**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФГБОУ ВО «БРЯНСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**Кафедра** «Информатика и программное обеспечение»

**КУРСОВАЯ РАБОТА ПО ДИСЦИПЛИНЕ**

**«Компьютерная графика»**

«Тетрис»

Выполнил студент гр. 15-ИВТ2

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Шалом В.Ю.

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2018

Руководитель

\_\_\_\_\_\_\_\_\_\_\_\_\_к.т.н., доц. Трубаков Е.О.

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2018

Брянск 2018

**СОДЕРЖАНИЕ**

[Введение 3](#_Toc533661948)

[Техническое задание 4](#_Toc533661949)

[Аналитическая часть 5](#_Toc533661950)

[Анализ предметной области 5](#_Toc533661951)

[Выбор языка и платформы 6](#_Toc533661952)

[Конструкторская часть 7](#_Toc533661953)

[Этапы выполнения работы 7](#_Toc533661954)

[Проектирование 7](#_Toc533661955)

[Экспериментальная часть 11](#_Toc533661956)

[Руководство пользователя 11](#_Toc533661957)

[Тестирование 11](#_Toc533661958)

[Заключение 12](#_Toc533661959)

[Список литературы 13](#_Toc533661960)

[Приложение 14](#_Toc533661961)

## Введение

В настоящее время компьютеры повсеместно внедряются в различные сферы человеческой жизни, занимая свое место в нашем сознании. Вместе с появлением компьютеров появились компьютерные игры, которые сразу же нашли массу поклонников. С совершенствованием компьютеров совершенствовались и игры, привлекая все больше и больше людей. Современный компьютерный мир без игр немыслим.

Первые игры были пиксельные, не такие красочные и сложные, как сейчас. Они требовали куда большие затраты на разработку. Современные технологии позволяют создавать быстрее, проще и более сложные игры, чем это было раньше. Поэтому сейчас значительно повышается количество разработчиков игр, и, что более важнее, пользователей, которые играют в них и тестируют. Курсовая работа посвящена разработке игры «Тетрис».

Тетрис — компьютерная игра, первоначально изобретённая и разработанная советским программистом Алексеем Пажитновым. Игра была выпущена 6 июня 1984 года — в это время Пажитнов работал в Вычислительном центре Академии наук СССР.

«Тетрис» представляет собой головоломку, построенную на использовании геометрических фигур «тетрамино» — разновидности полимино, состоящих из четырёх квадратов. Первоначальная версия игры была написана на языке Паскаль для компьютера «Электроника-60». В последующие годы «Тетрис» во множестве различных версий был портирован на великое множество устройств, включая все возможные компьютеры и игровые консоли, а также такие устройства, как графические калькуляторы, мобильные телефоны, медиа-плееры, КПК и другие.

## Техническое задание

Целью данной курсовой работы по дисциплине «Компьютерная графика» состоит в закреплении и углублении знаний и навыков, полученных при изучении дисциплины. Курсовая работа представляет собой индивидуальное задание по реализации игры «Тетрис» на изучаемом языке высокого уровня с использованием библиотеки утилит «GLUT» (OpenGL Utility Toolkit) для написания приложений под OpenGL.

**Требования к функциональным характеристикам:**

* Реализация тетриса в 3D.
* Перемещение и вращение фигур.
* Вращение камеры/игрового поля.
* Вывод очков на экран.

## Аналитическая часть

### Анализ предметной области

Тетрис является одной из «классических» и самых узнаваемых практически любым человеком видеоигр. Существует бессчетное количество его реализаций на самых различных платформах, в том числе совершенно не рассчитанных для воспроизведения медиа-контента, таких как осциллографы.

Случайные фигурки тетрамино падают сверху в прямоугольный стакан шириной 10 и высотой 20 клеток. В полёте игрок может поворачивать фигурку на 90° и двигать её по горизонтали. Также можно «сбрасывать» фигурку, то есть ускорять её падение, когда уже решено, куда фигурка должна упасть. Фигурка летит до тех пор, пока не наткнётся на другую фигурку либо на дно стакана. Если при этом заполнился горизонтальный ряд из 10 клеток, он пропадает и всё, что выше него, опускается на одну клетку. Дополнительно показывается фигурка, которая будет следовать после текущей — это подсказка, которая позволяет игроку планировать действия. Темп игры постепенно увеличивается. Игра заканчивается, когда новая фигурка не может поместиться в стакан. Игрок получает очки за каждый заполненный ряд, поэтому его задача — заполнять ряды, не заполняя сам стакан (по вертикали) как можно дольше, чтобы таким образом получить как можно больше очков.

