# Individual group project

# Tutorial

## In this tutorial I’ll try to explain to you: How to implement a chromatic dispersion to a shader?

First you will need to have a scene with a skybox (it will be the easiest way to demonstrate the effect produced later). This effect is applied to an object, find any model that you want to use on the Internet (I’m personally using Stanford’s bunny).

Assuming that you know how to load and initialise shader files, and how to draw geometry using VBOs under the OpenGL core profile, it should be easy for you to jump directly to shader files.

But first, don’t forget to set a float in the main that will contain what I will call “dispersionSize”. This is not mandatory but if you want to be able to control this easily from the main that’s the best solution.

## What is inside this tutorial:

In this tutorial I tried to make it easier possible. These shaders will contain only the minimum to demonstrate the chromatic dispersion without using hard stuff.

### Vertex Shader:

In this shader make sure that you are inserting the most important things in common with almost all existing shaders: modelView / modelMatrix, lightPosition, projection and obviously the cameraPosition. Then make sure to multiply each vertex position by the Model View Projection Matrix. Now that you have your WorldNorm and your WorldView you can calculate V, L, N and D the distance between vertices and the light. To calculate N, take the rotation part of the modelView matrix to generate the normalMatrix and then calculate the surface normal through eye coordinates.

You can stop here for the vertex shader (don’t forget to calculate the gl\_Position) to simplify I’ll do all the stuff related to the chromatic dispersion in the fragment shader.

### Fragment Shader:

For the fragment shader you will have to pass some uniforms like the textureUnit set to 1. PLEASE, don’t forget to bind the textures in the main that you want your object to reflect or to see through. Using a **glUniform1f** give to the shader the “dispersionSize” and don’t forget to set the cameraPos outside the Init to it will update every time you are moving in the scene. Now, calculate all the color like if that was just a reflection shader. But also separate each component of the light by applying a refract with for the three of them: the WorldView, the WorldNorm and the famous dispersionSize. For the dispersion size it needs to be different for each component so it will clearly appear on the model color (I’m multiplying by 2 for the Green and by 3 for the Blue). Then create a vec4 that will contain the chromatic dispersion and the Environment mapping together. I’m also mixing the classic reflect and the refracted colors using the mix function.

Then just output the combined color that will contain the textures and the chromatic dispersion.

### Supposed result or similar:

