

Basic GPU techniques

© 2005-2017 Josef Pelikán
CGG MFF UK Praha

pepca@cgg.mff.cuni.cz
<http://cgg.mff.cuni.cz/~pepca/>



Content

- ◆ visibility computation (“depth-buffer”)
 - ◆ frame buffers
 - ◆ “double-buffering”, “triple-buffering”
- ◆ simple clipping (“scissor test”)
- ◆ stencil buffer
- ◆ transparency (“alpha blending”)
- ◆ anti-aliasing
- ◆ accumulation buffer (obsolete)
- ◆ lighting in GPU (fixed pipeline – obsolete)



Visibility

◆ depth-buffer

- ◆ Z-buffer or W-buffer

◆ Z-buffer

- ◆ 16-32 bits, typical **24 bits per pixel**
- ◆ mind the **nonuniform depth** transform into the **z** range (“**far / near**“ ratio)
- ◆ distant parts of a frustum have less precise depth!

◆ W-buffer

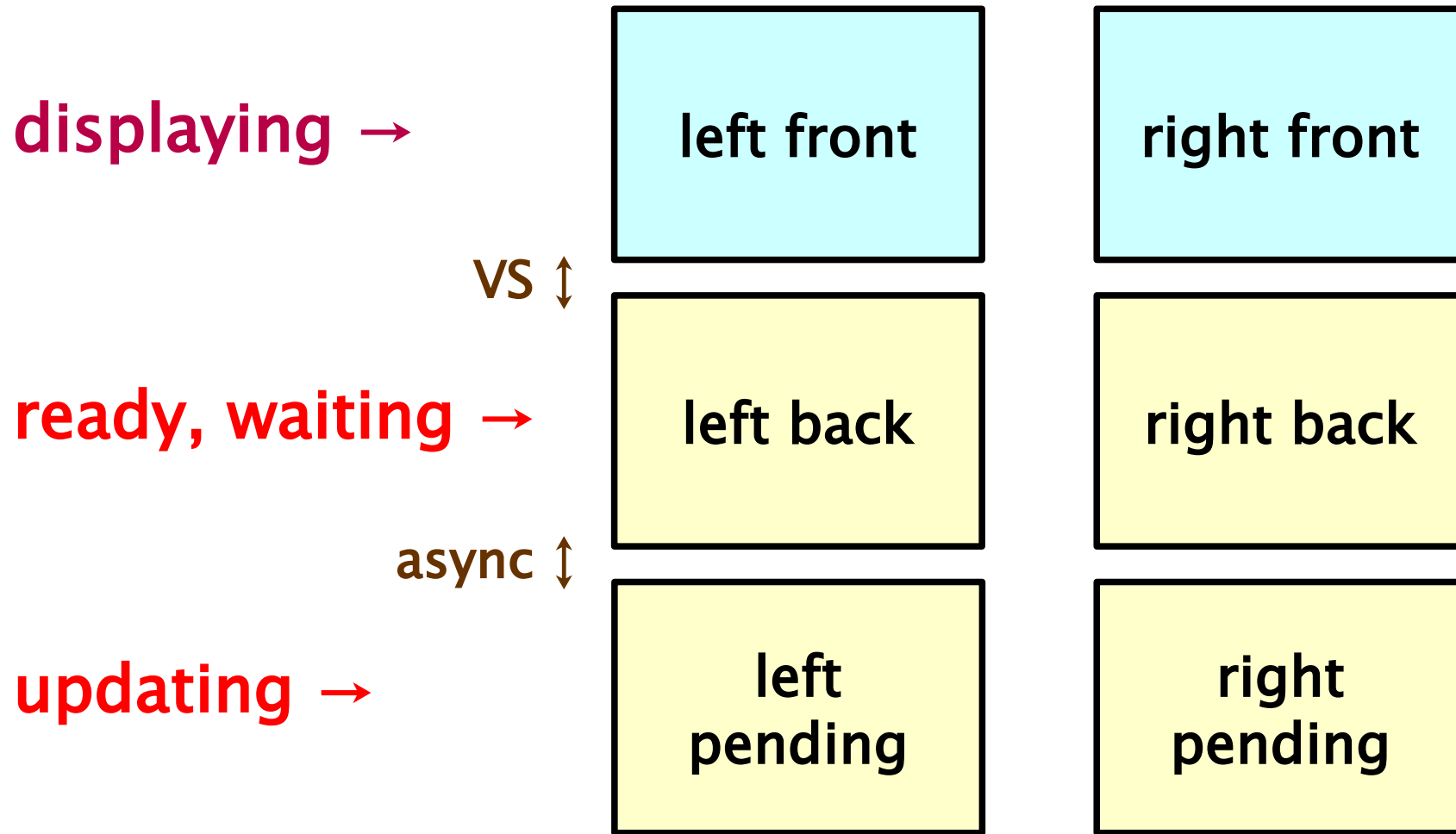
- ◆ no problems with depth nonlinearity



Frame buffers

- ◆ according to the depth settings 8 to 32 bpp (**RGB[A]**)
 - ◆ mostly **”true-color“ (24-32 bpp)** today
- ◆ **“double-buffering“**
 - ◆ updating buffer (**“front“**), displaying (**“back“**)
 - ◆ **buffer swapping:** “flipping“ or “blitting“
 - flipping: fast HW switch
 - blitting: bit-blt operation (archaic)
- ◆ **“triple-buffering“**
 - ◆ one more buffer (**“pending“**) for better load-balancing (multi-thread) and stable fps

Frame buffers, triple buffering





Buffers

- ◆ selective buffer usage
