
xna
Gamefest
MICROSOFT GAME TECHNOLOGY CONFERENCE 2008
Microsoft

Building Worlds with the Content Pipeline

Shawn Hargreaves
Software Development Engineer
XNA Community Game Platform
Microsoft

Content Pipeline 101

- Add FBX or X file to Visual Studio
- Press F5
- Content.Load<Model>(…)

Content Pipeline 202

- Import from custom file formats
- Apply custom data processing
  - Change materials and shaders
  - Modify textures
  - Generate procedural geometry
- Output custom data types

Content Pipeline 303

- My game is big and complex!
- Levels are more than just graphical models
  - Collision data
  - Spawn points
  - Lighting
  - Sounds
- More content than Visual Studio can handle

Marker Objects

- Annotate levels by adding dummy objects
- Spheres work well
- Identified by name
- Processor extracts game format data
- Processor removes marker geometry

Marker Objects Example

```
[ContentProcessor]
public class SpawnPointProcessor : ModelProcessor
{
    public override ModelContent Process(NodeContent input, …)
    {
        MeshContent marker = FindMeshByName(input, "Spawn");

        if (marker == null)
            throw new InvalidContentException("no marker!");

        Vector3 spawn = ConvertMarkerToPosition(marker);

        ModelContent model = base.Process(input, context);

        model.Tag = spawn;

        return model;
    }
```

Marker Objects Example

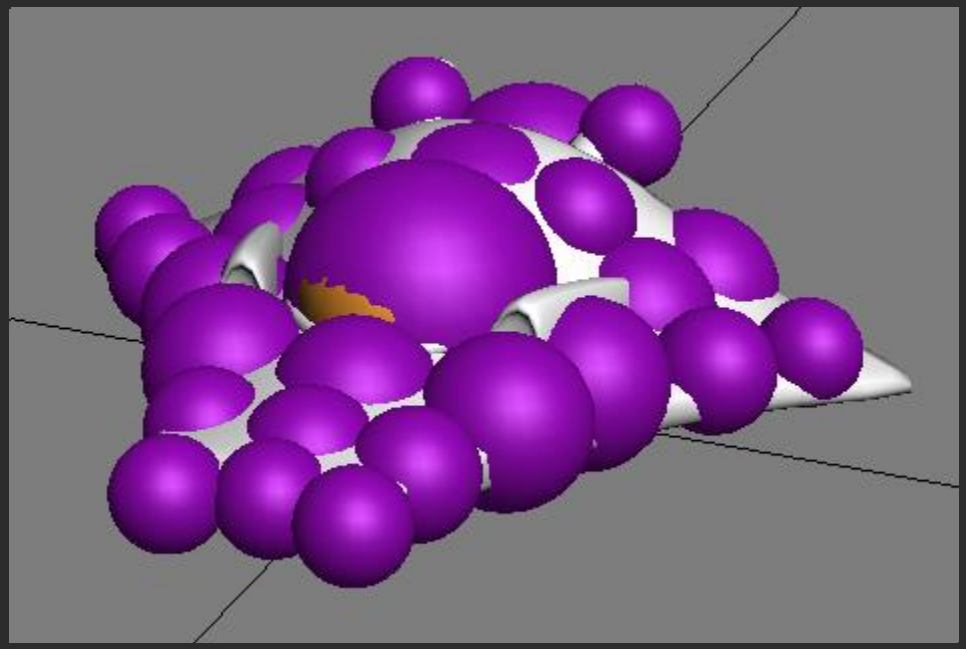
```
MeshContent FindMeshByName(NodeContent node, string name)
{
    if (node.Name == name)
        return node as MeshContent;

    foreach (NodeContent child in node.Children)
    {
        MeshContent found = FindMeshByName(child, name);

        if (found != null)
            return found;
    }

    return null;
}
```

Marker Objects Example

```
Vector3 ConvertMarkerToPosition(MeshContent marker)
{
    BoundingSphere sphere = BoundingSphere.CreateFromPoints(
                                                marker.Positions);

    Vector3 position = Vector3.Transform(sphere.Center,
                                        marker.AbsoluteTransform);

    marker.Parent.Children.Remove(marker);

    return position;
}
```

Collision Spheres

- Approximate collision for complex shapes
- Use any number of marker spheres
- CollisionSphere01, CollisionSphere02, etc.



