Gamefest

MICROSOFT GAME TECHNOLOGY CONFERENCE 2007

Take it to the Next Level

Microsoft

Understanding XNA Framework Performance

Shawn Hargreaves
Software Development Engineer
XNA Community Game Platform
Microsoft

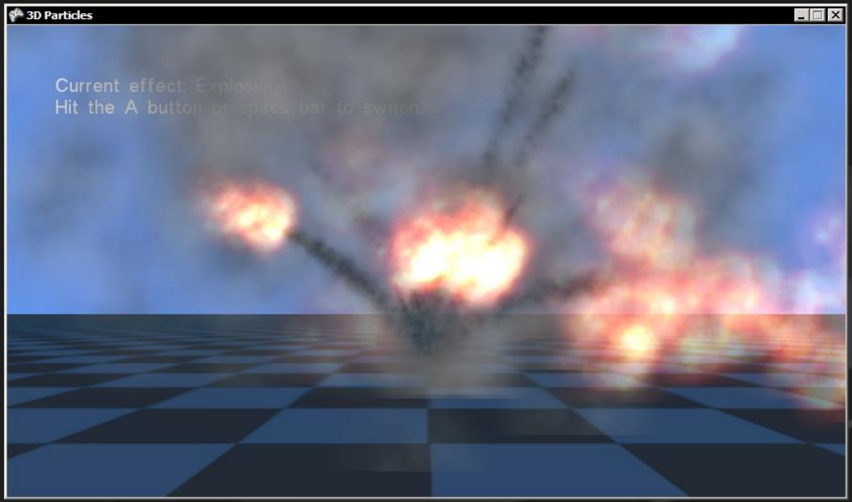
Contents

- Graphics
  - Offload to the GPU
  - Understand Xbox 360 system calls
  - SpriteBatch, Effects, Renderstates
- Math
- Multithreading
- Profiling tools

OFFLOAD TO THE GPU

The GPU Is a Powerful Beastie

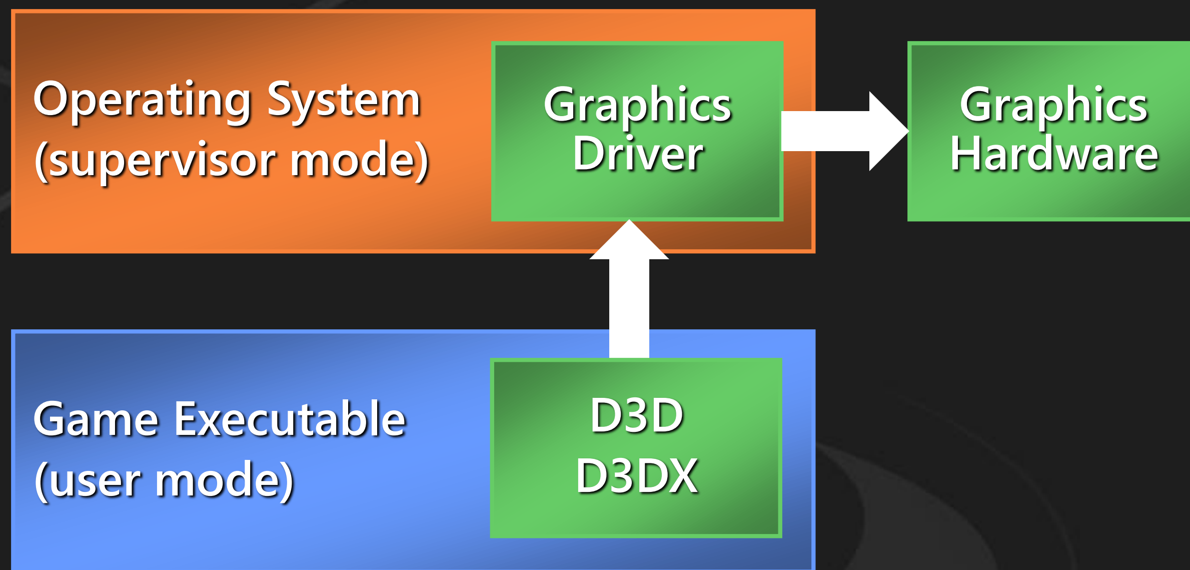
- Offload tasks from CPU to GPU
- Consider GPU instancing
- Particle 3D sample (http://creators.xna.com)
  - GPU effect with low CPU overhead



