4.4 Facetted Shadow Mapping for Large Dynamic Game Environments

RAY TRAN

INTRODUCTION

Shadows play an important role in providing a convincing representation of the game world to the game player, and with the current generation of game consoles and the GPUs they employ, rendering soft dynamic shadows even in large dynamic game environments has become possible.

This article presents a novel shadowing algorithm that was used in the game *Grand Theft Auto IV* (*GTA4*). In the following section we discuss the challenges posed when developing a shadow system in large dynamic game environments and describe how our approach addressed these challenges. We then review existing shadowing techniques that have influenced the development of our new approach and show how our approach overcomes some issues with these techniques. A detailed description of our approach, including source code examples, is then provided, showing how to construct facetted shadow maps and how to use them when applying shadows to a scene.

THE CHALLENGES

In this section we describe some of the challenges faced when developing a shadow system for a game environment and explain the motivation behind some of the choices made in developing the facetted shadow approach. It should be noted that facetted shadow maps are suitable for light sources that can be simulated as global directional lights in a game environment, such as the sun or the moon. When choosing a shadow approach for a game, you need to consider the requirements of your scene, and for *GTA4* these were:

- **Dynamic time of day:** The relative sun position would continuously change as time passed in the game world, starting in the east at dawn and swinging around to the west at dusk. Precalculating shadows for a continuously changing sun was therefore not possible without producing a large amount of data.

- **Consistent shadowing:** To produce a convincing environment, our aim was to have all objects cast and receive shadows using a consistent approach. It was important for all shadows to match the originating light source.

- **High-detail soft shadows:** Our aim was to produce highly detailed shadows without harsh edges.

- **Low memory usage:** Fitting the dense game world into memory is a real challenge. A shadow system that has low memory usage was required.

- **Low streaming bandwidth:** It is often impossible for large games such as *GTA4* to fit entirely in the fixed memory available on a game console, so you have to stream data in and out from a secondary storage device. A shadow system that has a small impact on streaming bandwidth was desirable.

The development of the facetted approach allowed us to meet these requirements in *GTA4*: “The real-time shadows are working across every object and surface in the game with everything self-shadowing and casting onto everything else” (Aaron Garbut, Art Director, Rockstar North) [Bramwell08].

EXISTING SHADOW MAP APPROACHES

There have been many adaptations of the original shadow map approach presented in [Williams78]. We will focus on approaches that can be implemented on current console hardware and have influenced the development of the facetted approach—broadly divided into two categories: warped shadow maps and split shadow maps.

- **Warped shadow maps:** These include perspective shadow maps [Stamminger02] and several variants of perspective shadow maps including light space perspective shadow maps [Wimmer04] and trapezoid shadow maps [Martin04]. These warping techniques

use a special projection matrix to warp the shadow map so that the important parts, close to the viewpoint of the scene, appear larger in the shadow map and therefore reduce aliasing artifacts. Perspective shadow maps suffer from *dueling frusta* problems, which occur when a certain range of light directions and view directions prevent a good projection matrix from being formulated. This problem has largely prevented their wide use in games that have varying light directions and unconstrained camera directions.

- **Split shadow maps:** These include cascaded shadow maps [Engel06] and a similar technique called parallel-split shadow maps [Zhang07]. These splitting techniques divide up the view frustum into several parts, rendering each part as a separate map or tile. By varying the size of each tile or the area that each tile covers, it is possible to render more detail into tiles closer to the view point and reduce aliasing in those tiles. Split shadow maps are widely used in games, as they do not suffer from dueling frusta problems and there are efficient methods to correctly choose which of the tiles should be used when applying shadows. Split shadow maps suffer from an artifact that can be seen on the border between tiles where there is a differing texel density.

The facetted shadow map approach uses both warping and splitting to reduce aliasing. The key advantage over other split shadow maps is that it is possible to apply a filter across the splits, but it does not suffer from dueling frusta issues.

Facetted Shadow Map Approach

In a facetted shadow map the scene is split into several equal-sized parts or facets (Figure 4.4.1). All facets sit on the same map, and the arrangement of each facet within the map allows filtering from one facet to an adjacent facet, avoiding the visible artifact found in other split shadow maps. Each facet has an individual projection matrix formulated to warp the objects in that facet, which is used when creating the facet and when using the facet to apply shadows.

FIGURE 4.4.1 These pictures show possible facet arrangements of four, five, and eight facets.

