Слайд 1.

В данной презентации я постаралась рассмотреть наиболее важные приёмы оптимизации игр на примере игрового движка Unreal Engine.

Слайд 2.

Решаемая проблема достаточно понятна: любой разработчик игр во время разработки своего проекта сталкивается с рядом затруднений: игра занимает слишком много места на диске, долго запускается и/или имеет низкий FPS. Такой программный продукт вряд ли будет иметь большой спрос, ведь игроки предпочитают играть в качественные, но в то же время красивые и завораживающие своими эффектами игры, а не смотреть слайд-шоу с низким FPS.

Слайд 3.

Так вот, сейчас мы рассмотрим несколько важных инструментов оптимизации. Как всем известно, любой игровой продукт можно условно разделить на две взаимодействующие между собой сущности – графика и логика игры. Касательной логики, в каждом проекте всё индивидуально. Зачастую для ее оптимизации разработчики избегают операций, которые выполняются на основе тиков процессора. Представим, что у вас есть мини-карта, на которой отображаются положение персонажа относительно окружающего мира. В данном случае логику игры можно оптимизировать следующим образом: если герой не стоит на месте, а перемещается, то его новое местоположение на мини-карте следует отображать не каждый тик процессора, а с интервалом раз в 0,1 секунды. Это сильно облегчит компьютеру процесс обновления мини-карты.

Слайд 4.

Далее поговорим об оптимизации графики. Здесь есть где разбежаться, поскольку можно оптимизировать всё, что бросается в глаза.

Главный принцип, о котором стоит думать в каждую секунду разработки проекта: использовать меньше ресурсов для достижения того же результата.

Слайд 5.

► Использование «Фейков», которые в игровой индустрии достаточно популярны. Как примером применения инструмента можно назвать лучи света, которые проходят сквозь деревья или через окно в комнате. Частицами это не сделаешь, каким-то туманом это может быть дорого, шейдерами долго и муторно. Поэтому мы можем взять просто 2D-объект прямоугольник и наложить на него текстуру, похожую на блики. В результате получится тот же эффект.

Слайд 6.

► Оптимизация 3D-моделей включает в себя несколько подпунктов.

* (Слайд 7.) Уменьшение количества полигонов за счёт создание более дешёвого аналога. (Полигональная сетка — это совокупность вершин, рёбер и граней, которые определяют форму многогранного объекта в трёхмерной компьютерной графике.) Как мы можем судить по изображенным камням, на отрисовку третьего камня нам понадобилось всего 106 полигонов, что примерно в 1400 раз меньше, чем для первого. Да, качество заметно ухудшилось, но для побочного камня где-то на фоне сойдёт. Если посмотрим первые два камня, то отличий так таковых уже значительно меньше, хотя полигонов на втором изображении почтив 90 раз меньше, чем на первом.
* (Слайд 8.) Использование LOD’ов. (Level of Detail — такой приём в программировании трёхмерной графики, заключающийся в создании нескольких вариантов одного объекта с различными степенями детализации, которые переключаются в зависимости процентного соотношения занимаемого объектом места на экране и размера самого экрана.) В зависимости от занимаемого объектом места на экране (или от степени приближённости к камере) мы можем установить для него несколько состояний, каждом из которых степень детализации будет различной. Так, если камень из предыдущего примера стоит прямо перед наши, отрисовку можно оставлять как на первом картинке. В противном случае количество полигоном уменьшаем до разумного минимума.
* (Слайд 9.) Использование normal map, или нормал карт. К примеру, каменная поверхность. Вместо того, чтобы прокладывать каждый камушек самостоятельно и как отдельный объект, мы накладываем на текстуру нормал карту. В результате на плоской поверхности явно наблюдается эффект рельефа.

► (Слайд 3. с отсылкой на отсутствие скринов.) Минимизация элементов окружения. Если у вас на сцене есть какая-нибудь вещь, которая для игроков не играет никакой роли, её можно просто удалить, чтобы она не занимала никаких ресурсов в проекте. Особенно важно учитывать внимание игрока. Это касается прямолинейных участков в игре, где игрок будет просто бежать по тому же коридору, едва ли рассматривая со снайперской винтовкой, что находится слева и справа от него. В 80% случаев фон остается незамеченным.

