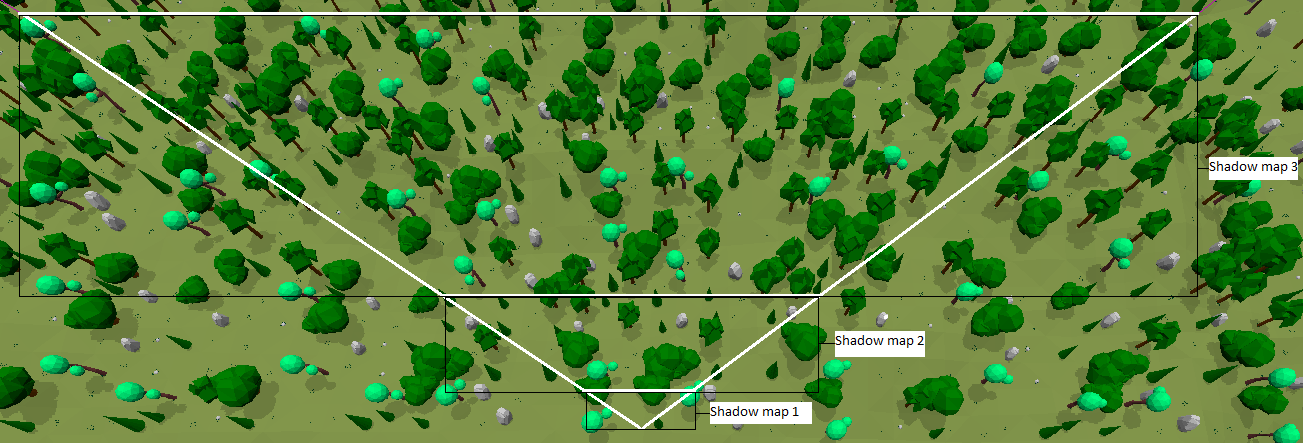
**Dynamic Shadows in Thieves in the Night**



**Introduction**

*Thieves in the Night* uses Cascaded Shadow Maps (CSMs) to shadow outdoor areas. The algorithm splits the camera’s view frustum into several subfrustums, and applies a shadow map to each separately. This helps reduce perspective aliasing by providing higher shadow map resolutions to the subfrustums near the camera. The following image show how shadow maps are applied to a view frustum that has been split into 3 sections. Note how there is higher shadow map resolution for the subfrustum near the camera due to covering a smaller area:



Shadow mapping can be broken up into **Shadow Map Generation** and **Shadow Projection.**

**Shadow Map Generation**

For each shadow map, the renderer determines which objects cast shadows on the affected area. Shadow culling is currently unimplemented in *Thieves in the Night*, however an octree can coarsely remove many shadow casters. The remaining ones can be tested directly against the camera’s view frustum planes which we extract from the camera’s view-projection matrix[1]. These planes can be used to form a collision mesh for each subfrustum providing a tighter culling test.

The final set of shadow casters are rendered into the shadow map by transforming them by light’s view-projection matrix. A LookAt(*at*, *from*, *up*) style camera function is used to generate the light’s view matrix. *at* is the center of the frustum section that is being shadowed. *From = at + -LightDirection\* frustumExtent*. This view matrix is then multiplied by an orthographic projection matrix.



The above image shows a scene that is split up into 4 shadow maps. For convenience, a single 2048x2048 shadow map is partitioned into 4 sections and each split’s shadow was rendered to one of them. The projection step is a single, full screen pass where the pixel shader picks the correct partition to sample for the current pixel. Memory, performance, and visual improvements may be achieved by using a single 1024x1024 shadow map and projecting the shadows onto each split individually.

*Shadow masking*

*Thieves in The Night* combines terrain and environment objects into a single renderable mesh. However, nice looking terrain shadows proved to be difficult due to severe projective aliasing[3], so the terrain is "masked out" of the shadow depth shader leaving only trees, rocks, etc to cast shadows. This masking is done by setting the w-component of each terrain's vertex color to 0. The shadow depth vertex shader moves these verts behind the shadow camera's near clip plane which culls the triangle from further graphics pipeline processing.

