



Оптимизация загрузки homescapes

Особенности производительности мобильных игр

Обо мне

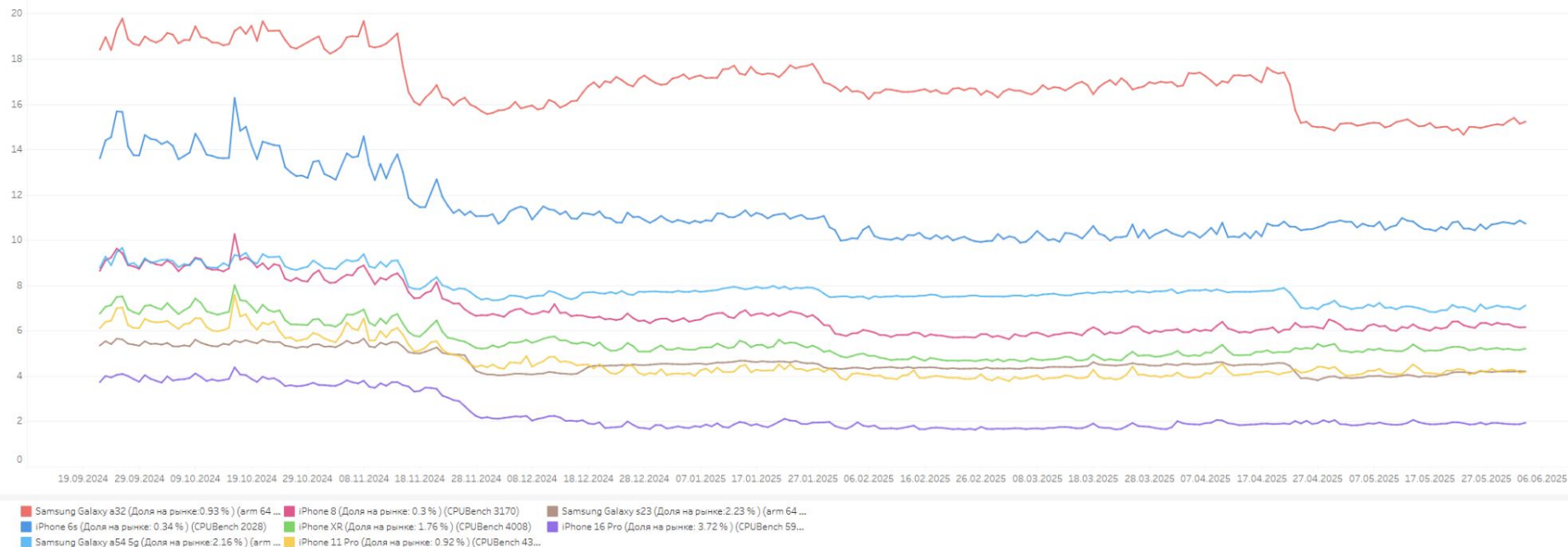


Lead developer, Playrix, Homescapes - tech team

- 14 лет в C++ и геймдеве
- 10+ успешных проектов в продакшене
- 6 лет в playrix:
 - Работал на движке ([Движок VSO: Под капотом нашего редактора](#))
 - Работал на Township
 - Работаю на Homescapes
- Пишу игровой движок мечты (<https://github.com/o2-engine/o2>)
- Пишу статьи на хабре (<https://habr.com/ru/users/anz/articles/>)

Результаты

Среднее ускорение по платформам - x2

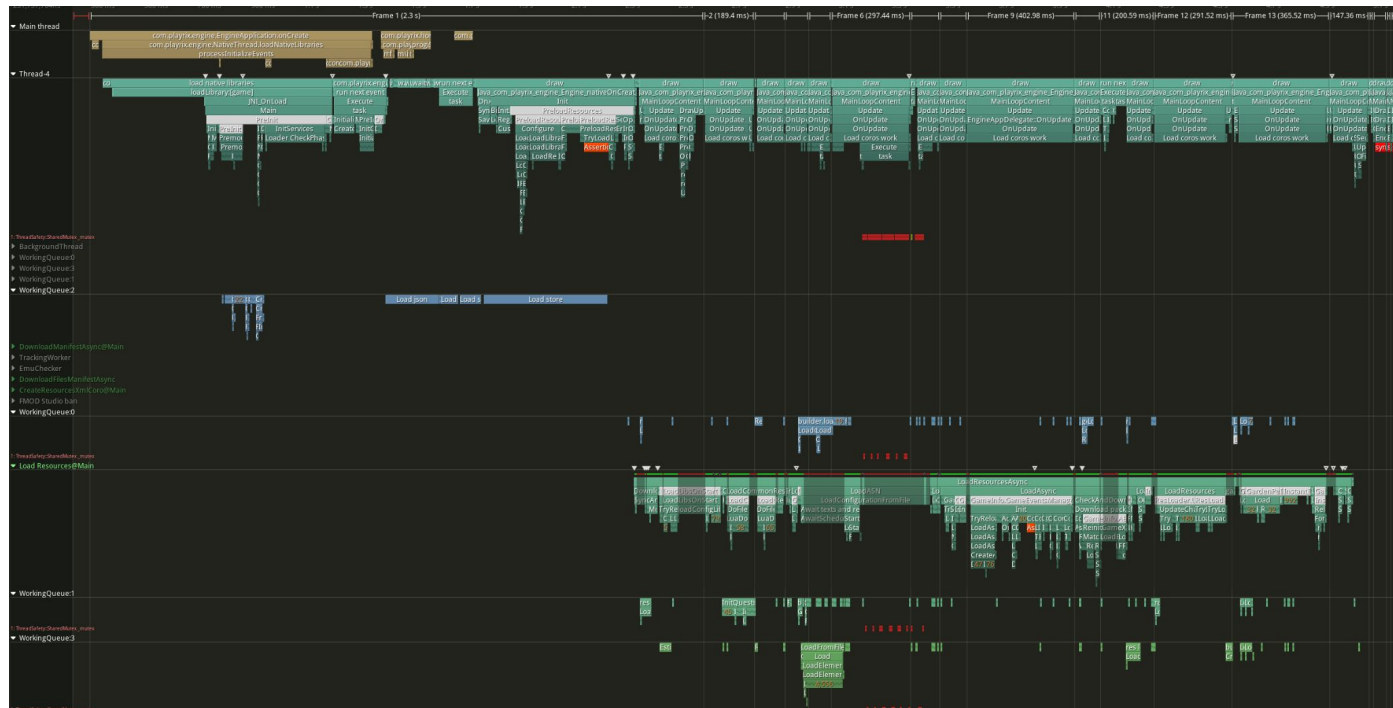


Наши особенности

- Зоопарк устройств
 - Разные платформы - ios, android, mac, uwp, web
 - Сотни моделей девайсов, десятки чипов
 - Даже самые проблемные приносят большой доход
- Огромная аудитория
 - 6 млн DAU (ежедневных уникальных игроков)
 - 30-40 млн MAU (уникальных игроков в месяц)
- Тонны легаси
 - 70 мб исходного кода (война и мир - 3мб)
 - 8ми летний проект, писавшийся сотнями людей, не всегда следовавших гайдлайнам

Что делали - профайлинг

Флоу загрузки нашей игры через tracy



Сеть - самая медленная часть

- Не у всех игроков есть хороший интернет
 - Игруют в метро, в дороге, за городом, где даже нет стабильного 4G
- Тестировать локально в офисе с wi-fi - не результативно
 - Пинг файла-манифеста 10-20кб в офисе - в пределах 100мс
 - Wi-fi в домике за городом - минимум 300мс
 - Статистика продакшена - секунды
- Кешируем все что можно
- Таймауты на ожидание
- Загружаем все в фоне наперед

Работа с файлами

- Работа с файлами через ОС - медленно
 - Положить все данные в архив, работать с ним, чтобы не упираться в ОС
- Разные алгоритмы сжатия
 - Разжатие - не бесплатно
 - Zip - не самый быстрый и эффективный. Используем zstd, исследуем другие
 - Читаем подряд, разжимая целые участки, напр. целые библиотеки ресурсов
- Предзагрузка и парсинг
 - Читать с диска - медленно. Можно прочитать заранее, и даже распарсить, если это xml/json

Многопоточность

- Тренд современных чипов - многоядерность
 - Ядра - не одинаковые. Производительных ядер мало
- Исторически игра была однопоточной
 - Проще писать однопоточную логику
 - 8 лет назад не было особо смысла в нескольких потоках
 - Тонны легаси содержат в себе тонны тротила, код не безопасный
- Затраты на синхронизацию, шеринг кеша.
 - Может случиться так, что многопоточная реализация медленнее однопоточной

Многопоточность

- Классическая стрельба по коленям
 - Словить дедлок в запутанной логике - не проблема в принципе
 - Зоопарк девайсов + Количество юзеров = креша даже в самых немыслимых ситуациях
 - Архитектура arm не защищает колени. Тесты на x86 могут пройти, но arm отстрелит
- Решение большинства проблем - джоб-система на нескольких тредах
 - Понятно для разработчика
 - Ложится на корутины
 - Безопасно на уровне архитектуры

Корутины

- Отличная вещь, если реализован хороший интерфейс
 - Отлично подходит для более читаемого флоу загрузки:
 - HTTP-запросы
 - Вынос работы в тред
 - Ожидание и синхронизация нескольких работ
 - Не завязывайте на главный поток - он станет узким горлышком
 - У нас главный тред “прокручивал” коллбеки корутин, но мог сам зависать
- Очень важны квоты на выполнение
 - Это не честная многопоточность, важно учитывать что корутины делят ресурсы треда

Снижение фреймрейта

- Апдейт кадра - 16мс
 - 100 кадров - 1.6 сек холостой работы вместо загрузки
- Экран загрузки преимущественно статичен
- Выход - снижение фреймрейта, квотирование загрузки
 - Меньше времени на отрисовку и апдейт - больше на загрузку
 - Зрительно не заметно
 - Останавливать все равно нельзя, иначе ОС посчитает что приложение зависло

Классика - не делай того, что можно не делать

- Тонны легаси содержали в себе тонны костылей
 - Распарсить одну и ту же xml 5 раз за загрузку - норма
 - Загрузка звуков, которые нужны “когда-то” в игре
 - Куча отладочной работы, которая вырезается для production
 - Просто лишние ресурсы и файлы, не нужны сразу на старте игры

Визуальные эффекты

- Анимация экрана загрузки может быть долгой
 - Пример: переход в стейт игры занимал 4 секунды.
 - Физически загрузка - 0.1 сек
 - Анимация загрузки - 3.9 сек

Финальные замеры

- Локальные замеры - половина дела
 - Измерять на РС - почти бесполезно
 - Замеры делать на high / low девайсах. Одни и те же оптимизации дают разные результаты на разных классах устройств
- Финальные результаты только с продакшена
 - Особенности поведения пользователей
 - Зоопарк девайсов
 - Разная доступность сети
 - Считаем результаты по 75му перцентилю