Middle East Technical University Department of Computer Engineering

CENG 469

Computer Graphics II

Spring 2023-2024
Assignment 3 (v1.0)
Grass Rendering in OpenGL (Geometry Shader, Bezier Curves, Perlin Noise)

Due date: June 2, 2024, Sunday, 23:59

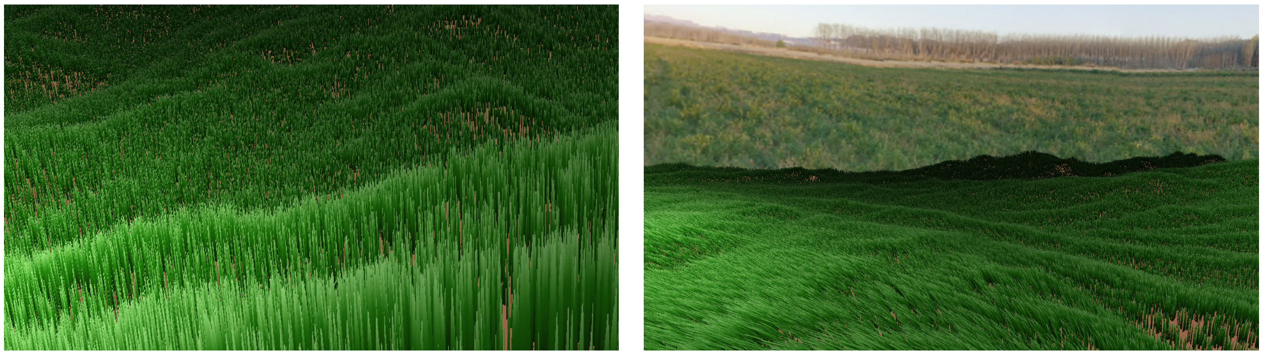

Figure 1: Some example visuals of grass in non-windy and windy weather, respectively.

1 Objectives

In this assignment, you are going to implement a field of grass in a windy weather using OpenGL's geometry shader. Additionally, wind calculations will be sampled from Perlin noise and be used to move each grass blade so that the field of grass will sway in harmony. The expected end product of this assignment is an OpenGL program which renders a scene that includes the items listed below:

- A flat-ground field composed of grass blades of some color and various heights, where each blade is calculated in the geometry shader as a Bezier curve that start at the bottom of the blade and end at the tip and whose control points animate smoothly and also vibrates and sways according to the speed of the wind at that location at each frame.

- The Perlin noise implementation, which will be sampled by each grass blade via its 3D position on the ground to calculate the wind speed at the location of that blade so that when some neighboring blades sample the Perlin noise to bend using that wind amount, a smoothly changing wind value will get returned by the noise algorithm to each of them, creating harmonious bends. The Perlin noise look-up calculation can be offset in a direction by some increasing amount at each frame so that each grass in-place would have that smoothly changing wind effect during runtime.
- Keyboard buttons, to change the color of the grass field (E & R), to increase/decrease the wind's speed (A & D), and to increase/decrease the height of the grass (W & S) where longer grass would be expected to have more bending due to their height and the wind's existence.

Some sample visuals taken from a different implementation are provided in Figure 1. We do not provide exact screenshots or a YouTube video as we did in the previous two homework assignments, but the sample visuals from the different implementation. In this text, we have provided you the homework's idea roughly, and expect you to have freedom during the implementation for some optional details such as having a skybox in the scene, or instead of a flat ground, creating a ground with some height whose height could be sampled from a Perlin noise calculation, or the visual of the wind's bending effect on the grass, or having a camera movement setup, etc. We also expect you to contact us for every detail that you might need further clarification.

2 Regulations

1. **Programming Language:** C/C++ is highly recommended. You also must use gcc/g++ for the compiler if you use those programming languages. Any C++ version can be used as long as the code works on Inek machines. If you use any other programming language, be sure that the Inek machines can compile and run your code successfully and also include a simple Readme file to describe how we should run your code on Inek machines during grading.
2. **Additional Libraries:** GLM, GLEW, and GLFW are typical libraries that you will need. You should not need to use any other library. But if you still want to use some other library, please first ask about it in the ODTUClass forum of the homework.
3. **Groups:** All assignments are to be done individually.
4. **Submission:** Submissions will be done via ODTUClass. You should submit your blog post link to Blog Links forum on ODTUClass, so that the URLs are publicly available for everyone's access, using the title "HW3 Blog Post". For code submission, create a **".zip"** file named **"hw3.zip"** that contains all your source code files, shader files, mesh files, any other content your program needs, and a Makefile. The executable should be named as **"main"** and should be able to be run using the following commands (any error in these steps may cause a grade deduction):

   ```
   unzip hw3.zip -d hw3
   cd hw3
   make
   ./main
   ```

5. **Blog Post:** You should write a blog post that shows some output visuals from your work, and explains the difficulties you experienced, and interesting design choices you made during the implementation phase. You can also write about anything you want to showcase or discuss, including an **FPS analysis** if your scene had different amounts of grass blades.

   The blog website you write your blog on should have the ability to automatically show the last edit date for your post which shouldn't be able to be edited by the writer.

   The deadline for the blog post is 3-days later than the original deadline for the homework. You can submit your code before the homework deadline, and finalize the blog post during the following 3 days without losing late days. You consume your late days only if you submit your "code" late. However, submitting the blog post more than 3 days after the original homework deadline will incur grade deductions.

   If you would like to put some piece of code or a link to the online repository of your homework implementation into your blog post, please do so by editing your post 3 days after the homework's code deadline, so that your code is not visible to the students who use 3 late days for the coding part. The rest of the post should already be published before the blog post deadline, which is 3 days after the original homework deadline. We will check the blog pages the moment the blog post deadline ends.

6. **Late Submission:** You can submit your codes up to 3 days late. Each late day will be deduced from the total 7 credits for the homeworks of the semester. However, if you fail to submit even after 3 days, you will get 0 regardless of how many late credits you may have left. If you submit late and still get zero, you cannot claim back your late days. You must e-mail the assistant if you want your submission not to be evaluated (and therefore preserve your late day credits).

7. **Cheating: We have zero tolerance policy for cheating**. People involved in cheating will be punished according to the university regulations and will get 0 from the homework. You can discuss algorithmic choices, but sharing code between students is strictly forbidden. Please be aware that there are "very advanced tools" that detect if two codes are similar.

8. **Forum:** Any updates/corrections and discussions regarding the homework will be on ODTUClass. You should check it on a daily basis. You can ask your homework related questions on the forum of the homework on ODTUClass.

9. **Grading:** Your codes will be evaluated on Inek machines. We will not use automated grading, but evaluate your outputs visually. Note that you should test your code on an Inek machine before submission, as the frame rate might not be as high as your home computer if you have a powerful PC, and might cause hangs when launched. This might require reducing/optimizing per frame calculations in the code. Blog posts will be graded by focusing on the quality of the content.