Computer Graphics **2024**

Students: Davide Frova, Jamila Oubenali

Assignment 2

---

1. Exercises done

We have done all the exercises. We implemented tone mapping with the power function.

2. Issues encountered

We believe that our end result is correct, but if you need it, we documented the problems we encountered beforehand.

Knowing which coordinate system to use At first, we struggled to situate ourselves within the two coordinate systems, so we followed the following guideline: Keep `Hit` values in global coordinates. It made sense to us, as the object's information should be stored in global coordinates. Furthermore, we decided always to compare distances in global coordinates. For example, when checking if a ray hit the base or the side of the cone first. Notably, according to our final tests and our understanding of linear transformations, as long as we stay coherent, comparing distances in local coordinates should work, too.

Forgetting the ray origin in $\gamma(t) = \mathbf{o} + \mathbf{v}t$ We wanted to implement the cone's base intersection with our existing `Plane` class, as recommended. We forgot to add the ray origin to the computation of the plane's intersect. The mistake in offset was invisible when looking at the simple planes, however it resulted in our cone's base not appearing, "flying away" or distort when we performed transformations on our cone. We traced back the problem by printing the intersection points between the rays and our plane in local coordinates.

3. Implementation choices

Clarity over performance We are aware that perhaps our code could be more efficient by refactoring, as there might be some useless "back and forth" transformations between the two coordinate systems.
Since we do not know the tasks for our next assignments, we have decided to leave it as is and spend our time playing with the tone mapping and material constants instead.
We will refactor the code if it makes sense based on what the next assignments will be.

Computation of the normal vector to the cone's surface Our first idea was that we would obtain the normal by rotating 180° around the y-axis. We did not realize immediately that this simply seemingly resulted in the negation of the y coordinate.

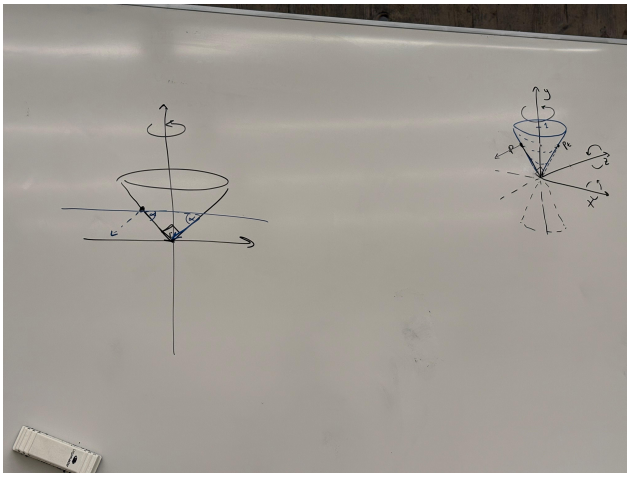

Figure 1: Our "intuition" drawing for the 180° rotation around the y-axis.

Therefore, we tried computing the normal with our rotation, but it failed. Our result was wrong, and we thought it was due to our normal. We hadn't realized yet that we had not included the ray origin in important computations.

We asked a friend who studied physics (not at USI) how he would compute the normal vector to a cone. He explained that if f was differentiable, its gradient at a point would either be zero or the normal to the surface. In this case, it indeed gave the normal vector.
We've explained our situation to the professor and asked if we could use the gradient to compute the normal, and he said that it was fine.
Let $\mathbf{e_x}$, $\mathbf{e_y}$, and $\mathbf{e_z}$ be the canonical basis for $\mathbb{R}^3$. According to the gradient's definition, we have that

$$\nabla f = \frac{\partial f}{\partial x}\mathbf{e_z} + \frac{\partial f}{\partial y}\mathbf{e_y} + \frac{\partial f}{\partial z}\mathbf{e_z}$$

Using the cone's equation $f(x, y, z) = x^2 - y^2 + z^2 = 0$, this yields

$$\nabla f(x, y, z) = 2x - 2y + 2z$$

As we normalize the normal vector after that, we see that the coefficients do not matter in the end, only their sign. We then realized that our previous guess about simply negating the y coordinate was probably valid, too.

Values of β and α We chose an arbitrary value for α, since only β is constrained (to be less than 1).