Command Buffers and Pipelines

Michael Worcester – Driver Engineer
(michael.worcester@imgtec.com)
26 May 2016
Command Buffers – Deferring the work

- OpenGL is immediate (ignoring display lists)
  - Driver does not know how much work is incoming
  - Has to guess
  - Bad!

- Vulkan splits recording of work from submission of work
  - Removes guesswork from driver
  - Reducing hitching
  - Helps eliminate unexplained resource usage
Command Buffers – Pooling Resource

- Command Buffers always belong to a Command Pool
  - Buffers are allocated from pools
  - Pools provide lightweight synchronisation
  - Pools can be reset, reclaiming all resources
  - Two flavours of pool:
    - Individual reset of command buffers
    - Group reset only
Command Buffers – Going wide

Single Thread

OpenGL Context

Thread 1

VkCommandBuffer

Thread 2

VkCommandBuffer

... 

Thread N

VkCommandBuffer
Command Buffers – Command Types

- Deferred recording of commands
  - Transfer
  - Graphics
  - Compute
  - Synchronisation
Command Buffers – Transfers

- Transfer commands are raw copies
  - However, they can change the *tiling* of an image (this is the only way!)
- CPU -> GPU
  - Texture upload
  - Static buffer data
- GPU -> CPU
  - Read back of data
- GPU -> GPU
  - Pipelined updates of data
  - Mipgen
Command Buffers – “Inside” or “Out”
Command Buffers – Secondaries

Primary
- Transfer
- Compute
- RenderPass
  - ExecuteCommands
  - ExecuteCommands
- Compute

Secondaries
- BindPipeline
- BindDescriptors
- Draw

- BindPipeline
- BindDescriptors
- Draw
- Draw
Command Buffers – Reuse
Command Buffers – Reuse
Command Buffers – Lifetime

- Allocated
  - Begin
- Record
  - End
  - Begin
  - Pending
  - Submit
- CPU
  - Wait
  - Active
- GPU
  - Ownership

© Imagination Technologies
Pipelines - An anatomy

- Fixed Function States
- Programmable Shaders
- Descriptor Layout
- Renderpass (more later)
- Dynamic State
Pipelines – Fixed Function States

- Everything that isn’t a shader
- Buffer formats/layouts

- VertexInput
- InputAssembly
- Tessellation
- Viewport
- Raster
- Multisample
- DepthStencil
- ColorBlend
Pipelines – Shader Stages

- Currently same as OpenGL
  - Vertex
  - Control
  - Evaluation
  - Geometry
  - Fragment

- Note: Tessellation and Geometry are optional features
Pipelines – Descriptor Layout

Describes the set of resources that a shader can access

- Uniforms
- Storage Buffers
- Images
- Samplers
- Push Constants
Pipelines – Dynamic State

- Per-draw state
- Tedious to compile each one
  - Combinatorial explosion
- Dynamic state!
  - Opt-in
  - Only use when required

- Viewport
- Scissor
- Line Width
- Depth Bias
- Blend Constant Colour
- Depth Bounds
- Stencil
  - Compare
  - Write
  - Reference
Pipelines – The Cache

- Share common state
- Load/Store