**Real-Time Volumetric Cloud Rendering with Ray-Marching Technique**

Yun Jiang, Victor Shu

Final Project Proposal for IGME 797.01

**Abstraction**

There are several common techniques that are used in cloud rendering. Despite the easiest one that bakes the cloud into the skybox texture, the most straightforward one is asset-based, which is very simple: create 3D models for the clouds and place them into the scene and render like a common object. However, this simple technique has its own limitations: it’s not compatible with dynamic skyboxes where the color of the sky will affect the color of the light scattered in the cloud, and it’s very hard to make the cloud change its shape during runtime. It’s also not possible for the camera to be inside the cloud since it’s an actual 3D model. Based on these limitations, a new technique called ray-marching is used in cloud rendering.

**Related Works**

There have already been many AAA games that use this approach for rendering clouds. *Horizon Zero Dawn* from Guerrilla Games is an open-world game with dynamic weather and day/night cycle. Their cloud renders in about 2ms and takes 20MB of RAM, and it responds to the color of the night and the weather. *Ace Combat 7: Skies Unknown* from *Bandai Namco* also uses the same approach, as the player who flies a fighter jet can actually go into the cloud. The game runs in 4K, 60FPS on a mid-end machine. We decided to implement a ray-marching based cloud rendering as our final project for IGME 797.01.

**Ray-Marching**

Volume ray-marching is an image-based volume rendering technique. It computes 2D images from 3D volumetric data sets. Different from ray-casting in the sense used in ray-tracing, which processes surface data, the computation of ray-marching doesn’t stop at the surface but pushes through the object, sampling the object along the ray. Ray-marching doesn’t spawn secondary rays, unlike ray-tracing.

A basic ray-marching algorithm is made up of four parts:

1. Ray casting. For each pixel of the final image, a ray is cast through the volume. The volume, in this stage, can be considered as a box shape just for intersecting the ray.
2. Sampling. Along the part of the ray within the volume, several evenly distributed sampling points are selected.
3. Shading. For each sampling point, a transfer function retrieves a color and a gradient of illumination values are computed. The gradient represents the orientation of local surfaces within the volume. The samples are then shaded according to their surface orientation and location of the light source.
4. Compositing. After all sampling points have been shaded, they are composited along the ray, and the final color value for the pixel will be calculated.

In a volumetric cloud rendering situation, the volume will be the cloud itself. The volume data can be stored in a set of 2D images, or it can be generated with different noise textures. We will further explore the options to generate cloud data.

**Technologies**

We will use the Unity3D Engine to implement the algorithm.

**References**

1. *The Real-time Volumetric Cloudscapes of HORIZON ZERO DAWN* from *Guerrilla Games*   
   <https://www.guerrilla-games.com/read/the-real-time-volumetric-cloudscapes-of-horizon-zero-dawn>
2. *Physically Based Sky, Atmosphere and Cloud Rendering in Frostbite* from *EA, Frostbite*<https://www.ea.com/frostbite/news/physically-based-sky-atmosphere-and-cloud-rendering>  
   <https://media.contentapi.ea.com/content/dam/eacom/frostbite/files/s2016-pbs-frostbite-sky-clouds-new.pdf>



Cloud in *Horizon Zero Dawn*



Cloud in *Ace Combat 7: Skies Unknown*



Cloud in *Battlefield 4*