Лабораторная работа №7.2:
3D графика в проектах WPF

Цель:

Целью данной работы является получение навыков работы с трёхмерной графикой на языке высокого уровня C# в среде программирования Microsoft Visual Studio 2017 Community

Задание:

1) Реализовать примеры, описанные в лабораторной работе.

Справочная информация:

Инициализация сцены:

Для начала работы с 3D графикой, необходимо подготовит трёхмерную сцену, в которой будут располагаться трёхмерные объекты. Для отрисовки сцены на экран можно использовать визуальный компонент - **`Viewport3D,`** с именем “scene”.

```
 9      <Grid x:Name="grd">
10
11          <Viewport3D Name="scene" KeyDown="Window_KeyDown">
12
13          </Viewport3D>
14      </Grid>
15  </Window>
```

Для проекции сцены на экран, необходимо инициализировать объект трёхмерной камеры:

```
//пространство имён для работы с 3D
using System.Windows.Media.Media3D;
…
//создание объекта “камера”
PerspectiveCamera camera = new PerspectiveCamera();
//установка позиции камеры
camera.Position = new Point3D(0, 2, 0.1);
//точка, на которую камера будет смотреть
Vector3D lookAt = new Vector3D(0, 0, 0);
//вычисление направления вектора камеры (можно задавать как вектор)
camera.LookDirection = Vector3D.Subtract(lookAt, new Vector3D(0, 2, 0.1));
//установка дальней и ближней плоскостей отсечения и вектора определяющего где верх
camera.FarPlaneDistance = 1000;
camera.NearPlaneDistance = 1;
camera.UpDirection = new Vector3D(0, 1, 0);
//угол обзора камеры
camera.FieldOfView = 75;

//установка камеры в сцену
scene.Camera = camera;
```

В завершении инициализации сцены, можно определить, каким цветом будет закрашен задний фон:

```
grd.Background = Brushes.LightGray;
```

В качестве заднего фона может быть использован не только цвет, но и изображение (объект ImageBrush).

Добавление в сцену объекта:

Для того, чтобы добавить в сцену трёхмерный объект, необходимо инициализировать:

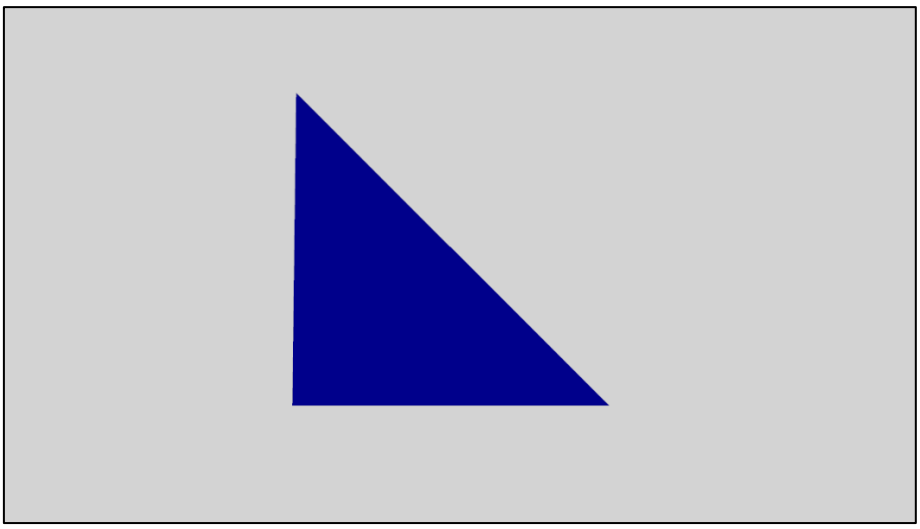
1. структуру для хранения геометрии объекта
2. структуру для хранения материала объекта

