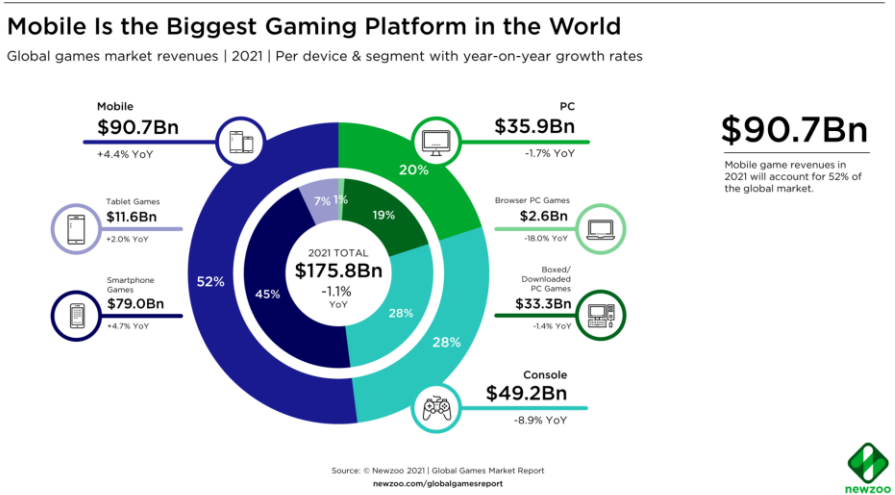
Реализация высокопроизводительной трассировки лучей в реальном времени на мобильных устройствах

Тюленева Эмилия Азаматовна, 21.M04-мм группа

Научный руководитель: д.ф. - м.н. Терехов Андрей Николаевич

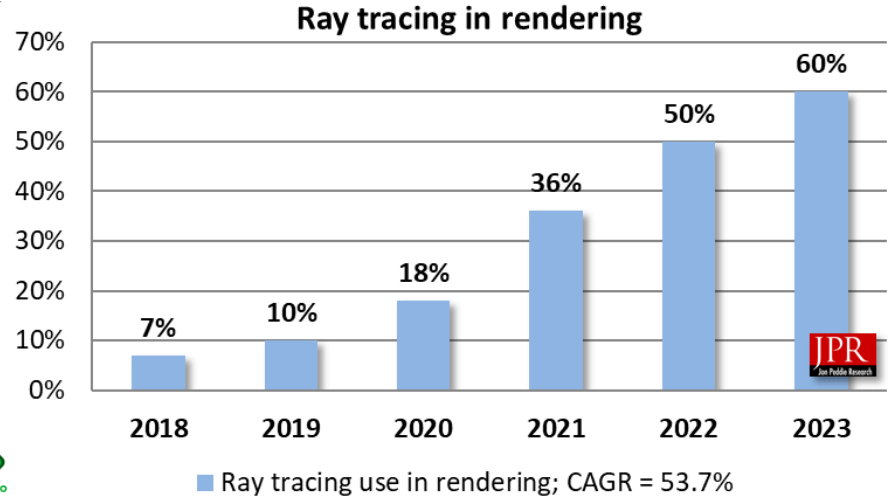
2023

Мотивация



$90.7Bn

Mobile game revenues in 2021 will account for 52% of the global market.



Цель дипломной работы

Цель:

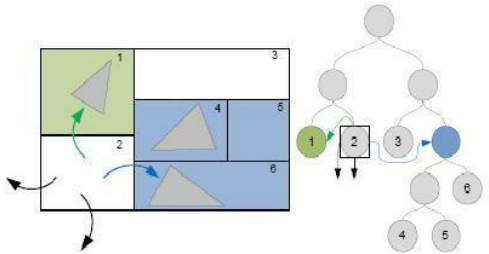
Оптимизировать алгоритм обхода акселерационной структуры сцены в методе трассировки лучей на мобильном GPU с целью дальнейшего применения в рендеринге мягких теней и отражений на широком классе мобильных устройств.

