JK Volumetric Clouds for Unity

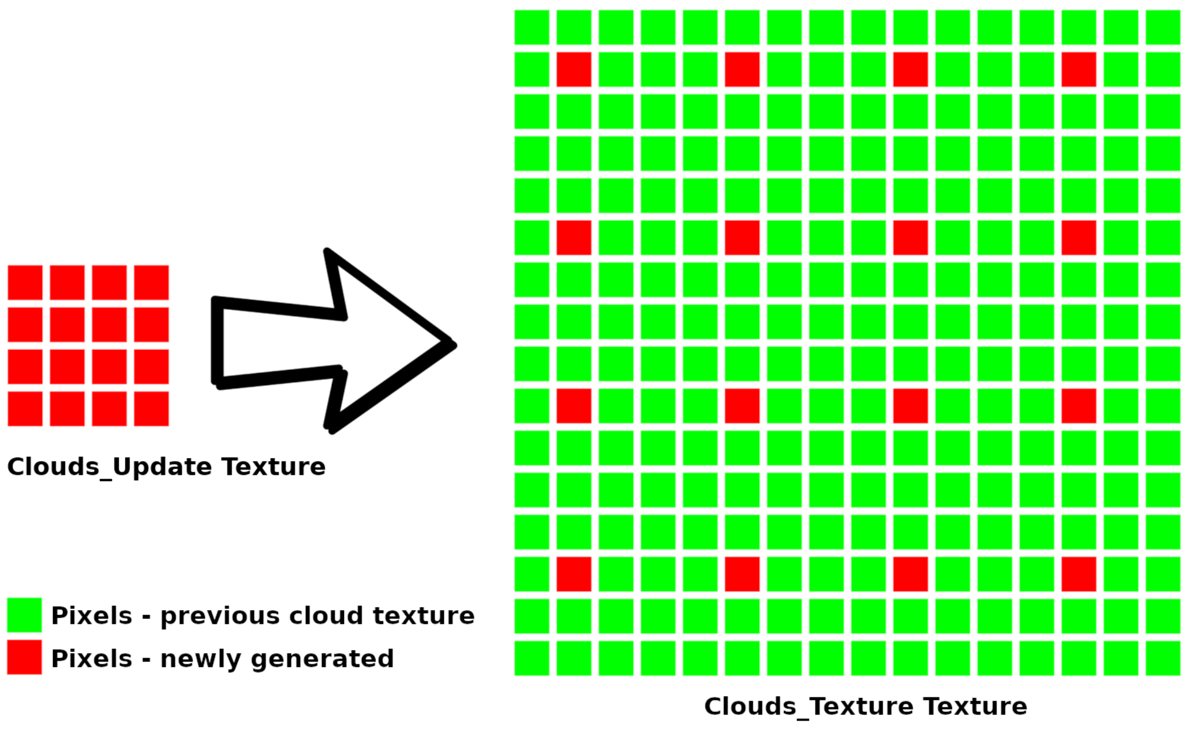
Documentation

Contents

Overview

Generating of the clouds is controlled by the Clouds_Creator.cs script. This script takes in all the parameters set by the user in the Sky.prefab. It then passes all the parameters to the relevant shader.

Rendering of the clouds itself takes place in the Clouds_Update.shader and Clouds_Texture.shader. The final output cloud texture is generated by the Clouds_Texture. Clouds_Update generates a sixteen times smaller texture used to update the cloud texture generated in the previous frame. This means that one-sixteenth of all the pixels change each frame.



(Pixels from the Clouds_Update generated texture are used to update 1/16 of the cloud texture generated by Clouds_Texture)

This texture is mapped on the skydome in the Skybox.shader over the standard Unity Procedural Skybox shader.

To get the amount of clouds at the specific coordinates in the sky Clouds_Update used three textures.

Cloud map is a 2D texture describing the amount and type of clouds at different places in the sky. It can be either supplied by the user or generated by Cloud_Map_Shader.shader.

3D texture generated by Noise_Shader.compute together with the cloud map describes the basic shape of the clouds

3D texture generated by the Small_Details_Shader.compute is used to add smaller details to the clouds.

Clouds_Creator.cs

The script that controls all of the cloud generation. It sets calls all the relevant shaders and sets up their properties.

