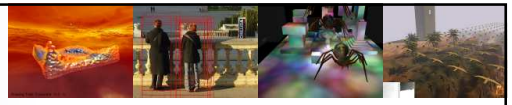


Mobile Graphics

Patrick Cozzi
University of Pennsylvania
CIS 565 - Spring 2012

Announcements

- Homework 5
  - Due today
  - In-class quiz this Wednesday
- Monday, 04/23
  - No class
- Wednesday, 04/25
  - Project presentations, 9am-12pm, 307 Towne
  - Post code, paper, and video on blog beforehand
  - One-on-one demos to follow presentations

Agenda

- Tile-Based Rendering
  - Motivation
  - Implementation
  - Implications on optimizing our code

Memory Bandwidth

- For 32-bit color, 4 bytes per pixel are needed for display
- How many bytes were transferred?

Memory Bandwidth

- For 32-bit color, 4 bytes per pixel are needed for display
- How many bytes were transferred?
  - 12 bytes?
    - Read/write depth and stencil
    - Write color

Memory Bandwidth

- For 32-bit color, 4 bytes per pixel are needed for display
- How many bytes were transferred?
  - 12 bytes?
    - Read/write depth and stencil
    - Write color
  - What about:
    - Textures
    - Overdraw
    - Blending
    - Multisampling

Immediate Mode Rendering

- *I*mmediate *M*ode *R*endering (*IMR*) specifies triangles to be drawn in a current state

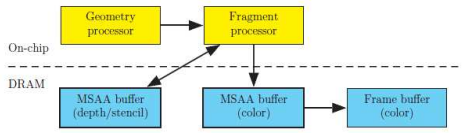


Image from http://www.openglinsights.com/

Immediate Mode Rendering

- IMRs can result in overdraw

Image from http://www.openglinsights.com/

Immediate Mode Rendering

- Sort front to back to minimize overdraw

Image from http://www.openglinsights.com/

Immediate Mode Rendering

- Depth pre-pass to minimize overdraw

Image from http://www.openglinsights.com/

Tile-Based Rendering

- Bandwidth uses a significant amount of power consumption
- *Mobile graphics* want to maximum battery life. How?
  - Minimize accessing the framebuffer in global memory

Tile-Based Rendering

- Break framebuffer up into *tiles*, e.g., 16x16 pixels

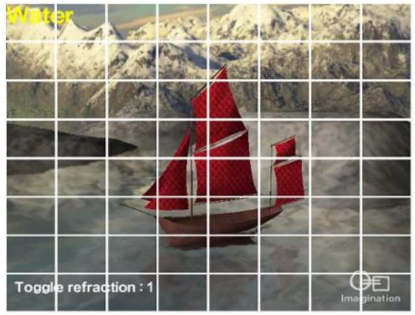
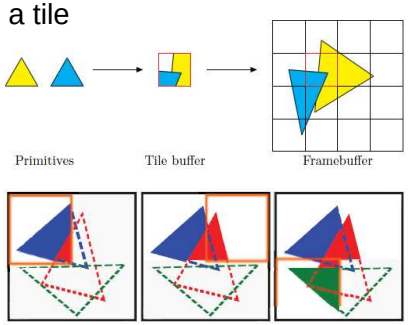


Image from http://www.imgtec.com/powervr/insider/docs/PowerVR%20Series5%20Graphics.SGX%20architecture%20guide%20for%20developers.1.0.8.External.pdf

Tile-Based Rendering

- Render one tile at a time using all primitives that overlap a tile



Images from http://www.openglinsights.com/

Tile-Based Rendering

- A tile is stored on-chip in the *tile buffer*
- All depth/stencil/color access is on-chip
- After the tile is rendered, color is written to global memory
- How does this affect vertex processing?

Tile-Based Rendering

- Avoid transforming all vertices in the scene per-tile by storing
  - `gl_Position` and vertex shader varyings
  - Fragment shader and uniforms
  - Fixed function state, e.g., depth test, etc.

  in a spatial data structure call the *frame data*

Tile-Based Rendering

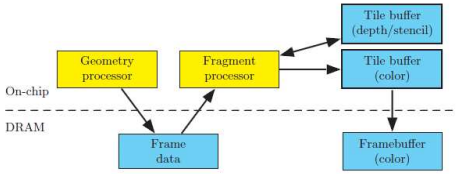


Image from http://www.openglinsights.com/

What's the Difference?

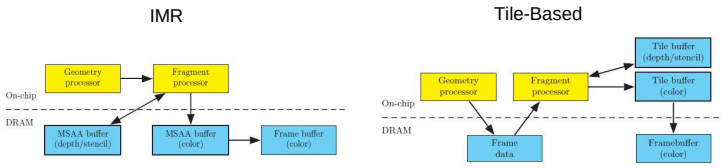


Image from http://www.openglinsights.com/

Framebuffer Clears

- `glClear` is cheap and clears frame data
  - Don't skip it
  - Clear everything, e.g., all buffers, no scissor test, etc.
- Even better, use `EXT_discard_framebuffer`:

```
const GLenum attachments[3] = { COLOR_EXT, DEPTH_EXT, STENCIL_EXT };
glDiscardFramebufferEXT(GL_FRAMEBUFFER, 3, attachments);
```

Tile-Based Deferred Rendering

- On PowerVR, only fragments that contribute to the scene are shaded:

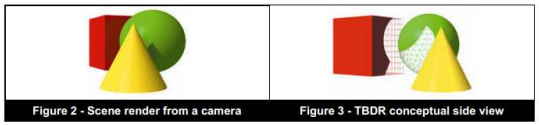


Figure 2 - Scene render from a camera | Figure 3 - TBDR conceptual side view

- Called *T*ile-*B*ased *D*eferred *R*endering (*TBDR*)

Image from http://www.imgtec.com/powervr/insider/docs/PowerVR%20Series5%20Graphics.SGX%20architecture%20guide%20for%20developers.1.0.8.External.pdf

Tile-Based Deferred Rendering

- How does sorting front-to-back and depth prepass affect TBDR?

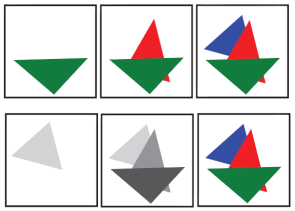


Image from http://www.openglinsights.com/