Задачи:

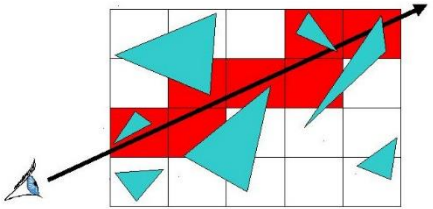
- Выполнить обзор предметной области, а также существующих методов по оптимизации обхода акселерационной структуры сцены
- Проанализировать и выбрать метрики для оценки эффективности алгоритмов обхода акселерационной структуры сцены
- Исследовать узкие места и оптимизировать метод обхода акселерационной структуры сцены с применением алгоритмов сжатия, арифметики пониженной точности и оптимального расположения узлов в памяти
- Провести эксперименты по применению оптимизированного метода на примере создания таких эффектов, как мягкие тени и отражения

Акселерационные структуры данных

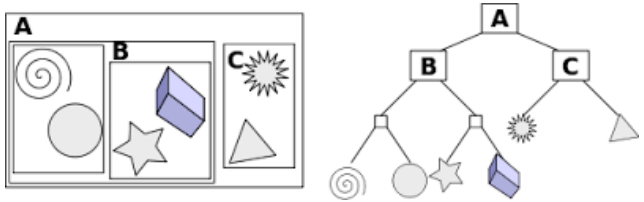
Аксерационные структуры помогают уменьшить количество тестов пересечений луча с объектами сцены от $O(n)$ до $O(\log n)$ путем организации геометрии сцены в особую структуру данных.



Kd-деревья



Регулярные решетки



BVH

BVH

BVH - древовидная структура данных, отделяющая сцену только вокруг объектов, в листовых узлах которой содержатся примитивы сцены.

Двухуровневый BVH:

- Нижний уровень (BLAS)
- Верхний уровень (TLAS)

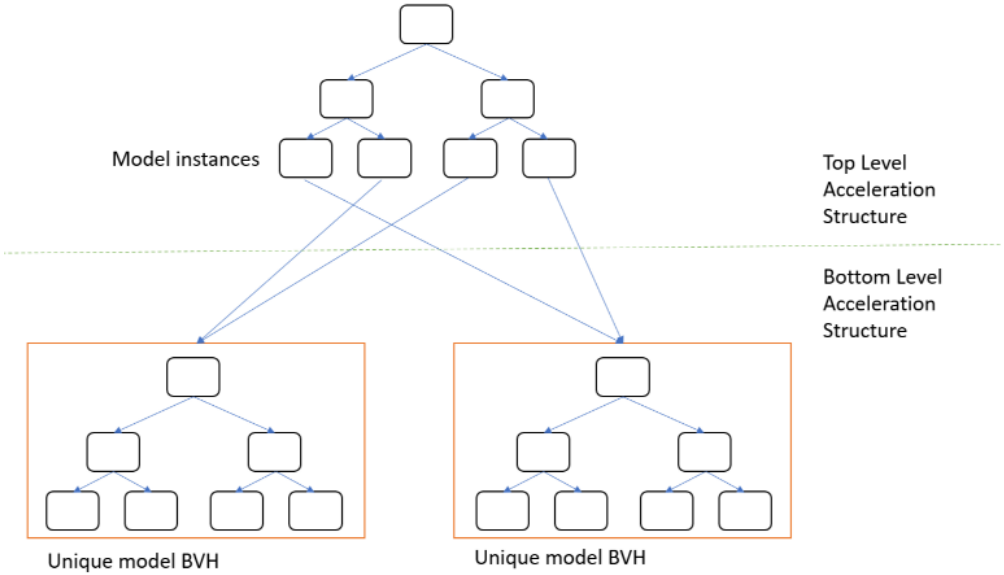


Схема двухуровневого BVH

Обзор алгоритмов эффективного траверсинга BVH

- Incremental traversing
- Квантизация BVH
- Переиспользование общих плоскостей между родительским и дочерним узлом
- Элиминация стека в алгоритме обхода BVH
- Эффективное расположение узлов BVH в памяти
- Пакетная трассировки лучей

Технологический стек

1. Vulkan API.

- Низкоуровневый, кроссплатформенный API
- Требует меньше накладных расходов в сравнении с API OpenGL и Direct3D
- Гибко и эффективно позволяет загрузить графический конвейер
- Работает с промежуточным двоичным форматом SPIR-V

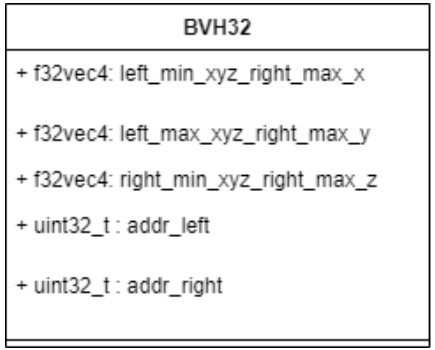
2. C++

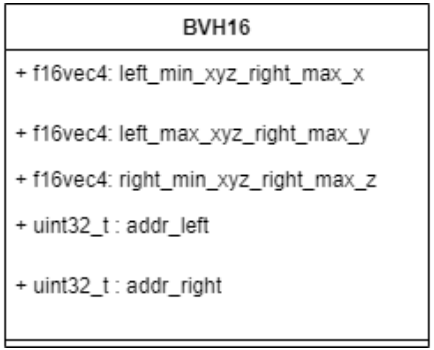
3. GLSL – язык для программирования шейдеров

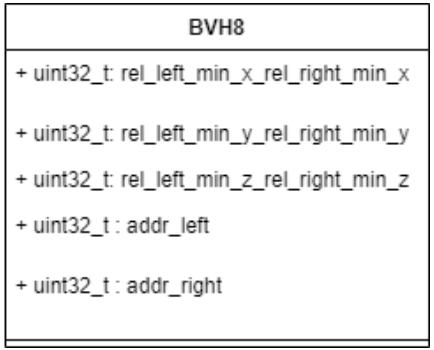
Оптимизация алгоритма траверсинга BVH

1.Компрессия BVH.

- BVH32
- BVH16
- BVH8







2.Расположение узлов BVH в памяти.

- BFS
- DFS

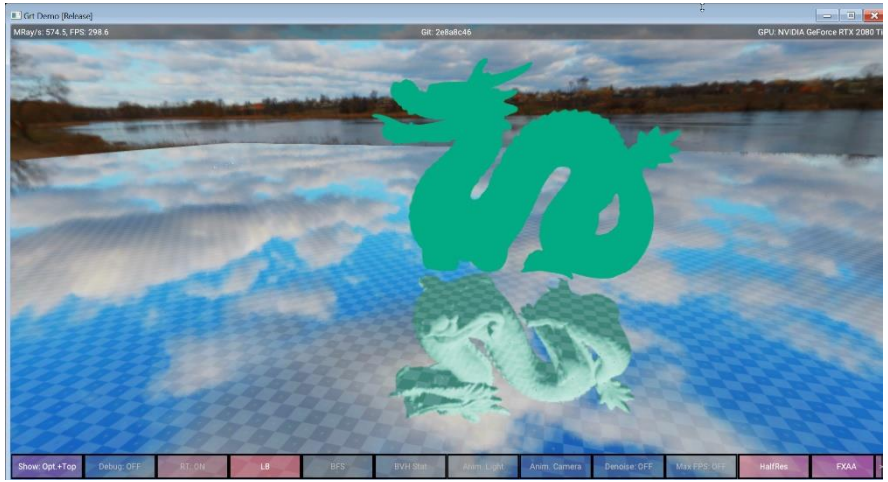
3. Использование расширений языка GLSL.

4. Оптимизация алгоритма обхода BVH16 (TLAS)+BVH8(BLAS).

5. Оптимизация хранения BVH16 (TLAS)+BVH8(BLAS).

