



Useful Tools for Making Video Games

Part V
An overview of



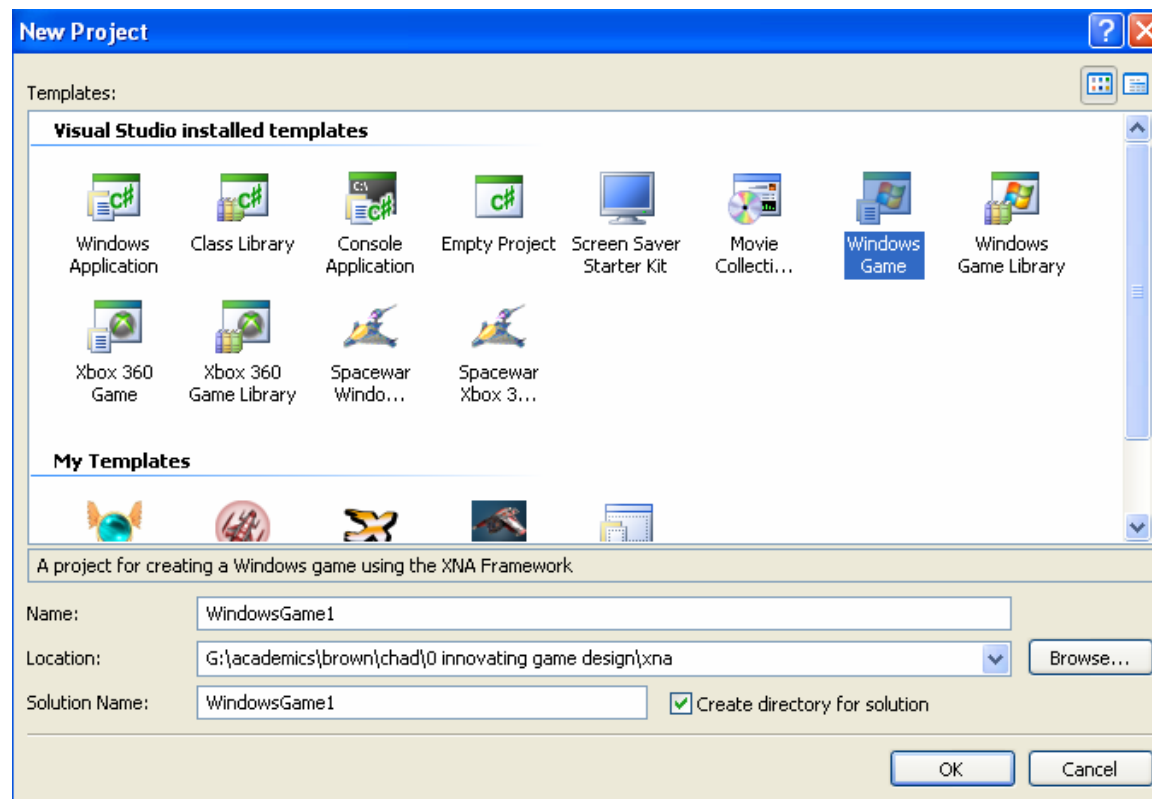


Things you need to install

- Visual Studio 2005 or Visual C# Express Edition
- Its service pack
- XNA Game Studio
- <http://creators.xna.com/Education/newtoxna.aspx>

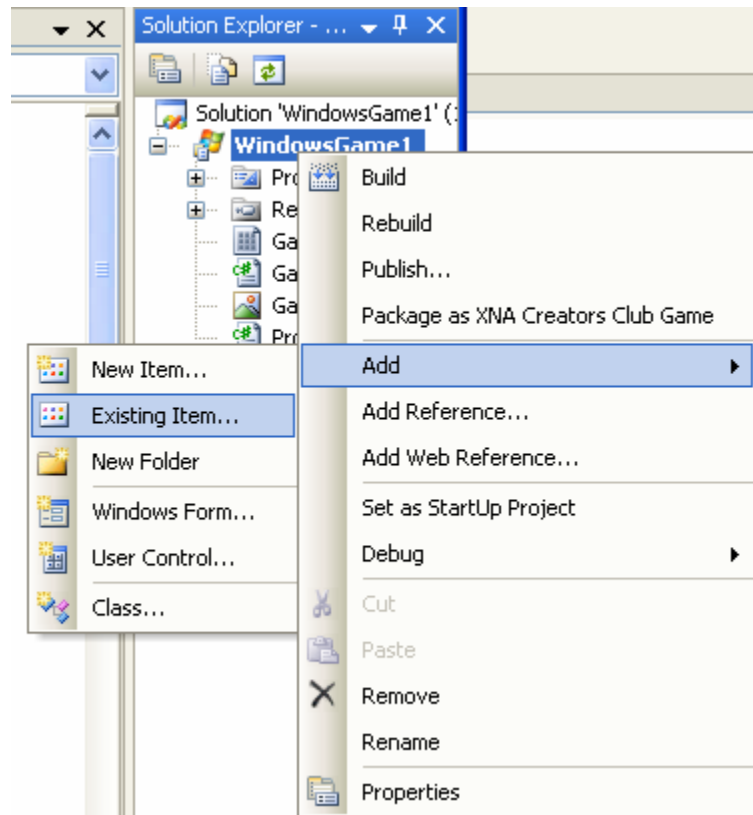
Displaying a 3D Model

- Create a Windows Game project



Displaying a 3D Model

- Add model to solution





Displaying a 3D Model

```
// Set the 3D model to draw.
Model myModel;

// The aspect ratio determines how to scale 3d to 2d projection.
float aspectRatio;

protected override void LoadGraphicsContent(bool loadAllContent)
{
    if (loadAllContent) {
        myModel = content.Load<Model>("myModel");
    }

    aspectRatio = graphics.GraphicsDevice.Viewport.Width /
        graphics.GraphicsDevice.Viewport.Height;
}
```

