МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ И.С. ТУРГЕНЕВА»

Кафедра информационных систем

Работа допущена к защите

_____Руководитель

«18» 05 2018 г.

КУРСОВАЯ РАБОТА

по дисциплине «Информатика и программирование»

на тему: «Реализация игры змейка»

Студент _____ Кубарев А.Н.

Шифр 170551

Институт приборостроения, автоматизации и информационных технологий

Направление подготовки 09.03.03 «Прикладная информатика»

Направленность (профиль) Интеллектуальная обработка данных

Группа 71-ПИ

Руководитель _____ Амелина О.В.

Оценка: «отлично» Дата 19.05.18

Орел 2018

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ И.С. ТУРГЕНЕВА»

Кафедра информационных систем

УТВЕРЖДАЮ:
_____ Зав. кафедрой
«20» 02 2018 г.

ЗАДАНИЕ
на курсовую работу

по дисциплине «Информатика и программирование»

Студент Кубарев А.Н. Шифр 170551

Институт приборостроения, автоматизации и информационных технологий

Направление подготовки 09.03.03 «Прикладная информатика»

Направленность (профиль) Интеллектуальная обработка данных

Группа 71-ПИ

1 Тема курсовой работы

«Реализация игры змейка»

2 Срок сдачи студентом законченной работы «29» мая 2018

3 Исходные данные Условие задачи, возможности среды разработки по обработке событий и графического представления интерфейса, особенности выбранных структур данных.

4 Содержание курсовой работы

1) Титульный лист
2) Задание на курсовую работу
3) Введение
4) Постановка задачи
5) Обоснование выбора метода решения
6) Обоснование выбора структур данных
7) Описание алгоритма решения задачи
8) Описание пользовательского интерфейса
9) Описание результатов
10) Заключение
11) Список используемой литературы
12) Приложение, включающее листинг программы

5 Отчетный материал курсовой работы

Пояснительная записка курсовой работы; программа (на C++), записанная на CD-диске; презентация

Руководитель _____ Амелина О.В.

Задание принял к исполнению: «20» февраля 2018

Подпись студента_____

Содержание

Введение

В процессе выполнения работы будет создана компьютерная игра змейка, с графическим интерфейсом, будет создан пользовательский интерфейс для этой игры. Будет осуществлено знакомство с графической библиотекой OpenGL (набором именованных функций в OpenGL). Будет использоваться язык C++. Среда разработки Dev-C++ является оптимальным выбором для разработки поставленной задачи.

1 Постановка задачи

Цель работы - написать игру «Змейка». Правила игры: на поле размером 25*25 движется змейка, состоящая из нескольких сегментов и разбросана еда. Цель игры - двигаться по полю стрелками (вверх), (вниз), (влево), (вправо) и собирать еду. Каждая собранная еда увеличивает длину «Змейки» на один сегмент. Если змейка ударяется о стену, или пытается пройти через саму себя, игра заканчивается. Игрок выиграл, если змейка достигла пятидесяти сегментов в длину.

Целями данного курсовой работы являются:

- написание игры змейка;
- изучение основ проектирования игр;
- практика работы с графической библиотекой OpenGL;
- обучение создания пользовательского интерфейса.

2 Обоснование выбора метода решения и структур данных

Методом решения задачи был выбран метод визуализации 2d сцены в Dev-C++.

Процесс создания сцены в OpenGL начинается с позиционирования объема видимости в пространстве. Представьте, что мы установили камеру в каких-либо координатах. Теперь в данном пространстве мы устанавливаем некую модель (объект), которая будет попадать в объем видимости нашей камеры. Например, перед установленной камерой появился человек. Следующим шагом будет проецирование, в котором мы определим форму того объема в пространстве, который мы видим. И заключительным шагом мы получаем изображение объекта в рамках порта просмотра.

Для создания пользовательского интерфейса используется принцип модульности OpenGL.

Для создания более гибкой системы будем использовать много файловый проект.

