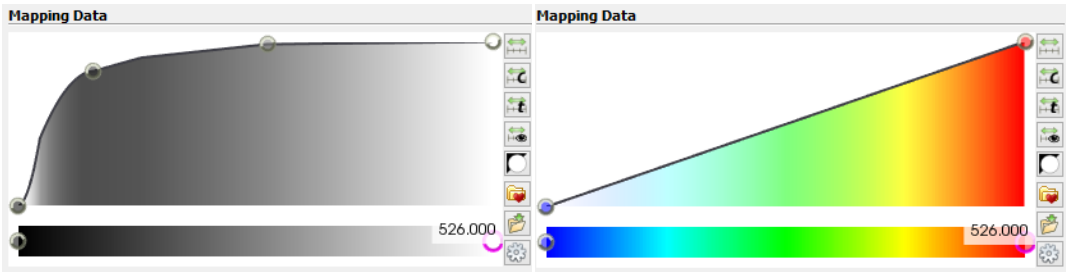
Лабораторная работа "Визуализация томограммы"

Томография - это получение послойного изображения внутренней структуры объекта.

Чаще всего, но далеко не всегда, объектом томографического исследования являются живые ткани. На рисунке ниже представлен слой томограммы головы.

Данные томографии представляют собой трехмерный массив вокселов - элементов трехмерной регулярной сетки. Каждый воксел содержит одно значение плотности, как правило, типа short или ushort.

Для перевода значения плотности в цвет используется передаточная функция = Transfer Function (TF). Transfer Function может быть серой, от черного до белого, или цветной, линейной или нелинейной.



В данной лабораторной работе будет использоваться линейная TF от черного к белому, так как ее очень просто создать, все значения рассчитываются по формуле:

$$intensity = \frac{x - min}{max - min} * 255$$

Создание нового проекта

Для решения задачи послойной визуализации томограммы мы будем использовать язык C# и стандарт и технологию OpenGL.

Создайте новый проект "Приложение Windows Forms" на языке C#, дайте ему название <Фамилия>_tomogram_visualizer.

В качестве библиотеки, реализующей стандарт OpenGL в проекте будет использоваться библиотека OpenTK. Подключение библиотеки к проекту подробно описано в документе "Подключение OpenTK в Visual Studio". Подключите библиотеку OpenTK к своему проекту согласно инструкции.

Чтение файла томограммы

Обычно файлы томограмм хранятся в файлах формата DICOM, но в связи с нетривиальностью данного формата в данной работе будет использоваться томограмма, сохраненная в бинарном формате. Для загрузки томограммы потребуется прочитать из бинарного файла размеры томограммы (3 числа в формате int) и массив данных типа short. Создайте класс для чтения данных файлов:

```
class Bin
{
    public static int X, Y, Z;
    public static short[] array;
    public Bin() { }

    public void readBIN(string path)
    {
        if (File.Exists(path))
        {
            BinaryReader reader =
                new BinaryReader(File.Open(path, FileMode.Open));

            X = reader.ReadInt32();
            Y = reader.ReadInt32();
            Z = reader.ReadInt32();

            int arraySize = X * Y * Z;
            array = new short[arraySize];
            for (int i = 0; i < arraySize; ++i)
            {
                array[i] = reader.ReadInt16();
            }

        }
    }
}
```

Создайте и инициализируйте объект класса Bin в классе Form1.

Классы для визуализации

Создайте класс View, который будет содержать функции для визуализации томограммы.

Настройка камеры

В классе View создайте функцию SetupView, которая будет настраивать окно вывода.

Включите интерполирование цветов, установив тип Smooth функцией GL.ShadeModel.

Матрицу проекции сначала инициализируйте, установив ее равной матрице тождественного преобразования (GL.LoadIdentity()). А затем задайте обращением к GL.Orto() ортогональное проецирование массива данных томограммы в окно вывода, которое попутно преобразует размеры массива в канонический видимый объем (CVV).

Настройте вывод в окно OpenTK таким образом, чтобы разрешение синтезируемого изображения было равно размеру окна OpenTK.

Все действия по настройке камеры записаны в коде ниже:

```
public void SetupView(int width, int height)
{
    GL.ShadeModel(ShadingModel.Smooth);
    GL.MatrixMode(MatrixMode.Projection);
    GL.LoadIdentity();
    GL.Ortho(0, Bin.X, 0, Bin.Y, -1, 1);
    GL.Viewport(0, 0, width, height);
}
```

Визуализация томограммы

В данной лабораторной работе будет проведено сравнение двух вариантов визуализации томограммы.

Вариант 1 - отрисовка четырехугольниками, вершинами которых являются центры вокселов текущего слоя регулярной воксельной сетки. Цвет формируется на центральном процессоре и отрисовывается с помощью функции `GL.Begin(BeginMode.Quads)`.

Вариант 2 - отрисовка текстурой. Текущий слой томограммы визуализируется как один большой четырехугольник, на который изображение слоя накладывается как текстура аппаратной билинейной интерполяцией.

Создание Transfer Function (TF)

TF - функция перевода значения плотностей томограммы в цвет. Диапазон визуализируемых плотностей называется окном визуализации. В нашем случае мы хотим, чтобы TF переводила плотности окна визуализации от 0 до 2000 линейно в цвет от черного до белого (от 0 до 255).

```
Color TransferFunction(short value)
{
    int min = 0;
    int max = 2000;
    int newVal = clamp((value - min) * 255 / (max - min), 0, 255);
    return Color.FromArgb(255, newVal, newVal, newVal);
}
```

Меняя параметры min и max, мы будет получать различные изображения для нашей томограммы. Часто TF имеет более сложную структуру, чем линейная зависимость от максимума и минимума, но в данной лабораторной работе нам достаточно такой.

Отрисовка четырехугольника

В классе View создайте функцию DrawQuads с параметром int layerNumber (номер визуализируемого слоя).

```
public void DrawQuads(int layerNumber)
{
    GL.Clear(ClearBufferMask.ColorBufferBit | ClearBufferMask.DepthBufferBit);
    GL.Begin(BeginMode.Quads);
    for (int x_coord = 0; x_coord < Bin.X - 1; x_coord++)
        for (int y_coord = 0; y_coord < Bin.Y - 1; y_coord++)
        {
            short value;
            //1 вершина
            value = Bin.array[x_coord + y_coord * Bin.X
                              + layerNumber * Bin.X * Bin.Y];
            GL.Color3(TransferFunction(value));
            GL.Vertex2(x_coord, y_coord);
            //2 вершина
            value = Bin.array[x_coord + (y_coord + 1) * Bin.X
                              + layerNumber * Bin.X * Bin.Y];
            GL.Color3(TransferFunction(value));
            GL.Vertex2(x_coord, y_coord + 1);
            //3 вершина
            value = Bin.array[x_coord + 1 + (y_coord + 1) * Bin.X
                              + layerNumber * Bin.X * Bin.Y];
            GL.Color3(TransferFunction(value));
            GL.Vertex2(x_coord + 1, y_coord + 1);
            //4 вершина
            value = Bin.array[x_coord + 1 + y_coord * Bin.X
                              + layerNumber * Bin.X * Bin.Y];
            GL.Color3(TransferFunction(value));
            GL.Vertex2(x_coord + 1, y_coord);
        }
    GL.End();
}
```

Из томограммы извлекаются значения томограммы в 4 ячейках: (x,y), (x+1,y), (x,y+1), (x+1,y+1). Эти значения заносятся в цвет вершин четырехугольника, и данный четырехугольник визуализируется. Данная операция происходит в цикле по ширине и высоте томограммы. Перечисление вершин четырехугольника происходит против часовой стрелки.

