Volumetric Clouds 3.5+

For 3.0/old shading version, go here.

Suggestions & comments for this document are open and welcome

Table of Contents

// Getting Started

// Material properties

- Colors
- Shape
- Animation
- Dimensions
- Raymarcher (cfr:Raymarching)
- Debug

// Raymarching

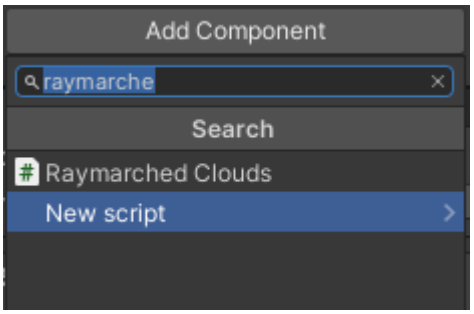
// Tips

// Optimisations

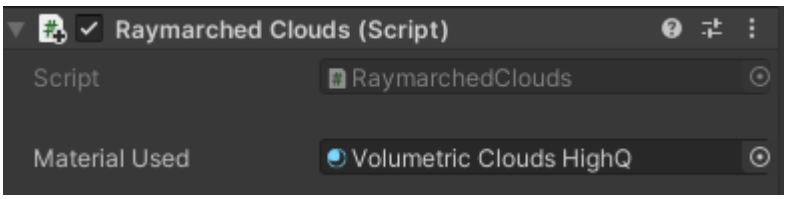
// Common issues

// Getting Started

1. Add “RaymarchedCloud” Component to your camera.



2. Assign either Volumetric Clouds HighQ or LowQ



// Material properties

- *New Shading*
  Switch the way clouds are rendered to the current version, only keep this off if you’re using an old version and don’t have the time to convert your materials to the new method.
- *Perlin Normal Map*
  Texture which dictates where and how clouds should be rendered.

Colors

- *Base Color*
  Main color, every other color will be tinted by this.
- *Shading Color*
  Darkest color that the clouds can render at.
- *Direct Contribution*
  Amount of direct lighting, that is the lighting coming directly from your main directional light, that contributes to the shading. Setting it to zero disables direct lighting computation entirely, saving performances.
    - *Normalmap*
      Enables normal-direction based lighting computation
        - *Shading Intensity*
          Intensity of the diffusion of light onto the clouds.
        - *Normalized*

Pulls back the gradient shading the clouds.

        - *Silver Lining*
          Amount of light passing through clouds.

- *Sky Contribution*
  Amount of indirect lighting, in this case mainly the sky's atmosphere, that contributes to the shading. Setting it to zero disables its computation entirely, saving performances.
    - *Sky Occlusion*
      Intensity of the natural occlusion that the sky creates: the bottom of the clouds will be darker than the top.
    - *Sky Occlusion Normalized*
      Pulls back the gradient of said occlusion.
    - *Occlusion Tint*
      Tints the occlusion.
    - *Advanced Lighting*
      Samples the sky's color for the indirect lighting instead of using a simple white color.
        - *Indirect Saturation*
          Contribution of said sample to the indirect lighting.
- *Distance Blend*
  Controls the blending of clouds into the horizon based on draw distance.
- *Post Blend*
  Keeping it disabled saves performance but enabling it makes the above look better, keep it disabled if you don't see any difference.
- *Screen Space Shadows*
  Enables the rendering of shadows cast from clouds onto the scene's geometry.
    - *Shadow Color*
      Tint of the shadow.
    - *Blend Out Of Shadowmap*
      Tries to remove shadows cast by clouds out of the already existing shadows in the scene to avoid overlap.
    - *Volumetric*
      Actually samples shadows instead of projecting them, more accurate but requires more performance.
- *Render Shadows Only*
  As the name implies, only render shadows, to be used in conjunction with the script's toggle.
- *High Quality Point Light*
  Keeping it disabled saves performance but enabling it makes point light look better, keep it disabled if you don't see any difference.

Shape

- *Coverage*
  Amount and size of clouds.
- *Coverage Map*
  Same as above but using a texture mapped in world space, the red channel controls the density (black -> red = low intensity -> high intensity), tiling and offset is in world space so 100 tilling means that the texture's size is equal to 100 unity units.
    - *Coverage Map*
      Said texture.
    - *Coverage Low*
      Amount of coverage for black parts.
    - *Coverage High*
      Amount of coverage for white/red parts.
- *Density*
  How transparent clouds are.
- *Local Scale*
  The overall vertical size of the clouds within the maximum size set through 'Cloud Transform'.
- *Cloud Center Height*
  The height of the center of the clouds.
- *Cloud Flatness*
  How puffy clouds are.
- *Lod Base*
  Changes the mipmaps used, similar to reducing texture resolution.
- *Lod Offset*
  Same as above but based on the distance to the camera.

Animation

- *Speed*
  Clouds render by using two of the textures you provided 'on top of each other', this is the wind speed for the first layer.
- *Speed Second Layer*
  Same as above but for the second layer this time.
- *Wind Direction*
  Direction of the wind, Y and W are not used.

Dimensions

- *Cloud Transform*
  X sets at which height clouds are, Y controls the maximum size of the clouds vertically, Z & W controls the base texture offset on the X & Y axis.
- *Tiling*

Controls the horizontal size of the clouds.

- *Spherical*
  Maps the clouds around a sphere instead of a plane.
  - *Sphere Position*
    The sphere position in 3D space, W is unused.
  - *Spherical Mapping*
    Maps clouds in a manner similar to unity's sphere model.
  - *Sphere Stretch Horizon*
    With spherical mapping it blends poles out, otherwise stretches the texture towards the horizon.
- *Plane Alignment*
  Sets the axis on which the cloud plane lays.
- *Orthographic Perspective*
  A factor which controls a fake perspective if clouds are rendered in orthographic mode.

Raymarcher *(cfr:Raymarching)*

- *Draw Distance*
  Maximum distance at which clouds render.
- *Steps Max*
  Maximum number of steps allowed, increasing this greatly impact performance.
- *Step Min Size*
  Minimum distance allowed between each step, increasing this tells the raymarcher to avoid wasting his time on small details and try to render things further away instead.
- *Step Skip*
  The basic multiplier for the distance between each step, should in most cases stay at 1.
- *Step Parallel Mult*
  Scales the length of each step when 'the camera looks towards the horizon', increasing this value helps with rendering distant clouds while mostly keeping details sharp.

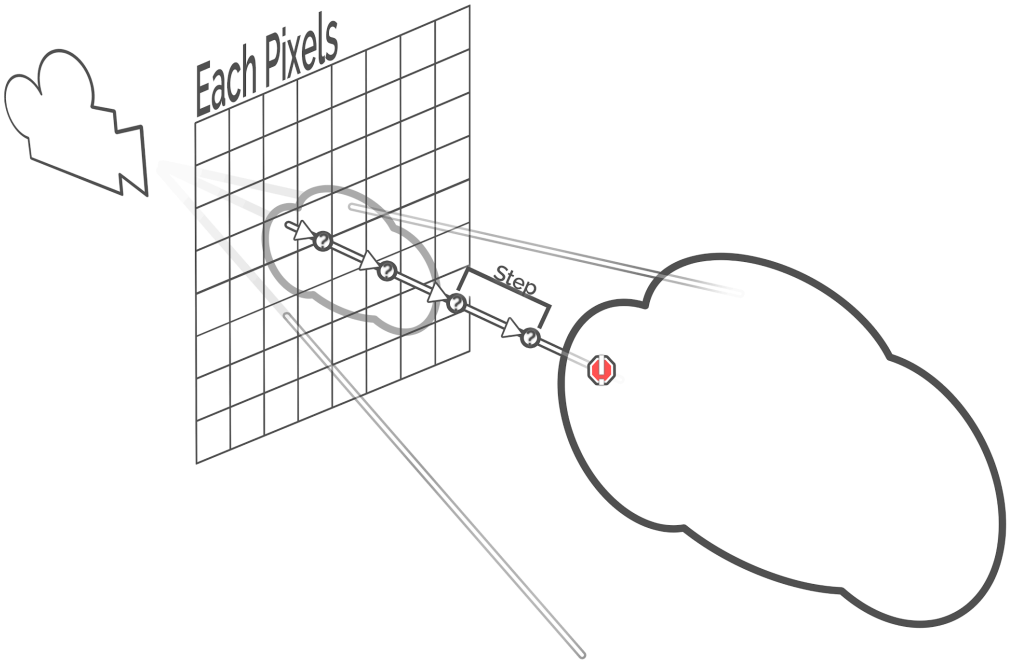
Debug

- *AlphaCut*
  Basic alpha cut, ask your favorite search engine for more information.
- *Render Queue*
  When should clouds be rendered in the rendering pipeline, see unity's manual on the subject for more information. Modify this value if clouds are rendered behind or on top of something it shouldn't.
- *Show Step Count*
  Render the amount of steps for each pixel on screen as a green to red gradient, the redder the pixel is, the closer it is to your 'Steps Max', and so the more demanding that pixel was overall.

You should use this toggle when trying to optimize this material. Don't forget that the amount of steps used is view dependent, it is pretty cheap to render from a top-down perspective but far less cheap when rendering from inside clouds.

// *Raymarching*

A raymarcher works by launching rays from the camera's point of view towards infinity and testing them against volumes, in this case, clouds. To do so, at each x unit along those rays we use a function to check whether we are inside or outside of a volume at that position, if we are inside we render clouds.
A step is the distance we travel between each check along the ray.



If the steps are too large the ray will skip past clouds and will appear like noise or cuts.
If the amount of steps is too low farther clouds won't be rendered.

// Tips

The two Speed variables shouldn’t have the same value, clouds won't be able to evolve through time if they are.

Draw distance too short ? Try increasing the variable Draw Distance to the max that you would like to render (distance is in unity unit, 1 to 1), read the paragraph on the raymarcher and skim through the description of the variables under the Raymarcher header.

// Optimisations

Everything under the raymarcher header changes performances a ton, play with those values and you should see significant changes.

All of the toggles influence performances as well as setting Direct Contribution or Sky Contribution to zero.

Direct3D 11(Dx11) and other more recent APIs should render them a LOT faster, consider using them in your project.

// Common issues

Clouds don’t show up :

Check that the variable “Material Used” on the “Raymarched Clouds” script is filled with your material.
You might have to enable depth map generation on your camera.

Clouds rendering blinks on and off :

Increase “Plane Offset” on the C# script until it doesn’t anymore.

Clouds render weirdly :

Skim through the chapter on the raymarcher.

“Blend out of shadowmap” doesn’t work/doesn’t work as expected :

You’ll have to decrease your render queue to 2500 or lower, unity removes the shadow mask from materials within that range.

The texture used with the shader needs four texture channels (RGB = Normals, A = Heightmap), do not mark your texture as a normal map and check if the compression used doesn’t discard one of those four channels.

If none of this helped then drop by the official thread on the Unity forum.