Preparation for Lesson 4

Lesson 4 is focused on deferred shading. For smooth lesson, review:

- Review Phong lighting model, especially how to evalute the illumination if you have the position, the normal, and the color of the material (not necessarily from the vertex shader).
- Review framebuffers, our fragment shader will output three variables.
- Review, what *glDepthMask* is good for and how to use it.
- Review the blending, function *glBlendFunc* and its parameters (*GL_ONE*, *GL_ZERO*, *GL_SRC_ALPHA*, *GL_ONE_MINUS_SRC_ALPHA*).
- Review what *discard* is good for in GLSL and how to use it.
- Review instancing, we will instance spheres.

Go through project 4 in the study materials. Focus on:

- We have a randomly generated scene, objects, and lights. You can go through the code that generates it, but it is not required, we will not change it. Note that the scene contains up to 150 point lights with an attenuation.
- We have two framebuffer objects. Note framebuffer object *GbufferFBO*, which was three textures attached to three attachments, so the fragment shader may output three variables, one into each textures. Also note that both FBOs are of the same size as the main window, so they must be resized every time the main window is resized and so must be the textures.
- Go through the code in render_scene for *if USE_FORWARD_SHADING* and shaders *notexture_vertex.glsl*, *texture_vertex.glsl*, *notexture_forward_fragment.glsl*, and *texture_forward_fragment.glsl*. It is the code that renders the scene as we did in previous lessons, it should be clear to you.
- The code should be compilable and runable, the scene can have up to 150 lights processed in the way we are used to. Note the time required to render a frame, which is displayed in GUI.

Project 4 also includes the following stuff, which has not been tought yet and which will be discussed at the lecture:

- Shader storage buffer objects (*GL_SHADER_STORAGE_BUFFER* in C++ and *buffer* in GLSL)
- Qualificators *flat* and *noperspective* in *evaluate_sphere_vertex.glsl* and in *evaluate_sphere_fragment.glsl*
- OpenGL queries for measuring the rendering time (variable *RenderTimeQuery* and the code around it)

Příprava na 4. cvičení

Na čtvrtém cvičení budeme probírat deferred shading. Pro hladký průběh cvičení si zopakujte a připravte:

- Zopakujte si Phongův osvětlovací model, zejména jak vyhodnotit osvětlení, máte-li pozici, normálu a barvu materiálu (ne nutně z vertex shaderu).
- Zopakujte si framebuffery, fragment shader bude mít na výstupu tři proměnné.
- Zopakujte si, k čemu slouží funkce *glDepthMask* a jak se používá.
- Zopakujte si míchání barev (blending), funkci *glBlendFunc* a její parametry (*GL_ONE*, *GL_ZERO*, *GL_SRC_ALPHA*, *GL_ONE_MINUS_SRC_ALPHA*).
- Zopakujte si, k čemu slouží v GLSL *discard* a jak se používá.
- Zopakujte si instancování, budeme instancovaně kreslit koule.

Projděte si projekt Cv4 ve studijních materiálech. Zaměřte se zejména na:

- Máme náhodně vygenerovanou scénu, objekty i světla. Kód generující scénu si můžete projít, ale nemusíte se na něj zaměřovat, upravovat to nebudeme. Všimněte si, že scéna obsahuje až 150 světel, která jsou bodová a mají útlum (attenuation).
- Máme 2 framebuffer objekty. Všimněte si zejména framebuffer objektu *GbufferFBO*, který má tři textury navázané na tři color attachmenty, takže můžeme z fragment shaderu dávat na výstup tři proměnné, do každé textury jednu. Také si všimněte, že oba FBO mají textury stejně velké jako je okno, do kterého kreslíme, takže je nutné při každé změně velikosti okna aktualizovat velikost těch tectur.
- Projděte si část kódu v render_scene pro *if USE_FORWARD_SHADING* a shadery *notexture_vertex.glsl*, *texture_vertex.glsl*, *notexture_forward_fragment.glsl* a *texture_forward_fragment.glsl*. Je to kód, který kreslí naši scénu tak, jak jsme byli doposud zvyklí, měl by vám být jasný.
- Kód by měl jít spustit, na počátku máme scénu s až 150 světly, kterou kreslíme tak, jak jsme byli doposud zvyklí. Všimněte si doby kreslení jednoho snímku, která se zobrazuje v GUI.

Ve Cv4 jsou navíc tyto věci, které ještě nebyly probrány a které budou probrány na přednášce:

- Shader storage buffer objekty (*GL_SHADER_STORAGE_BUFFER* v C++ a *buffer* v GLSL)
- Kvalifikátory *flat* a *noperspective* v *evaluate_sphere_vertex.glsl* a v *evaluate_sphere_fragment.glsl*

- OpenGL query pro měření doby výpočtu (proměnná *RenderTimeQuery* a věci okolo ní)