Model

305890
Spring 2012
4/16/2012
Kyoung Shin Park

Overview

- Model class
  - represents a 3D model composed of multiple ModelMesh objects which may be moved independently.
- ModelMesh class
  - represents a mesh that is part of a Model.
- ModelMeshPart class
  - represents a batch of geometry information to submit to the graphics device during rendering. Each ModelMeshPart is a subdivision of a ModelMesh object.
- To learn how to load the data of an .X file into a Model object and render a 3D model.

Model Class

- Bones property
  - Gets a collection of ModelBone objects which describe how each mesh in the Meshes collection for this model relates to its parent mesh.
- Meshes property
  - Gets a collection of ModelMesh objects which compose the model. Each ModelMesh in a model may be moved independently and may be composed of multiple materials identified as ModelMeshPart objects.

Model Class

- CopyAbsoluteBoneTransformsTo method
  - Copies a transform of each bone in a model relative to all parent bones of the bone into a given array.
  - When using more complicated models, which often use hierarchical structure (where mesh positions, scales, and rotations are controlled by "bones"), this method ensures that any mesh is first transformed by the bone that controls it, if such a bone exists. The mesh is then transformed relative to the bone transformation.

SimpleModel

Draw a Model

- Load a model using the XNA Framework Content Pipeline
  - You need some art assets (i.e., a 3D model and an associated texture files), and extract its contents to the project Content directory.
  - Add a model "p1_wedge.fbx" in the Content
  - To load the model by using the Content Pipeline
  - Model model = Content.Load<Model>("Models₩₩p1_wedge");
  - This function can load any of the following model formats: FBX and X.

Draw a Model

- Render a model using the XNA Framework Content Pipeline
  - Create a new private method called DrawModel(Model m)

```
private void DrawModel(Model m)  {
    Matrix[] transforms = new Matrix[m.Bones.Count];
    m.CopyAbsoluteBoneTransformsTo(transforms);
    // Draw the model. A model can have multiple meshes, so loop.
    foreach (ModelMesh mesh in m.Meshes)     {
        foreach (BasicEffect effect in mesh.Effects) {
            effect.EnableDefaultLighting();
            // ... world, view, projection matrix 중간생략
        }
        // Draw the mesh, using the effects set above.
        mesh.Draw();
    }
}
```

Draw a Model with a Custom Effect

- Load and render a model using a custom effect without modifying the Content Pipeline.
  - Add a model "Terrain.fbx" in the Content
  - Load the model, typically using the Content Pipeline

```
Model terrain = Content.Load<Model>("Terrain");
Matrix terrainWorld = Matrix.Identity;
Texture2D terrainTex = Content.Load<Texture2D>("TerrainTex");
```

  - Load the effect, typically using the ContentManager

```
Effect effect = Content.Load<Effect>("CustomEffect");
```

Draw a Model with a Custom Effect

- Iterate through each ModelMeshPart in the model, and assign the effect to the Effect property of the ModelMeshPart

```
public static void RemapModel(Model model, Effect effect)
{
    foreach (ModelMesh mesh in model.Meshes) {
        foreach (ModelMeshPart part in mesh.MeshParts) {
            part.Effect = effect;
        }
    }
}
```

- Draw a model

```
foreach (ModelMesh mesh in terrain.Meshes) {
    foreach (Effect effect in mesh.Effects) {
        mesh.Draw();
    }
}
```

Make a Model Move Using Input

- Connect Xbox360 controller or Use keyboard
- Create variables to turn and move the model
- Take input from the user to control the model

Make a Model Move Using Input

- HandleInput

```
private void CheckKeyboardInput(GameTime gameTime) {
    // 중간생략
    if (currentKeyboardState.IsKeyDown(Keys.Left))
        modelRotation += elapsedTime * 3.0f;
    else if (currentKeyboardState.IsKeyDown(Keys.Right))
        modelRotation -= elapsedTime * 3.0f;
    else if (currentKeyboardState.IsKeyDown(Keys.Up)) {
                modelVelocity = Vector3.Forward * 3.0f;
                modelPosition += modelVelocity;
    }
    else if (currentKeyboardState.IsKeyDown(Keys.Down)) {
                modelVelocity = Vector3.Backward * 3.0f;
                modelPosition += modelVelocity;
    }
}
```

Draw Models

- Load and render various models

```
models[0] = Content.Load<Model>("airplane2");
models[1] = Content.Load<Model>("starfish");
models[2] = Content.Load<Model>("cactus");
```

