Aanpak Kruskal's algorithm verwerken in Unity

Sources:

https://github.com/willy1989/kruskal

https://github.com/brwagner/3d-maze/tree/5dbbe490be6c8a5e9108e620307f6573594c36eb

http://weblog.jamisbuck.org/2011/1/3/maze-generation-kruskal-s-algorithm

http://codehowtos.blogspot.com/2011/04/building-maze.html

Inleiding.

Ik heb onderzoek gedaan naar bestaande projecten van het Kruskal's algoritme in C#/Unity maar ook in andere talen. Enkel loop ik er enorm tegen aan dat ik het wel kan copy pasten en kan verwerken maar dan heb ik er niks van geleerd en zou ik het niet gemakkelijk nog een keer kunnen doen met een ander voorbeeld.

Generating nodes.

Referentie punt:

https://catlikecoding.com/unity/tutorials/procedural-grid/

Ik heb een klein deel van de tutorial gevolgt, voornamelijk heb ik naar deze 2 screenshots gekeken:

Let's visualize these vertices so we can check that we position them correctly. We can do so by adding an `OnDrawGizmos` method and drawing a small black sphere in the scene view for every vertex.

```
private void OnDrawGizmos () {
	Gizmos.color = Color.black;
	for (int i = 0; i < vertices.Length; i++) {
		Gizmos.DrawSphere(vertices[i], 0.1f);
	}
}
```

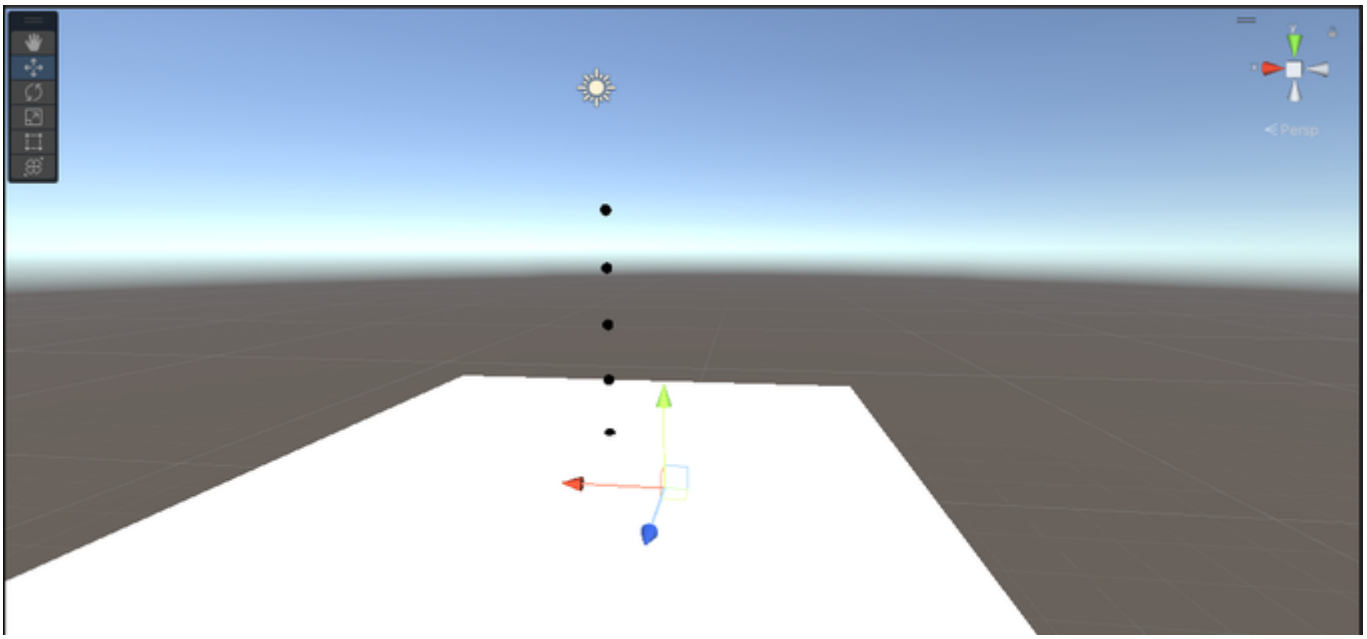
```
private void Generate () {
	vertices = new Vector3[(xSize + 1) * (ySize + 1)];
	for (int i = 0, y = 0; y <= ySize; y++) {
		for (int x = 0; x <= xSize; x++, i++) {
			vertices[i] = new Vector3(x, y);
		}
	}
}
```

