Chroma Kilter: Experimenting with Chromatic Aberration in OpenGL

Team 08: John Alberse, James Dizikes, Jordan Crawford

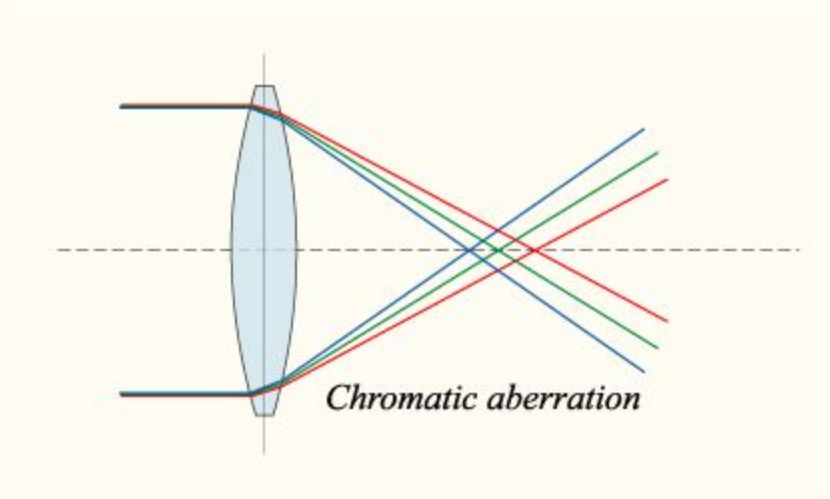
Introduction

The use of chromatic aberration in animation was recently explored by the critically acclaimed film *Spider-Man: Into the Spider-Verse*. In the film, the effect is used with great success as a surrogate for depth-of-field. The use of chromatic aberration over traditional blurring techniques is unique in its ability to express a 3D render in the visual language of 2D comic books. Inspired by the film's success, Chroma Kilter is an environment to explore the aesthetic uses of chromatic aberration in rendered scenes.

An example of chromatic aberration used in the film *Spider-Man: Into the Spider-Verse.*

An example of chromatic aberration as a substitute for traditional depth-of-field effects

Chromatic aberration is a lense effect caused by the difference in refractive index for different wavelengths of light. Similar to how a prism ‘splits’ white light into a rainbow, lenses, like those found in eyes or cameras, refract incoming light at slightly different incidence angles. This results in a subtly different focus point for, e.g. red light and blue light, as shown in figure 1. While depth of field effects result in the blurring of out-of-focus objects, chromatic aberration causes the amount of blur to depend on color. Generally, this is appears as ‘fringing’ around an object, where blue and red ghost copies of sharp edges appear subtly offset from the main edge, as seen in *Spider-Man: Into the Spider-Verse* scene above. Critically, the visual impact of chromatic aberration increases the more out of focus an object is - this lets designers use chromatic aberration to enhance or replace traditional gaussian blur as depth of field effects.

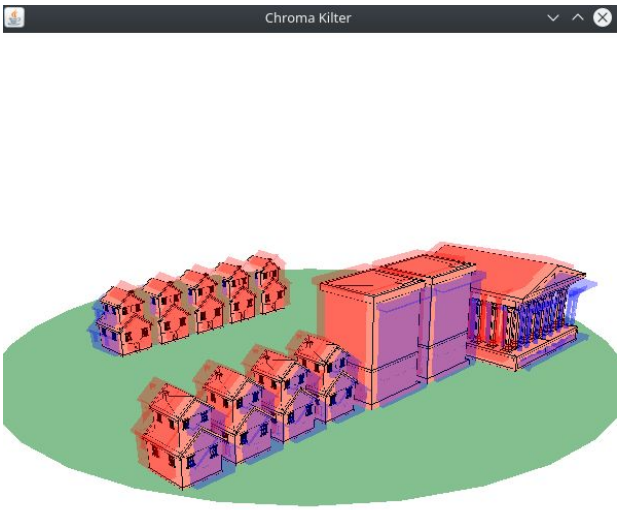


Chroma Kilter is an environment, made with OpenGL, in which models can be viewed with chromatic aberration effects applied to them. The program allows the user to explore how scenes are rendered using a number of different “render modes,” each of which take a unique approach to generating chromatic aberration. The modes are dubbed “non-destructive,” “basic,” and “no-depth.” These modes are discussed in depth in the results section. Chroma Kilter would be used by a graphics programmer or other animation industry professional who is considering adding chromatic aberration functionality to their production pipeline or render engine. Rather than by modifying the scene via one of the three render modes specified above, Jordan additionally explored an option to create the same effect in GLSL, which, while limiting in some key ways, both allows the effect to better fit into modern OpenGL pipelines and offered better performance characteristics.

