

# Lesson 8 – Geometry shaders

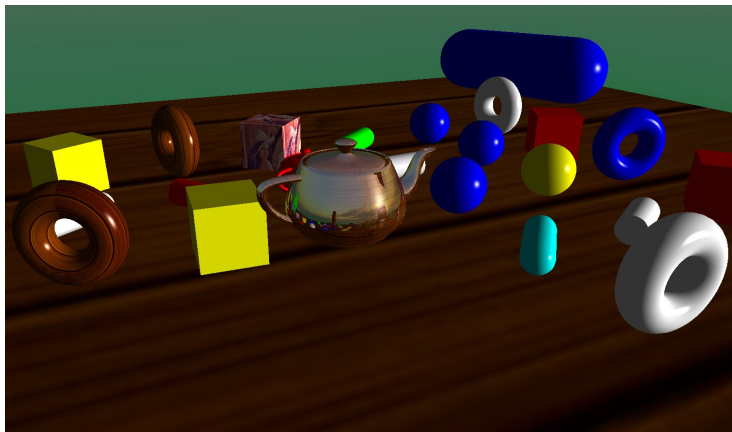
## Environment mapping

### PV227 – GPU Rendering

Jiří Chmelík, Jan Čejka  
Fakulta informatiky Masarykovy univerzity

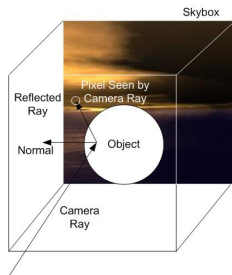
29. 10. 2019

## Intermezzo – Environment mapping



Reflections: Environment mapping

# Environment mapping



Source: Wikipedia

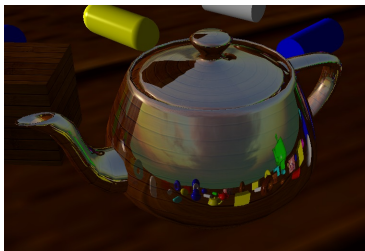
- $\overrightarrow{ReflectedRay} = \overrightarrow{CameraRay} - 2 \cdot \vec{N} \cdot \text{dot}(\vec{N}, \overrightarrow{CameraRay})$
- In GLSL:  $\text{ReflectedRay} = \text{reflect}(\text{CameraRay}, N)$
- Assumes  $N$  is normalized

# Task: Implement environment mapping

- **Task 1:** Implement environment mapping in *reflection\_fragment.glsl*
  - ▶ Mix the environment reflection with the color of the object

# Updating the cube map

- When the surrounding changes, the cube map with the environment should be updated.
- Six faces of the cube map means:
  - ▶ six cameras,
  - ▶ six framebuffer objects
  - ▶ six times traversing the scene
- Already implemented in the code.



# Layered rendering

- Renders into multiple textures at the same time
  - ▶ Good for cube maps, stereo rendering etc.
- Different from attachments of FBOs
  - ▶ Attachments: Primitives are rasterized at the same places
  - ▶ Layers: Each layer has different primitives
- Renders triangles into layered textures:
  - ▶ cube maps (6 layers)
  - ▶ 2D texture arrays
  - ▶ 3D textures, 1D texture arrays, cube map arrays
- Use *glFramebufferTexture* to attach a layered texture into a framebuffer
  - ▶ All textures at all attachments must be layered
- Another usage of geometry shaders
- New output variable in geometry shaders: *gl\_Layer*
  - ▶ Specifies the index of the layer into which the primitive is sent

# Updating the cube map – layered rendering

- Updating all faces simultaneously means:
  - ▶ six cameras available at the same time
  - ▶ one framebuffer object with all faces
  - ▶ traversing the scene once
  - ▶ special vertex and geometry shaders

# Task: Implement layered rendering

- **Task 2:** Implement layered rendering in *texture\_to\_cube\_geometry.glsl* and compare the rendering speed
  - ▶ Generate 6 triangles (18 vertices), one for each face
  - ▶ Some vertex data do not change, they are computed in VS
    - ★ Pass them through geometry shader without change
  - ▶ Some vertex data (*gl\_Position* and *gl\_Layer*) are different for each face.
    - ★ Compute their values in geometry shader
- **Optional task:** Implement the same for the skybox in *skybox\_to\_cube\_geometry.glsl*
- Test on the central object, use cube or sphere without reflections



# Instanced geometry shader

- Problem: the geometry shader processes 18 vertices sequentially, not in parallel
- Possible solution: Instanced geometry shaders
  - ▶ Similar to instancing
  - ▶ Geometry shader is run multiple times per each input primitive
  - ▶ In GS: Instances = Invocations
  - ▶ Defined in geometry shader:
    - layout (triangles, invocations = 6) in;*
    - layout (triangle\_strip, max\_vertices = 3) out;*
  - ▶ Special variable *gl\_InvocationID*:
    - ★ Only in geometry shader
    - ★ Similar to *gl\_InstanceID*

# Task: Implement instanced geometry shaders

- **Task 3:** Implement instanced geometry shaders in *texture\_to\_cube\_invocations\_geometry.glsl* and compare the rendering speed
  - ▶ Generate 1 triangle (3 vertices), 6 invocations, one invocation for each face
- **Optional task:** Implement the same for the skybox in *skybox\_to\_cube\_invocations\_geometry.glsl*

# Parallelize even more

- Use instancing, i.e. render each object six times.
- Everything is computed in vertex shader, all vertices in parallel
- Geometry shader only copies the data of each vertex and sets *gl\_Layer*
- **Task 4:** Implement this in *texture\_to\_cube\_instancing\_vertex.glsl* and *texture\_to\_cube\_instancing\_geometry.glsl* and compare the rendering speed
  - ▶ **Option:** Implement the same for the skybox in *skybox\_to\_cube\_instancing\_vertex.glsl* and *skybox\_to\_cube\_instancing\_geometry.glsl*

# Parallelize even more, skip geometry shaders

- Modern graphics card may set *gl\_Layer* also in vertex shaders, thus skipping the geometry shader completely
  - ▶ We need OpenGL extension *GL\_ARB\_shader\_viewport\_layer\_array*, unfortunately, it is not available on computers in B311
    - ★ No, it **is** available, hurray!
- **Optional task 5:** Implement this in *texture\_to\_cube\_instancing\_no\_gs\_vertex.glsl* and in *skybox\_to\_cube\_instancing\_no\_gs\_vertex.glsl* and compare the rendering speed

# Geometry Shaders Today

- not for tessellation, surpassed by tessellation shaders
- probably not for culling (not necessary)
- expanding a point to a quad (particle systems), compete with instancing
- expanding a line to a quad (grass, hair), in combination with tessellation shaders
- transform feedback (outputs vertices back into VBOs)
- layered rendering, instanced geometry shaders
  - ▶ not enough parallel, compete with instancing, not necessary when *gl\_Layer* is set in vertex shaders