Properties

public QUALITY quality

Modifies the resolution of the output texture. High = (1024*512), Medium = (512*256), Low = (256*128)

public float coverage

Float from 0.0f to 1.0f. Sets up initial coverage if you let the program generate a cloud map for you. Coverage describes the amount and thickness of the clouds.

public float coverageChangeTo

Float from 0.0 to 1.0. If you let the program generate the cloud map it defines the coverage in the newly appearing areas with clouds.

public Vector2 windDirection

Describes the direction the wind blows for the dynamic generation of the cloud map.

public float windSpeed

The speed of the wind for the dynamic generation of the cloud map. The higher the speed, the faster the clouds move on the sky. Do not set this value higher than 20.

public Color atmosphereTint

Modifies the color of the atmosphere.

public color groundColor

Color, of the ground in the skybox.

public color sunTint

Color of the Sun dot. It doesn’t modify the color of the sunlight itself.

public floatHDRExposure

The output color from the skybox is multiplied by this value. Higher values make the sky appear lighter.

public Color ambientColorTop

Environmental light in the higher levels of the clouds.

public Color ambientColorBottom

Environmental light in the lower levels of the clouds.

public Texture cloudMap

It can be used by the user to input a custom cloud map.

public bool generateCloudMap

Tells the program if it should generate its own cloud map

private ComputeShader noiseShader

Compute shader used to generate a 3D texture for basic cloud shapes.

private ComputeShader noiseShaderDetail

Compute shader used to generate a 3D texture used to add details to the clouds.

private RenderTexture noiseTexture_3D

A 3D texture used for basic cloud shapes generated by the noiseShader.

private RenderTexture noiseTexture_3D

A 3D texture generated by the noiseShaderDetail, used to add details to the clouds.

private RenderTexture skyTextureUpdate

A 2D texture used to update the skyTexture

private RenderTexture skyTexture

A 2D texture with clouds, that can be mapped on the skydome.

private Material updateTextureMaterial

A material used to generate the skyTextureUpdate for the current frame.

private Material skyTextureMaterial

A material used to generate the skyTexture.

private Material cloudMapMaterial

Material, that generates a dynamic cloud map.

private RenderTexture generatedCloudMap

Cloud map generated by the cloudMapMaterial.

private int frameNumber

The number of the frame, tells the program which pixels should be updated this frame.

private float lastUpdateTime

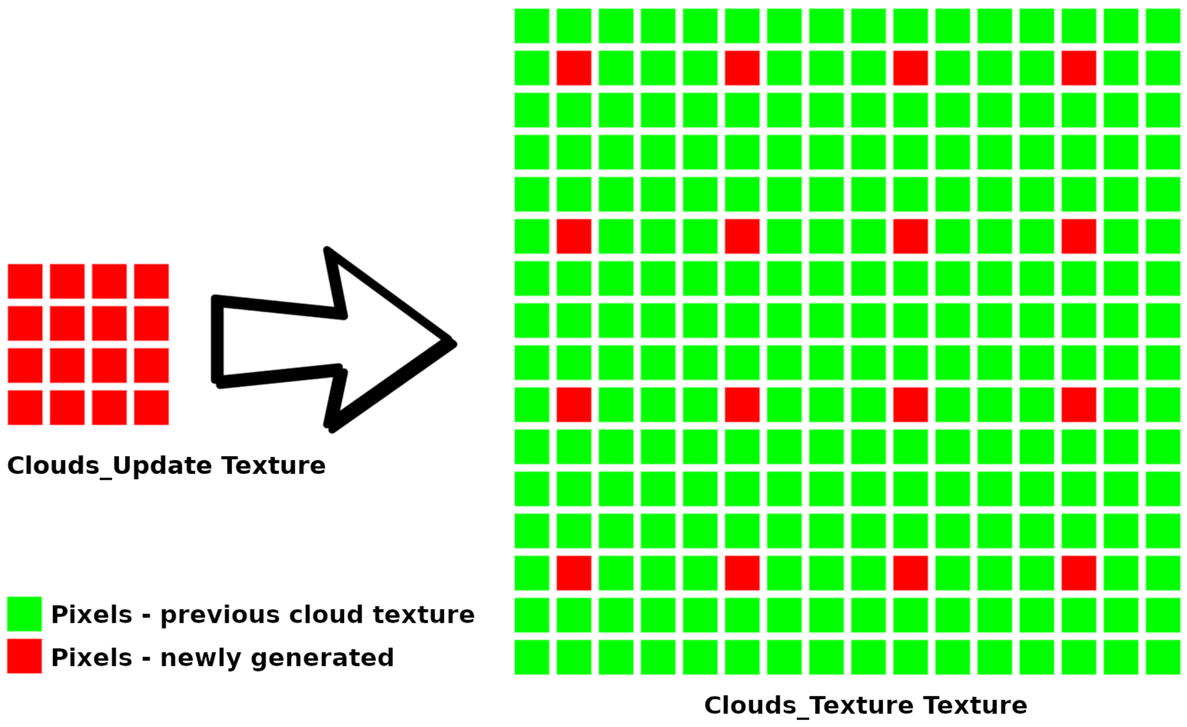
Time of the last update.