Daar heb ik vervolgens dit van gemaakt:

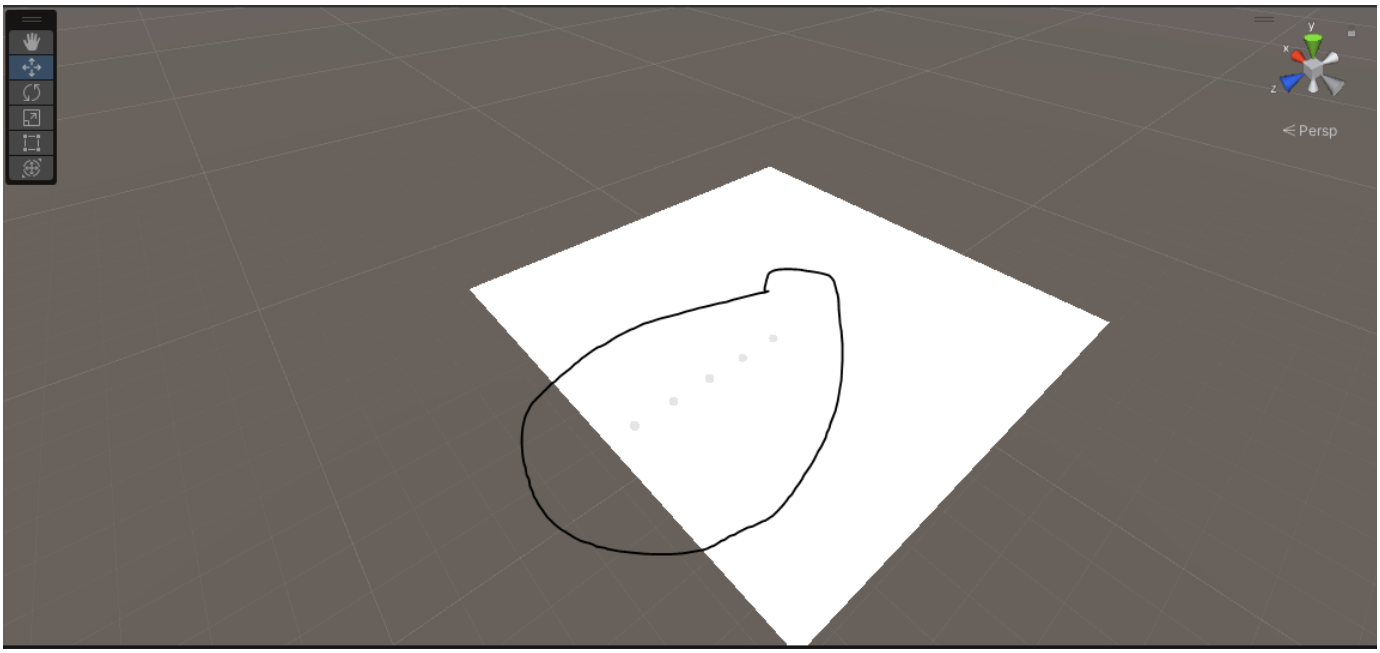
```
    /**
     * Loops through the given width and height, based on the width and height a node object is created.
     */
    1 usage
    private void GenerateNodes()
    {
        nodes = new List<Node>();


        for (int x = 0; x < width; x++)
        {
            for (int y = 0; y < height; y++)
            {
                nodes.Add( item: new Node(x, y));
            }
        }
    }

    Event function
    private void OnDrawGizmos() {
        if (nodes == null || nodes.Count == 0) return;

        Gizmos.color = Color.black;

        // Loop through nodes and draw a sphere based on the nodes X and Y, this is done for debugging purposes
        foreach (Node node in nodes)
        {
            Gizmos.DrawSphere( center: new Vector3(node.X, node.Y),  radius: 0.1f);
        }
    }
}
```

Nu is het de bedoeling dat ik de nodes altijd op de “floor“ plaats.

