1 Examination and assessment

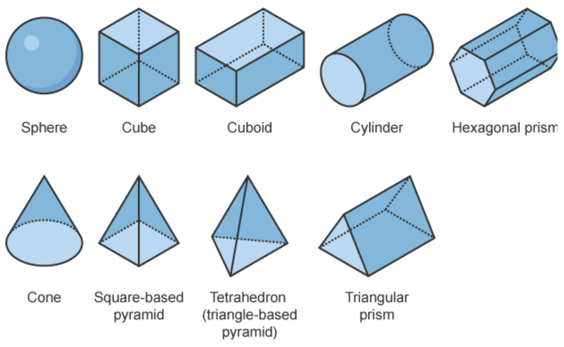
1.1 Introduction

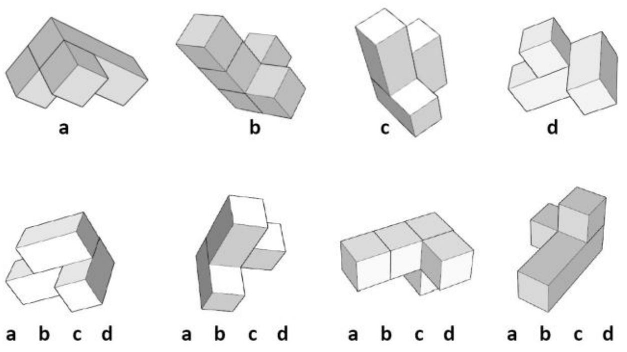
You are expected to work on an **individual** assignment that can be completed over the length of the course. The assignment is split up into smaller parts that focus on a certain topic and skills you should master. Two progress meetings allow you to discuss the quality of your work with an assessor.

1.2 Assignment

Please upload a .zip file containing a report and your c++ project files. The report documents the implementation of your assignment, relevant design choices and implementation techniques. Give your .zip file the following name: <studentnummer lastname firstname>.zip

1.2.1 Assignment, part 1: Primitives





Luna’s pipeline is able to render all sorts of 3D shapes like a box, a cylinder or a sphere. Find an interesting (unique) shape that isn’t supported yet and render it in a scene.

- Basic: Render a (unique-not included in the Luna library) fixed 3D shape with a known topology with a fixed number of vertices/edges.
- Intermediate: Render a (unique) 3D shape on the CPU with a topology/mesh with a fixed number of vertices/edges. This time the user should be able to configure the amount of detail or set a property influencing the mesh. The CPU calculates the geometry and sends it (in a buffer) to the GPU for rendering.
- Advanced: Render a configurable (unique) 3D shape on the GPU. This time you let the GPU calculate extra geometry details in the Geometry Shader.

1.2.2 Assignment, part 2: Texturing

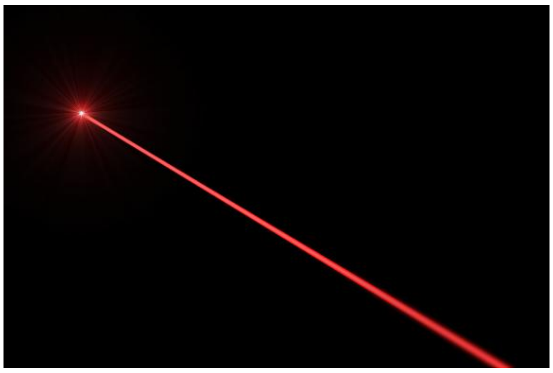
In this assignment you are going to apply textures to an object to create a realistically looking smart phone in your graphics application.

- Basic: **Unwrapping**.
  First, create a geometric object that resembles the geometry of your smart phone. Next, take pictures of all sides of your smart phone and combine these into a *single* texture image to be applied to the object using the "Unwrapping" principle. Now apply this texture image to your geometric object such that you obtain a realistically looking smart phone in your application.
- Intermediate: **Multiple Textures and Billboarding**.
  Take an image with the camera of your (physical) smart phone and store this in an additional texture file. Now apply this additional texture to your virtual smart phone such that it looks like you took the picture with your virtual smart phone: the texture will cover the "screen space" of your virtual smart phone. Be aware that you are not allowed to create any new geometry: the texture has to be applied to the same face that is already textured by the texture from the basic assignment. That is, you will have to apply *two* textures to *one* face of your virtual smart phone.
- Advanced: **Dynamic Texturing**.
  Now, in stead of taking a picture with the camera of your physical phone, render an image as if taken from the camera of your virtual smart phone and apply this texture to the virtual smart phone like you did in the intermediate assignment. So, align your viewpoint with the virtual phone's camera, render the scene (without the smart phone), store this in a texture, reposition the viewpoint, render the scene again including the smart phone and apply the texture. Make sure there are some extra objects in the scene to show that "the camera is working".

