Collision detection

Вычисления на видеокартах. Лекция 5

Sort and Sweep
Parallel Spatial Subdivision
Radix sort

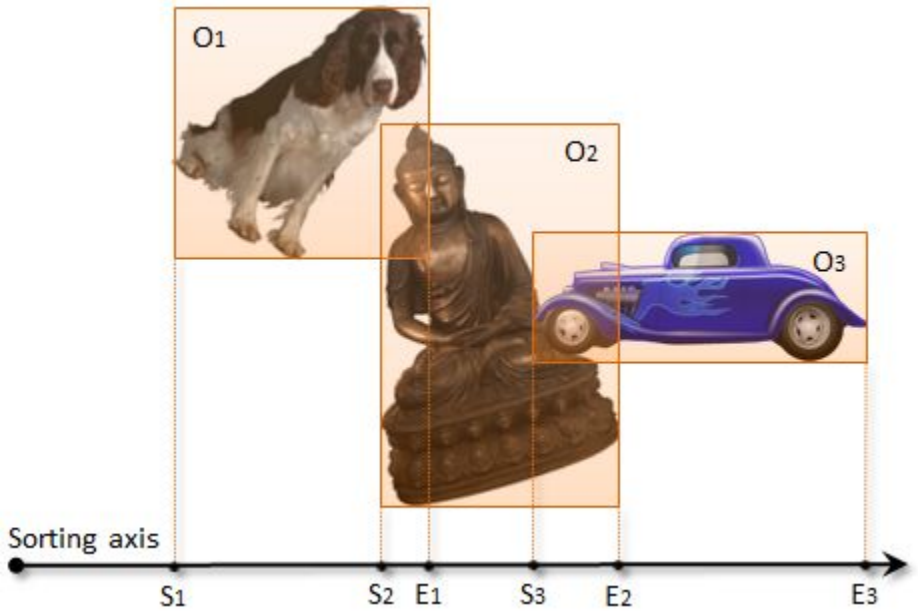
Полярный Николай

polarnick239@gmail.com

Collision detection

Пусть дано множество объектов содержащихся в своих axis-aligned bounding box (AABB).

Нужно найти все пары потенциально пересекающихся объектов, т.е. объекты чьи AABB пересекаются.

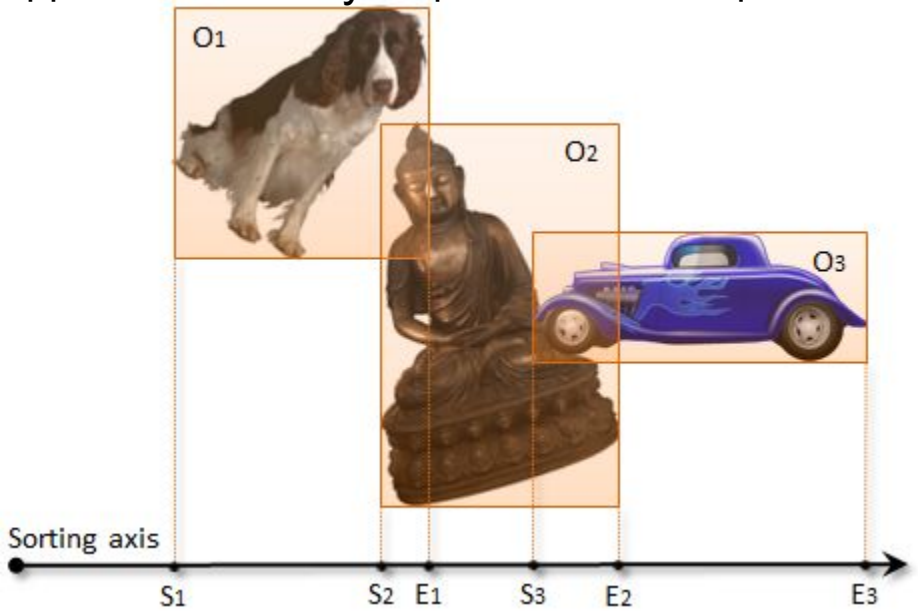


Collision detection: Sort and Sweep

Простой способ - спроецировать AABB на ось, отсортировать события “начался AABB объекта X” и “закончился AABB объекта X” по этой оси.