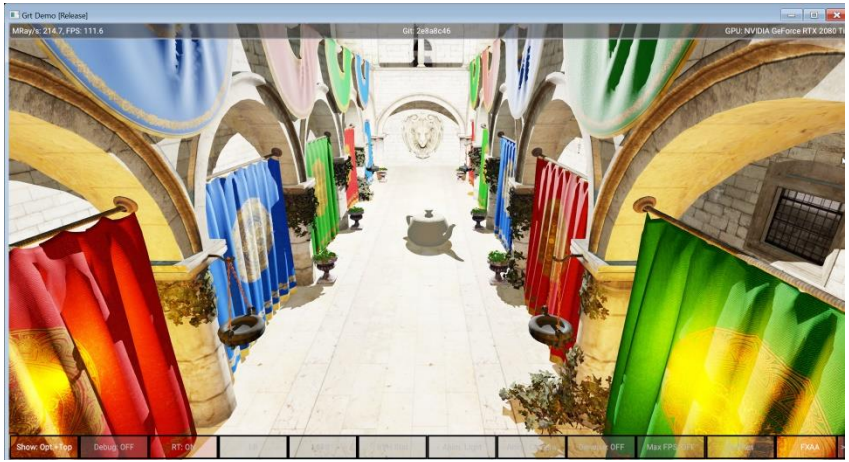
Апробация алгоритмов

Bistro: 1005421 полигонов

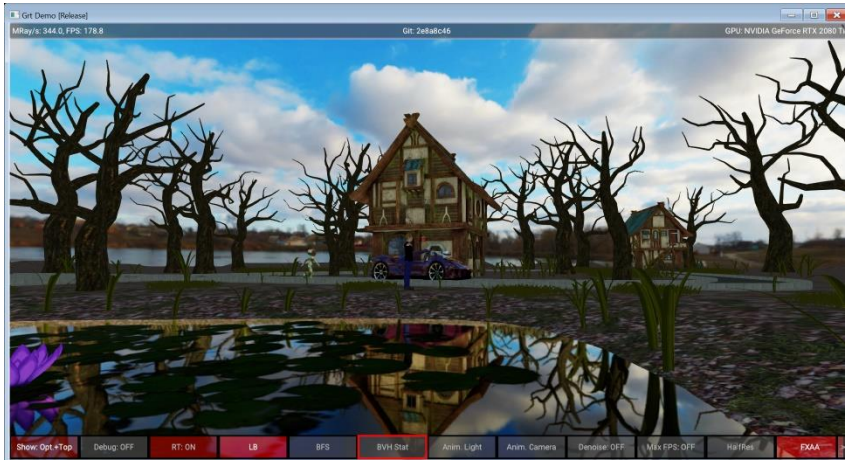


Dragon: 871308 полигонов

Апробация алгоритмов



Sponza: 277971 полигонов



Village: 157926 полигонов

Апробация алгоритмов

BVH метрики:

- BVH nodes
- BVH size
- Tree depth
- Uncompressed BLAS Depth (UBD)

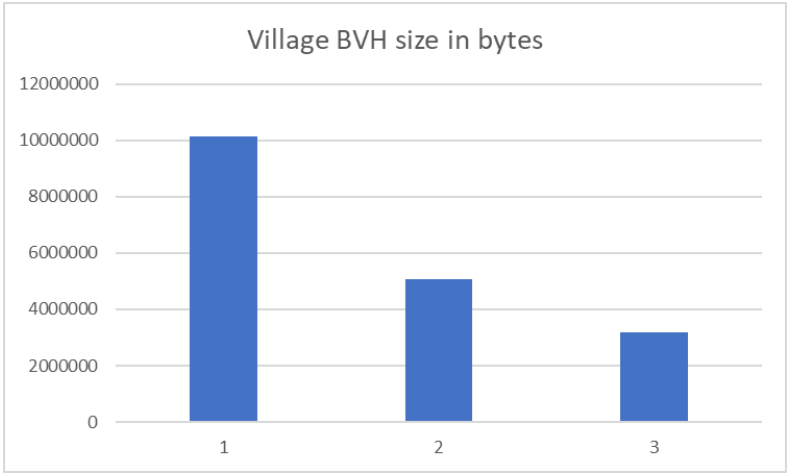
Метрики, связанные с эффективностью алгоритма обхода BVH:

- Rays
- Tris load
- Tris Intrs
- Nodes loaded
- Node Intrsc

Метрики, связанные с эффективностью использования стековой и кэш памяти:

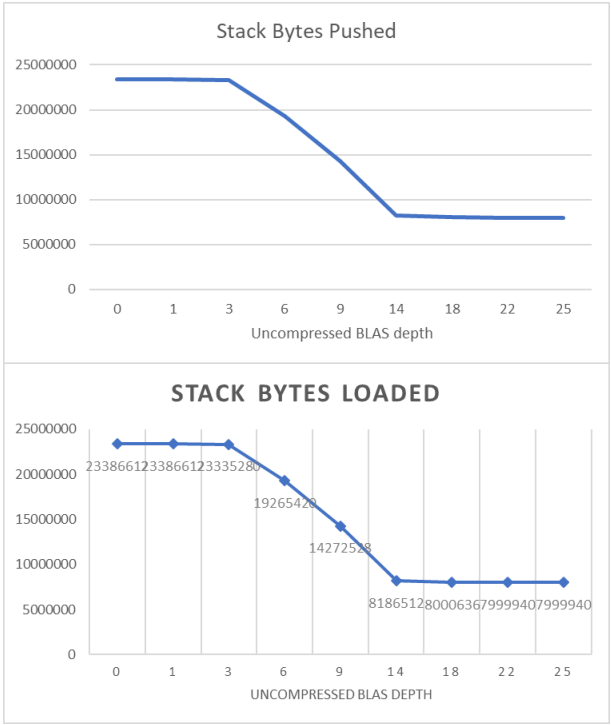
- Stack Bytes Loaded (SBL)
- Stack Bytes Pushed (SBP)
- Leaf count dispersion (LCD)
- Long jumps count (LJC)
- Cache miss count (CMC)
- Working set size (WSS)
- Cache line size (CLS)

