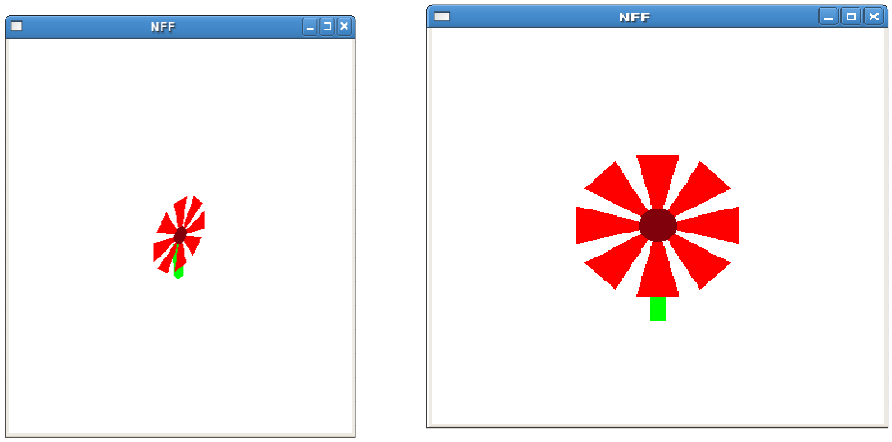
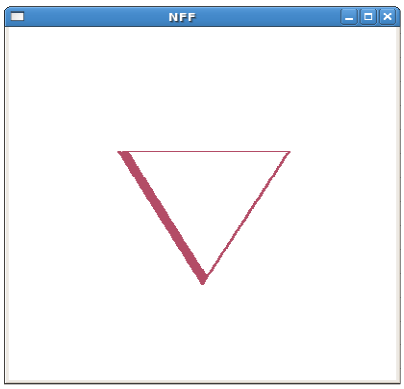
I will include everything in the pdf and this file. I included this file too because the other pdf file is missing many pictures and more details of the methods.

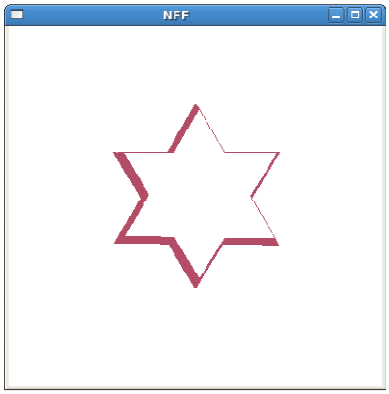
Environment : OIT Lab Machine / Redhat Linux / Kdeveloper tool.

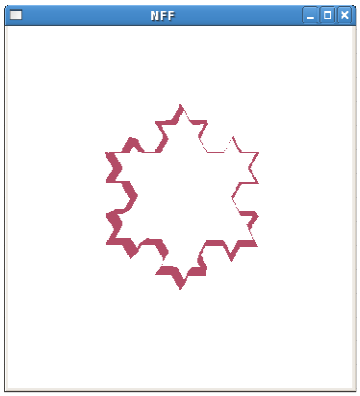
Daisy Flower: Colors may vary from the level Project.

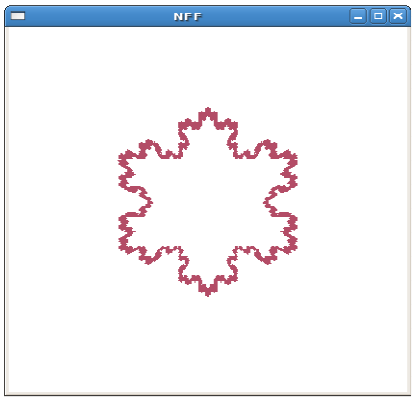


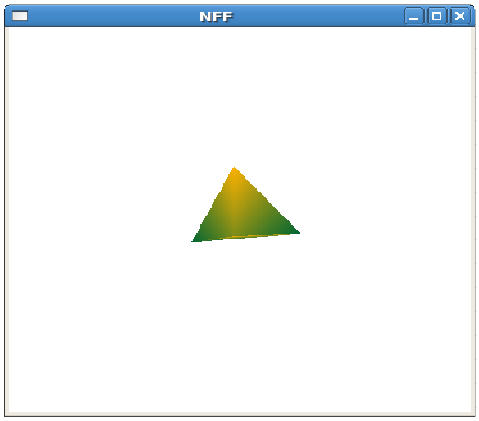
Snowflakes :
Depending on the level, the shapes vary. Here are the pictures from level 0 , 1 , 2, 3, 5.







