Федеральное государственное автономное образовательное учреждение
высшего образования
«Московский физико-технический институт (государственный университет)»
Физтех-школа прикладной математики и информатики
Центр обучения проектированию и разработке игр

Направление подготовки: 09.04.01 Информатика и вычислительная техника

Направленность (профиль) подготовки: Анализ данных и разработка информационных систем

Архитектура рендеринга реального времени через вычислительный граф

(магистерская диссертация)

Студент:

Санду Роман Александрович

_____

(подпись студента)

Научный руководитель:

Щербаков Александр Станиславович

_____

(подпись научного руководителя)

Москва 2023

Аннотация

Данная работа посвящена одному из подходов к построению архитектуры приложений реального времени, называемого неформально ”фреймграфом” или ”рендерграфом”. Подход основывается на использовании вычислительного графа как представления процесса вычисления итоговой картинки одного кадра приложения.

Содержание

1. Введение

1.1. Аллокация ресурсов

В процессе вычисления картинки одного кадра любое нетривиальное приложение использует *транзиентные ресурсы* – промежуточные хранилища данных, содержимое которых не требуется после окончания вычисления кадра. Основная отличительная черта рассматриваемого подхода заключается в известности всей информации о транзиентных ресурсах заранее, что позволяет управлять ими более эффективно. Более того, наш подход позволяет добиться в определённом смысле оптимальной работы с такими ресурсами, как будет видно дальше. Однако обязательным пререквезитом для эффективной аллокации ресурсов является использование современного графического API, предоставляющего возможность ручного управления видеопамятью. До появления подобных API большая часть приложений использовало один из следующих наивных подходов к управлению ресурсами.

Самый простым подходом является выделение и освобождение транзиентных ресурсов по ходу их нужды при помощи соответствующих вызовов графического API. Этот подход сильно похож к управлению памятью объектов в системных языках программирования: драйвер операционной системы содержит аллокатор, на который пользователь перекладывает обязанность управления памятью и другими ресурсами GPU, аналогично куче в языке C. Системный аллокатор переиспользует освободившуюся память, тем самым достигая низкого её потребления. Однако такой подход не масштабируется на более сложные приложения. (фрагментация, отложенное удаление, рефкаунтинг, етц)

Альтернативным подходом служит отказ от переиспользования памяти. Все транзиентные ресурсы создаются заранее и не удаляются в ходе работы приложения. ...

Наконец, наиболее практичным подходом является пулинг ресурсов. ...

2. Обзор существующих работ

2.1. Имплементации

Frostbite

EA выступление[1]

Halcyon

EA выступление[2] идеален во всём, но пока только R&D

Unity

документация[3] закрытая, но вроде хорошая

Unreal Engine

документация[4]

Anvil

Ubisoft выступление[5] есть алиасинг, есть автобарьеры (сплит), умеет в несколько очередей сабмита

Granite

блог[6]

Прочие

Неинтересные: https://github.com/azhirnov/FrameGraph – нет алиасинга, очень много ООП, намертво привязан к вулкану, вершины не реордерятся, содержимое вершин – фиксированные таски, а не произвольный код, нет хистори ресурсов, есть барьеры, ВРОДЕ БЫ нет алиасинга https://github.com/skaarj1989/FrameGraph – нет алиасинга, нет хистори ресурсов, нет барьеров, кросс-АПИ, прикольный интерфейс на C++, видимо заброшен https://github.com/Raikiri/Leg – ОТЕЧЕСТВЕННОЕ!!!

2.2. Аллокация ресурсов

Задача поиска расписания аллокации ресурсов в графе кадра в своей простейшей формулировке является классической задачей динамической аллокации памяти (dynamic storage allocation, DSA[7, стр. 226]). Однако с названием и формулировкой этой задачи вышел казус: название задачи подразумевает онлайн-алгоритм, отвечающий на запросы аллокации по мере их поступления, а формулировка подразумевает оффлайн-алгоритм, знающий расписание использования ресурсов заранее. К сожалению большая часть работ посвящённых этой задаче фокусируется именно на динамическом аспекте, в то время как в данной работе интересен именно статический аспект. Некоторые авторы рассматривают обобщённую задачу об аллокации памяти (storage allocation problem, SAP), акцентирующую оффлайн аспект проблемы. Недавним прорывом в эффективном решении этой задачи было нахождение полиномиального алгоритма с ошибкой относительно оптимума менее чем в 2 раза[8].