Functions

private void updateTexture()

Renders a new update texture using Clouds_Update, and then uses it to update the sky texture in Clouds_Texture. In the end, it increases the frame number by one.

Only one-sixteenth of the sky texture is updated each frame (which pixels are updated changes based on the frame number)



private void cloudMapGenerate(int first)

This function is used to call the shader generating the cloud map for the current frame. The parameter first is used to tell the shader, if this is the first time it is used. In the Start() function the value is set to 1 – the cloud map is initialized. In the Update() it is 0 – the shader will use previously rendered cloud map.

Functions also updates all necessary shader properties (like the amount the clouds shifted).

internal void ChangeSettings(int qual, float cover)

This function was used for debugging purposes to change the quality and coverage of the sky at runtime.

internal void ChangeCoverage(float cover)

Was used for debugging purposes to change the sky coverage at runtime.

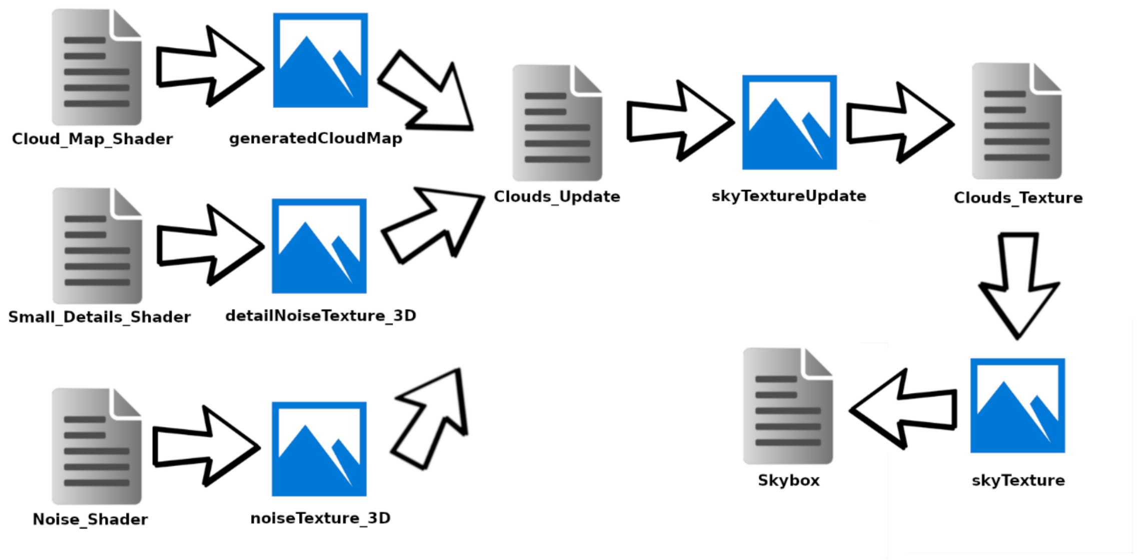
Start()

The function called by Unity when the program starts. It initializes the properties of the materials and creates textures program should render to.

It calls two compute shaders noiseShader and detailNoiseShader and generates two 3D textures.

If the generateCloudMap is set to true cloudMapGenerate is called, otherwise cloudMap property is used.

In the end, updateTexture() is called 16 times to update all the pixels in the sky texture.



(JK Volumetric Clouds workflow)

Update()

Called every frame, when the game is running, or when editor settings change.

