1. Introduction

Eve Online is a single shard MMO with tens of thousands of spaceships battling it out. Recently, its character customization system has seen a major upgrade, providing the players with a high definition avatar rendering system to sculpt, customize and tweak their in-game characters.

This customization runs at interactive rates in the client, with a static portrait rendered using the same system with higher settings at the online render farm.

In this presentation we survey the array of tricks and techniques used to sculpt and render high quality skin, hair, eyes and shadows. We break down a final render into its individual components, show what worked and what didn’t, and point out a few engine gems. Even if the techniques are relatively low key in order to get good results out of commodity hardware, we hope to offer one or two new tricks to everyone in the audience.

2. Contributions

Eve Online uses an established engine that was initially geared towards rendering massive space battles. In this 20 minute talk we give an overview of the changes and additions made to introduce high quality avatar rendering into this system.

2.1 Elaboration

We start with a quick recap of proper linear rendering [4], filmic tonemapping [9], and the skin specific specular model we used [6].

Next, we discuss our implementation of diffuse subsurface scattering [6] with a Poisson-sampled texture-space light diffusion [9]. So called renderjobs automatically translate rendering algorithms expressed in Python into a display-list-like recording [5]; this gives us very quick prototyping and iteration capabilities at minimal loss in runtime performance.

Subtle facial animation requires a good wrinkle system. Python code monitors animated bone positions to independently activate any of 12 facial zones per frame. An active zone will call on the shader to apply a normal offset and skin recoloring to simulate proper skin deformation.

Turning to shadow rendering, we survey a few techniques that were tested, including Variance Shadow Mapping [5, 9] and Fourier Opacity Mapping [3], before discussing our PCF solution. A tightening of the scene OBB to the view frustum leads to a low complexity but satisfying improvement in shadow quality. For hair and glasses, a screendoor technique is used to emulate transparent shadows [1].

3. Results

Using these techniques we achieve high quality character customization on commodity hardware, using DirectX9. By simply increasing a few basic settings such as input texture resolution and shadow map resolution, the same rendering system works as a render farm producing high definition player portraits such as Figure 1.

Figure 1. Player created portrait, rendered on the server.

References


Copyright is held by the author / owner(s).
ISBN 978-1-4503-0921-9/11/0008