Parameterized Markers

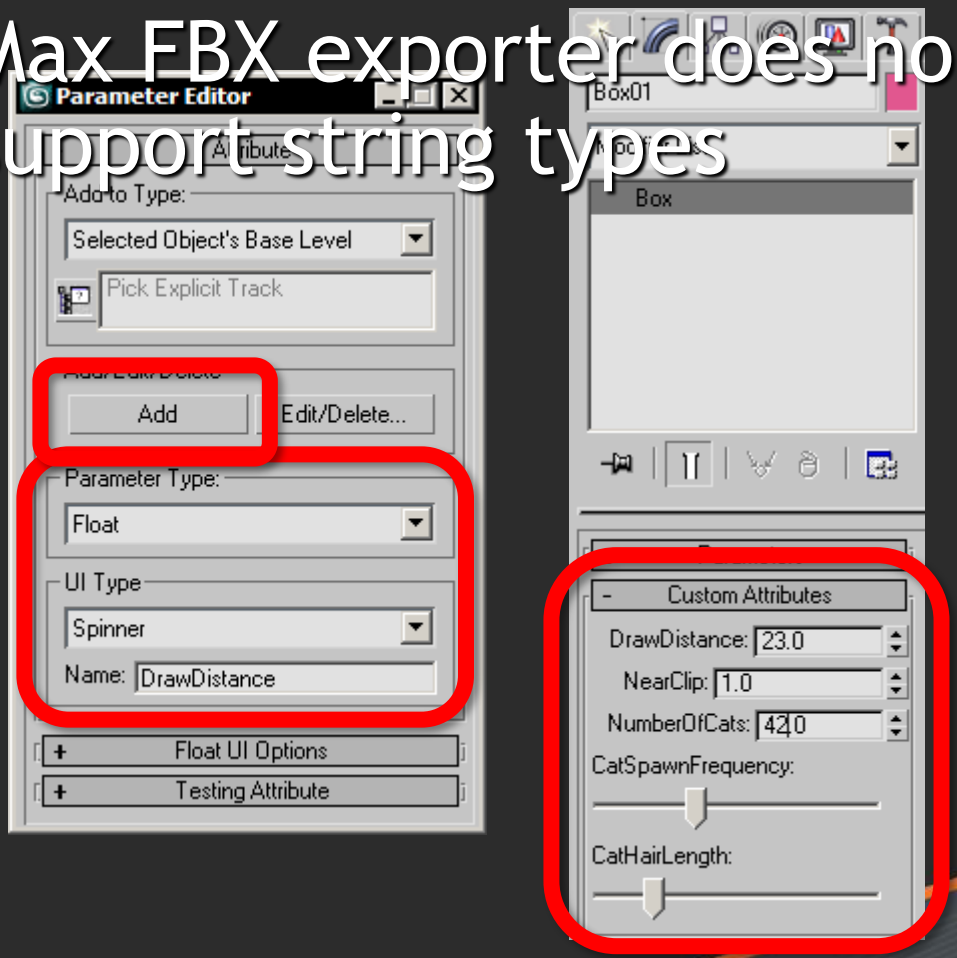
- Encode data into the marker object name
  - '*SpawnPoint Type=Cat Time=23*'
  - Works with all modeling programs
- Marker name could point to an external file
  - '*SpawnPoint CatSpawnSettings.xml*'
  - Processor reads these settings
  - context.AddDependency