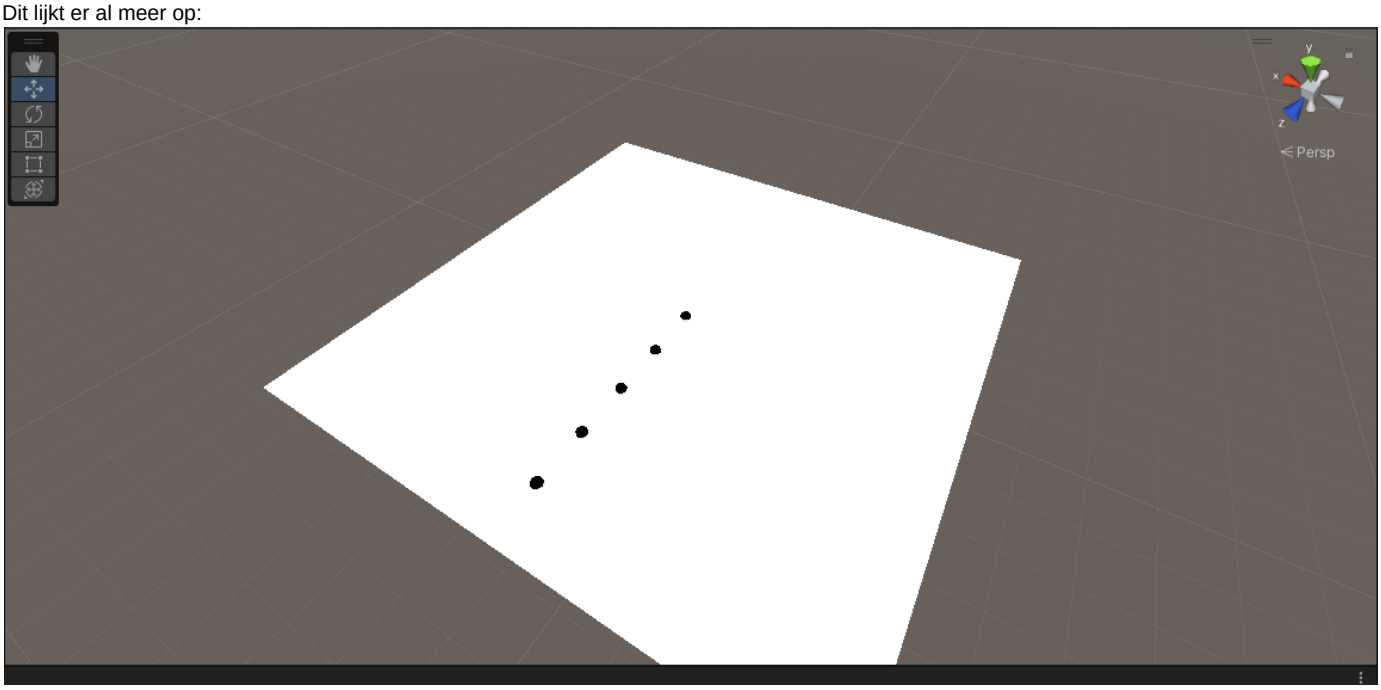
werk met X en Z niet X en Y



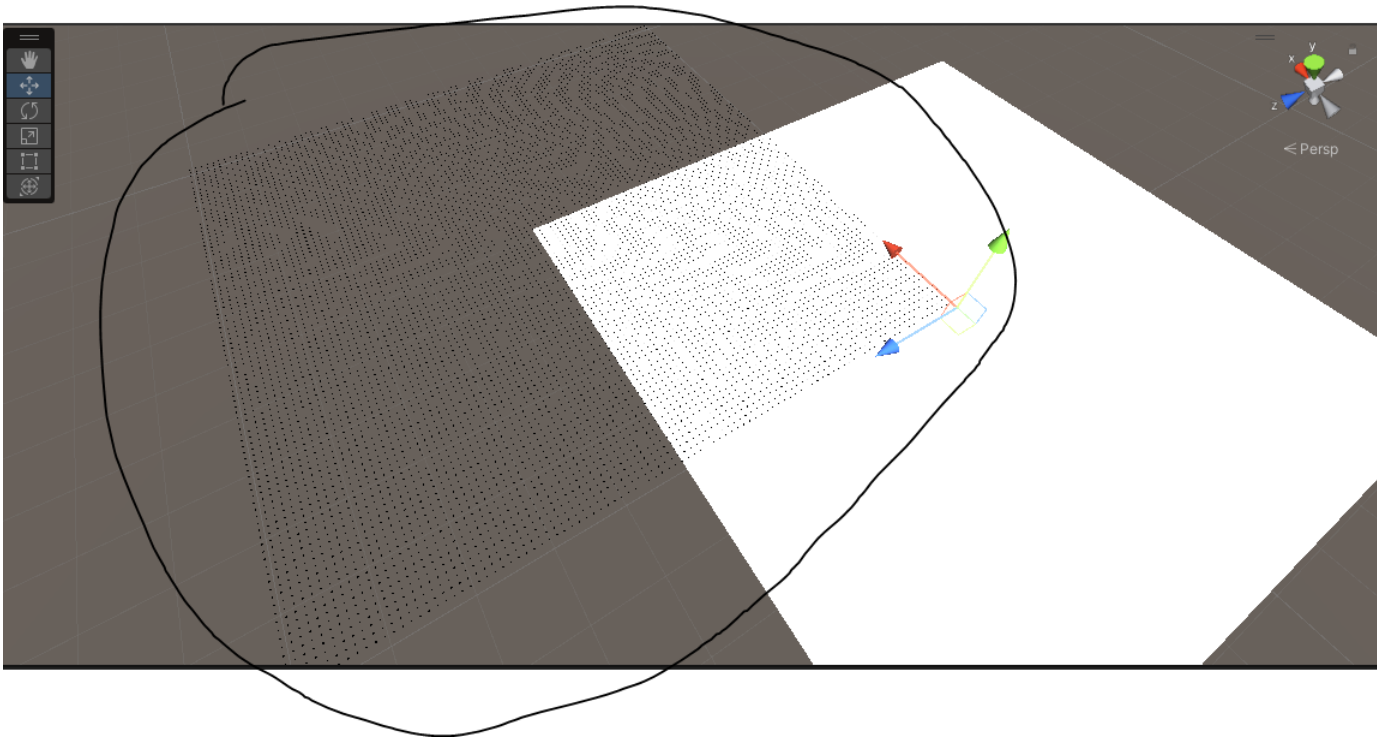
Moet ook met een Y werken, maar die is fixed



Dit lijkt er al meer op:



Zoals ik had verwacht moet ik niet het middelpunt makken maar een hoek.



Heb een hoekpunt, maar ben nu aan het strugglen met het verspreiden richting de rest van de floor.

Om alle hoeken te berekenen heb ik dit als referentie punt gebruikt:

https://answers.unity.com/questions/1833942/how-to-align-gameobjects-based-on-their-corner-not.html

