Imposter Generation for Terrain Backgrounds

Bill Hess

1 Introduction

Traditionally, computer games have had closed off, indoor settings. In classic games such as Doom and Descent, players would only be able to observe a subset of rooms at one time within the playable area of the game. This allowed game developers to make drastic optimizations to the way graphics were displayed in their game engines.

More modern games have commonly taken on a different setting: outdoors. The player can now observe not only rooms and objects in the foreground, but see far off objects in the distance. A common technique for overcoming this leap in viewable scenery is to render the background scenes, where the player cannot explore, ahead of time onto a static image and simply display that image behind the foreground area. Since the player cannot get close to objects in the background, they only need to be rendered once. This technique is called Skyboxing.

This optimization cannot be applied to all outdoor games. In the genre of Open World games players are given large virtual worlds to explore. In these games there is no distinction between which parts of the game world are in the foreground and which are in the background. For example a player can see a mountain far off in the distance, and travel to and climb to the top of that mountain. This means that the mountain cannot be rendered onto a Skybox since the combinations of viewing angles and distances is vast.

If players are to be able to see far off objects in Open World games, they cannot be rendered the same way as objects in the foreground. Due to the perspective transformation, the amount of geometry required to display increases with the square of depth of the scene. The cost of rendering a scene diverges as the maximum viewing distance increases. In order to overcome this problem, the detail of backgrounds is dramatically lowered.

These background scenes in Open World games are limited by the depth buffer as well. Lower level of detail meshes can simplify the problem of rendering too much geometry, but the vast distances between objects in the background forces the graphics hardware to lower the precision of the depth buffer. This lack of precision causes graphical artifacts on objects in the foreground.

There is a rendering technique related to Skyboxing called impostors. Impostors are sets of complex geometry rendered onto a flat image called a billboard. This billboarded image always faces the camera so that the camera cannot detect that it is a flattened image. An Impostor has the limitation that if the viewing angle changes too much, it becomes invalid and is no longer a proper representation of the geometry it has replaced.
My technique will be a combination of these two optimizations. I will generate Impostors for terrain geometry that appears beyond the foreground. The Impostors will be faces of a cube much like what is used in the Skybox technique. The Impostors will be sampled at varying depths. Since the depths of the impostors are known ahead of time, these images can be layered on top of each other without the use of the depth buffer. As the player moves, the Impostor background images can be moved independently, giving the impression of relative distances.

Since this technique does not use the depth buffer, its full precision can be used for objects in the foreground where it is better utilized. This technique also allows only foreground objects to be rendered in real time, simplifying the vast amount of geometry in the background. Furthermore, since background geometry does not need to be sampled at as high of a rate, backgrounds can be displayed in high detail.

There are a few potential limitations of this technique. First, since objects are rendered onto flat images, they cannot easily be animated and certainly cannot be interactive. Second, as the viewing angle and position changes, new Impostors must be drawn which may draw rendering power away from foreground objects.

There are several reasons why I believe this is still an appropriate optimization to make. First, in Open World games, player movement is limited to a known maximum. As long as the time it takes to redraw the necessary Impostors is less than the shortest time that a player can create a noticeable view change, then the effect will not be observable. Second, since objects in backgrounds are typically static, I feel it is appropriate to make static Impostors for background geometry.

This technique will pose challenges which I will have to solve to successfully implement this technique. I will need to be able to transition between one Impostor to the next for the same set of background geometry without the image noticeably “popping”. There must also be a transition between geometry in an Impostor and geometry that appears in the foreground. I will also have to prevent large pauses from occurring while an Impostor is rendered, which would be undesirable in a real-time Open World game. This thesis will explore techniques to solve these problems.

I will also perform a comparison and analysis of performance data between the implementation developed here and others currently used for Open World games. For the work to be successful, the technique needs to be able to greatly extend the viewable area while maintaining real-time speeds. The average time taken to render a frame should increase by no more than 50%.
References

