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 come from the implementation of using Delaunay Triangulation & Binary Space Partitioning. I plan to make use of Bowyer-Watsons implementation of Delaunay Triangulations to assist in the procedural creation of dungeon rooms, alongside this implementation I hope to make use of either prim's or Kruskal's implementation of minimum spanning trees. Alongside using triangulations to create procedurally made dungeon rooms I will be making an alternate way for users to create dungeons using Binary Space Partitioning.

DELAUNAY TRIANGULATON MATH

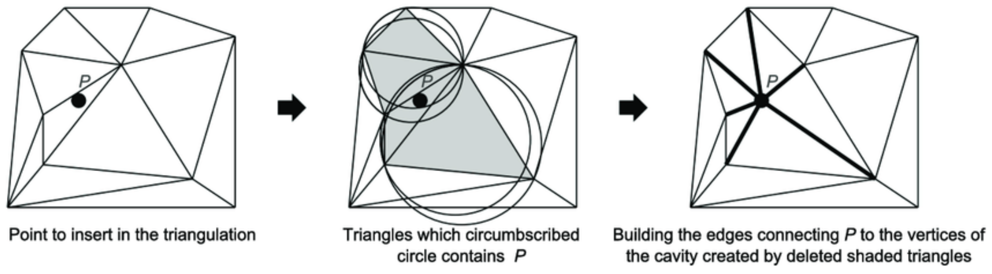
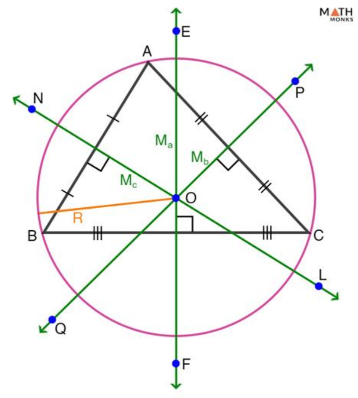

P
Point to insert in the triangulation
P
Triangles which circumbscribed circle contains P
P
Building the edges connecting P to the vertices of the cavity created by deleted shaded triangles

fig 1.0 – Image taken from Research Gate - Bowyer Watson Algorithm

The required math for Bowyer-Watsons implementation of a Delaunay Triangulation involves lots of geometry calculations, the first step in ensuring a successful implementation is to find the circumcircle of a triangle and finding whether the randomly placed point lies within the super-triangle. This crucial math is done to identify if triangles that exist within the current list of triangulations needs to be removed or not and how it will connect when new points are created, when this happens new triangles are created as well.

To calculate the circumcircle of the triangle I must first find the perpendicular bisectors of the triangle (fig 1.01) which in this case is 0. From there the circumradius needs to be found. The circumradius is the distance from the centre, in our case 0, to each of the triangles three vertices (A, B, C) from there we need to test if the point lies within the triangle, if so it is positive, if not it is negative and then removed. For the example to the right M(a), M(b) and M(c) are the perpendicular bisectors as 0 intersects all those points, this would make the circumcenter

OA = OB = OC.

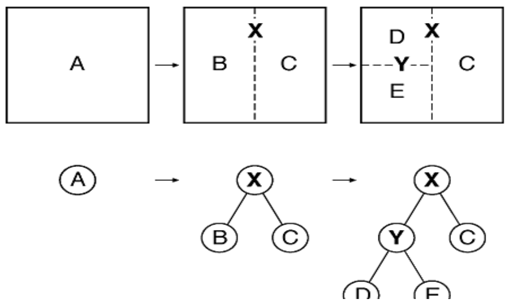


In ΔABC, 'M_a', 'M_b', & 'M_c' are the 3 perpendicular bisectors

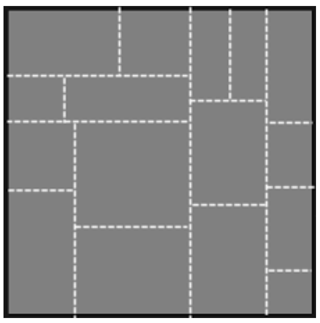
Fig 1.01

BINARY SPACE PARITIONING MATH

The Binary Space Partitioning algorithm works by recursively dividing the initial space into two half-spaces. As shown to the right the initial space is A, the algorithm is applied to A and then once it has been split the algorithm is applied to B and C where A becomes the parent node of both, from here we need to start portioning from the leftmost nodes and then to the rightmost nodes.



By doing this we are able to end up with something such as this:



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 included within the project will be algorithms such as the Delaunay triangulation algorithm, I will be making use of Bowyer-Watsons approach to this. Binary Space Partitioning and to accompany these algorithm I will be making use of prims algorithm for minimum spanning trees.