Пробежавшись по событиям за один проход поддерживая список AABB содержащих текущую скользящую точку - найти пересечения.

Источник параллелизма - стартовать из каждой точки начала AABB независимо вплоть до соответствующей точки конца AABB.

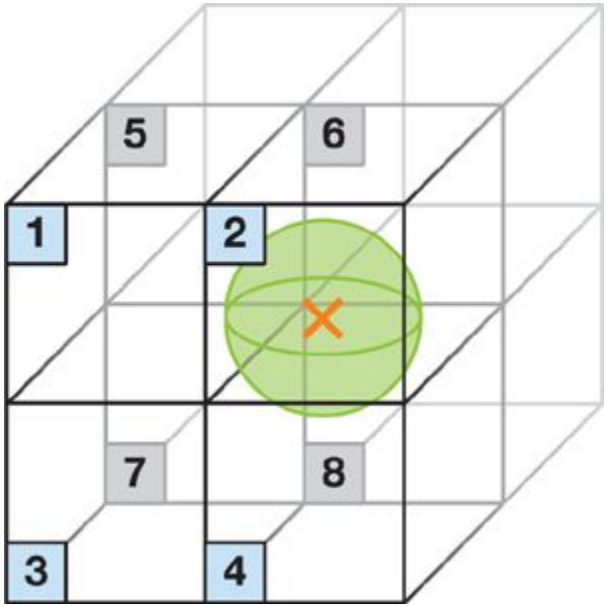


Spatial Subdivision

Создали регулярную решетку. Размер ячейки - максимальный размер объектов.

Два варианта:

1. Ячейка хранит список всех объектов чьи центры лежат внутри ячейки
2. Ячейка содержит список всех объектов чьи AABB пересекают ячейку

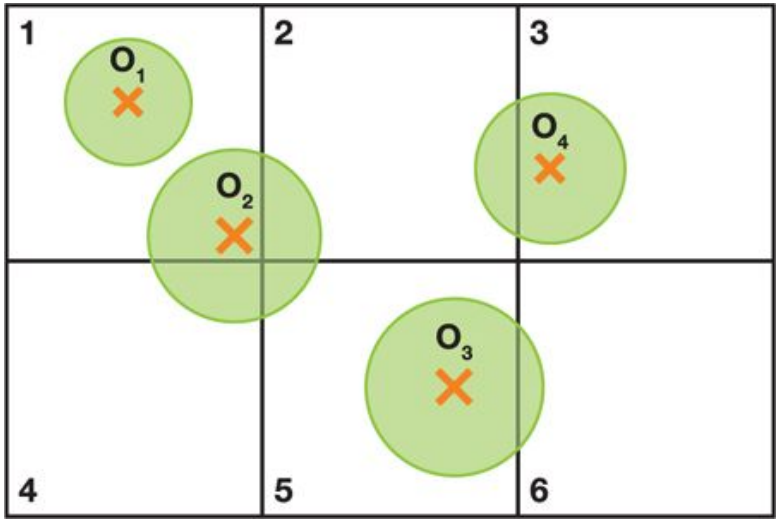


Spatial Subdivision

Пусть ячейка содержит список всех объектов чьи AABB пересекают ячейку.

Тогда пересекать достаточно лишь пары объектов которые:

- Попали в одну ячейку (пересекают ее своими AABB)
- Хотя бы у одного из пары центр лежит в этой общей ячейке

