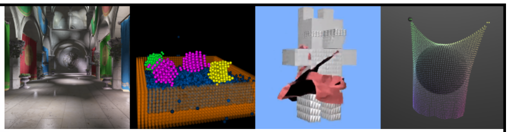


Deferred Shading

Patrick Cozzi
University of Pennsylvania
CIS 565 - Fall 2016

Renderer Design

- Design an engine that renders lots of
  - □ Objects with different materials
  - □ Dynamic lights
  - □ Different light types
- Cleanly. Efficiently.

Forward Rendering – Multi-Pass

```
foreach light
{
  foreach visible object
  {
    Render using shader for
    this material/light;

    accumulate in framebuffer;
  }
}
```

- Pros and cons?

Forward Rendering – Multi-Pass

- One shader per material/light-type
- Performance
  - □ Need to do vertex transform, rasterization, material part of fragment shader, etc. multiple times for each object.
  - □ Occluded fragments are shaded
  - □ Not all lights affect the entire object
-

Forward Rendering – Single Pass

```
foreach visible object
{
  find lights affecting object;

  Render all lights and materials using
  a single shader;
}
```

- Pros and cons?

Forward Rendering – Single Pass

- Lots of shaders
  - □ One shader per material/light-combination
  - □ Hard to author shaders
  - □ May require runtime compile/link
  - □ Long ubershader increase compile times
  - □ More potential shaders to sort by
- Same as multi-pass
  - □ Occluded fragments are shaded
  - □ Not all lights affect the entire object

Deferred Rendering

```
foreach visible object
{
  write properties to g-buffer;
}

foreach light
{
  compute light using g-buffer;
  accumulate in framebuffer;
}
```

- Pros and cons?

Deferred Rendering

- Decouple lighting from scene complexity
- Few shaders
  - □ One per material
  - □ One per light type
- Only transform and rasterize each object once
- Only light non-occluded objects

Deferred Rendering

- Memory bandwidth usage - read g-buffer for each light
- Recalculate full lighting equation for each light
- Limited material properties in g-buffer
- MSAA and translucency are difficult

G-Buffer Layout in Leadwerks 2.1

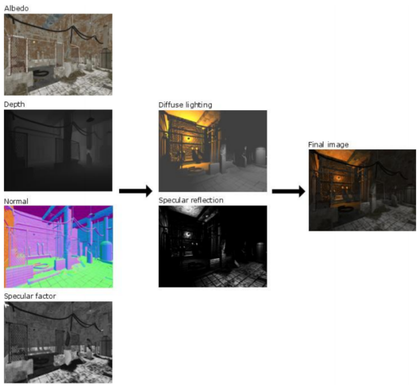
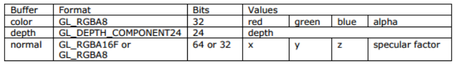


Image from http://www.leadwerks.com/files/Deferred_Rendering_in_Leadwerks_Engine.pdf