Похожая задача, как бы это не было удивительно, возникает в области оперирования морских контейнерных терминалов. С ростом сложности и нагруженности глобальных транспортных цепочек, прикладные задачи оперирования верфей стали слишком сложны для интуитивного их решения. В связи с этим за последние несколько десятилетий было сформулировано и в той или иной степени решено множество вариаций *задачи об аллокации верфи*, покрывающих широкий спектр прикладных задач, возникающих в портах по всему миру. Задача динамической аллокации памяти с некоторыми допущениями эквивалентна одной из формулировок этой задачи, а именно вариации $cont|dyn|fix|max(res)$ по классификации обзорной статьи Бирвирта и Мизла[9]. Решая эту задачу, авторы статей фокусируются на статическом аспекте аллокации, используя глобальные знания о временах аллокации ресурсов. Именно из-за этого задача об аллокации верфи представляет интерес в рамках данной работы.

Одним из первых интересующую нас формулировку задачи об аллокации верфи рассмотрел в своей статье Эндрю Лим[10]. Ресурсы, имеющие фиксированные и известные размер и времена аллокации и деаллокации, могут быть рассмотрены как корабли с соответствующей длинной, временем прибытия и временем отплытия, а тип используемой видеопамяти как секция верфи. Задача нахождения минимальной длинны всех секций верфи и точек прибытия всех кораблей аналогична нахождению минимального необходимого объёма памяти и локаций всех ресурсов в этой памяти. Однако, в отличии от рассматриваемой Лимом зада-

чи, ресурсы не накладывают требований на отступ между друг другом и началом или концом верфи, зато требуют определённого выравнивания их начала в памяти. Более того, некоторые алгоритмы компьютерной графики требуют доступа к данным ресурса с прошлых кадров, а в ходе разработки фреймграфа для Dagor Engine выяснилось, что среди функционала движка таких алгоритмов большинство. Требование поддерживать аллокацию и переиспользование памяти ресурсов, переживающих границу кадра, заставляет нас считать время циклическим. В терминах задачи о верфи это требование можно сравнить с составлением неизменного расписания работы верфи на неделю, где ни в один момент времени верфь не простаивает. Последние замечание существенно, ведь иначе момент простаивания можно считать началом недели и задача сводится к сформулированной Лимом. Геометрически, циклическое время означает что задача заключается в упаковке прямоугольников на поверхности бесконечного в одну сторону цилиндра, а не бесконечной в одну сторону полосы.

Список литературы

1. *O'Donnell Y.* FrameGraph: Extensible Rendering Architecture in Frostbite. — 2017. — URL: https://www.gdcvault.com/play/1024612 ; Game Developers Conference.
2. *Wihlidal G.* Halcyon: Rapid innovation using modern graphics. — 2019. — URL: https://www.youtube.com/watch?v=da_6dsWz8yg ; Reboot Develop.
3. *Technologies U.* Unity render graph system. — URL: https://docs.unity3d.com/Packages/com.unity.render-pipelines.core%4014.0/manual/render-graph-system.html.
4. *Games E.* Unreal Engine rebder dependency graph. — URL: https://docs.unrealengine.com/5.0/en-US/render-dependency-graph-in-unreal-engine/.
5. *Gruen H.* DirectX™ 12 Case Studies. — 2017. — URL: https://www.gdcvault.com/play/1024343 ; Game Developers Conference.
6. *Arntzen H.-K.* Render graphs and Vulkan — a deep dive. — 2017. — URL: https://themaister.net/blog/2017/08/15/render-graphs-and-vulkan-a-deep-dive/.
7. *Garey M. R.*, *Johnson D. S.* Computers and Intractability; A Guide to the Theory of NP-Completeness. — USA : W. H. Freeman & Co., 1990. — ISBN 0716710455.
8. *Mömke T.*, *Wiese A.* Breaking the Barrier of 2 for the Storage Allocation Problem. — 2019. — DOI: 10.48550/ARXIV.1911.10871. — URL: https://arxiv.org/abs/1911.10871.
9. *Bierwirth C.*, *Meisel F.* A survey of berth allocation and quay crane scheduling problems in container terminals // European Journal of Operational Research. — 2010. — Т. 202, № 3. — С. 615—627. — ISSN 0377-2217. — DOI: https://doi.org/10.1016/j.ejor.2009.05.031. — URL: https://www.sciencedirect.com/science/article/pii/S0377221709003579.
10. *Lim A.* The berth planning problem11This research is part of a project the author has with the Port of Singapore Authority. The author was a senior technical consultant in that project. This research was supported in part by the NUS Research Front RP3972679. // Operations Research Letters. — 1998. — Т. 22, № 2. — С. 105—110. — ISSN 0167-6377. — DOI: https://doi.org/10.1016/S0167-6377(98)00010-8. — URL: https://www.sciencedirect.com/science/article/pii/S0167637798000108.