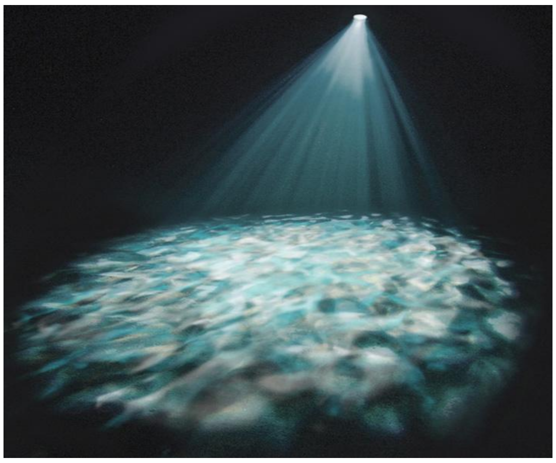
1.2.3 Assignment, part 3: Lighting

Luna's pipeline is able to render several types of light. Extend an existing c++ project to get an interesting light effect.

- Basic: Render a magic wand that emits a globe of light at the tip (point light).



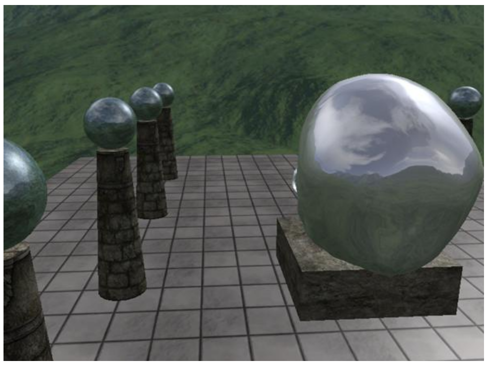
- Intermediate: Render configurable red flash light with a variable cone (spot light).



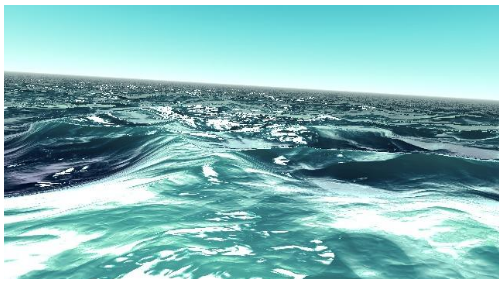
- Advanced: Render a light (effect) hovering over water.

1.2.4 Assignment, part 4: Shading

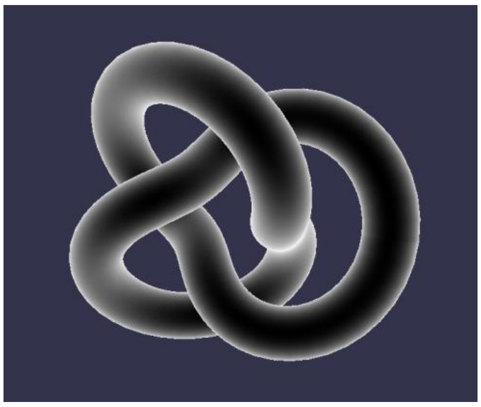
An important part of 3D graphics is the pixel shader and its possibilities.



- Basic: Make the Luna 'reflective chrome' shader *adjustable* by keyboard; The user can adjust the amount of reflectiveness, from 0 tot 100%.



- Intermediate: Replicate an *existing* shader from ShaderToy into your project. (Shadertoy already gives you GLSL code)



- Advanced: Program a *custom* "Fresnel" shader, shown above (the amount of the white color is affected by the *angle* between the surface and the camera). Both the color of the effect, as well as the angle has to be adjustable (by mouse / keyboard).

1.2.5 Assignment, part 5: Choice

Program an advanced 3D technique of your own choice, where you display your knowledge and skill in either *performance* enhancing techniques or advanced *visual* effects.
Examples for visual effects: Dynamic shadow maps, deferred rendering, post render Ambient Occlusion, motion blur, depth of field, terrain rendering, volume rendering etc.
Examples for performance: batching, (simple) voxel engine (Minecraft), hardware instancing, .. etc.

1.3 Grading rubric

Advanced Graphics Programming Rubric: Grade = 1+Score/1,5.	
Student Number:	Student Name:
De student delivers a GIT repository that includes a report and several c++ projects (assignments). The report documents the rendering technique applied in each c++ project (assignment) and motivates the score on each assessment category.	

Category	**Basic**	**Intermediate**	**Advanced**
	1 pt	**+1 pt**	**+1 pt**
1. Primitives (3)	Renders a **unique static** (hard coded) primitive topology using Luna's pipeline.	Renders **dynamic** primitive topology (mesh) on the CPU using Luna's pipeline.	Renders dynamic primitive topology (mesh) **on the GPU** using a geometry shader in Luna's pipeline.
2. Texturing (3)	Render a virtual smart phone with an unwrapped texture.	Apply a second texture to the virtual phone taken with your physical camera.	Render a dynamic texture and apply it to the virtual phone as if the camera of the virtual phone is switched on.
3. Lighting (3)	Implement a magic wand (not the particles)	Implement a (laser) beam with a (configurable) variable cone.	Implement a light (effect) hovering over water.
4. Shading (3)	Make one or more parameters of an existing shader *at runtime* (with keyboard/mouse)	Implement an *existing* pixel shader (from Shadertoy)	Implement a custom shader which uses the angle between surface and camera, with parameters adjustable at runtime (mouse/keyboard)
5. Choice (1,5)	Program an advanced 3D technique of choice, where you display your knowledge and skill in either *performance* enhancing techniques or advanced *visual* effects. (Consult *beforehand* with your assessor if your subject is valid)		

Remarks: