Final Project Project Review - Giselle Koo - CSCI 4229: Computer Graphics - Fall 2022 - Willem Schreuder

Hi Professor Schreuder,
It might not look like much but I wasted a bunch of time on a gridded array type implementation before figuring out and implementing a Delaunay triangulation algorithm (which in and of itself took a few days to actually get it to make even just a two-triangle square… but i did it!)

This is not a short list (I had hoped to get more done but essentially got stuck on the first two for way too long) but I will try my best to get as far as I can. Let me know what you think is realistic!

Giselle

Project roadmap:

- [x] ~~Display an empty canvas which gets called in displayScene()~~
- [x] ~~Display a single crater in the center of the canvas (default rendering)~~
    - [x] ~~Circular, fixed size, fixed cross-section, no texture~~
- [x] ~~Implement add crater/clear canvas function on keypress~~
    - [x] ~~Limited to n = 0 / 1~~
- [ ] Display multiple craters incrementally that don't overlap on keypress
    - [ ] All identical, Linear placement, spacing based on diameter = no overlap, remembers last location
    - [ ] Test reset canvas again, have to reset last location
- [ ] Display overlapping craters
    - [ ] All identical, Linear placement, spacing based on diameter to force overlap
    - [ ] ID points of overlap: Distance between center < distance to edge in the same unit vector (?)
    - [ ] Apply transfer function to these points (delete for now, add task to fix if it looks weird)
    - [ ] Add new points to canvas
    - [ ] Test reset canvas again
- [ ] At this point putting in a graphical UI instead of just keypresses would be helpful.
    - [ ] Crater add / clear canvas buttons
    - [ ] Crater placement mode:
        - [ ] Single/center
        - [ ] Linear + no overlap
        - [ ] Linear + overlap
        - [ ] Make sure this changes crater placement algo but remembers last placement unless canvas is cleared
- [ ] Incremental addition of variables + thorough testing
    - [ ] Test procedure:
        - [ ] Hard-coded fixed from list

- [ ] Placement: Single / Linear no overlap / Linear overlap
- [ ] Combine incrementally with prior variables
- [ ] Variables:
  - [ ] Diameter
    - [ ] Cyclically pick from list [0,1]
      - [ ] D = l*(max-min) + min
    - [ ] Add min / max setting to UI where min=max=disabled
  - [ ] Eccentricity
    - [ ] min/max eccentricity
  - [ ] Eccentricity angle
    - [ ] Setting = Angle
    - [ ] Remembers last angle & next angle just added to last
  - [ ] Xsection variation
    - [ ] From a list of profiles
    - [ ] https://agupubs.onlinelibrary.wiley.com/cms/asset/9e112894-dd3f-4c88-8280-92ccf180bd2a/jgre21119-fig-0002-m.jpg
- [ ] Add randomness within range
  - [ ] Location - from finite 2-D range [xmin, xmax] [ymin, ymax]
  - [ ] Eccentricity angle - randint*angle%360
  - [ ] Diameter - from finite range [min, max]
  - [ ] Eccentricity - from finite range [min, max]
  - [ ] Xsection - just random int to pick from list instead of in order
- [ ] Make it all look good
  - [ ] Rough lighting and texture adjustments
- [ ] In no particular order:
  - [ ] Fix motion control of the sun
  - [ ] Add xsection parameterizationz; http://lroc.sese.asu.edu/posts/864
  - [ ] Fix POV functionality
  - [ ] Random noise to edges
  - [ ] Tweak lighting/textures ad nauseum