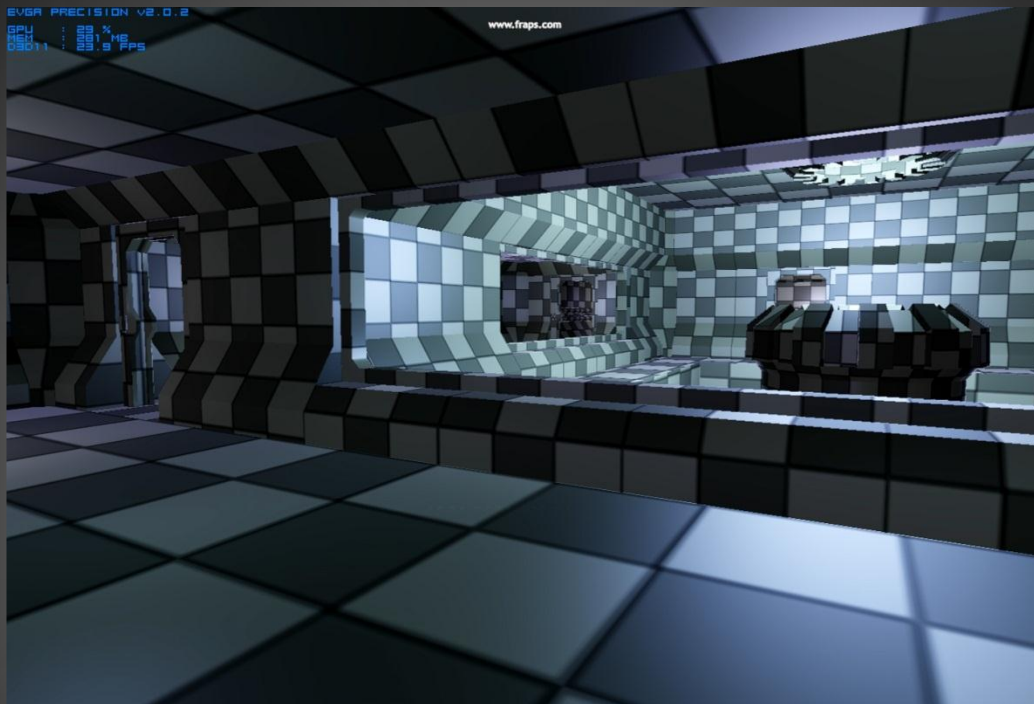

Building a Modern Game Engine

Joshua Carr

The design of a modern game engine has changed significantly from what it was ten years ago. The Warzone 3D Engine is a study in the graphics rendering techniques used in many of today's top game engines. This paper presents these techniques, considers the advantages and disadvantages of each, and discusses the challenges involved in their implementation.

ETSU

[Type the company address]

[Type the phone number]

[Type the fax number]

[Pick the date]

Building a Modern Game Engine

Joshua Carr

The design of the modern game engine has changed significantly from what it was ten years ago. Game engines are now being built to depend on today's more advanced processing technologies to meet the expectations of modern gamers. As a result, the design of the modern game engine has changed to support a more robust set of graphics rendering features. This paper presents some of these features and the techniques involved in integrating them into a game engine. The advantages and disadvantages of each are discussed, as well as the challenges involved in their implementations.

The Warzone 3D Engine is a small engine designed to explore many of the features of today's top game engines. As the engine is still under development, this paper will focus on its graphics rendering capabilities. The Warzone Engine's main design goal is to support fully dynamic environments without the need to prebuild lighting. Traditionally, game engines have rendered dynamic lights by first determining the geometry they affect, then drawing that geometry and additively blending the result of the lighting equation (usually Phong and Lambert) with the framebuffer. This means that the scene's geometry has to be redrawn at least once for each dynamic light that affects it, and twice for shadow casting lights, which is why traditional game engines use lightmaps to prebuild most of the lighting for a scene. This method is referred to as "forward shading."

With recent advances in graphics hardware and much more VRAM than was previously available, it is now possible to rasterize games in much different ways. These methods are called "deferred rendering." There are two main types of deferred rendering: deferred shading and deferred lighting. Deferred shading draws the scene's geometry only once to an array of framebuffers (referred to as the geometry buffer, or G-buffer). The geometry's surface and material properties are encoded into the RGBA components of the G-buffer textures. Such properties include the surface normal, world-space position, diffuse and specular colors and velocity. The G-buffer is then used in a series of post processing shaders that access these properties to render a virtually unlimited number of effects, including dynamic light sources. By storing the relevant surface data needed for subsequent effects, deferred shading has the advantage that geometry need only be rendered once at the beginning of each frame. However, managing the G-buffer can cost a great deal of VRAM, especially when 64-bit framebuffers are used.

Deferred lighting makes a trade between VRAM usage and geometry batching. In a deferred lighting engine, such as Warzone, only the surface normal and depth are needed, which greatly reduces the size of the G-buffer. Lighting is performed as a post process just as in deferred shading, and the diffuse and specular components are additively blended together into an off-screen framebuffer. Once the lighting is complete, a second geometry rendering pass is performed. During this final shading pass, the result from the lighting passes is simply sampled from a texture and added to the surface's color.