COMP6761 FALL 2003

Final Project Documentation

Professor: Dr. Peter Grogono

Student: Pedro Maroun Eid, ID# 5041872.

Date of Submission: Tuesday 2 December, 2003

Table of Contents

Table of Figures

Introduction

The idea of this project came from watching a candle flame flicker on the effect of wind. The first thought was to do something similar to lighting in OpenGL, called Wind Effect. However, this turned out to be a very vague idea. The goal was to start with a simulation of wind effect on an easy object. After advice, an easy object would be grass or particularly a grass strip. If this can be done, then there would be continuity and the effect could be made to apply to harder objects like tree leaves or even trees by themselves. Moving a ball was too obvious; it's a simple motion problem and could be solved in seconds. The hard thing about wind effect on grass is to make the grass move (bend) according to the wind force you apply to it. The next part is to simulate a nice and natural effect which makes it more realistic to look at; this is defined by Turbulence.

Description

Structure

The project is divided into 4 phases. These are: the modeling of a nice grass strip, the construction of an array of strips which represents grass, the effect of wind being able to change the look of each of the grass strips and finally some environment to include all the connections between the mentioned parts.

Process

The project was done using Object Oriented Programming and it mainly uses the following libraries: OpenGL, GLUT, and CUGL. There are 3 main classes of objects in this project and these are, as mentioned above, Grass Strip, Grass field, and Wind. Grass field uses Grass Strip to create a static double dimension array of grass strips and controls them accordingly. Grass Strip only needs one parameter to draw the strip at the required position. Wind is set to a particular speed and displacement interval then when applied should return a parameter of the new position of the object it is hitting. CUGL is only used for texturing.

Implementation

A nice grass strip

After that the idea has been set, the project was started by experimenting with 2D Bezier curves. Bezier curves give nicely modeled shapes with a few parameters. The experimentation was initiated on the Java applet available at: http://www.cs.berkeley.edu/~j-yen/splines/beziercurve/. It was found that actually with 5 vertexes on a 2D plane and a high level of slices, one can model quite a nice grass strip and that by moving only the top point, the whole curve would move in a correct manner. The constraint is that the top point that is to be moving should be bound; it can only move inside an interval. Taking the bottom of the strip as an origin to a semi-circle and the distance from the origin to the top point to be the radius, the top point should always be inside this semi-circle with the specified radius and the grass strip would look just fine. Therefore, to move this strip, we would only have to move this top or third point defined by the Bezier curve. The idea was found and now it was time to implement it in 3D. To start learning, a piece of code from CUGL was extracted which was done using 3D Bezier surfaces. This is the fuselage of the plane model. A lot of tune-ups and trials were required to give a nice curve using the same idea of the 2D curve experiment. That is, 5 points and the 3rd point is the control of the whole grass strip. This is an output of one of the first version of the modeled curve. It can be moved by mouse left and right while button is down. It can be remarked that this is a skull object made with black lines and that it is very detailed so that, even when it is curved, the strip would look nice and smooth on its edges. Remember that this is a 3D object with an oval base which means

that it looks flat when looking at it from a particular perspective. This is done using 5 control points for every slice and using 30 slices and 30 stacks.

