

SAPPHIRE

1.0 BETA

DirectX engine written in C# using SlimDX

by Sigurd Suhm

DOCUMENTATION

Content

Introduction.....	3
Setup.....	4
Samples and Tutorials.....	5
Reference.....	6
Sapphire.....	6
DebugText.....	6
Game.....	8
GameComponent.....	11
Sapphire.Graphics.....	11
Sapphire.Input.....	12
Credits.....	13

Introduction

Hello there and thanks for reading and for trying out the Sapphire Engine. As I'm writing this Sapphire is a one man project and an attempt to create a flexible and easy-to-use DirectX 10 engine in managed code (.NET). The whole project is written in C# and it is based on the SlimDX framework which is a pure implementation of the DirectX framework in managed code made by a small independent group.

The reason I chose this approach was that I – first of all – really enjoy the .NET framework and writing code in C# and secondly because I still wanted a low level approach to game programming. I have experience with Microsoft's XNA and there's no doubt that I'm being inspired by that framework in my implementations of high level functionality.

In this documentation I'll store information about all of the content in the Sapphire assembly as well as a few programming and setup guides. The point of the documentation is simply to help people work their way around the entire library. As a supplement everything in the assembly will be documented in XML readable by Visual C#'s Intellisense.

Good luck and may your games be 1337 and numerous.

Sigurd Suhm

SlimDX Web Site: <http://slimdx.mdxinfo.com/>

Setup

You can download the Sapphire Engine from <http://sigurdsuhm.comoj.com/> as a .RAR file which contains 3 files. The first file is this documentation as a .PDF file. The second file (Sapphire.dll) is the actual engine. The third file (Sapphire.xml) is the XML documentation for Visual Studio's Intellisense feature. Make sure this file stays in the same folder as the .DLL.

To make a project using Sapphire create a new project in Visual Studio 2008 or Visual C# 2008. Right click on *References* in your Solution Explorer and choose "Add Reference...". Go to the *Browse* tab and then search your computer for the Sapphire.dll file. After you have added this file as a reference you will have full access to all of the content of the Sapphire Engine. You can now add the Sapphire namespace and the encapsulated namespaces to your code files with a normal *using statement*.

NOTE: Sapphire requires you to install the SlimDX framework so if you haven't installed that yet then go to <http://slimdx.mdxinfo.com/> and download it before you attempt to make a game. You will also need a fair share of the functionality from SlimDX even if you use all of the features of Sapphire.

Samples and Tutorials

Reference

In this section I'll keep a description of all the classes, structs, enumerators and such of the Sapphire assembly. Everything is contained within the Sapphire.dll file and parted into namespaces. All content here is arranged alphabetically within sections for each of the namespaces.

Sapphire

The Sapphire namespace is the wrapping namespace for the entire assembly and contains the core functionality of the engine.

Contains:

- DebugText - Class
- Game - Class
- GameComponent - Class
- GameComponentCollection - Class
- GameClock - Class
- Globals - Class
- MathHelper - Class
- RandomHelper - Class
- SapphireContentException - Class
- SapphireException - Class
- SapphireGraphicsException - Class
- SapphireInputException - Class
- ThreadedGameComponent - Class
- Vector2 - Struct
- Vector3 - Struct
- WindowState - Enumerator

DebugText

```
/// Information text drawn to the screen showing FPS and more.
```

Type: Class

Modifiers: Public
Inherits from: GameComponent
Implements: IDrawable

PROPERTIES

<u>Name</u>	<u>Type</u>	<u>Get/Set</u>
Color	SlimDX.Color4	get, set
	<i>/// Gets or sets the color of the debug text.</i>	
Position	Vector2	get, set
	<i>/// Gets or sets the position of the debug text.</i>	

EVENTS

<u>Name</u>	<u>Type</u>	<u>Modifiers</u>
Disposed	EventHandler	public
	<i>/// Occurs when the object has been disposed.</i>	

METHODS

<u>Name</u>	<u>Return</u>	<u>Arguments</u>	<u>Modifiers</u>
ColorToUInt	uint	System.Drawing.Color color	private
	<i>/// Converts Color to UInt32.</i>		
DebugText	None (ctor)	Game game	public
	<i>/// Default constructor.</i>		
Dispose	void		public, override
	<i>/// Disposes of the DebugText object.</i>		
Draw	void		public
	<i>/// Draws the debug text.</i>		
Update	void	GameTime gameTime	public
	<i>/// Updates the debug text.</i>		

DESCRIPTION

The DebugText class automatically gets instantiated with during the Game class' constructor phase. You can access the instance through a property in the Game class. As of now the DebugText simply draws your framerate onto the screen during each Draw call which happens automatically during Game.Draw(). During the update phase the class fetches the framerate from the GameTime class.