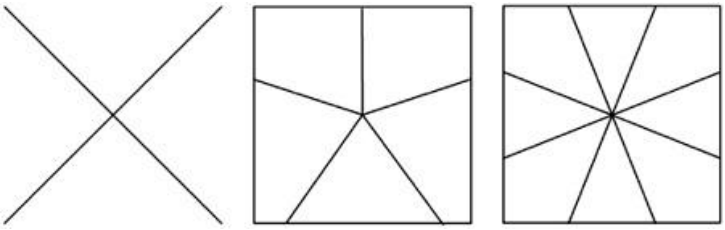
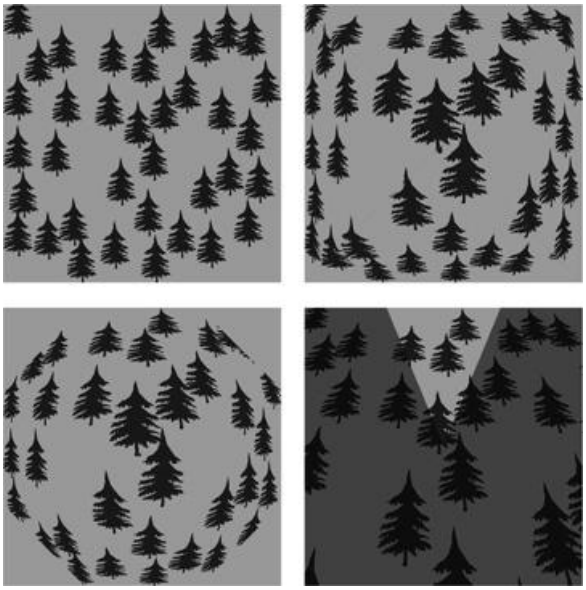


FIGURE 4.4.2 (Top left) A standard shadow map. (Top right) A facetted shadow map with four facets. (Bottom left) An eight-facet facetted shadow map. (Bottom right) An isolated facet from an eight-facet arrangement. The dark area would normally be clipped because other facets making up the shadow map would occupy this area.



Notice in Figure 4.4.2 that the trees in the center of the facetted shadow maps are bigger than they would appear in the standard shadow map and thus have more shadow information close to the center of the map, reducing aliasing where the viewpoint would be located. Also note that when trees fall into more than one facet, the edges of the facets line up even though the warp direction is different for each facet. This allows filtering across facet boundaries.

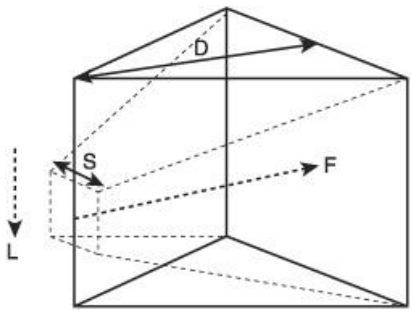
Creating AND Using Facetted Shadow Maps

The formulation of the perspective projection matrix for each of the facets depends on the size of the region in the scene in which you want to cast shadows and the desired detail at the center of the map. Three values are required to create a facet projection matrix equation: **F**, *D,* and *S,* where:

- **F** is the facet direction—a vector in the direction of the perspective warp.
- *D* is the distance to the edge of the shadow map—a scalar value in world space.
- *S* is the detail level of at the center of the shadow map—a scalar value in world space.

The units of *D* and *S* should match the units in the game world and should remain constant across all facets. **F** should be normalized and have a magnitude of 1.0 and can be described as the direction of the line that bisects each facet, perpendicular to the light direction. Figure 4.4.3 illustrates how **F**, *D,* and *S* relate to a single facet; **L** is the light direction:

FIGURE 4.4.3 The magnitude of *S* is the width of the near clip plane on the dotted frustum, most of which will be clipped by the facet shown as the triangular wedge.



A derived value of *d* is calculated from *D* and *S*:

$$d = \frac{D + S}{D}$$

d is then used in the facet projection matrix, **M**:

$$\mathbf{M} = \begin{bmatrix} d & 0 & 0 & F_x \\ 0 & d & 0 & F_y \\ 0 & 0 & d & F_z \\ 0 & 0 & 0 & S \end{bmatrix}$$

The values chosen for *S* and *D* will affect the severity of the warping and therefore the detail that is stored in the shadow map closer or further from the center point. The greater the detail required at the center (controlled by *S*) and the larger the desired total area covered by the shadow map (controlled by *D*), the more severe the warping will be. This means it is possible to balance where the pixels of a fixed-size shadow map are used in a scene.

From an implementation standpoint, the facet projection matrix should be combined with the world matrix and light view matrix and applied together in a vertex shader, and polygons produced for each facet should be clipped along the edges of that facet. Clipping can be achieved by using the stencil buffer to cut out the shape of the facet on the map, or by dropping pixels rendered outside the facet (using instructions such as `clip( )` or `texkill`) or by simply enabling hardware clipping planes.

