Accelerating $k^+$-buffer using efficient fragment culling

Andreas-Alexandros Vasilakis and Georgios Papaioannou
{abasilak,gepap}@aueb.gr

Department of Informatics,
Athens University of Economics and Business, Greece
A number of image-based applications require operations on more than one (maybe occluded) fragment per pixel:

- **Photorealistic Rendering**
  - [global illumination]
  - [order-independent transparency]
  - [shadowing]

- **Visualization & Processing**
  - [flow], [molecular], [hair], [solid] geometry
Prior Art

1. A-buffer methods

- capture [all] fragments per pixel → [sort] them by depth
- [memory overflow] & [fragment contention]
1. **A-buffer methods**
   - capture [all] fragments per pixel → [sort] them by depth
   - [memory overflow] & [fragment contention]

2. **k-buffer methods**
   - capture [k-closest] ones → [reduce] memory & sorting costs
   - prior solutions suffer from
     1. [RMW hazards], [geometry pre-sorting], [k < 32]
     2. [extra rendering pass], [depth precision conversion]
     3. [unbounded memory]
### Prior Art

#### 1. A-buffer methods
- capture [all] fragments per pixel → [sort] them by depth
- [memory overflow] & [fragment contention]

#### 2. $k$-buffer methods
- capture [[$k$-closest] ones → [reduce] memory & sorting costs
- prior solutions suffer from
  1. [RMW hazards], [geometry pre-sorting], [$k < 32$]
  2. [extra rendering pass], [depth precision conversion]
  3. [unbounded memory]
- $k^+$-buffer solution (*I3D’14, Vasilakis & Fudos*)
  - [overcomes] all previous limitations
  - [fragment culling] & [pixel synchronization]
Accelerating $k^+$-buffer

$k^+$-buffer’s fragment culling

- Concurrently [discards] an incoming fragment that is farther from all currently maintained fragments (guided by the [max element]).
- [Limitations]
  1. Depends on the [fragment arrival order].
  2. Requires $k^+$-buffer to [be initially filled] before starts culling.
  3. Fragment elimination is performed inside the [pixel shader].
Accelerating $k^+$-buffer

$k^+$-buffer’s fragment culling

- Concurrently [discards] an incoming fragment that is farther from all currently maintained fragments (guided by the [max element]).
- [Limitations]
  1. Depends on the [fragment arrival order].
  2. Requires $k^+$-buffer to [be initially filled] before starts culling.
  3. Fragment elimination is performed inside the [pixel shader].

Ideal fragment culling

- Knowing the exact depth of the $k$-th fragment a priori allows us to insert all fragments with $\leq$ depth in constant time, discarding the rest (farther ones).
Accelerating $k^+$-buffer

Our Idea:

- Approximate this value via *fragment occupancy maps* !!!

$k^+$-buffer culling | our culling
---|---
63.66% | 98.28%