3 Описание алгоритма

На старте программы запускается главный цикл OpenGL, glutmainloop. Программа должна иметь два состояния: пауза и процесс игры. Смена состояний будет производиться нажатием клавиши «Alt». При запуске игры должно происходить заполнения игрового поля элементами: ограждениями, фруктами, сеткой, частями «Змейки». Далее с каждым обновлением поля, элементы будут изменяться, а также будет происходить пересчет времени обновления поля. Процесс игры будет представлять собой управление змейкой при помощи клавиатуры. Если змейка натыкается на ограждение или на саму себя, то игра заканчивается. В заголовке окна программы будут отображаться игровые данные: имя программы и сам заголовок.

Общая схема работы программы представлена на Рисунке 1. Сначала создается игровое поле, после чего запускается основной цикл программы, который это поле обновляет.

Рисунок 1 - Начало основного цикла программы

Рисунок 2 демонстрирует алгоритм обновления игрового поля. При первом обновлении поле заполняется элементами, далее идет проверка на состояние паузы. Если состояние отлично от паузы, то происходит перемещение «Змейки» и перерисовка игрового поля, иначе выводится экран паузы.

Рисунок 2 - Алгоритм обновления игрового поля

На Рисунке 3 показана схема перемещения «Змейки». Первым делом проверяется направление «Змейки». Если оно обратно предыдущему, то изменения отменяются. Далее идет проверка на пересечение «Змейки» ограды

и фруктов. Если «Змейка» натолкнулась на ограду, то игра заканчивается. Если же змейка съела фрукт, то происходит переход на следующий уровень.

Рисунок 3 - Схема перемещения змейки, лист 1

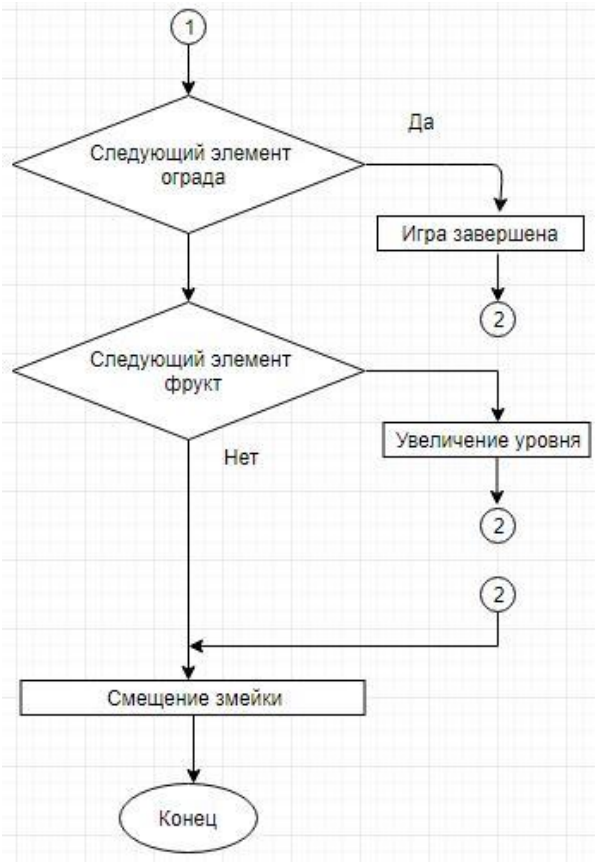

Рисунок 3 - Лист 2

4 Описание пользовательского интерфейса

В окне программы представлено поле иллюстрирующее игровое поле (рисунок 5).

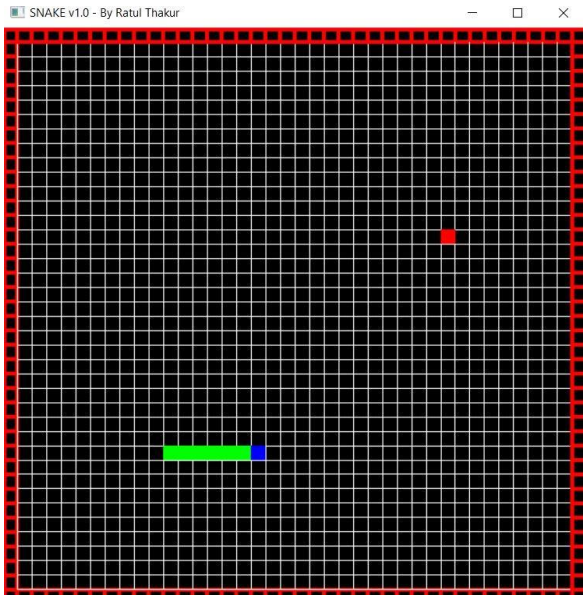

Рисунок 5 - Пользовательский интерфейс

Пользователю предоставляется выбрать направление «Змейки» путем, нажатия клавиш на клавиатуре, или нажать на паузу и выйти(рисунок 6).

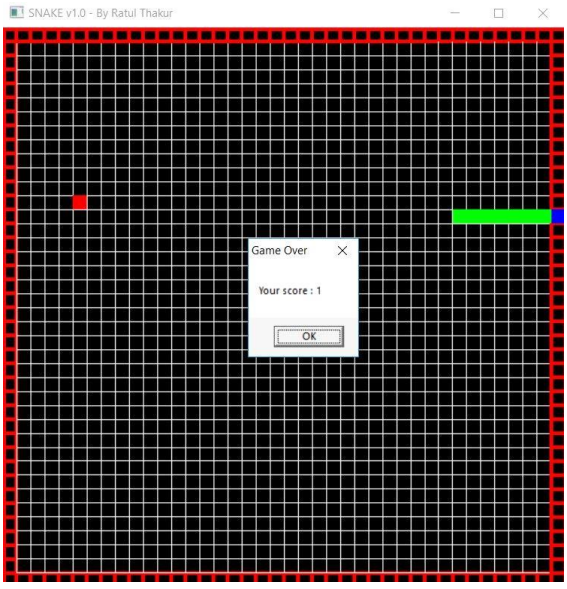

Рисунок 6 – Окончание игры

5 Описание результатов

Проведена реализация игры Змейка, и исследование библиотеки OpenGL .

Выбран наиболее подходящий графический набор методов OpenGL для решения данной задачи. Была изучена библиотека <GL/glut.h>, принципы реализации методов в OpenGL. Была проведена большая подготовительная работа по написанию блок схемы и разработан алгоритма системы.

Изучены основы проектирования игр. В задаче используется только один вводимый параметр, влияющим на передвижение «Змейки» Тем самым задача отражает реальность в больше степени, т.к. все остальные параметры в ней генерируются случайно (фрукты на карте), без вмешательства пользователя.

Перспективы развития данной игры: могут возникать препятствия появляться несколько фруктов одновременно, возможность ограничения фруктов по времени, появление уровней переход на игровой движок Unreal Engine.

Заключение

В процессе выполнения работы была создана компьютерная игра змейка, с графическим интерфейсом, был создан пользовательский интерфейс для этой игры. Осуществлено знакомство с графической библиотекой OpenGL (набором именованных функций в OpenGL).

Все задачи и цели были успешно выполненны и достигнуты:

- написание игры змейка;
- изучение основ проектирования игр;
- практика работы с графической библиотекой OpenGL;
- обучение создания пользовательского интерфейса.

Список используемой литературы

1 Канер, С. Тестирование программного обеспечения[Текст]: Пер. с англ./С. Канер, Д. Фолк, Кек Нгуен [и др.]-Киев: ДиаСофт, 2016. – 544 с.

2 Рудаков, А.В. Технология разработки программных продуктов[Текст]: Учеб. Пособие для студ. Сред. Проф. Образования./А.В. Рудаков.-Москва, Издательский центр «Академия», 2017. – 192 с.

3 Фридман, А.Л. Основы объектно-ориентированной разработки программных систем[Текст] / Л.И. Фридман - Москва, Финансы и статистика, 2017. – 192 с.

4 Плис, А.И. Математический практикум для инженеров и программистов[Текст]: Учеб. пособие. – 2-е изд. перераб. и доп. / А.И. Плис, Н.А. Сливина. – Москва, Финансы и Статистика, 2016. – 565 с.

5 Культин, Н.Б. C/C++ в задачах и примерах[Текст]: 2-е изд., перераб. и доп. (+CD) / Н.Б. Культин - И: «ЛАНЬ»,2016 г.

6 Кузнецов, М.В. C++. Мастер-класс в задачах и примерах [Текст]: (+ CD) / М.В. Кузнецов. - И: «ЛАНЬ», 2017 г.

