A. Name the three kinds of reflected light that we have used. If you had to pick just one of them to have in your graphics engine, which would you pick? Which quantities are used in that lighting calculation? Where is the calculation carried out? How does each quantity get there?

ambient: surface diffuse color (any), light color (uniform), ambient intensity

diffuse: surface diffuse color (any), light color (uniform), normal direction (varying), light direction [camera position (uniform), fragment position (varying)]

specular: surface specular color (any), light color (uniform), camera direction, reflected direction [normal direction (varying), light direction (camera position [uniform], fragment position [varying])]

B. Summarize the shadow mapping algorithm. How does it depend on the style of light used? How does it depend on the number of lights used?

common: extra rendering pass for each light, scene rendered from light's point of view, capturing depth but not color, later used as texture in main rendering pass, where fragment's distance from light is compared to depth map, and light is included or not

spot light: perspective frustum, containing spot light

directional light: orthographic frustum, containing camera frustum

omnidirectional light: six perspective frustums, cube map

C. Given that 000pixel.o is written in OpenGL using GLFW, how do you think it works?

`pixSetRGB` alters a single texel in a texture, on each frame this texture is mapped across a rectangle that fills the window

achieves high frame rates by redrawing only if that texture has been altered

D. Our software engine had a `renRenderer` data type. What is the corresponding entity or entities in our hardware engine?

shader program: `colorPixel`, `transformVertex`, `updateUniform`, rendering settings

scene graph: `updateUniform`

`camCamera`: rotation, translation, projection

OpenGL viewport: viewport