Extending Content Pipeline

Slides built from Carter Chapter 8 and MSDN
Extending Content Pipeline

- Some content doesn’t fall within the normal purview of XNA
  - Maps, etc.
- Extend XNA Content Pipeline so content available everywhere
- We’ll discuss the Carter book example of a skybox
Our Own SkyBox

- What is a skybox?
  - Instead of build geometry for background objects (mountains, sun, moon, etc) lets use texture
  - Can use hemisphere or box, box is easier.
    - How many sides should box have?

- What we need
  - Content Processor
  - Content Compiler
  - Project to contain these
  - ContentReader
  - Demo
New Project

- Create a new Project
- Copy code from last week's sound demo grom game1.cs
- Project -> add reference -> browse and add a reference to XELibrary.dll
Allowing Resize of Window

- Initialize()
  
  Window.AllowUserResizing = true;
  Window.ClientSizeChanged +=
    new System.EventHandler(Window_ClientSizeChanged);

void Window_ClientSizeChanged(object sender, System.EventArgs e)
{
  graphics.PreferredBackBufferWidth = Window.ClientBounds.Width;
  graphics.PreferredBackBufferHeight = Window.ClientBounds.Height;
}
Toggle Fullscreen Mode

- In Update()  // Easy place, should be in input handler

```csharp
KeyboardState kState = Keyboard.GetState();
if (kState.IsKeyDown(Keys.F))
{
    graphics.ToggleFullScreen();
}
```
Creating the Project

A project for creating an XNA Framework 3.0 Content Pipeline Extension Library (.NET Framework 3.5)

Name: ContentPipelineExtension1
Location: C:\Users\king\Desktop\GameProgramming\Projects
Solution: Create new Solution
Solution Name: ContentPipelineExtension1

Template: Visual Studio installed templates
- Windows Game (3.0)
- Xbox 360 Game (3.0)
- Zune Game (3.0)
- Content Pipeline Extension Library (3.0)

Other Templates:
- Windows Game Library (3.0)
- Xbox 360 Game Library (3.0)
- Zune Game Library (3.0)
- Platformer Starter Kit (3.0)
Creating the Project

- File->new project->Content Pipeline Extension Library
  - Name it SkyBox
- Delete the default ContentProcessor
- Add a new class to store the content (next slide)
- Add a new content processor (following)
- Add a new content compiler (following)
Add Class to Store Content

Add New Item - SkyBox

Categories:
- Visual C# Items
  - Code
  - Data
  - General
  - Web
  - Windows Forms
  - WPF
  - Reporting
  - Workflow
  - XNA Game Studio 3.0

Templates:
Visual Studio installed templates
- Application Configuration File
- Assembly Information File
- Class
- Code File
- Cursor File
- DataTable
- HTML Page
- Installer Class
- Invoice File
- Local Database
- MDI Parent Form
- Report Wizard
- Service-based Database

An empty class definition

Name: SkyBoxContent
using System;

namespace SkyboxPipeline
{
    public class SkyboxContent
    {
        public ModelContent Model;
        public Texture2DContent Texture;
    }
}
Add A Content Processor

<table>
<thead>
<tr>
<th>Categories:</th>
<th>Templates:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visual C# Items</td>
<td>LINQ to SQL Classes</td>
</tr>
<tr>
<td></td>
<td>Local Database Cache</td>
</tr>
<tr>
<td></td>
<td>Report</td>
</tr>
<tr>
<td></td>
<td>Resources File</td>
</tr>
<tr>
<td></td>
<td>Settings File</td>
</tr>
<tr>
<td></td>
<td>Text File</td>
</tr>
<tr>
<td></td>
<td>User Control (WPF)</td>
</tr>
<tr>
<td></td>
<td>Windows Form</td>
</tr>
<tr>
<td></td>
<td>Windows Service</td>
</tr>
<tr>
<td></td>
<td>XML Schema</td>
</tr>
<tr>
<td></td>
<td>Content Importer</td>
</tr>
<tr>
<td></td>
<td>Content Type Writer</td>
</tr>
<tr>
<td></td>
<td>My Templates</td>
</tr>
<tr>
<td></td>
<td>Search Online Templates</td>
</tr>
</tbody>
</table>

A class used to apply custom processing to XNA Framework Content Pipeline data

Name: SkyBoxProcessor
ContentProcessor

- Builds 6 sides of a box and textures them.
- Textures stored in one image as:

