## horizontal line



Procedurally Generated City

2.05.2019

**─**

Scott Ritchie

AIE Sydney

# 

# Overview

A system that can procedurally generated cities at runtime, and been visually and functionally modifiable so the user can change the system to suit their needs.

# Required Libraries

1. Delaunay Triangulation & Voronoi Fracture

<https://github.com/OskarSigvardsson/unity-delaunay>

Used the generate roads for suburbs and blocks.

1. Mesh Slicing

<https://github.com/DavidArayan/ezy-slice>

Used to slice ground tiles for positioning and alignment

# 

# 

# Implementation

The system is planned to be completely open source, so it can be modified for the users needs. It will be compiled into a package with example assets.

**General**

How to use the Procedural City Generator

Import the Package and Download the required Libraries.

Open the Sample Scene.

Press Play.

To Integrate into your own game, Copy the Script from the sample scene to a gameobject in your project. Apply a scene and call “GenerateCity”.

Modify the values in the inspector to change how the city is generated.

**Buildings**

New buildings can be added by creating a prefab of a building, with the pivot point at the bottom center.

Drag the Prefab into the Building List in the appropriate class in the inspector (Residential, Urban or City )

Add a size in the building size list below.

**Paths**

Change Paths by creating a 5x5 plane with the pivot in the bottom right facing inwards for the object.

Create a prefab and drag it into the Path prefab slot. This can be left empty to not generate paths.

# Algorithms & Mathematical Operations

## Line and polygon *(Mathematics)*

* 1. Line intersection and checks to use with road creation
  2. Polygon shrinking to shrink polygons so other objects can be created with offsets from the street. Using line intersections and checks.

## Procedural Generation & Noise *(Mathematics)*

* 1. Using Unity Perlin Noise and Random seed values

## Suburb and Block Position Placement *(Mathematics & Algorithms)*

* 1. Using Noise and seeded random values to generate aligned and unaligned suburbs, blocks and roads.

## Road & Path placement *(Algorithm)*

* 1. Calculate angles between road vectors to use with a mesh slicer to slice the ends of roads so they are parallel with each other.
  2. Use trigonometry to find the offset of the road tile placements.

## Building Placement *(Algorithm)*

* 1. trigonometry to find the if a building can be placed along a street and to start and end offset to make sure it does not intersect with any streets or other buildings.

# Polygon Shrink Algorithm

FOREACH vertex

Move vertex perpendicular by the shrink amount

Calculate the new points using a line intersection current vertex to the next vertex and the previous and current vertex

FOREACH new vertex

IF the next vertex p is to the right of the direction v from the previous p to current p

STORE p and begin a search for the next vertex thats to the right of its previous v , this means the invalid/invertex polygon section has ended

Find the line interesection between the first right and the end right vertex.

Add the intersection point to the new polygon

If its to the left

Just add it to the new polygon normally

RETURN the new polygon

# Block Position Placement

Seed Random

Shrink Suburb polygon

Find the bottom left pivot point of the polygon

Find Bounding volume of the polygon

FOREACH point in polygon

Rotate The point around the pivot

FOR EACH point spaced within the bounding volume

Generate a new adding the seeded random deviation from the location

IF the new point is within the Polygon Triangulation

Rotate the point back around the pivot

Calculate the Voronoi Diagram with these points

# Road Placement

FOR EACH vertex in the street polygon

Calculate this road direction (current to next vertex)

Calculate the next road direction(next to next next vertex)

Calculate the Previous road direction(previous to current vertex)

Calculate half the angle between the current and next road direction

Calculate half the angle between the previous and current direction

Calculate the Distance of the currentRoad

Calculate the Distance from the front to the lenght of the road where the perpendicular angle of the road direction is equal to the road width – begin length

Calculate the Distance from the end back along the length of the road where the perpendicular angle of the road direction is equal to the road width – known as the middle end

FOR the distance from 0 to the begin length

Create a new road object

Slice the road object along the front slice direction

IF the lenght after this segment is greater then the begin length

Slice the back off the current road object so its equal to the begin length

FOR the distance from the begin to the middle length

Create a new road object

IF the length after this segment is greater then the begin length

Slice the back off the current road object so its equal to the begin length

FOR the distance from the begin to the middle length

Create a new road object

Slice the road object along the end slice direction







