

ПРОБЛЕМИ УПРАВЛІННЯ РЕСУРСАМИ OpenGL та DirectX

Худік Д.В.¹, Жульковський О.О., к.т.н., доц.²
^{1, 2} Дніпровський державний технічний університет

^{1, 2} Україна, Кам'янське

¹ khudik.dmytro@gmail.com; ² olalzh@ukr.net

Анотація. Досліджено проблеми управління пам'яттю в OpenGL та DirectX, зокрема обмеження пам'яті GPU та фрагментацію. Запропоновано методи оптимізації, такі як управління буферами та стиснення текстур. Отримані результати показують можливість підвищення продуктивності графічних застосунків.

Ключові слова: управління пам'яттю; методи оптимізації; управління буферами; стиснення текстур.

Вступна частина. В процесі розвитку технологій роботи з комп'ютерною графікою та візуальними ефектами, питання оптимізації використання пам'яті в створюваних застосунках стає актуальним завданням для розробників. Зокрема, API OpenGL та DirectX широко застосовуються для реалізації комп'ютерної графіки, однак при більш глибокому вивченні, можна зробити однозначний висновок, що обидва інструмента стикаються з певними проблемами під час управління ресурсами. До ключових завдань оптимізації даного процесу належать обмежений обсяг пам'яті графічного процесору, неефективна передача даних між центральним та графічним процесором, а також фрагментація пам'яті, що призводять до зниження продуктивності роботи програм.

Актуальність теми дослідження обумовлена постійним зростанням вимог до графіки в ігровій індустрії, 3D-моделюванні, віртуальній реальності та машинному навчанні. Існуючі попередні дослідження демонструють, що традиційні методи вирішення проблеми оптимізації, такі як стиснення текстур або оптимізація буферів, часто повністю не вирішують проблеми продуктивності. Розробки останніх років вказують на необхідність нових підходів, які будуть включати в себе покращене управління ресурсами на рівні API, зокрема в контексті багатопоточності та асинхронної передачі даних.

Метою дослідження є аналіз існуючих проблем управління пам'яттю в графічних застосунках, побудованих на OpenGL та DirectX, а також дослідження традиційних методів оптимізації та більш суттєвих підходів, спрямованих на максимально ефективне використання ресурсів для підвищення продуктивності використання графіки.

Основна частина. Проведено аналіз основних викликів, з якими стикаються розробники: фрагментація пам'яті; передача даних між CPU та GPU [1]; обмеження в управлінні ресурсами. Для вирішення вищезгаданих проблем у дослідженні було випробувано декілька підходів до оптимізації використання пам'яті.

Оптимізація управління буферами – один із основних підходів до вирішення проблеми фрагментації пам'яті. Замість використання великої кількості малих буферів, що можуть сприяти фрагментації, запропоновано застосовувати великі єдині буфери для зберігання різних типів даних. Це знижує кількість перемикачів між буферами і зменшує навантаження на GPU. У DirectX 12 був протестований підхід «Resource Aliasing», який дозволяє кільком різним об'єктам використовувати ту саму область пам'яті, коли вони не використовуються одночасно. Це дозволяє зекономити значні обсяги пам'яті та зменшити ризик фрагментації [2].

Оскільки текстури займають значну частину пам'яті GPU, застосування стиснення текстур та методів адаптивного рендерингу може значно покращити продуктивність. У дослідженні застосовано методи стиснення текстур за допомогою алгоритмів DXT (для DirectX) та ETC (для OpenGL). Крім того, було використано технологію MIP-mapping, яка дозволяє використовувати текстури з різною роздільною здатністю залежно від відстані до об'єкта. Цей підхід дозволив не лише зменшити використання пам'яті для текстур, але й оптимізувати продуктивність, оскільки GPU завантажується меншою кількістю пікселів на далеких об'єктах [3].

Кешування та передача даних дозволяє зменшити кількість передач між CPU та GPU. Було застосовано техніки кешування, які дозволяють зберігати найбільш використовувані ресурси без необхідності їх постійного оновлення. Для тестування розроблено систему асинхронного завантаження даних, яка дозволяє CPU і GPU працювати паралельно без очікування завершення передачі ресурсів. Такий підхід забезпечує безперервний рендеринг навіть при великих обсягах даних [4]. Також в процесі дослідження було протестовано методи використання асинхронних обчислень, зокрема шейдери, які виконують обробку даних незалежно від основного потоку рендерингу. В результаті основний процес рендерингу був розвантажений, знижуючи при цьому затримки під час обробки великих сцен та складних ефектів.

Експериментальні тести продемонстрували суттєві покращення в продуктивності при застосуванні описаних методів оптимізації:

- використання великих буферів дозволило знизити фрагментацію пам'яті на 30 %, що призвело до підвищення загальної продуктивності на 15 % при рендерингу складних сцен;
- стиснення текстур та використання MIP-mapping дало змогу зменшити обсяг пам'яті, що використовується для текстур, на 40 % без втрат якості графіки;
- кешування та асинхронна передача даних знизили кількість звернень до GPU на 25 %, що дозволяє досягти плавного рендерингу при великому навантаженні.

Висновки. Незважаючи на досягнуті покращення, ще залишаються відкриті питання, які можуть стати напрямками для подальших досліджень, наприклад: автоматичне управління пам'яттю; необхідність у більш розумних алгоритмах автоматичного управління пам'яттю на рівні API, щоб зменшити втрати від людських помилок; робота з новими технологіями (інтеграція сучасних графічних технологій, таких як Ray Tracing та AI-інструментів; паралелізація обробки графіки, зокрема, використання кількох GPU та багатоядерних процесорів для зниження навантаження на один обчислювальний блок [5].

Дослідження показало, що існуючі проблеми управління пам'яттю в OpenGL та DirectX можна суттєво нівелювати за рахунок використання вдосконалених методів оптимізації. Однак залишається потенціал для подальшого покращення, особливо у сфері автоматизації управління ресурсами та паралелізації обчислень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

- [1] Microsoft. (2022). Resource: Microsoft Learn. *Optimizations and advanced topics for DirectX games*. <https://learn.microsoft.com/en-us/windows/uwp/gaming/directx-optimization-and-advanced-topics>
- [2] Microsoft. (2022). Resource: Microsoft Learn. *Asynchronous programming (DirectX and C++)*. <https://learn.microsoft.com/en-us/windows/uwp/gaming/asynchronous-programming-directx-and-cpp>
- [3] Microsoft. (2022). Resource: Microsoft Learn. *Optimize input latency for Universal Windows Platform (UWP) DirectX games*. <https://learn.microsoft.com/en-us/windows/uwp/gaming/optimize-performance-for-windows-store-direct3d-11-apps-with-coredispatcher>

- [4] Yamagiwa, S., Sousa, L., & Antão, D. (2007). Data buffering optimization methods toward a uniform programming interface for gpu-based applications. *CF '07: Proceedings of the 4th international conference on Computing frontiers* (pp. 205-212). ACM Press. DOI: <https://doi.org/10.1145/1242531.1242562>
- [5] McReynolds, T., & Blythe, D. (2005). *Advanced Graphics Programming Using OpenGL*. Elsevier.

Khudik D.V., Zhulkovskyi O.O.

Problems of OpenGL and DirectX resource management

Abstract. *The article investigates memory management issues in OpenGL and DirectX, including GPU memory limitations and fragmentation. Optimization methods such as buffer management and texture compression are proposed. The obtained results show the possibility of improving the performance of graphic applications.*

Keywords: *memory management; optimization methods; buffer management; texture compression.*