Multi-pass Renderer Architecture

A BETTER WAY TO RENDER

Arnav
PI GAME ENGINE

Index

Multi-Pass Renderer Architecture

Basic Single-Pass Rendering Architecture:

1) Clear the RenderCommand and Framebuffer
2) Scene.Draw()
    i. Get the Active Camera to render from.
    ii. Loop over all GameObjects and update their transforms.
    iii. Bind materials & set Uniforms
    iv. Perform RenderCommand.DrawIndexed()

In this approach we only render the scene once. We have no options to add Post Processing effect or render shadows first and overlay it on scene when we render it second time. We also waste a lot of time setting the Light data over and over for each material.

The solution to all these problems is a Multi-Pass Renderer. In this document I will walk you over a simple Multi-pass Renderer Architecture & how it is implemented in PI, my game engine.

Material System:

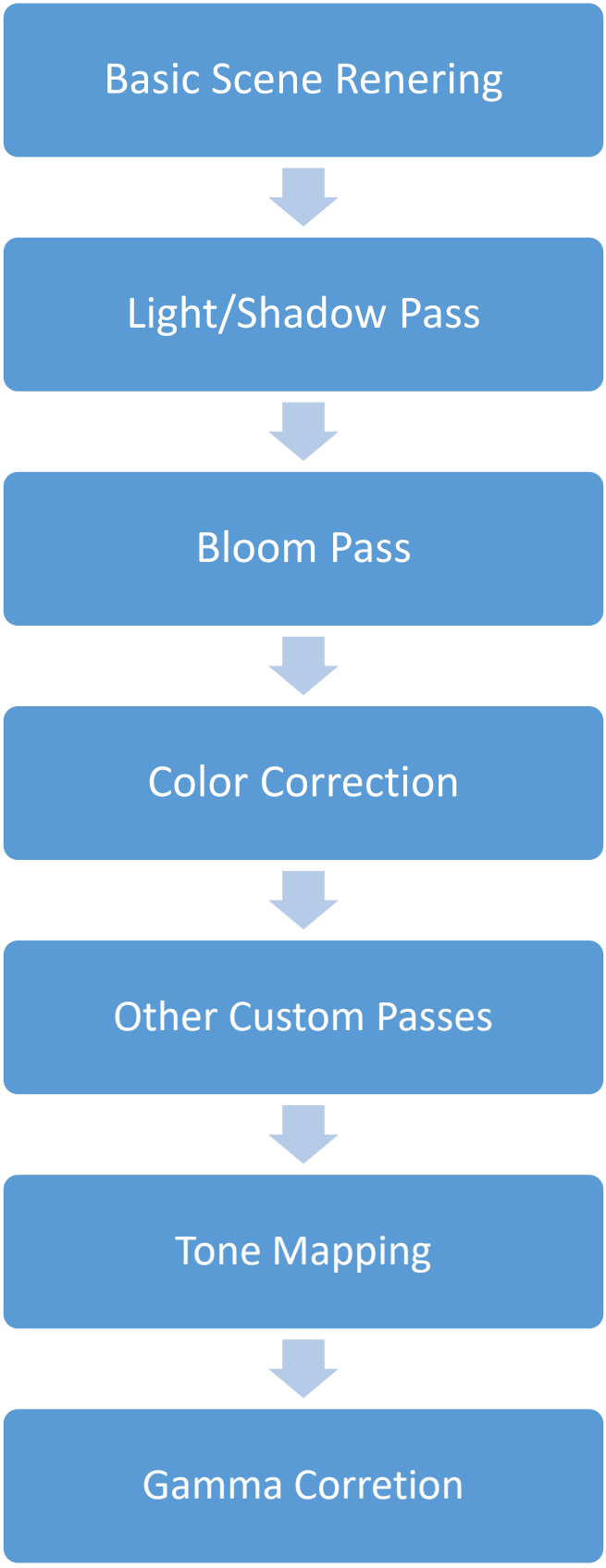
The basic unit of a Renderer is its Material System. And being the system which will be active for quite a lot during rendering it must be really efficient. So we must design it very carefully.

The Materials:

We will encapsulate Shaders as Materials (We might also use Spir-V to get Shader Uniform fields in automatically). Then when user attaches a material to an object we will attach a **MaterialReference** to it. Then when draw call is made we will sort the object material wise. Then just bind the shader upload all the lighting data and set material specific properties and pass the draw call to the rendering pipeline.

The Rendering Pipeline:

We will render first to a HDR-enabled framebuffer so that we can iteratively apply effects to it. We will do multiple pass to render the scene. The pipeline might look something like this:


Basic Scene Renering
Light/Shadow Pass
Bloom Pass
Color Correction
Other Custom Passes
Tone Mapping
Gamma Corretion

Some passes here, like bloom, might also require multiple passes. As you can see this architecture is quite slow but produces very high quality results.

Now we will go into details about some of the effects.

1. Basic Scene Rendering: