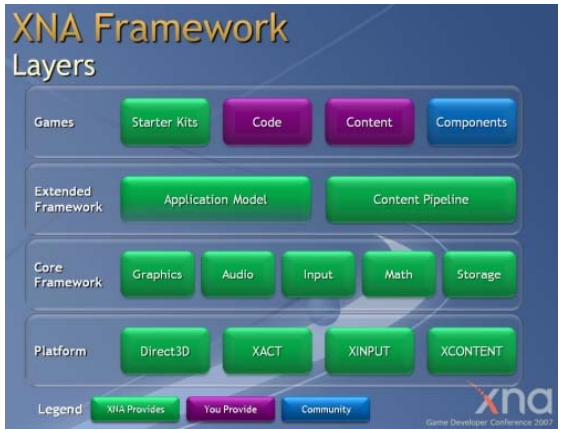
XNA Initialization

305890
Spring 2013
3/8/2013
Kyoung Shin Park

XNA

- Microsoft XNA is a set of tools with a managed runtime environment for computer game development and management.
- XNA Framework is based on native implementation of .NET Compact Framework 2.0 for Xbox 360 development and .NET Framework 2.0 on Windows.
- XNA games can run on any platform that supports the XNA Framework with minimal or no modification.
- But, only C# in XNA Game Studio Express IDE and all versions of Visual Studio 2008 and 2010 (as of XNA 4.0) are officially supported.

XNA Framework



XNA 3.0 => XNA 4.0

What's New in XNA Game Studio 4.0

- Develop Games for Windows Phone 7
- Leverage Windows Phone-specific Features Through Silverlight
- Simplified Graphics Interfaces
  - Reach profile is designed for compatibility across the largest possible range of devices
  - HiDef profile allows you to use platform showcase features
- Configurable Effects
- Built-in State Objects
- System Support for Scalars and Orientation
- Cross-Platform Input API
- Enhanced Audio Support
- Music and Picture Enumeration and Video Playback

XNA 4.0 Profiles

	Reach	HiDef
Supported platforms	Windows Phone 7 Series, Xbox 360, and any Windows PC with a DirectX 9 GPU that supports at least shader model 2.0	Xbox 360, and any Windows PC with a DirectX 10 (or equivalent: see below) GPU
Shader model	2.0 (but Windows Phone does not support custom shaders)	3.0+ (Xbox 360 supports custom shader extensions such as vfetch, which are not available on Windows)
Max texture size	2048	4096
Max cubemap size	512	4096
Max volume texture size	Volume textures are not supported	256
Non power of two textures	Conditional: cannot use wrap addressing mode, mipmaps, or DXT compression when the size is not a power of two	Yes
Non power of two cubemaps	No	Yes
Non power of two volume textures	Volume textures are not supported	Yes
Max primitives per draw call	65535	1048575
Index buffer formats	16 bit	16 and 32 bit
Vertex element formats	Color, Byte4, Single, Vector2, Vector3, Vector4, Short2, Short4, NormalizedShort2, NormalizedShort4	All of the Reach formats, plus HalfVector2, HalfVector4
Texture formats	Color, Bgr565, Bgra5551, Bgra4444, NormalizedByte2, NormalizedByte4, Dxt1, Dxt3, Dxt5	All of the Reach formats, plus Alpha8, Rg32, Rgba64, Rgba1010102, Single, Vector2, Vector4, HalfSingle, HalfVector2, HalfVector4. Floating point texture formats do not support filtering.
Vertex texture formats	Vertex texturing is not supported	Single, Vector2, Vector4, HalfSingle, HalfVector2, HalfVector4
Render target formats	Variable (see below)	Variable (see below)
Multiple render targets	No	Up to 4. Must all have the same bit depth. Supports alpha blending and independent write masks per rendertarget.
Occlusion queries	No	Yes
Separate alpha blend	No	Yes
Blend.SourceAlphaSaturation	Only for SourceBlend, not DestinationBlend	Yes
Max vertex streams	16	16
Max stream stride	255	255