Opaque Data

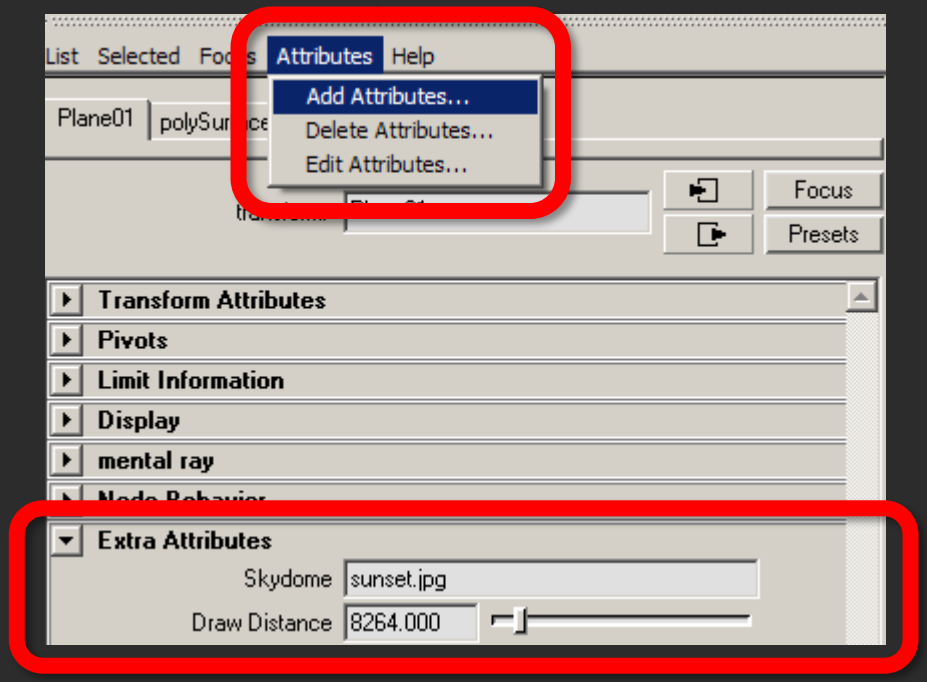
- Attached to marker objects
- Not supported by all modeling packages
- In FBX, but not X

Opaque Data in Max

- Select marker object
- Select Animation / Parameter Editor
- Max FBX exporter does not currently support string types



Opaque Data in Maya



Processing Opaque Data

```
public override ModelContent Process(NodeContent input, …)
{
    NodeContent marker = FindNodeByName(input, "LevelData");

    string skydome = (string)marker.OpaqueData["Skydome"];
    float farClip = (float)marker.OpaqueData["DrawDistance"];
```

What Data Do I Have?

- Look in the obj directory after you build
- Imported content is cached in XML files
- Human readable
  - Node hierarchy
  - Object names
  - Opaque data
- Cache files are useful for tooling
  - IntermediateSerializer reads and writes them
  - ContentPipeline.xml lists them

Cached Content Example

```
<?xml version="1.0" encoding="utf-8"?>
<XnaContent xmlns:Graphics="..." xmlns:Framework="...">
  <Asset Type="Graphics:NodeContent">
    <Name>RootNode</Name>
    <Identity>
      <SourceFilename>C:\gamefest\terrain.fbx</SourceFilename>
      <SourceTool>FbxImporter</SourceTool>
    </Identity>
    <Transform>1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1</Transform>
    <Children>
      <Child Type="Graphics:MeshContent">
        <Name>Spawn</Name>
        <OpaqueData>
          <Data Key="Skydome">sunset.jpg</Data>
          <Data Key="DrawDistance" Type="float">8264</Data>
        </OpaqueData>
        <Transform>2.54 0 0 0 0 1.55525E-16 ...</Transform>
        <Positions>0 0 8.003922 ...</Positions>
        ...
```

One Processor to Rule Them All

- Levels may have multiple source files
  - Different geometry for graphics vs. collision
  - Change sky based on time of day or weather
  - Sky texture shared between multiple levels
- Processors can call into other importers and processors
- A level could just be an XML file that lists what other files to build