If the cause for the call is a change in the editor Update() calls Start() function. This initializes the sky again with the new properties.

Otherwise, the function calls updateTexture() function and if the generateCloudMap variable is set to true also cloudMapGenerate(0) (0 tells shader, that this is not the first time cloud map is generated).

Noise_Shader.compute

Shader used to generate a 3D texture that defines basic cloud shapes.

The texture consists of channels each with different Worley noise frequency stored in it.

Small_Details_Shader.compute

Similar to Noise_Shader.compute. It generates a 3D texture used to add small details to the cloud shapes. The texture has different frequencies of Worley noise stored in its channels.

Cloud_Map_Shader.shader

A shader that generates a dynamic cloud map.

Properties

2D _MainTex

Texture with the cloud map rendered in the previous frame

float _Coverage

Initial sky coverage (only affects the initial frame)

float _CoverageChangeTo

Cloud coverage in the newly appearing areas in the sky

Vector _PositionDirection

X and y coordinates describe the amount the clouds have shifted in a certain direction. Z and w coordinates store the direction the wind is blowing.

float _Speed

Speed the wind is blowing.

float _DeltaTime

Time since the last frame.

int _First

Tells the shader, if this is its initial frame. I fit is true the value passed in the variable is 1 otherwise it is 0.

Functions

Float noise(float2 pos, int mul)

Generates a Perlin noise with one octave at specified coordinates.

float perlin(float2 pos, int mult)

It generates a Perlin noise with 8 octaves.

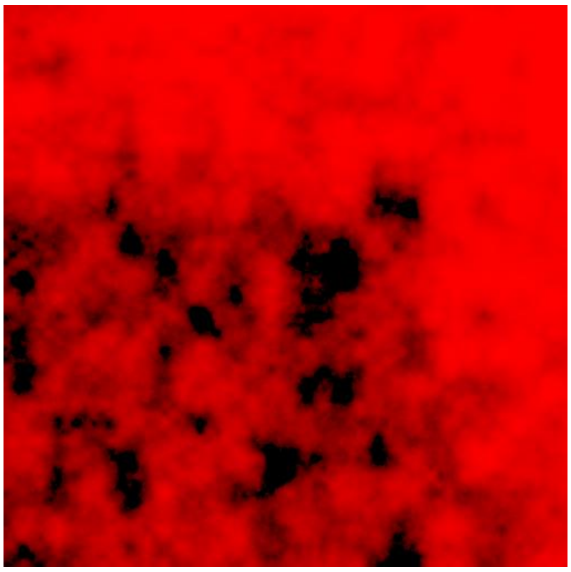
fixed4 frag (v2f i)

Fragment shader returns the value, that will be stored at specific coordinates in the cloud map.

The red channel describes the amount of cumulus clouds, the alpha channel stores the coverage. In the first frame the value of the red channel is generated by the code shown below (it is a modified Perlin noise).

```
float perlinValue = clamp(perlin((uv) * 16, 16) + (0.8*_Coverage - 0.5), 0., 1.);
return float4(pow(perlinValue, pow(2 - 2*perlinValue, 1)), 0., 0., _Coverage);
```

If it isn't the first frame, the coverage values are taken from the previous cloud map. If the _Speed property is set to anything other than 0, there was a shift in the position of the clouds between this and the previous frame. Shader makes sure, that it reads the value from the position the cloud was in the previous frame. If this read is very close to the edge of the cloud map, the channel value is slightly changed towards the _CoverageChangeTo value. This makes the cloud map slowly change towards the desired coverage. The coverage first changes in the direction the wind blows from.


(cloud map generated by the shader, the coverage is increasing in the top right)

Clouds_Update.shader

Properties

Color _Clouds_Ambient_Top

Ambient color in the higher levels of the clouds.

Color _Clouds_Ambient_Bottom

Ambient color in the lower levels of the clouds.

3D _3dTexture

3D noise texture used by the shader to define basic cloud shapes.

3D _3dTexture_Distort

3D noise texture used to add smaller details to the clouds.

2D _Cloud_Map

2D texture describing the presence and type of clouds in different parts of the sky.

Float _Density

The density of the outputted clouds. Denser clouds are darker and their borders are more pronounced. Currently, this property can not be modified by the user.

Vector _PositionDirection

The first two vector coordinates define the distance the clouds have shifted since the first frame. The second two coordinates store the current direction of the wind.

float _Speed

The current speed of the wind.

