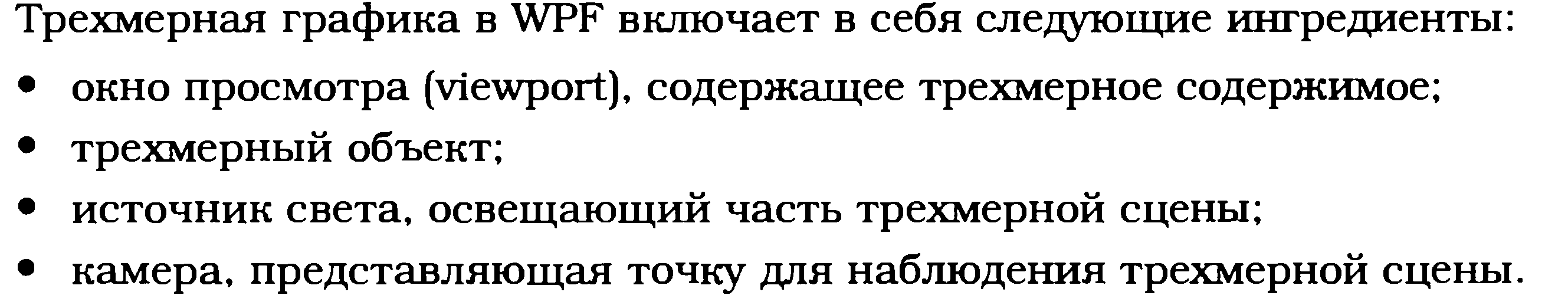
[Макет](javascript:void(0))

Дочерние элементы управления получают доступ к системе макета через базовые классы WPF. Для распространенных макетов, таких как сетки, вложение и закрепление, WPF включает несколько элементов управления макетом.

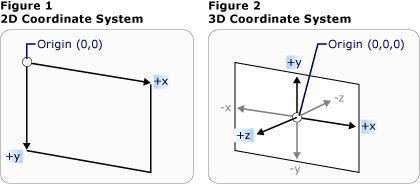
* [Canvas](http://msdn.microsoft.com/ru-ru/library/system.windows.controls.canvas(v=vs.100).aspx) : дочерние элементы управления предоставляют свои собственные макеты.
* [DockPanel](http://msdn.microsoft.com/ru-ru/library/system.windows.controls.dockpanel(v=vs.100).aspx) : дочерние элементы управления выравниваются по краям панели.
* [Grid](http://msdn.microsoft.com/ru-ru/library/system.windows.controls.grid(v=vs.100).aspx) : дочерние элементы управления располагаются по строкам и столбцам.
* [StackPanel](http://msdn.microsoft.com/ru-ru/library/system.windows.controls.stackpanel(v=vs.100).aspx) : дочерние элементы управления располагаются либо горизонтально, либо вертикально.
* [VirtualizingStackPanel](http://msdn.microsoft.com/ru-ru/library/system.windows.controls.virtualizingstackpanel(v=vs.100).aspx) : дочерние элементы управления являются виртуальными и располагаются в одной горизонтальной или вертикальной строке.
* [WrapPanel](http://msdn.microsoft.com/ru-ru/library/system.windows.controls.wrappanel(v=vs.100).aspx) : дочерние элементы управления располагаются в порядке слева-направо и переносятся на следующую строку, когда в текущей строке не хватает места.



Окно просмотра

Графическое содержимое 3-D в приложении WPF инкапсулировано в элементе [Viewport3D](http://msdn.microsoft.com/ru-ru/library/system.windows.controls.viewport3d(v=vs.100).aspx), который может участвовать в структуре двумерного элемента. Графическая система рассматривает [Viewport3D](http://msdn.microsoft.com/ru-ru/library/system.windows.controls.viewport3d(v=vs.100).aspx) как двухмерный визуальный элемент, подобный многим другим в WPF. [Viewport3D](http://msdn.microsoft.com/ru-ru/library/system.windows.controls.viewport3d(v=vs.100).aspx) функционирует как окно — окно просмотра — трехмерной сцены. Говоря точнее, это поверхность, на которую проецируется сцена 3-D.

Представления традиционных двухмерных и трехмерных систем координат



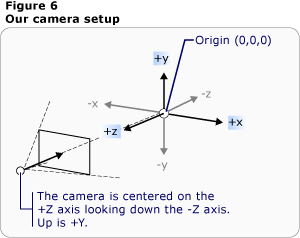
Пространство, определяемое этими осями, является стационарной системой отсчета координат для объектов 3-D в приложении WPF. При построении моделей в этом пространстве и создании источников света и камер для их отображения необходимо отличать стационарную систему отсчета координат (или "мировую систему координат") от локальной системы отсчета, которая создается для каждой модели при применении к ней преобразований. Помните, что в зависимости от настройки освещения и камеры, объекты в мировой системе координат могут выглядеть совсем по-другому или вообще быть невидимыми, но положение камеры не изменяет расположения объектов в мировой системе координат.

