

Rendering and Performance

Agenda

- Memory
- Triangles
- Visibility
- Drawcalls
- Level of Detail
- Mipmapping
- Shaders
- Profiling

Why?

- Want a smooth player experience
- Want best possible visuals
- Set yourself apart from the competition
- Making a good looking game is “easy”

What?

- Frames per second (FPS)
- Goal: 60 FPS
- Requirement: 16.67ms

Memory

- Main concern is texture memory
- Geometry, animation and sound are also heavy
- No linear scale with performance
- Fixed ceiling on consoles, variable on PCs

Texture memory

- Simple to optimize by reducing resolution
- Compression: DXT

Triangles

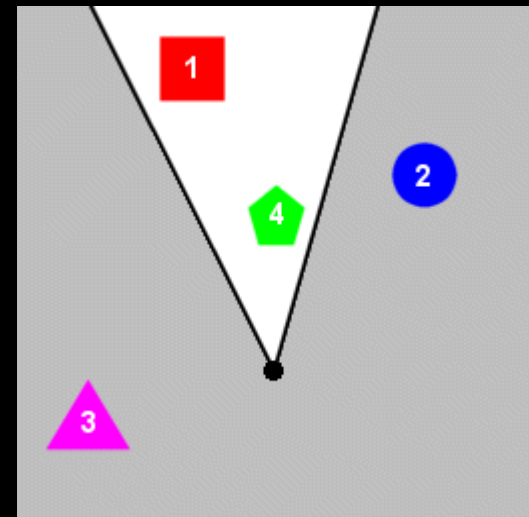
- Simplest way to work with optimization
- More triangles = lower framerate
- Not as important as it used to be
- Less of a static limit today
- Max for the Unreal 3 engine is ~3M

Drawcalls

- A drawcall is each batch processed by the GPU
- Becoming more and more of an issue
- Instances, e.g. static meshes, can help
- As few materials as possible
- Recommended max: 2000

Visibility

- View Frustum Culling
- Backface culling
- Object culling
 - BSP
 - 50/50 cuts
 - PVS
 - VIS
 - Rooms and gates
 - Z-buffer checks



Level of Detail

- A technique where a model switches to a lower detail version at a set distance
- Allows for less detail further away
- Highly effective as it can give a massive decrease in tri count
- Small memory hit

Mipmapping

- Latin for *multum in parvo* meaning “much in small space”
- Low-res copies of the same texture



Mipmapping

- Decreased texture aliasing
- Decreased required amount of texels(texture pixels) rendered
- Tex size += $1/4 + 1/8 + 1/16$
- 1/3 increased memory consumption per texture
- <http://number-one.com/product/Mipmapping,%20Part%201/index.html>

