Low-Poly Generator Project Proposal

Rohan Jha - rjha@andrew.cmu.edu

November 16, 2019

1 Project Description

My 15-112 Term Project is called LowPolyGen. It generates a stylized, user-tuned, low-poly render of a given photo using Canny Edge Detection and Delaunay Triangulation algorithms. These generations will be displayed in an Adobe Bridge-style graphical recursive user file interface.

2 Competetive Analysis

First and foremost, I consider each of the following projects as inspiration, rather than competition; each takes the same base concept in different algorithmic and stylistic directions. I would identify Crystal Qian (cjqian on Github)'s polygen, Peter Maldonado (pmaldonado)'s PyTri, and Georg Fischer (snorpey)'s Triangulation webapp as the most comprehensive examples of competing projects. They span the gamut of delivery, purpose, and algorithmic approach from JavaScript web app to Python terminal call. Qian's implementation, as described in her paper (*Generating low-poly abstractions*, Qian 2016)[1] focuses primarily on subjective user opinion after an excellent mathematical portion detailing the effects of the well-chosen parameters of the algorithm (including sampling rate σ, and blur) and, being more of an academic paper, has little emphasis on the surrounding infrastructure. On the other hand, Maldonado's implementation[2] has less focus, and is a more terminal-centric application, with many optional arguments that have less pertinence to the algorithmic side of the generation. Finally, Fischer's Triangulation web app[3] demonstrates an appealing user experience, with near-real-time rendering on changed slider input, but this comes at the cost of a better refined product, specifically in how unintuitive the parameter's effects are on the output image.
In terms of dimensions of detail from other projects I'd like to pay close attention to in my own, Qian's thoughtful parameters merged with Fischer's sliders and image visualization rank very high. I do, however, forsee real-time generation based on user slider input as untenable due to the computational expense of my

[1] http://cjqian.github.io/docs/tri_iw_paper.pdf
[2] https://github.com/pmaldonado/PyTri
[3] https://snorpey.github.io/triangulation/

proposed algorithmic function. Instead, I will allow users to alter parameters with sliders, then lock them in and render with a button.

3 Structural Plan

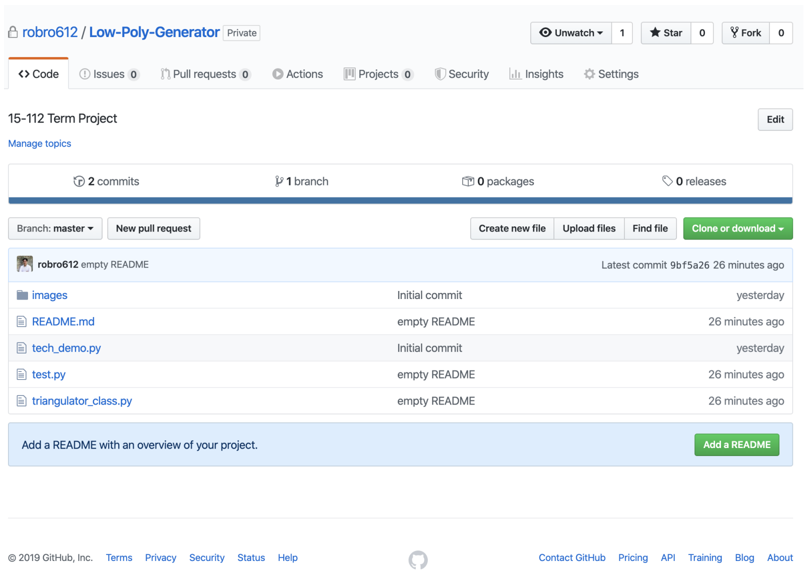
LowPolyGen will be divided into four files, one handling preprocessing and the actual Delaunay generation of the images, one small one with a low poly image class, one handling the animation of the interactive Bridge-esque file system, and one wrapper file to ultimately run the experience.

4 Timeline

1. TP1 (11/20)
   Basic low poly generation algorithm implemented with a start on the closest-neighbors quality optimization.
2. TP2 (11/26)
   Completed (optimized) closest-neighbors optimization. Completed basic Bridge-like visual file system.
3. TP3 (12/5)
   Optiouch ups and extra functionalities: slider tuning and extra file info..

5 Version Control

For version control, I have initialized a private Github repository called Low-Poly-Generator. In addition to this, all supplemental materials like this proposal document, project notes, and image assets will be stored on Overleaf, Google Drive, and Imgur respectively.


robro612 / Low-Poly-Generator Private

15-112 Term Project

Manage topics

2 commits · 1 branch · 0 packages · 0 releases

Branch: master

robro612 empty README — Latest commit `9bf5a26` 26 minutes ago

images	Initial commit	yesterday
README.md	empty README	26 minutes ago
tech_demo.py	Initial commit	yesterday
test.py	empty README	26 minutes ago
triangulator_class.py	empty README	26 minutes ago

Add a README with an overview of your project.

© 2019 GitHub, Inc.

