# 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. The end goal is to have users install my custom package from Unity’s asset store to implement into their own game. The features that I will showcasing in my project will be features 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)

# 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.

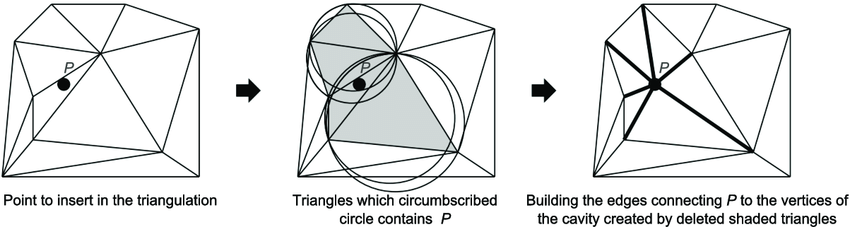
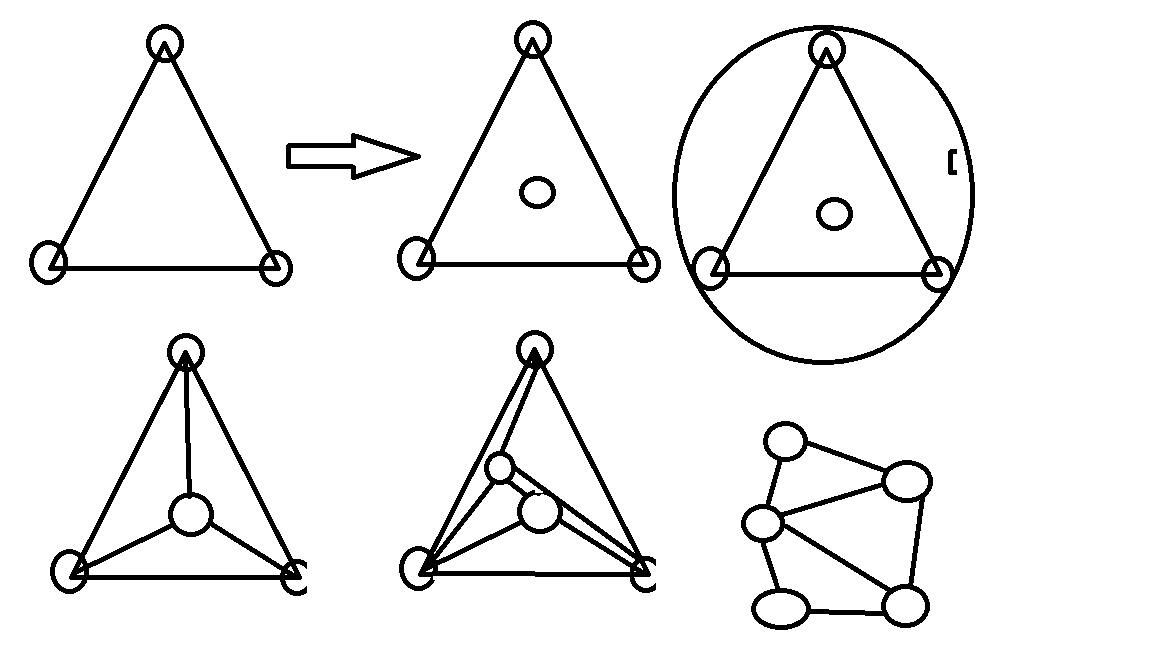


fig 1.0 – Image taken from Research Gate - [Bowyer Watson Algorithm](https://www.researchgate.net/figure/Insertion-of-one-point-in-a-Delaunay-triangulation-Bowyer-Watson-algorithm-modified_fig22_280881453)

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.

The math involved for the circumcircle

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

My System will incorporate different algorithms for the creation of my dungeon generator, The algorithms that will be include in the product being the Delaunay triangulation algorithm, the Binary Space Partitioning and to accompany these algorithms I will be using a prim’s implementation of a minimum spanning tree to link the rooms together.

[Chart, radar chart

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

fig 1.1 – Image taken from tinfour docs - [Delaunay Triangulation (gwlucastrig.github.io)](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 right) would represent the different rooms whilst the lines connecting these nodes would be the corridors linking them together. The Delaunay triangulation is a triangulation where no points or vertices lie within the circumcircle of any triangle. A circumcircle is a circle which passes through the three vertices of the triangle or super-triangle when

[A picture containing shape

Description automatically generated](https://www.geeksforgeeks.org/binary-space-partitioning/)

fig 1.1 – Image taken from geeksforgeeks - [Binary Space Partitioning - GeeksforGeeks](https://www.geeksforgeeks.org/binary-space-partitioning/)

Binary Space Partitioning works by getting a large starting area and then splitting it into smaller rooms which creates smaller rooms for the dungeon. The way the splitting works is by choosing a random direction (either being horizontal or vertical) then choosing a random position (same x-y directions) and then from there splitting the dungeon into subsections.

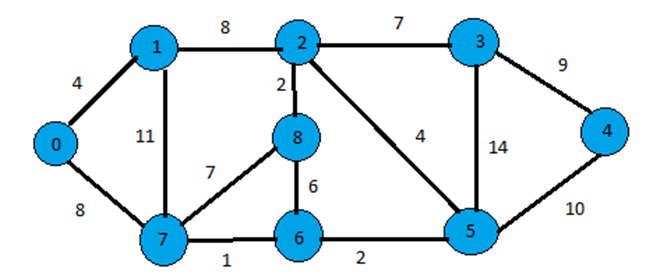


fig 1.2 – Image taken from includehelp – [Prim’s Minimum Spanning Tree](https://www.includehelp.com/data-structure-tutorial/prims-minimum-spanning-tree.aspx)

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 within the dungeon.

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

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 the system will be modular it will work for tile-based games as well as 3d games. When users install my custom unity package they will be able to drag either the 2D or 3D generator script into an empty game-object which will feature a list of functionality depending on which script has been bought onto the game-object. From there they can adjust to fit their specific needs. When users view the script on the inspector, they will be able to change variables such as:

* Floor & Wall prefabs
* Various game objects to be scattered around the dungeon procedurally
* The option to enable more than one level and stairs to navigate up and down.

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

My complex system will be 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).