Начисление очков в разных версиях «Тетриса» довольно разнообразное. Очки могут начисляться за убранные линии, за сброшенные фигурки, за переход на новую скорость и тому подобное.

При начислении очков за линии количество очков обычно зависит от того, сколько линий убрано за один раз. Например, в китайских «Тетрисах», популярных в СНГ в 1990-х годах, начисление очков обычно было таким: 1 линия — 100 очков, 2 линии — 300 очков, 3 линии — 700 очков, 4 линии (то есть, сделать Тетрис) — 1500 очков. То есть, чем больше линий убирается за один раз, тем больше отношение количества очков к количеству линий. Любопытно, что тетрисом во многих версиях игры также называется действие, после которого исчезает сразу 4 линии. Это можно сделать только одним способом — сбросить «палку» (фигурку, в которой все клетки расположены на одной линии) в «шахту» ширины 1 и глубины как минимум 4.

### Выбор языка и платформы

В качестве языка программирования для реализации данного курсового проекта был выбран язык C++. C++ – компилируемый, статически типизированный язык программирования общего назначения. Поддерживает такие парадигмы программирования, как процедурное программирование, объектно-ориентированное программирование, обобщённое программирование.

OpenGL (Open Graphics Library) — спецификация, определяющая независимый от языка программирования платформонезависимый программный интерфейс для написания приложений, использующих двумерную и трёхмерную компьютерную графику.

OpenGL Utility Toolkit (GLUT) — библиотека утилит для приложений под OpenGL, которая в основном отвечает за системный уровень операций ввода-вывода при работе с операционной системой. Из функций можно привести следующие: создание окна, управление окном, мониторинг за вводом с клавиатуры и событий мыши. Она также включает функции для рисования ряда геометрических примитивов: куб, сфера, чайник. GLUT даже включает возможность создания несложных всплывающих меню.

Для реализации поставленной задачи была выбрана среда разработки Microsoft Visual Studio 2017, так как она наиболее подходит для создания данного проекта, имеет привычный и понятный интерфейс, большой набор функций и инструментов.

## Конструкторская часть

### Этапы выполнения работы

* Получение задания.
* Сбор и анализ информации.
* Проектирование.
* Разработка.
* Тестирование.
* Написание и защита отчета.

### Проектирование

#### Реализация программы

Для создания трехмерного тетриса был выбран язык C++ с использованием библиотеки «GLUT», упрощающей написания приложений на OpenGL. Объекты в трехмерном измерении имеют 3 координаты x, y, z, а на экране монитора только две координаты. По этой причине очевидным выходом будет вывод проекции. Также, предметы не попадающие в область видимости, например те, что находятся сзади камеры или стоящие сбоку – проецироваться не должны. Также не должны проецироваться предметы, находящиеся слишком далеко, т.е. тоже за пределами области видимости камеры. Эта область видимости устанавливается усеченной пирамидой, которая называется «Фрустум» (Frustum), чтобы показать предмет на экране, должно определять помещается ли он в этой усеченной пирамиде (части объекта, не помещающиеся в фрустум должны усекаться), только после этого проекция должна быть выведена на экран.

Приступим к рассмотрению основных функций.

Для объекта, представляющего фигуру, был написан класс «Shape». Он имеет такие поля как «size» и «color», описывающие размер фигуры и её цвет соответственно. По сути, фигура представляет из себя набор мелких кубов, каждый из которых может быть в «выключенном» или «включенном» состоянии, состояния в свою очередь выбираются на основе генератора случайных чисел, что позволяет всякий раз при выводе фигуры генерировать абсолютно случайные фигуры с максимальным размером 3x3x3, при этом логика генератора фигур не допускает генерации фигур, имеющих видимые разрывы, т.е. фигура не может иметь непристыкованные к ней кубы.