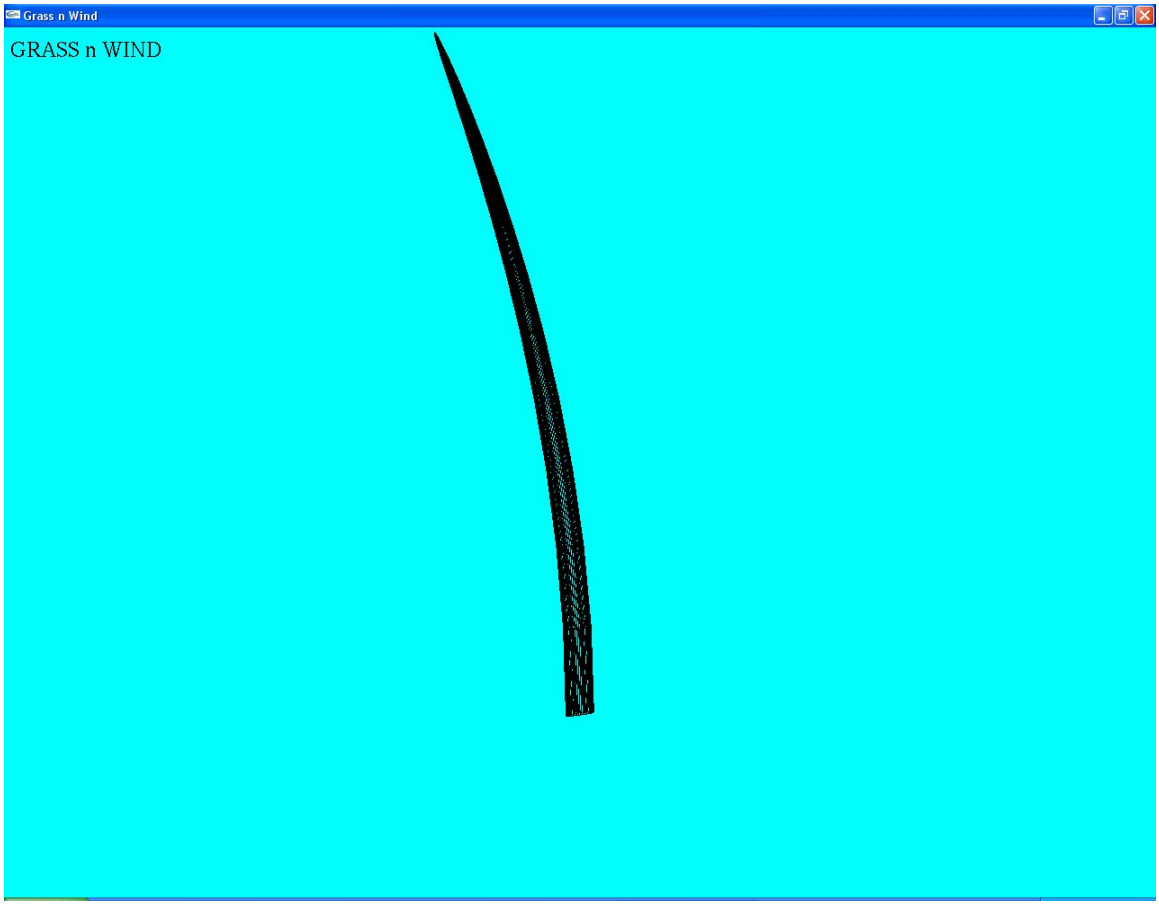

Figure 1: Grass Strip

An array of strips

All strips when created look the same, but the control which is Grass field sets different sizes to each grass strip in the array according to a maximum size given when setting the grass field object. Moreover, a different color is created for each strip in this array. There are 5 chosen main colors and these are all taken from the background texture of the scene so that it looks somehow realistic. The 2D array tessellates the grass

strips according to the number of grass strips given when initializing. However, if this number is not a square of a certain whole number, the program only uses its whole part. This is done by the square root since each side of the square should be the square root of the given number of grass strips. For each grass strip, a size and a color are specified and then it is drawn with a certain force value uniform on the entire grass field. The program would give a force value to the field and the field, by itself, would draw the entire objects that it handles accordingly with this same force.

The effect of wind

Effect of wind is rather messy and has a lot of undiscovered threads. My ignorance about wind led me to read some books about meteorology, wind and atmosphere physics, etc… A lot of high level concepts would have made the matter more complicated if wind was to be implemented as it was described in the big books. The question was: would I have to implement concepts like geostophic winds that create gradient winds which in turn have turbulent movement when friction existed with the ground it is moving above. The formulas were too complicated to be understood clearly and a lot of concepts are still indefinite since there exists more than 6 kinds of implementations available for turbulence effect and some depend on others. It is clear that the science of atmosphere isn't mature enough yet. A colleague of mine, rendering smoke and including wind effect to move the smoke particles suggested her solution. The idea was rather simple to start with and the addition of turbulence would be added later if possible. Consider the equation of motion, given by Newton, which can compute the time

given a velocity and a distance. These are the speed and displacement interval which should be given when setting the wind object. The time to travel on this distance would be obtained and then a time interval chosen to be 0.001 of the whole time needed to travel would be used to get positions along the path until the required time is reached. The force given to grass objects would be a set of positions on that displacement interval computed using the equation $P_n = P_{n-1} + v*t$, where P_{n-1} is saved from the last value computed by the function and 't' increases by 0.001 on each iteration until t is reached. After a lot of testing and tuning-up, the animation of the strip worked perfectly with different configurations of wind speeds and displacement intervals. The wind speed is selected to be between 0 and 30 miles per hour which is a 100 km/hr and the displacement interval to be [-20, 20]. 't = d/v' won't work well if 'd' is big, so while setting the wind, the speed is multiplied by 100 (which is the same as dividing the distance by 100) to give a reasonable amount for the movement of the grass strip.

Now the grass strip was moving smoothly and it was time to figure out how to imitate turbulence effects because wind on grass is characterized by the many blows you can see affecting the grass. To go back to the big formulas was rather hard though more research was going on to discover simpler ways, but in vain. Turbulence is the effect of many blows with giving the chance of the grass strips to be resistant; this means to let them restore their original positions. To easy up this concept, I assumed that these blows start at position 0 and end at the position given by the displacement. Then, the only challenge would be to make the strips go back to the original position they started from after reaching the end of their path. This would be easily implemented using the same

displacement and the opposite of the time needed to surf this displacement (t = -t) along with making t0 decremented by 0.001 (-0.001). This would make the grass go back smoothly where it came from. This was the easiest way to create this effect and after testing and tuning-up again, it looked rather nice. The point position is made to be initialized to 0 each time the wind is initialized so that the counter of the position would reset and not go into an abstract position which the grass strips can't be at. The function that applies the wind effect is only given a variable that it changes by reference and this changed value is then passed to the grass object to be drawn with as described above. This makes the wind object abstract and editable while the passed value is consistent.