Int _Frame_Number

The current frame number tells shader which pixels are being updated this frame.

Important Constants

CLOUD_MARCH_STEPS

The number of march steps the program takes. A lower number makes cloud rendering much faster but results in some loss in quality.

CLOUD_SELF_SHADOW_STEPS

The number of self shadow steps performed by the shader. Lowering the number significantly speeds up the program, but results in less detailed shadows.

CLOUDS_SHADOW_MARCH_STEP_SIZE,
CLOUDS_SHADOW_MARCH_STEP_MULTIPLY

As the shader takes self shadow steps towards the Sun the length of these steps changes. The initial step has the length defined in the CLOUDS_SHADOW_MARCH_STEP_SIZE constant. The length of each following step is multiplied by the CLOUDS_SHADOW_MARCH_STEP_MULTIPLY making each step longer than the previous one.

CLOUDS_BOTTOM, CLOUDS_TOP

Defines bounds of the space clouds are rendered in. These bounds take a form of planes, CLOUDS_BOTTOM defines the distance of the lower plane from the ground, CLOUDS_TOP distance of the higher plane.

CLOUDS_DETAIL_STRENGTH

Modifies the strength of the cloud detail obtained using _3dTexture_Distort.

Functions

float HenyeyGreenstein(float sundotrd, float g)

Calculates the Henyey-Greenstein phase function. Makes clouds closer to the Sun dot lighter.

Float sundotrd – cosine between the direction to the rendered part of the cloud and direction to the Sun.

Float g – constant with the anisotropy factor for the function.

float intersectClouds(float3 rd, float r)

Finds the distance to the intersection with a plane in a distance r from the ground, with a ray in direction rd.

float cumulusGradient(float norY)

Gradient function for cumulus clouds. Input parameter norY should be between 0 and 1. It tells the function height of the sampled point relative to the cloud space.

float stratusGradient(float norY)

Gradient function for stratus clouds. Input parameter norY should be between 0 and 1. It tells the function height of the sampled point relative to the cloud space.

float cloudMapDetail(float3 p)

Sampling of the _3dTexture_Distort at world space coordinates p. Clouds shifting over time is taken into account.

float cloudMapBase(float3 p, inout float3 detail)

The function samples the _3dTexture for basic cloud shapes, cloud shift over time is taken into account. It returns the basic cloud density for cumulus clouds at position p. In the output parameter detail, three channels with a higher frequency that can be used for stratus clouds.

float cloudMap(float3 pos, float dist)

The function samples the clouds at position pos for the raymarching. Parameter dist tells function the distance of the sampled point from the viewer.

The function takes a sample from the cloudMapBase. Then it is multiplied by the value from the cloud gradient (so the clouds appear at correct heights).

This value is then further multiplied by the _Cloud_Map texture, so the clouds only appear at specified places.

In the end, value is subtracted (so clouds do not appear everywhere), the detail is added, the value is multiplied by the _Density and it is returned.

float cloudMapShadow(float3 pos)

Function similar to the cloudMap used for the sampling for self-shadowing. Clouds are only made more uniform and bigger.