UNDERSTAND SYSTEM CALLS

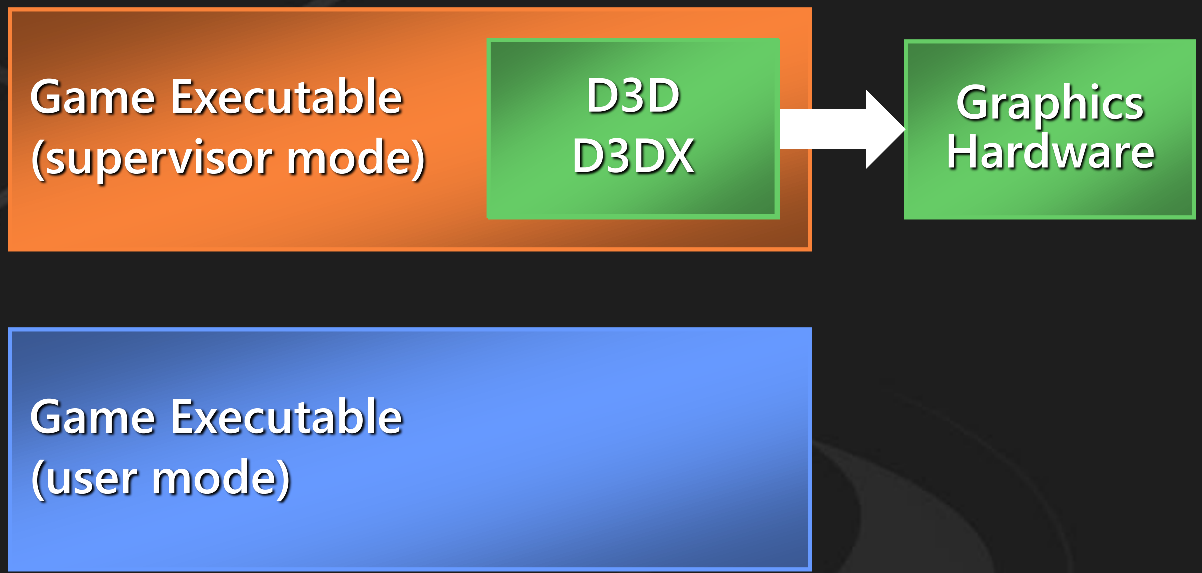
Windows Architecture

User programs cannot directly access hardware



Xbox Architecture

Consoles typically just run everything directly in supervisor mode



- No mode transitions = reduced overhead
- Small batches less expensive than on Windows