 - ◆ **write permission** (even to the individual bit-planes)
 - ◆ **depth-test enable** (setting compare operation)
- ◆ more output buffers for rendering – “**multiple render targets**”
 - ◆ used to be nonstandard (extensions)
 - ◆ “**pixel buffer**” (pbuffer) – direct output to texture memory (faster than `glCopyTexImage*()`)
 - ◆ today (OpenGL 3.0+): **Frame Buffer Object (FBO)**
 - `glGenFramebuffers()`, `glBindFramebuffer()`, `glFramebufferParameter()`, ..



Fragment operations

- ◆ the final phase of the “**fragment utilisation**“ (“Per-Fragment Operation”) after fragment processor
- ◆ fragment operation order:
 1. **scissor** test
 2. **alpha** test
 3. **stencil** test
 4. **depth** test
 5. **blending**
 6. **dithering**
 7. **logical buffer-write** operation



Simple tests

◆ “scissor test”

- ◆ fast and simple test
- ◆ clipping against a rectangular area (HW)

◆ ”alpha test”

- ◆ fast test
- ◆ comparing the **α -component** of a fragment to the reference value
- ◆ configurable **comparison operator** (GL_NEVER, GL_LEQUAL, ...)

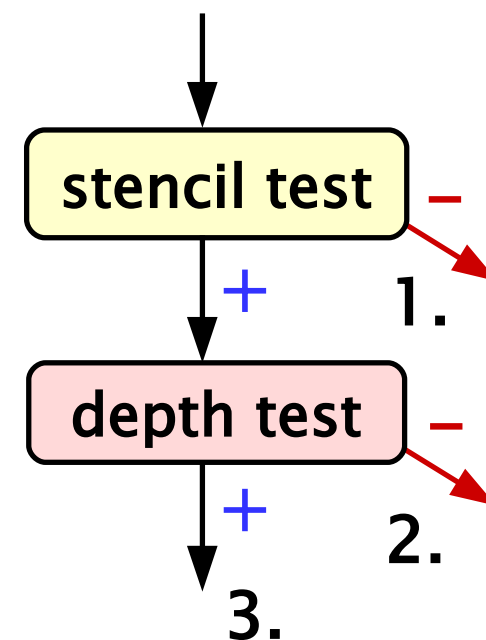


Stencil buffer

- ▶ additional **screen-size buffer**
 - ◆ typically **8 bits per pixel** (uint8)
 - ◆ can **restrict rendering of fragments** (by canceling further fragment processing)
 - ◆ configurable **buffer-update operation**
 - ◆ configurable **stencil-test operation**