3. структуру описывающую объект
4. структуру представляющую визуальный объект

Пример добавления в сцену треугольника:

```
//создание гкометрии
MeshGeometry3D geometry = new MeshGeometry3D();
//добавление координат вершин треугольника
geometry.Positions.Add(new Point3D(-0.5, 0, -0.5));
geometry.Positions.Add(new Point3D(-0.5, 0, 0.5));
geometry.Positions.Add(new Point3D(0.5, 0, 0.5));
//перечисление индексов вершин в порядке их соединения (против часовой стрелки)
geometry.TriangleIndices.Add(0);
geometry.TriangleIndices.Add(1);
geometry.TriangleIndices.Add(2);
//создание материала (тёмно синего цвета)
DiffuseMaterial mat = new DiffuseMaterial(new SolidColorBrush(Colors.DarkBlue));
//создание модели
GeometryModel3D model = new GeometryModel3D(geometry, mat);
//создание визуальной модели
ModelVisual3D triangle = new ModelVisual3D();
triangle.Content = model;
//добавление модели в сцену
scene.Children.Add(triangle);
```

Результат:



Текстурирование:

Для того, чтобы наложить текстуры на объект, необходимо задать его текстурные координаты и использовать в качестве материала ImageBrush:

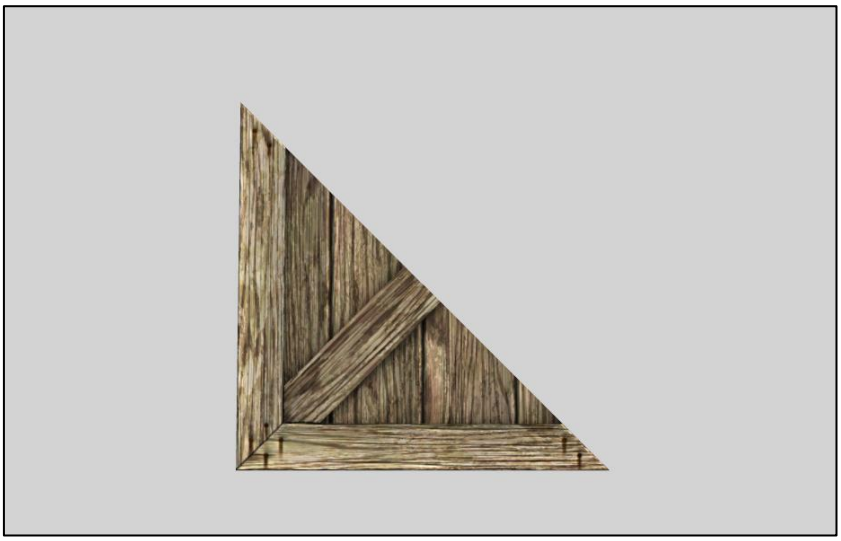
```
//добавление текстурных координат
geometry.TextureCoordinates.Add(new Point(1, 0));
geometry.TextureCoordinates.Add(new Point(1, 1));
geometry.TextureCoordinates.Add(new Point(0, 1));
//создание кисти
ImageBrush ib = new ImageBrush();
//загрузка изображения и назначение кисти
ib.ImageSource = new BitmapImage(new Uri(@"pack://application:,,,/yachik3.jpg", UriKind.Absolute));
//создание материала
DiffuseMaterial mat = new DiffuseMaterial(ib);
```

Для того, что бы объект было видно не зависимо от источников освещения, можно использовать:

```
//создание фоновой подсветки
AmbientLight al = new AmbientLight(Colors.LightYellow);
//добавление фоновой подсветки в сцену
```

```
ModelVisual3D visualModel = new ModelVisual3D();
visualModel.Content = al;
```

Результат:



Добавление освещения:

Для того, чтобы применить освещение к загруженным объектам, необходимо рассчитать их нормали и инициализировать источник освещения.