Загрузка файла с данными и его визуализация

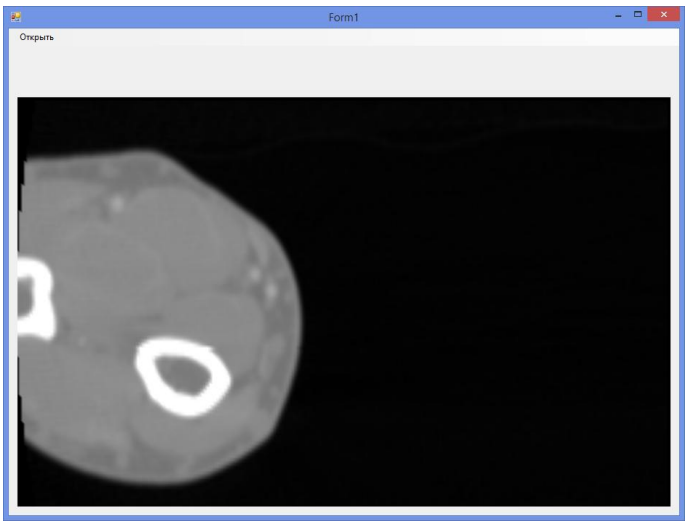
Создайте кнопку, либо элемент меню, по нажатию на который будет вызываться функция, которая будет открывать файл с томограммой и настраивать OpenGL окно. Код функции приведен ниже. В классе Form1 необходимо объявить переменную bool loaded = false, чтобы не запускать отрисовку, пока не загружены данные.

```
private void открытьToolStripMenuItem_Click(object sender, EventArgs e)
{
    OpenFileDialog dialog = new OpenFileDialog();
    if (dialog.ShowDialog() == DialogResult.OK)
    {
        string str = dialog.FileName;
        bin.readBIN(str);
        view.SetupView(glControl1.Width, glControl1.Height);
        loaded = true;
        glControl1.Invalidate();
    }
}
```

Откройте конструктор формы, откройте свойства OpenGL окна и в событиях (Events) выберите событие Paint, двойным щелчком создайте новую функцию, в ней вызовите функцию DrawQuads и функцию SwapBuffers. В OpenGL используется двойная буферизация (буфер, выводящий изображение на экран и буфер, используемый для создания изображения), функция SwapBuffers загружает наш буфер в буфер экрана. currentLayer - переменная типа int, которая хранит номер слоя для визуализации.

```
private void glControl1_Paint(object sender, PaintEventArgs e)
{
    if (loaded)
    {
        view.DrawQuads(currentLayer);
        glControl1.SwapBuffers();
    }
}
```

Запустите программу, удостоверьтесь, что томограмма визуализируется.



Перемотка слоев

Добавьте на форму Trackbar, по умолчанию значения значение Maximux равно 10. В функции загрузки томограммы измените значение максимума на количество слоев (переменная Z в классе Bin).

```
private void trackBar1_Scroll(object sender, EventArgs e)
{
    currentLayer = trackBar1.Value;
}
```

Проверьте, что теперь слои томограммы перелистываются.

Создание бесконечного цикла рендеринга и счетчика кадров

Производительность визуализации, измеряется в кадрах в секунду (Frames per second, FPS), чтобы её измерить нужно после рендера одного кадра и вывода его на экран автоматически начинать рендерить следующий кадр. Функция Application_Idle проверяет, занято ли OpenGL окно работой, если нет, то вызывается функция Invalidate, которая заставляет кадр рендериться заново.

```
void Application_Idle(object sender, EventArgs e)
{
    while (glControl1.IsIdle)
    {
        glControl1.Invalidate();
    }
}
```

Чтобы функция Application_Idle работала автоматически, вам нужно подключить ее в программе. В конструкторе формы создайте функцию Form1_load, в которой вы подключите Application_Idle на автоматическое выполнение.

```
private void Form1_Load(object sender, EventArgs e)
{
    Application.Idle += Application_Idle;
}
```