XNA 4.0 Effects

- Basic Effects
  - Contains a basic rendering effect
- Dual Texture Effects
  - Contains a configurable effect that supports two-layer multitexturing
- Alpha Test Effects
  - Contains a configurable effects that supports alpha testing
- Skinned Effects
  - Contains a configurable effect for rendering skinned character models
- Environment Map Effects
  - Contains a configurable effect that supports environment mapping

XNA 4.0 Built-in State Objects

- BlendState
  - Controls how color and alpha values are blended when combining rendered data with existing render target data
- DepthStencilState
  - Controls how the depth buffer and the stencil buffer are used
- RasterizerState
  - Gets/Sets rasterizer state – The default value is RasterizerState.CullCounterClockwise
- SamplerState
  - Contains sampler state, which determines how to sample texture data

Installing XNA

- Visual Studio 2010 Installation & Rebooting
- DirectX9.0c Runtime Installation
  - directx_jun2010_redist.exe
  - http://www.softwarepatch.com/windows/directx.html
- XNA Game Studio 4.0 Installation
  - http://www.microsoft.com/download/en/details.aspx?Id=23714
- Create a new project & Run
  - Visual Studio 2010 메뉴에서 File->New->Projects
  - Visual C#->XNA Game Studio 4.0 ->Windows Game (4.0) 선택
  - 프로젝트 이름 지정

Getting Started with XNA 4.0

- Start a Visual Studio 2010 VC#
- Create a new XNA4.0 project
  - File->New->Projects
  - Visual C#->XNA Game Studio 4.0->Windows Game (4.0)
  - Specify the project name
- Build (F7) & Execute (F5)

Your First XNA Game

- XNA4.0 Example

```
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;
using Microsoft.Xna.Framework.Net;
using Microsoft.Xna.Framework.Storage;
```

```
/// <summary>
/// This is the main type for your game
/// </summary>
public class Game1 : Microsoft.Xna.Framework.Game
{
    GraphicsDeviceManager graphics;
    SpriteBatch spriteBatch;

    public Game1()
    {
        graphics = new GraphicsDeviceManager(this);
        Content.RootDirectory = "Content";
    }
    /// <summary>
    /// Allows the game to perform any initialization it needs to before starting to run.
    /// This is where it can query for any required services and load any non-graphic
    /// related content.  Calling base.Initialize will enumerate through any
    /// components and initialize them as well.
    /// </summary>
    protected override void Initialize()
    {
        // TODO: Add your initialization logic here
        base.Initialize();
    }
```

```
/// <summary>
/// LoadContent will be called once per game and is the place to load
/// all of your content.
/// </summary>
protected override void LoadContent()
{
    // Create a new SpriteBatch, which can be used to draw textures.
    spriteBatch = new SpriteBatch(GraphicsDevice);

    // TODO: use this.Content to load your game content here
}

/// <summary>
/// UnloadContent will be called once per game and is the place to unload
/// all content.
/// </summary>
protected override void UnloadContent()
{
    // TODO: Unload any non ContentManager content here
}
```

```
/// <summary>
/// Allows the game to run logic such as updating the world,
/// checking for collisions, gathering input, and playing audio.
/// </summary>
/// <param name="gameTime">Provides a snapshot of timing values.</param>
protected override void Update(GameTime gameTime)
{
    // Allows the game to exit
    if (GamePad.GetState(PlayerIndex.One).Buttons.Back == ButtonState.Pressed)
        this.Exit();
    // TODO: Add your update logic here
    base.Update(gameTime);
}
/// <summary>
/// This is called when the game should draw itself.
/// </summary>
/// <param name="gameTime">Provides a snapshot of timing values.</param>
protected override void Draw(GameTime gameTime)
{
    GraphicsDevice.Clear(Color.CornflowerBlue);
    // TODO: Add your drawing code here
    base.Draw(gameTime);
}
} // end of Game1 class
```

Your First XNA Game

- Add the 500x500 screen size & "Test1" title

```
protected override void Initialize()
{
        // TODO: Add your initialization logic here
        graphics.PreferredBackBufferWidth = 500;
        graphics.PreferredBackBufferHeight = 500;
        graphics.IsFullScreen = false;
        graphics.ApplyChanges();
        Window.Title = "Test1";

        base.Initialize();
}
```

Your First XNA Game

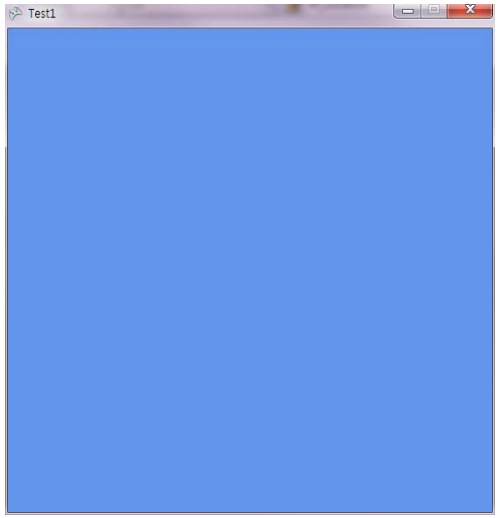
- Add a code for ESC-key to exit program

```
// keyboard & gamepad variables
private KeyboardState currentKeyboardState = new KeyboardState();
private GamePadState currentGamePadState = new GamePadState();

// add HandleInput
protected override void Update(GameTime gameTime)
{
        // Allows the game to exit
        HandleInput();

        // TODO: Add your update logic here
        base.Update(gameTime);
}
```

Your First XNA Game

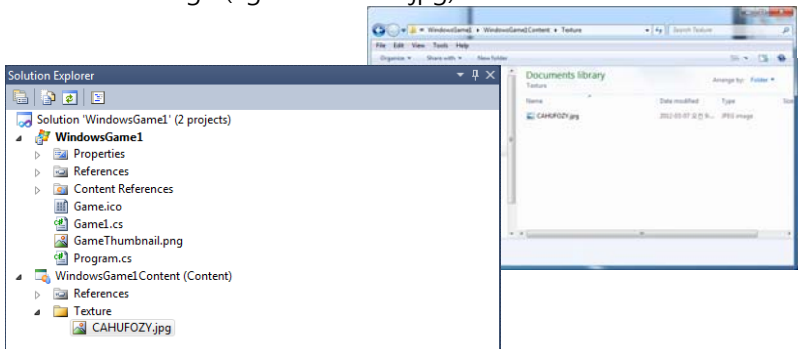
```
// ESC-key 또는 게임패드의 버튼1이 눌렸을 경우 프로그램 종료
#region Handle Input
        /// <summary>
        /// Handles input for quitting the game.
        /// </summary>
        private void HandleInput()
        {
            currentKeyboardState = Keyboard.GetState();
            currentGamePadState = GamePad.GetState(PlayerIndex.One);

            // Check for exit.
            if (currentKeyboardState.IsKeyDown(Keys.Escape) ||
                currentGamePadState.Buttons.Back == ButtonState.Pressed)
            {
                this.Exit();
            }
        }
 #endregion
```

XNA Example



XNA Example

- Add a sprite
  - Add an image (eg: CAHUFOZY.jpg) into the Content



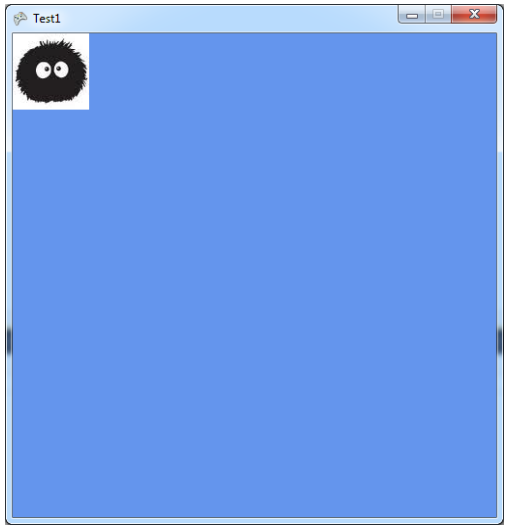
Your First XNA Game

```
// texture
private Texture2D myTexture;   // set texture we can render
private Vector2 spritePosition = Vector2.Zero; // set coordinates

protected override void LoadContent()
{
      // Create a new SpriteBatch, which can be used for draw texture
      spriteBatch = new SpriteBatch(GraphicsDevice);

      // TODO: use this.Content to load your game content here
      myTexture = Content.Load<Texture2D>("Texture₩₩CAHUFOZY");
 }
```