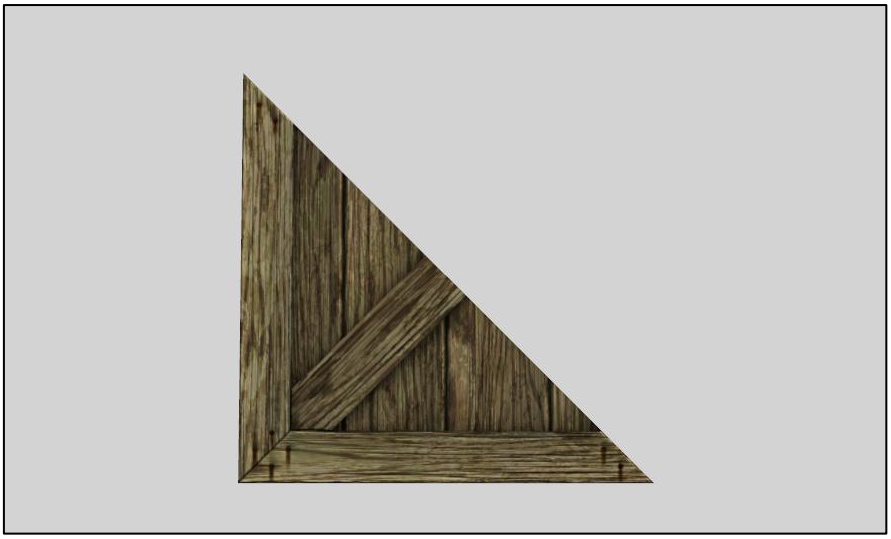
Инициализация точечного источника освещения выглядит следующим образом:

```
//создание точечного источника освещения
PointLight pl = new PointLight();
//установка цвета света
pl.Color = Colors.LightYellow;
//установка позиция источника
pl.Position = new Point3D(0, 5, -5);
//создание модели описывающей источник в сцене
ModelVisual3D light = new ModelVisual3D();
light.Content = pl;
//добавление источника в сцену
scene.Children.Add(light);
```

Добавление нормалей осуществляется следующим образом:

```
//добавление нормалей для каждой из вершин
geometry.Normals.Add(new Vector3D(0, 1, 0));
geometry.Normals.Add(new Vector3D(0, 1, 0));
geometry.Normals.Add(new Vector3D(0, 1, 0));
```

Результат:



Преобразования объектов:

Трёхмерные объекты, находящиеся в сцене, можно вращать, перемещать и масштабировать при помощи матричных преобразований. Матричные преобразования могут быть использованы как по отдельности, так и группами. Следует помнить, что порядок применения преобразований имеет значение.

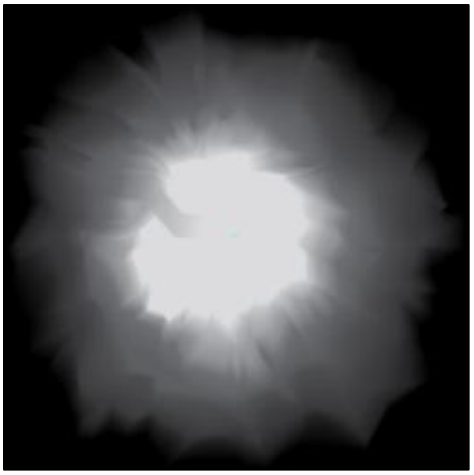
Пример применения группы преобразований к модели:

```
//преобразование переноса модели в точку с координатами 10:0:10
TranslateTransform3D tr = new TranslateTransform3D(10, 0, 10);
//преобразование масштабирования модели (уменьшение в 2 раза по всем осям)
ScaleTransform3D sc = new ScaleTransform3D(0.5, 0.5, 0.5);
//преобразование вращения модели вокруг оси Y на 90 градусов
AxisAngleRotation3D ax3d = new AxisAngleRotation3D(new Vector3D(0, 1, 0), 90);
RotateTransform3D rt = new RotateTransform3D(ax3d);
//создание группы преобразований
Transform3DGroup tg = new Transform3DGroup();
//последовательное применение преобразований (масштабирование, вращение, перенос)
tg.Children.Add(sc);
tg.Children.Add(rt);
tg.Children.Add(tr);
//назначение преобразований модели
model.Transform = tg;
```

Создание ландшафта по карте высот:

Одним из примеров применения 3D графики, является приложение для генерации трёхмерной модели ландшафта по карте высот.

