Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
Московский государственный университет технологии и управления имени К.Г. Разумовского (Первый казачий университет)
Университетский колледж информационных технологий

Специальность 09.02.03 Программирование в компьютерных системах

К У Р С О В О Й П Р О Е К Т

Модуль ПМ.01 Разработка программных модулей программного обеспечения для компьютерных система
МДК.01.02 Прикладное программирование

на тему «РАЗРАБОТКА ПРОГРАММЫ “Кубик Рубика”»

Пояснительная записка
УКИТ 09.02.03.2015_303.019ПЗ

Группа ___*П-303*___

Студент _____ Мельдианов А.А.

(*личная подпись*)

Руководители проекта _____ Глускер А.И.

(*личная подпись*)

СОДЕРЖАНИЕ

I. ВВЕДЕНИЕ

Суть курсового проекта разработка программы «Кубик Рубика». Программа должна создавать кубик Рубика и давать возможность его собирать на время.

Данная программа актуальна тем что развивает логическое мышление пользователя.

Основной целью курсового проектирования является самостоятельная реализация программного продукта.

Задачи курсового проекта: приобретение и развитие навыков прикладного программирования, овладение навыками тестирования и отладки программного продукта, а также обучения использования технологии OpenGl для реализации 3D графики в программных продуктах.

Структура пояснительной записки состоит из этапов разработки программного продукта, оформленных по ГОСТам и объединенных в одном документе. Состав пояснительной записки:

-Титульный лист

-Содержание

Включает в себя название разделов и номера их страниц

-Введение

-Основная часть

1. Спецификация
2. Программа и методика испытаний
3. Технический проект программного изделия
4. Реализация программного изделия на языке программирования
5. Тестирование программного продукта

-Заключение

-Список используемой литературы

-Приложения

Основные методы, которые были использованы:

Анализ - представляет собой процессов или явлений на составные компоненты и предполагает их дальнейшее изучение.

Синтез – предполагает соединение нескольких свойств исследуемого объекта в один компонент.

Структурное и модульное программирование. Принципы структурного программирования - управляющие структуры. Можно использовать только 3 типа управляющих структур:

- Следование (операторы)
- Условные конструкции
- Циклы

II. ОСНОВНАЯ ЧАСТЬ

1. СПЕЦИФИКАЦИЯ

Данный этап нужен для формирования структуры программного продукта. Здесь описаны требования к функциональным характеристикам, требования к интерфейсу программы, требования к надежности, условия эксплуатации, требования к составу и параметрам технических средств, требования к исходным кодам, требования к программной документации.

1.1. Требования к функциональным характеристикам

1.1.1. Требования к составу выполняемых функций

1.1.1.1. отображения «кубика Рубика» на экране

1.1.1.2. осуществление поворотов его граней

1.1.1.3. Вращение «кубика Рубика» вокруг своей оси (x.y.z)

1.1.1.4. возможность «собирания» «кубика Рубика»

1.1.1.5. Запись\чтение рекордов

1.1.2. Требования к надежности

1.1.2.1. Надежное функционирование программы обеспечивается проверкой нажатий мыши на игровую область, а также проверка нажатия клавиш на клавиатуре

1.1.3. Дополнительные сведения по программному продукту

1.1.3.1. Сборка «кубика Рубика» выполняется по алгоритмам, которые можно найти в с помощью поисковых систем (Google ,Yandex ,Mail Поиск и другими) или самостоятельно разработав алгоритм.

1.1.3.2. Сборка считается законченной только в том случае если каждая сторона кубика будет одного цвета

1.1.3.3. Изменение выбора стороны для последующего поворота осуществляется с помощью кнопок на клавиатуре «A», «W», «S», «D», а поворот грани на кнопку «Space» («Пробел»)

1.2. Требования к составу и параметрам технических средств

1.2.1. Процессоры Intel, AMD

1.2.2. Мышь

1.2.3. Клавиатуру

1.2.4. Монитор

1.2.5. HDD или SDD накопитель

1.2.6. Видео адаптер

1.3. Требования к Информационной и программной совместимости

1.3.1. Требования к используемой технологии: OpenGl

1.3.2. Требования к исходным кодам изложены в документе А.И. Глускер Методические рекомендации по выполнению курсового проекта МДК 01.02 «Прикладное программирование».

1.3.3. Программа должна работать на Windows Xp или более поздней

1.4. Специальные требования

1.4.1. Программа должна быть реализована с использованием OpenGl технологии для от рисовки графики

1.4.2. Программа должна обеспечивать взаимодействие с пользователем посредством графического пользовательского интерфейса

2. Состав программной документации

2.1. Состав программной документации должен содержать

2.1.1. Техническое задание

2.1.2. Пояснительную записку

2.1.3. Текст программы

2.1.4. Программу методики испытаний

2.2. Специальные требования к пояснительной записке

2.2.1. Пояснительная записка должна содержать блок-схему(блок-схемы) алгоритма(алгоритмов) используемых в программе

3. Стадии и этапы разработки

3.1. Разработка осуществляется в три стадии

3.1.1. Техническое задание

3.1.2. Технической проект

3.1.3. Рабочий проект

3.2. Этапы разработки

3.2.1. На стадии технического задания должен быть выполнен этап разработки, согласования и утверждения настоящего технического задания.

На стадии рабочего проектирования должны быть выполнены перечисленные ниже этапы работ:

3.2.1.1. Разработка программы

3.2.1.2. Разработка программной документации

3.2.1.3. Испытания программы

4. Порядок контроля и приемки

4.1. Приемосдаточные испытания должны проводиться в соответствии с программой и методикой испытаний, разработанной, согласованной и утвержденной не позднее 31 декабря 2015 года.

Я описал основные требования к документации и программному продукту. В требованиях к функциональным характеристикам, я описал основные функции, которые должны быть в программе, В требованиях к надежности, я описала, средства используются для считывания действий пользователя, чтобы программа работала корректно. В условиях эксплуатации, я описал минимум пользователей, которые могут использовать программу. В требованиях к составу и параметрам технических средств, я описал, что должно входить в компьютер, для использования программы. В требованиях к исходным кодам, я описал, какая документация используется для создания кода. В требованиях к программной документации, я описала, состав программной документации, специальные требования.

2. ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ

Этот этап нужен для проведения приемочных испытаний и разрабатывается на основе документации спецификации. Здесь описаны программные средства, порядок проведения испытаний, методы испытаний, метод проверки требований к составу программной документации, метод проверки требований к исходным кодам.

1. Программные средства, используемые при проведении испытаний

1.1.1. В состав программных средств входит

1.1.1.1. лицензионная копия операционной системы Windows Seven

1.1.1.2. Visual Studio 2012

2. Порядок проведения испытаний

2.1. Подготовка к проведению испытаний заключается в обеспечении наличия компьютера, описанного в п. 1.2 (спецификации), и программных средств, указанных в пункте выше (п.1.1), установленных на этом компьютере

2.2. Состав испытания

2.2.1. Проверка состава программной документации в соответствии с методом, описанном в п. 3.2

2.3. Проверка требований к программе

2.3.1. Проверка обеспечения требований к программе (п. 1.1.1(спецификации)) в соответствии с методом, описанным в п. 3.1

2.3.2. Проверка требований к программной документации

2.3.3. Проверка пояснительной записки (п. 2 (спецификации)) в соответствии с методом, описанным в п. 3.3

2.3.4. Проверка текстов программ (п. 1.3.2(спецификации)) в соответствии с методом, описанным в п. 3.4

3. Методы испытаний

3.1. Метод проверки требований к программе

3.1.1. Картинка открытой игры

3.1.2. Картинка при нажатии Новая игра

3.1.3. Картинка открытого меню при этом должна быть пауза

3.1.4. Картинки где я нажимаю на AWSD

3.1.5. Картинка где я нажал Space

3.1.6. Картинка где я верчу кубик

3.1.7. Картинка где выводится уведомление что кубик собран

3.1.8. Картинка записи рекорда

3.1.9. Картинка где нажата кнопка рекорды

3.2. Метод проверки требований к составу программной документации

3.2.1. Проверка состава программной документации осуществляется визуально путем сравнения набора предъявленных документов (в форме распечатки или в рукописной форме) При этом исходные тексты программ должны быть предоставлены так же и в электронной форме.

3.2.2. В случае если набор предъявленных документов соответствует списку, а исходные тексты предоставлены также в электронной форме, то в протокол заносится запись: «Состав программной документации» – соответствует; в противном случае: «Состав программной документации» – не соответствует.

3.3. Метод проверки требований к пояснительной записке

3.3.1. Проверка состоит из следующих этапов:

3.3.1.1. проверка наличия блок-схемы (блок-схем) в пояснительной записке

3.3.1.2. проверка соблюдения требований ГОСТ 19.701-90 для каждой блок-схемы

3.3.1.3. проверка соблюдения локальных стандартов для блок-схем

3.3.1.4. проверка соответствия каждой блок-схемы алгоритму, закодированному в программе

3.3.2. Проверка соблюдения требований ГОСТ 19.701-90 состоит из следующих работ:

3.3.2.1. проверка использования только тех символов, которые указаны как применимые к схемам программ в п. 5 ГОСТ 19.701-90

3.3.2.2. проверка соответствия символов их назначению (экспертная оценка лица, проводящего испытания)

3.3.2.3. проверка правильности выполнения соединения линий (п. 4.2.3 ГОСТ 19.701-90)

3.3.2.4. проверка того, что линии потока управления, выходящие из символа «решение» подписана (п. 4.3.1.2 ГОСТ 19.701-90

3.3.3. Проверка соблюдения локальных стандартов для блок-схем состоит из следующих работ:

3.3.3.1. проверка того, что все символы (кроме терминаторов, соединителей, линий и комментариев) имеют одинаковые размеры

3.3.3.2. проверка того, что терминаторы имеют ту же ширину, что и другие символы

3.3.3.3. проверка того, что отношение ширины к высоте составляет 2 к 1 для каждого символа, кроме терминаторов, комментариев и линий

3.3.3.4. проверка того, что отношение ширины к высоте составляет 4 к 1 для терминаторов

3.3.3.5. проверка того, что высота соединителей совпадает с высотой терминаторов

3.3.3.6. проверка того, что линии потока управления входят в символ слева или сверху, а выходят снизу или справа

3.3.3.7. проверка того, что подписи к линиям не находятся на самих линиях.

3.3.3.8. Проверка соответствия каждой блок-схемы алгоритму, закодированному в программе, осуществляется путем экспертной оценки лицом, осуществляющим проведение испытаний

3.3.4. В случае, если все вышеприведенные проверки прошли успешно, в протокол заносится запись: «Специальные требования к пояснительной записке» – соответствует; в противном случае «Специальные требования к пояснительной записке» – не соответствует.

3.4. Метод проверки требований к исходным кодам

3.4.1. Изложенный ниже метод применяется ко всем файлам, содержащим исходный текст, и входящим в состав программной документации по

отдельности. Для каждого файла вносится в протокол запись: «Требования к исходным кодам для файла ####» – соответствует, /не соответствует (где вместо #### указывается название файла).

3.4.2. Проверка состоит из следующих этапов:

- Наличие комментариев к неочевидным действиям (проверяется методом экспертной оценки лицом, осуществляющим испытания)
- Для каждой подпрограммы наличие комментария, содержащего полное описание ее работы, описание всех аргументов и результатов. Достаточность этого комментария для возможности использовать подпрограмму в других программах (без изучения собственно, текста подпрограммы).
- Для каждой глобальной переменной указание ее назначения.
- Для всех переменных, кроме переменных цикла, использование «говорящих» названий
- Для всех подпрограмм использование говорящих названий.
- Использование одного оператора на одной строке программы.
- Количество пробелов перед строкой программы должно соответствовать уровню вложенности (по два пробела на уровень вложенности).
- Слова begin и end, соответствующие друг другу, располагаются строго с одной и той же позиции по вертикали.

- Количество строк в подпрограмме и в самой программе (между begin и end) – не более 50 строк.
- Использование модулей для трех и более сходных по назначению подпрограмм.
- Отсутствие в подпрограммах использования глобальных переменных (напрямую).
- Разделение подпрограмм на предназначенные для вычислений (в них не должно
- быть ввода-вывода) и на предназначенные для ввода-вывода (в них вычисления должны быть только такие, что нужны для ввода-вывода).
- Отсутствие операторов goto, break, continue; процедур halt и exit.
- Проверка того, что вместо явно указанных значений чисел, в тексте программы используются константы.

В случае, если все перечисленные этапы пройдены, то в протокол о соответствии файла требованиям, в противном случае – о несоответствии.

Я описал программные средства, которые будут использованы для проведения испытаний. Порядок проведения испытаний, здесь я описал, состав испытаний. Методы испытаний, здесь я описал, методы проверки программы, там написаны функции, и как их проверить. Метод проверки требований к составу программной документации, там описаны методы проверки документации по ГОСТам, метод проверки требований к исходным кодам, там описаны методы проверки исходного кода по ГОСТам.

3. ТЕХНИЧЕСКИЙ ПРОЕКТ ПРОГРАММНОГО ИЗДЕЛИЯ

Этот этап нужен для определения внутренних свойств программы и детализации внешних свойств. Процесс проектирования и его результаты зависят не только от требований, изложенных в этапах выше, но и от выборной модели процесса, здесь нарисована блок-схема по алгоритму «название алгоритма»

1. Технические характеристики

1.1Описание алгоритма

Блок Схема

2.1 Описание и обоснование выбора языка программирования и программных средств

1.1 Для разработки программного продукта был выбран язык C# так как мне очень импонирует этот язык программирования и хотел начать писать программы на этом языке, а курсовая работа была наилучшим «толчком» к началу изучения этого языка программирования

2.1 Название программ где я буду которые я использовал

1. MS Word - с помощью этой программы я написал пояснительную записку
2. MS Power Point - с помощью этой программы я сделал презентацию
3. LibreOffice Draw – с помощью этой программы я сделал блок-схему
4. Git – с помощью этой программы я создавал систему контролей версии
5. Tao Framework – графическая библиотека, была использована для получение возможностей OpenGL
6. Yandex Browser - для поиска информации

Я составил блок-схему для наглядности алгоритма «Название алгоритма» в программе. Также я описал все программы, которые использовались для выполнения курсового проекта.

4. РЕАЛИЗАЦИЯ ПРОГРАММНОГО ИЗДЕЛИЯ НА ЯЗЫКЕ ПРОГРАМИРОВАНИЯ

Этот этап нужен для создания работающей программы на выбранном языке программирована, в ходе которого осуществляется тестировании и отладка продукта. здесь записаны ошибки и трудности, с которыми я столкнулся и решения этих трудностей

Была разработана программа «Кубик Рубика» . Исходный код которого содержится в ПРИЛОЖЕНИИ А. так же «Аналог роботка» в ПРИЛОЖЕНИИ Б, а Git в ПРИЛОЖЕНИИ Е. При разработки программного продукта сталкивался с некоторыми проблемами и трудностями.

Ошибки

1. Из-за того, что язык для меня был нов, я часто ошибался в синтаксисе языка
2. Из-за того, что язык для меня был нов, а также использование OpenGl графики, мне пришлось переписывать многие аспекты программы на «ходу» (выполнять частый рефакторинг кода)
3. Из-за того, что язык для меня был нов, я не осознавал некоторые возможности языка (частый рефакториг)

Трудности

1. Трудности возникали на стадии обучения OpenGl и реализации графики.
2. Из-за некоторых особенностей OpenGl возникали «тупиковые» ситуации

На этом этапе я разрабатывал код продукта, также я указал свои ошибки при написании кода, а также ошибки.

4. ТЕСТИРОВАНИЕ ПРОГРАМНОГО ПРОДУКТА

Этот этап нужен для выявления степени соответствия готового программного продукта спецификации, разработанной на первом этапе, с помощи. Программы и методики испытаний, разработанной на втором, здесь приведена таблица тестирования программного продукта

Таблица с функциями и их готовностью

Я сделал тестировании своего программного продукта по всем функциям, которые были написаны на предыдущих этапах, когда я их проверит они были завершены удачно.

III. ЗАКЛЮЧЕНИЕ

Мною была разработана программа «кубик Рубика», считаю что все задачи поставленные передо мною были выполнены. В ходе выполнения курсового проекта я научится основам языка программирования C#, а также основам OpenGl графики.

Доклад (ПРИЛОЖЕНИЕ)

Презентация (ПРИЛОЖЕНИЕ)

Пользовательская документация (ПРИЛОЖЕНИЕ)

Я не собираюсь бросать разработку этого программного продукта. В будущем я постараюсь добавить более практичный и красивый интерфейс, переделаю систему взаимодействия пользователя с приложением, добавлю систему настроек приложения, и многое другое.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

Книга © Игорь Тарасов http://Opengl.ogr.ru/ 1 августа 1999 года

Сайт https://Esate.ru/uroki/Opengl /

Статья http://habrahabr.ru/post/41514/ (Разработка Документации)

Сайт https://msdn.microsoft.com/ru-ru/ (обучение C#)

Приложение А

Код программы

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO;


using Tao.OpenGl;
using Tao.FreeGlut;
using Tao.Platform.Windows;

using System.Xml;



namespace Tao_OpenGl_inicialised11
{

   /// <exclude />
   public partial class Form1 : Form
   {
      /// <summary>
```

```
/// Class RecType  - Класс используется для работы с записанным временем
/// </summary>
public class RecType
{
  /// <summary>
  /// Initializes a new instance of the <see cref="RecType"/> class.
  /// </summary>
  /// <param name="_Name">Имя игрока</param>
  /// <param name="_RecTime">Класс типа Time в котором содержится время  time.</param>
  public RecType(String _Name, Time _RecTime)
  {
    Name = _Name;
    RecMin = _RecTime.MinuteTime;
    RecSec = _RecTime.SecondsTime;
    RecMilleSec = _RecTime.MilleSecondsTime;


  }
  /// <summary>
  /// Initializes a new instance of the <see cref="RecType"/> class.
  /// </summary>
  /// <param name="_Name">Имя игрока </param>
  /// <param name="_RecMin">Минуты</param>
  /// <param name="_RecSec">Секунды</param>
  /// <param name="_RecMilleSec">Милли секунды </param>
  public RecType(String _Name, int _RecMin, int _RecSec, int _RecMilleSec)
  {
    Name = _Name;
    RecMin = _RecMin;
    RecSec = _RecSec;
```

```
    RecMilleSec = _RecMilleSec;

}
/// <summary>
/// Initializes a new instance of the <see cref="RecType"/> class.
/// </summary>
public RecType()
{
    Name = "No name";
    RecMin = 0;
    RecSec = 0;
    RecMilleSec = 0;
}

/// <summary>
/// поля с именем
/// </summary>
public String Name;
/// <summary>
/// поля с минутами
/// </summary>
public int RecMin;
/// <summary>
/// поле с секундами
/// </summary>
public int RecSec;
/// <summary>
/// поле с милли секундами
/// </summary>
public int RecMilleSec;
```

```
}

/// <summary>
/// Класс для описания кубика в общем кубе
/// </summary>
public class ClassCube
{
    /// <summary>
    /// Псевдо указатель элемент общего куба
    /// </summary>
    /// <value>The name of the int.</value>
    public int IntName { get; set; }
    /// <summary>
    /// Угол поворота по X оси
    /// </summary>
    /// <value>The rotation angle x.</value>
    public double RotationAngleX { get; set; }
    /// <summary>
    /// Угол поворота по Y оси  .
    /// </summary>
    /// <value>The rotation angle y.</value>
    public double RotationAngleY { get; set; }
    /// <summary>
    /// Угол поворота по Z оси
    /// </summary>
    /// <value>The rotation angle z.</value>
    public double RotationAngleZ { get; set; }
    /// <summary>
    /// Прозрачность куба
    /// </summary>
    /// <value>The color of the alpha.</value>
```

```
        public float AlphaColor { get; set; }
        /// <summary>
        /// массив Векторов для позиции
        /// </summary>
        /// <value>The vertex.</value>
        public float[] Vertex { get; set; }
        /// <summary>
        /// Массив цветов куба
        /// </summary>
        /// <value>The color cube.</value>
        public float[][] ColorCube { get; set; }

        /// <summary>
        /// Initializes a new instance of the <see cref="ClassCube"/> class.
        /// </summary>
        /// <param name="_IntName">Псевдо имя(указатель) куба.</param>
        /// <param name="AngleX">The angle x.</param>
        /// <param name="AngleY">The angle y.</param>
        /// <param name="AngleZ">The angle z.</param>
        /// <param name="_AlphaColor">Color of the _ alpha.</param>
        public ClassCube(int _IntName, float AngleX, float AngleY, float AngleZ, float _AlphaColor)
        {
            IntName = _IntName;
            RotationAngleX = AngleX;
            RotationAngleY = AngleY;
            RotationAngleZ = AngleZ;
            AlphaColor = _AlphaColor;
            Vertex = new float[4];
            ColorCube = new float[7][];
        }
```

```
/// <summary>
/// Initializes a new instance of the <see cref="ClassCube"/> class.
/// </summary>
public ClassCube()
{

}
/// <summary>
/// Указывает угол поворота по 3 Осям
/// </summary>
/// <param name="AngleX">The angle x.</param>
/// <param name="AngleY">The angle y.</param>
/// <param name="AngleZ">The angle z.</param>
public void SetRotation(float AngleX, float AngleY, float AngleZ)
{
   RotationAngleX = AngleX;
   RotationAngleY = AngleY;
   RotationAngleZ = AngleZ;
}
/// <summary>
/// Инициальизирование (задание параметров) куба
/// </summary>
/// <param name="X">Вектор X .</param>
/// <param name="Y">Вектор Y.</param>
/// <param name="Z">Вектор Z.</param>
/// <param name="Color1">The color1.</param>
/// <param name="Color2">The color2.</param>
/// <param name="Color3">The color3.</param>
/// <param name="Color4">The color4.</param>
/// <param name="Color5">The color5.</param>
/// <param name="Color6">The color6.</param>
```

```
        /// <param name="_AlphaColor">Прозрачность.</param>
        public void SetObjectCube(float X, float Y, float Z, float[] Color1, float[] Color2, float[] Color3, float[] Color4, float[] Color5, float[] Color6, float _AlphaColor)
        {
            Vertex[1] = X;
            Vertex[2] = Y;
            Vertex[3] = Z;

            ColorCube[1] = Color1;
            ColorCube[2] = Color2;
            ColorCube[3] = Color3;
            ColorCube[4] = Color4;
            ColorCube[5] = Color5;
            ColorCube[6] = Color6;
            AlphaColor = _AlphaColor;

            RotationAngleX = 0;
            RotationAngleY = 0;
            RotationAngleZ = 0;
        }
        /// <summary>
        /// Рисование куба
        /// </summary>
        public void WriteCube()
        {
            Gl.glEnable(Gl.GL_LINE_STIPPLE);

            Gl.glBegin(Gl.GL_QUADS); //front

            Gl.glColor4f(ColorCube[1][0], ColorCube[1][1], ColorCube[1][2], AlphaColor);
```

```
Gl.glVertex3d(0, 0, 0);
Gl.glVertex3d(1, 0, 0);
Gl.glVertex3d(1, 1, 0);
Gl.glVertex3d(0, 1, 0);
Gl.glEnd();

Gl.glBegin(Gl.GL_QUADS); //top
Gl.glColor4f(ColorCube[4][0], ColorCube[4][1], ColorCube[4][2], AlphaColor);
Gl.glVertex3d(0, 1, 0);
Gl.glVertex3d(0, 1, 1);
Gl.glVertex3d(1, 1, 1);
Gl.glVertex3d(1, 1, 0);
Gl.glEnd();

Gl.glBegin(Gl.GL_QUADS); // Right
Gl.glColor4f(ColorCube[2][0], ColorCube[2][1], ColorCube[2][2], AlphaColor);
Gl.glVertex3d(1, 0, 0);
Gl.glVertex3d(1, 0, 1);
Gl.glVertex3d(1, 1, 1);
Gl.glVertex3d(1, 1, 0);
Gl.glEnd();