Разработчики, работающие в координатах 2-D, привыкли к размещению графических примитивов на двухмерном экране. При создании сцены 3-D важно помнить, что фактически создается представление 2-D объектов 3-D. Поскольку трёхмерная сцена выглядит по-разному в зависимости от точки наблюдения, эту точку наблюдения для сцены необходимо указать. Указать эту точку наблюдения для сцены 3-D позволяет класс [Camera](http://msdn.microsoft.com/ru-ru/library/system.windows.media.media3d.camera(v=vs.100).aspx).

Другой способ понимания того, как представляется сцена 3-D на поверхности 2-D, — это описание сцены как проекции на поверхность просмотра. Камера [ProjectionCamera](http://msdn.microsoft.com/ru-ru/library/system.windows.media.media3d.projectioncamera(v=vs.100).aspx)позволяет указать различные проекции и их свойства для изменения того, как наблюдатель видит модели 3-D. Камера [PerspectiveCamera](http://msdn.microsoft.com/ru-ru/library/system.windows.media.media3d.perspectivecamera(v=vs.100).aspx) указывает проекцию сцены в перспективе. Другими словами, камера [PerspectiveCamera](http://msdn.microsoft.com/ru-ru/library/system.windows.media.media3d.perspectivecamera(v=vs.100).aspx) предоставляет точку схода перспективы. Можно указать положение камеры в пространстве координат сцены, направление и поле зрения камеры и вектор, определяющий направление "вверх" в сцене. Следующая схема иллюстрирует проекции [PerspectiveCamera](http://msdn.microsoft.com/ru-ru/library/system.windows.media.media3d.perspectivecamera(v=vs.100).aspx).

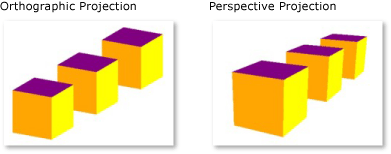
Свойства [NearPlaneDistance](http://msdn.microsoft.com/ru-ru/library/system.windows.media.media3d.projectioncamera.nearplanedistance(v=vs.100).aspx) и [FarPlaneDistance](http://msdn.microsoft.com/ru-ru/library/system.windows.media.media3d.projectioncamera.farplanedistance(v=vs.100).aspx) камеры [ProjectionCamera](http://msdn.microsoft.com/ru-ru/library/system.windows.media.media3d.projectioncamera(v=vs.100).aspx) ограничивают диапазон проекции камеры. Поскольку камеры могут быть расположены в любом месте сцены, фактически можно расположить камеру внутри модели или очень близко от нее, что усложняет правильное распознавание объекта. Свойство [NearPlaneDistance](http://msdn.microsoft.com/ru-ru/library/system.windows.media.media3d.projectioncamera.nearplanedistance(v=vs.100).aspx)позволяет определить минимальное расстояние от камеры, за которым не будут располагаться объекты. И наоборот, свойство [FarPlaneDistance](http://msdn.microsoft.com/ru-ru/library/system.windows.media.media3d.projectioncamera.farplanedistance(v=vs.100).aspx) позволяет задать расстояние от камеры, дальше которого объекты не будут нарисованы; это гарантирует, что объекты, расположенные слишком далеко для распознавания, не будут включены в сцену.

Позиция камеры



Камера [OrthographicCamera](http://msdn.microsoft.com/ru-ru/library/system.windows.media.media3d.orthographiccamera(v=vs.100).aspx) указывает ортогональную проекцию модели 3-D на визуальную поверхность 2-D. Подобно другим камерам, она указывает позицию, направление просмотра и направление "вверх". Однако в отличие от камеры [PerspectiveCamera](http://msdn.microsoft.com/ru-ru/library/system.windows.media.media3d.perspectivecamera(v=vs.100).aspx), камера [OrthographicCamera](http://msdn.microsoft.com/ru-ru/library/system.windows.media.media3d.orthographiccamera(v=vs.100).aspx) описывает проекцию, которая не включает ракурс перспективы.Другими словами, камера [OrthographicCamera](http://msdn.microsoft.com/ru-ru/library/system.windows.media.media3d.orthographiccamera(v=vs.100).aspx) описывает призму, стороны которой параллельны, вместо призмы, стороны которой сходятся в точке камеры. На следующем рисунке показана одна модель, отображенная с использованием камер [PerspectiveCamera](http://msdn.microsoft.com/ru-ru/library/system.windows.media.media3d.perspectivecamera(v=vs.100).aspx) и [OrthographicCamera](http://msdn.microsoft.com/ru-ru/library/system.windows.media.media3d.orthographiccamera(v=vs.100).aspx).

Перспективная и ортогональная проекции



В следующем коде показано несколько обычных параметров камеры.