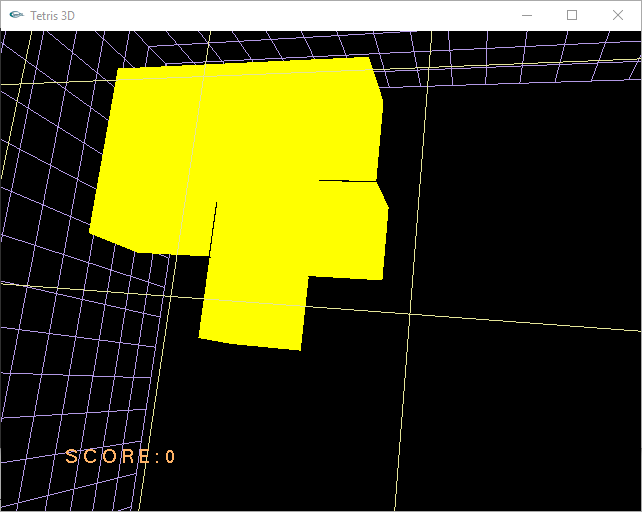


Рис. 1. Пример случайной фигуры

Также в классе реализованы методы, «поворачивающие» фигуру вокруг требуемой оси координат, т.е. rotateShape\_x() – вокруг оси «X», rotateShape\_y() – вокруг оси «Y», rotateShape\_z() – вокруг оси «Z».

Стандартные функции обратного вызова, определенные при инициализации для обработки ввода с клавиатуры обрабатывают нажатия таких клавиш как «Esc» - завершение программы, стрелки – перемещение фигуры, «X», «C», «V» - для поворота фигуры относительно осей координат. «F» и «D» - для приближения и отдаления соответственно.

Для вывода надписей на экран используется функция printTxt(), принимающая в качестве параметров координаты рисования текста на экране, а также строку, которую необходимо нарисовать.

|  |
| --- |
| void printTxt(float x, float y, char \*String) {  char \*c;  glPushMatrix();  glLoadIdentity();  glMatrixMode(GL\_PROJECTION);  glPushMatrix();  glLoadIdentity();  gluOrtho2D(0, 10, 0, 10);  glColor3f(9.0f, 0.7f, 0.4f);  glRasterPos2f(x, y);  glDisable(GL\_LIGHTING);  for (c = String; \*c != '\0'; c++)  glutBitmapCharacter(font, \*c);  glPopMatrix();  glMatrixMode(GL\_MODELVIEW);  glPopMatrix();  } |

Далее на рисунках изображены некоторые результаты взаимодействия с игровым полем.