XNA Skinned Model with Skeletal Animation

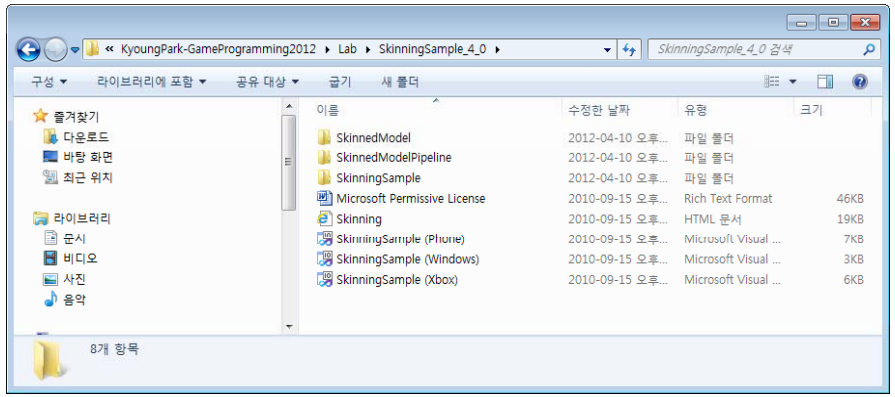
Skinned Model with Skeletal Animation

- Character animation is widely used in video games.
- Skinned mesh with skeleton animation is usually used for character animation.
- This example shows how to create a **skinned mesh content processor**
  - to import a skinned model through the content pipeline,
  - establish skeletal hierarchy,
  - store and use animation information,
  - blend animation's transformations,
  - and play animations.
- A skinned model has a skeleton.
  - This skeleton has a number of bones starting from a root.

http://create.msdn.com/en-US/education/catalog/sample/skinned_model

SkinningSample

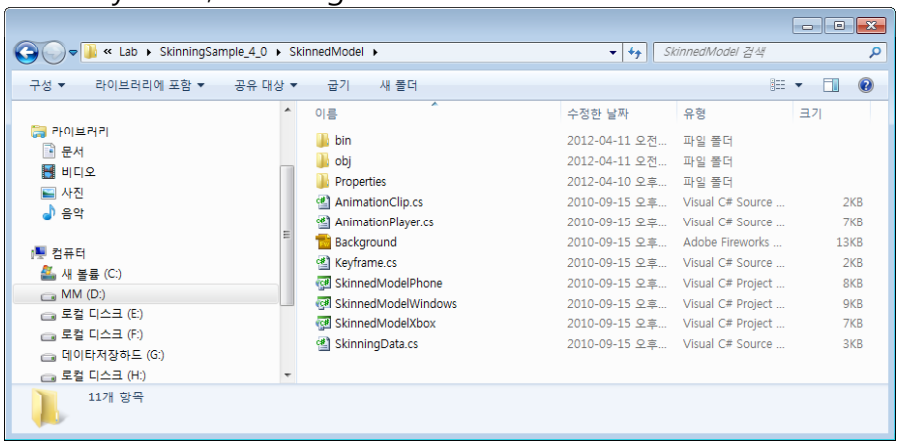
- SkinnedModel Sample



http://create.msdn.com/en-US/education/catalog/sample/skinned_model

SkinningSample

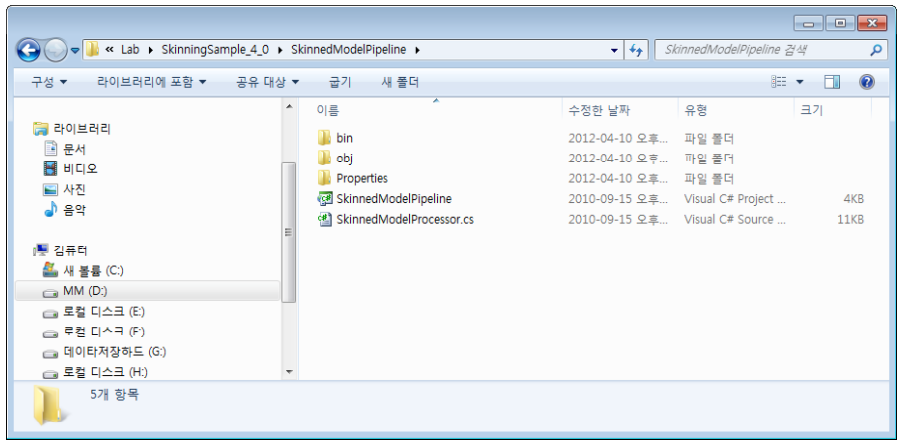
- SkinnedModel – AnimationClip, AnimationPlayer, Keyframe, SkinningData