ContentProcessorContext

- context.Convert
  - Input: object in memory
  - Output: object in memory
- context.BuildAsset
  - Input: ExternalReference to source asset file
  - Output: ExternalReference to compiled XNB file
  - Enables asset sharing and incremental rebuild
- context.BuildAndLoadAsset
  - Input: ExternalReference to source asset file
  - Output: object in memory

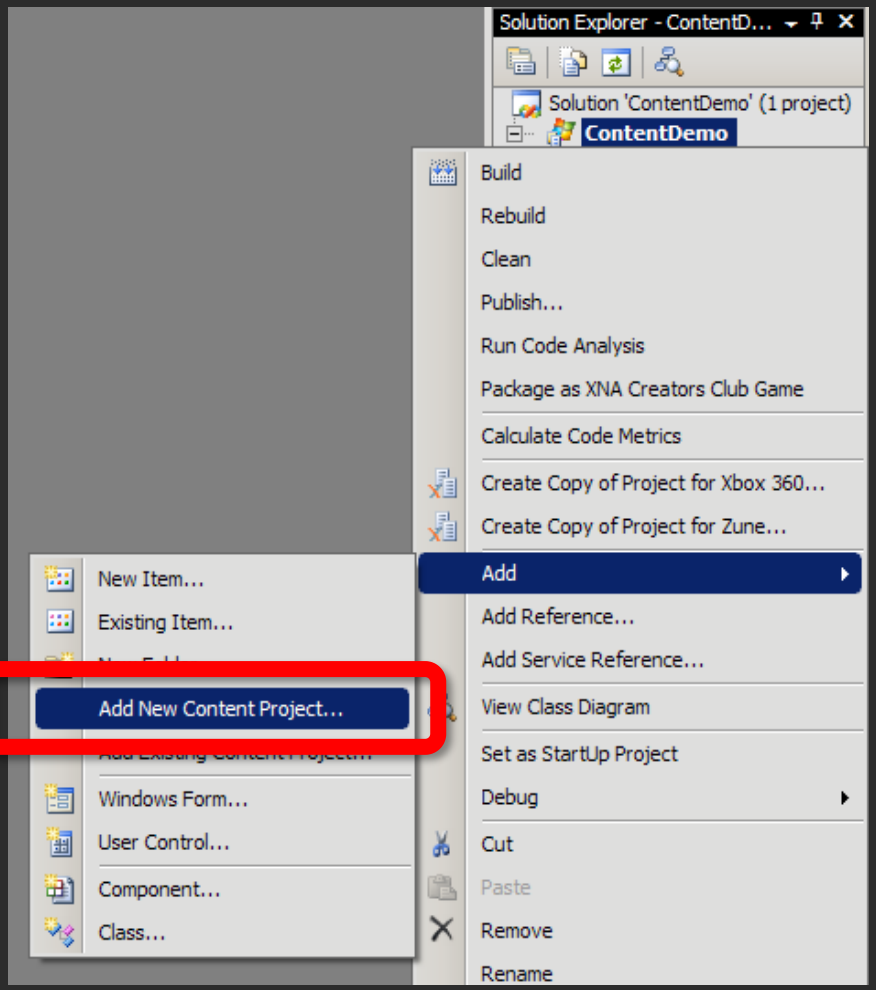
BuildAsset Example

```
NodeContent marker = FindNodeByName(input, "LevelData");

string skyFilename = (string)marker.OpaqueData["Skydome"];

// Create a reference to the source sky texture.
ExternalReference<TextureContent> sourceSky =
    new ExternalReference<TextureContent>(skyFilename,
                                          input.Identity);

// Request that this texture be built into a
// ModelContent, using the custom SkyProcessor.
ExternalReference<ModelContent> compiledSky =
    context.BuildAsset<TextureContent, ModelContent>(sourceSky,
                                                     "SkyProcessor");

model.Tag = compiledSky;
```