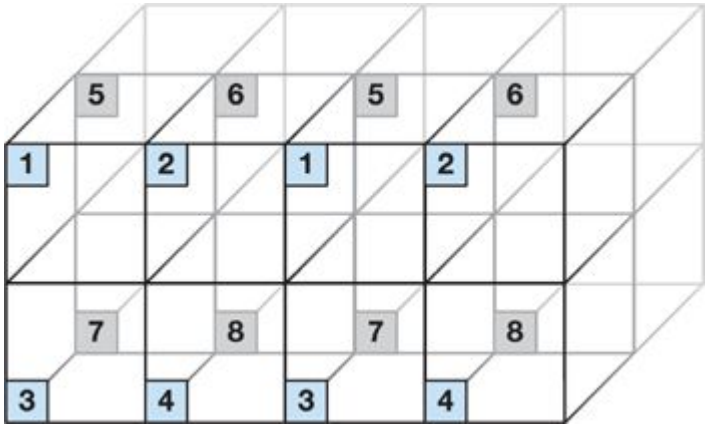


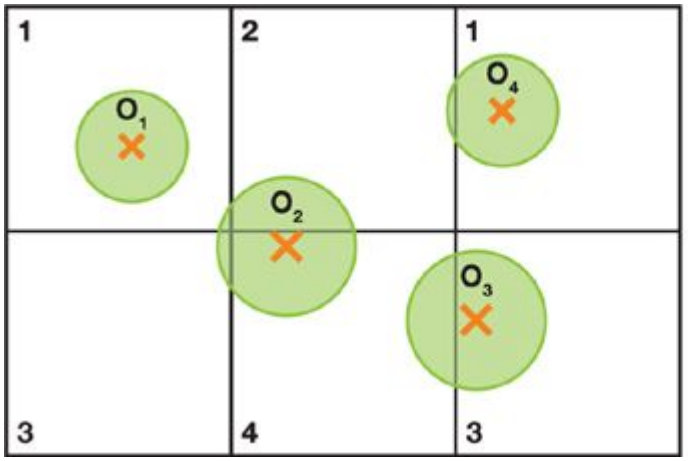
Алгоритм хорошо работает лишь в предположении что все объекты похожего размера.

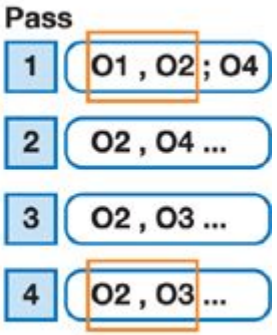
Parallel Spatial Subdivision - опасности

При обновлении состояния объекта может случиться гонка (если он пересекся больше чем с одним объектом).

Поэтому ячейки красятся шахматной раскраской и в одну параллельную обработку обрабатываются ячейки лишь одного и того же типа.







Так же одну и ту же пару можно засчитать дважды, если их центроиды в разных ячейках.

Parallel Spatial Subdivision on GPU

1. Создаем список пар:
   - ID объекта
   - ID ячейки

При этом каждый объект состоит в стольки парах, сколько ячеек он пересекает.

Parallel Spatial Subdivision on GPU

1. Создаем список пар:
   - ID объекта
   - ID ячейки

При этом каждый объект состоит в стольки парах, сколько ячеек он пересекает.

Лучше создать не один массив из пар чисел, а два массива чисел. Т.к. тогда дальнейшая работа будет ближе к идеальному coalesced memory access.

Parallel Spatial Subdivision on GPU

1. Создаем список пар:
   - ID объекта
   - ID ячейки

При этом каждый объект состоит в стольки парах, сколько ячеек он пересекает.

Лучше создать не один массив из пар чисел, а два массива чисел. Т.к. тогда дальнейшая работа будет ближе к идеальному coalesced memory access.

2. Сортируем список по ID-ячейки. (Radix sort)

Parallel Spatial Subdivision on GPU

1. Создаем список пар:
   - ID объекта
   - ID ячейки

При этом каждый объект состоит в стольки парах, сколько ячеек он пересекает.

