Assignment 1: Semi-procedural, Top-down Village Generator with WebGL

Due Date: October 25, 2019, Friday

The demonstration of the assignment will be during the class hour (09:40-10:30) in B-Z05.

Every student must attend the demonstration in the class.

Grade Value: 15 %

Important Notice

This assignment will NOT be done in groups. Every individual student must do his/her homework.

The assignments will be submitted to MOSS (Measure Of Software Similarity), which is an automatic system for determining the similarity of programs. It is very successful in identifying similar codes even if you change variable names, indentation, places of functions, and so on. Any similarities between the programs of students in the class, as well as any similarities to the programs on the Internet, mandates disciplinary action.

You will demonstrate your programs to Dr. Güdükbay in the scheduled time. All the students must be physically available in the demonstrations. Demonstrations are a kind of exam. You will be asked questions to understand your contribution to the assignments in the demos.

If you need any assistance, you must request it from the TA. email: aytek.aman@cs.bilkent.edu.tr

You must also submit your source codes (including online documentation (comments and a README.txt file explaining how it is compiled and run) to the TA. You should email (with the Subject CS465 Assignment #1) to Aytek Aman (aytek.aman@cs.bilkent.edu.tr) a single zip file named as LastName_Name_CS465_Asst1.zip, containing two directories: Source, Executable, and a README.txt file describing how to compile and run your assignment.

Requirements

You will design and implement a simple semi-procedural WebGL app that will paint a very simple top-down village. In this program, there will be 5 different entities: terrain, river, houses, trees, and rocks.

- **Terrain:** Terrain should cover the whole scene.
- **River:** River should be in the middle of the screen and lay vertical. River width should be a random value between user-specified values(river_width_min, river_width_max).
- **Houses:** Houses should be represented using a roof (use slightly different colors as shown in the image) and a chimney.
- **Trees:** Trees should be represented using a circle. You should use additional circles to draw fruits.
- **Rocks:** Rocks should be represented as simple polygons.

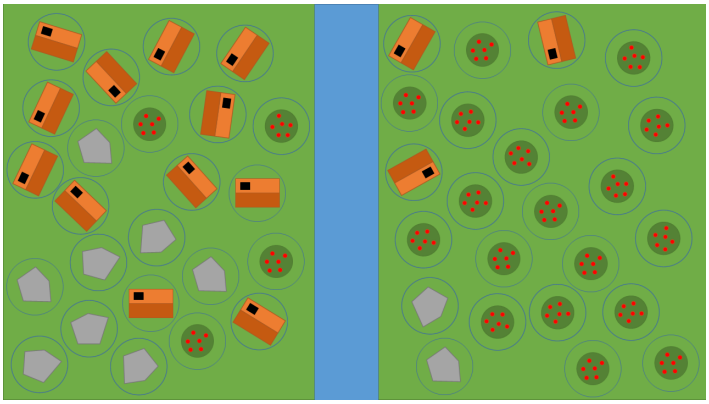
The placement of **houses**, **trees** and **rocks** will be driven by simple Poisson-Disc sampling process. To do that, assign a radius to each entity. Entities' drawable content should never overflow from the circular shape defined by its radius. Start generating 2-D coordinates on the canvas and check for previously placed entities and river for collisions/overlap. During these checks, you should only consider the circular borders of the entities. If there is an overlap, simply do not generate an entity at the given position and choose a new

random position. This process should continue until a certain number of entities are generated. This value should be user-defined.

Entities must have a random rotation to ensure variation.

Before drawing an entity, you should decide on the type of the entity. To decide, you will use 'attractors' that are placed on the canvas by the user. You should be able to create an attractor for each kind of placeable entity (houses, trees, and rocks). Once you generate a coordinate and there is no overlap on this coordinate, probabilistically decide on the type of the entity considering the distance between entity and attractors. If an entity is very close to a house attractor, then this entity should be a house with a relatively large probability. Attractors should be placed on the canvas by the user. When the user places a new attractor on a canvas, the whole village should be regenerated.

In the User Interface (UI), there should be appropriate controls for variables and attractor types, and so on. In the following figure, a sample village drawing is shown. In this figure, there are three attractors. On the top-left, there is a house attractor. On the bottom-left, there is rock attractor. On the right part of the canvas, there is a tree attractor. Borders of the entities are shown in blue for illustrative purposes. You should **not** display those in the demo (it is useful for debugging though).



You need to use suitable GL primitives for each entity (quads, tris, etc). **Don't hardcode the coordinates of the vertices.** For example, circular entities (trees) should have at least 32 segments which you can compute using trigonometric functions.. Don't be afraid to get creative! You could add random variations to entities to generate better-looking drawings. Also, additional effects & entities are welcome! Since each drawing will be random, you might want to save them if one of them looks interesting. Provide a save and load mechanism that saves the properties of the drawings (coordinates and colors of the houses, trees and so on). You should be able to load these files from disk and get the exact drawing. (Do not save the whole framebuffer pixel by pixel!). You can also save the whole drawing by just saving the random seed, attractor positions and user defined parameters. However, for this assignment, you are not allowed to do that.

Important: Always comment your code. The code will also be checked during the demos.

Grading Criteria

- Does your program meet the requirements listed above? Your programs will be graded with respect to the number of the requirements they satisfy.
- You should construct an instructive user interface for your program. The user interface should be easy to use. There should not be any menus without any information. For example, having an empty window without any information but opening a menu when a mouse button is clicked is not a good user interface. The functionality of the user interface is a part of the grading criteria.

Tips

Please make sure your program runs in preparation for the demos.

You could use simple HTML UI elements or JavaScript (jQuery, AngularJS, jQWidgets, and so on) UI widget libraries.

There are vast number of WebGL examples on the web. **(Do not use these codes directly)**. If you use some code for user interface generation or some other purpose with modifications, you must state so at the beginning of that code segment (e.g., method header) and properly state how you modified that code segment.

Also DO NOT use the third party wrappers like three.js. Use WebGL from scratch.