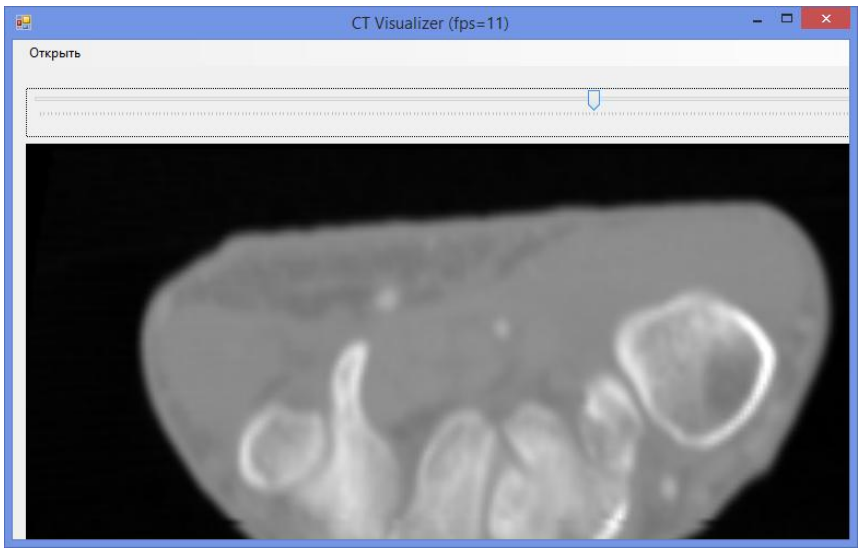
В классе Form1 создайте переменные int FrameCount, DateTime NextFPSUpdate и функцию displayFPS.

```
int FrameCount;
DateTime NextFPSUpdate = DateTime.Now.AddSeconds(1);
void displayFPS()
{
    if (DateTime.Now >= NextFPSUpdate)
    {
        this.Text = String.Format("CT Visualizer (fps={0})", FrameCount);
        NextFPSUpdate = DateTime.Now.AddSeconds(1);
        FrameCount = 0;
    }
    FrameCount++;
}
```

Вызовите данную функцию обновления FPS в функции Application_Idle.

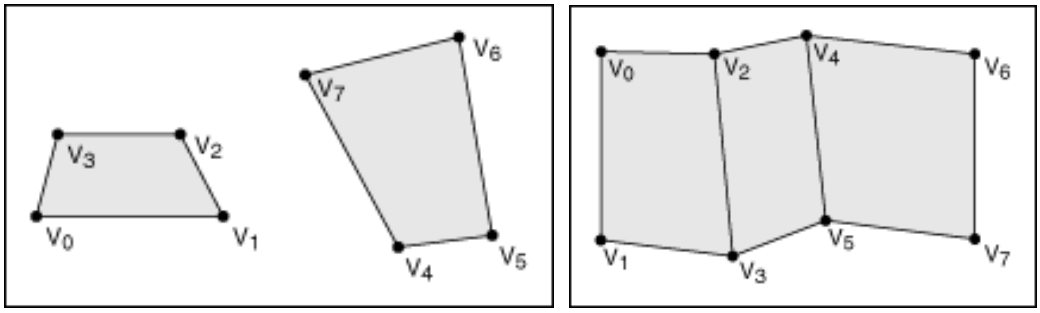
```
void Application_Idle(object sender, EventArgs e)
{
    while (glControl1.IsIdle)
    {
        displayFPS();
        glControl1.Invalidate();
    }
}
```

Запустите программу, посмотрите, какой fps будет выдавать ваша программа.



Дополнительное задание

В OpenGL есть тип визуализации QuadStrip, когда первый четырехугольник рисуется 4 вершинами, а последующие - 2 вершинами, присоединенными к предыдущему четырехугольнику (рис. ниже). Таким образом для отрисовки N четырехугольников требуется не 4*N вершин, а 2*N+2 вершин, что положительно сказывается на скорости работы программы.



Визуализация томограммы как текстуры

Создайте функцию загрузки и функцию визуализации.