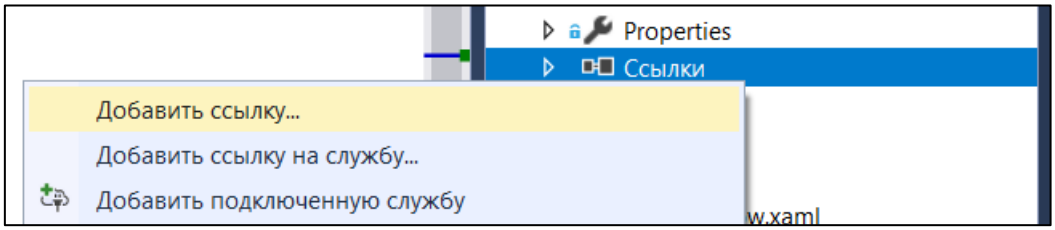
Карта высот – растровое изображение, хранящее высоты ландшафта в определённых точках в виде значения цвета в диапазоне от 0 до 255. Пример карты высот “плато”:



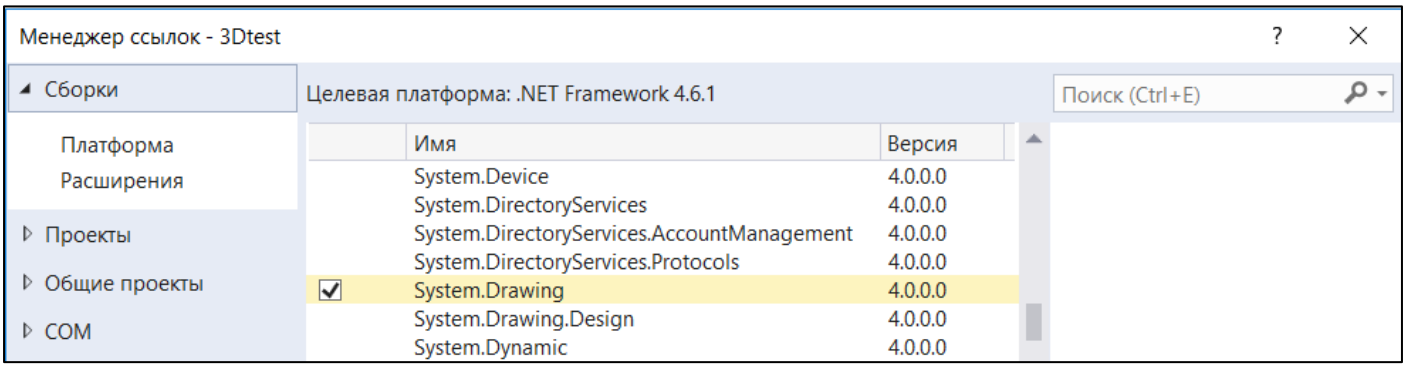
Светлые точки – возвышенности.

Для того, чтобы загрузить карту высот, можно использовать тип данных `System.Drawing.Bitmap`.

В случае, если пространство имён `System.Drawing` не доступно в вашем проекте, нажмите правой кнопкой мыши на “Ссылки” в “Обозревателе решений” и выберите “Добавить ссылку”:



В появившемся окне, найдите пространство имён `System.Drawing`, отметьте “галкой” и нажмите Ок.



