XNA Game Studio

William Howard

CSCI 5448 – Fall 2012

CAETE

Why Object Oriented and Games?

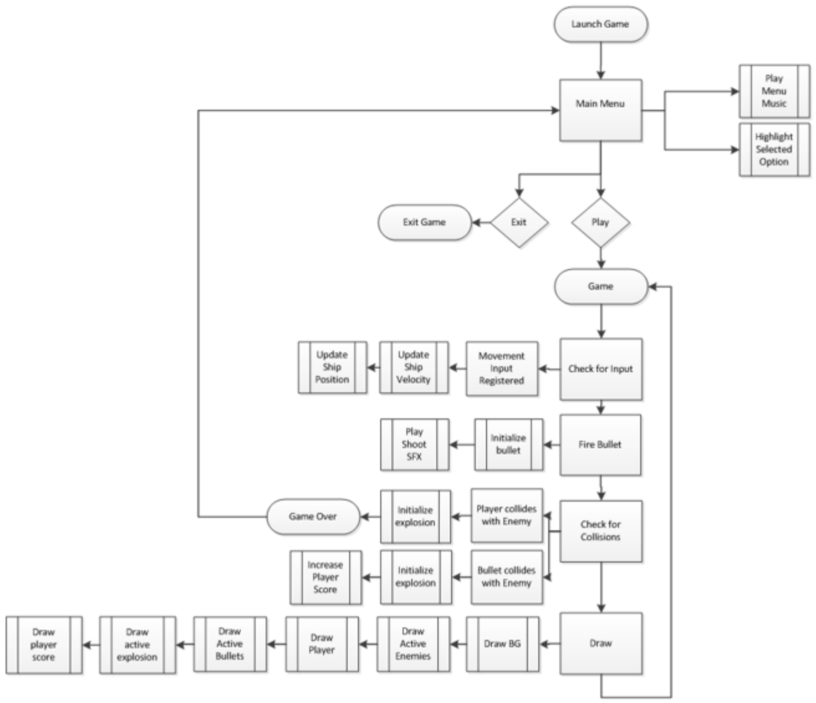
* Everything in games is an object!

Game Objects

- Camera
- World
- Input device (keyboard, gamepad, mouse)
- Player
- Enemies
- Sound effects
- Music
- Splash screens
- Skybox
- Bullets
- Sprites
- Weapons
- Armor
- Backpacks
- Powerups

Game elements

- Game loop
- User interaction
- Graphics
- Sound



XNA Provides:

- An XNA game gets these methods out of the box:
  - Initialize()
  - LoadContent() - Graphics/Sound
  - UnloadContent() - Graphics/Sound
  - Update() - Game Loop/User Interaction
  - Draw() - Graphics/User Interaction

About XNA

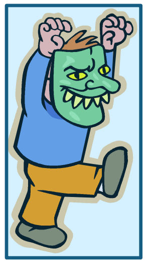
- Extension of .Net Framework
- Visual Studio IDE (Primarily C#)
- Windows
- Windows 7 Phone
- XBOX 360
- Free to develop
- Create.msdn.com/gamedevelopment

History of XNA

- Microsoft wanted a way to create games for Windows
- Needed a way to access the graphics hardware directly to get the speeds necessary for games
- Microsoft created Windows Game SDK to provide game API for Windows 95
- Changed to DirectX after a reporter poked fun at API names DirectDraw, DirectSound, DirectPlay
- Hardware Extraction Layer (HAL) provided API to the hardware drivers
- XNA began as a wrapper for DirectX 9.0

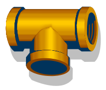
XNA Provides Help With:

- Collision detection
- Bounding rectangles (each object)
- Loading and storing Xml files
- Networking
- Forums support (forums.create.msdn.com)



XNA

- XNA Content Pipeline makes loading assets easy
  - Content.load(“file”)



- Group drawable sprites to increase efficiency
  - spriteBatch.Begin()
- Makes getting input easy
  - KeyboardState
  - GamePadState
  - TouchPanel

XNA Namespaces

* Microsoft.Xna.Framework
* Microsoft.Xna.Framework.Audio
* Microsoft.Xna.Framework.Content
* Microsoft.Xna.Framework.Design
* Microsoft.Xna.Framework.GamerServices
* Microsoft.Xna.Framework.Graphics
* Microsoft.Xna.Framework.Graphics.PackedVector
* Microsoft.Xna.Framework.Input
* Microsoft.Xna.Framework.Input.Touch
* Microsoft.Xna.Framework.Media
* Microsoft.Xna.Framework.Net
* Microsoft.Xna.Framework.Storage

Microsoft.Xna.Framework

- Provides commonly needed game classes such as timers and game loops
- GameComponent - Base class for all XNA Framework game components.
- GameTime – Allows the developer to keep track of real-time or game-time in their games.
- Structures:
  - BoundingBox
  - Vector2
  - Vector3

Microsoft.Xna.Framework.Content

- ContentManager - The ContentManager is the run-time component which loads managed objects from the binary files produced by the design time content pipeline. It also manages the lifespan of the loaded objects, disposing the content manager will also dispose any assets which are themselves IDisposable.
- Content Manager enables developers to easily upload different asset formats without having to worry about the specific format. Simply provide a file, and let XNA worry about what type it is!

