# Fondamenti di Computer Graphics LM
## GLSL

Questa esercitazione puo' essere eseguita sia in ambiente Windows che Linux.

1. **Example 1: Wave motion**

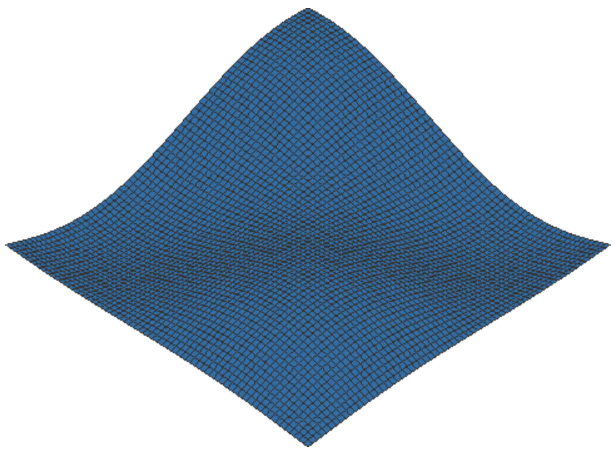

Figure 1: A screenshot from the `WAVE` application.

The `WAVE` application shows how to modify the geometry of an height field mesh by modifying the vertex positions in the vertex shader (see Fig. 1). The `WAVE` application is implemented in the source files:

- WAVE/wave.c
- WAVE/v.glsl
- WAVE/f.glsl

Extend the `WAVE` application implementing the following features:

(a) when the user clicks the left mouse button, set a different amplitude, cycling through the values: $[0.05, 0.1, 0.2]$.

(b) when the user clicks the right mouse button, set a different oscillation frequency, cycling through the values: $[0.0005, 0.001, 0.002]$.

2. **Example 2: Particle System**

The `PARTICLE` application implements a simple particle system simulation (see Fig. 2). The particle position is computed in the vertex shader based on velocity and acceleration (gravity) at the current simulation time step. Press the space key to restart the system and randomize particles velocities and color. The `PARTICLE` application is implemented in the source files:

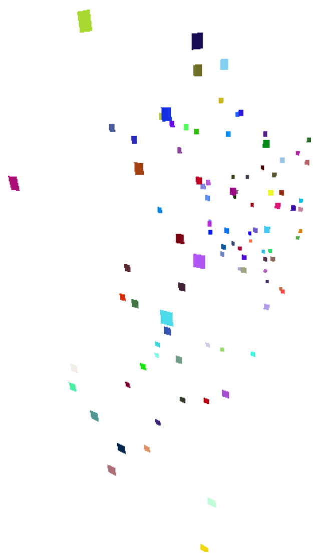

Figure 2: A screenshot from the `PARTICLE` application after implementing particles movement in the $z$ direction.

- PARTICLE/particle.c
- PARTICLE/v.glsl
- PARTICLE/f.glsl

Extend the `PARTICLE` application implementing the following features:

(a) height dependent point size. The higher, the bigger ( *TIP:* use the built-in variable `gl_PointSize`)

(b) make the particles move also in the $z$ direction

3. **Example 3: Phong Lighting**

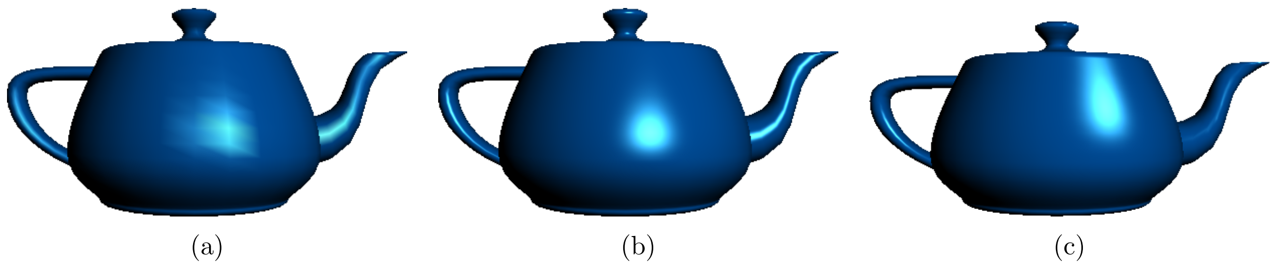

Figure 3: Phong lighting model implemented in the `PHONG` application with fragment color interpolation (a) and with normals interpolation in the fragment shader (b). After implementing the exact reflection ray (c).

The `PHONG` application implements the Phong lighting model in two different GLSL programs with fragment color interpolation and with a more precise normal interpolation performed in the fragment shader (see Fig. 3). The `PHONG` application is implemented in the source files:

- PHONG/phong.c
- PHONG/v1.glsl
- PHONG/f1.glsl
- PHONG/v2.glsl
- PHONG/f2.glsl

Extend the `PARTICLE` application implementing the following features:

(a) add a third shader (starting from `v2.glsl` and `f2.glsl` that, in the computation of the specular coefficient $K_s$, implements the exact light reflection ray instead of the approximate half-way vector. See Fig. 3c for the expected result and note that it's not very different since the half-way vector represents a good approximation of the reflection ray.

(b) show three teapots in the same scene with the three different shaders

4. **Example 4: Toon Shading**

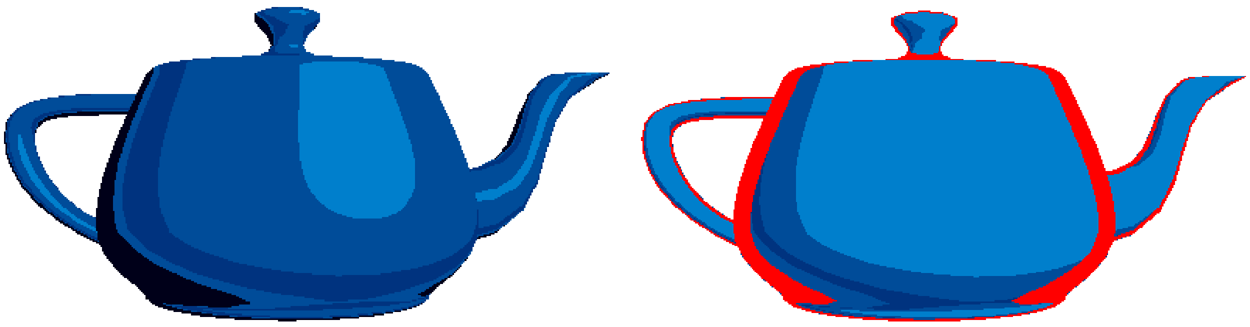

Figure 4: A screenshot from the `NONPHOTO` application (a). After adding a red outline (b).

The `NONPHOTO` implements a non-photorealistic rendering commonly known as "Toon shading" via vertex and fragment shaders (see Fig. 4a). The `NONPHOTO` application is implemented in the source files:

- NONPHOTO/nonphoto.c
- NONPHOTO/v.glsl
- NONPHOTO/f.glsl

Extend the `NONPHOTO` application implementing the following features:

(a) add an outline/silhouette effect (see Fig. 4b). *TIP:* Think about the angle formed by the eye vector $\overrightarrow{E}$ with the normal vector $\overrightarrow{N}$.

5. **(OPTIONAL) Modify as you like one of the following shaders: MORPH/BUMPMAP/CUBEMAP.**