Your First XNA Game

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

    // TODO: Add your drawing code here
    // draw the sprite
    spriteBatch.Begin(SpriteSortMode.BackToFront,BlendState.AlphaBlend);
    spriteBatch.Draw(myTexture, spritePosition, Color.White);
    spriteBatch.End();

     base.Draw(gameTime);
 }
```

Your First XNA Game



Your First XNA Game

- Add the sprite's movement

```
// sprite's motion
private Vector2 spriteSpeed = new Vector2(50.0f, 50.0f);

void UpdateSprite(GameTime gameTime)
{   // Move the sprite by speed, scaled by elapsed time.
    spritePosition += spriteSpeed *
          (float)gameTime.ElapsedGameTime.TotalSeconds;
    int MaxX = graphics.GraphicsDevice.Viewport.Width –
          myTexture.Width;
    int MinX = 0;
    int MaxY = graphics.GraphicsDevice.Viewport.Height - myTexture.Heigh
    int MinY = 0;
```

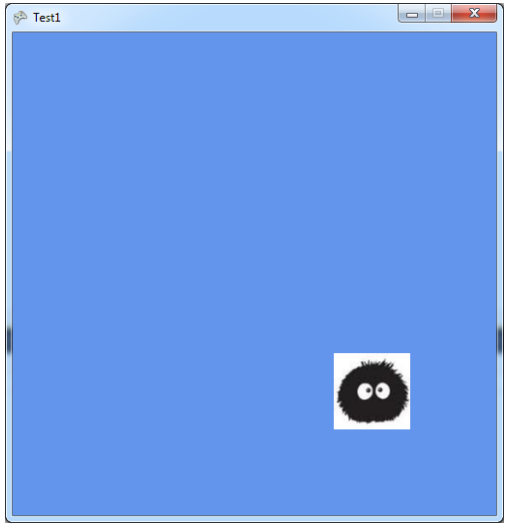
Your First XNA Game

```
    // Check for bounce.
    if (spritePosition.X > MaxX) {
          spriteSpeed.X *= -1;
          spritePosition.X = MaxX;
    } else if (spritePosition.X < MinX) {
          spriteSpeed.X *= -1;
          spritePosition.X = MinX;
    } if (spritePosition.Y > MaxY) {
          spriteSpeed.Y *= -1;
          spritePosition.Y = MaxY;
    } else if (spritePosition.Y < MinY) {
          spriteSpeed.Y *= -1;
          spritePosition.Y = MinY;
    }
}
```

Your First XNA Game

```
protected override void Update(GameTime gameTime)
{
// 중간생략
    // add UpdateSprite
    UpdateSprite(gameTime);
// 중간생략
}
```

Your First XNA Game



What is a Game Loop?

- The ***Game*** class implements a game loop, which provides not only the window which displays your game, but also provides overloadable methods that your game implements to facilitate communication between your game and OS.
- Creating a new game is to make a class that derives from **Game**. The new class needs to override **Update, Draw, Initialize**.
- A fixed-step Game tries to call its Update method on the fixed interval specified in TargetElapsedTime.
- **Game components** provide a modular way of adding functionality to a game.
- **Game services** are a mechanism for maintaining loose coupling between objects that need to interact with each other.

XNA Game Components

- XNA game component allows us to separate pieces of logic into their own file that will be called automatically by the XNA Framework.
- You derive the new component from GameComponent class, or, if the component loads and draws graphics content, from DrawableGameComponent class
- Method
  - Constructor
  - Initialize() – called by the Framework when the component starts
  - Update() – called by the Framework when the component needs to be updated
  - Draw() – called by the Framework when the component needs to be drawn (for only DrawableGameComponent)

XNA Game Components

