52. VIDEO GAME GRAPHICS OPTIMIZATION

Minov I.E.

Belarusian State University of Informatics and Radioelectronics
Minsk, Republic of Belarus

Maksimchuk R.T. – Senior Lecturer

The necessity of optimizing video games is described. Key elements influencing performance are delineated. The most common ways to optimize video game graphics are listed.

Video game optimization is a process of upgrading the game's performance and smoothness. It is vital, because it directly affects a game's playability and accessibility [1]. Video game optimization requires extra work and expenses on the part of the publisher, but at the same time a number of advantages are achieved. A well-known performance metric that greatly benefits from optimization is frames per second (FPS), but this is not the only indicator. Due to better optimization, video games have lower input latency, they have a wider range of devices capable of processing the game, they tend not to load hardware as much, so they are more energy sufficient. All of these advantages are likely to expand the consumer base, so it is usually economically beneficial.

There are several approaches to optimize video game graphics [2]. The general rule is that the fewer polygons are drawn, the easier it is for a graphics processing unit (GPU) to process the scene. To achieve this goal developers can use the following techniques:

- Clipping and culling. These are the two most basic ways to optimize rendering. The idea is not to draw invisible polygons or parts of them (due to the fact that they are outside the screen space or are the inside of the model). This technique is important and is usually implemented in the graphics application programming interface (API) [3].
- Assets optimization. While creating assets, specialists must be sure that the number of polygons used will be reduced. Usually, the same shape can be created from triangles in several ways. In addition, the graphics can be simplified in order to improve performance.
- Level of Detail (LOD). LOD is the process of creating less detailed versions of a model that replace the original model when it is further away from the camera. Distant objects may look indistinguishable from their simplified versions, so there is no need to detail them [4].

In addition to reducing the number of polygons, the resolution can also be reduced. This is usually done with shadows and reflections, but it could be done in general, at the cost of visual clarity. To preserve the look of the game and to improve the performance, various upscaling technologies could be used. For example, Nvidia DLSS Super Resolution uses artificial intelligence to output higher-resolution frames from a lower-resolution data, although special hardware is required for this particular technology [5].

Another way to speed up the game is to render some elements beforehand. Developers can draw shadows or reflections to a texture on their computers, and then draw the prepared texture on the user's computer. Thus, a lot of computations are performed even before the game starts, but this cannot be used for dynamic scenes, since the player's behavior can change shadows and reflections. Optimization also involves refactoring the game's codebase for better efficiency. This refactoring often includes optimizing algorithms and data structures to reduce computational complexity and memory usage. Also, by implementing shader optimization, developers can create visually stunning effects that are computationally less intensive.

In conclusion, it is worth noting that optimizing video games is not an easy task. It is important to maintain a balance between the resources invested in the project and its condition. Although there will always be things, that could be upgraded, there will always be a moment after which further optimization will not affect the user's experience.

References:

1. The Essentials of Video Game Optimization | Maximizing Performance [Electronic resource]. – Mode of access: https://polydin.com/video-game-optimization. – Date of access: 09.03.2024 .

2. Graphics performance fundamentals [Electronic resource]. – Mode of access: https://docs.unity3d.com/Manual/OptimizingGraphicsPerformance. – Date of access: 09.03.2024.

3. Clipping, Culling, and Hidden Surface Removal [Electronic resource]. – Mode of access: https://graphicscompendium.com/opengl/24-clipping-culling. – Date of access: 09.03.2024.

4. What is LOD (Level of Detail) in 3D Modeling? [Electronic resource]. – Mode of access: https://conceptartempire.com/3d-lod-level-of-detail. – Date of access: 09.03.2024.

5. Max FPS. Max Quality. Powered by AI [Electronic resource]. – Mode of access: https://www.nvidia.com/en-gb/geforce/technologies/dlss/?nvid=nv-int-solr-416571. – Date of access: 09.03.2024.