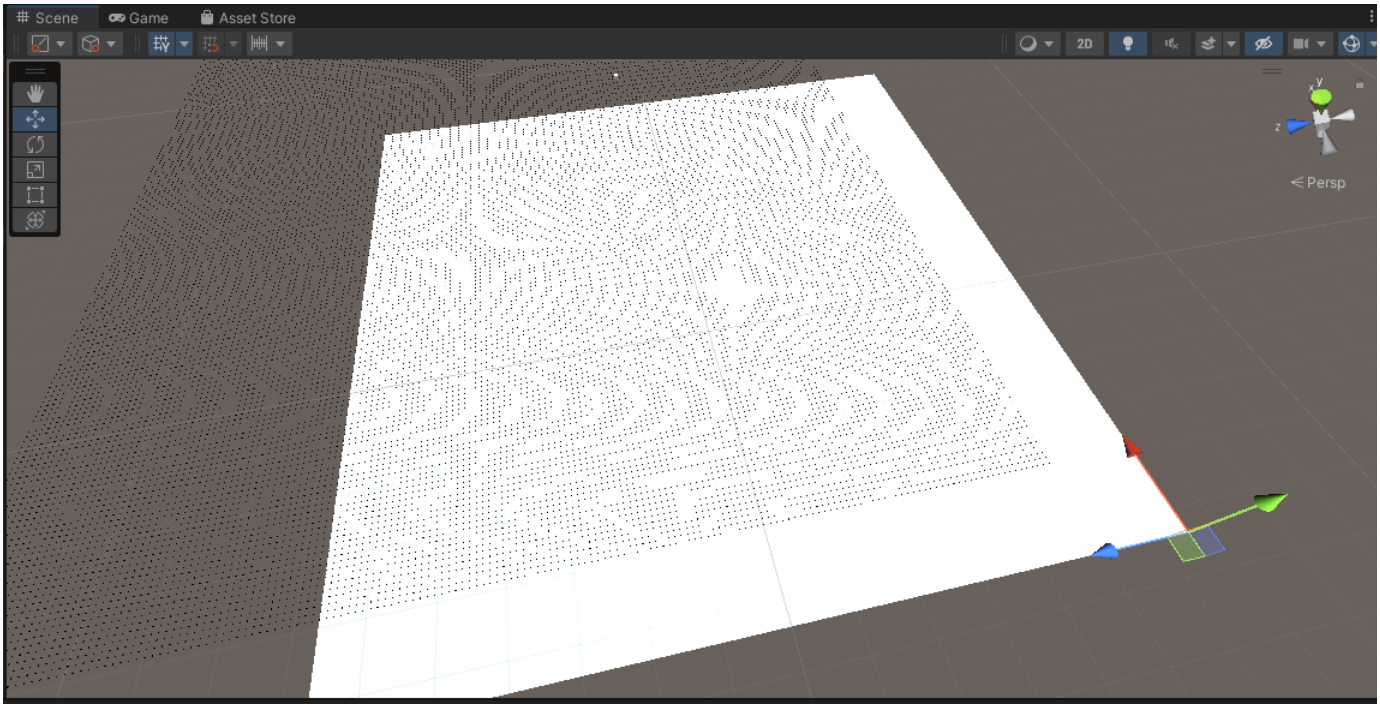
Dat heb ik vervolgens verwerkt naar:

```
2 usages
private Vector3[] GetCornersOfGrid()
{
    Vector3[] corners = new Vector3[4];
    Vector3 middlePoint = gridTransform.position;

    Bounds bounds = collider.bounds;

    // bottom left
    corners[0] = bounds.min;
    // upper right
    corners[1] = bounds.max;

    // upper left
    corners[2] = new Vector3(x: middlePoint.x - bounds.extents.x, middlePoint.y, z: middlePoint.z + bounds.extents.z);

    // bottom right
    corners[3] = new Vector3(x: middlePoint.x + bounds.extents.x, middlePoint.y, z: middlePoint.z - bounds.extents.z);


    return corners;
}
```

Ik had enkel even een voorbeeld nodig hoe ik “upper left“ uitreken, toen snapte ik “bottom right“ ook.

Nu is het tijd om te zorgen dat de nodes correct over het grid (floor) word verspreid