Microsoft.Xna.Framework.GamerServices

- GamerServices provides a way for the developer to access information about a user’s XBOX Live account
- Gamer - Abstract base class for types that represent game players (profiles that have an associated gamertag).
- GamerProfile - Profile settings describing information about a gamer such as the gamer's motto, reputation, and gamer picture. This data is accessible for both locally signed in profiles and remote gamers that you are playing with in a multiplayer session.

Microsoft.Xna.Framework.Input

- GamePad - Allows retrieval of user interaction with an Xbox 360 Controller and setting of controller vibration motors.
- Keyboard - Allows retrieval of keystrokes from a keyboard input device.
- Mouse - Allows retrieval of position and button clicks from a mouse input device.

Microsoft.Xna.Framework.Input.Touch

* Contains classes that enable access to touch-based input on devices that support it.
* TouchPanel - Provides methods for retrieving touch panel device information.

XNA Garbage Collection

- .Net Framework managed heap (Windows)
  - 3 generations
- .Net Compact Framework (Xbox 360, Windows Phone)
  - 1 generation

.Net Framework

According to Microsoft:
The .NET Framework is designed to fulfill the following objectives:

- To provide a consistent object-oriented programming environment whether object code is stored and executed locally, executed locally but Internet-distributed, or executed remotely.
- To provide a code-execution environment that minimizes software deployment and versioning conflicts.
- To provide a code-execution environment that promotes safe execution of code, including code created by an unknown or semi-trusted third party.
- To provide a code-execution environment that eliminates the performance problems of scripted or interpreted environments.
- To make the developer experience consistent across widely varying types of applications, such as Windows-based applications and Web-based applications.
- To build all communication on industry standards to ensure that code based on the .NET Framework can integrate with any other code.

.Net Framework

- Whew! That's a lot of words!
- Basically what it's saying is that Microsoft wanted to create a consistent and secure development environment where developers can use the same framework across languages and application uses.
- While there are of course differences in the languages, it is easier to switch between them because you don't have to relearn a new environment.

.Net Objects

* Two types of objects: reference and value
* Value – enums, integer, bool, etc.
* Reference – arrays, classes, attributes, etc.

.Net Properties

- Enable data hiding
  - Accessor methods hide the implementation of the property
  - Referenced the same way a field would be, and getter and setter methods are hidden from the user
- ‘Value’ is keyword in property definition
  - ‘Value’ is assigned to the property in the calling code
- By convention, fields begin with lower-case, properties begin with upper-case.

.Net Properties Example

```
public class SimpleProperty
{
		private int number = 0;
		public int MyNumber
		{
				// Retrieves the data member number.
				get { return number; }

				// Assigns to the data member number.
				set { number = value; }
		 }
}
public class UsesSimpleProperty
{
		public static void Main()
		{
				SimpleProperty example = new SimpleProperty();
				// Sets the property.
				example.MyNumber = 5;

				// Gets the property.
				int anumber = example.MyNumber;
		}
 }
```

Saving Data

- .Net Provides a simple way to save and retrieve data for your game using System.runtime.serialization.formatter.binary
- This saves data in a binary format that allows storage or transmission across a network
- Fast and efficient, but may not work with other serializers

Conclusion

- Because everything in a game is an object, Object Oriented programming is the perfect choice for creating games
- Microsoft XNA provides a great set of abstractions to deal with game objects
- Frameworks like XNA give independent developers a fighting chance to actually write their own games

References

* Msdn.microsoft.com
* Create.msdn.com/gamedevelopment
* Chad Carter, Microsoft XNA Game Studio 3.0 Unleashed
* Kurt Jaegers, XNA 4.0 Game Development by Example
* Jim Perry, RPG Programming using XNA Game Studio 3.0

References

- Clip art provided by Microsoft
- Trashcan icon provided by iCLIPART
- Game Flow Chart is from Create.msdn.com/gamedevelopment
- Demo graphics provided by:
  - Reiner “Tiles” Prokein - http://www.reinerstilesets.de/