The Ray-Trace Game

Часть I. Общие сведения

Предыстория

Микропроцессорная техника развивается невероятными темпами. Тем не менее геймеры всего мира, затаив дыхание, который год ждут революции в игровой индустрии: когда наконец-то графика станет реалистичной, когда настанет эра raytracing'a? Чтобы хоть немного приблизить чудесное будущее, было решено создать игру с простой графикой, но на основе метода трассировки лучей.

Про что игра? Про трассировку луча, естественно.

Краткое описание

На каждом уровне есть источник лазерного луча, приёмник, зеркала, "фишки", стены и другие объекты. Цель – поворачивая зеркала, необходимо "поразить" лучом все фишки (пораженная фишка исчезает с поля), после этого исчезает объект, препятствующий прохождению луча к приёмнику. Для завершения уровня необходимо направить луч в приёмник.

Сложности: лазер имеет ограниченный заряд (по времени); установка может выйти из строя из-за перегрева (когда отраженный луч возвращается в источник) и другие.

Интерфейс

На основном экране изображена схема установки и параметры (заряд, температура). Можно включить дополнительный экран – "Микроскоп". На этом экране будет отображаться участок схемы методом трассировки лучей.

Установка

Игра не требует установки, можно запускать сразу после распаковки. Для работы не требуется наличия DirectX, OpenGL и других дополнительных библиотек.

О технологии

Главным желанием в процессе разработки было продемонстрировать технологию ray-tracing (которая используется при создании мультфильмов и спецэффектов в кинофильмах) и проверить, готовы ли современные процессоры к широкому её внедрению. Оказалось, что для несложных сцен (как в данной игре), процессорной мощности современных компьютеров вполне достаточно. Основной плюс технологии – реалистичное освещение, блики и тени, корректное отображение прозрачных (линз) и зеркальных тел.

Часть II. О разработке

Рабочее окружение и сроки

Разработка была начата 9 июня, велась, в основном, по вечерам после работы. На разработку первой версии “микроскопа” ушло 5 вечеров.

Ещё несколько дней в июле и августе ушло на придание игре завершенного вида (отображение в микроскопе источника и приёмника, режим сглаживания).

В качестве рабочих инструментов использованы Intel C++ Compiler, FAR Manager, Paint.Net, Blender, Adobe Audition.

Использованы звуковые материалы с сайта freesound.org и композиции группы Pink Floyd (что придаёт особый колорит).

Первые трудности

Поскольку было решено отказаться от использования «внешних» библиотек, то некоторые трудности появились ещё в период разработки графического интерфейса.

В основном меню игры, в схематическом отображении поля, в таблице рекордов, на экране "о лаборатории" используются различные графические эффекты (полупрозрачность, "частицы"). Замер производительности показал, что на обработку элементов меню уходило огромное количество процессорного времени. Первая оптимизация позволила сузить область применения эффекта, что незамедлительно дало четырехкратный прирос производительности.

Вторая оптимизация была хитрее. Анализ генерируемого компилятором кода показал, что выполнение простого действия — покомпонентного умножения цвета пиксела на число — влечёт за собой ненужные операции знакового расширения числа и знакового сужения. Это происходило из-за того, что промежуточный результат вычислений имел тип «байт».

Небольшое изменение алгоритма позволило отказаться от таких преобразований и вместо 3-х операций умножения (по одной на каждую компоненту цвета) использовать две. В результате производительность повысилась ещё в несколько раз.

Пример кода, выполняющего смешение двух цветов:

```
DWORD vF0F = ((((v1 & 0xFF00FF) * zz) + ((v3 & 0xFF00FF) * z)) >> 8) & 0xFF00FF;
DWORD v0F0 = ((((v1 & 0xFF00) * zz) + ((v3 & 0xFF00) * z)) >> 8) & 0xFF00;
DWORD v = vF0F | v0F0;
```

Многопоточность

Трассировка лучей – ресурсоёмкая задача, которая получает огромный выигрыш при использовании многоядерных систем. Для реализации многопоточной обработки используется библиотека Intel TBB.

Библиотека Intel TBB была скомпилирована и слинкована статично, чтобы избежать зависимостей выполняемого файла от других продуктов (MSVC redistributable).

Задействованы неделимые (atomic) операции для синхронизации данных основного цикла игры и главного потока визуализации.

Параллельные потоки занимаются визуализацией строк, память под строчки результатов выделена с выравниванием на строки кэша (cache aligned). Таким образом, выполнение потоков не сопровождается накладными расходами на поддержание когерентности кэша.

Для уменьшения времени "холостого хода" (когда нет новых задач, и некоторые потоки уже закончили работу) использована особенность "микроскопа". Микроскоп имеет круглое поле отображения. То есть строки имеют длину. Если раздавать задачи потокам в порядке уменьшения длины строки (и, в среднем, объёма вычислений), то разброс времени окончания вычислений заметно сокращается.

Для поиска объектов, пересекающихся с лучом отображения, использована особенность расположения объектов. Объекты располагаются в одномерном слое смежных кубов, что даёт хорошее разбиение пространства даже без построения дополнительных структур (kd-tree, …).

Нестандартный подход

В июле в ISN была опубликована статья в которой рассказывалось о том, что использование PGO (profile guided optimization) позволяет в большинстве случаев добиться повышения производительности на 10%. Испытания показали, что

использование этой технологии напрямую даёт желаемый результат не всегда.

Дело в том, что разные режимы отображения в игре задействуют разные участки кода. Это сбивало оптимизатор с толку. Было решено сделать следующее:

1. заменить условия, проверяющие опции отображения, на условную компиляцию
2. перед началом компиляции создавать копии файла с классом
3. подключать исходный код этих классов с разными определениями

Таким образом исходный код одного класса порождал четыре класса (то есть умножался). В зависимости от опций отображения выбирался класс, осуществляющий отображение. При таком подходе PGO собирал отдельную статистику по методам разных классов, что положительно сказалось на результатах.

Что дальше

Дальнейшее развитие игры предполагает добавление новых уровней, опций визуализации и графических эффектов. Поскольку нет завязок на графические библиотеки, то, возможно, игра будет переписана под SDL и станет в кросс-платформенной.