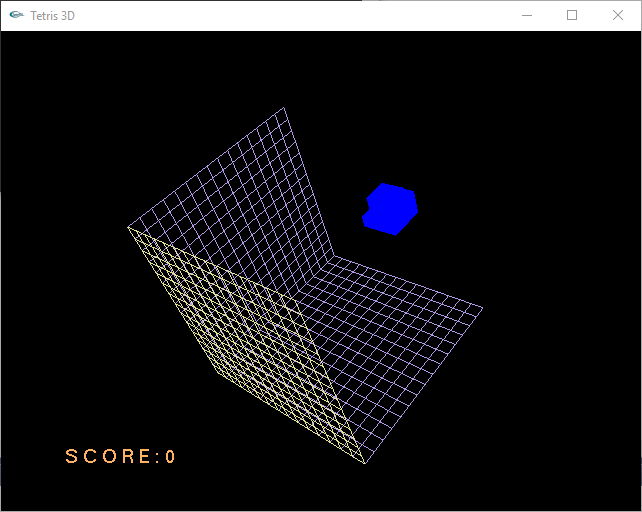


Рис. 2. Игровое поле после вращения

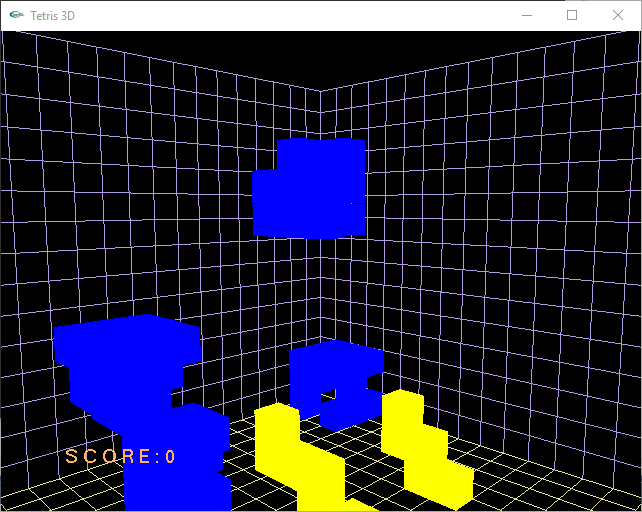


Рис. 3. Приближение камеры

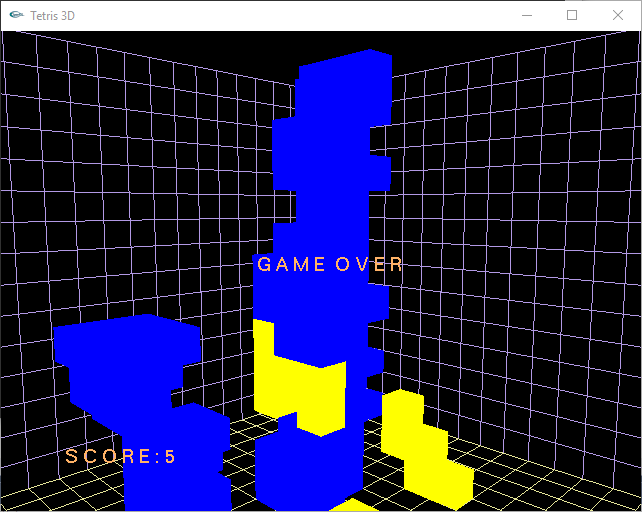


Рис. 4. Конец игры

.

## Экспериментальная часть

### Руководство пользователя

Для запуска программы необходимо запустить исполняемый файл «Tetris.exe», для корректного запуска программы в директории рядом с исполняемым файлом также должна находиться библиотека с именем «glut32.dll».

### Тестирование

#### Обработка перемещений и вращения фигур.

При обработке перемещений фигуры учитываются возможные столкновения данной фигуры с уже существующими, либо с границами трехмерной области. Тестирование системы обработки столкновений показало, что некоторых исключительных ситуациях возможны наслаивания фигур друг на друга. Статистически данная недоработка не имеет существенного влияния на игровой процесс из-за сложных условий ее повторения.

При проверке вращения фигур по каждой из трех осей координат не было выявлено существенных недоработок.

#### Тестирование камеры

Проверки вращений, а также приближения и отдаления камеры не выявили существенных ошибок или недоработок, влияющих на игровой процесс.

#### Вывод

Тестирование прошло удовлетворительно. По результатам тестирования были внесены правки с систему вращения фигур, тем самым предотвращая возможный выход фигуры за рамки игрового поля в результате чего тестирование можно считать успешно выполненным.

## Заключение

Для разработки приложения по теме курсового проекта была проанализирована графическая оболочка «OpenGL» и некоторые библиотеки-надстройки из которых было решено выбрать наиболее функциональную и оптимальную для решения поставленной задачи.

В рамках курсового проекта было разработано приложение, являющееся трехмерной интерпретацией классического «2D Тетриса» с перемещением и вращением фигур в трехмерном пространстве.

Тестирование показало наличие одной существенной и нескольких несущественных недоработок, в результате существенные недоработки были исправлены.

При выполнении данной работы пригодился опыт программирования в интегрированной среде разработки Microsoft Visual Studio и опыт разработки приложений с использованием языка высокого уровня C++. Чтобы подробно разобраться с выводом графики с использованием OpenGL и библиотеки GLUT приходилось обращаться к множеству дополнительных источников.

.

## Список литературы

1. Лафоре Р., Объектно-ориентированное программирование в С++. 4-е издание, 2013 г., 670 с.
2. Березин, Б. И. С и С++ / Б. И. Березин. -М.: Диалог-мифи, 2014. – 190 с.
3. Хейлсберг, А. Язык программирования C++, 2014 – 784c.
4. Стиллмен Э.,Грин, Д. Изучаем С++.3-е издание, 2017 – 360с.
5. Павловская, Т.А. С/С++ Программирование на языке высокого уровня - Спб.: Питер, 2013. – 350 с.
6. Вольф, Д. OpenGl 4. Язык шейдеров. Книга рецептов / Д.Вольф. – М.: PACKT, 2015. – 236 c.
7. Создание графических моделей. Режим доступа: https://www.intuit.ru/studies/
8. Learn OpenGL, extensive tutorial resource for learning modern OpenGL, [Электронный ресурс]: https://learnopengl.com/