<table>
<thead>
<tr>
<th>Front</th>
<th>Right</th>
<th>Back</th>
<th>Left</th>
</tr>
</thead>
<tbody>
<tr>
<td>Up</td>
<td>Down</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
namespace SkyBox
{
    /// <summary>
    /// This class will be instantiated by the XNA Framework Content Pipeline
    /// to apply custom processing to content data, converting an object of
    /// type TInput to TOutput. The input and output types may be the same if
    /// the processor wishes to alter data without changing its type.
    ///
    /// This should be part of a Content Pipeline Extension Library project.
    ///
    /// TODO: change the ContentProcessor attribute to specify the correct
    /// display name for this processor.
    /// </summary>
    [ContentProcessor(DisplayName = "SkyBox.SkyBoxProcessor")]
    public class SkyBoxProcessor : ContentProcessor<Texture2DContent, SkyBoxContent>
    {
        private int width = 1024;
        private int height = 512;
        private int cellSize = 256;
    }
}
public override SkyBoxContent Process(Texture2DContent input, ContentProcessorContext context)
{
    // Uncomment to debug (Must have CLR Debugger 2.0 installed)
    MeshBuilder builder = MeshBuilder.StartMesh("XESkybox");

    CreatePositions(ref builder);

    AddVerticesInformation(ref builder);

    // Create the output object.
    SkyBoxContent skybox = new SkyBoxContent();

    // Chain to the ModelProcessor so it can convert the mesh we just generated.
    MeshContent skyboxMesh = builder.FinishMesh();

    MeshBuilder class is designed for content importers to be able to build meshes more easily.
//Now that we have the mesh created, let the default Model processor kick off to actually compile it!
skybox.Model = context.Convert<MeshContent, ModelContent>(skyboxMesh, "ModelProcessor");

skybox.Texture = input;

//If we wanted to do something else with the texture then we could
//create our own texture processor to manipulate the data before storing it
//skybox.Texture = (Texture2DContent)context.Convert<Texture2DContent, TextureContent>(input, "TextureProcessor");

return skybox;
}
private void CreatePositions(ref MeshBuilder builder) {
    //Cube has 8 unique vertices
    Vector3 position;
    //-------front plane
    //top left
    position = new Vector3(-1, 1, 1);
    builder.CreatePosition(position); //0
    //bottom right
    position = new Vector3(1, -1, 1);
    builder.CreatePosition(position); //1
    //bottom left
    position = new Vector3(-1, -1, 1);
    builder.CreatePosition(position); //2
    //top right
    position = new Vector3(1, 1, 1);
    builder.CreatePosition(position); //3
    //-------back plane
    //top left
    position = new Vector3(-1, 1, -1); //4
    builder.CreatePosition(position);
    //bottom right
    position = new Vector3(1, -1, -1); //5
    builder.CreatePosition(position);
    //bottom left
    position = new Vector3(-1, -1, -1); //6
    builder.CreatePosition(position);
    //top right
    position = new Vector3(1, 1, -1); //7
    builder.CreatePosition(position);
}
private Vector2 UV(int u, int v, Vector2 cellIndex)
{
    return (new Vector2(
        (cellSize * (cellIndex.X + u) / width),
        (cellSize * (cellIndex.Y + v) / height)));
}

width = 1024;
height = 512;
cellSize = 256;
private void AddVerticesInformation(ref MeshBuilder builder)
{
    //texture locations:
    //F,R,B,L
    //U,D
    Vector2 fi = new Vector2(0, 0);  //cell 0, row 0  //Front
    Vector2 ri = new Vector2(1, 0);  //cell 0, row 0  //Right
    Vector2 bi = new Vector2(2, 0);  //cell 2, row 0  //Back
    Vector2 li = new Vector2(3, 0);  //cell 3, row 0  //Left
    Vector2 ui = new Vector2(0, 1);  //cell 0, row 1  //Upward (Top)
    Vector2 di = new Vector2(1, 1);  //cell 1, row 1  //Downward (Bottom)

    int texCoordChannel = builder.CreateVertexChannel
                            <Vector2>
                                (VertexChannelNames.TextureCoordinate(0));
ContentProcessor

//-------front plane first column, first row

//bottom triangle of front plane
builder.SetVertexChannelData(texCoordChannel, UV(0, 0, fi));
builder.AddTriangleVertex(4); //-1,1,1
builder.SetVertexChannelData(texCoordChannel, UV(1, 1, fi));
builder.AddTriangleVertex(5); //1,-1,1
builder.SetVertexChannelData(texCoordChannel, UV(0, 1, fi));
builder.AddTriangleVertex(6); //-1,-1,1