Gl.glBegin(Gl.GL_QUADS); // down
Gl.glColor4f(ColorCube[5][0], ColorCube[5][1], ColorCube[5][2], AlphaColor);
Gl.glVertex3d(0, 0, 0);
Gl.glVertex3d(1, 0, 0);
Gl.glVertex3d(1, 0, 1);
Gl.glVertex3d(0, 0, 1);
Gl.glEnd();
```

```
            Gl.glBegin(Gl.GL_QUADS);//left
            Gl.glColor4f(ColorCube[3][0], ColorCube[3][1], ColorCube[3][2],
AlphaColor);
            Gl.glVertex3d(0, 0, 0);
            Gl.glVertex3d(0, 1, 0);
            Gl.glVertex3d(0, 1, 1);
            Gl.glVertex3d(0, 0, 1);
            Gl.glEnd();

            Gl.glBegin(Gl.GL_QUADS); //bot
            Gl.glColor4f(ColorCube[6][0], ColorCube[6][1], ColorCube[6][2],
AlphaColor);
            Gl.glVertex3d(0, 0, 1);
            Gl.glVertex3d(0, 1, 1);
            Gl.glVertex3d(1, 1, 1);
            Gl.glVertex3d(1, 0, 1);
            Gl.glEnd();

            Gl.glDisable(Gl.GL_LINE_STIPPLE);
        } //рисование одного кубика
        /// <summary>
        /// Рисование граней куба (Обводки)
        /// </summary>
        public void WriteSoidCub()
        {
            Gl.glPolygonMode(Gl.GL_FRONT_AND_BACK, Gl.GL_LINE);
            Gl.glLineWidth(7);
            Gl.glColor3f(0, 0, 0);
            Gl.glBegin(Gl.GL_QUADS); //front
            Gl.glVertex3d(0, 0, 0);
            Gl.glVertex3d(1, 0, 0);
```

```
Gl.glVertex3d(1, 1, 0);
Gl.glVertex3d(0, 1, 0);
Gl.glEnd();


Gl.glBegin(Gl.GL_QUADS); //top
Gl.glVertex3d(0, 1, 0);
Gl.glVertex3d(0, 1, 1);
Gl.glVertex3d(1, 1, 1);
Gl.glVertex3d(1, 1, 0);
Gl.glEnd();


Gl.glBegin(Gl.GL_QUADS); // Right
Gl.glVertex3d(1, 0, 0);
Gl.glVertex3d(1, 0, 1);
Gl.glVertex3d(1, 1, 1);
Gl.glVertex3d(1, 1, 0);
Gl.glEnd();


Gl.glBegin(Gl.GL_QUADS); // down
Gl.glVertex3d(0, 0, 0);
Gl.glVertex3d(1, 0, 0);
Gl.glVertex3d(1, 0, 1);
Gl.glVertex3d(0, 0, 1);
Gl.glEnd();


Gl.glBegin(Gl.GL_QUADS);//left
Gl.glVertex3d(0, 0, 0);
```

```
            Gl.glVertex3d(0, 1, 0);
            Gl.glVertex3d(0, 1, 1);
            Gl.glVertex3d(0, 0, 1);
            Gl.glEnd();



            Gl.glBegin(Gl.GL_QUADS); //bot
            Gl.glVertex3d(0, 0, 1);
            Gl.glVertex3d(0, 1, 1);
            Gl.glVertex3d(1, 1, 1);
            Gl.glVertex3d(1, 0, 1);
            Gl.glEnd();
            Gl.glPolygonMode(Gl.GL_FRONT_AND_BACK, Gl.GL_FILL);




        } // рисоване граней кубика
        /// <summary>
        /// Завершающее рисование куба (Объеденяет void WriteCube и void WriteSoidCub )
        /// </summary>
        public void DrawCube()
        {
            WriteCube();
            WriteSoidCub();
        }
    }
    /// <summary>
    /// Класс дял работы с камерой.
    /// </summary>
    public class CameraClass
    {
```

```
    float AngleX;
    float AngleY;

    /// <summary>
    /// Initializes a new instance of the <see cref="CameraClass"/> class.
    /// </summary>
    /// <param name="RotationAngleX">The rotation angle x.</param>
    /// <param name="RotationAngleY">The rotation angle y.</param>
    public CameraClass(float RotationAngleX, float RotationAngleY)
    {
        AngleX = RotationAngleX;
        AngleY = RotationAngleY;
    }

}
/// <summary>
/// Класс для работы со временем .
/// </summary>
public class Time  // структура времени
{

    /// <summary>
    /// Initializes a new instance of the <see cref="Time"/> class.
    /// </summary>
    public Time()
    {
        MilleSecondsTime = 0;
        SecondsTime = 0;
        MinuteTime = 0;
    }
    /// <summary>
```

```
        /// Gets or sets the minute time.
        /// </summary>
        /// <value>The minute time.</value>
        public int MinuteTime { get; set; }
        /// <summary>
        /// Gets or sets the seconds time.
        /// </summary>
        /// <value>The seconds time.</value>
        public int SecondsTime { get; set; }
        /// <summary>
        /// Gets or sets the mille seconds time.
        /// </summary>
        /// <value>The mille seconds time.</value>
        public int MilleSecondsTime { get; set; }

    }

    #region Create Vars

    string NameDataFolder ="Data";
    string NameRecordFolder ="Record";
    string NameRecFile = "Records.Xml";
    string PatchExeFile = Directory.GetCurrentDirectory();
    string NameSettingFolder = "Setting";
    string NameSettigFile = "Setting.Xml";
    float DeltaAngleForAllCube = 45; // угол поворота кубика (сейчас не используется )
    float DeltaAngleForCubeRotation = 90; // угол поворота раней кубика
    float SelectAlphaColor = 0.6f; //Алтфа цвет выделелния
    float NormalAlphaColor = 1.0f; // альфа нормального цвета
    bool StartGame = false; // начата ли игра
    bool PauseGame = false;  //  стоил ти игра на пайзе
```

```
bool EndGame = false; // если кубик собран
double TempX;
double TempY;
double OldMousePosX;
double OldMousePosY;
int MouseStartPositionX = 0;
int MouseStartPositionY = 0;
double MouseDeltaX = 0;
double MouseDeltaY = 0;
int SizeCube = 10; // начальный размер куба
Color ColorWorld; //Цвет мира и интерфейса

/// <summary>
/// Массив кубов
/// </summary>
public ClassCube[, ,] Cube = new ClassCube[3, 3, 3];
/// <summary>
/// Массив записей рекордов
/// </summary>
public RecType[] Records = new RecType[10];
Time TimeForRecords = new Time();
struct SSelectCube
{
   public int X;
   public int Y;
   public int Z;
};
SSelectCube SelectedCube;

#endregion
```

```
int[][] VariationOFSelectedCybeMidle = { // хранилище возможных вборов середин для их переворотов * не используется
    new int[] {2,2,1}, //front
    new int[] {2,3,1}, //top
    new int[] {3,2,2}, //right
    new int[] {1,2,2}, //left
    new int[] {2,1,2}, //down
    new int[] {2,2,3}  //bot
};

float[][] COlorHande =   { // укаатель цветов
    new float [] {1,0,0}, //красный 0
    new float [] {0,1,0}, // зеленый 1
    new float [] {0,0,1} ,// синий 2
    new float [] {1,1,1}, //Белый 3
    new float [] {1,1,0} , //желтый 4
    new float [] {1,0.5f,0} ,  //оранджеый 5
    new float [] {0,0,0}  , //черный 6
  };
/// <summary>
/// Initializes a new instance of the <see cref="Form1"/> class.
/// </summary>
public Form1() // первичная инициализация
{
    InitializeComponent();
    OpenGlControl.InitializeContexts();
}
/// <summary>
///  Настройка OpenGl для дальнейшей работы
///  Проверка файловой системы
///  запуск дополнительных процедур для настройки
/// </summary>
```

```
        /// <param name="sender">The source of the event.</param>
        /// <param name="e">The <see cref="EventArgs"/> instance containing the event data.</param>
        public void Form1_Load(object sender, EventArgs e) // инициальзация
        {
            Glut.glutInit();
            Glut.glutInitDisplayMode(Glut.GLUT_RGB | Glut.GLUT_DOUBLE | Glut.GLUT_DEPTH);
            Gl.glClearColor(0, 0, 1, 1);
            Gl.glViewport(5, 5, OpenGlControl.Width, OpenGlControl.Height);
            Gl.glMatrixMode(Gl.GL_PROJECTION);
            Gl.glLoadIdentity();
            Glu.gluPerspective(45, (float)OpenGlControl.Width / (float)OpenGlControl.Height, 0.1, 200);
            Gl.glMatrixMode(Gl.GL_MODELVIEW);
            Gl.glLoadIdentity();
            Gl.glEnable(Gl.GL_DEPTH_TEST);
            Gl.glEnable(Gl.GL_LINE_SMOOTH);
            Gl.glEnable(Gl.GL_BLEND);
            Gl.glBlendFunc(Gl.GL_SRC_ALPHA, Gl.GL_ONE_MINUS_SRC_ALPHA);
            //Gl.glEnable(Gl.GL_LIGHTING);
            // Gl.glEnable(Gl.GL_LIGHT1
            if (!Directory.Exists(PatchExeFile + "//" + NameDataFolder + "//" + NameSettingFolder))
            {
                Directory.CreateDirectory(PatchExeFile + "//" + NameDataFolder + "//" + NameSettingFolder);
            }
            if (!Directory.Exists(PatchExeFile + "//" +  NameDataFolder + "//" + NameRecordFolder))
            {
                Directory.CreateDirectory(PatchExeFile + "//" + NameDataFolder + "//" + NameRecordFolder);
```

```
        }
        SecondSettingUp();
        RenderTimer.Enabled = true;
    }
    /// <summary>
    /// Дополнительная настройка
    ///     задает многие параметры,создает массив Общего куба
    /// </summary>

    public void SecondSettingUp()
    {
        CameraClass Camera = new CameraClass(0, 0);
        SelectedCube.X = 2;
        SelectedCube.Y = 2;
        SelectedCube.Z = 1;
        Gl.glClearColor(0, 0.5f, 1, 1);
        PanelMenu.Left = OpenGlControl.Width + 10;
        //for (int j = 0 ; j<=9;j++)

        //{
        //    Records[j] = new RecType("Name_"+ j.ToString() , j,20,20);
        //}
       // SaveRecordFileXml();
        LoadRecordFileXml();

        int SelectIntName = 1;
        for (int Z = 0; Z < 3; Z++)
            for (int X = 0; X < 3; X++)
                for (int Y = 0; Y < 3; Y++)
                {
                    Cube[X, Y, Z] = new ClassCube(SelectIntName, 0, 0, 0,
NormalAlphaColor);
```

```
            SelectIntName += 1;
        }

        Cube[0, 0, 0].SetObjectCube(-1, -1, -1, COlorHande[2], COlorHande[6], COlorHande[0], COlorHande[6], COlorHande[3], COlorHande[6], Cube[0, 0, 0].AlphaColor);

        Cube[0, 1, 0].SetObjectCube(-1, 0, -1, COlorHande[2], COlorHande[6], COlorHande[0], COlorHande[6], COlorHande[6], COlorHande[6], Cube[0, 1, 0].AlphaColor);

        Cube[0, 2, 0].SetObjectCube(-1, 1, -1, COlorHande[2], COlorHande[6], COlorHande[0], COlorHande[4], COlorHande[6], COlorHande[6], Cube[0, 2, 0].AlphaColor);


        Cube[1, 0, 0].SetObjectCube(0, -1, -1, COlorHande[2], COlorHande[6], COlorHande[6], COlorHande[6], COlorHande[3], COlorHande[6], Cube[1, 0, 0].AlphaColor);

        Cube[1, 1, 0].SetObjectCube(0, 0, -1, COlorHande[2], COlorHande[6], COlorHande[6], COlorHande[6], COlorHande[6], COlorHande[6], Cube[1, 1, 0].AlphaColor);

        Cube[1, 2, 0].SetObjectCube(0, 1, -1, COlorHande[2], COlorHande[6], COlorHande[6], COlorHande[4], COlorHande[6], COlorHande[6], Cube[1, 2, 0].AlphaColor);


        Cube[2, 0, 0].SetObjectCube(1, -1, -1, COlorHande[2], COlorHande[5], COlorHande[6], COlorHande[6], COlorHande[3], COlorHande[6], Cube[2, 0, 0].AlphaColor);

