A.A. The chronological order is

- configuration of uniforms, mesh, textures
- modeling isometry
- inverse camera isometry
- projection
- clipping at the near plane
- viewport, homogeneous division (in either order)
- rasterization and interpolation
- choice of fragment color, choice of fragment depth (in either order)
- depth test
- writing of fragment to pixel.

A.B. Typically the vertex shader includes the modeling isometry, inverse camera isometry, and projection. The fragment shader includes the choice of fragment color and the choice of fragment depth. [OpenGL chooses the depth for us, so I did not deduct points for leaving that out of the list above.]

A.C. OpenGL 3.2 does clipping, viewport, homogeneous division, rasterization and interpolation, choice of fragment depth, depth test, and writing of fragment to pixel. [The user can affect the fragment depth, so I did not deduct points for leaving that out of the list above.]

A.D. Perspective-corrected interpolation requires changes to homogeneous division and the choice of fragment color. (In theory, the whole fragment shader might change, so the choice of fragment depth might also change. But we have seen no examples of that, so I didn't require it.)

B.A. [Instead of drawing, I'll describe.] There are four roots to the scene graph: water, land, temple base, and robot giraffe baby body. The water node has no descendants. The land node has three children, namely the three tree trunks; all of those tree trunks share the same three children. (To emphasize: There are exactly six nodes implementing all three trees.) The temple base node has five children: temple walls, roof, window, standing column, lying column. The robot giraffe baby body node has five children, for the legs and neck; the neck node has the head node as a child.

B.B. We were given the coordinates of the stained glass window as $\vec{a} = (4.0, 0.5, 4.0)$, $\vec{b} = (6.0, 0.5, 4.0)$, $\vec{c} = (4.0, 0.5, 1.0)$ relative to its parent. Its parent, the temple base, has no parent, which simplifies our work. We transform $\vec{a}$, $\vec{b}$, $\vec{c}$ by the temple base's isometry to get the world coordinates.

C. In the fragment shader, we replace the uniform for the light direction $\vec{d}_{\text{light}}$ with a uniform for the light position $\vec{p}_{\text{light}}$. We already have a varying for the fragment position $\vec{p}_{\text{fragment}}$. We compute $\vec{p}_{\text{light}} - \vec{p}_{\text{fragment}}$, scale it to have length 1, and store that vector in $\vec{d}_{\text{light}}$. The rest of the calculations are unchanged.

D. Before we had textured light, our diffuse contribution was

$$i_{\text{diff}} \cdot \vec{c}_{\text{diff}} \cdot \vec{c}_{\text{light}}.$$

The idea of textured light is that the light color gets modulated by the glass color before it hits the object. So $\vec{c}_{\text{light}}$ is replaced by $\vec{c}_{\text{glass}} \cdot \vec{c}_{\text{light}}$ in the diffuse contribution:

$$i_{\text{diff}} \cdot \vec{c}_{\text{diff}} \cdot \vec{c}_{\text{glass}} \cdot \vec{c}_{\text{light}}.$$

The same goes for the specular contribution: $\vec{c}_{\text{glass}} \cdot \vec{c}_{\text{light}}$ replaces $\vec{c}_{\text{light}}$.