VGP352 – Week 10

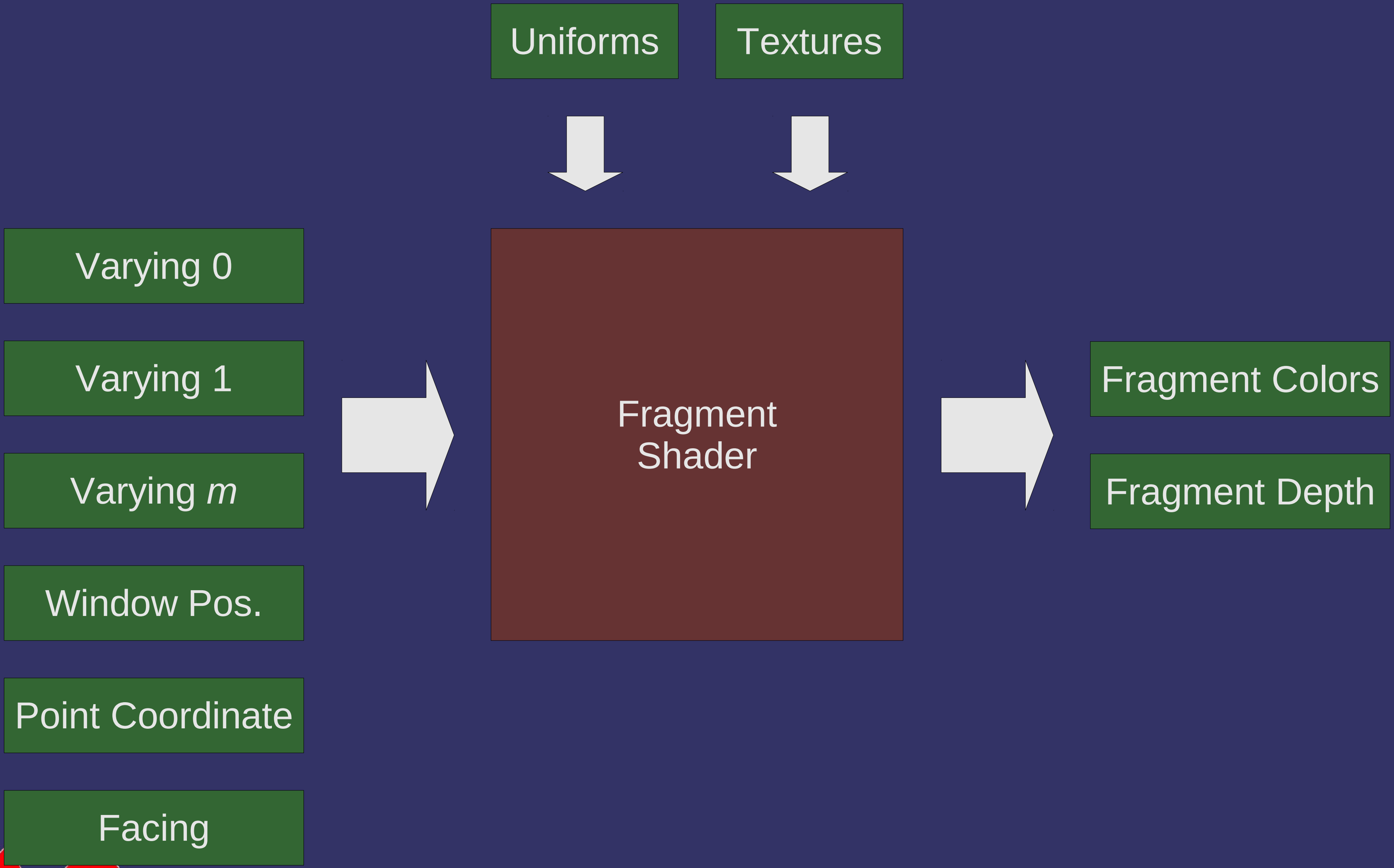
- Agenda:
  - Quiz #3
  - Multiple render targets
  - Deferred shading
  - Discuss the final

16-March-2010

MRT

- Multiple color outputs from the fragment shader
  - For practical purposes, requires the use of framebuffer objects
  - Slightly changes GLSL syntax

MRT


Uniforms
Textures
Varying 0
Varying 1
Varying *m*
Window Pos.
Point Coordinate
Facing
Fragment
Shader
Fragment Colors
Fragment Depth

Framebuffer Objects

- Attach one or more renderable objects to it
  - 1D, 2D, and 3D versions exist

```
void glFramebufferTexture2DEXT (GLenum target,
    GLenum attachment, GLenum textarget,
    GLuint texture, GLint level);

void glFramebufferRenderbufferEXT(
    GLenum target, GLenum attachment,
    GLenum renderbuffertarget,
    GLuint renderbuffer);
```

Selects how the buffer is used:

- Color buffer: `GL_COLOR_ATTACHMENT0`
- Depth buffer: `GL_DEPTH_ATTACHMENT`
- Stencil buffer: `GL_STENCIL_ATTACHMENT`

MRT – FBO Usage

- Use additional color attachments
  - e.g. `GL_COLOR_ATTACHMENT1`
  - Maximum number of attachments queryable with `GL_MAX_COLOR_ATTACHMENTS`
  - EXT_fbo requires that all color attachments have the same internal format
    - ARB_fbo / OpenGL 3.0 allow drivers to relax this restriction
    - The driver can still reject a particular combination
    - Most hardware can handle combinations with the same size internal formats
      - e.g. `GL_RGBA8` with `GL_RGBA_10_10_10_2`

MRT – Setting Draw Buffers

- Connect attachments with shader outputs:

```
void glDrawBuffers(GLsizei n,
    const GLenum *bufs);
```

  - `bufs` gives a list of attachments points to connect, in the specified order, with shader outputs
    - Shader output 0 gets the first listed attachment, output 1 gets the second, etc.
  - Maximum number of outputs queryable with `GL_MAX_DRAW_BUFFERS`

MRT – GLSL Usage

- `gl_FragColor` is but one output. What to do?
  - Replace with a new output that is declared as an array:

```
vec4 gl_FragData[];
```

  - Each element in `gl_FragData` corresponds to one of the outputs set by `glDrawBuffers`

References

Jones, Rob. “OpenGL Frame Buffer Object 201.” GameDev.net. December 14th, 2006. Accessed on June 10th, 2009. http://www.gamedev.net/reference/articles/article2333.asp

Deferred Shading

- Scenes with high depth complexity or many lights suffer from several problems:
  - Many passes to implement the lights
  - Lots of wasted fragment processing
  - Difficulty with per-batch storage for shadow maps
  - Difficulty with stencil shadows from multiple lights
  - etc.
  - End result: poor performance

Deferred Shading

- What if we could easily:
  - Light each pixel (not fragment) exactly once
  - Only apply lights to the fragments they affect
  - Reduce per-light cost in scenes with many lights

Deferred Shading

- General idea:
  - Render scene information needed for shading to an off-screen geometry buffer (G-buffer)
  - Draw per-light geometry to screen sampling from G-buffer to calculate shading

Deferred Shading – G-Buffer

- All per-fragment data required for shading:
  - Normal
  - Position
  - Diffuse / specular color
  - etc.
- Emit this during per-object rendering
  - Output this data instead of performing lighting calculations
  - Use MRT!

Deferred Shading – G-Buffer

- Example G-buffer layout:
  - 2 RGBA16F outputs:

Diffuse (red)	Diffuse (green)	Diffuse (blue)	m
Normal (X)	Normal (Y)	Normal (Z)	n

  - m is the Cook-Torrance roughness
  - n is the index of refraction

Deferred Shading – G-Buffer

- Tough choices:
  - Explicitly store position or derive from screen X/Y and depth value?
  - Explicitly store the normals Z or derive from its X and Y?
  - One of the most important parts of designing a deferred shading engine is selecting the parameters and the packing

Deferred Shading – G-Buffer

- CryEngine 3 stores normals in 2 components
  - Encode:

```
normal_g = normalize(normal.xy) *
    sqrt((normal.z / 2.0) + 0.5);
```

  - Decode:

```
normal.z  = (length(normal_g.xy) * 2.0) — 1.0;
normal.xy = normalize(normal_g.xy) *
    sqrt(1.0 – (normal.z * normal.z);
```

  - Very similar to the mapping for spherical reflection maps
  - More expensive to compute, but has better precision

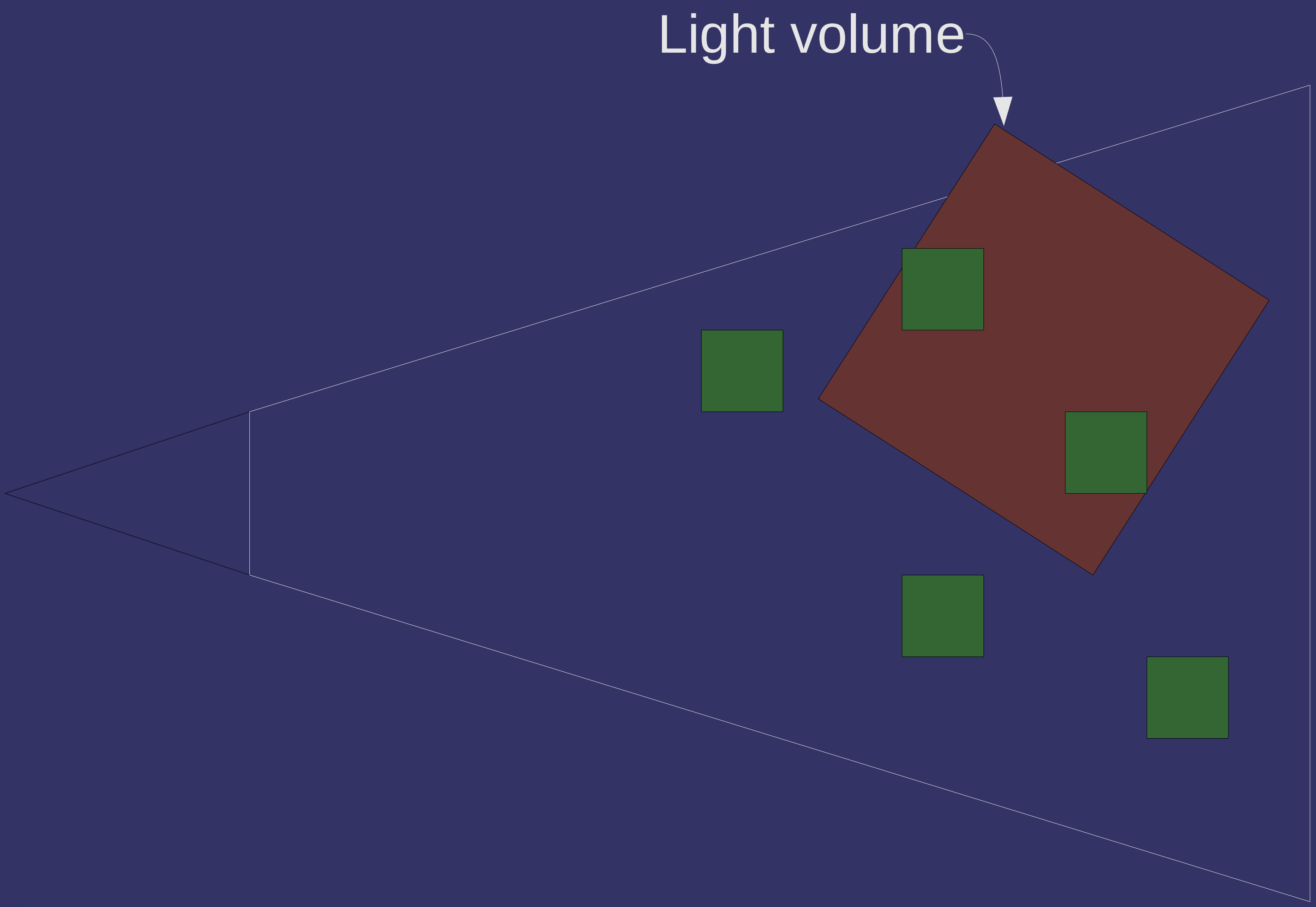
Deferred Shading – Lighting

- For each light, draw simplified bounding geometry
  - Perform lighting for each fragment drawn
    - Only light the areas of the scene that need lighting
    - Read from G-buffer at the screen X/Y position
    - Add calculated lighting to existing values
  - Examples:
    - Directional light: box
    - Point light: sphere
    - Spot light: cone

Deferred Shading – Lighting

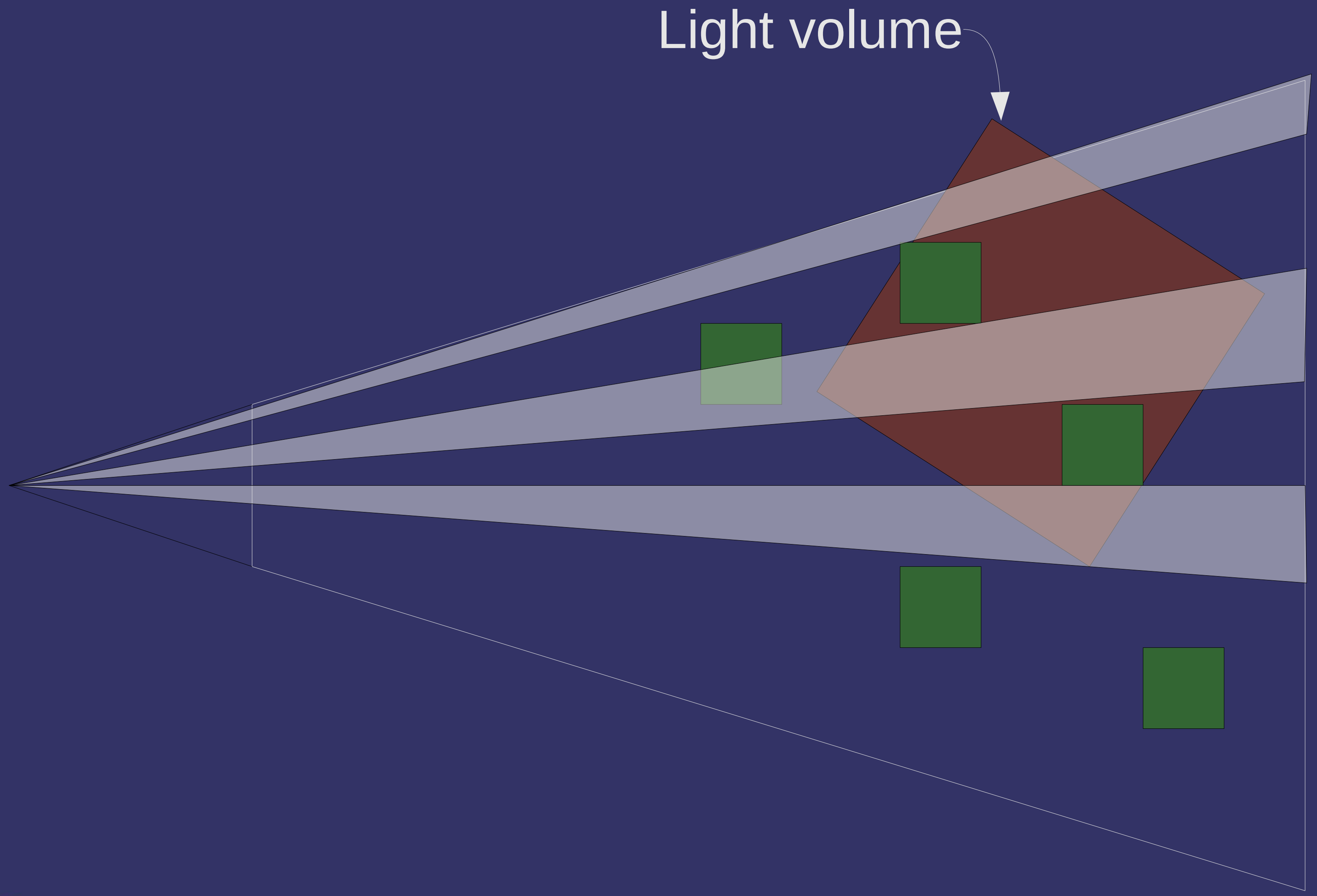
- Optimize by letting the early stencil test discard many fragments
  - Draw the light volume once:
    - Disable color writes
    - Set depth function to `GL_LESS` and stencil function to `GL_ALWAYS`
    - Set Z-fail stencil operation to `GL_REPLACE` and all others to `GL_KEEP`
  - Draw the light volume again:
    - Enable color writes
    - Set depth function to `GL_LEQUAL` and stencil function to `GL_EQUAL`
    - Set all stencil operations `GL_KEEP`

Deferred Shading – Lighting



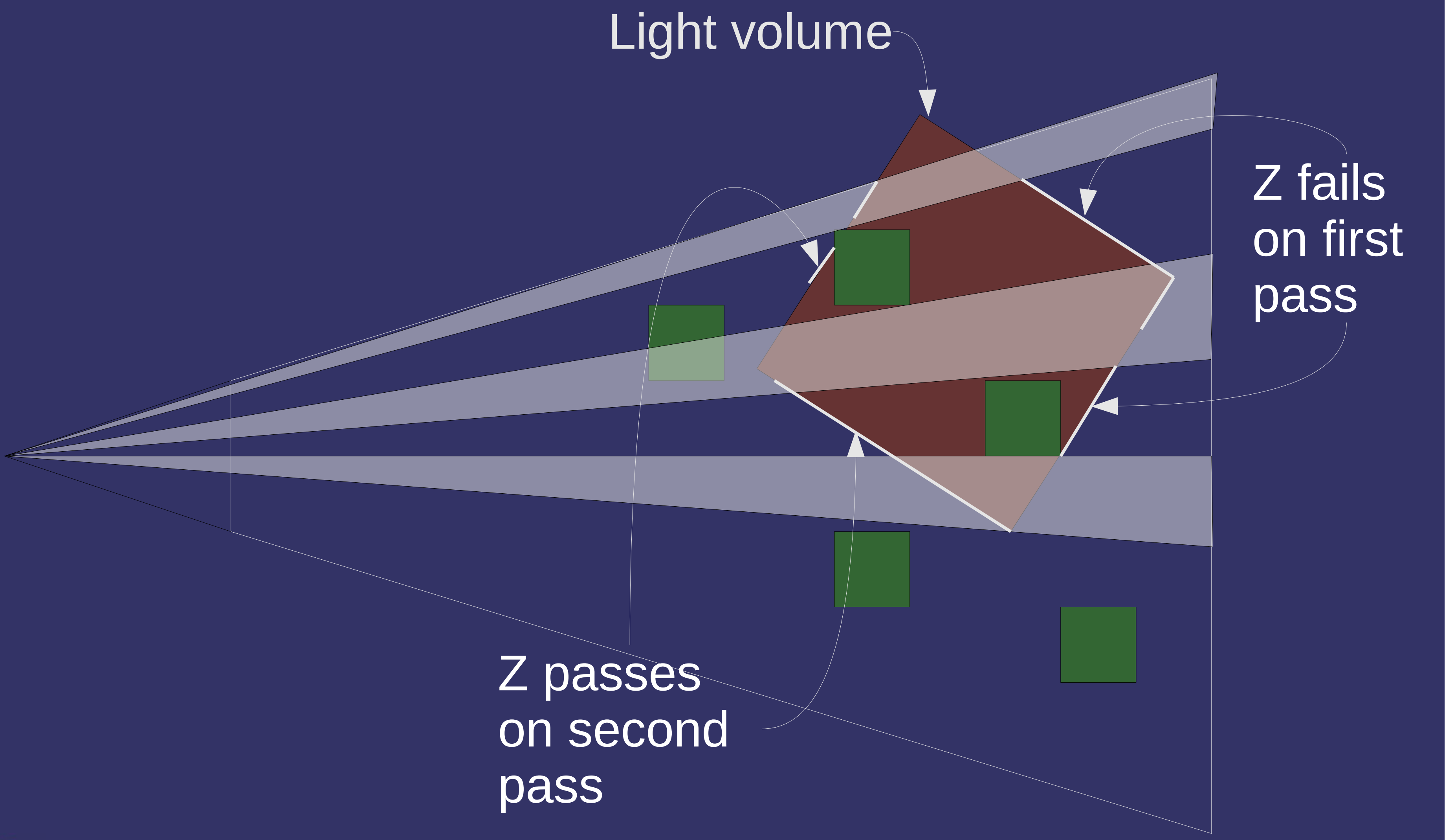
16-March-2010

Deferred Shading – Lighting



16-March-2010

Deferred Shading – Lighting



Light volume

Z fails on first pass

Z passes on second pass

16-March-2010

Deferred Shading – Drawbacks

- What could go wrong?

Deferred Shading – Drawbacks

- What could go wrong?
  - Transparency effects won't work
  - Traditional anti-aliasing (multisampling) has problems

References

Hargreaves, S., Harris, M. “Deferred Shading.” Nvidia 6800 Leagues Under the Sea. June 2004. http://developer.nvidia.com/object/6800_leagues_deferred_shading.html

Fabio Policarpo, Francisco Fonseca, *Deferred shading tutorial*. Pontifical Catholic University of Rio de Janeiro. 2005. http://www710.univ-lyon1.fr/~jciehl/Public/educ/GAMA/2007/Deferred_Shading_Tutorial_SBGAMES2005.pdf

Shishkovtsov, Oles. "Deferred Shading in S.T.A.L.K.E.R." in Fernando, Randima (editor) GPU Gems 2, Addison Wesley, 2005. http://http.developer.nvidia.com/GPUGems2/gpugems2_chapter09.html

Mittring, M. “A bit more deferred – CryEngine3.” Triangle Game Conference 2009. http://www.crytek.com/technology/presentations/

Global Illumination

- Can deferred shading be used to implement global illumination?
  - Yes, but...
    - Only for a single “bounce”
    - Only for diffuse inter-reflections
- Deferred shading makes using many lights very cheap
  - Where many can mean 100's
  - Generate a bunch of fake lights that represent the reflection of light from surfaces
    - Call these *virtual point lights* (VPLs)

Virtual Point Lights

- Generate VPLs:
  - Trace paths from each light to first intersection
    - This determines the position of the VPL
    - Treat all VPLs as 180° spot lights
  - Calculate reflection at intersection
    - This determines the intensity of the VPL

References

Samuli Laine, Hannu Saransaari, Janne Kontkanen, Jaakko Lehtinen, and Timo Aila. "Incremental Instant Radiosity for Real-Time Indirect Illumination." Eurographics Symposium on Rendering 2007. http://www.tml.tkk.fi/~timo/

Next week...

- The final

Legal Statement

This work represents the view of the authors and does not necessarily represent the view of Intel or the Art Institute of Portland.

OpenGL is a trademark of Silicon Graphics, Inc. in the United States, other countries, or both.

Khronos and OpenGL ES are trademarks of the Khronos Group.

Other company, product, and service names may be trademarks or service marks of others.