        Cube[2, 1, 0].SetObjectCube(1, 0, -1, COlorHande[2], COlorHande[5], COlorHande[6], COlorHande[6], COlorHande[6], COlorHande[6], Cube[2, 1, 0].AlphaColor);

        Cube[2, 2, 0].SetObjectCube(1, 1, -1, COlorHande[2], COlorHande[5], COlorHande[6], COlorHande[4], COlorHande[6], COlorHande[6], Cube[2, 2, 0].AlphaColor);

        //

        Cube[0, 0, 1].SetObjectCube(-1, -1, 0, COlorHande[6], COlorHande[6], COlorHande[0], COlorHande[6], COlorHande[3], COlorHande[6], Cube[0, 0, 1].AlphaColor);

        Cube[0, 1, 1].SetObjectCube(-1, 0, 0, COlorHande[6], COlorHande[6], COlorHande[0], COlorHande[6], COlorHande[6], COlorHande[6], Cube[0, 1, 1].AlphaColor);
```

```
Cube[0, 2, 1].SetObjectCube(-1, 1, 0, COlorHande[6], COlorHande[6], COlorHande[0], COlorHande[4], COlorHande[6], COlorHande[6], Cube[0, 2, 1].AlphaColor);

Cube[1, 0, 1].SetObjectCube(0, -1, 0, COlorHande[6], COlorHande[6], COlorHande[6], COlorHande[6], COlorHande[3], COlorHande[6], Cube[1, 0, 1].AlphaColor);

Cube[1, 1, 1].SetObjectCube(0, 0, 0, COlorHande[6], COlorHande[6], COlorHande[6], COlorHande[6], COlorHande[6], COlorHande[6], Cube[1, 1, 1].AlphaColor);

Cube[1, 2, 1].SetObjectCube(0, 1, 0, COlorHande[6], COlorHande[6], COlorHande[6], COlorHande[4], COlorHande[6], COlorHande[6], Cube[1, 2, 1].AlphaColor);

Cube[2, 0, 1].SetObjectCube(1, -1, 0, COlorHande[6], COlorHande[5], COlorHande[6], COlorHande[6], COlorHande[3], COlorHande[6], Cube[2, 0, 1].AlphaColor);

Cube[2, 1, 1].SetObjectCube(1, 0, 0, COlorHande[6], COlorHande[5], COlorHande[6], COlorHande[6], COlorHande[6], COlorHande[6], Cube[2, 1, 1].AlphaColor);

Cube[2, 2, 1].SetObjectCube(1, 1, 0, COlorHande[6], COlorHande[5], COlorHande[6], COlorHande[4], COlorHande[6], COlorHande[6], Cube[2, 2, 1].AlphaColor);

Cube[0, 0, 2].SetObjectCube(-1, -1, 1, COlorHande[6], COlorHande[6], COlorHande[0], COlorHande[6], COlorHande[3], COlorHande[1], Cube[0, 0, 2].AlphaColor);

Cube[0, 1, 2].SetObjectCube(-1, 0, 1, COlorHande[6], COlorHande[6], COlorHande[0], COlorHande[6], COlorHande[6], COlorHande[1], Cube[0, 1, 2].AlphaColor);

Cube[0, 2, 2].SetObjectCube(-1, 1, 1, COlorHande[6], COlorHande[6], COlorHande[0], COlorHande[4], COlorHande[6], COlorHande[1], Cube[0, 2, 2].AlphaColor);

Cube[1, 0, 2].SetObjectCube(0, -1, 1, COlorHande[6], COlorHande[6], COlorHande[6], COlorHande[6], COlorHande[3], COlorHande[1], Cube[1, 0, 2].AlphaColor);

Cube[1, 1, 2].SetObjectCube(0, 0, 1, COlorHande[6], COlorHande[6], COlorHande[6], COlorHande[6], COlorHande[6], COlorHande[1], Cube[1, 1, 2].AlphaColor);
```

```
        Cube[1, 2, 2].SetObjectCube(0, 1, 1, COlorHande[6], COlorHande[6], COlorHande[6], COlorHande[4], COlorHande[6], COlorHande[1], Cube[1, 2, 2].AlphaColor);


        Cube[2, 0, 2].SetObjectCube(1, -1, 1, COlorHande[6], COlorHande[5], COlorHande[6], COlorHande[6], COlorHande[3], COlorHande[1], Cube[2, 0, 2].AlphaColor);
        Cube[2, 1, 2].SetObjectCube(1, 0, 1, COlorHande[6], COlorHande[5], COlorHande[6], COlorHande[6], COlorHande[6], COlorHande[1], Cube[2, 1, 2].AlphaColor);
        Cube[2, 2, 2].SetObjectCube(1, 1, 1, COlorHande[6], COlorHande[5], COlorHande[6], COlorHande[4], COlorHande[6], COlorHande[1], Cube[2, 2, 2].AlphaColor);


        Cube[SelectedCube.X - 1, SelectedCube.Y - 1, SelectedCube.Z - 1].AlphaColor = SelectAlphaColor;
    }
    /// <summary>
    /// Рисовани кубика Рубика
    /// </summary>
    public void WriteCubeRub()
    {
        for (int Z = 0; Z < 3; Z++)
            for (int X = 0; X < 3; X++)
                for (int Y = 0; Y < 3; Y++)
                {
                    Gl.glPushMatrix();
                    Gl.glTranslatef(X - 1, Y - 1, Z - 1);
                    Gl.glTranslatef(0.5f, 0.5f, 0.5f);

                    //Gl.glMultMatrixd(
                    Gl.glRotated(Cube[X, Y, Z].RotationAngleX, 1, 0, 0);
                    Gl.glRotated(Cube[X, Y, Z].RotationAngleY, 0, 1, 0);
                    Gl.glRotated(Cube[X, Y, Z].RotationAngleZ, 0, 0, 1);
```

```
                Gl.glTranslatef(-0.5f, -0.5f, -0.5f);
                Cube[X, Y, Z].DrawCube();
                Gl.glPopMatrix();
            }
        } // рисование кубика-Рубика
        /// <summary>
        /// Процедура в которой происходят выховы функции перерисовки сцены
        /// </summary>
        /// <param name="sender">The source of the event.</param>
        /// <param name="e">The <see cref="EventArgs"/> instance containing the event data.</param>
        public void button1_Click(object sender, EventArgs e) // основная работа
        {
            Gl.glClear(Gl.GL_COLOR_BUFFER_BIT | Gl.GL_DEPTH_BUFFER_BIT);

            Gl.glLoadIdentity();
            Gl.glPushMatrix();
            Glu.gluLookAt(SizeCube, 1, 0, -10, 0, 0, 0, 1, 0);

            Gl.glTranslated(0.5f, 0.5f, 0.5f);
            Gl.glRotated(MouseDeltaX, 0, -1, 0);
            Gl.glRotated(MouseDeltaY, 1, 0, 0);
            Gl.glTranslated(-0.5f, -0.5f, -0.5f);

            WriteCubeRub();

            Gl.glPopMatrix();
            Gl.glFlush();
            OpenGlControl.Invalidate();
```

```
        }
        /// <summary>
        /// Animations the open menu.
        /// </summary>
        /// <param name="sender">The sender.</param>
        /// <param name="e">The <see cref="EventArgs"/> instance containing the event data.</param>
        public void AnimationOpenMenu(object sender, EventArgs e)
        {

            if (PanelMenu.Left >= OpenGlControl.Width - PanelMenu.Width)
            {
                PanelMenu.Left -= 10;
            }
            else
            {
                RenderAnimation.Enabled = false;
                RenderAnimation.Tick += EventHandler;


            }



        } // анимация для открывания "меню "
        /// <summary>
        /// Animations the close menu.
        /// </summary>
        /// <param name="sender">The sender.</param>
        /// <param name="e">The <see cref="EventArgs"/> instance containing the event data.</param>
        public void AnimationCloseMenu(object sender, EventArgs e)
        {
```

```
        if (PanelMenu.Location.X <= OpenGlControl.Width + PanelMenu.Width + 10)
        {

            PanelMenu.Left += 10;
            //RenderAnimation.Enabled = true;
        }
        else
        {
            RenderAnimation.Enabled = false;
            RenderAnimation.Tick += EventHandler;


        }
    }// анимация для закрывания "меню ;
    public void EventHandler(object sender, EventArgs e)
    {
        // throw new NotImplementedException();
    } // заглушка
    /// <summary>
    /// Функция вызывающаяся в таймере для перерисовки сцены
    /// </summary>
    /// <param name="sender">The source of the event.</param>
    /// <param name="e">The <see cref="EventArgs"/> instance containing the event data.</param>
    public void timer1_Tick(object sender, EventArgs e)
    {
        button1_Click(sender, e);
    } // вызов перерисовки
    public void button2_Click_1(object sender, EventArgs e)
    {
        RenderTimer.Enabled = !(RenderTimer.Enabled);
    }
```

```
public void label1_MouseEnter(object sender, EventArgs e)
{
   RenderAnimation.Tick += AnimationOpenMenu;
   MenuLabel.Visible = false;
   RenderAnimation.Enabled = true;
   if (StartGame == true)
   {
      MenuLabel.Visible = true;
      RecTimer.Enabled = false;
      PauseLabel.Visible = true;
      PauseGame = true;
   }
   PanelMenu.Focus();

}
public void panel1_Leave(object sender, EventArgs e)
{
   RenderAnimation.Tick += AnimationCloseMenu;
   MenuLabel.Visible = true;
   RenderAnimation.Enabled = true;
   RecordsPanel.Visible = false;
   if (PauseGame == true)
   {
      RecTimer.Enabled = true;
      PauseLabel.Visible = false;
      PauseGame = false;

   }

}
public void Exitbutton_Click(object sender, EventArgs e)
```

```
        {
            Application.Exit();
        }
        public void AnT_MouseMove(object sender, MouseEventArgs e)
        {

            switch (e.Button)
            {
                case MouseButtons.Left:
                    {



                        TempX = (OldMousePosX - e.Location.X);
                        TempY = (OldMousePosY - e.Location.Y);

                        OldMousePosX = e.Location.X;
                        OldMousePosY = e.Location.Y;
                        if ((TempX != 0) || (TempY != 0))
                        {
                            MouseDeltaX += 1.5 * TempX / Math.Sqrt(TempX *
TempX + TempY * TempY);
                            MouseDeltaY += 1.5 * TempY / Math.Sqrt(TempX *
TempX + TempY * TempY);
                        }



                        break;
                    }
                case MouseButtons.None:
                    {
```

```
                break;
            }


        }
    }
    public void AnT_MouseDown(object sender, MouseEventArgs e)
    {

        OldMousePosX = MouseStartPositionX;
        OldMousePosY = MouseStartPositionY;

        MouseStartPositionX = e.Location.X;
        MouseStartPositionY = e.Location.Y;
    }
    public void RecTimer_Tick(object sender, EventArgs e)// пофиксить синхрониацию
    {
        TimeForRecords.MilleSecondsTime += 1;
        if (TimeForRecords.MilleSecondsTime >= 10)
        {
            TimeForRecords.MilleSecondsTime = 0;
            TimeForRecords.SecondsTime += 1;
            if (TimeForRecords.SecondsTime >= 60)
            {
                TimeForRecords.SecondsTime = 0;
                TimeForRecords.MinuteTime += 1;
            }
        }
        TimeLabel.Text = Convert.ToString(TimeForRecords.MinuteTime + ":" + TimeForRecords.SecondsTime + ":" + TimeForRecords.MilleSecondsTime);
```

```
        }
        public void NewGameButton_Click(object sender, EventArgs e)
        {
            if (StartGame == true)
            {
                RecTimer.Enabled = false;
                switch (MessageBox.Show("Вы Действительнохотите начать заного ? ", "Внимание!", MessageBoxButtons.YesNo))
                {
                    case (System.Windows.Forms.DialogResult.Yes):
                        {
                            //процедура сбрасывания кубика
                            StartGame = true;
                            PauseGame = false;

                            TimeForRecords.MilleSecondsTime = 0;
                            TimeForRecords.SecondsTime = 0;
                            TimeForRecords.MinuteTime = 0;
                            RecTimer.Enabled = true;
                            break;