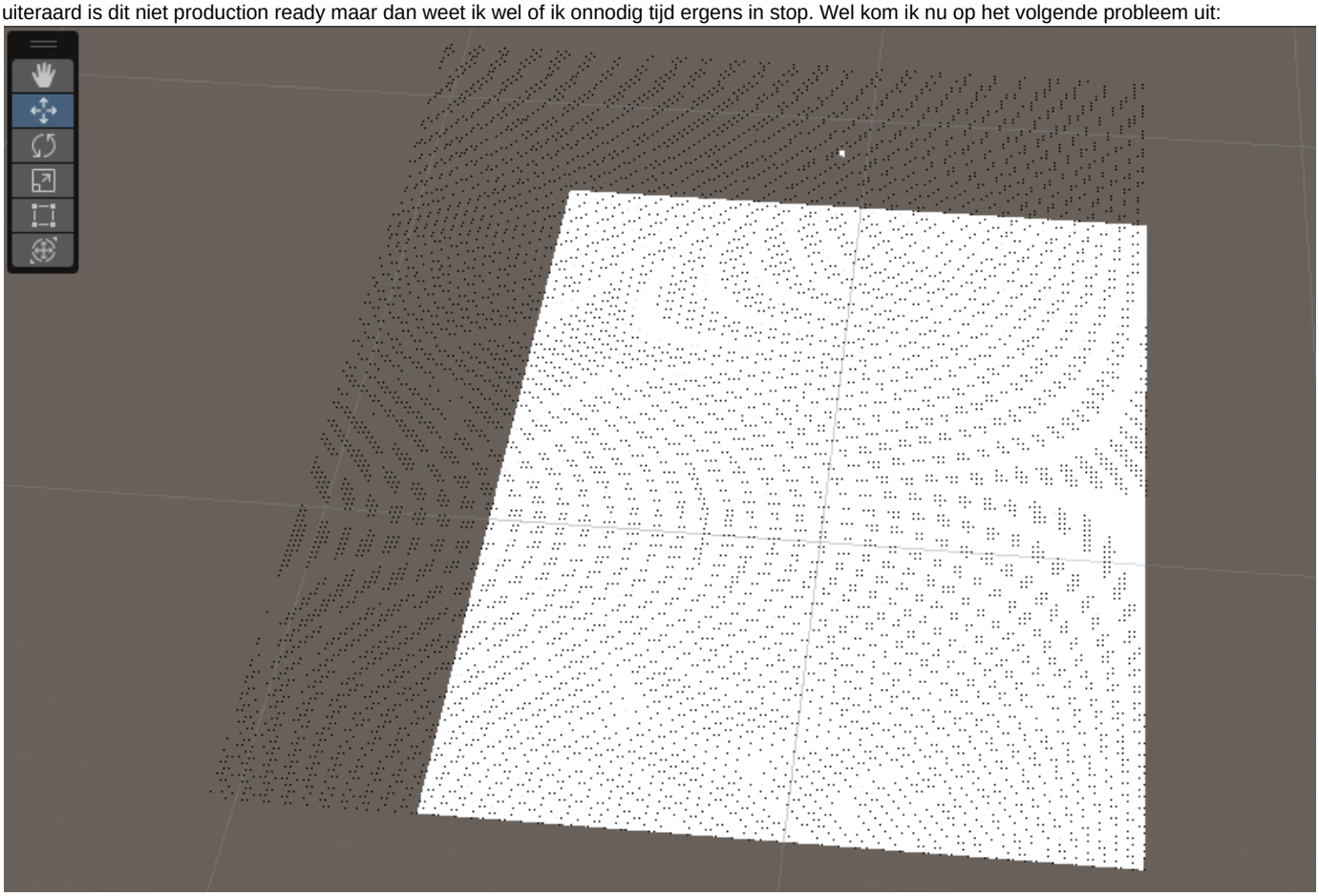
Op dit moment is het probleem dat de nodes niet beginnen bij de bottom left corner, het ziet er zo uit:



Wat raar is want volgens de debugs (en het gameobject wat ik inspawn) klopt de locatie wel van bottom left (LET OP: door perspectief lijkt het nu bottom right)

Ik denk dat het probleem ligt dat ik niet snel genoeg de mesh collider update nadat ik de scale van het transform verander, daardoor denkt de mesh collider dat bottom left nog heel ergens anders is.

Na het bovenstaande te debuggen lijkt het erop dat ik gelijk heb en heb ook al het juiste resultaat gekregen door het constant te updaten, uiteraard is dit niet production ready maar dan weet ik wel of ik onnodig tijd ergens in stop. Wel kom ik nu op het volgende probleem uit:



Zoals je kan zien spawn ik teveel nodes in, dus ik ga kijken of ik er tevaak doorheen loop (omdat de “rekensom“ dan niet klopt).

Ik denk dat het komt omdat ik altijd een spacing van “1“ hou op de X en Z as (omdat de for loop altijd + 1 doet), daarom komt het er buiten te staan. Ik ga nu nadenken over de keuze of ik de spacing minimaliseer (naar 0.01) of ik de spacing uitreken en eerder klaar ben met loopen

UPDATE.

Vanaf hier heb ik besloten om hoe ik het project aanpak compleet te veranderen, lees document “zelfreflectie” voor details en extra informatie.