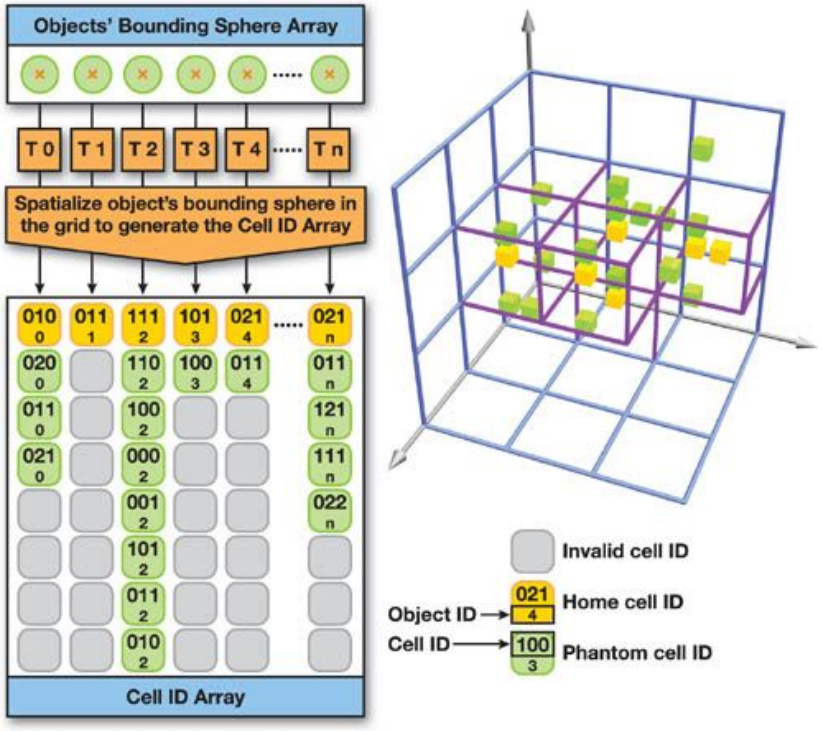
Лучше создать не один массив из пар чисел, а два массива чисел. Т.к. тогда дальнейшая работа будет ближе к идеальному coalesced memory access.

2. Сортируем список по ID-ячейки. (Radix sort)

3. Проверяем сгруппированные по ID-ячеек объекты на взаимные пересечения.

Создаем список пар Объект-Ячейка

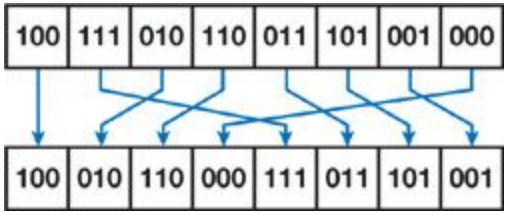
1. Для каждого объекта посчитать сколько ячеек пересекает AABB
2. Reduction для нахождения префиксных сумм по числу пересекаемых ячеек
3. Для каждого объекта записать пары с пересекаемыми ячейками по offset совпадающему с соответствующей префиксной суммой найденной на предыдущем шаге



