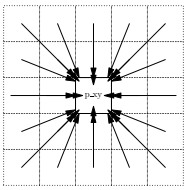
Tiled Depth of Field Splatting

Kai Selgrad, Linus Franke, Marc Stamminger
University of Erlangen-Nuremberg, Germany

LGDV COMPUTER GRAPHICS • ERLANGEN

Fast Depth of Field Rendering

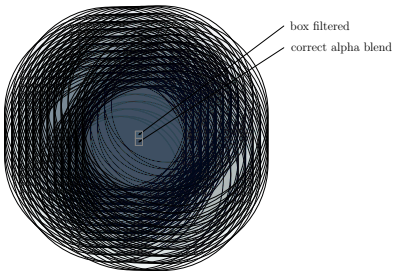
Gathering approaches [RTI03] are very fast, but suffer from from incorrect blending:



Reducing leakage [RTI03, ZCP07] and separating near-field and far-field [Dem04, Ngu07] has been tackled in previous work. Our method further provides for blending using correct fragment order by employing recent advances in particle splatting.

Blending vs Filtering

With a simple bilateral filter it is possible to distinguish near- and far-field, but the order of fragments accumulated in either of those segments is determined by the filter implementation, not their relative distance to the observer.



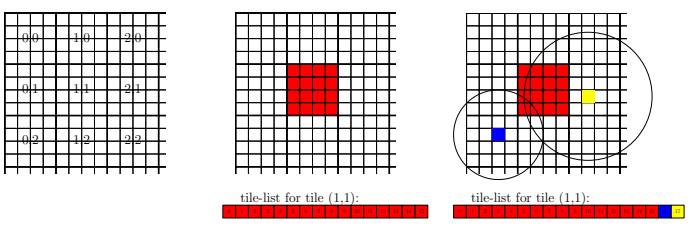
This can lead to noticeably different results as the following example from the balcony scene (right) shows.
Approaches that use blending are typically scattering approaches that splat the blurred (enlarged, semi-transparent) fragments [PC81, KZB03]. However, this requires that a global sorting of all the rendered fragments along z is established, which is a costly operation.

Efficient Particle Rendering

We rely on tiled particle accumulation as described by Thomas [Tho14], an extension of Forward+ [HMY12].
In this approach particles are splatted to a tiled screen buffer which is sorted in parallel and then traversed for each pixel to accumulate the particles in proper z-order.

Tiled Depth of Field Splatting

We also keep tile-lists (image below, left) which we seed with the $n \times n$ fragments present in the framebuffer area overlapping each tile (image below, center). This is a simple operation where each fragment has a predetermined storage location. We then compute the circle of confusion for each fragment and enter it in the lists of all neighboring tiles that it overlaps (image below, right). These lists can then be accumulated very fast.



Implementation Details

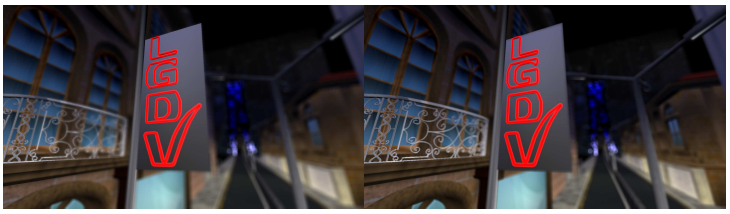
- Tile size of 16×16, avg. list length 1100 elements
- Use of atomics to enter data in adjacent tiles is very fast
- Tile-parallel radix sort using CUB
- Tight 16-byte format (16bit colors and positions, 32bit CoC)
- Accumulation: all threads in a warp traverse the same tile list
  Chunks of the tile-buffer are loaded to shared memory
  $\rightarrow$ coherent loads, coherent traversal

Fragment Merging

Before fragment distribution it is possible to merge spatially close fragments to reduce sorting and accumulation load. Merging 2×2 fragments does not noticeably change image quality, while merging 4×4 fragments produces minor errors. However, with this more aggressive setting our method is as fast as filter based approaches.
The following table shows the influence of fragment merging on list length and performance:

Merge	—	2×2	4×4	Gathering
Average list length	1160 entries	384 entries	199 entries	—
Build lists	1.35 ms	1.07 ms	1.08 ms	—
Sort lists	7.28 ms	4.71 ms	4.04 ms	—
Apply blur	14.36 ms	5.85 ms	2.95 ms	8.15 ms
Sum	22.99 ms	11.63 ms	8.07 ms	8.15 ms

The following comparison shows that our method supports the same feature-set as gathering approaches (nice far field blur) but also that out-of-focus fragments are accumulated correctly. In the left image the railing is incorrectly averaged with the windows behind it (both are in the far-field), whereas in the right image the railing is clearly in front of the windows.



(previous work) (our results)

Future Work

An interesting direction for future work will be to incorporate multi-layer input data to provide a variant of the point-based DOF by Křivánek [KZB03]. This would enable us to have both correct blending and partial occlusion such as with Selgrad et al. [SRP*15], but not suffer from artifacts due to mip-mapped list structures.

References

[Dem04] Demers J.: Depth of field: A survey of techniques. In *GPU Gems*, Fernando R., (Ed.). Pearson Higher Education, 2004.

[HMY12] Harada T., McKee J., Yang J. C.: Forward+: Bringing deferred lighting to the next level. In *Eurographics 2012 - Short Papers Proceedings, Cagliari, Italy, May 13-18, 2012* (2012), pp. 5–8.

[KZB03] Krivanek J., Zara J., Bouatouch K.: Fast depth of field rendering with surface splatting. In *Computer Graphics International, 2003. Proceedings* (2003), IEEE, pp. 196–201.

[Ngu07] Nguyen H.: *Gpu Gems 3*, first ed. Addison-Wesley Professional, 2007, ch. Practical Post-Process Depth of Field.

[PC81] Potmesil M., Chakravarty I.: A lens and aperture camera model for synthetic image generation. In *Proceedings SIGGRAPH 1981* (1981), ACM, pp. 297–305.

[RTI03] Riguer G., Tatarchuk N., Isidoro J. R.: Real-time depth of field simulation. In *ShaderX2: Shader Programming Tips and Tricks with DirectX 9.0*, Engel W., (Ed.). Wordware, Plano, Texas, 2003.

[SRP*15] Selgrad K., Reintges C., Penk D., Wagner P., Stamminger M.: Real-time Depth of Field Using Multi-layer Filtering. In *Proceedings of the 19th Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2015), i3D '15, ACM, pp. 121–127.

[Tho14] Thomas G.: Compute-Base GPU Particle Systems, 2014. GDC'14.

[ZCP07] Zhou T., Chen J. X., Pullen M.: Accurate Depth of Field Simulation in Real Time. *Computer Graphics Forum* (2007).