЛЮЧЕНИЕ 18](#_Toc473817100)

[Список литературы 19](#_Toc473817101)

# 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

Описываем игровой процесс с точки зрения пользователя

## Представление и отображение объектов игры

## Взаимодействие игровых объектов

1. АНАЛИТИЧЕСКАЯ ЧАСТЬ.103

1.1. ИСТОРИЯ И КЛАССИФИКАЦИЯ КОМПЬЮТЕРНЫХ ИГР.103

1.2. ХАРАКТЕРИСТИКА АРКАДНЫХ ИГР.105

**1.3. ИСТОРИЯ ИГРЫ PACMAN.106**

1.4. ОБЗОР МЕТОДОВ ПРОГРАММИРОВАНИЯ АРКАДНЫХ ИГР.107

1.4.1. Представление игрового пространства.107

**1.4.2. Представление и отображение объектов игрового поля108**

**1.4.3. Взаимодействие объектов программы.109**

1.5. МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ.110

1.6. ПОСТАНОВКА ЗАДАЧИ.111

**1.6.1. Требования к инструментам разработки.112**

**1.6.2. Требования к программной совместимости.112**

1.6.3. Общие требования к функциональным характеристикам112

1.6.4. Требования к способам реализации.113

# 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 сам по себе довольно прост. Его можно рассматривать как некую прослойку, обеспечивающую поддержку для операций над пикселами, звука, доступа к файлам, обработки событий и т. п. SDL 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 библиотек 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. 2.

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. Схема изменений состояния игры представлена на рис. 1. 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. 1

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

}

}

}

}



общую блок-схему (структурограмму) и текстовое описание программы

Рис. 2

блок-схему (структурограмму) и текстовое описание, подкрепленное листингом, для наиболее интересного алгоритма программы;

## 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;

TMenu game\_menu;

Тип данных «Результат»

typedef struct TScore // Тип данных Результат

{

char name[ScoreNameLength];

Uint16 score;

SDL\_Texture \*scoreTexture;

} TScore;

TScore score\_current;

TScore Scores[ScoresCount];

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;

TSnake Python;

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;

TStuffType StuffTypes[stCount]; // Массив типов объектов

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;

TStuffList Stuffs;

## **Описание модулей приложения**.

game\_main.c

game\_main.h

game\_menu.c

game\_menu.h

game\_scores.c

game\_scores.h

game\_snake.c

game\_snake.h

game\_stuff.c

game\_stuff.h

game\_text.c

game\_text.h

# 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. 1):

* 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. 2).

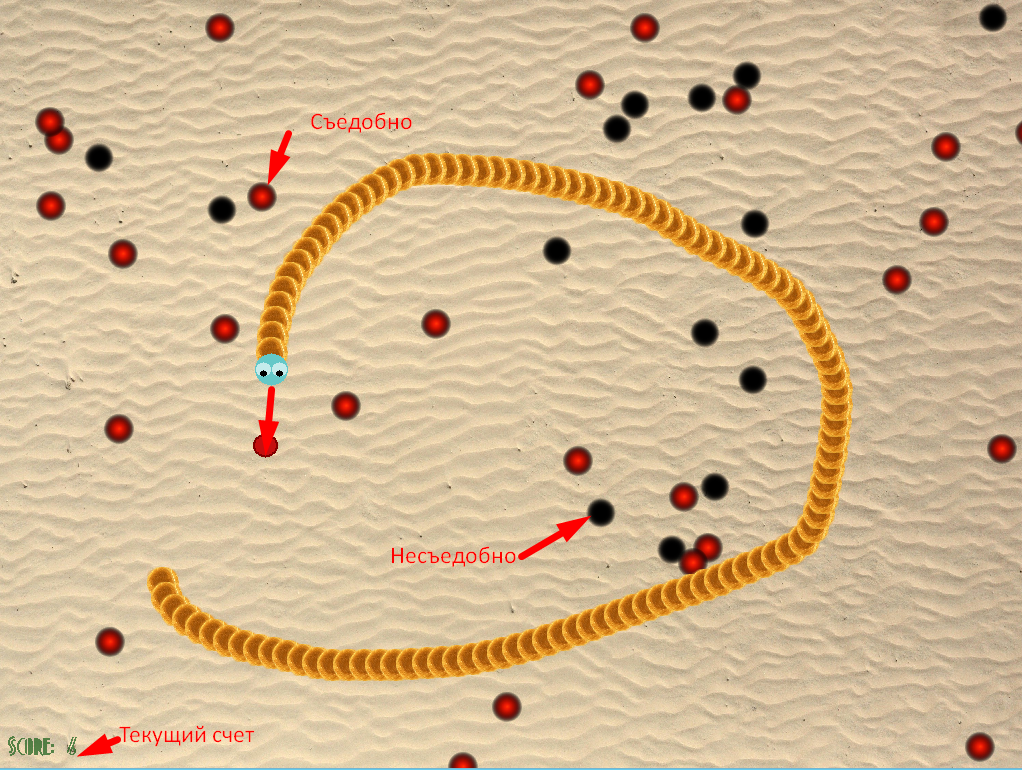
При нажатии клавиши Escape во время игры змея останавливается и отображается меню программы.

Рис. 2

По окончании игры появляется окно ввода имени пользователя для текущего результата (рис. 3).

При нажатии клавиши Enter после ввода имени появляется таблица результатов с введенным ранее именем (рис. 4).

Рис. 3

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

Рис. 4

## Выход из игры

Для выхода из приложения нужно выбрать пункт меню EXIT.

# ЗАКЛЮЧЕНИЕ

В результате выполнения работы создана ХХХХ

автоматизированная система, сочетающая в себе автоматизированное рабочее место врача-кардиолога и автоматизированное рабочее место для постовой медсестры.

В ходе была рассмотрена проблема информатизации медицинских учреждений, и определен круг задач, автоматизация которых позволяет

Изучены основные теоретические положения продукционных моделей данных. Получены практические навыки создания приложений баз данных на

В дальнейшем предполагается расширение функциональных возможностей разработанной автоматизированной системы в следующих направлениях:

1. Реализация автоматизированных рабочих мест для врачей приемного отделения, врачей-функционалистов, врачей- физиотерапевтов и лаборантов, что позволит перейти к электронным историям болезни пациентов и значительно сократить бумажный документооборот.

2. Разработка экспертной системы, позволяющей по вводимым жалобам пациента рекомендовать обследования и анализы.

Разработанная программа полностью соответствует полученному заданию, имеет удобный интерфейс и стройную организацию.

При разработке в полной мере использованы основные принципы объектно-ориентированного программирования и элементы технологии объектного моделирования.

Разработанная архитектура программы позволяет развивать прое кт,добавляя новых персонажей, усложняя их поведение, предоставляя дополнительные возможности пользователю

# Список литературы

1. Брайан У. Керниган. Язык программирования C / Брайан У. Керниган, Деннис М. Ритчи – М: Вильямс, 2016. – 288с.
2. SDL\_image library Documentation. Режим доступа: <https://www.libsdl.org/projects/SDL_image/docs/index.html>
3. SDL\_ttf library Documentation. Режим доступа: <https://www.libsdl.org/projects/SDL_ttf/docs/index.html>
4. SDL2\_gfx library Documentation. Режим доступа: <http://www.ferzkopp.net/Software/SDL2_gfx/Docs/html/index.html>
5. Shaun Ross Mitchell. SDL Game Development – Birmingham: Packt Publishing, 2013. – 256p.
6. Simple DirectMedia Layer library Wiki. Режим доступа: <https://wiki.libsdl.org/>