Environment

The environment includes a volume cylinder which is textured dynamically using a clever loop. This cylinder imitates a natural environment and should include grass. However, as a constraint of hardware speed, no more than a specific number of grass strips could be drawn according to the speeds of different CPU's. Therefore, I chose to have a function that adds and deletes grass field arrays which show the difference in CPU speeds and the good effect when 1 field is displayed. The volume is made to rotate with the mouse left and right movement while pressing the mouse button. A grass field by default is constructed using 50 strips (where only 7*7=49 will display). This is experimented to be a reasonable value before the algorithm gets heavy on the CPU. The letter 'q' is made to add a grass field to the scene, and the letter 'w' is made to hide one. Therefore pressing 'q' 2 times will result in 2 fields appearing with one of them translated to the side of the other. The letter 'x' will assign a wind speed and

displacement interval to the wind object and the idle function of the OpenGL program will be responsible for calling the wind.apply() function along with passing it a force attribute to be given a value. The function returns the referenced value of the next position of the grass object and the scene is redisplayed again with this new value. The apply() function never stops working; the grass object will always appear to be going back and forth with no end until a new set up of the wind speed and displacement is set with the letter ‘x’.

Figure 2: Scene Output

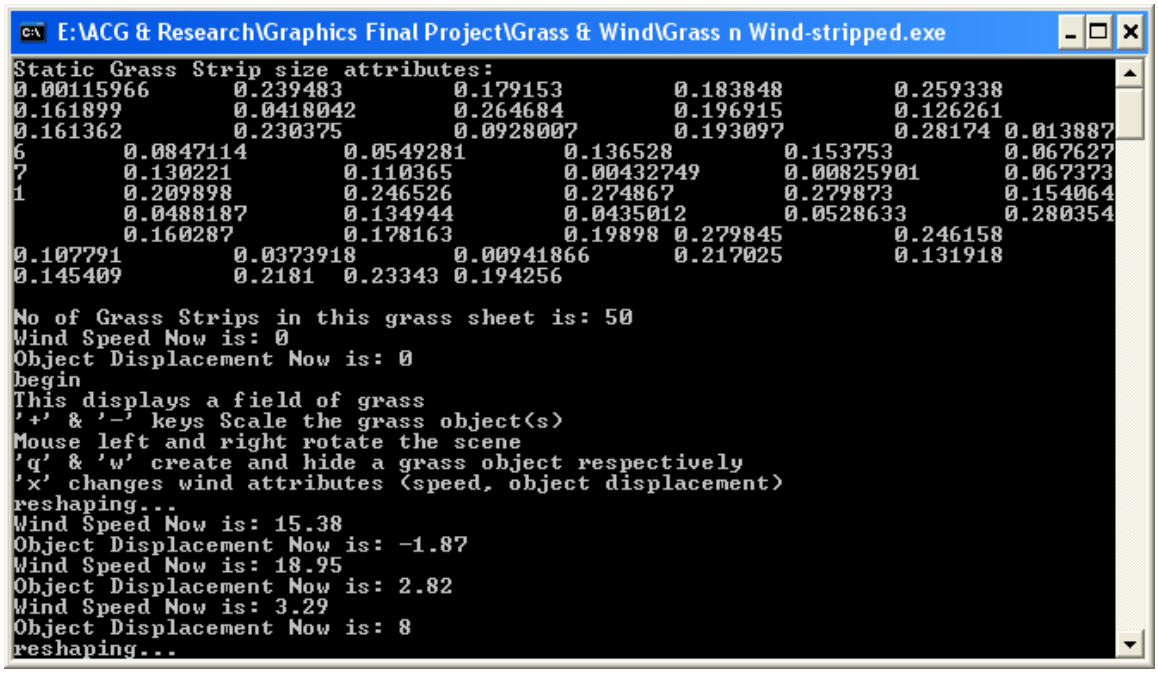

Figure 3: Program Output, info and data

Future work and improvements

The future of this project holds a lot of interesting research. It is not impossible to implement a real turbulence effect of wind on grass considering that it might be the easiest scenario you can find. The only needed clues are clear and exact formulas to do so. This is dependent on either the advances in meteorology and wind physics research or finding true approximation formulas to do so. I will continue working on that project myself since the subject fascinates me. Also a good improvement of speed would be to change the Bezier surface into a set of dependent polygons which are faster to render than Bezier surfaces. Natural effects to make the scene look more realistic would include sunlight shadows and fog. All these could be done if enough time can be invested on this project. It was a very pleasant experience.