None of the render modes came away as a clear winner - different situations seem to call for different approaches. For example, objects in the foreground (such as characters) would be best served by the “non-destructive” render mode. This mode preserves the local colors of the model, and so gives artists the most control over the most important objects in a scene. However, for distant objects, the “no-depth” mode provides the strongest replication of a depth-of-field effect. This may be because it is the render mode which causes the most visual noise.

Example

Using Chroma Kilter is a relatively simple process once you get the hang of it; as soon as you open up the application, you're greeted with the basic implementation of chromatic aberration that we've added: it's a rotating cube using the 'basic' mode. Immediately, you get a feel for what the application can be used for as you get a sense of how Chroma Kilter renders aberration. By pressing the z, x, or c keys, you can change the render mode to any of 'basic', 'non-destructive', and 'no-depth,' each of which shows a new different way that we have implemented chromatic aberration for different users, as is described in the results section.



An example of the effect in a multi-object scene.

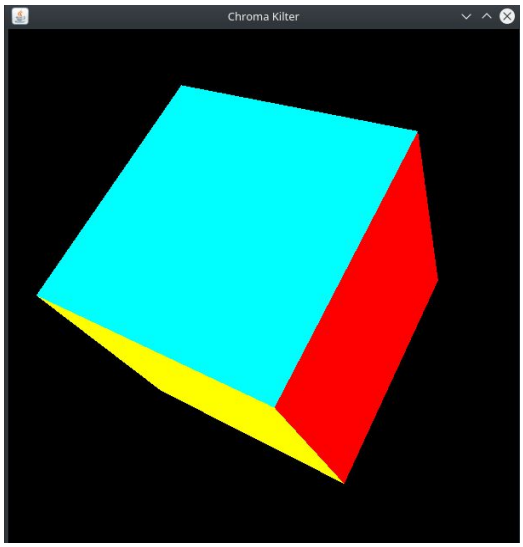
Of course, however, the way that users want to render chromatic aberration will depend on many things that aren't captured by a simple rotating square on a black background. To support these other use cases, Chroma Kilter supports a variety of modifications to the scene to explore how each rendering mode behaves under different conditions. As an example, a user can press a number key (currently, 0-7 are supported) to view a different scene, until they find one that is similar to the scene they want. Once they've settled on a object, they can modify the background color (- and =) and object color (_ and +). To adjust their view, they can pause the object rotation (p) and move the camera closer or further from the object (k,l).

Now, the user has a fully modeled and colored scene with a view setup exactly as they wish, which means they can start modifying how the chromatic aberration appears. Chroma Kilter supports a variety of parameters that can be changed here, depending on what the user wants. In particular, the user can increase/decrease both displacement (up,down) and the transparency (shift+up, shift+down) of the 'ghost' replicas used to imitate chromatic aberration.

They can also rotate the axis around which aberration occurs. Especially when the scene is paused, this rotation can have a substantial impact on how visible the aberration is. After tweaking these parameters, the user will have not only a very good sense of not only whether or not they want to use aberration in their visualization pipeline, but a deep understanding of how exactly they want to render that aberration.

Process

In the first stage of our project, we created a demonstration environment to render a few simple geometric primitives in three dimensions. Basic controls were added to this program to allow the user to change and rotate the shapes and vary their distance from the camera.



A cube rotating in an early version of our demonstration environment.

Using this environment, we then experimented with different ways of rendering chromatic aberration. Our experiments produced two main versions of the effect. The first, which we ultimately chose to pursue, simulates chromatic aberration by making two transparent copies of an object and translating them in opposite directions along a common axis. These copies can be rendered in a number of ways, which produces the three rendering modes described in the following section.

The second version of the effect draws each polygon in the scene three times, once for each of the RGB color channels. Each polygon color is offset in a different direction from the center of the object, and the polygons are then drawn with a blending function that adds the channels together. This produces an effect analogous to misaligned color printing. Unfortunately, we were unable to make this effect work consistently with geometric primitives in different orientations, so it was not included in the final product.