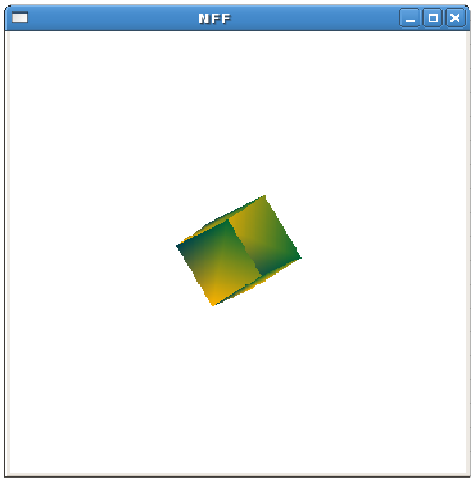




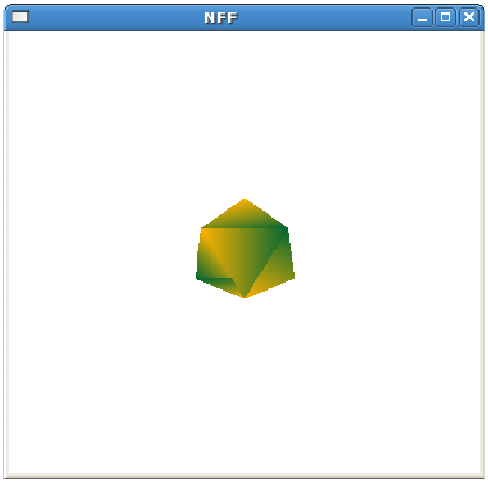
Since this is just repetitive, I only included a few examples.