6 Module List

I plan on using the following modules in my project:

1. numpy for its highly optimized implementation of N-dimensional arrays in which we store our image pixel data as well as its seamless integration with matplotlib's plotting and cv2's convolutional filters.

2. matplotlib for intermediate image and data visualization (currently blur, sharpen, and edge detection).

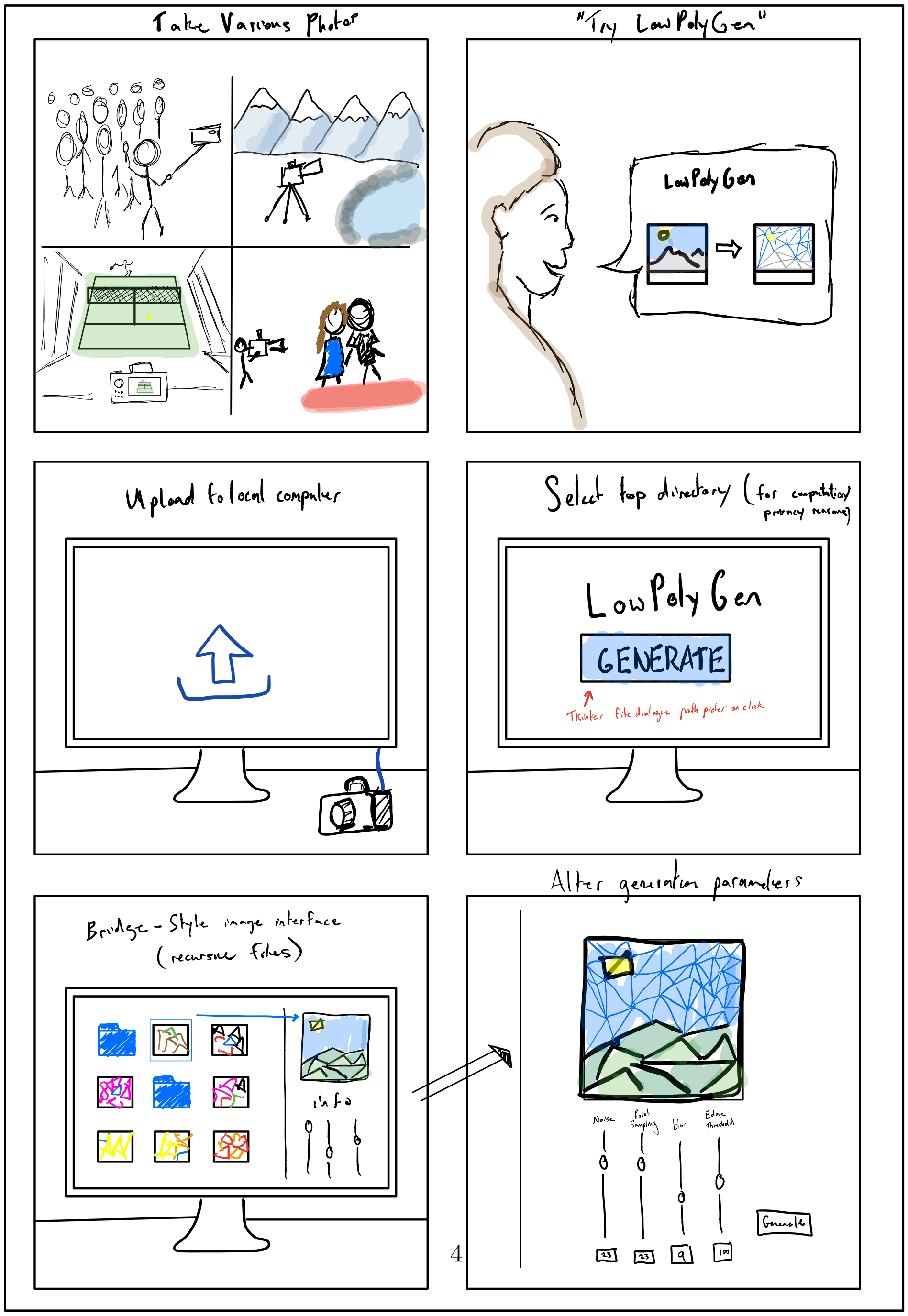
3. cv2 for its convolutional filters, including blur, Canny edge detection, and a customizable filter with a user-set kernel.

4. scipy for its implementation of Delaunay Triangulation, the algorithm that triangulates an image space Ω with triangles that have no other points in its circumcircle.

5. tkinter and cmu_112_graphics for the final rendering and animation of the stylized image and surrounding user experience.

6. random, time, and os for homemade Gaussian noise, efficiency testing, and file system interaction respectively.

7 Storyboard



Take Various Photos
"Try LowPolyGen"
LowPolyGen
Upload to local computer
Select top directory (for computation/ privacy reasons)
LowPolyGen
GENERATE
Tkinter file dialogue path picker on click
Bridge-Style image interface (recursive files)
info
Alter generation parameters
Noise
Point Sampling
blur
Edge threshold
23
23
9
100
Generate