Xbox 360 Architecture

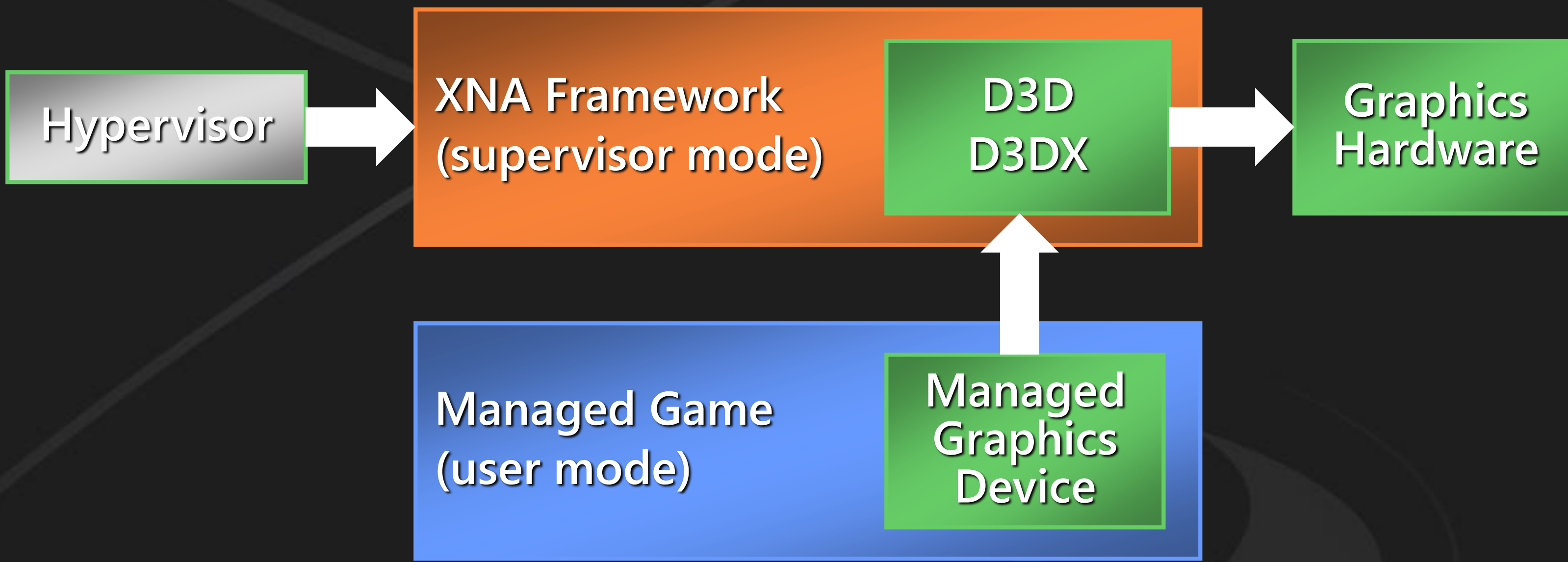
Xbox 360 hypervisor enforces security

- Hypervisor ensures only signed memory pages can execute
- Games are signed during certification

If only signed code can execute, how is a dynamically jitted runtime even possible?

XNA Framework

Managed code runs in separate user address space



- Managed code cannot directly call D3D or D3DX
- User to supervisor transitions are expensive
  - 4 microseconds per system call
- Command buffer batches up API calls

Batchable APIs

These APIs are currently batched into a single system call

Assigning to:

- VertexShader
- PixelShader
- VertexDeclaration
- IndexBuffer
- RenderState
- SamplerStates
- Textures
- DepthStencilBuffer
- Viewport
- ScissorRectangle
- ClipPlanes
- Effect.CurrentTechnique

Calling:

- Effect Begin/End
- EffectPass Begin/End
- Effect.CommitChanges
- EffectParameter.SetValue
- VertexStream.SetSource
- Set*ShaderConstant
- StateBlock Capture/Apply
- SetRenderTarget
- Draw[Indexed]Primitives
- DrawUser[Indexed]Primitives
  - *If the primitive count is small*
- Clear
- Resolve

Nasty Unbatchable APIs

These APIs currently require one system call each

- Present
- Creating or destroying graphics resources
- *.SetData, *.GetData
- DrawUser[Indexed]Primitives
  - *If the primitive count is large*
- Reading from:
  - VertexShader
  - PixelShader
  - RenderState
  - SamplerStates
  - Textures
  - Get*ShaderConstant
  - EffectParameter.GetValue

Cached Managed State

These can be read without any system call at all

- DisplayMode
- Viewport
- VertexDeclaration
- VertexStream
- IndexBuffer
- Effect.CurrentTechnique

SPRITEBATCH, EFFECTS, RENDERSTATES

Speedy Sprites

SpriteBatch is well optimized

- Draw many sprites inside one Begin/End pair
- If possible, use SpriteSortMode.Immediate
  - Draw in texture order
  - Use sprite sheets to combine multiple tiles or animation frames into a single texture
- Otherwise, use SpriteSortMode.Texture

	SpriteSortMode		
	Immediate	Deferred	Texture
1000 batches, one sprite in each *(please don't do this!)*	34 ms	34 ms	34 ms
One batch, 1000 sprites, all using the same texture	0.6 ms	0.7 ms	1.8 ms
One batch, 1000 sprites, alternating between two different textures	11.5 ms	11.6 ms	1.9 ms

Efficient Effects

Two different ways to think about effects

- *Effect = shader*
  - One effect instance per shader algorithm
  - Material parameters are stored elsewhere
- *Effect = material*
  - One effect instance per unique material, created from an original archetype effect using Effect.Clone
  - Material parameters are stored directly inside the effect
  - The Content Pipeline does this by default

	Microseconds
Begin/End on a cloned instance of BasicEffect	9.2
Using EffectParameter.SetValue and CommitChanges to update a shared BasicEffect instance	19.2
Using EffectParameterBlock.Apply to update a shared BasicEffect instance	19.6

Rapid Renderstates

- Assigning directly to managed state properties is fastest
- Using dummy effect passes to manage state can be convenient, but not faster
- State blocks are particularly slow on Xbox 360
  - Do not specify SaveStateMode.SaveState when calling SpriteBatch.Begin or Effect.Begin

	Microseconds
Assigning directly to renderstates	4.8
Using a dummy effect pass	5.4
Using a StateBlock	34.5