                        }
                    case (System.Windows.Forms.DialogResult.No):
                        {
                            StartGame = false;
                            TimeForRecords.MilleSecondsTime = 0;
                            TimeForRecords.SecondsTime = 0;
                            TimeForRecords.MinuteTime = 0;
                            RecTimer.Enabled = false;
                            TimeLabel.Text = Convert.ToString(TimeForRecords.MinuteTime + ":" + TimeForRecords.SecondsTime + ":" + TimeForRecords.MilleSecondsTime);
```

```
                TimeLabel.Visible = false;

                break;


            };
        };
    }

    else
    {
        // RenderAnimation.Enabled = true;
        // RenderAnimation.Tick += AnimationCloseMenu;
        SecondSettingUp();
        ShaflCube();
        TimeLabel.Visible = true;
        RecTimer.Enabled = true;
        StartGame = true;
    }
}
public void AnT_KeyDown(object sender, KeyEventArgs e)
{
    switch (e.KeyCode)
    {
        case Keys.W:
            {
                if (SelectedCube.Y == 2)
                {
                    {
                        if (SelectedCube.Z == 1)
                        {
```

```
                    SelectedCube.Y += 1;
                    SelectedCube.Z = 2;
                }
                else
                {
                    if (SelectedCube.Z == 3)
                    {
                        SelectedCube.Y -= 1;
                        SelectedCube.Z = 2;
                    }
                    else
                    {
                        if (SelectedCube.Z == 2)
                        {
                            if ((SelectedCube.X == 1) || (SelectedCube.X ==
3))
                            {
                                { SelectedCube.Y += 1; SelectedCube.Z = 2;
SelectedCube.X = 2; }
                            }

                        }
                    }
                }
            }
            else
            {
                if (SelectedCube.Y == 3)
                {
                    SelectedCube.Y -= 1;
                    SelectedCube.Z = 3;
```

```
            }
            else
            {
                if (SelectedCube.Y == 1)
                {
                    SelectedCube.Y += 1;
                    SelectedCube.Z = 1;
                }
            }
        }
        break;
    }
case Keys.S:
    {
        if (SelectedCube.Y == 2)
        {
            {
                if (SelectedCube.Z == 1)
                {
                    SelectedCube.Y -= 1;
                    SelectedCube.Z = 2;
                }
                else
                {
                    if (SelectedCube.Z == 3)
                    {
                        SelectedCube.Y += 1;
                        SelectedCube.Z = 2;
                    }
                    else
                    {
```

```
                    if (SelectedCube.Z == 2)
                    {
                        if ((SelectedCube.X == 1) || (SelectedCube.X ==
3))
                        {
                            SelectedCube.X = 2;
                            SelectedCube.Y -= 1;
                            SelectedCube.Z = 2;
                        }
                    }
                }
            }
        }
        else
        {
            if (SelectedCube.Y == 3)
            {
                SelectedCube.Y -= 1;
                SelectedCube.Z = 1;
            }
            else
            {
                if (SelectedCube.Y == 1)
                {
                    SelectedCube.Y += 1;
                    SelectedCube.Z = 3;
                }

            }
        }
        break;
```

```
                }

            case Keys.D:
                {
                    if (SelectedCube.X == 1)
                    {
                        SelectedCube.X += 1;
                        SelectedCube.Z = 3;
                    }
                    else
                    {
                        if (SelectedCube.X == 2)
                        {
                            if (SelectedCube.Z == 1)
                            {
                                SelectedCube.X -= 1;
                                SelectedCube.Z = 2;
                            }
                            else
                            {
                                if (SelectedCube.Z == 3)
                                {
                                    SelectedCube.X += 1;
                                    SelectedCube.Z = 2;
                                }
                                else
                                {
                                    if (SelectedCube.Z == 2)
                                    {
                                        if ((SelectedCube.Y == 3) || (SelectedCube.Y ==
1))
                                        {
```

```
                        SelectedCube.Y = 2;
                        SelectedCube.Z = 2;
                        SelectedCube.X -= 1;
                    }
                }
            }

        }
    }
    else
    {
        if (SelectedCube.X == 3)
        {
            SelectedCube.X -= 1;
            SelectedCube.Z = 1;
        }
    }
}

    break;
}
case Keys.A:
{
    if (SelectedCube.X == 1)
    {
        SelectedCube.X += 1;
        SelectedCube.Z = 1;
    }
    else
    {
        if (SelectedCube.X == 2)
```

```
                {
                    if (SelectedCube.Z == 1)
                    {
                        SelectedCube.X += 1;
                        SelectedCube.Z = 2;
                    }
                    else
                    {
                        if (SelectedCube.Z == 3)
                        {
                            SelectedCube.X -= 1;
                            SelectedCube.Z = 2;
                        }
                        else
                        {
                            if (SelectedCube.Z == 2)
                            {
                                if ((SelectedCube.Y == 1) || (SelectedCube.Y ==
3))
                                {
                                    SelectedCube.Y = 2;
                                    SelectedCube.Z = 2;
                                    SelectedCube.X += 1;
                                }
                            }
                        }
                    }
                }
                else
                {
                    if (SelectedCube.X == 3)
                    {
```

```
                    SelectedCube.X -= 1;
                    SelectedCube.Z = 3;
                }
            }
        }
        break;
    }
case Keys.Space: // переворот
    {
        FlipSide(SelectedCube.X, SelectedCube.Y, SelectedCube.Z);

        int SelectIntName = 1;
        for (int Z = 0; Z < 3; Z++)
            for (int X = 0; X < 3; X++)
                for (int Y = 0; Y < 3; Y++)
                {
                    if (Cube[X, Y, Z].IntName == SelectIntName)
                    {
                        SelectIntName += 1;
                        EndGame = true;
                    }
                    else
                    {
                        EndGame = false;
                        break;
                    }
                    if ((EndGame == true) && (StartGame == true))
                    {
                        GameIsOver();
                    }
                }
```

```
                break;
            }
        }
        Cube[1, 1, 0].AlphaColor = NormalAlphaColor;
        Cube[1, 2, 1].AlphaColor = NormalAlphaColor;
        Cube[2, 1, 1].AlphaColor = NormalAlphaColor;
        Cube[0, 1, 1].AlphaColor = NormalAlphaColor;
        Cube[1, 0, 1].AlphaColor = NormalAlphaColor;
        Cube[1, 1, 2].AlphaColor = NormalAlphaColor;
        Cube[SelectedCube.X - 1, SelectedCube.Y - 1, SelectedCube.Z - 1].AlphaColor = SelectAlphaColor;
    }
    /// <summary>
    /// Переворот Грани
    /// </summary>
    /// <param name="X">Середина грани по X.</param>
    /// <param name="Y">Середина грани по Y.</param>
    /// <param name="Z">Середина грани по Z.</param>
    public void FlipSide(int X, int Y, int Z)
    {
        if ((X == 2) && (Y == 2) && (Z == 1))//1,1,0
        {

            Cube[0, 0, 0].RotationAngleZ -= 90;
            Cube[0, 2, 0].RotationAngleZ -= 90;
            Cube[2, 2, 0].RotationAngleZ -= 90;
            Cube[2, 0, 0].RotationAngleZ -= 90;

            Cube[0, 1, 0].RotationAngleZ -= 90;
            Cube[1, 2, 0].RotationAngleZ -= 90;
            Cube[2, 1, 0].RotationAngleZ -= 90;
            Cube[1, 0, 0].RotationAngleZ -= 90;
```

```
    Cube[1, 1, 0].RotationAngleZ -= 90;
    SwapArrCube(ref Cube[0, 0, 0], ref Cube[0, 2, 0]);
    SwapArrCube(ref Cube[0, 0, 0], ref Cube[2, 2, 0]);
    SwapArrCube(ref Cube[0, 0, 0], ref Cube[2, 0, 0]);

    SwapArrCube(ref Cube[0, 1, 0], ref Cube[1, 2, 0]);
    SwapArrCube(ref Cube[0, 1, 0], ref Cube[2, 1, 0]);
    SwapArrCube(ref Cube[0, 1, 0], ref Cube[1, 0, 0]);
}
else
{
    if ((X == 2) && (Y == 3) && (Z == 2))//1,2,1
    {

        Cube[0, 2, 0].RotationAngleY += 90;
        Cube[0, 2, 2].RotationAngleY += 90;
        Cube[2, 2, 2].RotationAngleY += 90;
        Cube[2, 2, 0].RotationAngleY += 90;

        Cube[0, 2, 1].RotationAngleY += 90;
        Cube[1, 2, 2].RotationAngleY += 90;
        Cube[2, 2, 1].RotationAngleY += 90;
        Cube[1, 2, 0].RotationAngleY += 90;

        Cube[1, 2, 1].RotationAngleY += 90;

        SwapArrCube(ref Cube[0, 2, 0], ref Cube[0, 2, 2]);
        SwapArrCube(ref Cube[0, 2, 0], ref Cube[2, 2, 2]);
        SwapArrCube(ref Cube[0, 2, 0], ref Cube[2, 2, 0]);
```

```
    SwapArrCube(ref Cube[0, 2, 1], ref Cube[1, 2, 2]);
    SwapArrCube(ref Cube[0, 2, 1], ref Cube[2, 2, 1]);
    SwapArrCube(ref Cube[0, 2, 1], ref Cube[1, 2, 0]);

}
else
{
    if ((X == 3) && (Y == 2) && (Z == 2))//2,1,1
    {
        Cube[2, 0, 0].RotationAngleX += 90;
        Cube[2, 2, 0].RotationAngleX += 90;
        Cube[2, 2, 2].RotationAngleX += 90;
        Cube[2, 0, 2].RotationAngleX += 90;

        Cube[2, 1, 0].RotationAngleX += 90;
        Cube[2, 2, 1].RotationAngleX += 90;
        Cube[2, 1, 2].RotationAngleX += 90;
        Cube[2, 0, 1].RotationAngleX += 90;
        Cube[2, 1, 1].RotationAngleX += 90;

        SwapArrCube(ref Cube[2, 0, 0], ref Cube[2, 2, 0]);
        SwapArrCube(ref Cube[2, 0, 0], ref Cube[2, 2, 2]);
        SwapArrCube(ref Cube[2, 0, 0], ref Cube[2, 0, 2]);

        SwapArrCube(ref Cube[2, 1, 0], ref Cube[2, 2, 1]);
        SwapArrCube(ref Cube[2, 1, 0], ref Cube[2, 1, 2]);
        SwapArrCube(ref Cube[2, 1, 0], ref Cube[2, 0, 1]);


    }
    else
```

```
{
    if ((X == 1) && (Y == 2) && (Z == 2))//0,1,1
    {

        Cube[0, 0, 0].RotationAngleX += 90;
        Cube[0, 2, 0].RotationAngleX += 90;
        Cube[0, 2, 2].RotationAngleX += 90;
        Cube[0, 0, 2].RotationAngleX += 90;

        Cube[0, 1, 0].RotationAngleX += 90;
        Cube[0, 2, 1].RotationAngleX += 90;
        Cube[0, 1, 2].RotationAngleX += 90;
        Cube[0, 0, 1].RotationAngleX += 90;

        Cube[0, 1, 1].RotationAngleX += 90;

        SwapArrCube(ref Cube[0, 0, 0], ref Cube[0, 2, 0]);
        SwapArrCube(ref Cube[0, 0, 0], ref Cube[0, 2, 2]);
        SwapArrCube(ref Cube[0, 0, 0], ref Cube[0, 0, 2]);

        SwapArrCube(ref Cube[0, 1, 0], ref Cube[0, 2, 1]);
        SwapArrCube(ref Cube[0, 1, 0], ref Cube[0, 1, 2]);
        SwapArrCube(ref Cube[0, 1, 0], ref Cube[0, 0, 1]);



    }
    else
    {
        if ((X == 2) && (Y == 1) && (Z == 2))//1,0,1
        {
            Cube[0, 0, 0].RotationAngleY -= 90;
```

```
            Cube[2, 0, 0].RotationAngleY -= 90;
            Cube[2, 0, 2].RotationAngleY -= 90;
            Cube[0, 0, 2].RotationAngleY -= 90;

