УДК 004.51

BUILDING A SCRIPTABLE RENDER PIPELINE IN UNITY ENGINE

Khodyka O. S.
Supervisor – Can. of Tech. Sciences., Sen. Res. Fellow Reshetnik V. M.
Kharkiv National University of Radio Electronics, Sys. Engineering Dep.
Kharkiv, Ukraine
e-mail: oleksandr.khodyka@nure.ua

This thesis is devoted to the process of building a custom Scriptable Render Pipeline (SRP) in Unity engine from scratch, as an alternative to using built-in solutions provided by Unity (Universal and High-Definition render pipelines). Developed SRP is designed for specific game and contains features such as physically based lighting, volumetric effects, support for image post processing effects, such as tone mapping, color grading and bloom, multiple available anti-aliasing techniques and a terrain system, which features virtual texturing, aimed to reduce overhead when rendering open environments.

Making a video game is a complicated process which involves a lot of decision making around technology that should be used. Using a general-purpose game engine such as Unreal Engine, Godot or Unity is a good option to save on development costs and reduce technical debt. However, changing engines mid-development is usually too costly and time consuming, especially when company already has legacy codebases designed to work with particular engine.

When developing in Unity, it is common that built in rendering system, called Built-In Render Pipeline (BRP), stops meeting project demands. Some of its problems can be avoided by using one of Scriptable Render Pipelines (SRP) [1] provided by Unity: Universal (URP) and High Definition render pipelines (HDRP). However, both of them have certain set of problems, such as poor visual quality or unacceptable execution time and memory consumption.

The focus of this work is to propose alternative, original, SRP, called Ed Render Pipeline (EDRP), developed using C# and HLSL programming languages. It is made for a game which targets DirectX 11 compatible GPUs and features densely populated forest scenes.

As a replacement for Unity's built-in deferred lighting pipeline, Forward+ [2] has been developed in combination with depth prepass. Compared to built-in light-silhouette rasterization, proposed solution reduces GPU frame computation. This is achieved by first drawing geometry in «depth only» mode, outputting image where pixels only contain distance from camera to the closest surface. Next, all objects are drawn again with all necessary lighting applied. This allows GPU to do less operations per pixel during «lighting» pass, since it can discard any pixel that doesn't have depth equal to one already drawn on screen.

Previously unavailable in BRP, volumetric fog effect is added by utilizing voxelization technique [3]. Compared to existing fog solution in HDRP

provided by Unity, presented solution does not suffer from visual noise, instead achieving smooth, blurry looking image. Proposed fog also has better fallback characteristics for long view-distances, as it doesn't suffer from sudden color changes. Additionally to volumetric fog, simplified analytical fog was developed which does not consider local lighting, relying only on sun light, but provides reasonable approximation for certain environments.

To better support rendering of natural environments, custom terrain rendering system was implemented, which utilizes virtual texturing. This allows to avoid multi pass terrain rendering of BRP. Instead, virtual texturing caches results of terrain layer blending and reuses them for multiple frames, which reduces computation time on GPU while also avoiding visual artifacts of multi pass system, such as incorrect surface normal blending. As a downside, this technique uses more of user's video memory.

Proposed render pipeline has been made to support post processing techniques such as color grading, bloom and tone mapping [4]. Post processing techniques are required to emulate features of real-world cameras, such as desaturation of bright objects, depending on camera exposure settings. Bloom effect simulates glow around very bright object, typically street lights.

Public API is exposed to third party developers as an effort to provide extensibility to the rendering pipeline. Public API allows to graphics programmers to create custom post-processing effects without having to modify source code of the pipeline itself, simplifying the process.

Intel's conservative morphological antialiasing is implemented, combined with custom temporal super sampling solution which aims reduce amount of under sampling artifacts, caused by limited screen resolution by approximating higher resolution image. Presented temporal super sampling achieves reasonably good results while avoiding typical problems of the algorithm such as ghosting and flickering.

As a result of this work, presented SRP provides all necessary features for the game. It supports personal computers on windows operating system with GPUs that compatible with DirectX 11 and above.

Literature:

1. Cooper T. Scriptable Render Pipeline Overview. Unity Blog. URL: https://blog.unity.com/technology/srp-overview (date of access: 02.03.2024).

2. Harada T., McKee J., Yang J. C. Forward+: Bringing Deferred Lighting to the Next Level. Eurographics 2012 – Short Papers. 2012. No. 33. P. 5–8.

3. Hillaire S. Physically-based & Unified Volumetric Rendering in Frostbite – Frostbite. Electronic Arts Inc. URL: https://www.ea.com/frostbite/news/physically-based-unified-volumetric-rendering-in-frostbite (date of access: 02.03.2024).

4. Haines E., Akenine-Möller T., Hoffman N. Real-Time Rendering, Fourth Edition. A K Peters/CRC Press, 2018. 1198 p.