- ▶ **OpenGL**: three different **update modes**

1. fragment **did not pass** the **stencil-test**
2. fragment **passed** the stencil-test, but **did not pass** the **depth-test**
3. fragment **did pass** both tests





Stencil operation

- ◆ **stencil modifications** (independent setup for each of the three modes)
 - ◆ GL_KEEP, GL_ZERO, GL_REPLACE, GL_INVERT
 - ◆ GL_INCR, GL_INCR_WRAP, GL_DECR, GL_DECR_WRAP
- ◆ **stencil test**
 - ◆ comparison to the configurable reference value
 - ◆ GL_NEVER, GL_ALWAYS, GL_LESS, GL_LEQUAL, ...
- ◆ **applications**
 - ◆ multi-pass algorithms: shadow casting, CSG rendering, mirrors, water surface, portal/cockpit view, ...

Transparency – “alpha blending”

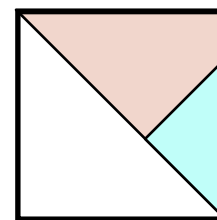
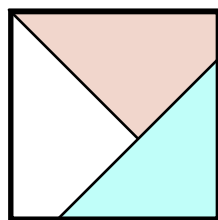
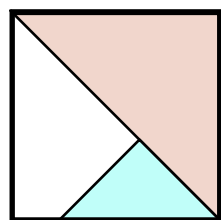


- ◆ **composition of pixels** with individual **opacity** (“transparency”)
 - ◆ **semi-transparent objects** (e.g. billboards, imposters)
 - ◆ **anti-aliasing** (partial coverage of boundary pixels)
- ◆ **frame-buffer** can store opacity α (A, “alpha”)
 - ◆ pixel formats **RGBA, BGRA, ..**
- ◆ **back-to-front rendering** doesn't need the α component in the frame buffer
 - ◆ rendered **fragments must have opacity**



Transparency – composition

- ◆ **composition** of semi-transparent pixels:
 - ◆ binary operations only (serial processing)
 - ◆ **linear combination** of the RGB[A] components
 - ◆ coefficients are fully configurable
 - ◆ **OpenGL**: specific coefficients for source and target, color and α -channel
- ◆ see **α -operations** (OVER, ATOP, HELD_OUT_BY, ...)



Scene with semi-transparent objects

- ◆ **rendering order** does matter!
 - ◆ usually **back-to-front rendering**
- ◆ **universal approach** for **combination** of opaque and semi-transparent objects in a scene:
 - ◆ RGBA frame-buffer not needed, RGB is sufficient
 - ◆ two passes:
 1. render **opaque objects** (arbitrary order, **depth-buffer on**)
 2. **depth-buffer write off** (but still used for **testing**)
 3. render **semi-transparent** objects back-to-front using the “**OVER**” operation (**depth test** is still **on**)

Anti-aliasing



◆ HW implementation ?

- ◆ basic primitives (points, lines, triangles) could have formulas for border-pixel coverage
- ◆ remember the back-to-front order!

◆ OpenGL anti-aliasing options:

- ◆ “**multisampling**” (MSAA) - number of (~equal) fragments rendered one over another using small (subpixel) shift. **Fragment shader** executed 1× per pixel, **depth/stencil tests** are performed multiple times per pixel
- ◆ “**supersampling**” (SSAA) – images are rendered in a higher resolution and then filtered to an original resolution. **Fragment shader** executed multiple times per pixel

Accumulation buffer (obsolete)



- ◆ replaced by **FBO functionality** in recent versions
- ◆ needs **HW support** (virtually one more buffer)
 - ◆ **anti-aliasing** (“multisampling“)
 - ◆ **motion blur**
 - ◆ **depth of field** simulation
 - ◆ **soft shadows**
- ◆ **technique**
 - ◆ accumulation buffer setup
 - ◆ **multiple** scene rendering (whole/partial)
 - ◆ for each rendering phase: transfer to the accumulation buffer using proper **combination-operation**



