APROBAT
De Miruna la 21 Mai 2019, 2:00 pm

Chindea Miruna

Laboratory work 11

1 Objectives

This laboratory work highlights a technique for hidden surface removal using the z-buffer algorithm.

2 Theoretical background

Determining which surfaces are visible and which are hidden is a fundamental problem in Computer Graphics. Usually, it is highly inefficient to process objects that will never appear in the final image.

We have already seen a technique for removing hidden surfaces in Laboratory work 8, namely Back Face Culling, which removes primitives on the back of opaque objects from the graphics pipeline. The z-buffer algorithm removes primitives that would appear behind other opaque objects.

The z-buffer, also known as the depth-buffer, is a 2D array having the same dimensions as the framebuffer (the final raster image) that keeps, for each pixel, the distance from the center of projection to the surface that gave the pixel's final color. Figure 1 shows an example of the contents of the frame-buffer and the z-buffer when drawing a red triangle parallel with the Z axis and situated at a depth of $z = -5$.

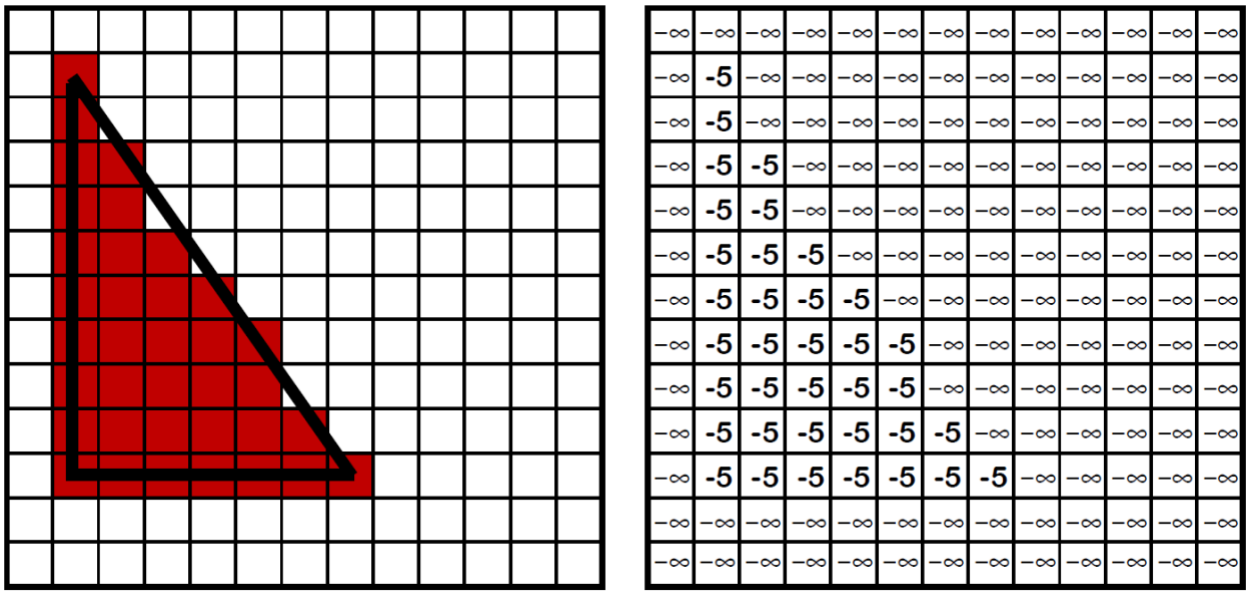

Figure 1 – Frame-buffer and z-buffer for drawing a red triangle with $z = -5$

When a new primitive is rasterized, each of its resulting candidate pixels' depth is compared with the existing depth value in the z-buffer. The final pixel is updated only if the candidate pixel is closer to the center of projection than the value already in the buffer. If the pixel is updated, the existing depth value in the z-buffer is replaced by the new pixel's depth.

For example, when drawing a yellow triangle parallel with the Z axis and situated at a depth of $z = -7$, if there are no previously processed primitives, the resulting frame-buffer and z-buffer are the ones illustrated in Figure 2.