We originally intended to finalize the characteristics of the effect by the end of March. However, there were a few aspects of the effect that required us to prolong development beyond our original timeline. The first was the destructive nature of the basic aberrations, which change the color of the object wherever they overlap. This issue was addressed by adding

multiple rendering modes to the program. The second was the orientation of the aberration axis, which was originally fixed to each object. This allowed the aberrations to rotate in front of and behind an object. This was addressed by fixing the aberration axis perpendicular to the screen.

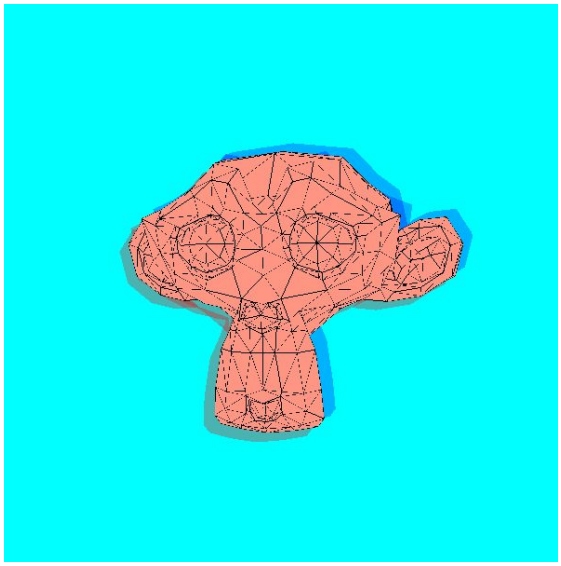
At the outset of the project, we planned to pursue two general approaches for producing chromatic aberration: manipulating geometry and manipulating light. The latter approach was eventually abandoned for two reasons. First, we ultimately decided it was more important to produce an aesthetically pleasing, stylized effect, rather than a realistic effect. Second, we simply lacked the time to investigate both approaches. James originally planned to produce an effect using a ray tracing algorithm, but this proved to be too ambitious within the time constraints of the semester.

Results

Chroma Kilter is a Java application which uses OpenGL and follows a MVC architecture. Gradle and Dr. Weaver's scripts were used as build tools while Git was used for version control. Users must have JRE 8.0 installed and an ability to follow documentation in the README.

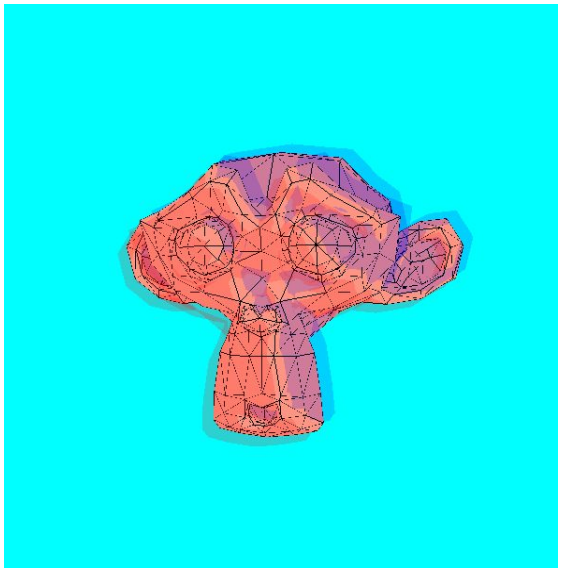
Chroma Kilter allows users to rotate models at varying speeds, move the models closer or further away, view 7 different scenes, toggle wireframes, change the background color, increase or decrease the strength of the chromatic aberration effect, and change the angle of the aberration axis, which is fixed perpendicular to the screen.

These functionalities serve to better experiment with the three render modes - "non-destructive," "basic," and "no-depth." All three render modes operate on the same basic principle. The object is drawn, and then two transparent "copies" of the object, one red and one blue, are drawn offset from the base object. Each render mode modifies OpenGL's state before drawing an object, between drawing the base object and the aberrations, and after drawing the object. How they modify the state at each of these stages is what produces the end results.



