1. Introduction

Tessellation is one of the main features of most recent Direct3D 11 API. It enables dicing of incoming geometry on the fly and gives developer control over newly created vertices. If displacement map is available, one can displace those vertices to create rich adaptively generated content in real-time. Until recently just a few games made use of displacement maps. Bump mapping with normal maps is used much wider. Weak point of bump mapping is lack of silhouettes, occlusion, and shadows, which reveals the coarseness of geometry. However, process of converting existing content to use displacement mapping is very long and expensive, and so few game developers risk significant budgets to perform such endeavor without guaranteed investment return. We propose a relatively inexpensive solution of automatic conversion of normal maps to displacement maps.

2. Exposition

Receiving normal map from displacement map is an easy task. Normal at every surface point is perpendicular to displacement map gradient. Inverse problem requires integration, which in general is never guaranteed to yield precise result because:

- The information about surface discontinuities is lost in normal map;
- Normal maps are usually lossy-compressed;

Because of those problems we do not hope to reconstruct displacement exactly, but instead target visually-plausible result.

First step of conversion is creating of Depth Difference Map (DDM). This is float2 texture map where X channel shows how much does the depth change when traversing a pixel in horizontal direction and Y channel shows the same for vertical direction. 1D example of DDM from normal map computation is shown below.

![Depth Difference Map](image)

**Figure 1.** Depth difference is signed value computed from normal map

In order to compute displacement of one texel in respect to another, one has to integrate DDM along the curve connecting those texels. Any curve can be chosen in principle. We use straight lines. DDM, as well as the normal map it is produced from, contains significant errors. To suppress the error and get nicely looking result, we need some heuristic assumptions. We are making two:

- Integral of displacement map should be 0;
- Whatever integration paths we chose, they should not prioritize one texel over another or treat one integration direction over another;

The resulting algorithm executes the same procedure for every texel and works completely on the GPU. Starting from the current sample position we trace a number of 2D rays, uniformly distributed over 360 degrees.

We integrate DDM along each ray and find approximate depth integral (denoted as AvgD) inside the circle, assuming that depth we started from was zero. Now to make the integral zero, we need to displace central texel by −AvgD. This value is recorded to displacement map.

Conversion process can be done offline. In general we found radiuses of 250 texels with 500 directions to be enough for most textures being commonly used.

If normal map consists of several pieces (atlas), those pieces should not affect each other, and generated displacement map should be continuous across atlas boundaries to avoid cracks in displaced geometry. We address those issues by breaking integration process on the boundaries and performing extra steps to fix the seams.

References