And now for something completely different…

Multiple Content Projects

- New in XNA Game Studio 3.0



MSBuild

- The Visual Studio project build system
  - .csproj
  - .vbproj
  - .contentproj
- Works at the command line, too
  - `msbuild MyGame.csproj`
  - `msbuild Content.contentproj /p:Configuration=Debug /p:ParentOutputDir=../bin/x86/Debug/ /p:ParentProjectDir=../`

MSBuild Projects

- Properties
  - Scalar values (Configuration=Debug)
- Items
  - Usually file names (Sources=Cat.tga;Dog.tga)
- Targets
  - Define build actions (Build, Rebuild, Clean)
  - Call tasks, passing properties, and item lists
- Tasks
  - Functions, usually written in C#

MSBuild Example

```
<Project xmlns="http://schemas.microsoft.com/developer/msbuild/2003">

  <PropertyGroup>
    <Purr>true</Purr>
  </PropertyGroup>

  <ItemGroup>
    <Cat Include="Tabby" />
    <Cat Include="Siamese" />
    <Cat Include="Lion" />
  </ItemGroup>

  <Target Name="PrintCats">
    <Message Text="Cats=@(Cat), Purr=$(Purr)" />
  </Target>

</Project>
```

Content Project Example

```
<Project DefaultTargets="Build" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">

  <PropertyGroup>
    <XnaFrameworkVersion>v3.0</XnaFrameworkVersion>
    <XnaPlatform>Windows</XnaPlatform>
    <OutputPath>bin</OutputPath>
  </PropertyGroup>

  <ItemGroup>
    <Reference Include="Microsoft.Xna.Framework.Content.Pipeline.FBXImporter" />
    <Reference Include="Microsoft.Xna.Framework.Content.Pipeline.TextureImporter" />
  </ItemGroup>

  <ItemGroup>
    <Compile Include="terrain.fbx">
      <Name>terrain</Name>
      <Importer>FbxImporter</Importer>
      <Processor>ModelProcessor</Processor>
    </Compile>
  </ItemGroup>

  <Import Project="$(MSBuildExtensionsPath)\Microsoft\XNA Game Studio\v3.0\
                   Microsoft.Xna.GameStudio.ContentPipeline.targets"/>

</Project>
```

Advanced MSBuild

- MSBuild can do things Visual Studio cannot
  - No need to be limited by UI features
- Edit the project XML by hand
  - The UI will not understand your changes
  - Building inside Visual Studio still works

MSBuild Conditionals

```
<ItemGroup>

  <Compile Include="terrain.fbx"
           Condition="$(Configuration)==Debug">
    <Name>terrain</Name>
    <Importer>FbxImporter</Importer>
    <Processor>MyTotallyAwesomeDebugModelProcessor</Processor>
  </Compile>

  <Compile Include="terrain.fbx"
           Condition="$(Configuration)==Release">
    <Name>terrain</Name>
    <Importer>FbxImporter</Importer>
    <Processor>ModelProcessor</Processor>
  </Compile>

</ItemGroup>
```

MSBuild Wildcards

```
<ItemGroup>
  <WildcardContent Include="*.fbx">
    <Importer>FbxImporter</Importer>
    <Processor>ModelProcessor</Processor>
  </WildcardContent>
</ItemGroup>

<Import Project="$(MSBuildExtensionsPath)\Microsoft\XNA Game Studio\v3.0\
                 Microsoft.Xna.GameStudio.ContentPipeline.targets"/>

<Target Name="BeforeBuild">
  <CreateItem Include="@(WildcardContent)"
              AdditionalMetadata="Name=%(FileName)">
    <Output TaskParameter="Include" ItemName="Compile" />
  </CreateItem>
</Target>
```

MSBuild Automation

- Programmatically create a .contentproj, then invoke MSBuild to build it
- Create temporary projects in memory using the MSBuild object model
- Capture errors and warnings
- Import models into level editor
- Note: Content Pipeline not part of redist

http://creators.xna.com/en-us/sample/winforms_series2

Gamefest

MICROSOFT GAME TECHNOLOGY CONFERENCE 2008

www.xnagamefest.com