OrthographicCamera - конструктор (Point3D, Vector3D, Vector3D, Double)

Инициализирует новый экземпляр класса OrthographicCamera с заданной позицией, направлением проекции, направлением вверх и шириной.

Пример

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using System.Windows.Media.Media3D;

using System.Windows.Media.Animation;

namespace WpfApp7

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private void Window\_Loaded(object sender, RoutedEventArgs e)

{

// Declare scene objects.

Viewport3D myViewport3D = new Viewport3D();

Model3DGroup myModel3DGroup = new Model3DGroup();

GeometryModel3D myGeometryModel = new GeometryModel3D();

ModelVisual3D myModelVisual3D = new ModelVisual3D();

OrthographicCamera myOCamera = new OrthographicCamera(new Point3D(0, 0, 8), new Vector3D(0, 0, -3), new Vector3D(0, 1, 0), 8);

myViewport3D.Camera = myOCamera;

DirectionalLight myDirectionalLight = new DirectionalLight();

myDirectionalLight.Color = Colors.White;

myDirectionalLight.Direction = new Vector3D(1, 1, -25);

myModel3DGroup.Children.Add(myDirectionalLight);

// The geometry specifes the shape of the 3D plane. In this sample, a flat sheet

// is created.

MeshGeometry3D myMeshGeometry3D = new MeshGeometry3D();

// Create a collection of normal vectors for the MeshGeometry3D.

Vector3DCollection myNormalCollection = new Vector3DCollection();

myNormalCollection.Add(new Vector3D(0, 1, 0));

myNormalCollection.Add(new Vector3D(0, 1, 0));

myNormalCollection.Add(new Vector3D(0, 1, 0));

myNormalCollection.Add(new Vector3D(0, 1, 0));

myMeshGeometry3D.Normals = myNormalCollection;

// Create a collection of vertex positions for the MeshGeometry3D.

Point3DCollection myPositionCollection = new Point3DCollection();

myPositionCollection.Add(new Point3D(0, 0, 0));

myPositionCollection.Add(new Point3D(1, 0, 0));

myPositionCollection.Add(new Point3D(1, 1, 0));

myPositionCollection.Add(new Point3D(0, 1, 0));

myPositionCollection.Add(new Point3D(0, 0, 1));

myPositionCollection.Add(new Point3D(1, 0, 1));

myPositionCollection.Add(new Point3D(1, 1, 1));

myPositionCollection.Add(new Point3D(0, 1, 1));

myMeshGeometry3D.Positions = myPositionCollection;

// Create a collection of triangle indices for the MeshGeometry3D.

Int32Collection myTriangleIndicesCollection = new Int32Collection();

myTriangleIndicesCollection.Add(4);

myTriangleIndicesCollection.Add(5);

myTriangleIndicesCollection.Add(6);

myTriangleIndicesCollection.Add(6);

myTriangleIndicesCollection.Add(7);

myTriangleIndicesCollection.Add(4);

myMeshGeometry3D.TriangleIndices = myTriangleIndicesCollection;

// Apply the mesh to the geometry model.

myGeometryModel.Geometry = myMeshGeometry3D;

DiffuseMaterial myMaterial = new DiffuseMaterial(Brushes.Red);

myGeometryModel.Material = myMaterial;

RotateTransform3D myRotateTransform3D = new RotateTransform3D(new AxisAngleRotation3D(new Vector3D(1, 1, 0), 0), new Point3D(0, 0, 0));

Transform3DGroup trGrp = new Transform3DGroup();

RotateTransform3D nrt1 = new RotateTransform3D(new AxisAngleRotation3D(new Vector3D(1, 1, 0), 45));

RotateTransform3D nrt2 = new RotateTransform3D(new AxisAngleRotation3D(new Vector3D(1, 1, 0), 45));

TranslateTransform3D trs = new TranslateTransform3D(new Vector3D(0, 0, 0));

trGrp.Children.Clear();

trGrp.Children.Add(nrt1);

trGrp.Children.Add(nrt2);

trGrp.Children.Add(myRotateTransform3D);

trGrp.Children.Add(trs);

myGeometryModel.Transform = trGrp;

DoubleAnimation rotAnimaion = new DoubleAnimation(360, 0, new Duration(TimeSpan.FromSeconds(2)));

rotAnimaion.RepeatBehavior = RepeatBehavior.Forever;

myRotateTransform3D.Rotation.BeginAnimation(AxisAngleRotation3D.AngleProperty, rotAnimaion);

// Add the geometry model to the model group.

myModel3DGroup.Children.Add(myGeometryModel);

// Add the group of models to the ModelVisual3d.

myModelVisual3D.Content = myModel3DGroup;

//

myViewport3D.Children.Add(myModelVisual3D);

// Apply the viewport to the page so it will be rendered.

this.Content = myViewport3D;

}

}

}

<Window x:Class="WpfApp7.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:WpfApp7"

mc:Ignorable="d"

Title="MainWindow" Height="450" Width="800" Loaded="Window\_Loaded"/>