            Cube[1, 0, 0].RotationAngleY -= 90;
            Cube[2, 0, 1].RotationAngleY -= 90;
            Cube[1, 0, 2].RotationAngleY -= 90;
            Cube[0, 0, 1].RotationAngleY -= 90;

            Cube[1, 0, 1].RotationAngleY += 90;

            SwapArrCube(ref Cube[0, 0, 0], ref Cube[2, 0, 0]);
            SwapArrCube(ref Cube[0, 0, 0], ref Cube[2, 0, 2]);
            SwapArrCube(ref Cube[0, 0, 0], ref Cube[0, 0, 2]);

            SwapArrCube(ref Cube[1, 0, 0], ref Cube[2, 0, 1]);
            SwapArrCube(ref Cube[1, 0, 0], ref Cube[1, 0, 2]);
            SwapArrCube(ref Cube[1, 0, 0], ref Cube[0, 0, 1]);

        }
        else
        {
            if ((X == 2) && (Y == 2) && (Z == 3))//1,1,2
            {

                Cube[0, 0, 2].RotationAngleZ -= 90;
                Cube[0, 2, 2].RotationAngleZ -= 90;
                Cube[2, 2, 2].RotationAngleZ -= 90;
                Cube[2, 0, 2].RotationAngleZ -= 90;

                Cube[0, 1, 2].RotationAngleZ -= 90;
```

```
                    Cube[1, 2, 2].RotationAngleZ -= 90;
                    Cube[2, 1, 2].RotationAngleZ -= 90;
                    Cube[1, 0, 2].RotationAngleZ -= 90;

                    Cube[1, 1, 2].RotationAngleZ -= 90;

                    SwapArrCube(ref Cube[0, 0, 2], ref Cube[0, 2, 2]);
                    SwapArrCube(ref Cube[0, 0, 2], ref Cube[2, 2, 2]);
                    SwapArrCube(ref Cube[0, 0, 2], ref Cube[2, 0, 2]);

                    SwapArrCube(ref Cube[0, 1, 2], ref Cube[1, 2, 2]);
                    SwapArrCube(ref Cube[0, 1, 2], ref Cube[2, 1, 2]);
                    SwapArrCube(ref Cube[0, 1, 2], ref Cube[1, 0, 2]);
                }
            }
        }
    }
  }

  //afqk nen

}

/// <summary>
/// Функция вызываемая для окончания игры
/// </summary>
public void GameIsOver()
{
  RecTimer.Enabled = false;
```

```
        MessageBox.Show("Поздравляю, Вы собрали кубик за - " + TimeForRecords.MinuteTime + "Мин " + TimeForRecords.SecondsTime + "Сек " + TimeForRecords.MilleSecondsTime + "МСек", "Внимание!");
        StartGame = false;
        EndGame = false;

        for (int i = 0; i <= 9; i++)
        {
           if (Records[i].RecMin >=TimeForRecords.MinuteTime)
           {
              if (Records[i].RecSec >=TimeForRecords.SecondsTime)
              {

if(Records[i].RecMilleSec>=TimeForRecords.MilleSecondsTime)
                 {
                    listBox1.SelectedIndex = i;
                    listBox1.Enabled = false;
                    RecordsPanel.Visible = true;
                    RecordSavePanel.Visible = true;
                    RecordsPanel.Focus();
                    break;
                 }
              }
           }
        }
      /// <summary>
      /// Функция меняет 2 элемента Cube[] местами
      /// </summary>
      /// <param name="First">Первый элемент массива.</param>
      /// <param name="Second">Второй элемент массива.</param>
      public void SwapArrCube(ref ClassCube First, ref ClassCube Second)
```

```
        {
            ClassCube Temp = new ClassCube();
            Temp = First;
            First = Second;
            Second = Temp;

        }
        public void button3_Click(object sender, EventArgs e)
        {
            SettingsPanel.Visible = true;
            SettingsPanel.Focus();
        }
        public void trackBar1_Scroll(object sender, EventArgs e)
        {
            SizeCube = trackBar1.Value;
        }
        public void label3_Click(object sender, EventArgs e)
        {
            colorDialog1.ShowDialog();

            ColorLabel.BackColor = colorDialog1.Color;
            ColorWorld = colorDialog1.Color;
            Gl.glClearColor(ColorWorld.R, ColorWorld.G, ColorWorld.B,
ColorWorld.A);

            PanelMenu.BackColor = ColorWorld;
            NewGameButton.BackColor = ColorWorld;
            SettingsButton.BackColor = ColorWorld;
            RecordButton.BackColor = ColorWorld;
            ExitButton.BackColor = ColorWorld;
            SettingsPanel.BackColor = ColorWorld;
            MenuLabel.BackColor = ColorWorld;
```

```
        TimeLabel.BackColor = ColorWorld;



    }
    public void SettingsPanel_Leave(object sender, EventArgs e)
    {
        SettingsPanel.Visible = false;
    }
    public void RecordButton_Click(object sender, EventArgs e)
    {
        RecordsPanel.Visible = true;
        PauseGame = true;
        RecTimer.Enabled = false;
        RecordsPanel.Focus();
    }
    /// <summary>
    /// Saves the record file XML.
    /// </summary>
    public void SaveRecordFileXml() // Сохранение Рекордов в XML с MSDN
    {
        System.Xml.Serialization.XmlSerializer writer = new System.Xml.Serialization.XmlSerializer(
        Records.GetType());
        System.IO.StreamWriter file = new System.IO.StreamWriter(PatchExeFile + "//" + NameDataFolder + "//" + NameRecordFolder + "//" + NameRecFile);
            writer.Serialize(file, Records);
            file.Close();
    }
    /// <summary>
    /// Loads the record file XML.
    /// </summary>
```

```
public void LoadRecordFileXml() //запись Рекордов в XML с MSDN
{
   System.Xml.Serialization.XmlSerializer reader = new System.Xml.Serialization.XmlSerializer(Records.GetType());
   System.IO.StreamReader file = new System.IO.StreamReader(PatchExeFile + "//" + NameDataFolder + "//" + NameRecordFolder +"//" + NameRecFile);
   Records = (RecType[])reader.Deserialize(file);
   file.Close();
   RewrireRecordsTables();
}
/// <summary>
/// Rewrites the records tables.
/// </summary>
public void RewrireRecordsTables()
{
   listBox1.Items.Clear();
   {
      for (int i = 0; i <= 9; i++)
      {
         listBox1.Items.Add(Records[i].Name + '(' + Records[i].RecMin + ':' + Records[i].RecSec + ':' + Records[i].RecMilleSec + ')');
      }
   }
}
public void button2_Click(object sender, EventArgs e)
{
   Records[listBox1.SelectedIndex] = new RecType(NewNameRecordText.Text,TimeForRecords);
   RecordSavePanel.Visible = false;
   listBox1.Enabled = true;
   RewrireRecordsTables();
```

```
        }
        /// <summary>
        /// Случайно вращает грани куба .
        /// </summary>
        public void ShaflCube()
        {
            Random Randomiser = new Random();
            int CountFlips =Randomiser.Next(15, 100) ;
            int SideFlip;
            for (int i = 1; i <= CountFlips; i++)
            {
              SideFlip = Randomiser.Next(0, 5);