http://create.msdn.com/en-US/education/catalog/sample/skinned_model

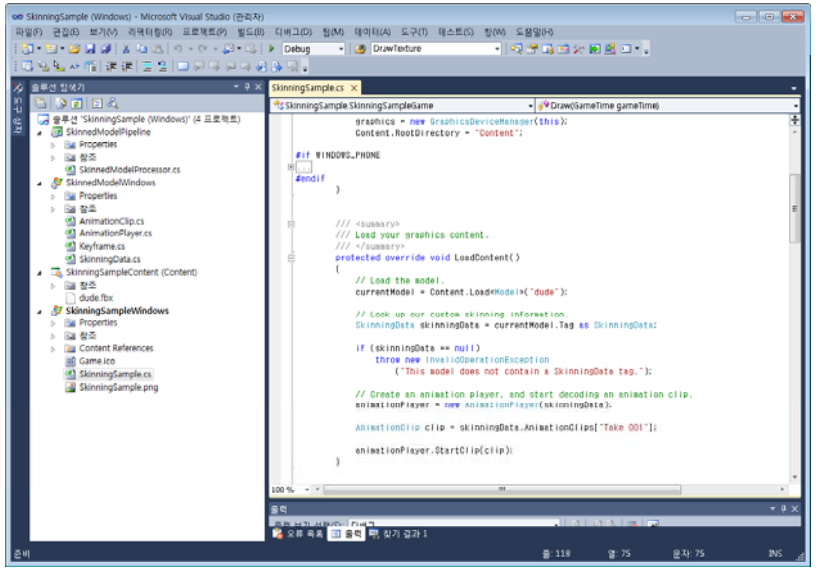
SkinningSample

- SkinnedModelPipeline – SkinnedModelProcessor



http://create.msdn.com/en-US/education/catalog/sample/skinned_model

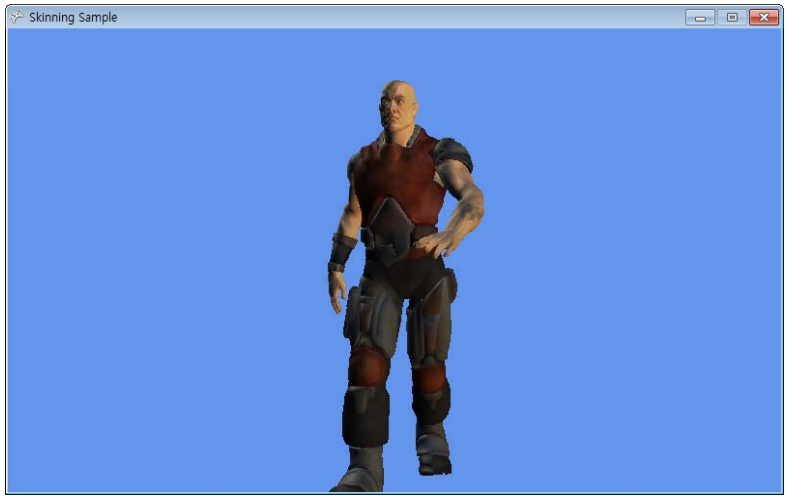
SkinningSample



http://create.msdn.com/en-US/education/catalog/sample/skinned_model

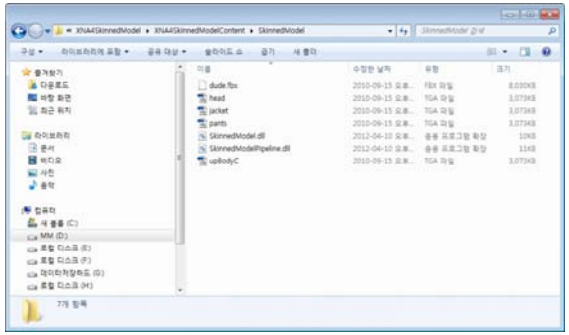
SkinningSample

- SkinningSample – Load dude.fbx



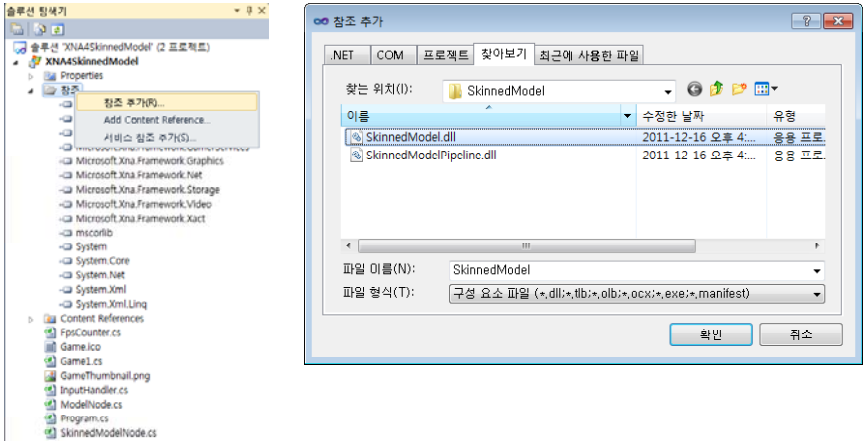
Skinned Model with Skeletal Animation

- Compile SkinningSample_4_0 to get **SkinnedModel.dll** & **SkinnedModelPipeline.dll** at