//top triangle of front plane
builder.SetVertexChannelData(texCoordChannel, UV(0, 0, fi));
builder.AddTriangleVertex(4); //-1,1,1
builder.SetVertexChannelData(texCoordChannel, UV(1, 1, fi));
builder.AddTriangleVertex(7); //1,1,1
builder.SetVertexChannelData(texCoordChannel, UV(1, 0, fi));
builder.AddTriangleVertex(5); //1,-1,1
//--------right plane
builder.SetVertexChannelData(texCoordChannel, UV(1, 0, ri));
builder.AddTriangleVertex(3);
builder.SetVertexChannelData(texCoordChannel, UV(1, 1, ri));
builder.AddTriangleVertex(1);
builder.SetVertexChannelData(texCoordChannel, UV(0, 1, ri));
builder.AddTriangleVertex(5);
builder.SetVertexChannelData(texCoordChannel, UV(1, 0, ri));
builder.AddTriangleVertex(3);
builder.SetVertexChannelData(texCoordChannel, UV(0, 1, ri));
builder.AddTriangleVertex(5);
builder.SetVertexChannelData(texCoordChannel, UV(0, 0, ri));
builder.AddTriangleVertex(7);
//--------back pane //3rd column, first row
//bottom triangle of back plane
builder.SetVertexChannelData(texCoordChannel, UV(1, 1, bi)); //1,1
builder.AddTriangleVertex(2); //−1,−1,1
builder.SetVertexChannelData(texCoordChannel, UV(0, 1, bi)); //0,1
builder.AddTriangleVertex(1); //1,−1,1
builder.SetVertexChannelData(texCoordChannel, UV(1, 0, bi)); //1,0
builder.AddTriangleVertex(0); //-1,1,1

//top triangle of back plane
builder.SetVertexChannelData(texCoordChannel, UV(0, 1, bi)); //0,1
builder.AddTriangleVertex(1); //1,−1,1
builder.SetVertexChannelData(texCoordChannel, UV(0, 0, bi)); //0,0
builder.AddTriangleVertex(3); //1,1,1
builder.SetVertexChannelData(texCoordChannel, UV(1, 0, bi)); //1,0
builder.AddTriangleVertex(0); //-1,1,1
setContentProcessor

//------left plane
builder.SetVertexChannelData(texCoordChannel, UV(1, 1, li));
bUILDER.AddTriangleVertex(6);
builder.SetVertexChannelData(texCoordChannel, UV(0, 1, li));
bUILDER.AddTriangleVertex(2);
builder.SetVertexChannelData(texCoordChannel, UV(0, 0, li));
bUILDER.AddTriangleVertex(0);

builder.SetVertexChannelData(texCoordChannel, UV(1, 0, li));
bUILDER.AddTriangleVertex(4);
builder.SetVertexChannelData(texCoordChannel, UV(1, 1, li));
bUILDER.AddTriangleVertex(6);
builder.SetVertexChannelData(texCoordChannel, UV(0, 0, li));
bUILDER.AddTriangleVertex(0);
//------- upward (top) plane
builder.SetVertexChannelData(texCoordChannel, UV(1, 0, ui));
builder.AddTriangleVertex(3);
builder.SetVertexChannelData(texCoordChannel, UV(0, 1, ui));
builder.AddTriangleVertex(4);
builder.SetVertexChannelData(texCoordChannel, UV(0, 0, ui));
builder.AddTriangleVertex(0);
builder.SetVertexChannelData(texCoordChannel, UV(1, 0, ui));
builder.AddTriangleVertex(3);
builder.SetVertexChannelData(texCoordChannel, UV(1, 1, ui));
builder.AddTriangleVertex(7);
builder.SetVertexChannelData(texCoordChannel, UV(0, 1, ui));
builder.AddTriangleVertex(4);
CCCCCC

ContentProcessor

    //--------downward (bottom) plane
    builder.SetVertexChannelData(texCoordChannel, UV(1, 0, di));
    builder.AddTriangleVertex(2);
    builder.SetVertexChannelData(texCoordChannel, UV(1, 1, di));
    builder.AddTriangleVertex(6);
    builder.SetVertexChannelData(texCoordChannel, UV(0, 0, di));
    builder.AddTriangleVertex(1);

    builder.SetVertexChannelData(texCoordChannel, UV(1, 1, di));
    builder.AddTriangleVertex(6);
    builder.SetVertexChannelData(texCoordChannel, UV(0, 1, di));
    builder.AddTriangleVertex(5);
    builder.SetVertexChannelData(texCoordChannel, UV(0, 0, di));
    builder.AddTriangleVertex(1);
Add SkyBox Compiler