FlipSide( VariationOFSelectedCybeMidle[SideFlip][0],VariationOFSelectedC
ybeMidle[SideFlip][1],VariationOFSelectedCybeMidle[SideFlip][2]);
            }
        }
        public void Form1_FormClosing(object sender, FormClosingEventArgs
e)
        {
            SaveRecordFileXml();
        }
    }
}
```

Приложение В

Report git

afa6258c4eaa688711cc12e8e887b0c4bc9162ef 03f0f4bbfd66705fb6311b7a821650b3bdd78113 Neogara <Neogara519@Gmail.com> 1449216000 +0300 commit: Test

03f0f4bbfd66705fb6311b7a821650b3bdd78113 99a568c783d8f8b71b21cee0c1709aa49d528874 Neogara <Neogara519@Gmail.com> 1449218351 +0300 commit: Добавлено:

99a568c783d8f8b71b21cee0c1709aa49d528874 ad8f85e7123a6ddc340a76c80d475bfdf4f4af9a Neogara <Neogara519@Gmail.com> 1449479289 +0300 commit: Исправдено:

ad8f85e7123a6ddc340a76c80d475bfdf4f4af9a 5a6b526076008c54f592ee0e3073875901a3c4a4 Neogara <Neogara519@Gmail.com> 1450763421 +0300 commit: попытки добаить реалицацию поворотов граней посредством работы с указателями

5a6b526076008c54f592ee0e3073875901a3c4a4 3961786dd66bc965d798b045d27bcb3bfa164263 Neogara <Neogara519@Gmail.com> 1450780433 +0300 commit: программа переделана для работы с Class

3961786dd66bc965d798b045d27bcb3bfa164263 ce0ada4d8c43a5117eb63fa6180e3f6ffb5cb197 Neogara <Neogara519@Gmail.com> 1451503403 +0300 commit: Измеенения:

ce0ada4d8c43a5117eb63fa6180e3f6ffb5cb197 6e23eae514c16365c22486524f710d45df33295a Neogara <Neogara519@Gmail.com> 1452305736 +0300 commit: добавленно :

6e23eae514c16365c22486524f710d45df33295a db233b815639764b06d355b89afde6eadf0f3954 Neogara <Neogara519@Gmail.com> 1452718598 +0300 commit: Добавлено

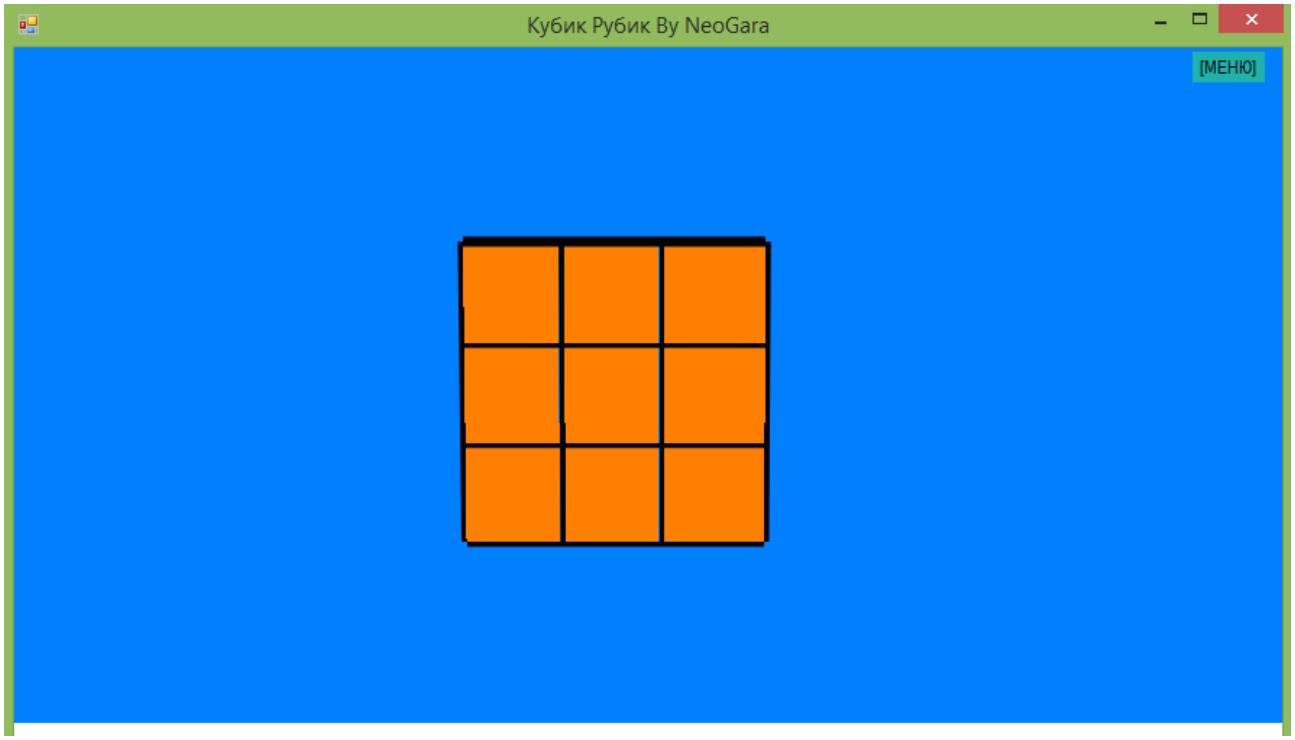
db233b815639764b06d355b89afde6eadf0f3954 cf436718f9311a370b3db0ab3baeefc8545b83d3 Neogara <Neogara519@Gmail.com> 1452726865 +0300 commit: Добавлены Xml-Комментарии

Приложение В

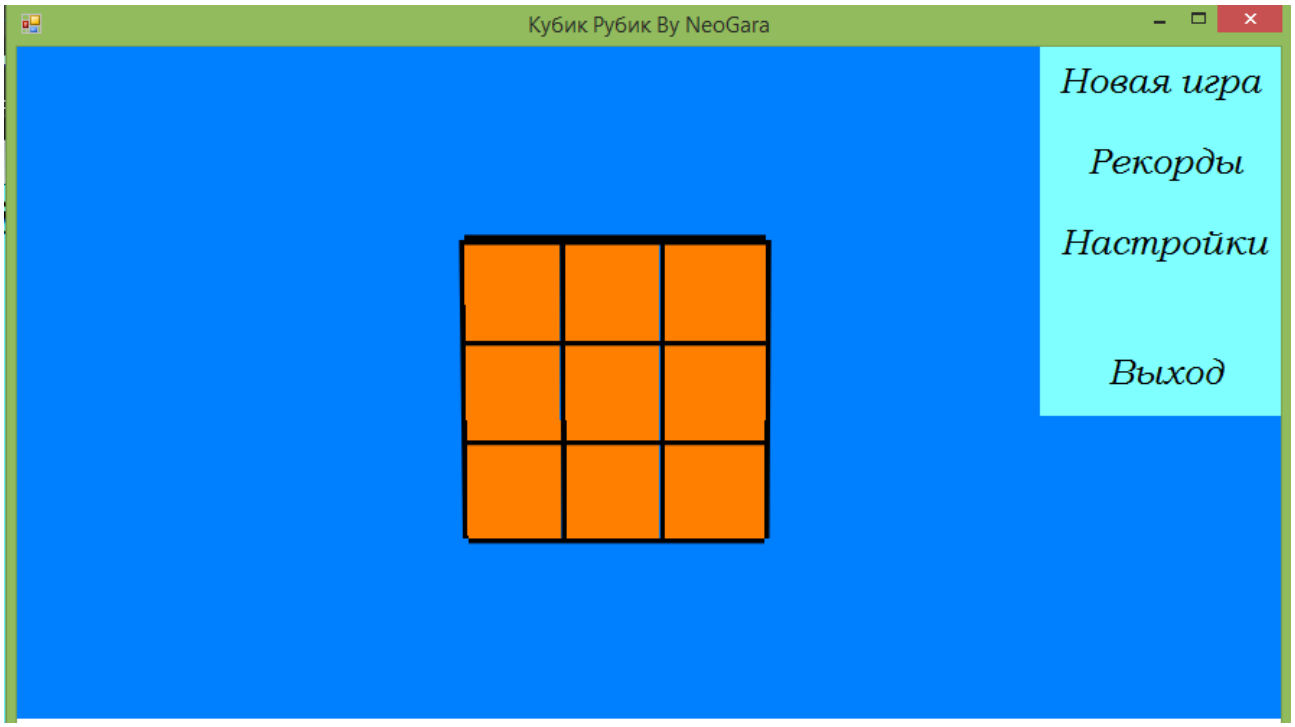
Пользовательское руководство

Для того что бы начать пользоваться этим программный продуктом надо сначала его запустить

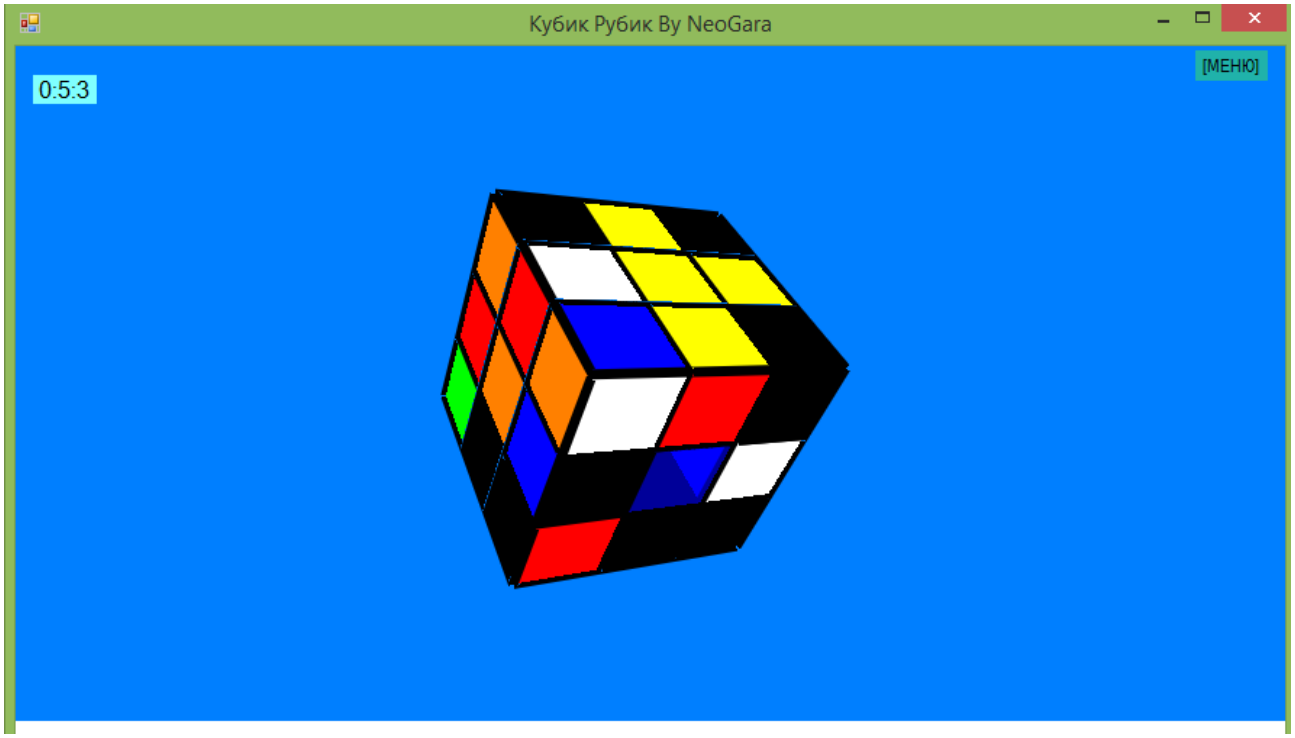
После запуска в окне будет нарисован «Кубик Рубика»



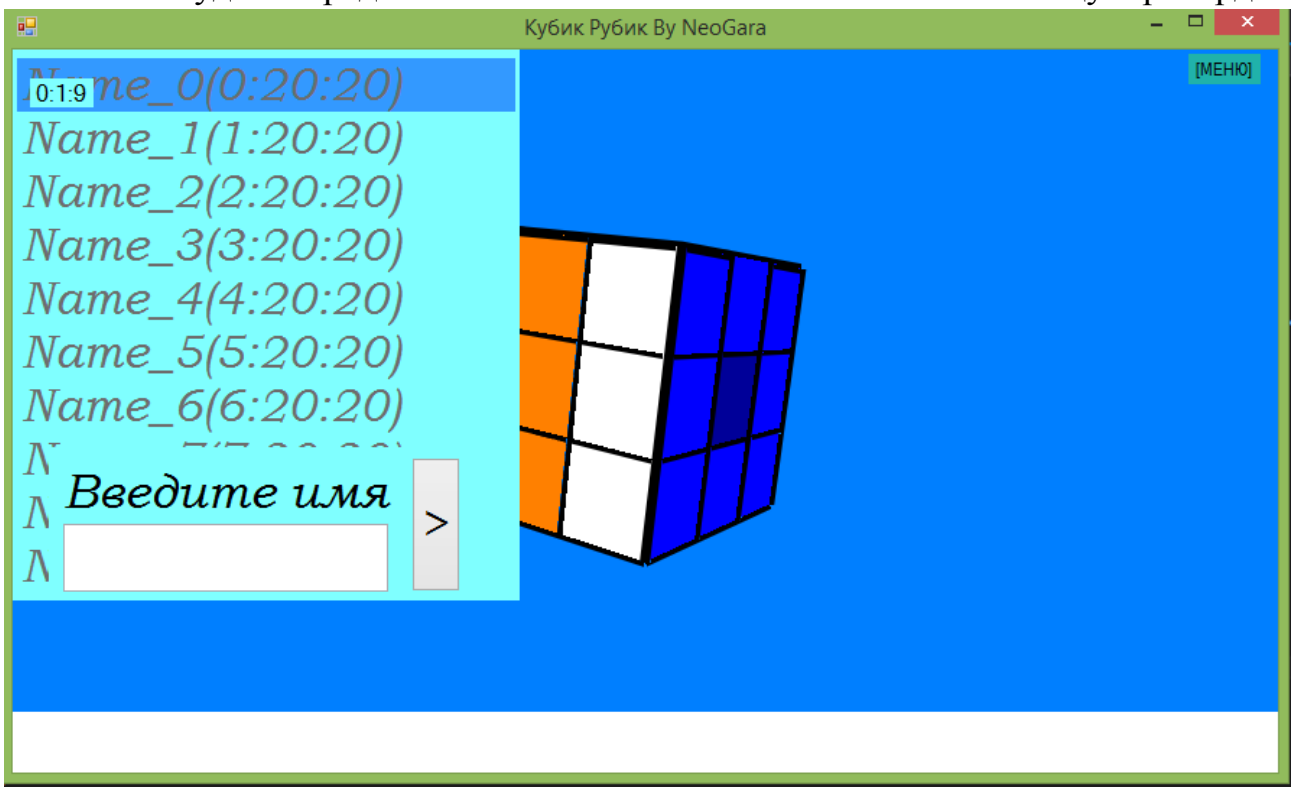
Для запуска игры нам потребуется навести мышь на «Меню» и в появившемся окне выбрать пункт «Новая игра» .



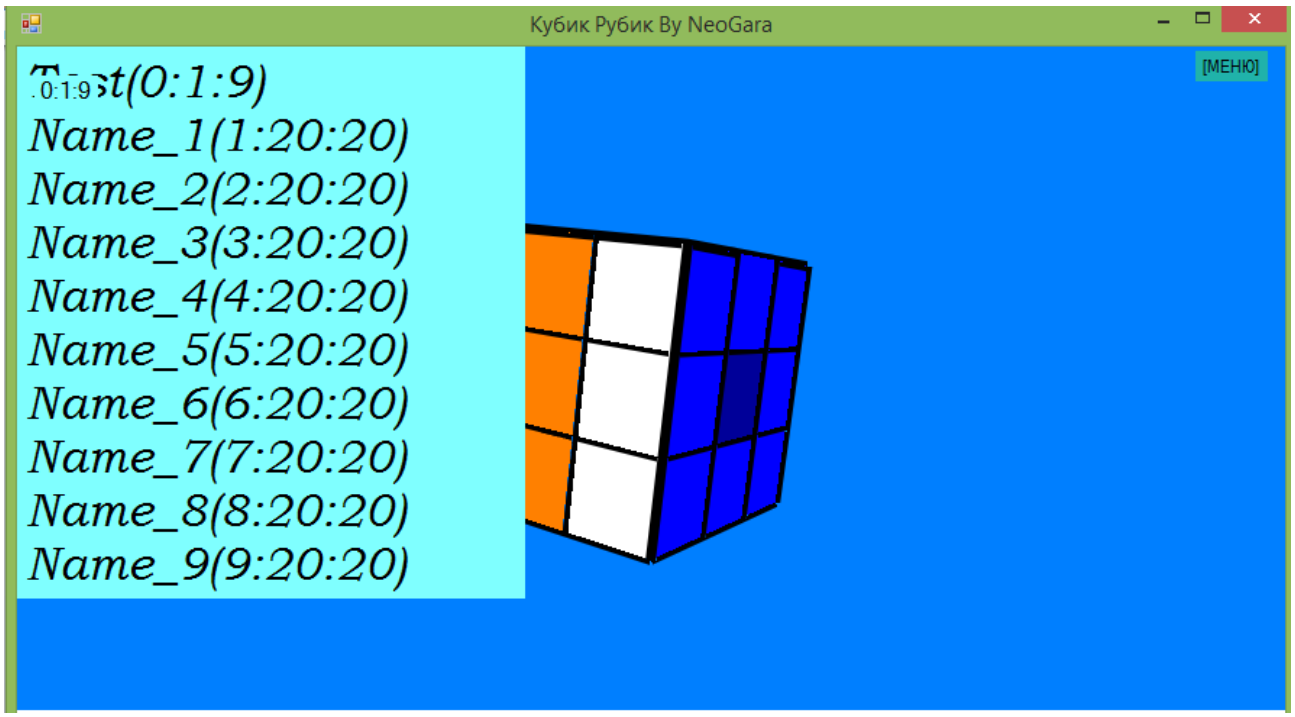
После этого запуститься таймер и мы можете собирать «кубик Рубика» на время.



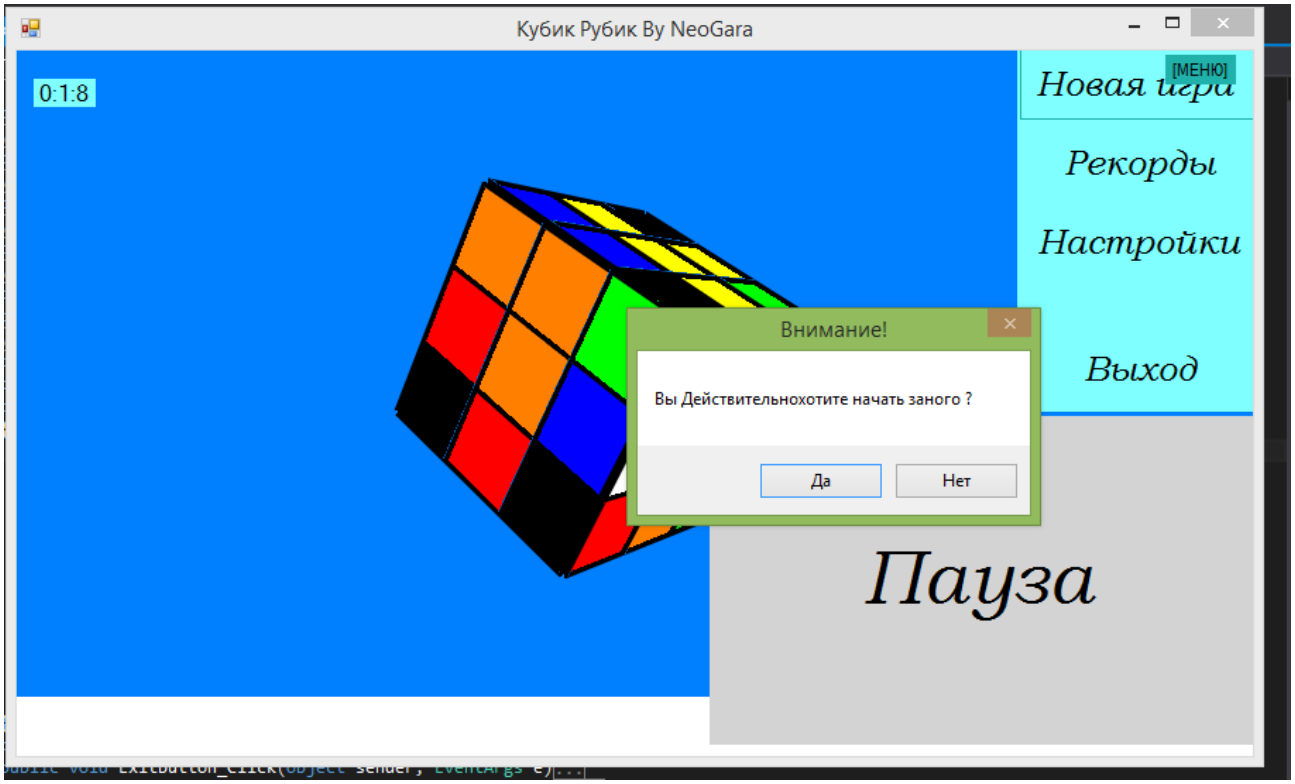
