проект пинбол

Логика игры:

В игре есть мяч – один пиксель, он характеризуется 2 8-и битными координатами, а также 2 8-и битными скоростями движения по каждой координате. Мяч ускоряет гравитация. Первые 6 бит координат мяча считаются целой частью, а остальные 2 – дробная часть, для большей реалистичности движения. Скорость вообще имеет дробную часть в 5 бит.

Также есть «объекты» - стены. Это отрезки в 4 возможных состояниях: горизонтальные, вертикальные, и 2 вида диагональных под углом 45. Каждая стена характеризуется 2 битами типа, двумя 6-и битными координатами левого нижнего угла и размером до 5-и бит. При столкновении со стеной мяч отскакивает и увеличивает текущий счёт игры на 1.

В основном все стены статичны, но есть 2(или 4), которые игрок может включать и выключать по кнопке.

Задача игрока – не дать мячу вылететь за пределы экрана.

Так как 1 экран максимального размера 32\*32 слишком мал для пинбола, мы сделали один большой экран 64\*64 из четырёх маленьких. В целях оптимизации на каждом экране максимум 1 управляемая стена и всего максимум 8 стен.

Устройство графической части.

В первую очередь была написана схема, конвертирующая стены в столбцы из пикселей. На вход подаётся 1 объект, координата столбца, в который нужно записать пиксели от новой стены, и состояние столбца до обработки стены.

В таком состоянии основная схема должна перебирать каждый столбец, и для каждого столбца перебирать каждую стену. Кроме того, надо хранить состояние всех столбцов, так как одновременно обрабатывается только 1 из них.

Это работает невероятно медленно. Одно заполнение экрана происходит за несколько секунд. Для ускорения можно не вычислять заново пересечения столбцов со стенами, а запоминать их после первой обработки. Если запустить этот процесс 1 раз в начале, а после этого использовать запомненные значения просто как «фон», то будет быстрее.

Кроме статичного поля есть ещё мяч и включаемые/выключаемые (управляемые) стены.

Схема graphics main получает на вход состояние мяча, 5-и битную координату столбца, который сейчас будет отрисован (это замедляет отрисовку в 32 раза с точки зрения тактов, но сами такты не тратят так много времени. Когда я пробовал запускать отрисовку 32 столбцов за раз, всё работало не в 32, а сразу в 100 раз медленнее), а также состояние фона столбцов с этим номером для каждого из 4 экранов, которые предпосчитаны и хранятся в 4 отдельных RAM. Возможно, позже я попробую переделать это, чтобы координаты столбцов были до 128 и 4 экрана рисовались не одновременно, а по очереди.

Также необходимо рисовать управляемые стены на экранах. Есть 2 варианта: один из них – честно рендерить эту стену каждый кадр. Это не так плохо, так как стена всего 1 на экран, а значит для каждого столбца рендер запускается всего по разу. Но схема всё равно довольно толстая, и может ещё замедлить систему. Вариант 2 – сделать второй предподсчёт, и хранить для каждого столбца по 2 состояния: с включенной стеной и без. Но тогда придётся использовать аж в 2 раза больше RAM, а это как-то неприлично.

Сейчас макс скорость графической части достигается при 512Hz, поэтому на старте стоит замедлитель в 8 раз, который переводит часы с 4KHz в 512Hz. Поэтому можно запускать CDM-8 на частоте 4000.

Устройство физики мяча

Каждый такт скорость мяча по оси y увеличивается на 1 (гравитация), если переводить это в пиксели, то на 1/32 пикселя в такт. И каждый такт к координатам мяча прибавляется его скорость (с битовым сдвигом на 3). Сейчас аппаратно сделан костыль для тестирования системы, симулирующий часть физики – если мяч должен перескочить с самой левой части экрана в правую или наоборот – то его скорость по оси x инвертируется. Аналогично для оси y. В будущем выполнение этого условия означает конец игры, ведь мяч вылетел за пределы поля.

План: написать для CDM-8 программу, которая проверяется столкнулся мяч с данной на вход стеной или нет. Программа принимает на вход 32 бита мяча и 2 + 24 бита стены (координаты переводятся в 8-и битные, потому что assembler работает с 8-и битными числами.) На выход программа должна выдать столкнулся ли мяч с этой стеной, а возможно ещё и обновленные координаты. Так как каждая стена существует только в 1 мониторе, то нужно перебирать по 8 стен перед тем, как действительно двигать мяч. Возможно, это уже будет катастрофически долго, потому что сейчас у нас всего 8 тактов между движениями мяча, а моя пробная программа для проверки столкновения уже работает около 64 (в худшем случае, а в лучшем за 8), и это только с 1 стеной! В теории, если бы CDM-8 действительно работал в 4KHz, то получается 4000/8/32 = 16 фпс. Для Logisim это даже неплохо, но такой скорости достичь очень вряд ли получится.

Варианты оптимизаций:

Отслеживать столкновение сразу с крупным сдвигом мяча. Сейчас за 1 такт мяч двигается на совсем чуть-чуть, иногда даже меньше чем на 1 пиксель на экране. Возможно, если двигать его сразу на большие расстояния предсказывая его следующую позицию, то погрешность увеличится не так сильно. Зато позволить сильно меньше зависеть от скорости программной части, например, проверяя столкновение заранее на 4 или даже 8 тактов вперёд. И если оно будет, то можно начать уже более точно вычислять, когда и где.

Правильно расставлять стены. Проверка сталкивается мяч со стеной или нет – это кучка ифов, а значит, если расставить их в правильном порядке, то программа будет быстро обрываться, и в идеальном случае она работает за примерно 8 или 16 тактов.