  - SkinningSample_4_0₩SkinnedModel₩bin₩x86₩Debug
  - SkinningSample_4_0₩SkinnedModelPipeline₩bin₩x86₩Debug
- Copy them into content's SkinnedModel directory



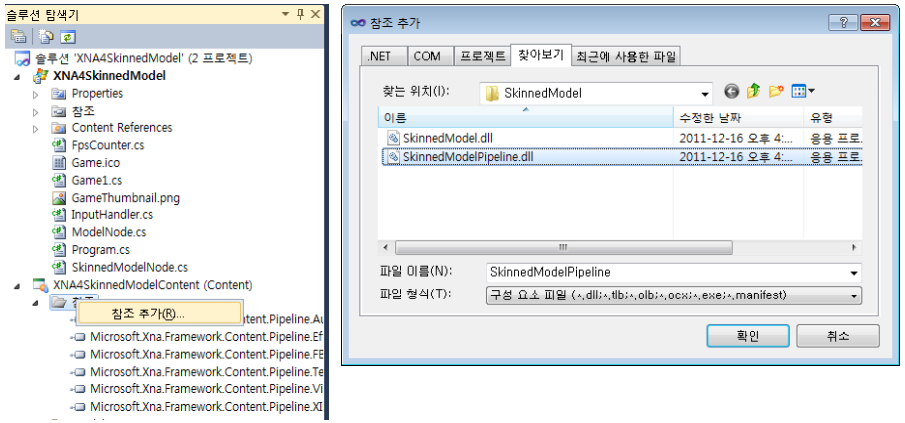
Skinned Model with Skeletal Animation

- Add SkinnedModel.dll into the project as a reference



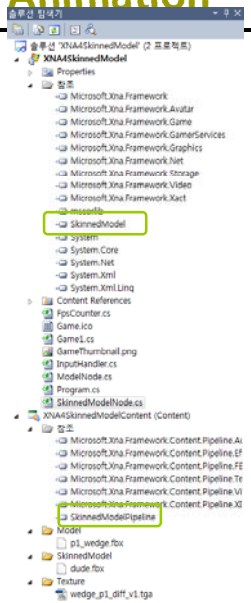
Skinned Model with Skeletal Animation

- Add SkinnedModelPipeline.dll into the content project as a reference



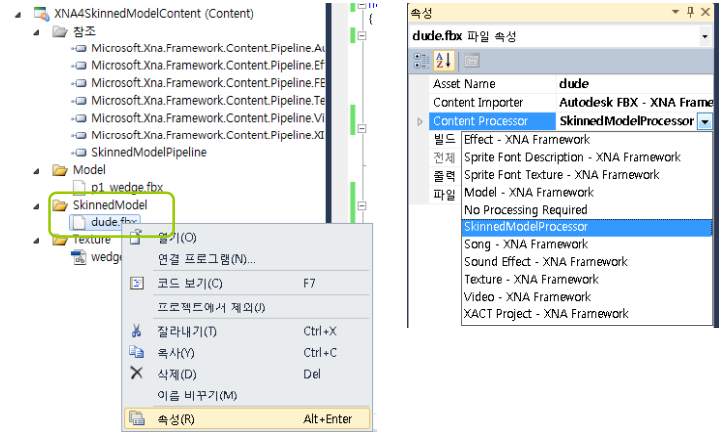
Skinned Model with Skeletal Animation

- Your solution explorer should be like this, showing SkinnedModel.dll & SkinnedModelPipeline