Загрузка текстуры в память видеокарты

В класса View создайте переменную int VBOtexture и функцию Load2dTexture. Переменная VBOtexture будет хранить номер текстуры в памяти видеокарты. Функция GenTextures генерирует уникальный номер текстуры, функция BindTexture связывает текстуру, делает ее активной, а также указывает ее тип, функция TexImage2D загружает текстуру в память видеокарты.

```
Bitmap textureImage;
int VBOtexture;
public void Load2DTexture()
{
    GL.BindTexture(TextureTarget.Texture2D, VBOtexture);
    BitmapData data = textureImage.LockBits(
        new System.Drawing.Rectangle(0, 0, textureImage.Width, textureImage.Height),
        ImageLockMode.ReadOnly,
        System.Drawing.Imaging.PixelFormat.Format32bppArgb);

    GL.TexImage2D(TextureTarget.Texture2D, 0, PixelInternalFormat.Rgba,
        data.Width, data.Height, 0, OpenTK.Graphics.OpenGL.PixelFormat.Bgra,
        PixelType.UnsignedByte, data.Scan0);

    textureImage.UnlockBits(data);

    GL.TexParameter(TextureTarget.Texture2D, TextureParameterName.TextureMinFilter,
        (int)TextureMinFilter.Linear);
    GL.TexParameter(TextureTarget.Texture2D, TextureParameterName.TextureMagFilter,
        (int)TextureMagFilter.Linear);

    ErrorCode Er = GL.GetError();
    string str = Er.ToString();
}
```

Визуализация томограммы одним прямоугольником

Суть визуализации томограммы одним прямоугольником будет заключаться в следующем: мы сделаем картинку из текстуры один раз на процессоре, передадим ее в видеопамять, и будем производить текстурирование одного прямоугольника.

В классе View создайте переменную Bitmap textureImage и функцию generateTextureImage, которая будет генерировать изображение из томограммы при помощи созданной Transfer Function.

```
public void generateTextureImage(int layerNumber)
{
    textureImage = new Bitmap(Bin.X, Bin.Y);
    for (int i = 0; i < Bin.X; ++i)
        for (int j = 0; j < Bin.Y; ++j)
        {
            int pixelNumber = i + j * Bin.X + layerNumber * Bin.X * Bin.Y;
            textureImage.SetPixel(i, j, TransferFunction(Bin.array[pixelNumber]));
        }
}
```

Создайте функцию drawTexture(), которая будет включать 2d-текстурирование, выбирать текстуру и рисовать один прямоугольник с наложенной текстурой, потом выключать 2d-текстурирование.

```
public void DrawTexture()
{
    GL.Clear(ClearBufferMask.ColorBufferBit | ClearBufferMask.DepthBufferBit);
    GL.Enable(EnableCap.Texture2D);
    GL.BindTexture(TextureTarget.Texture2D, VBOtexture);

    GL.Begin(BeginMode.Quads);
    GL.Color3(Color.White);
    GL.TexCoord2(0f, 0f);
    GL.Vertex2(0, 0);
    GL.TexCoord2(0f, 1f);
    GL.Vertex2(0, Bin.Y);
    GL.TexCoord2(1f, 1f);
    GL.Vertex2(Bin.X, Bin.Y);
    GL.TexCoord2(1f, 0f);
    GL.Vertex2(Bin.X, 0);
    GL.End();

    GL.Disable(EnableCap.Texture2D);
}
```

Визуализация с помощью четырехугольников разбивается на 2 подзадачи:

1. Генерация текстуры и загрузка в видеопамять. Выполняется один раз для слоя.

2. Визуализация текстуры. Происходит постоянно.