MATH

Math Performance

- Simple things you can do to help the JIT
  - Pass vector + matrix arguments by reference
  - Manually inline performance critical routines
- But these optimizations:
  - Can affect readability
  - May not be necessary in the future

Math Performance

A particular example

```
class Particle
{
    public Vector3 Position;
    public Vector3 Velocity;

    const float Friction = 0.9f;

    public void Update()
    {
        Position += Velocity;
        Velocity *= Friction;
    }
}
```

- Updates per second: 3380000

Math Performance

Pass structures by reference

```
public void Update()
{
    ~~Position += Velocity;~~
    ~~Velocity *= Friction;~~

    Vector3.Add(ref Position, ref Velocity, out Position);
    Vector3.Multiply(ref Velocity, Friction, out Velocity);
}
```

- Updates per second: 5540000 (x1.6)

Math Performance

Manually inline computations

```
public void Update()
{
    Position += Velocity;
    Velocity *= Friction;

    Position.X += Velocity.X;
    Position.Y += Velocity.Y;
    Position.Z += Velocity.Z;

    Velocity.X *= Friction;
    Velocity.Y *= Friction;
    Velocity.Z *= Friction;
}
```

- Updates per second: 12840000 (x3.8)

Math Performance

- XNA Framework math library is heavily inlined

```
// These alternatives perform identically
Position = Vector3.Add(Position, Velocity);
Position += Velocity;
```

- Constructors can be manually inlined

~~`Position = new Vector3(23, 42, -1);`~~

```
Position = new Vector3();
Position.X = 23;
Position.Y = 42;
Position.Z = -1;
```

MULTITHREADING

Run, Thread, Run!

- Xbox 360 has three independent CPU cores
  - CPU horsepower is idle if you have fewer than three parallel threads
- Xbox 360 does not automatically schedule threads across multiple cores
  - You must explicitly assign threads to cores
  - Current Xbox 360 ThreadPool is not optimized

Reentering the Framework

- GraphicsDevice is somewhat thread-safe
  - Cannot render from more than one thread at a time
  - Can create resources and SetData while another thread renders
- ContentManager is not thread-safe
  - Ok to have multiple instances, but only one per thread
- Input is not threadable
  - Windows games must read input on the main game thread
- Audio and networking are thread-safe

PROFILING TOOLS

Profiling on Xbox 360

XNA Framework Remote Performance Monitor for Xbox 360

- Provides basic garbage collector information
- Can tell if you have a GC problem, but not usually enough to diagnose the cause
- Shows the number of system calls
- Not much help for identifying computational bottlenecks

Profiling on Xbox 360

XNA Framework Remote Performance Monitor for Xbox 360

File Edit Options Help

Device: shawnx (Xna) Application: Vector Rumble Arguments:

Category	Name	Value	Delta	Description
GC				
	Bytes Collected By GC	38716504	1093480	The count of bytes collected by the Garbage Collector.
	Garbage Collections (GC)	36	1	The number of times the Garbage Collector has run.
	GC Compactions	1	0	The number of times the Garbage Collector has compacted the hea
	Managed Bytes In Use After GC	559896	-1296	The number of live objects after the last Garbage Collection.
	Managed Bytes Allocated	39461816	256200	The count of bytes allocated by the Garbage Collector.
	Managed Objects Allocated	2106407	14030	The count of objects allocated by the Garbage Collector.
	Bytes of String Objects Allocated	456768	1952	The count of bytes of string objects allocated by the Garbage Colle
	Managed String Objects Allocated	24998	183	The number of managed string objects allocated by the Garbage C
	Code Pitchings	0	0	The number of times the Garbage Collect has pitched JIT compiled
	Objects Finalized	3	0	The count of objects for which a finalizer have been run.
	Objects Not Moved by Compactor	50	0	The count of the objects that could not be moved by the Garbage
	Boxed Value Types	2024573	13481	The number of value types that have been boxed.
	Objects on Finalizer Queue	0	0	The number of objects on the finalizer Queue.
	GC Latency Time (ms)	119	3	The total time (in milliseconds) that the Garbage Collector has taken
	Calls to GC.Collect	0	0	The number of times the application has called the GC.Collect() me
	Objects Moved by Compactor	1723	0	The count of objects moved by the Garbage Collector during a com
	Pinned Objects	9	0	The count of pinned objects encountered while performing a Garba
Generics				
	Closed Methods Loaded	60	0	The count of unique generic methods that have been loaded acros
	Closed Methods Loaded per Definition	0	0	The maximum number of unique generic methods created for a give

