Project 4
CSCI-4962: Three-Dimensional Computer Graphics

Due: Monday, December 2, 2002, 11:59:59pm

1 Overview

In this project, you are to write a 3D flight simulator program for a plane flying over a terrain environment you create. Your project has two components: generating a realistic looking terrain and simulating the view from the plane as it moves.

Please see the course web page at `www.cs.rpi.edu/~sakella/graphics/` for project updates and additional information. Also check the course WebCT bulletin board for relevant posts.

2 The Flight Simulator Task

You are to write an OpenGL program to create a simple flight simulator. The displayed scene is the pilot's view as the plane moves. Your project consists of the following tasks:

1. Create an environment that includes mountains, water, and man-made structures (for example, buildings). The environment is defined over a rectangular region where the XZ coordinate plane corresponds to sea level height, with the positive X axis pointing north and the positive Z axis pointing east. You should color and/or texture map objects in the scene for greater realism. For example, you could have snow capped mountain peaks, plains with vegetation, or an ocean region. The region of water may be simulated by a blue colored (or texture mapped) polygon.

2. To create mountainous terrain, use the recursive midpoint subdivision method to generate fractal terrain over a rectangular region. The color of the terrain should vary with height to achieve a more natural look (for example, higher elevations of mountains can be colored white to represent snow). Your midpoint subdivision algorithm should have a parameter that specifies the maximum height variation, so the height variation decreases with increasing levels of the recursion. You can use the `rand()` function to generate random numbers.

3. You must also create an adjoining rectangular region of the environment with man-made structures (for example, a field with a rotating windmill, or a runway, or buildings).

4. Create a simple model of the plane's cockpit interior, so the displayed scene is the pilot's view of the environment from the cockpit.

5. Provide the plane with motion capabilities. The initial position of the plane is specified by its (x, y, z) world coordinates. Its initial orientation is specified by its $(\theta_x, \theta_y, \theta_z)$ angles.

These three angles specify sequential rotations of the plane about the X_w, Y_w, and Z_w world coordinate axes respectively. There is an orthogonal local coordinate frame attached to the plane such that the X_p axis points to the pilot's right, the Y_p axis points up from the pilot's seat, and the Z_p axis points to the rear of the plane. Then *roll* corresponds to a rotation about the plane's Z_p axis, *yaw* corresponds to a rotation about the plane's Y_p axis, and *pitch* corresponds to a rotation about the plane's X_p axis. The user should be able to control the position and the orientation of the plane as specified by the keyboard and mouse functions below.

- `f` key: The plane speed is set to zero and the plane translates forward (in the direction of the negative Z_p axis) by a constant amount Δf (specified by you).
- `w` key: The plane speed is increased by a constant amount Δv (specified by you) from its current speed.
- `s` key: The plane speed is decreased by the constant amount Δv from its current speed.
- `a/d` keys: The plane rotates (rolls) 10 degrees left/right respectively about its Z_p axis with each key press.
- Left/right arrow keys: The nose of the plane rotates (yaws) 10 degrees left/right about its Y_p axis with each key press.

  Additionally, with the left mouse button pressed, a leftward horizontal motion of the mouse should rotate (yaw) the nose of the plane left, and a rightward horizontal motion of the mouse should rotate the nose of the plane right. Use the `glutMouseFunc()` and `glutMotionFunc()` functions for this. Scale the mouse motion so that the yaw angle changes by 90 degrees across the width of the screen.
- Up/down arrow keys: The nose of the plane rotates (pitches) 10 degrees up/down respectively about its X_p axis with each key press.

  Additionally, with the left mouse button pressed, an upward motion of the mouse should rotate (pitch) the nose of the plane up, and a downward motion of the mouse should rotate the nose of the plane down. Scale the mouse motion so that the pitch angle changes by 90 degrees across the height of the screen.
- `p/P` key: Pauses and resumes the motion of the plane, assuming it has a non-zero speed.
- `g/G` key: Changes the window to full screen mode by using the `glutFullScreen()` command, and returns to normal size. The default mode is a normal size window.
- `i` key: The plane is returned to its initial configuration. Note that the initial configuration of the plane should be above nearby terrain features, and its initial speed should be zero.
- `q` key: The program quits.

6. Use the `glutTimerFunc()` callback to better specify the motion of your plane when it flies at a specified velocity.

7. Your README file must describe the complete flight simulator functionality, including plane initial configuration, environment modeling information, and especially any extensions.

8. Here are some possible extensions to your project:

   - You can create multiple types of terrains to let the user select what type of terrain to fly through.
   - To create the illusion of infinite terrain, you can specify how the plane's position gets updated when the plane reaches the environment boundary. You may wish to tile the region surrounding your environment rectangle so the terrain is always visible.
   - Create an animation of the scene as the plane flies over the environment. The flyby should include changes in both the position and orientation of the plane. A click of the right mouse button should start and stop the animation. For example, you may wish to animate a barrel roll.
   - Model two or more planes flying in formation, such that the user can select the current plane (and viewpoint) by cycling through the planes in sequence by pressing the `c` key. Input keyboard/mouse events modify the state of the current plane.

3 Grading

Your project will be graded for a total of 100 points as follows:

- 40 points for creating scene of environment and cockpit.
- 40 points for translational and rotational motion control for plane.
- 10 points for the realism and creativity of your scene design.
- 10 points for code structure, clarity, and documentation.

Extra credit: You can earn up to 10 additional points for special features and creative enhancements to the project requirements. Describe your enhancements in your README file, and ensure that they do not conflict with the required project functionality.

Lateness policy: Please read the lateness policy in the course syllabus. Use your late days carefully, if at all.

4 Submission

The code must be submitted no later than **11:59:59 pm on Monday, December 2, 2002**. You will follow the same submission procedure as for Project 1. Specific instructions for handing in your code are provided on the course web page. Remember that you will need to use your CS login ID and password for project submission. **You are responsible for ensuring that your code can compile and run on the PCs in VCC North or South, or the Sun Ultras in Amos Eaton 217.** Your source code must be readable and commented. The comments need not be extremely long, just explain clearly the purpose of each block of code.

You must submit a single zip file containing your README file, any texture files your program uses (along with the code to read them in to your program), your source code (source and header

files), and if necessary, a Makefile to compile it. Your submission should NOT include any object files or executable files. Every file you create (except the texture files) should have your name in a comment line at the top. The README file should contain the following information: your name, instructions on how to compile the code and run it, known bugs or limitations, any extra credit enhancements, and any other relevant information.

Proper submission is entirely **your responsibility**. Contact the TA if you have any doubts whatsoever about your submission. Do **NOT** submit your project via email. Be sure to keep copies of your files in a protected directory of your RCS or CS account and **do not change them after submitting**. After grades are posted, you have exactly one week to resolve all problems. All grades are final two weeks after they are posted.

A project that does not follow the submission guidelines will receive a **10 point deduction**. If you are unable to submit your project zip file according to the above guidelines, be sure to place it in a protected RCS or CS directory and send email to the TA with the location of the file.