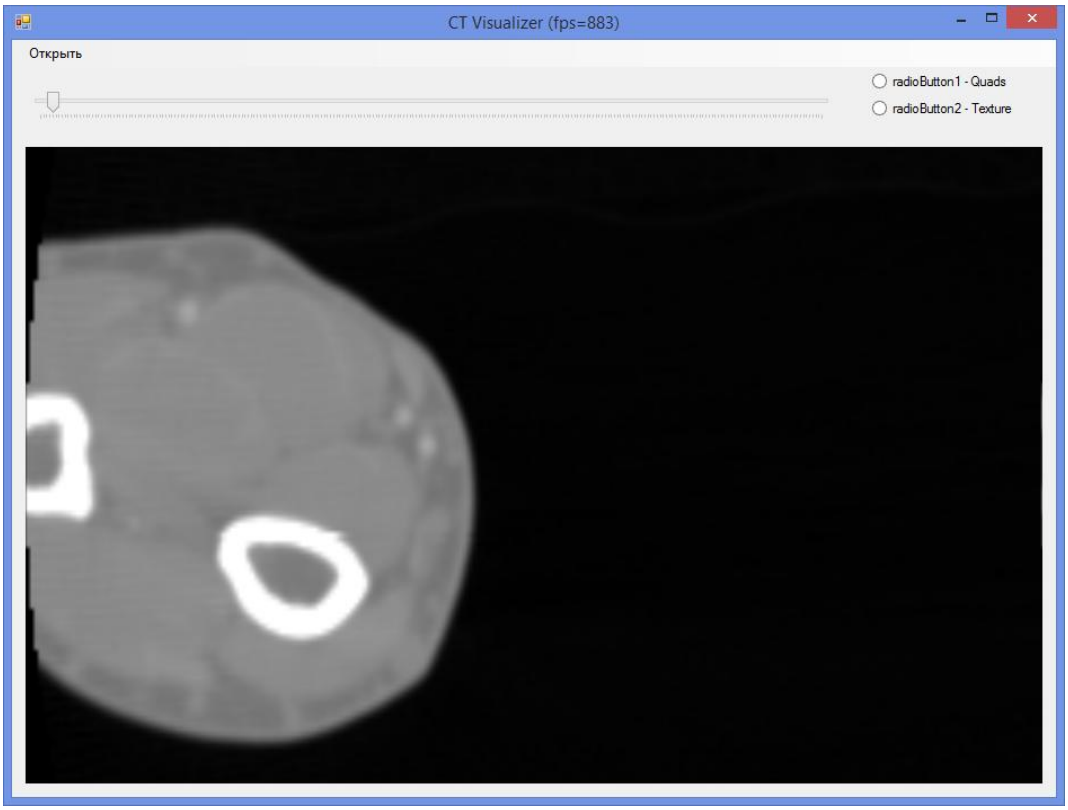
В классе Form1 измените функцию glControl1_Paint так, чтобы она рисовала томограмму с помощью текстуры, а загружала текстуру только когда переменная needReload будет установлена в true.

```
bool needReload = false;
private void glControl1_Paint(object sender, PaintEventArgs e)
{
    if (loaded)
    {
        //view.DrawQuads(currentLayer);
        if (needReload)
        {
            view.generateTextureImage(currentLayer);
            view.Load2DTexture();
            needReload = false;
        }
        view.DrawTexture();
        glControl1.SwapBuffers();
    }
}
```

Переменную needReload необходимо устанавливать в значение true тогда, когда мы изменяем trackbar.

```
private void trackBar1_Scroll(object sender, EventArgs e)
{
    currentLayer = trackBar1.Value;
    needReload = true;
}
```

Запустите программу. Сравните FPS версии рисования текстурой с FPS версии рисования четырехугольниками.



Сделайте возможность переключаться между режимами визуализации четырехугольниками и текстурой (например, при помощи CheckBox или RadioButton).

Изменение Transfer Function

Создайте два TrackBar, которые будут использоваться для задания Transfer Function.

Первый TrackBar будет указывать на значение минимума, второй - на ширину TF, тогда значение максимума можно вычислить как сумму данных двух значений. Не забудьте, что при изменении TF в режиме рисования текстурой необходимо загружать новую текстуру в видеопамять.