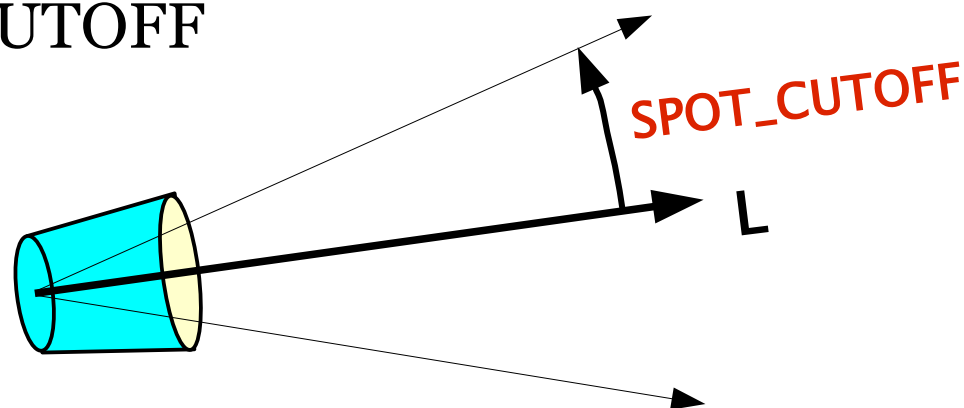
Lighting on GPU (obsolete)

- ◆ simple lighting model (**Blinn-Phong**)
 - ◆ “**ambient**” – the same color as diffuse
 - ◆ “**diffuse**” – diffuse component (ideal diffuse reflection, Lambert material)
 - ◆ “**specular**” – specular reflection (Phong)
- ◆ **normal vectors** at vertices
 - ◆ GPU normal computation (not FFP)
- ◆ **primary** (diffuse) and **secondary** (specular) colors and appropriate **material constants**
- ◆ **light sources**: positions, colors
 - ◆ limited number of sources (HW restriction)



Light sources (obsolete)

- ◆ **point source**
 - ◆ omnidirectional, finite distance
- ◆ **directional source**
 - ◆ parallel light rays = source in infinity
- ◆ **spotlight**
 - ◆ directional source in finite distance
 - ◆ intensity falls off with deviation from the L axis
 - ◆ limit angle `GL_SPOT_CUTOFF`





Light attenuation (obsolete)

- ◆ all sources are attenuated **by a distance**
 - ◆ quadratic polynomial

$$Att(d) = \frac{1}{k_C + k_L d + k_Q d^2}$$

- ◆ additional **spotlight** attenuation:

$$Spot(L, V) = (\cos \widehat{LV})^{SE}$$

L ... spotlight direction

V ... light-to-target direction

SE ... GL_SPOT_EXPONENT



Lighting summary (obsolete)

◆ **primary color** (diffuse light)

$$Pri = Emiss_{mat} + Amb_{lightmodel} \cdot Amb_{mat} + \sum_{i=1}^N Att_i \cdot [Amb_{light} \cdot Amb_{mat} + \cos \alpha \cdot Diff_{light} \cdot Diff_{mat}]_i$$

◆ **secondary color** (specular highlight)

- ◆ needs not be implemented
- ◆ applies after texturing

$$Sec = Spec_{mat} \cdot \sum_{i=1}^N [Att \cdot (\cos \beta)^{shininess} \cdot Spec_{light}]_i$$

Sources



- ◆ Tomas Akenine-Möller, Eric Haines: ***Real-time rendering, 2nd edition***, A K Peters, 2002, ISBN: 1568811829
- ◆ OpenGL ARB: ***OpenGL Programming Guide, 4th edition***, Addison-Wesley, 2004, ISBN: 0321173481
- ◆ J. Žára, B. Beneš, J. Sochor, P. Felkel: ***Moderní počítačová grafika, 2nd edition***, Computer Press, 2005, ISBN: 8025104540