# Z-Fighting aware Depth Peeling

Andreas A. Vasilakis and Ioannis Fudos
{abasilak, fudos}@cs.uoi.gr
*Dept. of Computer Science, University of Ioannina, Greece*

## 1. Abstract

We introduce a methodology for handling Z-fighting in depth peeling techniques. Our method is compatible with commodity graphics hardware. We quantitatively and qualitatively compare the resulting depth peeling Z-aware variants with other depth peeling techniques that have been presented in the literature with respect to performance, robustness and scope. Finally, we provide visual results for a number of applications such as transparency and translucency and a demonstration video.

## 2. Depth Peeling

- An efficient process of capturing the entire topological and geometric information of a 3D scene peeling off one or more layers per pass.
- Applications: Transparency, Volume rendering and tests, CSG, Trimming, Collision detection
- Classification based on the #peeling layers/pass:

1. One layer: *O(n)*
   *Front to Back (F2B)* [1]: Slow
2. K layers: *O(n/k), extra memory, primitive pre-sorting*
   *K-Buffer (KB)* [2] : RMW hazards
   *Stencil Routed A-Buffer (SRAB)* [3]: MSAA not supported

- None of these methods can correctly peel all fragments due to Z-fighting.

## 3. Z-fighting

- Two or more primitives have the same z- values.
- Manifests itself through:

1. intersecting surfaces that result in intersecting triangles that belong to the same or different objects
2. overlapping surfaces

## 4. Proposed Methods

- Need one extra rendering pass
- Compatible with commodity graphics hardware

### F2B_ZF: Extending F2B

Algorithm

***1. using max blending***

(a) If all fragments at this depth have been peeled extract next **depth layer** else **stay at this layer.**
(b) Extract the **fragment** with the largest ID [4]

***2. using add/max blending***

From the remaining, not peeled z-fighting fragments:
(a) Calculate the **sum** of them
(b) Find which of them has the **largest ID**.

### F2BKB_ZF: Combining F2B with KB

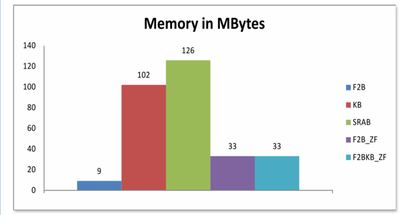
- Approximate method
- Faster for scenes with serious z-fighting artifacts

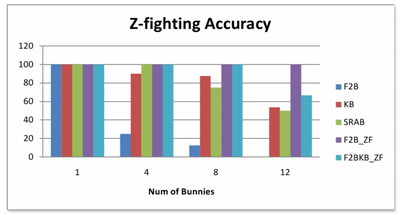
Algorithm

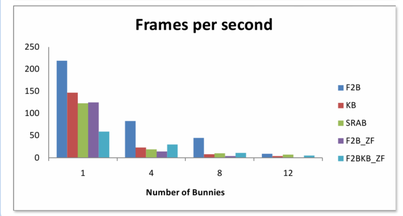
1. Extract next **depth layer** using the F2B.
2. Extract ***k* fragments** located at the current depth layer using a variation of KB.

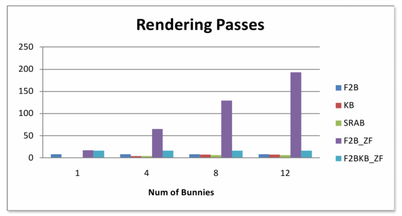
## 5. Results

Following tables show a comparison in terms of peeling accuracy, performance and memory storage of the *F2B*, *KB* and *SRAB* methods and both of our proposed alternatives for a scene consisting of *[1, 4, 8, 12]* Bunnies (69,451 triangles) at a *1024×768* viewport on an nVidia Geforce GTX 480.