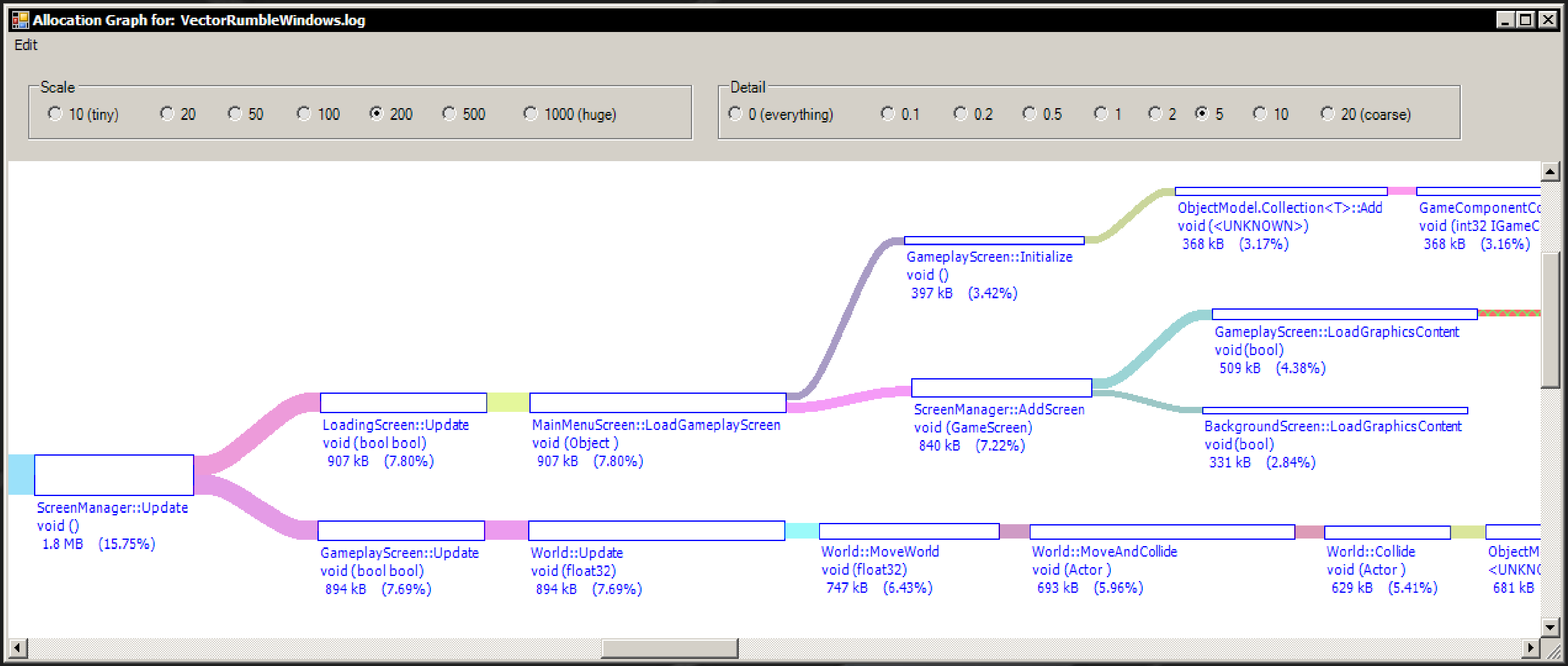
Ready

Profiling on Windows

- Inference to the rescue!
  - The XNA Framework is similar enough on both platforms that measurements taken on Windows are also applicable to the Xbox 360 version of your game
- There are many great Windows perf tools
  - The CLR Profiler for garbage collection issues
  - Sampling profilers: Visual Studio Team System, ANTS, NProf, Optimizeit, VTune

Profiling on Windows

CLR Profiler for the .NET Framework 2.0



Gamefest
MICROSOFT GAME TECHNOLOGY CONFERENCE 2007

Recommendations

- Graphics
  - Offload to the GPU
  - Understand Xbox 360 system calls
  - Choose an appropriate SpriteSortMode
  - Avoid StateBlock
- Optimize math where necessary
- Take advantage of multiple threads
- Profile on both Xbox and Windows

Gamefest 2007

MICROSOFT GAME TECHNOLOGY CONFERENCE 2007

http://www.xna.com

xna™