Приложение А. Листинг программы

```
#include <GL/glut.h>
#include <iostream>
#include <fstream>
#include "game.h"

#define ROWS 40.0
#define COLUMNS 40.0

std::ofstream ofile;
std::ifstream ifile;
bool game_over=false;
extern int direction;
int score=0;

void init();
void display_callback();
void input_callback(int,int,int);
void reshape_callback(int,int);
void timer_callback(int);

int main(int argc,char**argv)
{
   glutInit(&argc,argv);
   // Режим отображения (ДВОЙНАЯ БУФЕРИЗАЦИЯ ОКНА)(RGBA)
   // ДЛЯ устранения мерцания при быстрой перерисовке окна
   glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB);
   // положение появления окна 10 px сверзу 10 px слева
   glutInitWindowPosition(10,10);
```

```
    // размер окна
    glutInitWindowSize(600,600);
    // заголовок окна
    glutCreateWindow("SNAKE v1.0 - Kubarev");

    glutDisplayFunc(display_callback);
    glutReshapeFunc(reshape_callback);
    glutSpecialFunc(input_callback);
    glutTimerFunc(100,timer_callback,0);
    init();
    // главный цикл openGl
    glutMainLoop(); return
    0;
}

void init()
{
        // цвет очистки окна - каким цветом будет наше окно
    glClearColor(0.0,0.0,0.0,0.0);
    // инициализируем сетку 40 x 40 (размер)
    initGrid(COLUMNS,ROWS);
}

//Callbacks
void display_callback()
{
    if(game_over)
    {
        ofile.open("score.dat",std::ios::trunc);
        ofile<<score<<std::endl;
```

```
            ofile.close();
            ifile.open("score.dat",std::ios::in);
            char a[4];
            ifile>>a;
            std::cout<<a;
            char text[50]= "Your score : ";
            strcat(text,a);
            MessageBox(NULL,text,"Game Over",0);
            exit(0);
        }
        // Очистка окна - в качестве параметра значение какой буфер нужно очистить
        // GL_COLOR_BUFFER_BIT - очистка буфера цвета
        glClear(GL_COLOR_BUFFER_BIT);
        //   очищаем матрицу - функция заменяет текущую матрицу на единичную
        glLoadIdentity();
            // рисуем сетку
        draw_grid();
        // рисуем еду
        draw_food();
        // рисуем змейку
        draw_snake();
        // переключаем буфер в режим двойной буферизации
        /*функция выполняет очистку конвеера OpenGl и переключение буферов(помещяет скрытое визуализированное изображение на экран)*/
        glutSwapBuffers();
    }
    void reshape_callback(int w, int h)
    {
```

```
        /* ОПРЕДЕЛЯЕМ ЗНАЧЕНИЕ ПОРТА ВЫВОДА */

        // определяем куда будет визуализироваться модель
    glViewport(0,0,(GLfloat)w,GLfloat(h));

    /*НАСТРОЙКА ПРОЕКЦИИ*/
    // Задаем матричный режим
    /*Будет определена матрица над которой будут производиться операции*/
    // GL_PROJECTION - Матрица проекций
    glMatrixMode(GL_PROJECTION);

    /*ОЧИЩАЕМ МАТРИЦУ*/
    // Функция заменяет заменяет текущую матрицу на единичную
    glLoadIdentity();
    glOrtho(0.0,COLUMNS,0.0,ROWS,-1.0,1.0);
    /*УСТАНАВЛИВАЕМ ОБЪЕКТНО ВИДОВУЮ
    МАТРИЦУ*/ glMatrixMode(GL_MODELVIEW);
    //   очищаем матрицу - функция заменяет текущую матрицу на единичную
    glLoadIdentity();
}
void timer_callback(int)
{
        // перерисовывае окна
    glutPostRedisplay();
    glutTimerFunc(100,timer_callback,0);
}
void input_callback(int key,int x,int y)
{
```

```
switch(key)
{
case GLUT_KEY_UP:
    if(direction!=DOWN)
        direction=UP;
    break;
case GLUT_KEY_DOWN:
    if(direction!=UP)
        direction=DOWN;
    break;
case GLUT_KEY_RIGHT:
    if(direction!=LEFT)
        direction=RIGHT;
    break;
 case GLUT_KEY_LEFT:
    if(direction!=RIGHT)
        direction=LEFT;
    break;
```