Project Candle Flame in EDAN35 High Performance Computer Graphics

Arvid Nilsson* Karl-Johan Arklid†

Lund University
Sweden

Abstract

This paper will describe the creation stages of a lit candle using RenderChimp[1] and OpenGL, the difficulties that were encountered, how they were solved and finally the result. A brief discussion about the flaws in implementation and possible fixes and optimization will also be included in this report.

1 Introduction

The idea of creating a lit candle was brought up early in the course with some small ideas of how it should look. The goal was to create a good looking 3D-model with as many of the real life features of a lit candle as possible while still keeping it simple.

- Transparency around wick
- Gradient coloring scheme
- Living flame animation
- Light dependent candle

2 3D Modeling

Modeling was done in four stages; creation of object, placement of object, displacement and amination of object and finally coloring of the object. Even though the stages are described in order each object went through all stages before a new object was added to the SceneGraph[2].

2.1 Creation

In this stage the flame and the aura was created using tessellation of an ellipsoid and a sphere respectively. This was done using the tessellation algorithm from the previous course EDA221[3] with some minor adjustments to allow a different value for the y-radius. The candlestick and wick was imported as *.obj* files due to their complex nature.

2.2 Placement

The geometry was added to the SceneGraph and moved manually with help from the built-in functions

```
Command::run("/grid 1");
Command::run("/axis_origin 1");
```

until a satisfying placement was found.

2.3 Displacement and animation

This stage was done inside of the vertex shader for each object (Except the wick). Displacement of the flame was done along the x- and z-axis dependent on the height of the ellipsoid, this meant that the flame was able to get a thick bottom while remaining small at the top - see figure 2.
Animation of the objects was done using sine-functions, the time[4] and the height. This allowed for a simple continuous animation that looks smooth. There were a lot of experimentation in this stage to get the right parameters for the animation of a $living\ flame$.
Even though most animation was done inside of the vertex shader the coloring of the candlestick had to be done inside of the fragment shader.

2.4 Coloring

To get a real-life feel to the flame coloring had to be done after some specific guide lines - the features meantioned in section 1. First off the $transparency\ around\ the\ wick$ was added, this was obtained using the distances from two points in space to the vertex to be colored. There were two distances to remove a sharp edge that was first created when using only one point. The $gradient\ coloring\ scheme$ of the flame was achieved by having an ambient orange color, and a distance dependant color to create the white middle part of the flame.
The aura was colored in the same way as the transparency of the flame with a static white ambient color. The wick was colored with our $gradient\ coloring\ scheme$.
At first the candlestick had a simple phong shading which was later modified to create a $light\ dependant\ candle$. By passing on the height[5]from the vertex shader we were able to modify the candle lighting to follow the aura and flame animation. With a bit of luck and after some time finding the right parameters the coloring looked like one particular effect called $subsurface\ scattering$. This is not the correct way to achieve this effect but it works simply because of the shape of the object and the height being a good estimation of the depth within the object.

3 Results

The result of this project is a good looking candle simulation which one can look at to get a relaxing feeling of warmth. Even though one might have had a different picture in the head at the start of the project, the final product looks nice and many of the properties that we wanted were achieved.
The candle could be used inside a game if wanted to but some changes would have to be done for this to work[6]. In figure 1 one can see a screenshot, taken from a distance, of the candle. As seen the aura is almost gone while in figure 2 the aura is more prominent, this might not be the real life behaivor of light but it added a nice touch to the final product. In figure 3 the screenshot is taken to get a more close up look at the fake $subsurface\ scattering$.

*e-mail: ada09an2@student.lu.se
†e-mail: ada10kar@student.lu.se

[1] http://fileadmin.cs.lth.se/cs/Education/EDAN35/RenderChimp/Doc/

[2] Renderchimps SceneGraph. The objects were created in order: flame, aura, wick and candle.

[3] http://cs.lth.se/english/course/eda221-computer-graphics/

[4] Platform::getFrameTime()

[5] which was more "the height + sine-function + small values to get the right color

[6] See section 4.2

Figure 1: A screenshot of the entire candle.