## 6. Future Work

The idea can be easily extended to other popular depth peeling techniques such as:

- *Dual depth peeling* [5]
- *Bucket peeling* [7]
- *Multi K-buffer* [6]

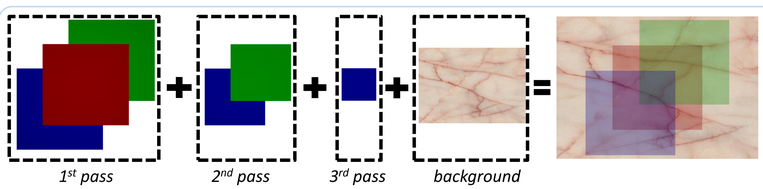
## 7. References

1. Everitt, C., 2001. "*Interactive Order-Independent Transparency*", Tech. Report, NVIDIA Corporation.
2. Bavoil, L., Callahan, S. P., Lefohn, A., Comba, J. A. L. D., and Silva, C. T. 2007. "*Multi-fragment effects on the GPU using the k –buffer*". Proceedings of the 2007 Symposium on Interactive 3D graphics and games - I3D '07.
3. Myers, K., and Bavoil, L. 2007. "*Stencil Routed A-Buffer*", SIGGRAPH '07: ACM SIGGRAPH 2007 sketches.
4. Kessenich, J., 2009. *The Opengl Shading Language version: 1.50, Document revision: 11*.
5. Bavoil, L., and Meyers, K. 2008. "*Order independent transparency with dual depth peeling*". Tech. Report, NVIDIA Corporation.
6. Liu, B., Wei, L.-Y., Xu, Y.-Q., and Wu, E. 2009. "*Multi-layer depth peeling via fragment sort*". 11th IEEE International Conference on CAD/Graphics.
7. Liu F., Huang M.-C., Liu X.-H., and Wu E.-H.. 2009. "*Efficient depth peeling via bucket sort*". In Proceedings of the Conference on High Performance Graphics.

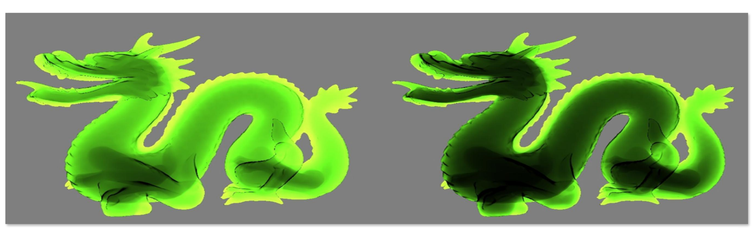
## 8. Software

http://www.cs.uoi.gr/~fudos/siggraph2011.html

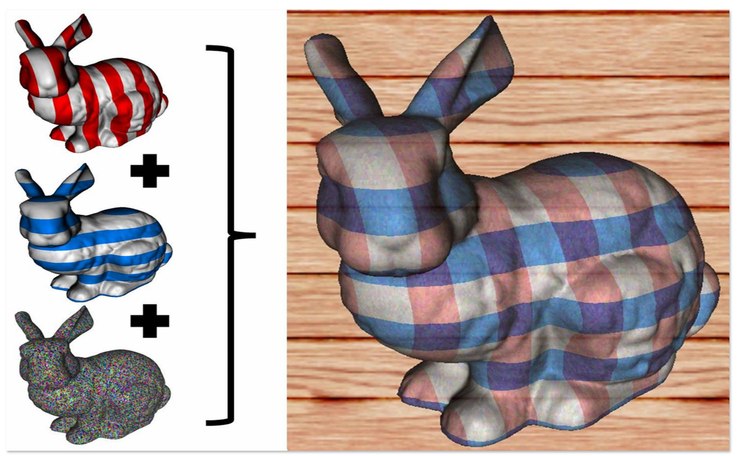
## Figures




1. Multi-pass depth peeling with Z-fighting correction for order independent transparency.



2. The difference of the translucency effect on two instances (placed at the same position) of the Dragon model without (left) and with (right) z-fighting correction.



3. Order independent transparency of three differently rendered Bunnies placed at the same position.

CGRG Computer Graphics Research Group
European Union European Social Fund — Operational Programme Education and Lifelong Learning — NSRF 2007-2013 — Co- financed by Greece and the European Union
Distributed Management of Data Lab
University of Ioannina Computer Science
SIGGRAPH2011 VANCOUVER