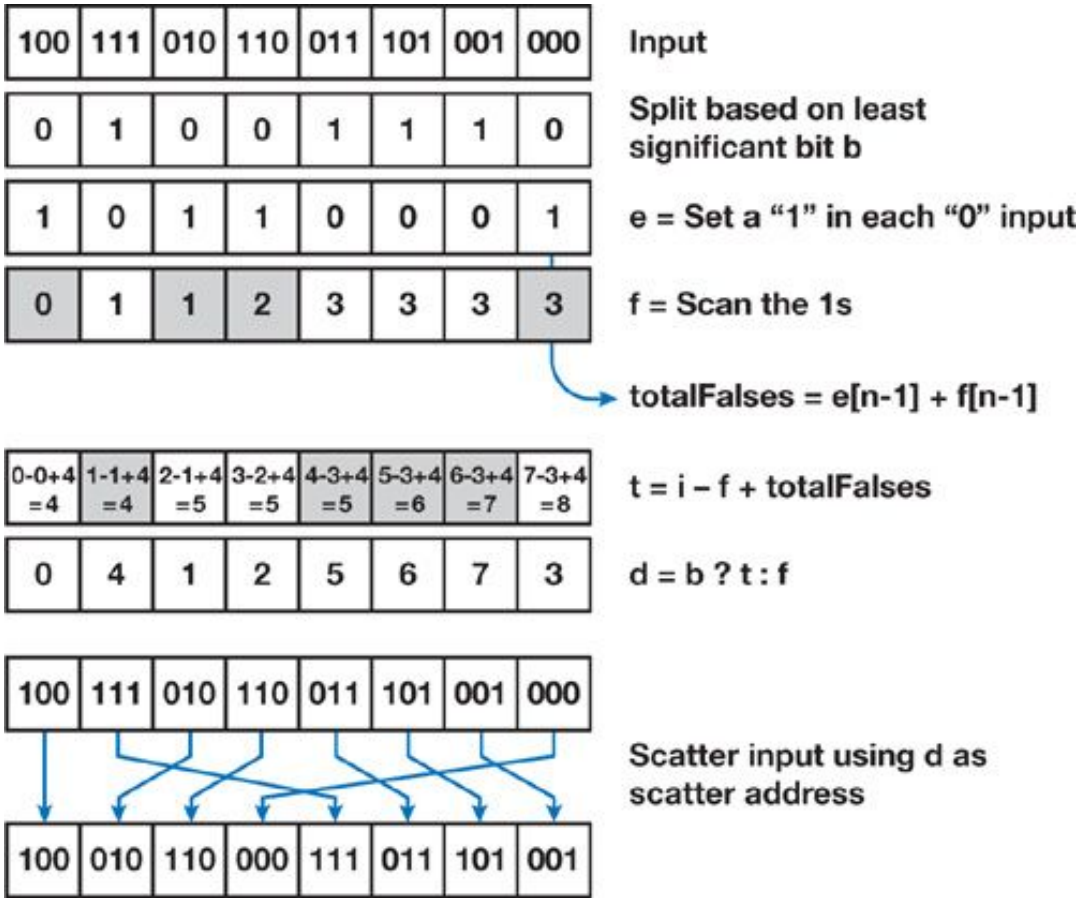
Radix sort

Сортируем в несколько проходов.

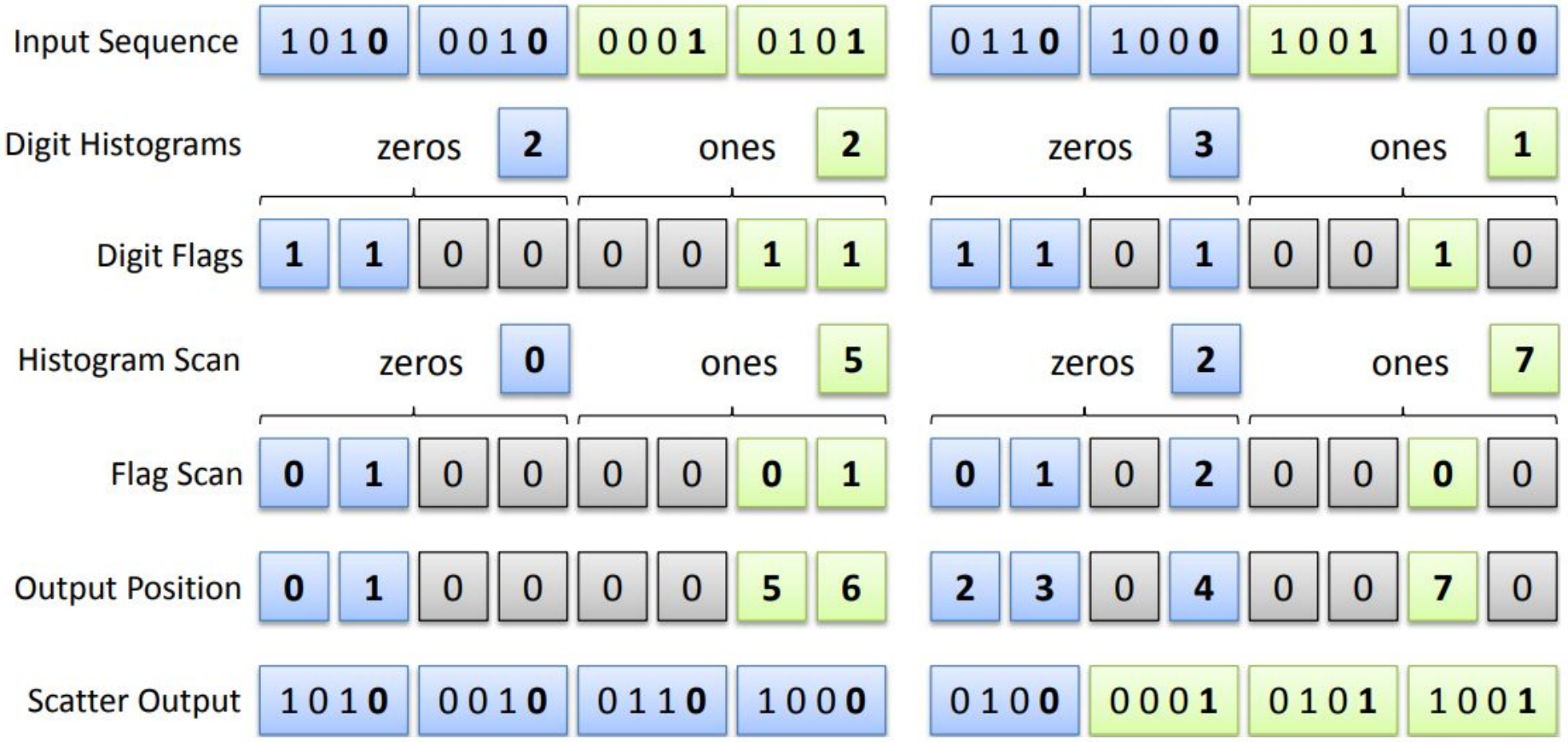
Сначала хотим отсортировать по младшему разряду:



Radix sort: Local

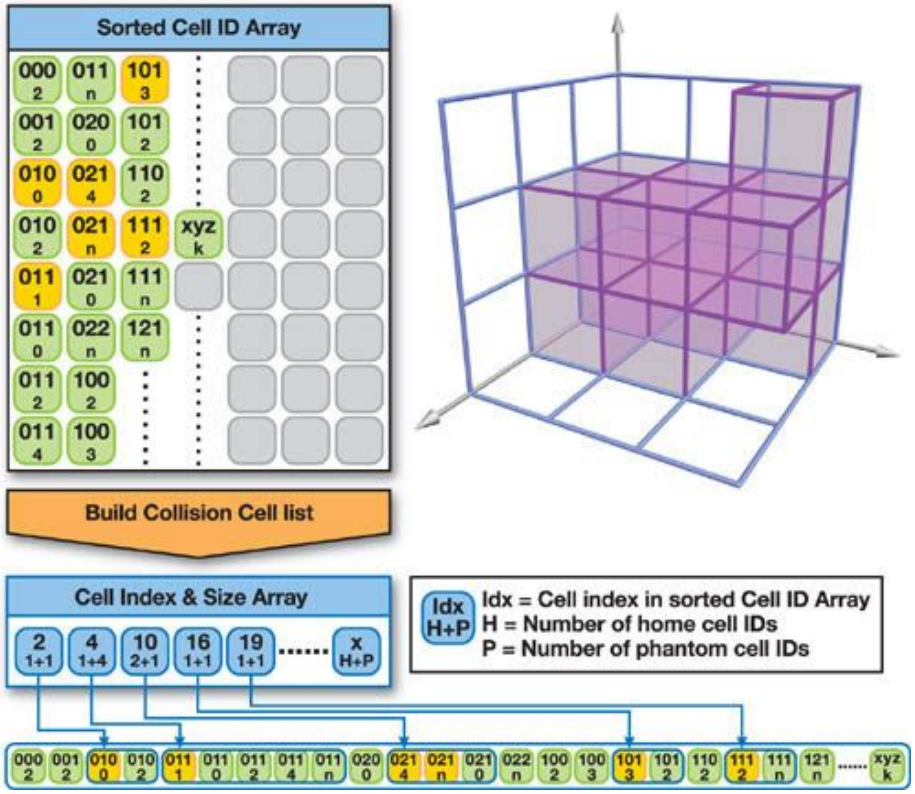


Radix sort: Global



Считаем число объектов в каждой ячейке

1. Считаем число объектов в каждой ячейке
2. Получили индекс для первого объекта каждой ячейки и ее размер
3. Обработаем каждую ячейку одним/несколькими потоками



Ссылки

- https://devblogs.nvidia.com/thinking-parallel-part-i-collision-detection-gpu/
- https://developer.nvidia.com/gpugems/GPUGems3/gpugems3_ch32.html
- https://developer.nvidia.com/gpugems/GPUGems3/gpugems3_ch39.html
- Fast Radix Sort for Sparse Linear Algebra on GPU, Polok et al, 2014