Апробация алгоритмов

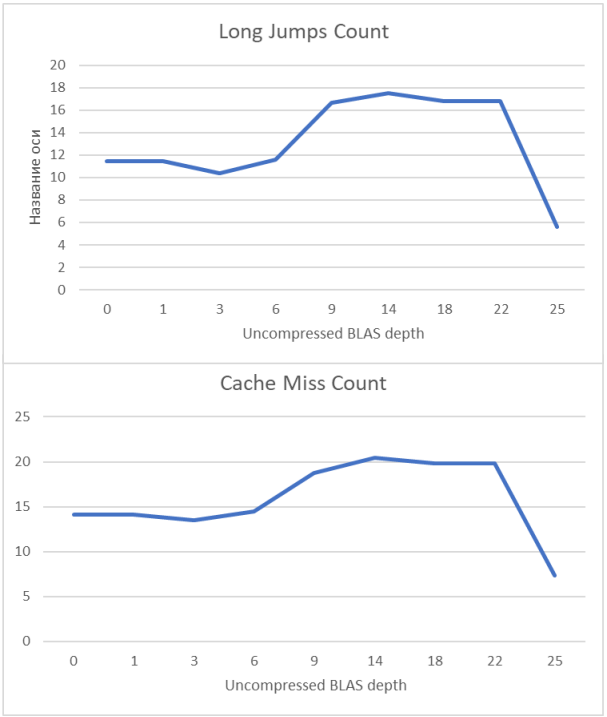


Размер BVH в сцене Village

Апробация алгоритмов



Метрики загрузки стековой памяти во время обхода BVH в сцене Village



Метрики кэш-эффективности во время обхода BVH в сцене Village

Апробация алгоритмов

Huawei Mate 30 Pro [Kirin 990]

FPS

Scene	BVH Layout	BVH32	BVH16	BVH16+ BVH8 [UMD =0]	BVH16+BV H8 [UMD =0] + Common buffer	BVH16+ BVH8 [UMD =0] +WW + Common buffer	BVH16+ BVH8 [UMD =13]	BVH16+ BHV8 [UMD =13] + Common buffer	BVH16+ BVH8 [UMD =13] + WW+ Common buffer	BVH16+ BVH8 [UMD =20]	BVH16+ BVH8 [UMD =20] + Common buffer	BVH16+8 [UMD =20] + WW + Common buffer
Village	DFS	12.5	13.9	1.7	2.3	1.3	5.6	7.4	4.6	6.2	8.4	4.8
Village	BFS	-	14.6	1.7	2.3	1.3	5.9	7.7	4.9	6.4	8.6	4.9
Dragon	DFS	12.9	13.5	2.2	2.5	2.1	7.2	7.4	7	5.7	9.4	5.4
Dragon	BFS	-	14.1	2.2	2.5	2.1	7.6	7.6	7.3	6.3	10.2	6.1
Sponza	DFS	8.4	12.6	1	1	1	3.1	3.7	3	3.5	4.3	3.2
Sponza	BFS	-	12.7	1	1	1	3.25	3.9	3.1	3.5	4.3	3.2

Апробация алгоритмов

Huawei Mate 30 Pro [Kirin 990]

Power consumption

Algorithm	Scene	Layout	avg fps	power consu-mption (mW)	% diff (mW)	normalized current (mA)	% diff (mA)
BVH32	village	DFS	10.78	1968.01	0%	518.16	0%
BVH16	village	BFS	15.48	1843.08	-6.30%	485.02	-6.30%
BVH16	village	DFS	15.49	1770.585	-10.06%	465.68	-10.20%
BVH32	dragon	DFS	13.65	1659.76	0%	436.78	0%
BVH16	dragon	BFS	14	1449.4	-12.60%	381.42	-12.60%
BVH16	dragon	DFS	14.05	1351.05	-18.56%	355.54	-18.50%

Заключение

В ходе работы были получены следующие результаты:

- Выполнен обзор существующих решений и алгоритмов по эффективной трассировке BVH
- Произведена оптимизация метода трассировки лучей, ориентируясь на мобильную платформу, а именно:
  - уменьшен объем памяти, занимаемый акселерационной структурой при помощи понижения точности координат BVH узлов до 50%
  - была применена арифметика с пониженной точностью с целью сокращения объема вычислений и ускорения траверсинга лучей
  - выбран наиболее оптимальный порядок расположения BVH узлов в памяти для улучшения переиспользования кэша
- Произведена реализация оптимизированного метода обхода акселерационной структуры данных на языке C++ с помощью технологий Vulkan, OpenGL ES и внедрение в существующее графическое приложение
- Реализованы графические пайплайны мягких теней и отражений в текущем графическом приложении
- Произведены эксперименты с применением оптимизированного метода обхода акселерационной структуры сцены в создании эффектов мягких теней, отражений