Слайд 10.

► Оптимизация частиц.

* Один из дорогих элементов графики. Неправильное использование частиц может свести производительность проекта в ноль. Не стоит делать частицы слишком большими, поскольку если частицы занимаю очень много места на экране, каждый пиксель стоит отрэндрить очень дорого. Поэтому используйте меньше частиц и сокращайте занимаемое ими место на экране.
* Устанавливайте границы эммитора. (Эммитор – область прогрузки частицы.) Если ваша камера видит границы, то она будет рендрить и обрабатывать и сами частицы, что не есть хорошо для производительности. На рисунке показан не самый лучший пример, здесь границы эммитора можно уменьшить.
* Используйте меньше дополнительных источников освещения на частицах. Не стоит извращаться и допускать, чтобы, к примеру, каждая искра испускала свой же источник света. Из-за большого количества таких источников сильно проседает FPS.

Слайд 11.

► Оптимизация при создании уровня. Конечно, здесь всё очень зависит от самого проекта, его стилистики и т.д. Для конкретного уровня оптимизация уровня всегда разная. Но сейчас мы посмотрим оптимизацию на примере открытого пространства.

* (Слайд 12.) Всё, что находится очень далеко и игрок этого не видит, стоит обрезать, ту же траву, камни и т.д., за счёт чего повысится производительность.
* (Слайд 13.) Использование instance – объектов. Речь здесь идёт о foliage – инструменте, с помощью которого можно расставлять траву, деревья, кусты и прочие элементы ландшафта. Это не только ускорит время расстановки всей этой растительности, но и повысит производительность сцены, особенно открытых пространств, где таких объектов очень много.
* (Слайд 14.) Объединение деталей в одно целое. Допустим, у нас есть какой-нибудь фоновый дом, у которого окна отдельно, двери отдельно, крыша отдельно, хотя нам всё это вовсе и не нужно в отдельном виде. На примере изображены домики, которые будут просто красоваться где-нибудь на фоне. Соответственно, мы их объединяем в одно целое, чтобы при рендринге оно обрабатывалось как единый объект, за счёт чего быстрее. Делается это с помощью инструмента Actor Merging.

Слайд 15.

► Динамическое освещение. Что касается самих источников освещения, то стоит их использовать в меньшем количестве. Следовательно, меньше источников освещения с тенями. Это очень важно, поскольку тени могут оказаться чуть ли не самыми дорогими в проекте. Обычно, когда есть открытое пространство, то с тенями делать рекомендуется только солнце или луну. У всего остального лучше отключить тени, иначе у проекта, опять же, произойдёт падение производительности.

Слайд 16.

► Профайлинг.

* Есть очень полезная команда ProfileGPU. Она выводит окно, которое вы можете видеть справа, в котором отображается список, что за сколько времени отрендрилось. Т.е. за сколько рендрятся объекты, динамическое освещение, частицы, туман и т.д.
* Консольная команда stat SceneRendering позволяет в реальном времени обрабатывать и смотреть, что у нас и за сколько рендрится. Если ProfileGPU останавливает кадр, снимает, грубо говоря, информацию о нём, а выведенном окне на эту информацию печатается, то SceneRendering команда работает в реальном времени.

Слайд 17.

Что касается моего личного опыта, то я решила оптимизировать пользовательскую карту “Forest”. Здесь я использовала instance – объекты, LOD’ы, а также настроила дальности отрисовки.

* В качестве instance – объектов здесь выступает все возможная растительность и камни. Материал накладывается не на каждый объект как отдельную сущность, а на все сразу.
* (Слайд 18.) Касательно LOD’ов здесь всё понятно. Чем меньше размер камня на экране, тем меньше и количество полигонов.
* (Слайд 19.) Дальность отрисовки. Тут вы можете наблюдать, что, находясь прямо на поляне, мы видим бурную растительность вокруг камней, которая исчезает сразу же, как только мы уходим в лес, т.е. отдаляемся.

Слайд 20.

В результате оптимизации FPS поднялся на 12, а количество времени, затрачиваемое на рендринг уменьшилось с 58 до 25 мс, что уже не мало. В принципе, цель достигнута.

Слайд 21.

Спасибо за внимание.