April 2022

Distributed Shader Cache

Doug Horn

Distributed Shader Cache

ABSTRACT

The compile time for shaders for certain graphics APIs can sometimes be unacceptably long. Compilation of shaders during gameplay can, depending on CPU/GPU load, cause awkward rendering pauses (janks) or artifacts in games. This disclosure describes a shader service wherein shaders are pre-compiled for multiple target GPUs and are cached on servers. A game device can download the appropriate pre-compiled, binary shaders from the servers based on various factors such as the version of the game, the model of the user's GPU, the driver version for that GPU, etc. The described techniques can speed up gameplay by avoiding shader compilation at the start of a game or during gameplay. The techniques can be utilized for video games, game services, or for other GPU applications.

KEYWORDS

- Shader
- Graphics API
- Direct3D
- OpenGL
- Pre-compilation
- D3D11
- Graphics processing unit (GPU)
- Remote procedure call (RPC)

BACKGROUND

Graphics application programming interfaces (APIs) such as Direct3D, OpenGL, etc., are used to render three-dimensional graphics in high-performance applications, e.g., video games. Graphics APIs use hardware acceleration, e.g., GPUs, for 3-D rendering. Shading (the creation of depth in 3D models) can be done by programs known as shaders, which work by varying darkness or hue to approximate the local behavior of light on the surface of an object.

Shaders are generally compiled when they are loaded by the game and are then cached locally. The compilation step results in a significant delay in some cases, directly impacting gameplay. The compile time for shaders on many graphics APIs, e.g., Direct3D 11 (D3D11), can be long. Compilation of shaders during gameplay can, depending on CPU/GPU load, cause awkward rendering pauses (janks) or artifacts in games.

Many graphics libraries/drivers include local shader caches that use the local disk as a main cache. The distributed cache stores compiled shader binaries (and/or linked programs) which are downloaded by a client when requested, e.g., during game installation. While local caching can address the problem of slow compilation, it still requires shaders to be compiled for the first time.

DESCRIPTION

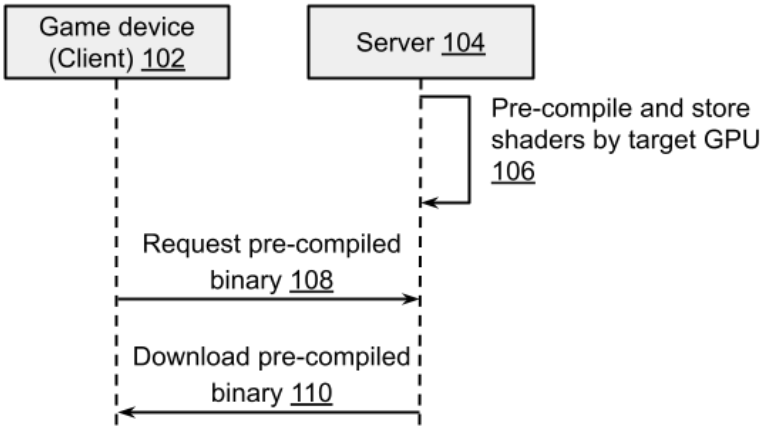

Fig. 1: Distributed shader cache

Illustrated in Fig. 1, this disclosure describes a shader service wherein shaders are pre-compiled, by target GPU as necessary (106) and are cached on one or more servers (104). A game device (102) can request (108) a pre-compiled binary using, e.g., a remote procedure call (RPC) interface. Upon receipt of the request by the server, the RPC interface is utilized to

download pre-compiled shader binary (110) from the servers based on various factors, such as the version of the game being installed, the model of the client device GPU, the driver version for the GPU, etc.

The described techniques can speed up gameplay by avoiding shader compilation at the start of a game or during gameplay. The techniques can be utilized for video games, game services, or for other GPU applications.

CONCLUSION

This disclosure describes a shader service wherein shaders are pre-compiled and cached on one or more servers. A game device can download pre-compiled, binary shaders from the servers based on a number of factors, e.g., the version of the game being installed, the model of the user's GPU, the driver version for that GPU, etc. The described techniques generally speed up gameplay by avoiding compilation at game-start or during gameplay. They apply to video games, game services, or other GPU applications.

REFERENCES

[1] Lalonde, Paul, Paul Leventis, and Jean-Francois Roy. "Memory management in gaming rendering." U.S. Patent 11,110,348, issued September 7, 2021.

[2] "Shader Cache and Generation - Lumberyard User Guide" available online at https://docs.aws.amazon.com/lumberyard/latest/userguide/mat-shaders-custom-dev-cache-intro.html accessed Apr 3, 2022.