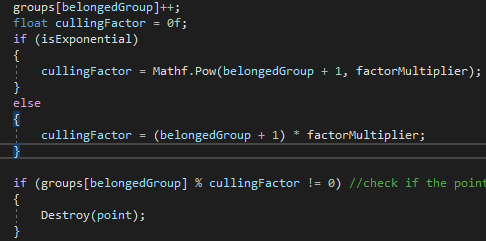
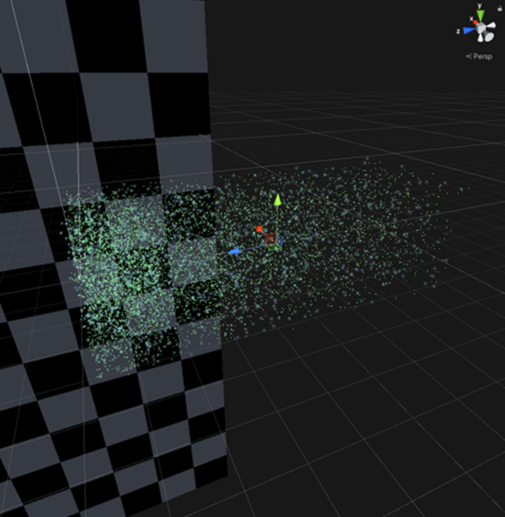
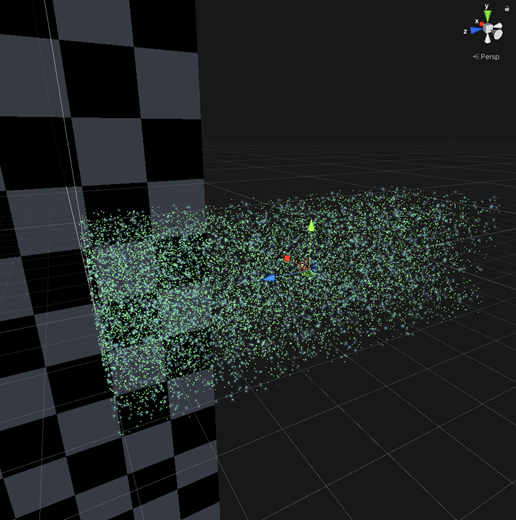
**Summary**This document explains how the distance optimization for point-cloud works and where it can be used. I made this optimization in Unity in one C# script for optimizing a large number of GameObjects by destroying the ones further away from the camera.  
  
This optimization method was meant to be efficient with iterating through thouthands of objects, without adding them to the lists or arrays, since their number would affect the performance greatly.  
  
Previous design that I used was using box collisions to detect the objects within them, adding them to several lists and then iterating through those lists to destroy the objects. However, it didn’t prove to be very efficient, since checking for collisions for the large number of objects cost a lot of time. Thus, I reworked it by making this method instead.  
  
**How it works**It is one simple C# script that is attached to the parent of all the objects that need to be optimized. It iterates through all the objects ones, checks each object’s position and compares it to the camera’s position, then it determines the group that object is supposed to be in with a linear interpolation, like so:

Then it adds +1 to the number that represents that group in the array of int[] to keep track of how many objects there are in the group.  
Lastly we calculate the culling factor which determines how many objects are going to be destroyed in the group. Culling factor takes the number of the group as a base and then either multiplies it by the factor multiplier or is being raised to the power of it, depending whether the developer chose isExponential option or not.  
  
  
  
The function will keep every *n* object, where *n* is the culling factor, and destroy all the other objects, so the bigger the culling factor is, the more objects are going to be destroyed.  
  
  
  
**How to use it**Attach it to the parent object of all the objects that are needed to be optimized, tweak the parameters as you like and while you run the game, press spacebar to activate the optimization. This script is only a prototype, so calling the optimization method is a placeholder and just set to be activated by pressing Spacebar - change that however you need.  
  
The “max distance” parameter determines the range of the optimization, anything beyond that is going to be deleted. Then a number of groups is going to be evenly placed within that range with the “number of groups” parameter.  
  
“Factor multiplier” amplifies the number of destroyed objects and “IsExponential” boolean enables the culling factor to be raised to the power of the factor multiplier instead of just multiplying, enabling it will reduce the number of objects to a great extent.  
  
In this example the picture to the left uses normal multiplication and the picture on the right uses exponential culling:  
  
While this script works only with objects in Unity, the method itself can be used in other various situations with small adjustments.

**Recommendations**

This script works with points as separate objects and it is not ideal with big point-clouds that have over a million points in them. I would recommend loading the point-cloud as one mesh with points as its vertices and then try to implement the method that I used for optimization. It may be possible to check the position of those vertices and then remove them as I did in this script before loading the mesh to improve the performance.