```
class FpsCounter : Microsoft.Xna.Framework.DrawableGameComponent
{
   FpsCounter(Game game) : base(game) {...}
   Initialize() {... }
   Update(GameTime gameTime) { ... }
   Draw(GameTime gameTime) { ... } // only for
   DrawableGameComponent
}
// Add XNA Game Components
static FpsCounter fpsCounter;
protected override void Initialize()
{ // .. 중간 생략
   fpsCounter = new FpsCounter(this);
   Components.Add(fpsCounter);
}
```

XNA Game Components

- XNA GameComponent class

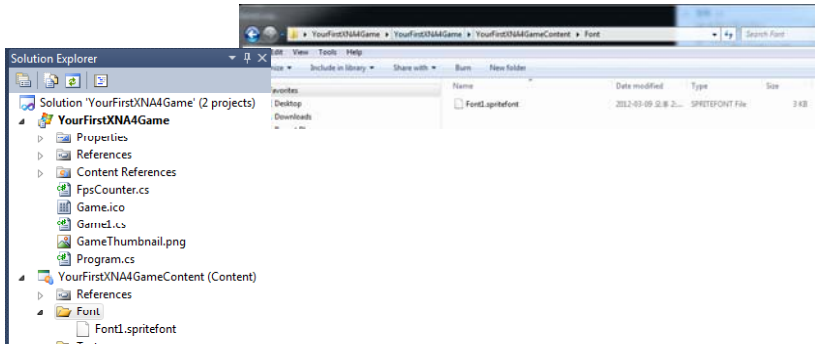
  http://msdn.microsoft.com/en-us/library/microsoft.xna.framework.gamecomponent_members.aspx
- XNA DrawableGameComponent class

  http://msdn.microsoft.com/en-us/library/microsoft.xna.framework.drawablegamecomponent_members.aspx
- Create a XNA GameComponent

  http://msdn.microsoft.com/en-us/library/bb199634(v=xnagamestudio.40).aspx

Your First XNA Game

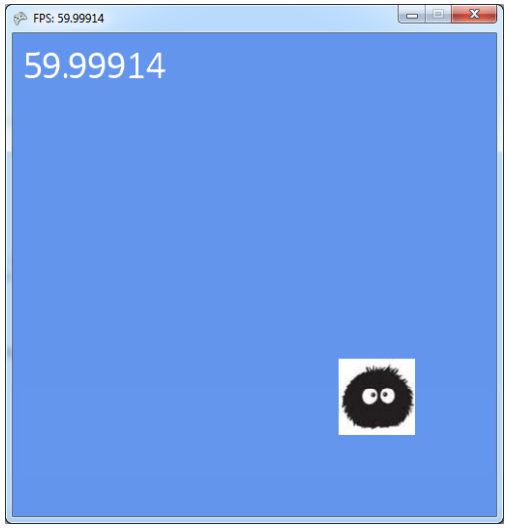
- Add a text for fpsCounter on the screen
  - Add a font (eg: Font1.spritefont) into the Content



Your First XNA Game

```
// font
private SpriteFont font1;
private Vector2 textPosition = new Vector2(10, 10); // set coordinates
protected override void LoadContent()
{      // 중간 생략
       font1= Content.Load<SpriteFont>("Font₩₩Font1");
 }
protected override void Draw(GameTime gameTime)
{ // draw the text
    spriteBatch.Begin();
    spriteBatch.DrawString(font1, fpsCounter.FPS.ToString(),
    textPosition, Color.White);
    spriteBatch.End();
 }
```

Your First XNA Game



Reference

- http://msdn.microsoft.com/en-us/library/bb417503(v=xnagamestudio.40).aspx
- http://msdn.microsoft.com/en-us/library/bb203893(v=xnagamestudio.40).aspx
- http://msdn.microsoft.com/en-us/library/bb203873(v=xnagamestudio.40).aspx