THE DELAUNAY TRIANGULATION ALGORTHM ALONGSIDE BOWYER-WATSON IMPLEMENTATION

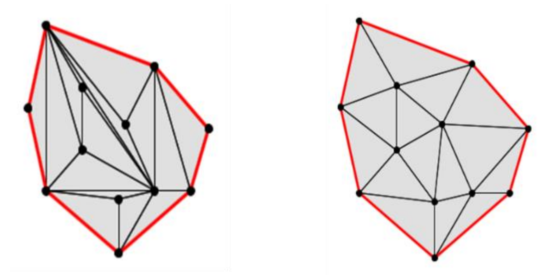


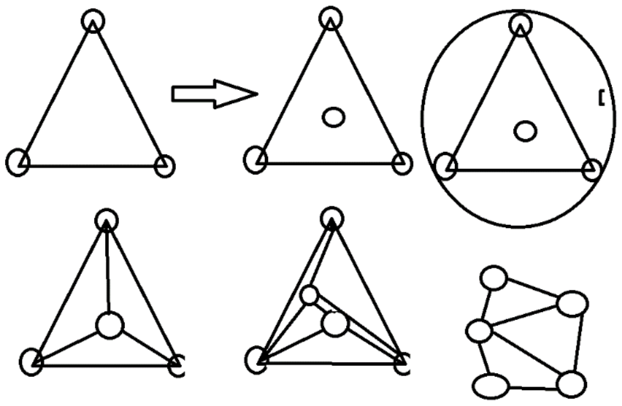
fig 1.1 – Image taken from tinfour docs - Delaunay Triangulation (gwlucastrig.github.io)

The Delaunay Triangulation is a very efficient way to map out the dungeon. The points in a Delaunay 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.

The Steps needed for the Bowyer-Watsons implementation to work for my implementation of the Delaunay Triangulation is as follows:

- Creating a list of empty triangles
- Create a large triangle or a 'super-triangle' within the list so that it can contain all points
- 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 boundary of the polygon formed by the intersecting triangles and create a new triangle formed edges and a new point

The following process would look like this:



THE BINARY SPACE PARITIONING ALGORITHM

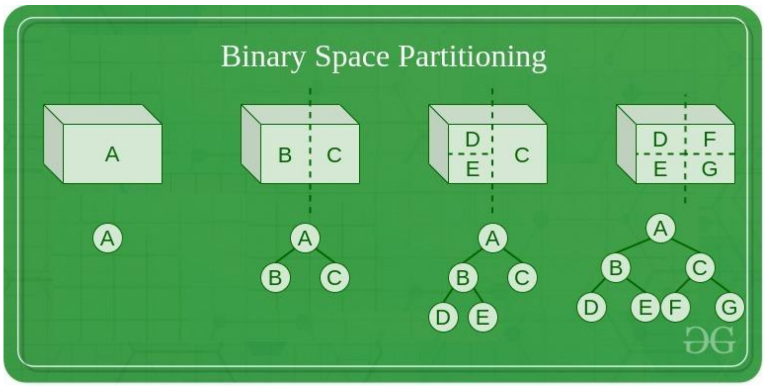
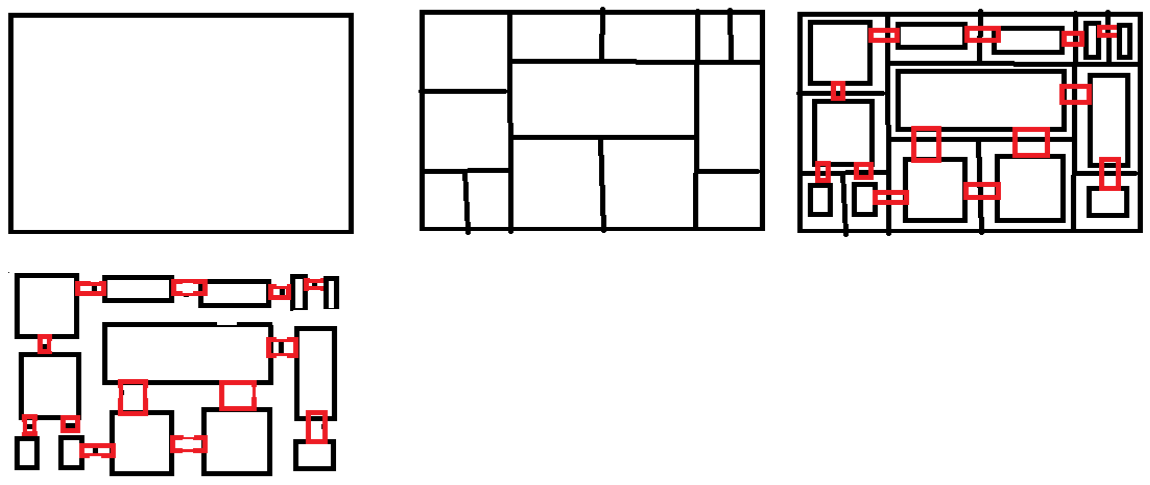


fig 1.1 – Image taken from geeksforgeeks - Binary Space Partitioning - GeeksforGeeks

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 initial starting room is recursively divided into two until various partitioning conditions are met. Below is a diagram that showcases how the process works until a dungeon like structure is generated. Each Iteration within the partitioning process the horizontal and vertical direction is assigned randomly this is also true for the size of the first partition or room, as it moves along each room is generated randomly based on how much space is left over.



MINIMUM SPANNING TREES – PRIMS ALGORITHM

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).

Delaunay triangulation and Triangle Storage (2021) *Delaunay Triangulation and Triangle Storage*. Available at: https://theor.xyz/mapgen/delaunay-triangulation-triangle-storage/ (Accessed: 09 May 2023).