Skinned Model with Skeletal Animation

- Specify SkinnedModelProcessor for dude.fbx Content Processor



Skinned Model with Skeletal Animation

```
SkinnedModelNode.cs
XNA4.SkinnedModelNode
SkinnedModelNode(Game game, string modelAssetName)

using Microsoft.Xna.Framework.Media;
using Microsoft.Xna.Framework.Net;
using Microsoft.Xna.Framework.Storage;
using SkinnedModel; // AnimationPlayer & SkinningData

namespace XNA4
{
    public class SkinnedModelNode : ModelNode
    {
        AnimationPlayer animationPlayer;
        SkinningData skinningData;

        public SkinnedModelNode(Game game, string modelAssetName)
            : base(game, modelAssetName)
        {
        }

        public SkinnedModelNode(Game game, string modelAssetName, Vector3 position, Vector3 scale, Quaternion rotate)
            : base(game, modelAssetName, position, scale, rotate)
        {
            skinningData = this.mesh.Tag as SkinningData;
            if (skinningData == null)
                throw new InvalidOperationException
                    ("This model does not contain a SkinningData tag.");

            animationPlayer = new AnimationPlayer(skinningData);
            AnimationClip clip = skinningData.AnimationClips["ani-1"];
            animationPlayer.StartClip(clip);
        }

        public override void Update(GameTime gameTime)
        {
            animationPlayer.Update(gameTime.ElapsedGameTime, true, Matrix.Identity);
        }

        public override void Draw(GameTime gameTime)
        {
            Matrix[] bones = animationPlayer.GetSkinTransforms();
            foreach (ModelMesh mesh in this.mesh.Meshes)
            {
                Matrix world = Matrix.CreateScale(this.Scale) * Matrix.CreateFromQuaternion(this.Rotation) * this.World * M
```

Skinned Model with Skeletal Animation

- Set a break point and check the *Tag* attribute of model

```
/// <summary>
/// ctor
/// </summary>
public ModelNode(Game game, string modelAssetName)
    : this(game, modelAssetName, Vector3.Zero, Vector3.One, Quaternion.Identity)
{
}

public ModelNode(Game game, string modelAssetName, Vector3 position, Vector3 scale, Quaternion rotate)
    : base(game)
{
    // model load
    this.name = modelAssetName;
    if (game.Content != null && this.mesh == null && this.name != string.Empty)
        this.mesh = game.Content.Load<Model>(this.name);

    // model tra
    this.Positio
    this.Scale =
    this.Rotatio

    // bounding
    this.volume
}
```

this.mesh {Microsoft.Xna.Framework.Graphics.Model}

Bones	Count = 60
Meshes	Count = 1
Root	{Microsoft.Xna.Framework.Graphics.ModelBone}
Tag	{SkinnedModel.SkinningData}
AnimationClips	Count = 1
[0]	{[Take 001, SkinnedModel.AnimationClip]}
Raw 뷰	
SkeletonHierarchy	Count = 58

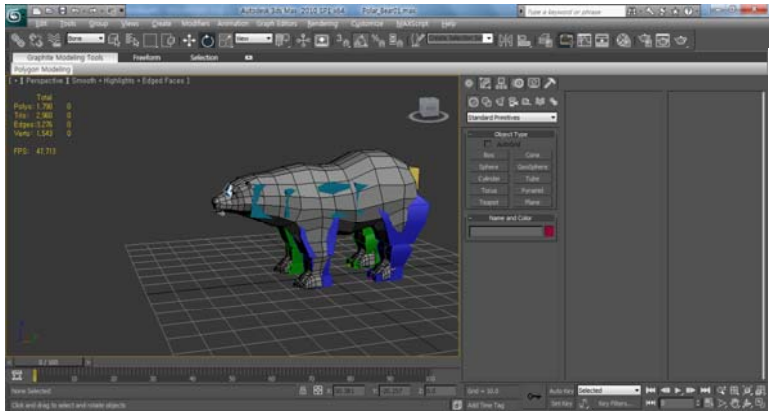
Skinned Model with Skeletal Animation

- SkinnedModel(dude.fbx) & Model(p1_wedge.fbx)

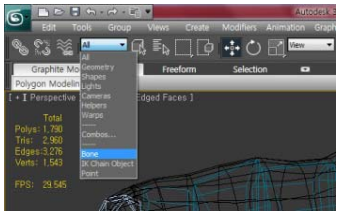
Creating a Skinned Model in 3DS MAX

Create a Skinned Model

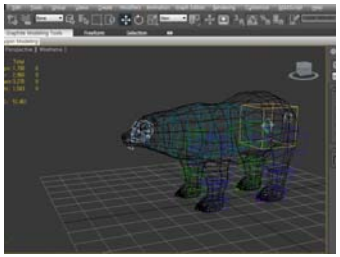
- 3DS Max에 3D model 을 불러온다



Create a Skinned Model



All을 Bone으로 선택한다.
 - 모델링에서 Bone만 선택이 가능하다.



F3을 눌러 Wireframe로 바꿔주면, 모델링 안의 Biped가 보인다.