*Note on performance*

An object may cast shadows into multiple subfrustums causing them to be rendered multiple times. However, shadow map generation typically involve simple shaders since we’re only concerned with writing out the depth of pixels. Also, if your hardware supports reading from depth textures (in the projection step), then depth-only writes can be used for shadow generation which gives a nice performance boost[2]. Sensible shadow map sizes can help as well.

**Shadow Projection**

Next, the rendering engine draws a single full-screen quad to project the shadows onto the world. We use the depth and screen position at each pixel to reconstruct its world space position. Then we transform the world position into light space by multiplying it by the light’s view-projection matrix. The coordinates are then transformed from homogenous clip space to texture space (ex, [0,0], [1,1]).

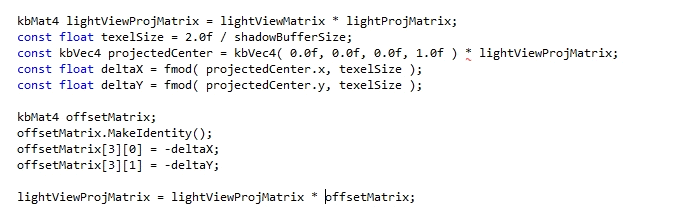
Because the subfrustums' shadow maps each take up a quarter of a single texture, the coordinates must also be scaled by 0.5. And since the projection is performed in a single pass, the coordinates must be shifted to the proper shadow map (starting at (0,0), (0.5, 0.0), (0.0,0.5f), or (0.5f,0.5f). This is done by finding which subfrustum the current pixel is in (by comparing its distance against the shadow split distances) and then adding the appropriate offset to the coordinates.

Now the engine samples the shadow map using the xy values of our projected value. If the shadow map value is greater than the z component of our projected value, the pixel is in shadow.

*Swimming edges*

The edges of view dependent dynamic shadows can “swim” as the camera moves. As a particular shadow moves across the shadow map, its edges slide across the texels and flicker. This can be fixed by keeping the texels aligned with world coordinates so that shadow edges no longer slide across texels.

kbEngine achieves this by first projecting the zero vector (0.0f, 0.0f, 0.0f) into the light’s view-projection matrix. The xy coordinates are divided by the size of a shadow map texel in clip space (2.0f/ShadowMapWidth). The remainder gives the delta between the zero-vector and the nearest texel. The negative of these deltas are put into a translation matrix and concatenated onto the light-view projection matrix. If the zero-vector is transformed by this matrix, it will always wind up on a texel, thus keeping the shadow map aligned with the world. The code for this follows:



**Results**

Without any culling or in-depth optimizations, the kbEngine currently renders the world (no characters) at 250 frames per second (4ms) on an AMD FX-8320 Eight Core Processor with a Nvidia 760 GTX. The CPU spends < 0.5ms rendering lighting while the GPU has not been profiled yet. A typical lighting pass may consist of 272+ draw calls and 900,000+ triangles. Adding a basic culling system would significantly reduce the number draw calls and triangle counts. Rendering optimizations are planned for a future update.

**References**

1. Gil Gribb; Klaus Hartman. “[Fast Extraction of Viewing Frustum Planes from the World-View-Projection Matrix”](http://gamedevs.org/uploads/fast-extraction-viewing-frustum-planes-from-world-view-projection-matrix.pdf)
2. Cem Cebenoyan. “[Graphics Pipeline Performance”](http://http.developer.nvidia.com/GPUGems/gpugems_ch28.html). GPU Gems 2
3. Windows Dev Center. ["Common Techniques to Improve Shadow Depth Maps"](https://msdn.microsoft.com/en-us/library/windows/desktop/ee416324(v=vs.85).aspx)