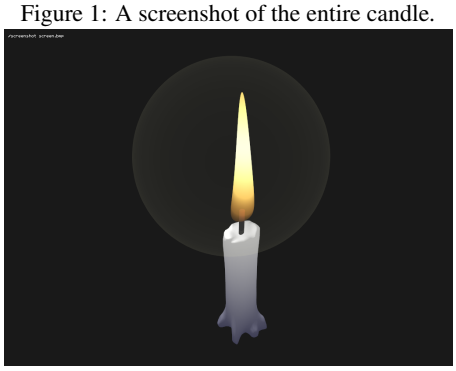

Figure 2: A close-up of the flame.

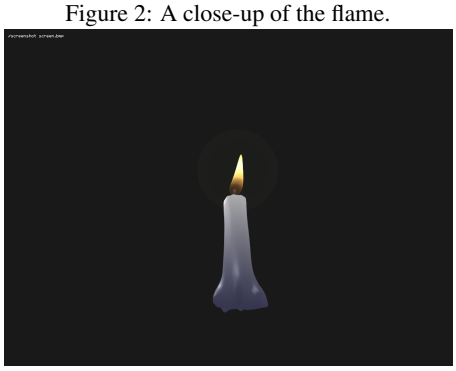

Figure 3: A picture showing the candle from underneath.

4 Discussion

4.1 Encountered difficulties

For the most part, the project went smoothly. However there were a few problems on the way. A few were due to the project being built nearly entirely from square one.

The first was an issue concerning culling. The pulsating sphere, meant to simulate an aura, covered the top part of the candlestick in an inappropriate way. This was not fixed but compensated for by using another light moving alongside the candlestick. Later this was removed to allow for another coloring of the candlestick.

Importing the Geometry objects that made out the candlestick and the wick was another problem until we realized that the objects had been exported from the 3D modeling program incorrectly.

The last obstacle was implementing anti-aliasing from scratch, using help given by lecturer and course contact Michael Dogget. After several failed attempts assistance from PhD Magnus Andersson was provided to enable the hardware anti-aliasing[7].

[7]Further discussed in section 4.2

Apart from these difficulties a few minor inconveniences concerning the aestetics were also encountered. Choosing the most appropriate colors for the respective parts took some time as well as picking parameters for the manipulated ellipsoid. Even though some difficulties were encountered, a solution was always available either by hard work until the goal was reached or from hard work until realization that help was needed.

4.2 Code flaws, fixes and optimization

With every finished product (one would hope it did not) comes flaws, possible fixes and optimization. This project also has its flaws, and possible fixes and optimization has been thought of, though not implemented. This was mostly due to the fact that most of the time was spent on optimizing the look and not the code, which gave us many hard-coded variables. With more time the points which decides the animation and coloring into specific points deciding the light and warmth midpoints[8]. Deciding the different colors as specific colors and send them into the shaders instead of coding them inside the shader would also have been a lot better. The main reason for things being hard-coded was the fact that one could recompile the shaders when running the program, which brought on a work method consisting of code-recompile-code-recompile until the objects looked close to finished and then moving on to the next object.

Another thing was the shaders doing all the sine-function calculations which on some places weren't needed to be recalculated i.e. sin(time) in all shaders. This could easily be fixed by instead of sending in the time to the shaders the value from the sine-function could be sent.

To be able to move the candle into other projects or into a game these hard-coded variables would have to be found and dealt with.

Using hardware anti-aliasing is never preferred, instead a self-written anti-aliasing that has been optimized for this project should have been used. Unfortunately the aliasing was first encountered in the late stages which caused alot of problems. The first approach was to try to enable to hardware AA inside the $CandleFlame.cpp$. This did not work, next step was to try to reform the project to work like assignment 2[9], using buffers. At this stage many problems were encountered, this was due to having four different shaders and displacement in one of them. After a long time with much help from Michael Dogget, the anti-aliasing was still a big problem for us. Fortunately Michael directed us to Magnus Andersson who helped us with enabeling the hardware anti-aliasing.

Even though there might be optimization and fixes for the project the final product looks good.

5 Conclusion

This project was overall an entertaining and interesting experience which provided a good opportunity to further study the subjects in the course.

References

[10] Lecturer and course contact, PhD Michael Dogget
PhD Magnus Andersson

[8]The light and warmth mid points would control from where and how the light and warmth would contribute to the color and animation respectively.

[9]http://cs.lth.se/english/course/edan35-high-performance-computer-graphics/assignment-2/

[10]Due to the fact that we built everything from scratch and without any guide-lines on how to do it this section is dedicated to people who helped us the questions he had.