Загрузка карты высот будет выглядеть следующим образом:

```
OpenFileDialog dlg = new OpenFileDialog();

dlg.ShowDialog();

System.Drawing.Bitmap hMap;
hMap = new System.Drawing.Bitmap(dlg.FileName);
```

Построение трёхмерной модели по карте высот состоит из генерации точек ландшафта:

```
//размер ландшафта (256x256 пикселей, как у карты высот)
const int N = 256;
//модель для отображения ландшафта
ModelVisual3D terrain = new ModelVisual3D();

MeshGeometry3D geometry = new MeshGeometry3D();

for (int i = 0; i < N; i++)
    for (int j = 0; j < N; j++)
    {
        //расстановка точек ландшафта
```

```
        double y = hMap.GetPixel(i, j).R/10.0;
        geometry.Positions.Add(new Point3D(i, y, j));
        //вычисление текстурных координат для точек ландшафта
        double tu = i / Convert.ToDouble(N);
        double tv = j / Convert.ToDouble(N);
        geometry.TextureCoordinates.Add(new Point(tu, tv ));
    }
```

генерации порядка объединения точек в треугольники:

```
for (int i = 0; i < N - 1; i++)
    for (int j = 0; j < N - 1; j++)
    {
        //вычисление индексов 4х точек, находящихся рядом
        int ind0 = i + j * N;
        int ind1 = (i + 1) + j * N;
        int ind2 = i + (j + 1) * N;
        int ind3 = (i + 1) + (j + 1) * N;
        //описание первого треугольника
        geometry.TriangleIndices.Add(ind0);
        geometry.TriangleIndices.Add(ind1);
        geometry.TriangleIndices.Add(ind3);
        //описание второго треугольника
        geometry.TriangleIndices.Add(ind0);
        geometry.TriangleIndices.Add(ind3);
        geometry.TriangleIndices.Add(ind2);
    }
```

Примечание: всю модель ландшафта, можно условно разделить на квадраты, каждый из которых будет состоять из двух треугольников, поэтому в цикле за одну итерацию описываются 2 треугольника.

Создание и установка материала:

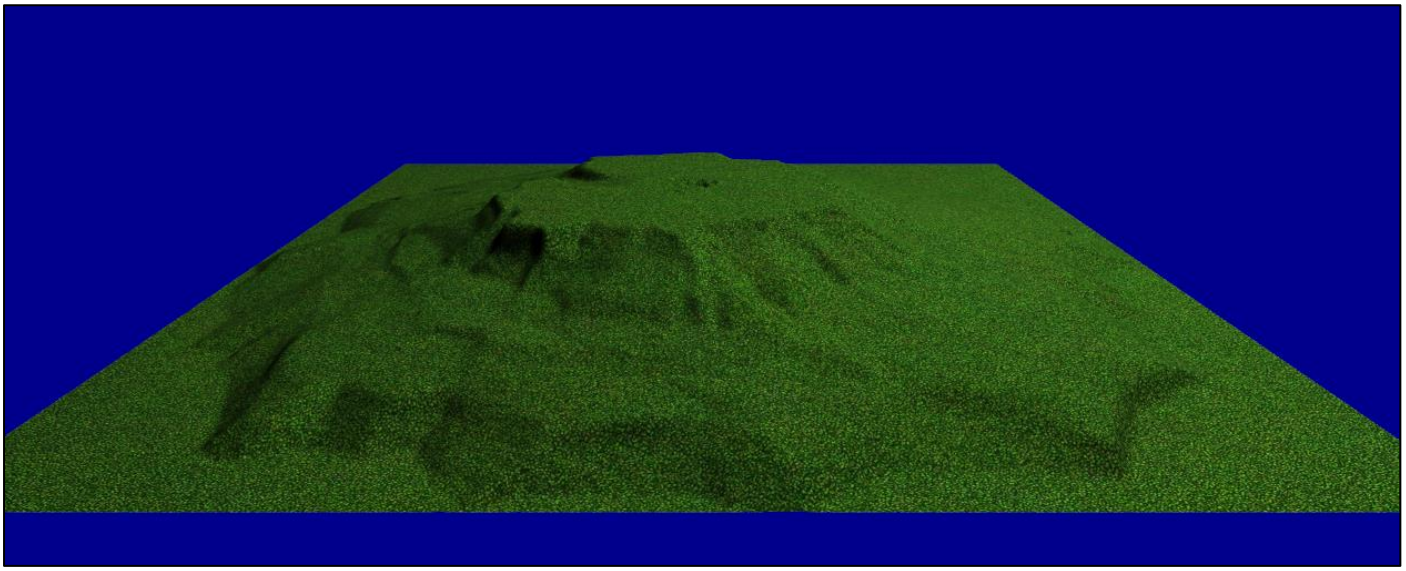
```
ImageBrush ib = new ImageBrush();
//загрузка изображения и назначение кисти
ib.ImageSource = new BitmapImage(new Uri(@"pack://application:,,,/imgs/grasstile.jpg",
UriKind.Absolute));
//масштабирование кисти, текстура будет повторена 4 раза по поверхности ландшафта
ib.Transform = new ScaleTransform(0.5, 0.5);
ib.TileMode = TileMode.Tile;
ib.Stretch = Stretch.Fill;
//создание материала
DiffuseMaterial mat = new DiffuseMaterial(ib);
//создание модели
GeometryModel3D model = new GeometryModel3D(geometry, mat);
//установка модели в визуальный объект
terrain.Content = model;
//добавление визуального объекта в сцену
scene.Children.Add(terrain);
```