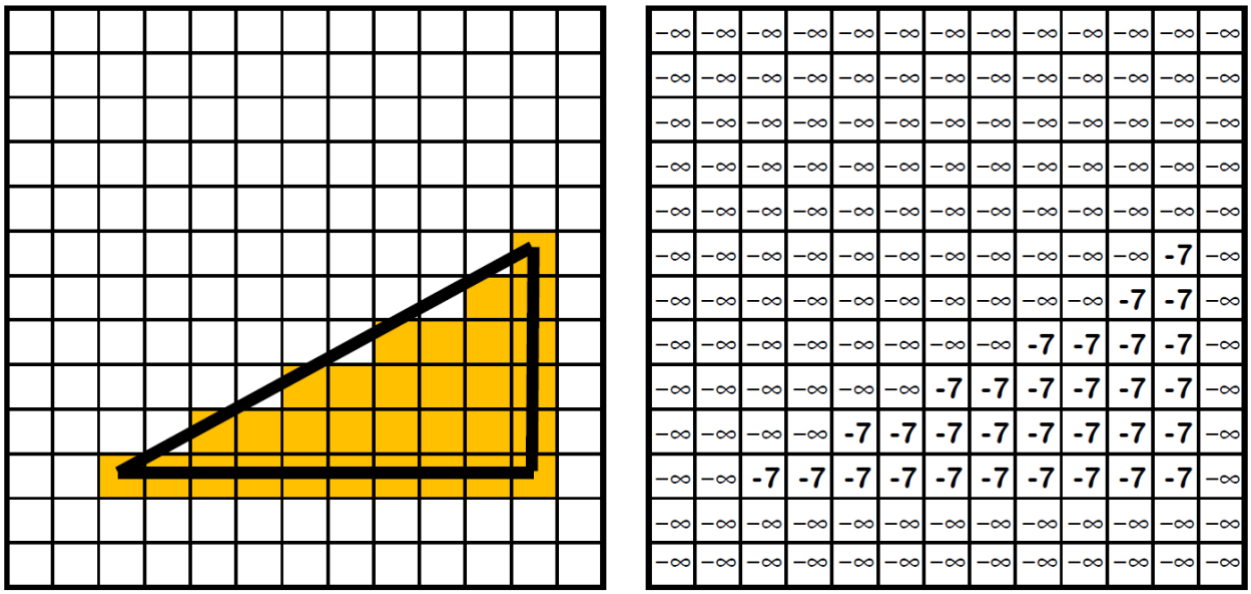

Figure 2 - Frame-buffer and z-buffer for drawing a yellow triangle with $z = -7$

However, if the yellow triangle is rasterized after the red triangle, then the contents of the frame-buffer and the z-buffer are as illustrated in Figure 3.