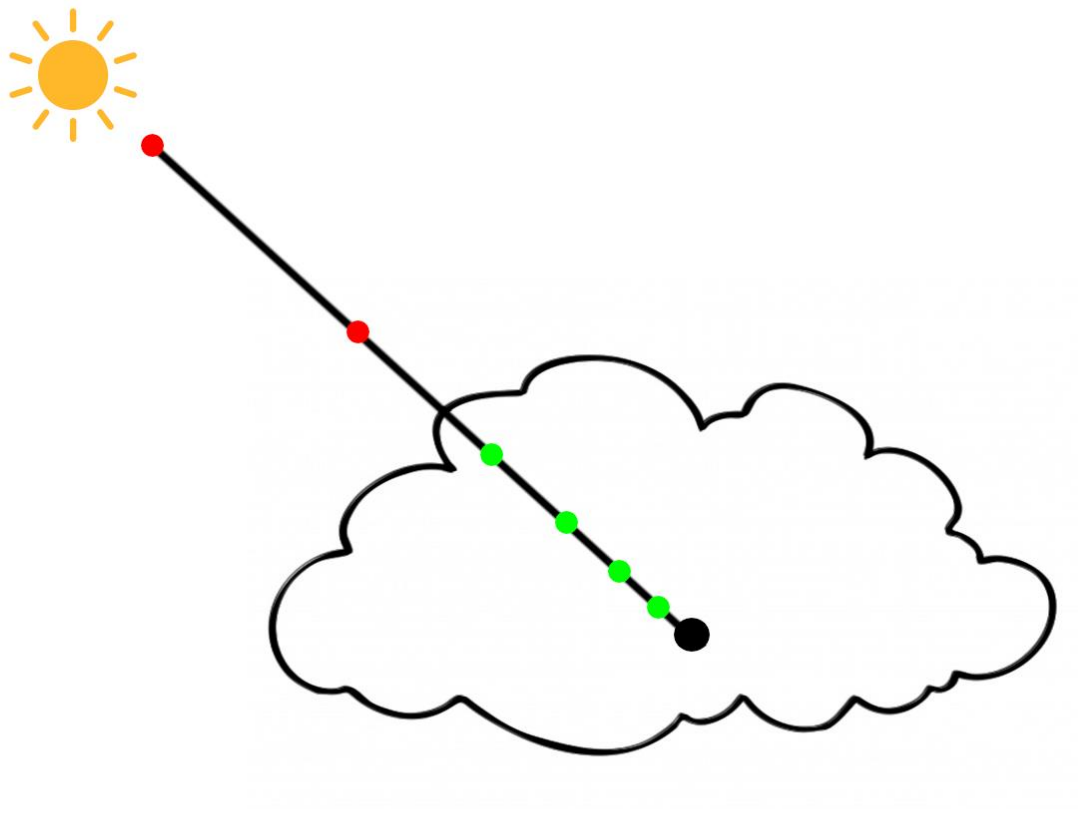
float volumetricShadow(in float3 from)

Calculates self-shadowing of the point specified by the from property.

The function takes samples in the direction towards the Sun. The first step size is defined by CLOUDS_SHADOW_MARCH_STEP_SIZE and the length of each new step size is multiplied by CLOUDS_SHADOW_MARCH_STEP_MULTIPLY.

With each step, the clouds are sampled using cloudMapShadow function. Based on the cloud density and step size the lighting received from the Sun is reduced.

Lighting is also modified by the Sun angle. If the angle is steep, the clouds receive less light.



(Self-shadowing algorithm takes steps of increasing lengths towards Sun and test if there is a cloud shadowing the origin point)

float4 clouds(float3 ro, float3 rd)

The main ray marching function. Ro describes the observer position, rd the direction of the sample.

The output of the function is the color and transparency of the clouds in the sample direction.

The function takes samples with steps of constant size in the direction rd. If the sample returns, that there is a cloud (alpha>0), the contribution of this sample to the final color is computed.

The color of the sample consists of the ambient light and sunlight modified by the self-shadowing and the Henyey-Greenstein phase function.

fixed4 frag(v2f i)

The function takes in the uv coordinates of the update texture. These coordinates are transformed into the coordinates of the sky texture. Each frame the sky texture coordinates change, because only one-sixteenth of them is rendered each frame. Every pixel in the update texture corresponds to 16 pixels in the sky texture. The order of the updates is described in the table on the left.

1	9	3	11
13	5	15	7
4	12	2	10
16	8	14	6

Once the uv of the updated in the sky texture is obtained the program transforms it to the corresponding ray direction.

The program calls the clouds function with the ray direction received from the sky texture. The ray origin is set to be the camera position. The output of this function is then tone mapped and returned.

Clouds_Texture.shader

The shader updates the sky texture using the update texture generated by the Clouds_Update.shader.

Only one-sixteenth of the pixels is updated each frame. These pixels are selected based on the

frame number and their pixel coordinates modulo 4. The table to the left describes the order the pixels are updated in.

1	9	3	11
13	5	15	7
4	12	2	10
16	8	14	6

Skybox.shader

This shader generates the standard Unity Procedural Skybox shader and maps the sky texture on it.

When the standard skybox is rendered the shader samples the sky texture for the clouds. Depending on the distance of the clouds from the viewer, a certain amount of fog is added over the clouds. The fog has the color of the sky without the Sun or clouds.

The sampled clouds with fog over them are added over the sky.