Для данного примера, настройки камеры и источника освещения имеет смысл сделать относительными:

```
PerspectiveCamera camera = new PerspectiveCamera();
//расположение камеры
camera.Position = new Point3D(N/2, N/2, N*1.5);
//точка, на которую направлена камера (центр ландшафта)
Vector3D lookAt = new Vector3D(N/2, 0, N/2);
camera.LookDirection = Vector3D.Subtract(lookAt, new Vector3D(N/2, N/2, N*2));

camera.FarPlaneDistance = 1000;
camera.NearPlaneDistance = 1;
camera.UpDirection = new Vector3D(0, 1, 0);
camera.FieldOfView = 75;
scene.Camera = camera;
```

```
PointLight pl = new PointLight();
pl.Color = Colors.LightYellow;
pl.Position = new Point3D(N, N, N/2);

ModelVisual3D light = new ModelVisual3D();
light.Content = pl;

scene.Children.Add(light);
```

Результат:



Для того, чтобы реализовать вращение трёхмерной модели ландшафта, необходимо реализовать обработчик события нажатия клавиши сцены:

```
//глобальная переменная, для хранения угла поворота
double angle = 0.0;
…
private void Window_KeyDown(object sender, KeyEventArgs e)
{
    //если нажата стрелка влево
    if (e.Key == Key.Left)
    {
        angle--;
    }
    //если нажата стрелка вправо
    if (e.Key == Key.Right)
    {
        angle++;
    }
    //создание поворота вокруг оси Y на угол angle
    AxisAngleRotation3D ax3d = new AxisAngleRotation3D(new Vector3D(0, 1, 0), angle);
    RotateTransform3D rt = new RotateTransform3D(ax3d);
    //создание переносов центра ландшафта в центр системы координат и обратно
    TranslateTransform3D tr1 = new TranslateTransform3D(-N/2, 0, -N/2);
    TranslateTransform3D tr2 = new TranslateTransform3D(N/2, 0, N/2);

    Transform3DGroup tg = new Transform3DGroup();

    //комбинирование преобразований
    tg.Children.Add(tr1);
    tg.Children.Add(rt);
    tg.Children.Add(tr2);

    terrain.Transform = tg;
}
```

Если всё сделано правильно, то при нажатии стрелок влево и вправо, модель ландшафта будет вращаться влево и вправо вокруг своего центра.

Список литературы:

Обзор трехмерной графики WPF:
https://docs.microsoft.com/ru-ru/dotnet/framework/wpf/graphics-multimedia/3-d-graphics-overview