

# **Compute Shaders**

**CSCI 4239/5239**

**Advanced Computer Graphics  
Spring 2017**

# Purpose of Compute Shaders

- Lightweight general purpose computing
  - Perform arbitrary computations outside of the vertex/fragment paradigm
  - Access to textures and buffers
  - Does not require additional drivers or run time
  - Easier initialization and invocation than OpenCL
- Requires OpenGL 4.3
  - Example 22 requires GL 4.4

# Using the Compute Shader

- Compiles just like other shaders
  - CreateShader(prog, GL\_COMPUTE\_SHADER, file)
  - Link only one shader into program
- Bind and access buffers
  - glBindBuffer(GL\_SHADER\_STORAGE\_BUFFER, x)
  - glBufferData()
    - Set size and type of buffer
  - glMapBufferRange()
    - Access to buffer on CPU
  - glUnmapBuffer(GL\_SHADER\_STORAGE\_BUFFER);

# Running the Compute Shader

- `glUseProgram(compute_shader)`
  - Select program
- `glDispatchCompute(Nx,Ny,Nz)`
  - Run Nx,Ny,Nz groups
- `glDispatchComputeGroupSize(Nx,Ny,Nz,x,y,z)`
  - Set both number of groups and work groups
- `glMemoryBarrier(GL_SHADER_STORAGE_BARRIER_BIT)`
  - Wait for compute shaders

# Pre-set Variables in Shader

- `uvec3 gl_NumWorkGroups`
- `uvec3 gl_WorkGroupSize`
- `uvec3 gl_WorkGroupID`
- `uvec3 gl_LocalInvocationID`
- `uvec3 gl_GlobalInvocationID`
- `uvec3 gl_LocalInvocationIndex`
- $gl\_GlobalInvocationID = gl\_WorkGroupID * gl\_WorkGroupSize + gl\_LocalInvocationID$
- $gl\_LocalInvocationIndex = gl\_LocalInvocationID.z * gl\_WorkGroupSize.y * gl\_WorkGroupSize.x + gl\_LocalInvocationID.y * gl\_WorkGroupSize.x + gl\_LocalInvocationID.x$