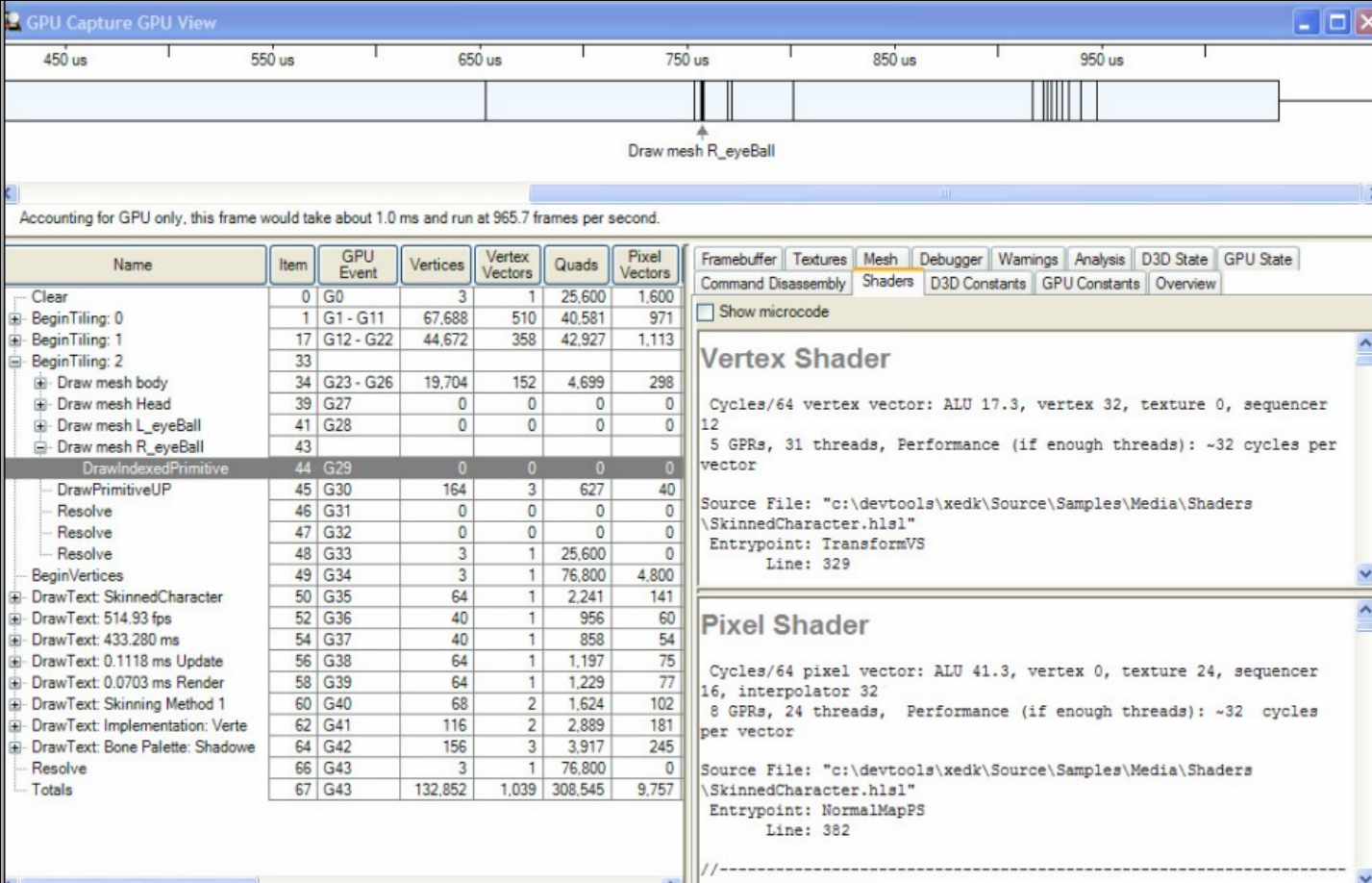
Shaders

- Post Processing Shaders
 - SSAO
 - DOF
 - Blur
- Per face shaders
 - Specular
 - Normal
 - Reflection
 - Parallax

Profiling

- Knowledge is power
- Helps determining bottlenecks
- Performance Investigator for directX (PIX)
- Nvidia PerfHUD

Profiling



GPU Capture GPU View

450 us | 550 us | 650 us | 750 us | 850 us | 950 us

Draw mesh R_eyeBall

Accounting for GPU only, this frame would take about 1.0 ms and run at 965.7 frames per second.

Name	Item	GPU Event	Vertices	Vertex Vectors	Quads	Pixel Vectors
Clear	0	G0	3	1	25,600	1,600
BeginTiling: 0	1	G1 - G11	67,688	510	40,581	971
BeginTiling: 1	17	G12 - G22	44,672	358	42,927	1,113
BeginTiling: 2	33					
Draw mesh body	34	G23 - G26	19,704	152	4,699	298
Draw mesh Head	39	G27	0	0	0	0
Draw mesh L_eyeBall	41	G28	0	0	0	0
Draw mesh R_eyeBall	43					
DrawIndexedPrimitive	44	G29	0	0	0	0
DrawPrimitiveUP	45	G30	164	3	627	40
Resolve	46	G31	0	0	0	0
Resolve	47	G32	0	0	0	0
Resolve	48	G33	3	1	25,600	0
BeginVertices	49	G34	3	1	76,800	4,800
DrawText: SkinnedCharacter	50	G35	64	1	2,241	141
DrawText: 514.93 fps	52	G36	40	1	956	60
DrawText: 433.280 ms	54	G37	40	1	858	54
DrawText: 0.1118 ms Update	56	G38	64	1	1,197	75
DrawText: 0.0703 ms Render	58	G39	64	1	1,229	77
DrawText: Skinning Method 1	60	G40	68	2	1,624	102
DrawText: Implementation: Verte	62	G41	116	2	2,889	181
DrawText: Bone Palette: Shadowe	64	G42	156	3	3,917	245
Resolve	66	G43	3	1	76,800	0
Totals	67	G43	132,852	1,039	308,545	9,757

Framebuffer | Textures | Mesh | Debugger | Warnings | Analysis | D3D State | GPU State

Command Disassembly | Shaders | D3D Constants | GPU Constants | Overview

Show microcode

Vertex Shader

Cycles/64 vertex vector: ALU 17.3, vertex 32, texture 0, sequencer 12
5 GPRs, 31 threads, Performance (if enough threads): ~32 cycles per vector

Source File: "c:\devtools\xedk\Source\Samples\Media\Shaders\SkinnedCharacter.hlsl"
Entrypoint: TransformVS
Line: 329

Pixel Shader

Cycles/64 pixel vector: ALU 41.3, vertex 0, texture 24, sequencer 16, interpolator 32
8 GPRs, 24 threads, Performance (if enough threads): ~32 cycles per vector

Source File: "c:\devtools\xedk\Source\Samples\Media\Shaders\SkinnedCharacter.hlsl"
Entrypoint: NormalMapPS
Line: 382

//-----

Unreal pitfalls

- Too big lightmaps
- Too much BSP
- Too many objects

Q u e s t i o n s ?