Project I: Flight Simulator

CS 440 Computer Graphics
Habib University
Fall 2021

Due: 2359h on Sunday, 7 November

In this project we will implement a flight simulator. This is a complicated task so the questions below break it down step by step.

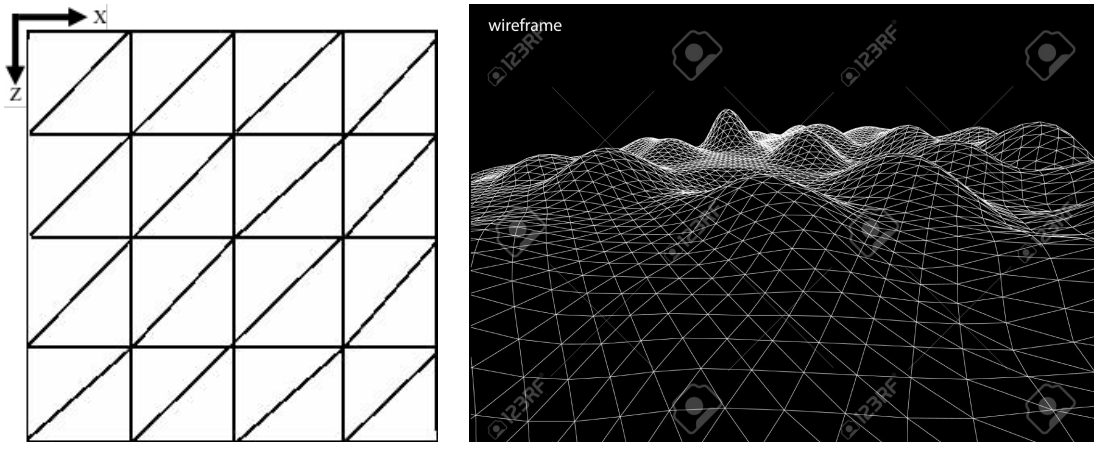
High level instructions are provided and you are expected to use your knowledge of math, computer science, and computer graphics to make reasonable implementation decisions and put the individual pieces together. Make use of the discussion avenues specified in this project's README as needed.

Even though both are equivalent, the expectation for this project is to achieve the illusion of flight by varying camera parameters, not by transforming geometry.

The simulator should run in a page titled `simulator.html`.

1. Plain Terrain

A *height map* is commonly used to generate a *terrain mesh*. A height map is a function that takes in 2 values, e.g. longitude and latitude, and generates a height value, h. Write a function to generate a height field over a given area in the xz- plane. The function would generate a grid in the given area, triangulate it, and randomly perturb the y coordinate of the grid points within $(-2, 2)$.



(a) (b)

Figure 1: (a) A triangulated grid in the xz- plane. (from Toymaker) (b) View of an example height field generated by perturbing the y- coordinate of the grid points. (from 123RF)

Implementation Tips

- Write a function `get_patch(xmin, xmax, zmin, zmax)` to generate a terrain patch in an appropriate format and return it.

2. Flight

Implement a flyby view of your generated terrain. This is the perspective view obtained from a plane

flying at a certain altitude (e.g. $y = 3$) above your terrain and facing straight ahead, i.e. parallel to ground level. Implement mechanisms to dynamically effect the following.

- alter the bounds of the view, i.e. *left*, *right*, *top*, *bottom*, *near*, and *far* of the viewing volume within reasonable limits. Use the following key map.
  - `1` and `Shift+1` to vary *left*.
  - `2` and `Shift+2` to vary *right*.
  - `3` and `Shift+3` to vary *top*.
  - `4` and `Shift+4` to vary *bottom*.
  - `5` and `Shift+5` to vary *near*.
  - `6` and `Shift+6` to vary *far*.
- quit the simulator. Map the `Escape` key to quit..

Implementation Tips

- Modifying a value beyond its allowed constraints should have no effect.

3. Varied Terrain

Specify a certain height ($y = 0$) in your terrain as *ground level*. Your terrain contains greenery at ground level and the points are colored green. Terrain below ground level is covered by water. Only the surface of the water is visible which is colored blue. Terrain above ground level is mountainous and the points are colored green to brown to white as the height increases. Implement mechanisms to dynamically effect the following.

- toggle the view of the terrain in the circular order: its points, its wireframe, and its faces. Map the key `V` to the toggling of the view.
- toggle the shading in the circular order: flat, smooth, and Phong. Map the key `C` to the toggling of the coloring scheme.

Implementation Tips

- The mapping from altitude to color could use the `map_point` function from previous assignments.
- The points in the *points* view must be large enough to be visible.
- To flat shade a polygon, apply the average color of the three vertices.

4. Some Flight Dynamics

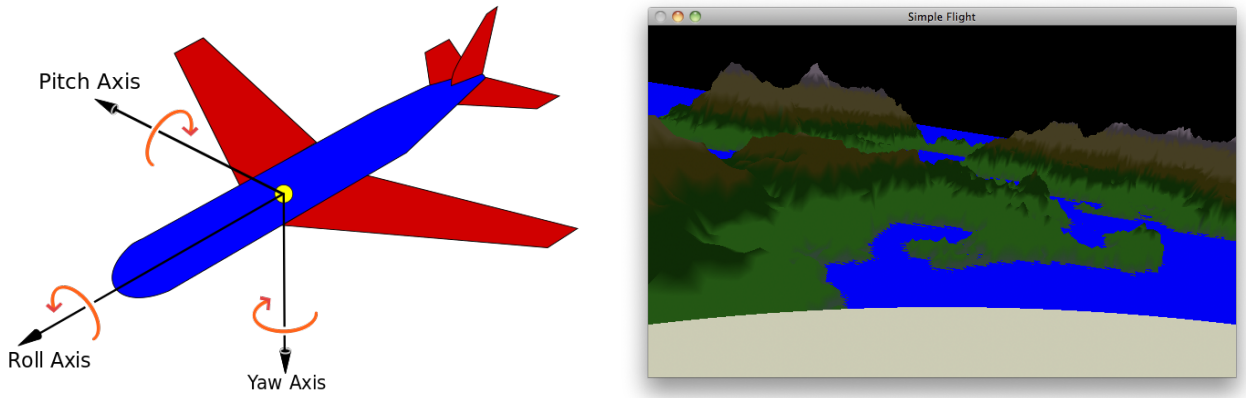
The orientation of a plane in flight is determined by its rotation about three axes. The rotations, termed *pitch*, *roll*, and *yaw*, and the corresponding axes are illustrated in Figure 2. Implement mechanisms to dynamically effect the following.

- each of the indicated rotations, constrained to $(-90°, 90°)$. Use the following key map.
  - `W, S` (and lower case) for pitch
  - `A, D` (and lower case) for yaw
  - `Q, E` (and lower case) for roll

  Constrain the rotations to $(-90°, 90°)$ and the plane's altitude as appropriate, e.g. $(2.5, 3.5)$.
- increase and decrease the speed of the plane between $[0, S]$ where S is a maximum speed of your choosing. Use the `up` and `down` arrows to vary speed.

Implementation Tips

- Modifying a value beyond its allowed constraints should have no effect.
- You will have to infer the changes caused by each rotation to the camera's parameters.
- The rotation constraints ensure that the plane will never turn so much as to travel backward.



(a) (b)

Figure 2: (a) The three rotations of a plane. (from Wikipedia) (b) An example view of the terrain. Your view need not contain the dashboard.

- The altitude constraint ensures that the plane will not run aground, nor will it fly up into space.
- The plane always sees ahead along its roll axis.
- The speed constraint allows for the plane to stand still.

5. To Infinity

As the terrain is finite, the plane will ultimately fly out of it. Modify your implementation such that a new terrain patch is dynamically computed and added when the plane approaches the boundary of the patch that it is currently flying over. This way, the plane never flies out of terrain.

Implementation Tips

- The more you fly, the more terrain patches will be added and the more your GPU's memory will be consumed, resulting in a slowdown. To prevent this, make sure to unload previous terrains.
- Make sure to add appropriate patches accounting for the plane's current flight direction.
- Ensure that the plane is never flying over empty terrain, no matter its speed or rotation.
- Instead of generating a very large terrain, load a terrain that is large enough for the current view and generate and add new terrain patches based on the plane's motion as it approaches the current boundary.
- Instead of unloading terrain, you can rewrite already allocated buffers. Making use of the fact that the flight constraints ensure that the plane will not travel backward, figure out how many patches you need to keep in memory at any given time.

6. Superpowers (Bonus)

You can include the following additional functionality for a bonus.

- The height map is not completely random but is sensitive to neighboring points, thus yielding a smoother terrain.
- The height field is read in from an image, e.g. as described here.
- Any other desired functionality.

Happy Flying!