Displaying a 3D Model

```
// Set the position of the model in world space, and set the rotation.
Vector3 modelPosition = Vector3.Zero;
float modelRotation = 0.0f;

// Set the position of the camera in world space, for our view matrix.
Vector3 cameraPosition = new Vector3(0.0f, 50.0f, 5000.0f);

protected override void Draw(GameTime gameTime) {
    graphics.GraphicsDevice.Clear(Color.CornflowerBlue);

    // Copy any parent transforms.
    Matrix[] transforms = new Matrix[myModel.Bones.Count];
    myModel.CopyAbsoluteBoneTransformsTo(transforms);

    // Draw the model. A model can have multiple meshes, so loop.
    foreach (ModelMesh mesh in myModel.Meshes) {
        // This is where the mesh orientation is set, as well as our camera and projection.
        foreach (BasicEffect effect in mesh.Effects) {
            effect.EnableDefaultLighting();
            effect.World = transforms[mesh.ParentBone.Index] * Matrix.CreateRotationY(modelRotation)
                * Matrix.CreateTranslation(modelPosition);
            effect.View = Matrix.CreateLookAt(cameraPosition, Vector3.Zero, Vector3.Up);
            effect.Projection = Matrix.CreatePerspectiveFieldOfView(MathHelper.ToRadians(45.0f),
                aspectRatio, 1.0f, 10000.0f);
        }
        // Draw the mesh, using the effects set above.
        mesh.Draw();
    }
}
```





Displaying a 3D Model

- Transformation of model done through modifying its world matrix

```
// update rotation angle
protected override void Update(GameTime gameTime)
{
    if (GamePad.GetState(PlayerIndex.One).Buttons.Back ==
        ButtonState.Pressed)
        this.Exit();

    modelRotation +=
        (float)gameTime.ElapsedGameTime.TotalMilliseconds *
            MathHelper.ToRadians(0.1f);

    base.Update(gameTime);
}
```

- Sample implementation can be found in Loading_3D_Model.zip





Keyboard and Mouse input

- In the Update() function check for input

```
KeyboardState keyboardState = Keyboard.GetState();  
if (keyboardState.IsKeyDown( Keys.Left ) {...}
```

- mouseState is pressed as long as you keep your finger on the button, usually you want to detect the event once

```
MouseState mouseState = Mouse.GetState();  
if (mouseState.LeftButton == ButtonState.Released)  
    released = true; // this is a global boolean  
  
if (mouseState.LeftButton == ButtonState.Pressed && released)  
{  
    released = false;  
    System.Console.WriteLine("left mouse clicked\n");  
}
```





Camera

- Determine the location and orientation of the camera object.
- Create a view matrix using the camera position, orientation, and the world space's up vector.
- Create a perspective matrix that determines the near and far clipping planes and the aspect of the projection.
- In the Draw method of game, initialize a BasicEffect object with the transformational matrices created earlier (world, view projection) and then render all existing 3D models.

Camera

- Matrix view = Matrix.CreateLookAt(cameraPosition, cameraLookat, cameraUp);
- Matrix proj = Matrix.CreatePerspectiveFieldOfView(viewAngle, aspectRatio, nearClip, farClip);

```
foreach (ModelMesh mesh in model.Meshes)
{
    foreach (BasicEffect be in mesh.Effects)
    {
        be.Projection = proj;
        be.View = view;
        be.World = world; // eg. Matrix.CreateTranslation(...)
    }
    mesh.Draw();
}
```

- Sample implementation can be found in Camera.zip



Skybox

- apply a skybox-style TextureCube ("cube map") to a sphere using shader

```
Effect SkySphereEffect = Content.Load<Effect>("SkySphere");  
TextureCube SkyboxTexture = Content.Load<TextureCube>("uffizi_cross");
```

```
// Set the parameters of the effect
```

```
SkySphereEffect.Parameters["ViewMatrix"].SetValue(myCamera.ViewMatrix);  
SkySphereEffect.Parameters["ProjectionMatrix"].SetValue(projectionMatrix);  
SkySphereEffect.Parameters["SkyboxTexture"].SetValue(SkyboxTexture);
```

- Since the Effect and the Model are loaded separately, you need to apply the SkySphere effect to each Effect property on the ModelMeshPart of the SkySphere model.
- Sample implementation can be found in skysphere.zip





Animation

- Curve's allows a path to be defined by a small number of control points with the Curve's calculating the points on the path between the control points
- Add Curve3D.cs (found under cs134/lib/xna/content) to your project. Create two instances of this class, one for the position of the camera and the other for the lookAt vector. Also add variable to keep track of time:



Animation

- Define a function called `InitCurve()` and specify the points that the camera will pass through and the points that the camera will be looking at.
- The number of points on the position and the `lookAt` curves don't need to match, but the first and the last points on the curves should have the same time values if the curves oscillate
- calculate the camera's current position and orientation and update the view matrix
- Sample implementation can be found in `Animation.zip`



References

<http://msdn2.microsoft.com/en-us/library/bb198548.aspx>

<http://creators.xna.com/Education/Tutorials.aspx>

<http://creators.xna.com/Education/Samples.aspx>

<http://creators.xna.com/Education/StarterKits.aspx>

<http://forums.xna.com/>