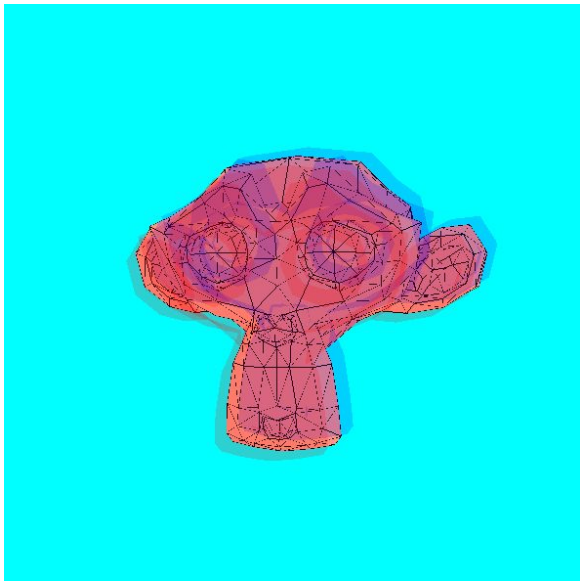
An example of the non-destructive render mode.

The “non-destructive” render mode is named as such because it preserves the local colors of the object being drawn. The chromatic aberration effects only appear on the periphery of the object. This render mode is ideal for situations where artists want to have finer control over the appearance of foreground objects such as characters. The effect is accomplished using the stencil buffer. As the base object is drawn, it increments the stencil buffer - the chromatic aberrations are then drawn only where the stencil buffer has not been incremented.



An example of the basic render mode.

The second render mode is simply dubbed “basic,” as there are no additional effects applied. The aberrations are simply drawn as any other object would be. Even still, this mode sometimes produces a more desirable effect than the others.



An example of the no-depth render mode.

The third mode, “no-depth,” is named as such because the aberrations are drawn without checking for depth. This mode is particularly noisy, and is not recommended for objects where maintaining visual clarity is important. However, this makes it perfect for rendering distant, out of focus objects.

Future Directions

The most glaring omission from Chroma Kilter is that objects, aside from primitives, are rendered in monocolor - without textures, shading, or lighting effects. Adding these are the highest priority as they would significantly impact the efficacy of chromatic aberrations.

After these are added, it would be best if the chromatic aberrations were not simply locked to pure red or blue, but instead were pulled directly from the RGB channels of each vertex on the original object.

Currently, the non-destructive render mode clears the stencil buffer before and after each object is rendered. This means the stencil buffer can not be used for other effects, such as reflections. This could potentially be a major limitation of the approach. With a more prudent approach to managing the stencil buffer, it would be possible to achieve non-destructive chromatic aberrations while preserving the buffer for other uses as well.

Finally, it would nice to be able to control the strength of the effect as a function of distance from a given plane. This would facilitate the use of the effect as an alternative to depth-of-field. Objects in the “focal plane” would have no aberrations, while objects out of the plane would have increasingly large aberrations as their distance from the plane grows.

Conclusion

Chroma Kilter is an efficient, powerful, and usable tool that allows designers to explore the worlds they create through a new dimension of depth and color with chromatic aberration. Since chromatic aberration was used to produce critically-acclaimed effects in *Spider-Man: Into the Spider-Verse*, designers have wanted to adapt it into their own work. This tool gives users exactly that ability, while supporting the flexibility and fine-grained control they need to render scenes precisely how they want to. By combining a large range of customization options with three distinct but related rendering techniques, Chroma Kilter delivers a better exploration experience to designers that results in higher quality end designs. While we determined that each of the rendering techniques produces substantially different output, we also suggest that each can be useful depending on the effect the user wants to attain.

Team Summary

John Alberse was responsible for: program architecture, foundation of all three render modes, drawing objects, user interaction (chromatic alpha, distance, rotation, model switching, wireframe toggle, render mode toggle), reading in .obj files, creating scenes, finding and cleaning models for use, and writing the progress report.

Jordan Crawford was responsible for: his graduate section of the work, enforcing clean development practices, thoroughly reviewing code. As the Mac user in the group, he also helped ensure cross-platform portability. He additionally helped compile this final report.

James Dizikes was responsible for: experimenting with different versions of the aberration effect, including transparency and distance-dependent characteristics, and writing the code to fix the aberrations to an axis perpendicular to the screen, which the user can rotate.