Terrain Engine 2D

A 2D Block Engine for Unity

Out now on the Unity Asset Store

BUY NOW!

Optimization

What is the purpose of this page...

Table of Contents

General

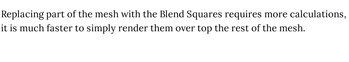
If you ever find yourself in the situation where you want to optimize your game, whether you need your app to run better on older mobile devices, or for any reason there are a number of ways in which you can make Terrain Engine 2D run more efficiently.

The biggest impact on your performance comes down to the features you are running from the engine. Every extra feature adds more load to the CPU, and the three largest culprits are the Lighting, Fluid and Falling Blocks systems. You can greatly increase your performance by either opting for the more basic versions of the Lighting and Fluid system or outright disabling them for a huge increase in performance.

To make things easier below I have listed 20 ways you can increase performance from greatest to least impact:

1. Disable Falling Blocks Falling Blocks is very CPU intensive as it iterates over a large array and performs calculations as blocks are falling
2. Disable Lighting Lighting is CPU intensive as it performs many lighting calculations and takes up memory
3. Disable Fluid Flowing fluid is quite CPU intensive and fluid data takes up memory
4. Decrease your Load Distance A lower Load Distance means less chunks loaded into scene which means everything runs faster
5. Use the Basic Lighting system This saves memory and is less CPU intensive as less light calculations are performed and there is no post processing
6. Use the Basic Fluid system This saves memory as less information is stored and is less CPU intensive due to not having to perform color calculations
7. Don't enable colliders Colliders have to regenerate every time a chunk is modified, don't use colliders for a bump in performance on chunk loading and terrain modification
8. Lower Lighting Post Processing values Post Processing puts a constant load on the gpu, lower these values to increase all around performance
9. Decrease Light Spread and Transmission This decreases the amount of calculations required for determining light values, decreasing this value will increase performance of terrain modification
10. Use a smaller world Width and Height This will decrease the amount of memory used by your program
11. Disable the OSD The OSD creates garbage which causes lag spikes, and performs constant text updates which can slow down the game
12. Increase the OSD Update Rate Decrease the rate at which the OSD updates its text objects for an overall bump in performance
13. Decrease the Max Modify Radius The more blocks being modified at once, the more calculations required, so reduce the Max Modifying Radius to prevent slow terrain modification
14. Don't render fluid as a texture Post processing effects are performed on the fluid texture which can decrease performance
15. Use Occlusion Culling Using occlusion culling reduce the amount of draw calls
16. Overlap Blend Squares Allowing the overlap of blend squares reduces the amount of calculations required for generating terrain meshes
17. Increase the Chunk Load Rate value A higher chunk load rate means chunks will load in slower, reducing how often chunks have to load in is a good way to increase performance as it is a demanding task
18. Use less Block Layers More Block Layers increases the amount of draw calls
19. Decrease the speed of the Load Transform A faster Load Transform means chunks have to load/unload more often which is a demanding task
20. Don't use Overlap Blocks Overlap Blocks require more calculations when generating meshes

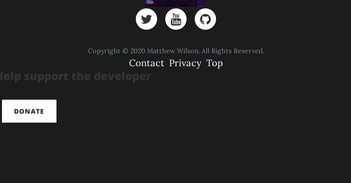
Optimization Properties



Terrain Engine 2D Optimization Properties

- Overlap Blend Squares Allows the option to overlap the 'blend squares' (used when generating Overlap Blocks) over the block's edges. By default the blend squares replace the block edge, but this adds a lot more vertices and triangles to the generated mesh
- Occlusion Culling If this option is selected blocks that are hidden behind other layers will not be rendered

The **Optimization** section of the World Inspector gives you a few extra options which you can enable if you wish to speed up your game. Currently there are two options; the first is to enable **Occlusion Culling**. This means blocks that are hidden behind others (in a background layer for example) will not be rendered as long as the frontal blocks are not transparent.



Occlusion Culling in Block Layers

In the image above the game has Occlusion Culling enabled, and it's foreground layers hidden. This way you can see how the background layer only has its visible blocks rendered. This reduces the amount of draw calls and vertices used to render the terrain mesh.

The second option is to **Overlap the Blend Squares** which optimizes the generation of Overlap Blocks. This may be an option for you depending on your textures. It is suggested that you test out this feature with your own game (zoom in and look closely at the edges of your terrain to see how it looks).

Overlap Blend Squares Comparison

In the image above you can see how when Overlap Blend Squares is enabled the Blend Squares are simply generated over top of the rest of the mesh, where as when Overlap Blend Squares is disabled the Blend Squares replace that part of the mesh.

Replacing part of the mesh with the Blend Squares requires more calculations, it is much faster to simply render them over top the rest of the mesh.