

OpenGL API

- library of routines to control graphics main kinds of things it does
- calls to specify triangles
- calls to specify vertices
- calls to specify/load textures
- calls to specify shaders
- calls to set various options in the pipeline

system issues

- needs include and library files, and installed drivers.
- cross platform
- the alternative would be DirectX graphics
 - dominant for PC games
 - not cross platform
- we will use 2.0
- “extensions” have been added to each version.
 - glsl is accessible through the proper extensions
 - if supported by card and driver
- we use the *glew* library to access the extensions
 - include and library files.

glut

- library of functions to talk with the windowing system
 - include and library files.
- open up windows of some size
- glut can inform you when some “event” occurs
 - mousemove, buttonpress, windowresize
- you register callback functions with glut
 - the callback function is called when the event occurs
 - and passed relevant info (ex. the mouse location)
- note: must flip the mouse 'y' coords
- cross platform (windows/X)
 - some other competitors.

glsl

- gl shading language
- you write small programs to be executed for each vertex.
- you write small programs to be executed for each fragment
- you tell OpenGL to compile/link/load these “shaders”

- they operate in parallel on the vertices and fragments (SIMD)
- competitors
 - microsoft’s hlsl:
 - * dominant for pc games
 - * only works with directX
 - nvidia’s cg
 - almost identical to hlsl
 - also works with openGL
 - * need cgGL library
 - future support unclear.

glsl information flow

- mostly through global variables
 - some have fixed names, others you create
- vertex shader inputs
- uniform: set infrequently
 - things like the current camera view description
 - lighting information
- attribute: set for each vertex
 - gl_Vertex and user defined
- vertex shader “varying” outputs
 - gl_Position and user defined
 - these values are properly interpolated at each fragment
- fragment shader inputs
- uniform: see above
- varying: comes out of the interpolated vertex outputs
- output goes (possibly) to the framebuffer
 - gl_FragColor
- NB: the shader compiler will kill off unused variables, and your C program will not be able to find pointers to them.