G-Buffer Layout in Leadwerks 2.1

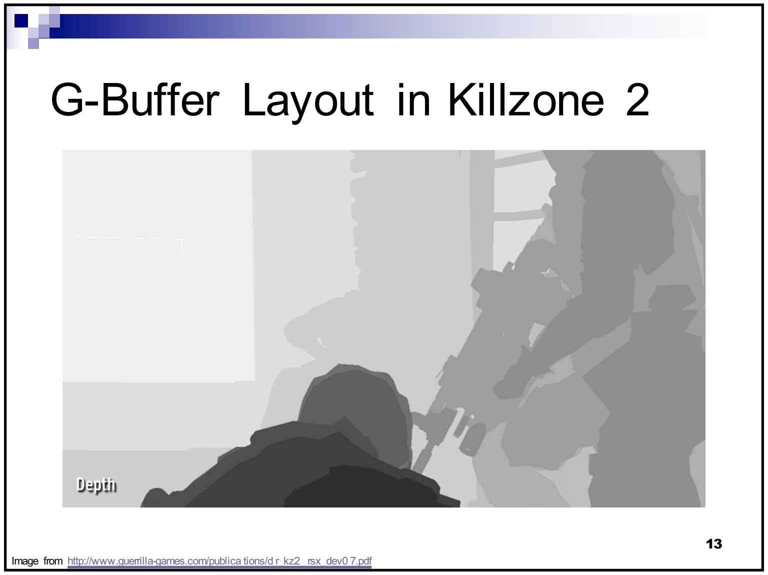


Buffer	Format	Bits	Values			
color	GL_RGBA8	32	red	green	blue	alpha
depth	GL_DEPTH_COMPONENT24	24	depth			
normal	GL_RGBA16F or GL_RGBA8	64 or 32	x	y	z	specular factor

Image from http://www.leadwerks.com/files/Deferred_Re ndering_in_Leadwe rks_Engine. pdf

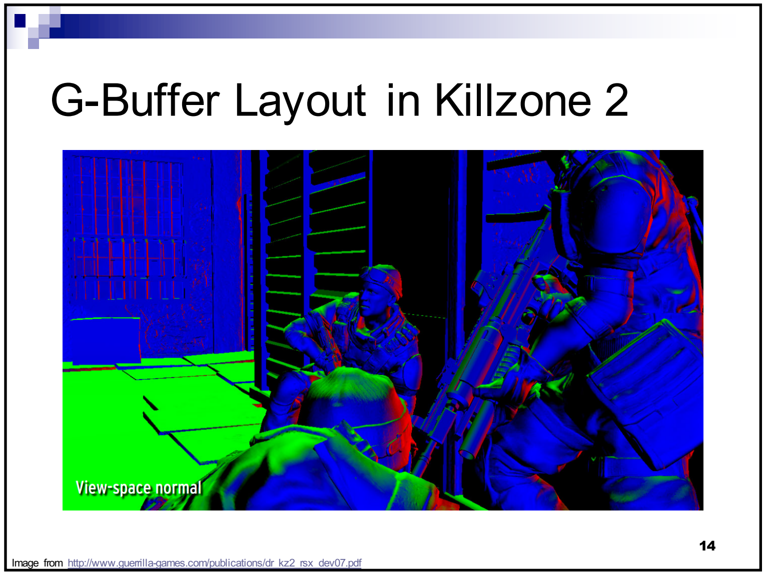
G-Buffer Layout in Killzone 2

Image from http://www.guerrilla-games.com/publications/dr_kz2_rsx_dev07.pdf



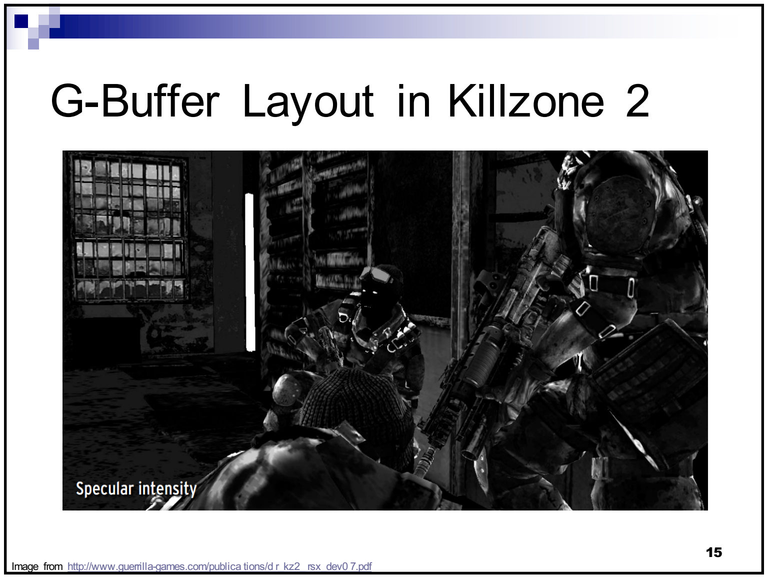
G-Buffer Layout in Killzone 2

Image from http://www.guerrilla-games.com/publications/dr_kz2_rsx_dev07.pdf



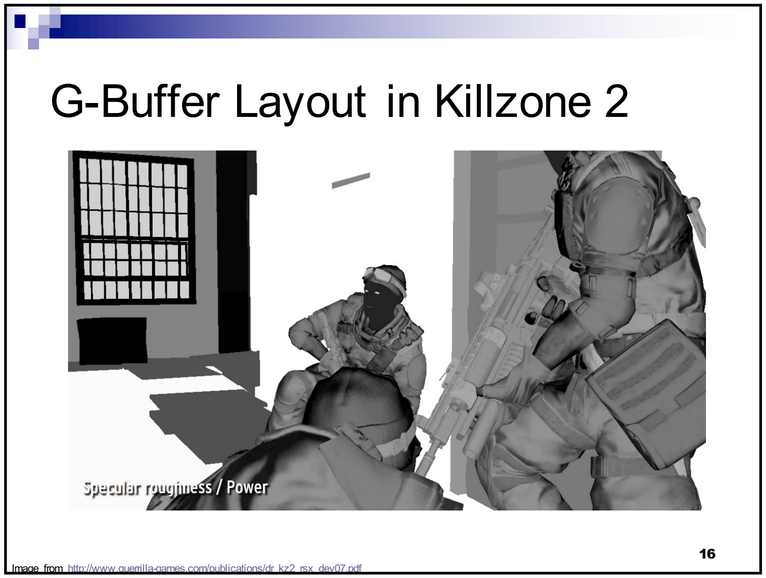
G-Buffer Layout in Killzone 2

Image from http://www.guerrilla-games.com/publications/dr_kz2_rsx_dev07.pdf



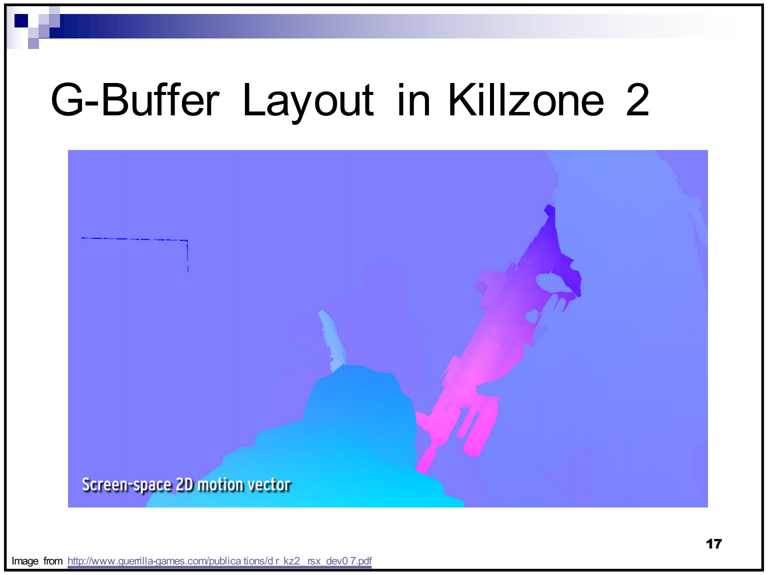
G-Buffer Layout in Killzone 2

Image from http://www.guerrilla-games.com/publications/dr_kz2_rsx_dev07.pdf



G-Buffer Layout in Killzone 2

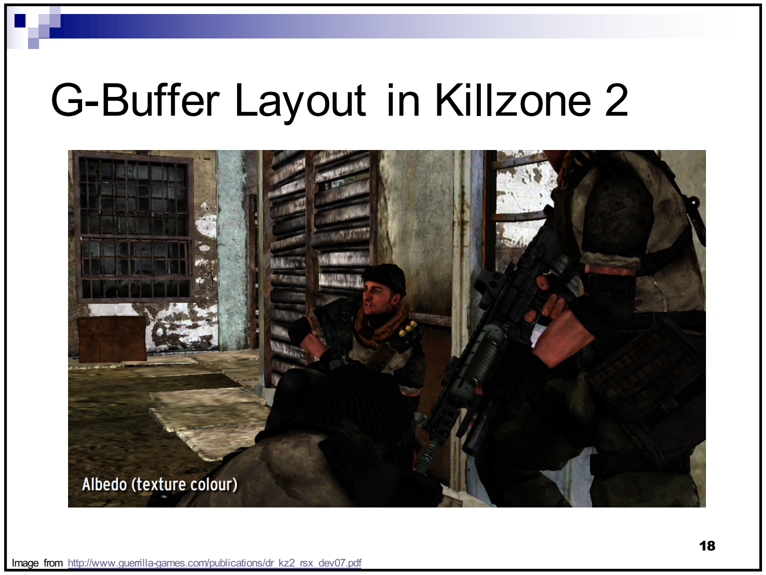
Image from http://www.guerrilla-games.com/publications/dr_kz2_rsx_dev07.pdf



G-Buffer Layout in Killzone 2

Screen-space 2D motion vector

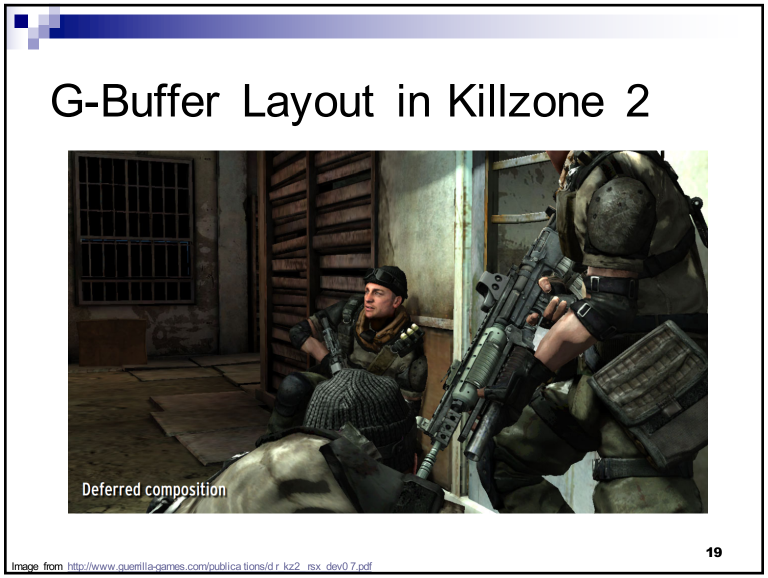
Image from http://www.guerrilla-games.com/publications/dr_kz2_rsx_dev07.pdf



G-Buffer Layout in Killzone 2

Albedo (texture colour)

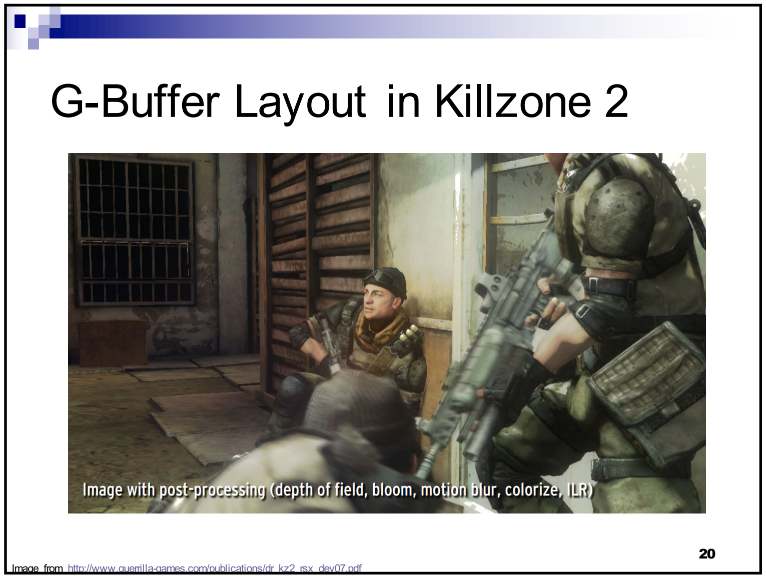
Image from http://www.guerrilla-games.com/publications/dr_kz2_rsx_dev07.pdf



G-Buffer Layout in Killzone 2

Deferred composition

Image from http://www.guerrilla-games.com/publications/dr_kz2_rsx_dev07.pdf

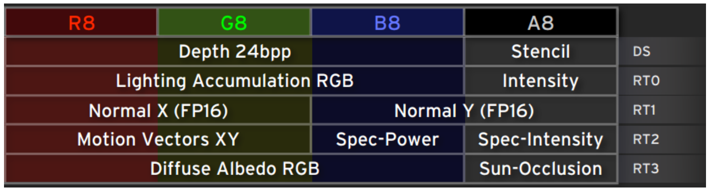


G-Buffer Layout in Killzone 2

Image with post-processing (depth of field, bloom, motion blur, colorize, ILR)

Image from http://www.guerrilla-games.com/publications/dr_kz2_rsx_dev07.pdf

G-Buffer Layout in Killzone 2



R8	G8	B8	A8	
Depth 24bpp			Stencil	DS
Lighting Accumulation RGB			Intensity	RT0
Normal X (FP16)		Normal Y (FP16)		RT1
Motion Vectors XY		Spec-Power	Spec-Intensity	RT2
Diffuse Albedo RGB			Sun-Occlusion	RT3

Image from http://www.guerrilla-games.com/publications/dr_kz2_rsx_dev07.pdf

Light Accumulation Pass

- Geometry for each light
  - □ Full-screen quad/triangle
    - with scissor/stencil test
  - □ 3D bounding geometry. Examples:
    - Point light – sphere
    - Spot light – cone

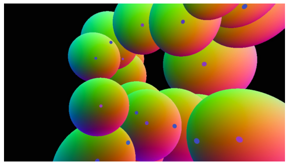
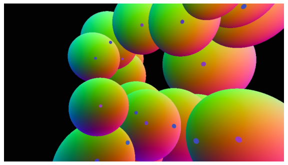
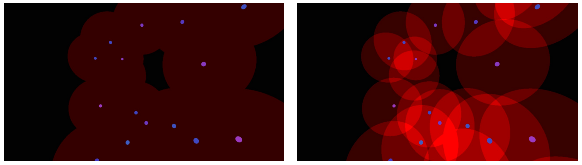


Image from http://marcinignac.com/blog/deferred-rendering-explained/

Optimizing Light Accumulation





Overlapping spheres: use additive blending

Images from http://marcinignac.com/blog/deferred-rendering-explained/

Optimizing Light Accumulation

- Render backfaces only (use frontface culling)
- Set depth test to `GREATER`
- Now pixels need to belong to an object and be inside a sphere

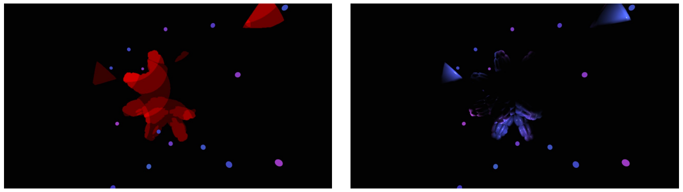


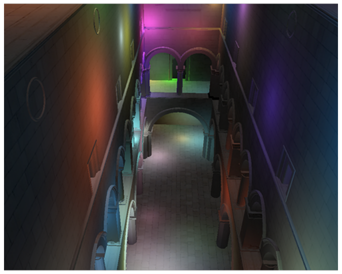
Image from http://marcinignac.com/blog/deferred-rendering-explained/