Game

```
/// Basic class for Sapphire games.
```

Type: Class
Modifiers: Public
Inherits from: None
Implements: IDisposable, IDrawable

PROPERTIES

<u>Name</u>	<u>Type</u>	<u>Get/Set</u>
BackBufferHeight	int	get, set
/// Get or set the height of the back buffer.		
BackBufferWidth	int	get, set
/// Get or set the width of the back buffer.		
Camera	Camera	get
/// Gets the main camera of the game.		
ClearColor	SlimDX.Color4	get, set
/// Get or set the color that the screen clears to every frame.		
Components	GameComponentCollection	get
/// List of all game components.		
Content	ContentManager	get
/// Gets the content manager of the game.		
DebugText	DebugText	get
/// Gets the debug text.		
FullScreen	bool	get, set
/// Get or set FullScreen mode.		
GameMainThread	System.Threading.Thread	get
/// Gets the main thread that holds the main loop of the game.		
GameTime	GameTime	get
/// Gets the game time of the game.		
Graphics	GraphicsDeviceManager	get
/// Gets the GraphicsDeviceManager of the game.		

```

Input                InputManager                get
    /// Handler for game input.
ShowDebugText       bool                        get, set
    /// Gets or sets the value for whether the game should show the debug text.
SpriteManager       SpriteManager              get
    /// Gets the game's sprite manager.
SuspendDraw         bool                        get, set
    /// Gets or sets the value for whether the game should skip draw logic.
SuspendUpdate       bool                        get, set
    /// Gets or sets the value for whether the game should skip update logic.
Window              GameWindow                 get
    /// Gets the game window.

```

EVENTS

<u>Name</u>	<u>Type</u>	<u>Modifiers</u>
Activated	EventHandler	public
/// Occurs when the game is activated.		
Deactivated	EventHandler	public
/// Occurs when the game is deactivated.		
Disposed	EventHandler	public
/// Occurs when the game has been disposed.		
Exiting	EventHandler	public
/// Occurs when the game is exiting.		
FrameEnd	EventHandler	public
/// Occurs when the game has drawn a frame.		
FrameStart	EventHandler	public
/// Occurs when a new frame starts.		

METHODS

<u>Name</u>	<u>Return</u>	<u>Arguments</u>	<u>Modifiers</u>
BeginDraw	void		private
/// Clears the screen.			
BeginUpdate	void	GameTime gameTime	private
/// Updates priority components.			
CommonConstructorTasks			private

```

        void
        /// Tasks shared by all constructors.
Dispose        void                public
        /// Disposes of the game.
Draw           void                public,
        virtual
        /// Draw logic.
Exit           void                public
        /// Exits the game.
Game           None (ctor)         public
        /// Default constructor.
Game           None (ctor)         int windowWidth,        public
        int windowHeight
        /// Constructor setting the size of the game window.
GameLoop       void                private
        /// Game loop.
Initialize     void                protected,
        virtual
        /// Initialization logic.
PreInitialize  void                private
        /// Performs tasks before the actual initialize method.
Run            void                public
        /// Initializes the game and starts the game loop.
Update         void                gameTime gameTime        protected,
        virtual
        /// Update logic.

```

DESCRIPTION

The Game class is the main component of the Sapphire Engine. To create your own game you make a class that inherits from this class and then override the virtual methods Initialize(), Update() and Draw(). To start the actual game loop simply run the public function Run(). Remember to keep the base.Function() in your overrides or game components that you add won't be updated and drawn. The components declared in this class as private fields (such as Camera and InputManager) are not contained in the collection of components. These will be updated and drawn during the Pre-... methods.

GameComponent

Type: Class
Modifiers: Public, Abstract
Inherits from: None
Implements: IDisposable

EVENTS

<u>Name</u>	<u>Type</u>	<u>Modifiers</u>
Disposed	EventHandler	public
	<i>/// Occurs when the object has been disposed.</i>	
Updated	EventHandler	public
	<i>/// Occurs when the object has updated.</i>	

METHODS

<u>Name</u>	<u>Return</u>	<u>Arguments</u>	<u>Modifiers</u>
Update	void	GameTime gameTime	public, virtual
	<i>/// Update logic.</i>		
Dispose	void		public, virtual
	<i>/// Disposes of the game component.</i>		

Sapphire.Graphics

Contains:

- Camera - Class
- DeviceSettings - Class
- EffectHelper - Class
- GameObject - Class

- GameWindow - Class
- GraphicsDeviceManager - Class
- IDrawable - Interface
- IVertex - Interface
- MeshHelper - Class
- Sprite - Class
- SpriteManager - Class
- VertexPosition - Struct
- VertexPositionColor - Struct

Sapphire.Input

Contains:

- InputManager - Class
- MouseButton - Enumerator

Credits

Even though I'm the only programmer on this project that doesn't mean that it's a solo effort. In this section I'd like to thank everyone that made this project possible.

- Microsoft - For C#, .NET and DirectX which makes game programming easy, flexible and allows for an amazing workflow.
- The SlimDX Group - For integrating DirectX in managed code. Without you this wouldn't have been possible.