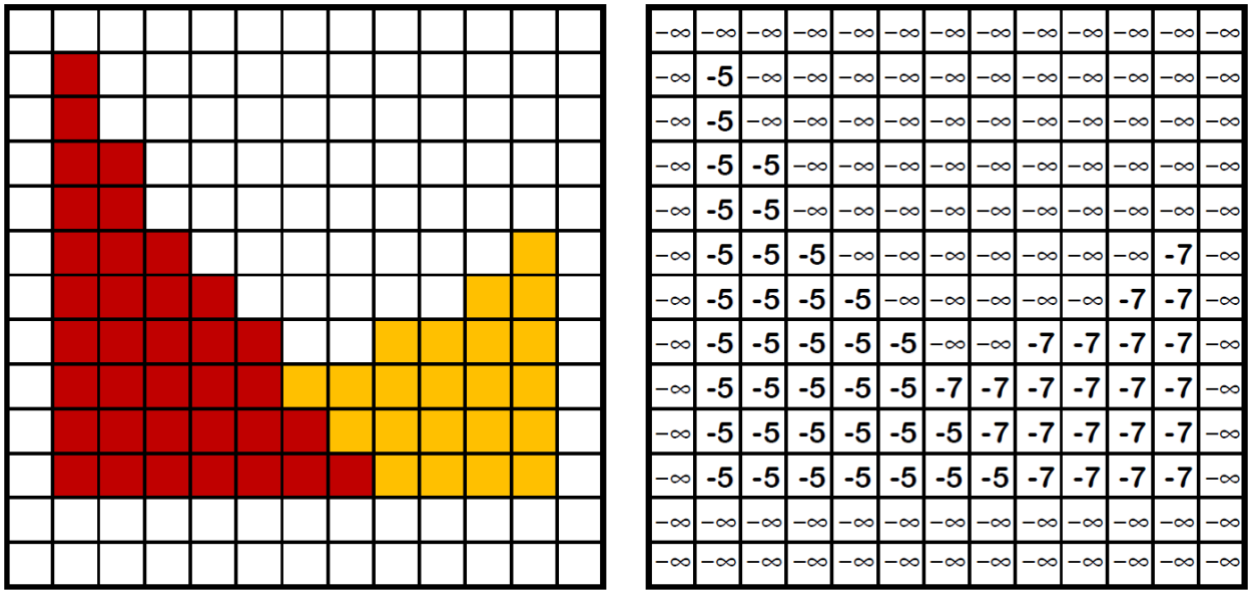

Figure 3 - Frame-buffer and z-buffer for drawing a red triangle with $z = -5$ and a yellow triangle with $z = -7$

The second processed primitive only updates the frame-buffer on the pixel positions for which the candidate depth is closer to the center of projection than the existing depth. The other positions remain unchanged.

3 Coloring the bunny based on depth

In order to obtain a better visual understanding of the importance of hidden surface removal, we will use a depth-based coloring scheme on the bunny from Laboratory work 9. The algorithm will compute each vertex' color based on its original Z coordinate (in object space) using the formula:

$$Color_{RGB} = (depthCoeff, depthCoeff, depthCoeff),$$

where

$$depthCoeff = 255 * (1 - \frac{vertex_z - min_z}{max_z - min_z})$$

$Vertex_z$ is the current vertex' Z coordinate, max_z and min_z are the maximum and the minimum Z values among all of the bunny's vertices.

The result should resemble the image in Figure 4.

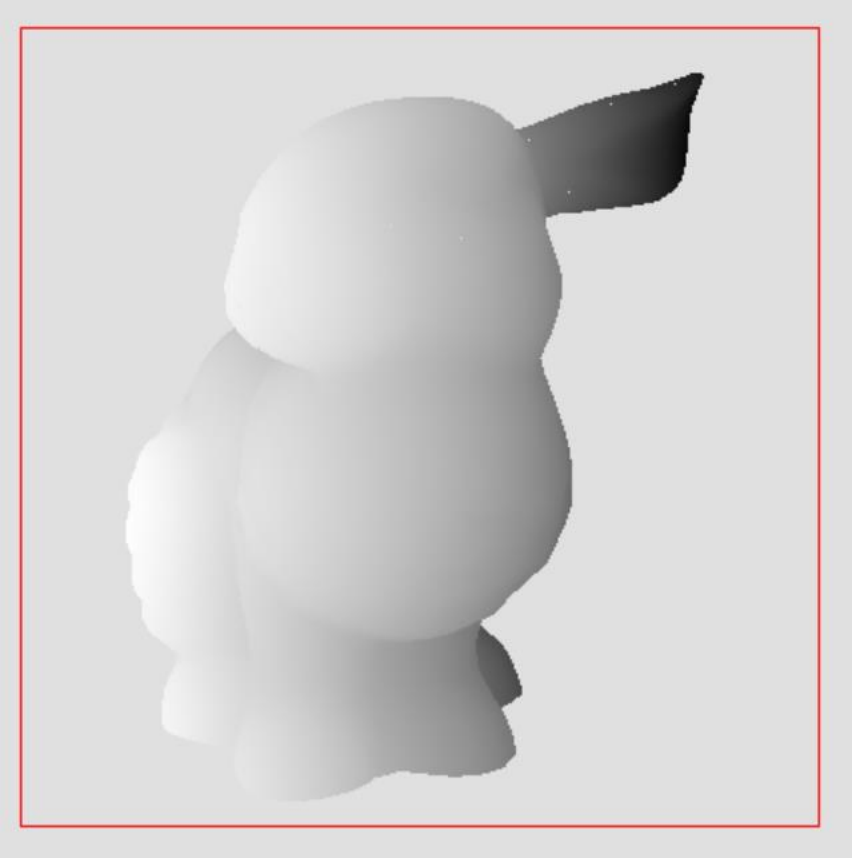

Figure 4 - Depth-based coloring

4 Assignment

- Apply the depth-based coloring scheme to the rasterized bunny from Laboratory work 10.
- Extend the triangle rasterization function from Laboratory work 9 to include a z-buffer hidden surface removal algorithm.
- Interactively switch between rasterizing with / without z-buffer, using the keyboard. To see the full impact of the algorithm, make sure to turn off Back Face Culling.