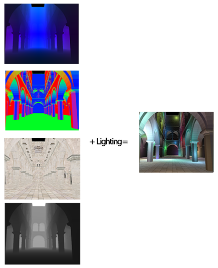
Optimizing Light Accumulation

- Demo: http://marcinignac.com/blog/deferred-rendering-explained/demo/

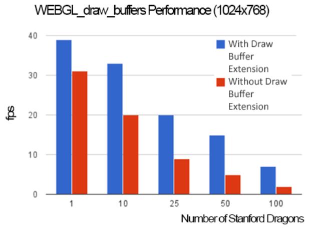
WebGL Demo

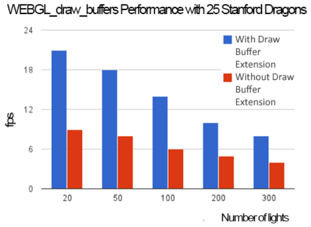
- https://hacks.mozilla.org/2014/01/webgl-deferred-shading/
- By Yuqin Shao and Sijie Tian, CIS 565 alumni





WEBGL_draw_buffers performance impact


WEBGL_draw_buffers Performance (1024x768)
fps
40
30
20
10
0
1
10
25
50
100
Number of Stanford Dragons
With Draw Buffer Extension
Without Draw Buffer Extension

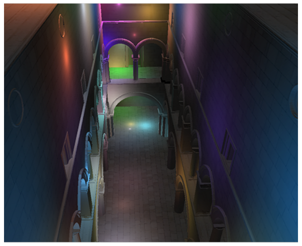


Helps most when g-buffer pass is expensive (high scene complexity)

Helps less when light accumulation pass is expensive

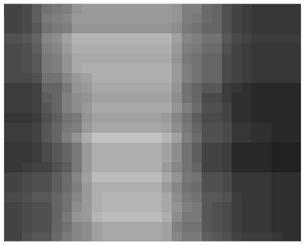
Tile-Based Deferred Shading

- Divide screen into 2D tiles, e.g., 16x16 pixels
- Determine which lights influence which tiles ➪ light-tile info
  - CPU or compute shader!
- Light accumulation pass
  - Read g-buffer once
    - Save bandwidth compared to once per light
  - Use light-tile info to find which lights affect a pixel

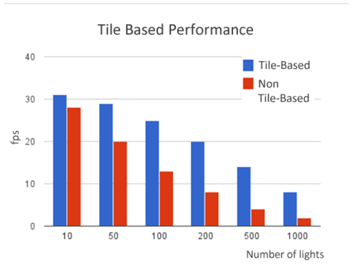


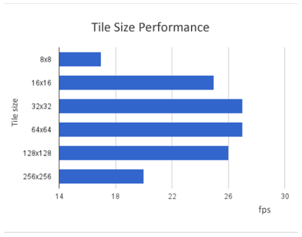
Tile-Based Deferred Shading

- Divide screen into 2D tiles, e.g., 16x16 pixels
- Determine which lights influence which tiles ⇨ light-tile info
  - □ CPU or compute shader!
- Light accumulation pass
  - □ Read g-buffer once
    - Save bandwidth compared to once per light
  - □ Use light-tile info to find which lights affect a pixel



Tile-Based Deferred Shading





References

- Deferred Rendering in Killzone 2 – Michal Valient
  - □ http://www.guerrilla-games.com/publications/dr_kz2_rsx_dev07.pdf
- Deferred Rendering in Leadwerks Engine - Josh Klint
  - □ http://www.leadwerks.com/files/Deferred_Rendering_in_Leadwerks_Engine.pdf
- Light Pre-Pass – Wolfgang Engel
  - □ http://www.slideshare.net/cagetu/light-prepass
- Compact Normal Storage for Small G-Buffers – Aras Pranckevičius
  - □ http://aras-p.info/texts/CompactNormalStorage.html
- WebGL Deferred Shading
  - □ https://hacks.mozilla.org/2014/01/webgl-deferred-shading/
- Deferred Rendering Explained
  - □ http://marcinignac.com/blog/deferred-rendering-explained/
- WebGL Deferred Shading (Floored)
  - □ http://www.floored.com/blog/2015webgl-deferred-shading-gbuffer-floating-point-texture/