# Complex Game Systems Design Brief

## Procedural Dungeon Generation

# Identify what your system is by name and description.

The system that I will be creating will be a modular procedural dungeon generator.

# Outline the objective and use of your system.

The system that I will be creating will be a modular procedural dungeon generator that will allow users to create either a 2D Dungeon or 3D Dungeon. My end goal aim is to have users install my custom package from Unity’s asset store to implement into their own game. I am hoping to include a heap of features into the project such as:

* Scalable levels (giving users the option for having multiple floors within their dungeon game)
* Include different algorithms allowing the user to experiment with what might suit their needs
* Allowing the user to use their own assets within the generator
* Include the ability to random add any environmental assets they have into the generator (doors, chests, enemies, loot, etc)

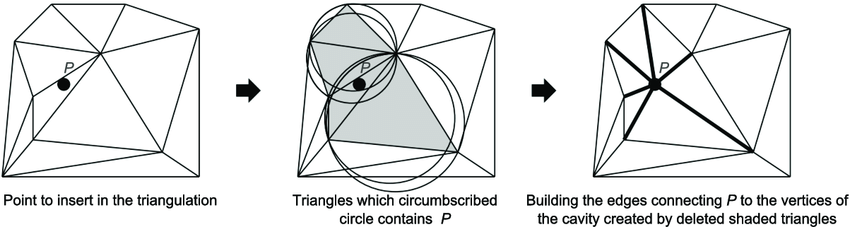
I am hoping that given the time that I must research and complete this task that I will be able to create this to an industry standard.

# If applicable, describe and reference any 3rdparty libraries that your system relies on, otherwise Identify you are not using any.

I do not believe that I will be using any third-party libraries for the creation of this dungeon generator, however this may change.

# Identify and outline the mathematical operations involve, allowing your system to function as intended.

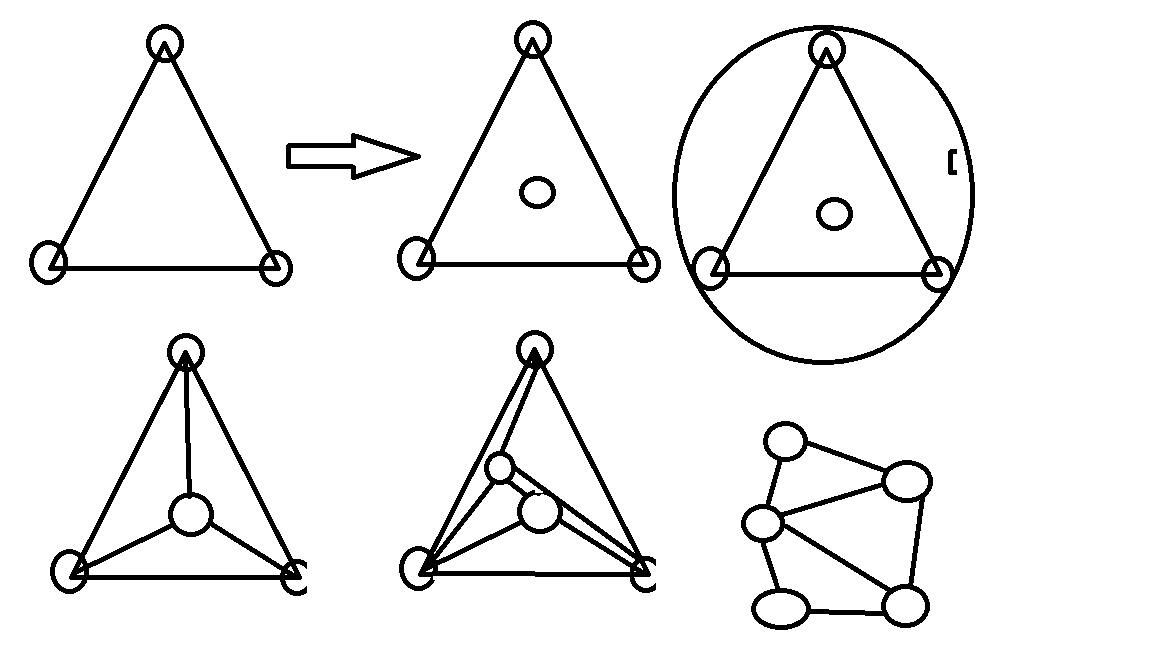
The mathematical operations that will be involved in this system will mainly come from the implementation of using Delaunay Triangulation. The plan that I have for this implementation is to take inspiration from the Bowyer-Watson algorithm to assist in creating the dungeon generator as well as bringing in prims algorithm for minimum spanning trees to allow rooms to be connected.



The Delaunay Triangulation is a triangulation where no vertex exists within the circumcircle (as shown in the figure). Bowyer-Watsons implementation of a Delaunay Triangulation works by

* Creating a list of empty triangles
* Create a large triangle or a ‘super-triangle’ within the list
* Adding different points randomly within the super triangle (one by one)
* When a point is added all triangles are checked, if an existing triangle interacts with the point the intersecting triangle is deleted.
* Find the position and create new triangles that form edges of the polygon and the new point.

The following process would look like this:



Obviously this is a very rough sketch but the process is still the same. The sixth drawing is the end result that I am looking for when implementing this algorithm into my dungeon generator.