Here is an example vertex shader for rendering to a facet:

```
//note that all three matrices can be combined to a single matrix


void VS_RenderShadowFacet(in float4 pos : POSITION, in int facetIndex, out
float4 posOut: POSITION)
{

      //transform into world space
      float4 worldPos=mul(pos, worldMatrix);

      //transform into shadow view space
      float4 lightPos=mul(worldPos, lightViewMatrix);

      //output projected position
      posOut=mul(lightPos, facetProjectionMatrix[facetIndex]);
}
```

To use a facetted shadow map to determine whether a test point is in shadow, you need to determine which of the facets the test point falls in, and then transform the test point by the same matrix formulated to create the facet. A comparison can be made between the depth of the transformed point and the depth read from the shadow map. After the facet has been selected, the approach is essentially the same as using individual perspective shadow maps as described in [Stamminger02].

To determine which of the facet projection matrices to use, you can perform a dot product between **F**, the direction vector for each facet in the map, and choose the facet with the highest dot product. You can alternatively calculate the angle of the test point and rescale it to the index of the matrix directly. Here is an example pixel shader function demonstrating the use of an angle to index the facet matrix:

```
//this function returns 0.0 if in position is in shadow and 1.0 if not in
shadow


float PS_GetShadowFactor(float4 pos)
{

      //transform into world space
      float4 worldPos=mul(pos, worldMatrix);

      //transform world position into light view space
      float4 lightPos=mul(worldPos, lightViewMatrix);

      //calculate the angle per facet in radians - this value can be
precalculated and reciprocated to avoid using a divide instruction
      float facetAngle=(2.0*PI)/NUM_FACETS;

      //calculate facet index by calculating the angle, then dividing by
the facet angle
      float angle=atan2(lightPos.x,lightPos.y)+PI;
      int facetIndex=(angle/facetAngle);


      //once the facet index is found the facetted map can be treated as a
standard perspective shadow map
      float4 facetPos=mul(lightPos, facetProjectionMatrix[facetIndex]);
      float2 facetUV=((facetPos.xy/facetPos.w)*0.5)+0.5;


      float facetDepth=facetPos.z/facetPos.w;
      float shadowMapDepth=tex2D(FacettedShadowMapSampler, facetUV);

      if (shadowMapDepth<facetDepth)
            return 0.0;
      else
            return 1.0;
```

}

The example code above takes a single “tap.” It does not apply an averaging filter to the shadow result. Using multiple taps can help hide aliasing and give a softer shadow. Facetted shadow maps can be used with all shadow filtering methods, because apart from having to determine the correct facet matrix to use, once a facetted shadow map is constructed it is possible treat it as you would an unwarped orthographic shadow map. The filtering technique used to produce softer shadows in *GTA4* was a four-tap rotated disk-filtering technique.

Results

The end results can be qualitatively assessed by playing *GTA4* or looking at the color pictures found on the front cover of this book. In terms of performance, in a typical scene, the rendering of the facetted shadow map would take between 10 and 15% of the total frame time. Table 4.4.1 shows results taken from *GTA4*. Each scene corresponds to a color picture found on the front cover of this book from top to bottom:

TABLE 4.4.1 Results from *GTA4*

Firefly Island	2,364,288 vertices	11,928,880 pixels	4.31 ms
Star Junction	1,315,401 vertices	10,434,240 pixels	3.78 ms
Applejack St	1,429,142 vertices	11,447,552 pixels	3.64 ms

The relationship between vertex count, pixel count, and time taken to render a facetted shadow map is not linear and is dependant on the composition of each individual scene.

Memory usage was kept very low. The facetted shadow map used in *GTA4* used a total of only 6.4 MB for a 1,280×1,280 shadow map. Clearly, a larger map would increase the quality of the shadows and reduce aliasing, but this was not an option for *GTA4* due to memory constraints.

Conclusion

This article described a shadow mapping method suitable for large-scale dynamic game environments under the constraints of current game console hardware. The technique has some advantages over some existing shadow map approaches and was used to good effect in *GTA4*.

Future developments for the facetted approach may include using geometry shaders to render to many or all facets at once, by generating extra vertices on the facet boundaries as required. Further investigation into methods to offset the center of the facet map, and adjust facet projection matrices accordingly, could result in reduced aliasing in some combinations of light direction and viewing direction.

Acknowledgments

We would like to give thanks to all those involved in the creation of *GTA4*, and special thanks to Adam Fowler for his support and guidance throughout the development process, Aaron Garbut for constantly pushing for graphical excellence, and Wolfgang Engel for making this article possible.

References

[Williams78] L. Williams, “Casting Curved Shadows on Curved Surfaces,” ACM SIGGRAPH 1978.

[Stamminger02] Marc Stamminger and George Drettakis, “Perspective Shadow Maps,” ACM SIGGRAPH 2002.

[Wimmer04] Michael Wimmer, Daniel Scherzer, and Werner Purgathofer, “Light Space Perspective Shadow Maps,” Eurographics 2004.

[Martin04] Tobias Martin and Tiow-Seng Tan, “Anti-Alias and Continuity with Trapezoidal Shadow Maps,” Eurographics 2004.