# MATLAB Procedural Shape Generation

Design Document (DRAFT)

By: Matt Vernon

**Introduction:**

By using the concept of procedural generation, this MATLAB program will allow the user to specify certain parameters (such as: noise-layering, amplitudes, frequencies, resolution, and size) to generate unique shapes. Procedural generation is commonly used because a computer can generate ‘organic’ and ‘natural’ looking shapes much quicker than someone could model them by hand. This process is useful for 3D simulations, animations, video games, and anywhere that computer generated shapes could be useful. Because of the heavy computation times of procedural generation and noise generation, this program will utilize MATLAB’s ‘GPU Array’ capabilities wherever possible. Initial tests have shown that matrix calculations done on the GPU are 100 to 200 times as quick as those done on the CPU, although, the program will be built to work whether the user’s computer has a GPU or not.

## **Requirements:** (Break down the problem statement in the introduction into a set of requirements)

## User Interface Requirements

## Input: (What must the user see when they run the program? How must they interact with the program? Is there error checking? How will program display output results?)

* A window with controls to change the parameters of the procedural generation
  + Slider for noise Amplitude
  + Slider for vertices resolution
  + Ability to shade the shape depending on height or local slope
  + A button to generate the shape
  + A button to save the shape to excel
  + Radio buttons for selecting whether to compute on GPU or CPU
  + A dropdown for selecting an image to map to the shape

## Output: (How will program display output results?)

* MATLAB’s 3D plot Window (visual shape)
* An [optional] export shape-vertices and C values to Excel for saving a generation

## Algorithms/Logic Requirements (What must the program do with the information gathered from the user/elsewhere?)

* Calculate vertices of a Cube sphere
* Use noise algorithms such as Perlin noise, to add detail to the shape.
* Calculate color values depending on height and local slope

## Data Requirements (How must the data be stored? Format?)

## Input Data: (This is for data being read in from a file or database)

* Input data should be GPU Arrays or normal arrays for the (x, y, z) coordinates, and c values.

## Output Data: (This is for output data being stored in a file or database)

* Users will have the ability to save generated shape’s vertices and c values to an excel file.

# Technical Summary: (At a high level, how will you solve the problem posed by the above requirements)

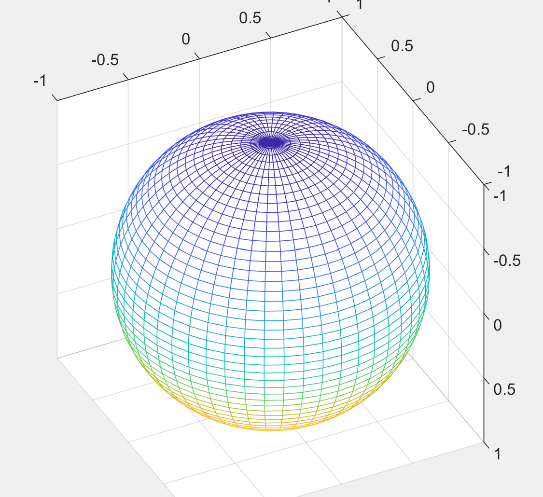
<What programming language and framework will you use?>

**MATLAB R2021a**

<What algorithms or equations will you use? Provide background information on the topic, if necessary>

**Mapping cube vertices onto a sphere (Quadrilateralized Spherical Cube):**

Since this program will use a sphere as the starting shape, it is important that we pick the correct sphere to work with. The traditional UV sphere offered in MATLAB and commonly used in computer graphics has an uneven distribution of vertices. A UV sphere has more of its vertices clumped together around the poles, and less near its equator. This would give high control (or resolution) near the poles, and low control near its equator. This can be seen in this picture:



*UV Sphere MATLAB*

By taking vertices of a cube and forcing them into a spherical shape, we can make something called a Quadrilateralized Spherical Cube, or a ‘Cube Sphere.’ A Cube Sphere is a type of sphere that has even distribution for all vertices and offers linear control for the number of vertices instead of exponential (like the Geodesic Polyhedron.) The math for mapping vertices of a cube onto a sphere can be shown by:

A picture containing surface chart

Description automatically generated A picture containing text, clock

Description automatically generated

*Source:* [*www.mathproofs.blogspot.com*](http://mathproofs.blogspot.com/2005/07/mapping-cube-to-sphere.html) *– Phil Nowell*

**Perlin Noise**

Since this program is going to be used to generate unique shapes, we must add noise to the shape. Perlin noise can help to give the shape natural and organic looking features. For example, adding Perlin noise to a flat plane can generate features that look mountains, hills, valleys, or craters. This image was computer generated using Perlin Noise.



*Picture by Simon Strandgaard from Kastrup, Danmark – Image Generated with Perlin Noise-* [*https://commons.wikimedia.org/w/index.php?curid=76348609*](https://commons.wikimedia.org/w/index.php?curid=76348609)

By adding Perlin noise across the surface of the 3D shape, we will be able to generate features so that the shape is no longer smooth. Allowing the user to add multiple layers of noise, and to change the amplitudes and frequencies of the noise, the program will be able to create a shape similar to what is desired by the user.

<What design principles will you use? At a high level, how will you implement them?>

**Modular Design:** Program should be modularized into multiple smaller components to keep the program directory easily readable. This will be done by creating separate files for functions and classes. The Main program will mostly consist of calling these other files.

# Design: (How will you translate the requirements into implementation using the technical summary)

## **Pseudocode**:

## Main program

Prompt user to enter parameters for program

* All user input will be handled through a GUI if possible

Detect if user has a GPU, default to using the GPU if one is available

* gpuDeviceCount()
* (Possibly test if GPU or CPUA has a quicker compute time)

Generate a Cube Sphere

* Create a Cube
* Convert the cube to a cube sphere

Apply noise and other parameters to the Cube Sphere

* Add a combination of Perlin noise, simplex noise, and other noise layers to the shape.
* This will likely be the most crucial part of the program and require a lot of fine tuning. It is important to properly layer noise in a way that produces a desirable shape, instead of an entanglement of vertices and lines.
* Optionally apply a texture image over the shape, this can be done by converting a texture image to a bitmap and setting c values of vertices accordingly.

Display Generated Result

* Mesh()
* The Mesh() function in MATLAB is a powerful function and allows a high level of control over various parameters of the mesh.

<List the top 5 MATLAB functions that will be critical in helping you implement your program?>

**gpuArray()** : allows matrix math to be done on the GPU instead of the CPU, this will significantly speed up the program.

**mesh()** : renders a shaded 3D mesh given x, y, z, and c values.

**Recommendations: (What would be nice to have but is out of scope because limited resources?)**

* More RAM
* More time

**Appendix: (Put all information you collect during the project over here, including reference URL’s, communications, discussions, notes etc.)**

1. http://mathproofs.blogspot.com/2005/07/mapping-cube-to-sphere.html

2. <https://en.wikipedia.org/wiki/Geodesic_polyhedron>

3. <https://en.wikipedia.org/wiki/Perlin_noise>

4. http://www.mit.edu/~jessicav/6.S198/Blog\_Post/ProceduralGeneration.html