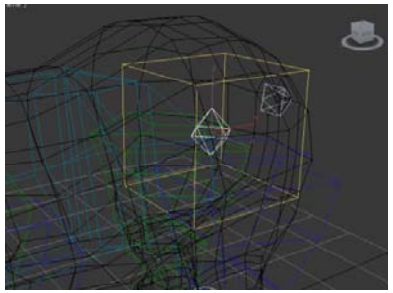
Create a Skinned Model

Time configuration을 누른다

Animation에서 End Time에 원하는 프레임 (시간) 만큼 입력해준다. (30fps = 1초)

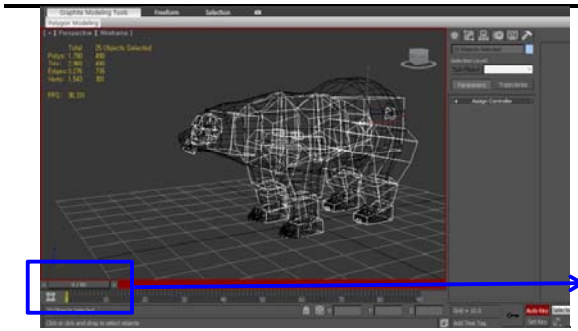
화면 아래쪽의 타임라인이 90fps로 변경된걸 확인 할 수 있다.

Create a Skinned Model



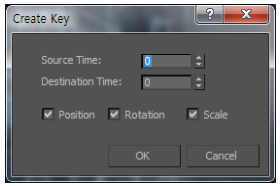
가운데 있는 다이아몬드 형태의 도형을 더블 클릭하여 Biped 전체를 선택한다.

Create a Skinned Model



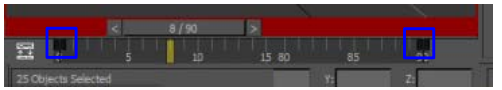
전체가 선택된 상태에서 Auto Key를 눌러 활성화 시킨 후 Time Slider 를 0fps로 이동시킨다.

(Time Slider)

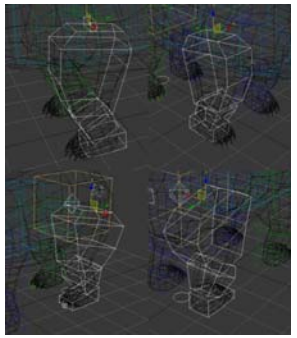


Time Slider를 우 클릭하면, 왼쪽과 같은 화면이 나오는데 Destination Time에 첫 프레임인 0과 마지막 프레임인 90을 입력해준다.
- 각각 해줘야 한다.
- 처음과 마지막을 같은 동작으로 해줘야 자연스럽게 움직이는 화면처럼 보인다.

Create a Skinned Model

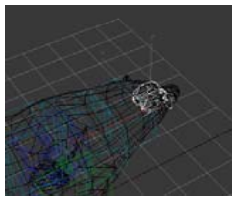


설정이 완료되면 0번과 90번 프레임에 표시가 되는 것을 확인할 수 있다.
앞에서 전체 선택한 Biped에 대한 표시이다.

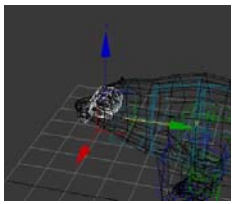


가장 위에 있는 Biped를 더블클릭 할 경우 아래 Biped 까지 선택이 된다.

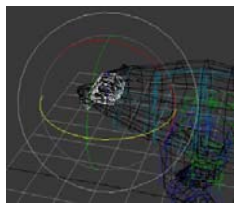
Create a Skinned Model



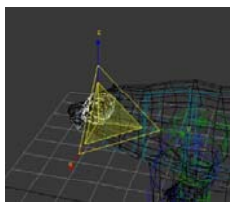
단축키 Q
- 선택



단축키 W
- 이동

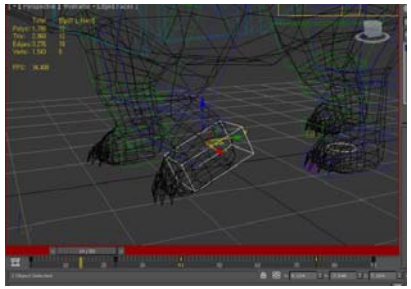


단축키 E
- 회전



단축키 R
- 크기조절

Create a Skinned Model

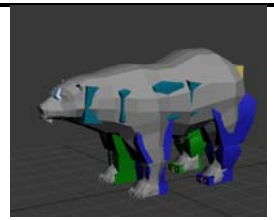


Timeline에서 움직임이 들어갈 프레임으로 위치를 옮긴 후 Biped를 선택, 움직임을 주면 자동으로 프레임에 표시가 된다.