## Приложение

Листинг 1. Метод генерации фигур

|  |
| --- |
| void Shape::initShape() {  srand(time(0));  bool ok = false;  for (int i = 0; i < 3; i++)  {  for (int j = 0; j < 3; j++)  {  for (int k = 0; k < 3; k++)  {  if (rand() % 100 > 40)  {  enableCubes[i][j][k] = 1;  size++;  if (size > 1)  {  if (i > 0 && !ok)  {  if (enableCubes[i - 1][j][k] == 1)  ok = true;  }  if (j > 0 && !ok)  {  if (enableCubes[i][j - 1][k] == 1)  ok = true;  }  if (k > 0 && !ok)  {  if (enableCubes[i][j][k - 1] == 1)  ok = true;  }  if (i < 2 && !ok)  {  if (enableCubes[i + 1][j][k] == 1)  ok = true;  }  if (j < 2 && !ok)  {  if (enableCubes[i][j + 1][k] == 1)  ok = true;  }  if (k < 2 && !ok)  {  if (enableCubes[i][j][k + 1] == 1)  ok = true;  }  if (!ok)  {  enableCubes[i][j][k] = 0;  size--;  }  else  {  ok = false;  }  }  }  }  }  }  if (size == 0)  {  enableCubes[1][1][1] = 1;  size++;  }  } |

Листинг 2. Функция рисовки поля

|  |
| --- |
| void display() {  float i;  char scr[20];  char scre[20];  strcpy\_s(scr, "S C O R E : ");  sprintf\_s(scre, "%d", score);  strcat\_s(scr, scre);  glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);  glMatrixMode(GL\_MODELVIEW);  printTxt(1.0f, 1.0f, scr);  glLoadIdentity();  gluLookAt(zoom, 2, 0, 0, 0, 0, 0, 1, 0);  glPushMatrix();  glColor3f(0, 1, 1);  glTranslatef(0, 0, 0);  glRotatef(leftRight\_rotation, 0, 1, 0);  glRotatef(upDown\_x, 1, 0, 0);  glRotatef(upDown\_z, 0, 0, 1);  glColor3f(0.7, 0.6, 0.9);  for (i = -N / 2; i <= N / 2; i++)  {  glBegin(GL\_LINES);  glVertex3f(i, N / 2, -N / 2);  glVertex3f(i, -N / 2, -N / 2);  glEnd();  }  for (i = -N / 2; i <= N / 2; i++)  {  glBegin(GL\_LINES);  glVertex3f(N / 2, i, -N / 2);  glVertex3f(-N / 2, i, -N / 2);  glEnd();  }  for (i = -N / 2; i <= N / 2; i++)  {  glBegin(GL\_LINES);  glVertex3f(-N / 2, N / 2, i);  glVertex3f(-N / 2, -N / 2, i);  glEnd();  }  for (i = -N / 2; i <= N / 2; i++)  {  glBegin(GL\_LINES);  glVertex3f(-N / 2, i, N / 2);  glVertex3f(-N / 2, i, -N / 2);  glEnd();  }  glColor3f(0.9, 0.9, 0.6);  for (i = -N / 2; i <= N / 2; i++)  {  glBegin(GL\_LINES);  glVertex3f(i, -N / 2, -N / 2);  glVertex3f(i, -N / 2, N / 2);  glEnd();  }  for (i = -N / 2; i <= N / 2; i++)  {  glBegin(GL\_LINES);  glVertex3f(N / 2, -N / 2, i);  glVertex3f(-N / 2, -N / 2, i);  glEnd();  }  glPopMatrix();  if (!gameOver)  {  shp.drawShape(d, movex, down, movez, leftRight\_rotation, upDown\_x, upDown\_z);  putInBase();  }  else  {  strcpy\_s(scr, "G A M E O V E R");  printTxt(4.0f, 5.0f, scr);  }  drawBase();  if (!stop)  {  down -= down\_speed;  }  glutSwapBuffers();  glFlush();  } |