После его сборки , если вы смогли собрать «кубик Рубика» за «топовое» время , то вам будет предложено записать свое имя в таблицу рекордов



После записи этих данных можно нажать на кнопочку немного левее ввода имени , тем самым вы сохраните ваш рекорд и перейдете в меню.



Если же во время сборки вы хотите начать званого то вам потребуется навести мышь на «меню» и выбрать новая игра



Приложение Г

Текст доклада

Введение(разрекламировать себя как программиста и свой программный продукт)

Меня зовут Мельдианов Александр , я учусь на 3 курсе по специальности Программирование в компьютерных системах (ПКС). В конце 2 курка нам начали раздавать темы курсовой , в то время я начал активно увлекаться сборкой «кубика рубика» и по этому мне досталась эта тема курсовой работы. Так как мне импонировал язык C# ,я предпочел изучить и его тоже.

Задание представляло собой написать эмулятор собирания «кубика рубика» с использованием OpneGl.

OpenGL – это кроссплатформенная открытая графическая библиотека .

Использование этой библиотеки для работы с графикой и являлось усложнением моей курсовой работы. Это было сделано для того что бы развить мои способности работы с графикой и создание кроссплатформенных приложений так как сейчас на рынке приложений это является очень нужным навыком для программистов.

Основная часть

Общие сведения

1.1. Программный продукт «Кубик Рубика» для ПК

1.2. Разработал программу Студент УКИТ имени К.Г Разумовскоро Группы П-303 , Мельдианов А.А

1.3. Основное назначение игры - развитие логического мышления , развлечение , обучение создание логических алгоритмов , а так же есть соревновательные элементы

1.4. Технические характеристики программного продукта

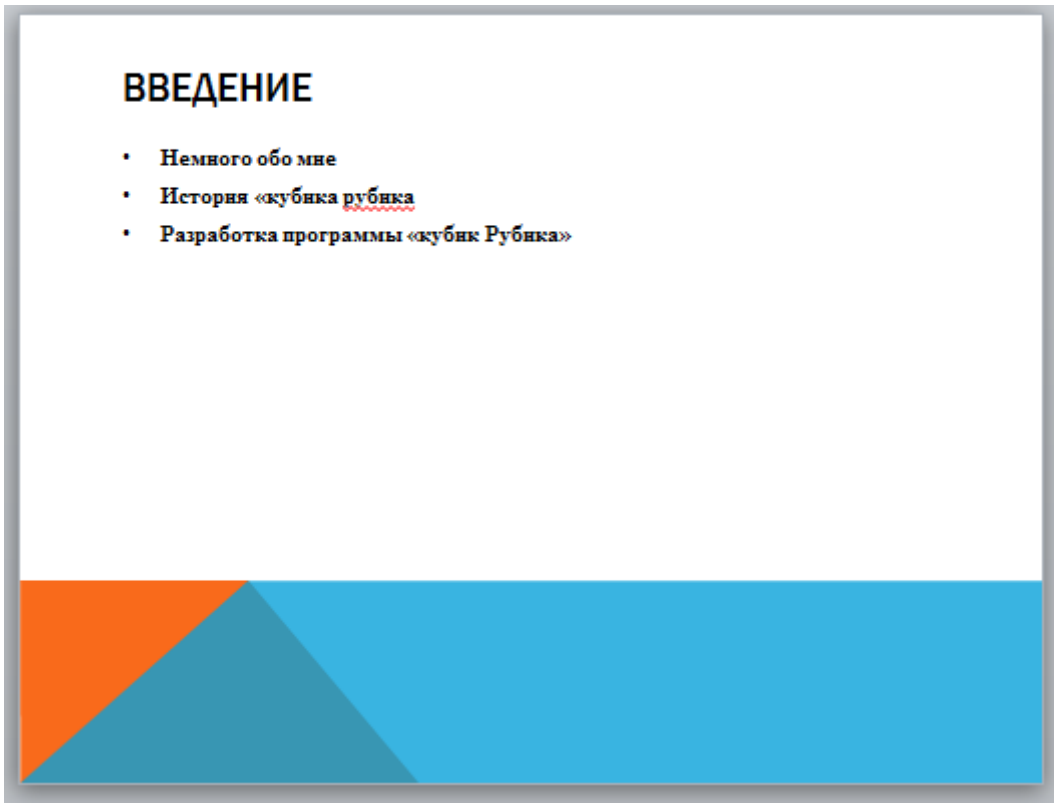
1.5. Программа должна уметь:

- Начало новой игры
- Повороты граней
- Запись и чтение рекордов по времени сборки
- Автоматическое перемешивание кубика
- Возможность собрать кубик

Преимущества

- Возможность записывать лучшее время собирания кубика
- Интуитивный и понятный интерфейс
- Так как программа написана с использованием OpenGl, то она имеет возможность переноса программы на мобильные платформы

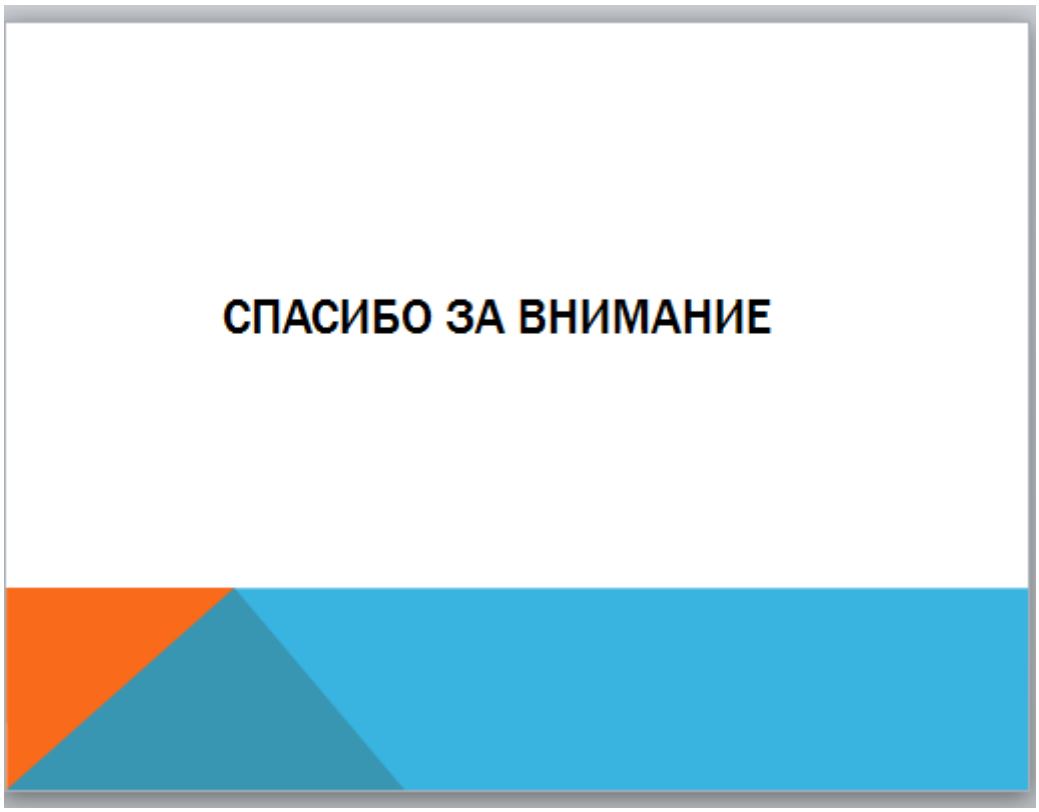
Приложение Д



ПОКАЗ ПРОГРАММЫ

ПРЕИМУЩЕСТВА

- Возможность записывать лучшее время собирания кубика
- Интуитивный и понятный интерфейс
- Так как программа написана с использованием OpenGl, то она имеет возможность переноса программы на мобильные платформы


СПАСИБО ЗА ВНИМАНИЕ