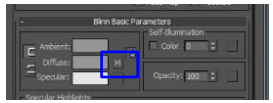
움직임을 확인하는 방법은  버튼을 누르면 재생이 된다.

<단축키>
/ : 재생, 정지 < : 이전 프레임 > : 다음 프레임

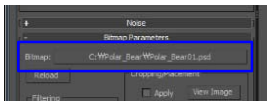
Tip



모델링에 Texture가 적용되지 않았을 경우 M을 눌러 Material Editor 창을 연다.

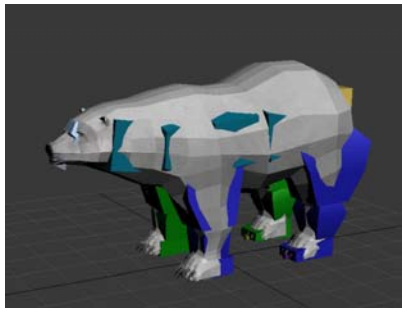


중간에 Diffuse 옆 M이라고 써있는 버튼을 클릭 한다.



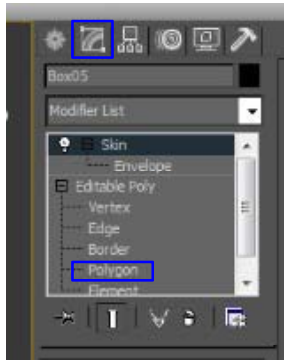
Bitmap 옆에 Texture의 경로가 있는데 버튼을 누른 후 적용하고자 하는 Texture를 찾아 선택해준다.

Tip



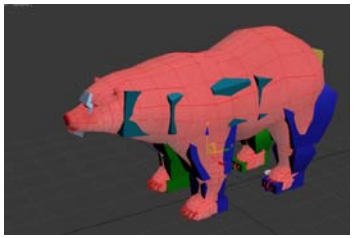
모델링에 Texture가 적용된 모습.

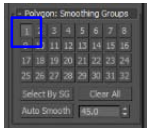
Tip



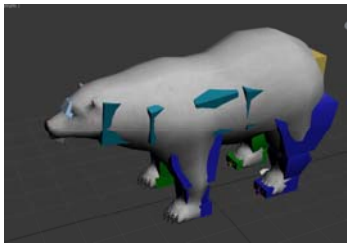
각진 모델링을 부드럽게 해주기 위해서는 smooth를 해줘야 하는데 오른쪽 Toolbar에서 두 번째 메뉴를 선택, 아래 Polygon을 선택 한다.

Tip

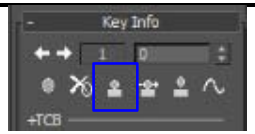




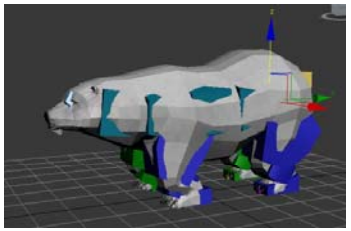
Polygon 전체가 선택된 상태에서 숫자를 누르면 smooth가 적용이 된다.



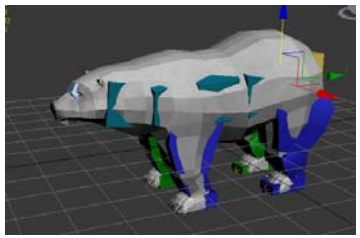
Tip



Key Info 에서 Set Planted Key를 누르면
설정한 지점 아래로 내려가지 않는다.

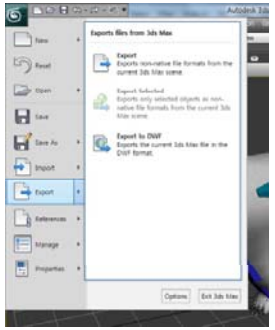


발 모두에 Set Planted Key를
적용한 상태.
몸 전체를 내려도 지면 아래로
발이 내려가지 않고 다리를
굽힌 모습처럼 된다.



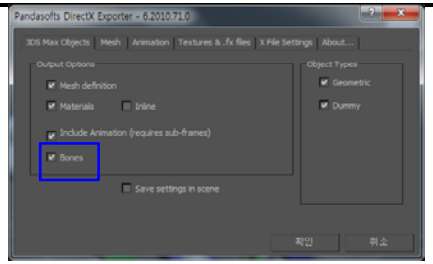
적용을 하지 않은 경우 몸을
내리면 지면 아래로 내려간다.