Platonic Solid : Depending on the parameter(1-5), the results follow:
1

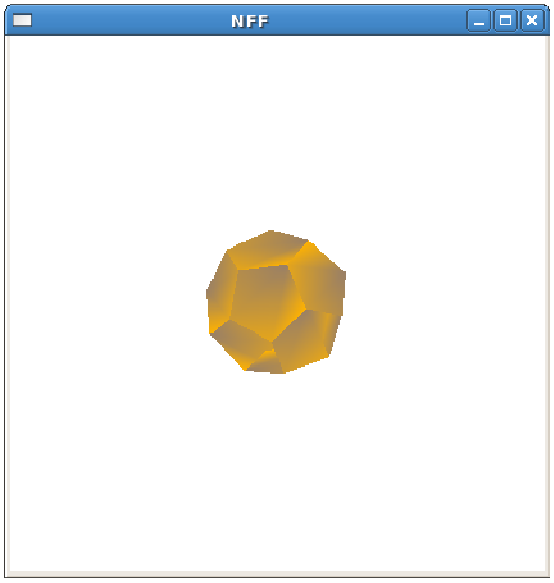
2



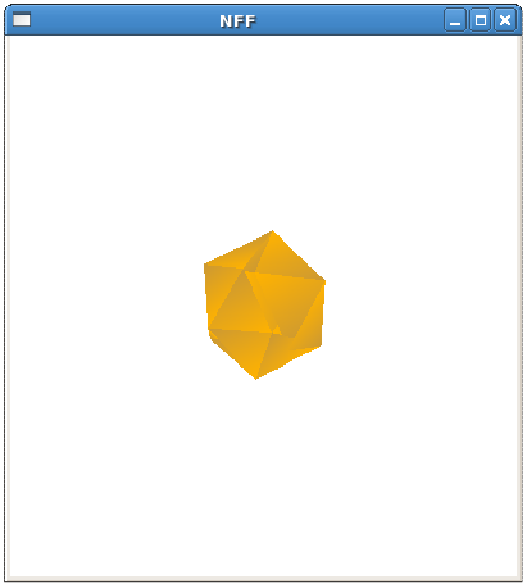
3



4



5

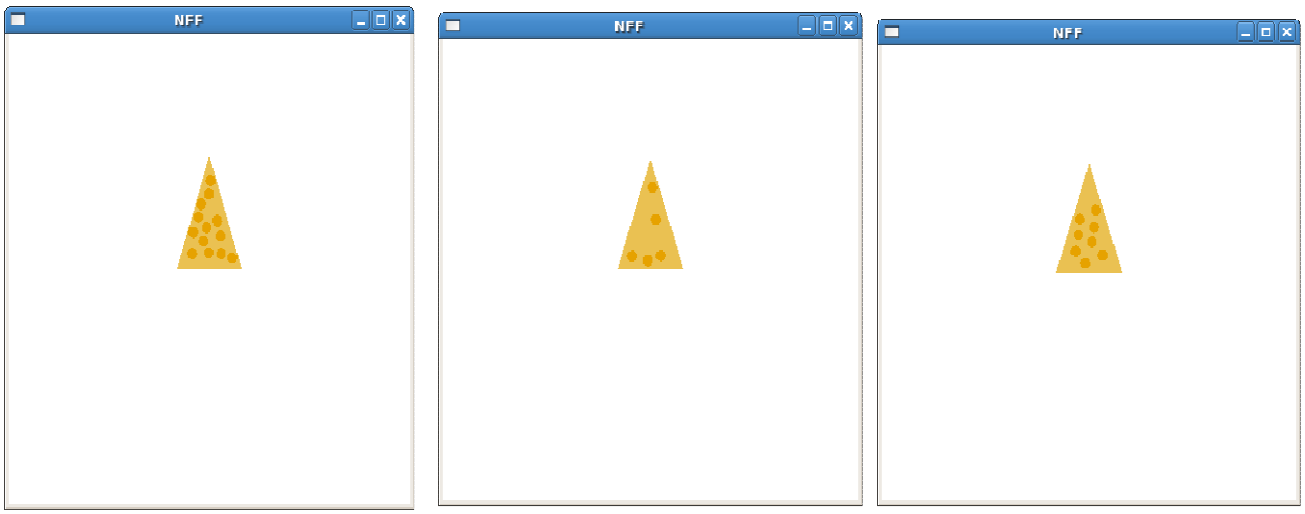


Like I explain in the other pdf file, the theory is really simple. Like openGL draws a sphere, it draws in the same way. It basically calculates each polygon that is crossed on the sphere surface. All we need to do is to divide by the number of the divisions. X , y values can be computed by applying sin and cos theta values. Once we get it, we only need to connect the right vertices.

Swisscheese :
depending on the parameter, it makes the number of holes. They never collide.

Pictures with different numbers of holes.

Respectively, 13 holes / 5 holes / 8 holes.



It generates random positions of holes automatically. When any holes collide with others, it generates a new position. (They check the radius and distance between the circles to see if they collide.)
Also, I used a technique to determine if the generated point is in the triangle to make sure the holes are nicely located on the cheese. Holes are made of 360 triangle polygons. It just looks like a polygon, but it is just 360 polygons together looking like a hole.

Basically, in the whole project, I never used any glu not glut function to draw objects.
Everything consists of just polygons, but nothing else.
(Even for porcupines, they are all polygons although they may look like cylinders.)
I spent a huge amount of time on making porcupine objects because it was very difficult to generate polygons that form a needle. The needles even bend. I generated some grammars to do it and sometimes manually modify the positions. However, it is really automatically generated. It does not require user inputs just due to the complexity of user inputs. (Radius, needle lengths, and extra porcupine head variables and translation variables.)

GameWithNFF : This is the level project.

Basically it can take a number of objects and display them. You have to specify it by parameters.
For example >

maze <number_of_objects> <filename1> <filename2> ...

However, NFF filenames must strictly be the filenames generated from each program. They never collide with walls, and they pick a random position.