# Explain what advance algorithm/s you will be implementing (diagram/s could be used to help support your explanation).

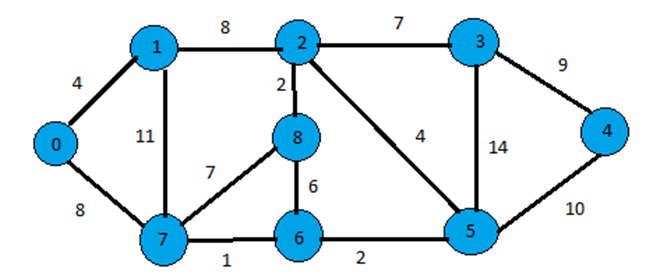
For the creation of my modular procedural dungeon generator I aim to implement algorithms such as the Delaunay Triangulation algorithm alongside Prim’s implementation of the minimum spanning tree to allow the randomly placed rooms to link together as corridors.

[Chart, radar chart

Description automatically generated](https://gwlucastrig.github.io/TinfourDocs/DelaunayIntro/index.html)

The Delaunay Triangulation is a very efficient way to map out the dungeon. The points in a Delaunay (as shown to the left) would represent the different rooms whilst the lines connecting these nodes would be the corridors linking them together.

fig 1.0 – Image taken from tinfour docs - [Delaunay Triangulation (gwlucastrig.github.io)](https://gwlucastrig.github.io/TinfourDocs/DelaunayIntro/index.html)



A minimum spanning tree is described as a ‘subset’ of edges that connect to a graph with various vertices. The algorithm that I will be working along with to allow my minimum spanning tree (MST) to work will be Prim’s algorithm. Prims algorithm works by starting at one vertex and as it goes along it keeps adding edges with a low weight until a goal is reached. In my case this would be a room

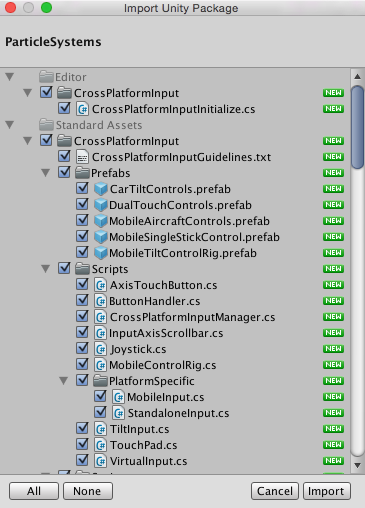
fig 1.2 – Image taken from includehelp – [Prim’s Minimum Spanning Tree](https://www.includehelp.com/data-structure-tutorial/prims-minimum-spanning-tree.aspx)

within the dungeon

# Illustrate how your system should be integrated into an application.

The hope is that my procedural generation system will allow people trying to make dungeon levels more streamlined and will ultimately take them less time then coming up with a solution themselves, because I’m aiming for it to be modular my hope is that it will be able to work for tile-based games as well as 3d games that players are able to walk around.

When users go to install my custom unity package they will be met with various files such as readme’s, changelogs, folders such as Editor, Runtime, and, tests. Below is an example of how it might look. The end goal is something that will easy to understand and implement into any project.



# Prove how you will design your complex system to be modular. (diagram/s could be used to help support your explanation)

I aim to make my complex system modular by incorporating different ways that the dungeon can be made. I am hoping to have a system in place where the user can attach scripts to a game-object and attach prefabs such as floors, walls, enemies, and other environmental objects. The users will be able to download the project from the asset store and import the scripts required to make the dungeon generator.

# Provide a reference list of the sites used following the Harvard Referencing method.

vaishnavi8055 (2020) *Binary space partitioning*, *GeeksforGeeks*. GeeksforGeeks. Available at: https://www.geeksforgeeks.org/binary-space-partitioning/ (Accessed: May 2, 2023).

*Delaunay triangulation* (2023) *Wikipedia*. Wikimedia Foundation. Available at: https://en.wikipedia.org/wiki/Delaunay\_triangulation (Accessed: May 2, 2023).

wo80 (2018) *WO80/triangle.net: C# / .net version of Jonathan Shewchuk's Triangle Mesh Generator.*, *GitHub*. Available at: https://github.com/wo80/Triangle.NET (Accessed: May 2, 2023).

Technologies, U. (no date) *Creating custom packages*, *Unity*. Available at: https://docs.unity3d.com/Manual/CustomPackages.html (Accessed: May 2, 2023).

Staff, C.G. (2021) *Delaunay triangles*, *CodeGuru*. Available at: https://www.codeguru.com/cplusplus/delaunay-triangles/ (Accessed: May 2, 2023).

Lucas, G.W. (no date) *Introduction*, *Delaunay Triangulation*. Available at: https://gwlucastrig.github.io/TinfourDocs/DelaunayIntro/index.html (Accessed: May 2, 2023).

Roguelike Celebrations (2020) *Herbert Wolverson - Procedural map generation techniques*, *YouTube*. YouTube. Available at: https://www.youtube.com/watch?v=TlLIOgWYVpI&t=333s (Accessed: May 2, 2023).

*Determinant* (2023) *Wikipedia*. Wikimedia Foundation. Available at: https://en.wikipedia.org/wiki/Determinant (Accessed: May 2, 2023).