Export the Model into .X File

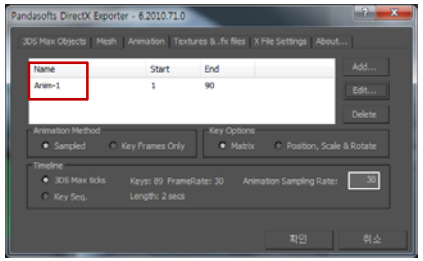


모델링에 움직임을 넣어주는 작업이 끝났으면
파일을 export 해야 한다.
File -> export를 선택한다.
파일 창이 열리면 파일형식을 **Panda DirectX**
(*.X)로 선택 한 후 원하는 위치에 저장을 한다.
(한글이 들어간 폴더는 되도록 피한다.)

Export the Model into .X File



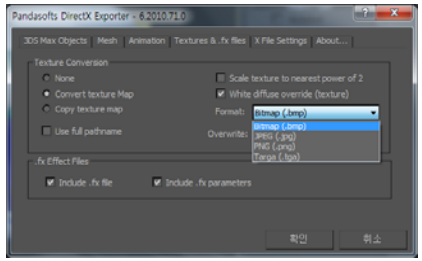
저장을 누르면 옵션 창이
나오는데 왼쪽과 같이 Bones에
체크를 해야 한다. 하지 않을
경우 애니메이션이 없는
모델링만 추출이 된다.



Animation 탭에서는 이름을
기억하고 있어야 한다.
**XNA SkinningSample에서
불러올 때 사용한 이름**을 써야
하기 때문이다.
Start, End는 시작 프레임과
끝나는 프레임을 정해준다.

Edit에서 수정이 가능하다.

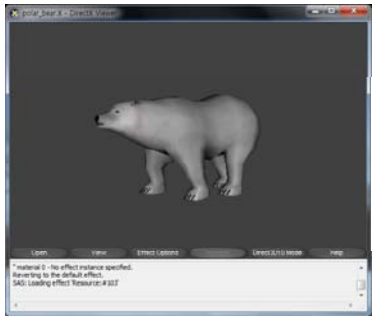
Export the Model into .X File



Textures & .fx files 탭에서는
Texture의 파일 형식 변경이
가능하다.

설정이 끝난 후 확인을 누르면
저장된 위치에 .X 파일과 Texture
파일이 저장된다.

Export the Model into .X File

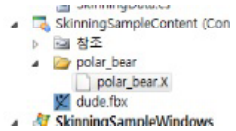


Export한 .x 파일은 **DirectX Viewer**로 확인이 가능하다.

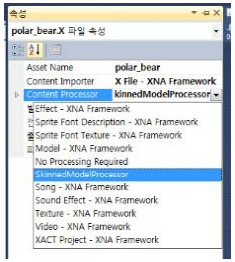
모델링에 Texture가 적용되지 않을 경우 Texture의 이름이 바뀌었거나 위치가 잘못되었기 때문에 위치 설정을 다시 해줘야 한다.

http://create.msdn.com/en-US/education/catalog/sample/skinned_model

Using XNA SkinningSample



앞에서 export한 .x 파일을 Content에서 불러온다.



polar_bear.x 파일을 선택한 후 속성에서 Content Processor을 SkinnedModelProcessor을 선택한다.

Using XNA SkinningSample

SkinningSample.cs 에서 수정 할 부분

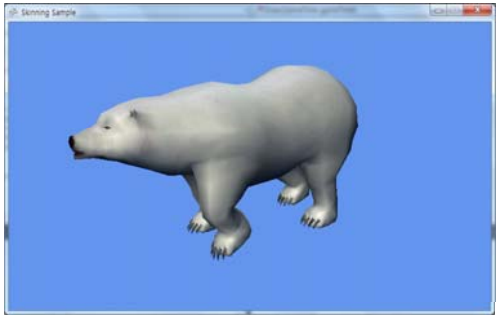
```
protected override void LoadContent()
        {
            // Load the model.
            currentModel = Content.Load<Model>("polar_bear₩₩polar_bear");
           // Load Content에서 북극곰 모델링의 위치를  지정해준다.

            if (skinningData == null)
                throw new InvalidOperationException
                    ("This model does not contain a SkinningData tag.");

            // Create an animation player, and start decoding an animation clip.
            animationPlayer = new AnimationPlayer(skinningData);

            AnimationClip clip = skinningData.AnimationClips["Anim-1"];
           // 앞에서  .x파일을 export할 때 사용한 이름으로 수정한다. (Take 001->Anim-1)
            animationPlayer.StartClip(clip);
        }
```

Using XNA SkinningSample



실행을 하면 움직이는 북극곰을 확인할 수 있다.

Kinect XNA
Skinned Model

Skinned Model + Kinect