[Image of a Windows application window showing the process of adding a SkyBox Compiler item to a Visual Studio project. The window is divided into categories and templates. Categories include Visual C# Items and XNA Game Studio 3.0. Templates include Visual Studio installed templates, with options for Content Importer and Content Type Writer highlighted. The window also shows a text field for the name of the new item, currently set to SkyBoxCompiler.cs.]
Next We’ll Create the Compiler

• The job of the compiler is to convert the content and to write it out in the .xnb format.
• Thus it is a ContentTypeWriter class
• The base class should do the bulk of the work for you
SkyBox Compiler or Writer

public class SkyBoxCompiler : ContentTypeWriter<SkyBoxContent> {
    protected override void Write(ContentWriter output, SkyBoxContent value) {
        output.WriteObject(value.Model);
        output.WriteObject(value.Texture);
    }

    public override string GetRuntimeType(TargetPlatform targetPlatform) {
        return "XELibrary.Skybox, " + "XELibrary, Version=1.0.0.0, Culture=neutral";
    }

    public override string GetRuntimeReader(TargetPlatform targetPlatform) {
        return "XELibrary.SkyBoxCompiler, XELibrary, Version 1.0.0.0";
    }
}
Getting Access to Content

- Let’s put the content processing for SkyBox into our existing XELibrary (since we already are using it)
  - So we can add the ContentProcessor and ContentWriter (compiler) to the library
- Next we’ll need a ContentReader
  - Open the XELibrary project
  - Add a new item -> ContentTypeReader (next slide)
Creating the SkyBox Reader

A class used to read data types from the XNA Framework Content Pipeline binary format

Name: SkyBoxReader.cs
The Read method gets the content data along with an existing object we could use to store the data. Here we create a new skybox and pass in the content data to its constructor.
public class Skybox
{
    private Model skyboxModel;
    private Texture2D skyboxTexture;

    // marked as internal since it is only called by our reader
    internal Skybox(ContentReader input)
    {
        skyboxModel = input.ReadObject<Model>();
        skyboxTexture = input.ReadObject<Texture2D>();
    }
}
We also need a draw method to actually render our skybox. It uses the standard cycle through all meshes and use a BasicEffect for each effect in the meshes. The draw method gets passed in a view, projection and world matrix.
Now to Use the SkyBox

- Open up any old game
  - I’ll use last weeks and modify it.
  - Add XELibrary as a reference if not already
- New Variable
  private Skybox skybox;
- Add some SkyBox textures
  - Let’s create a SkyBoxes folder in content
  - Put some images in there
    - Change their content processor to the skybox
    - But first
Adding Content Processor

- Carter book show how to do in 2.0 with double clicking on project properties and going to content pipeline tab
- In 3.0, this doesn’t work. From MSDN you must
  - Right click on content and add reference
  - Go to project with the SkyBox and drill down to the dll
- Now you can change the properties of your image files for the skybox to use the skybox content processor
Using new ContentProcessor
Finish the Code

- LoadContent()
  ```csharp
  skybox = content.Load<Skybox>("@"Content\Skyboxes\skybox")
  ```
- Draw()
  ```csharp
  world = Matrix.CreateScale(5000.0f);
  skybox.Draw(camera.View, camera.Projection, world)
  ```
- `<f5>` and `##@*I`
Debugging the ContentPipeline

- Add Processor to project, set up dependencies in Visual Studio, or make sure you always recompile the contentProcessor and the XELibrary before you test the game.
- At the top of the Process method of SkyBoxProcessor:
  ```csharp
  System.Diagnostics.Debugger.Launch();
  ```
- `<f5>` - step through code. **@#S#F##** no help.
- Do the same thing to the SkyBoxReader in XELibrary.
- `*@##**@#F@#$D@#####`
[ContentTypeWriter]

public class SkyBoxCompiler : ContentTypeWriter<SkyBoxContent> {
    protected override void Write(ContentWriter output, SkyBoxContent value) {
        output.WriteObject(value.Model);
        output.WriteObject(value.Texture);
    }

    public override string GetRuntimeType(TargetPlatform targetPlatform) {
        return "XELibrary.Skybox, " + "XELibrary, Version=1.0.0.0, Culture=neutral";
    }

    public override string GetRuntimeReader(TargetPlatform targetPlatform) {
        return "XELibrary.Skyboxcompiler, XELibrary, Version 1.0.0.0";
    }
}

Attribute to tell Content Pipeline what this class is

All we need to do is write out our data, which in this case is the box and the texture image

Tell the content pipeline the runtime type of this content.

Tell the Content Pipeline which object is to read and process the .xnb file for this content. It is unclear of the format of this string.
And Now
With a Different Texture