Adaptive Transparency

Marco Salvi Jefferson Montgomery Aaron Lefohn

Motivation

*"Order-**dependent** transparency has always been a big limitation for content creators & developers*

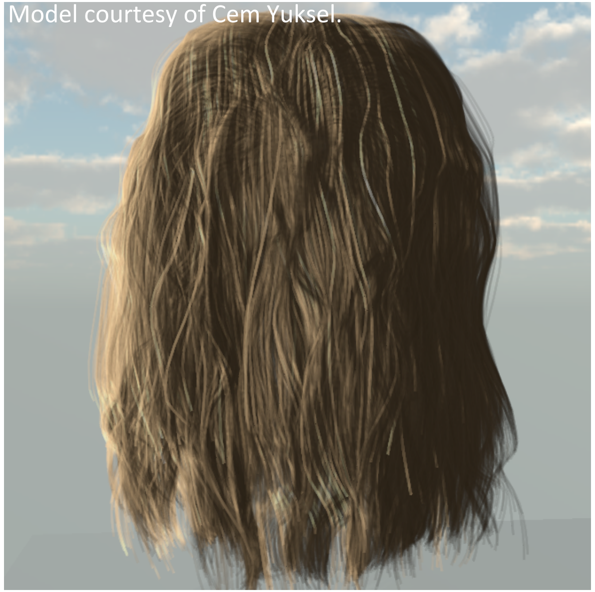
- *Restrictive art pipeline: no glass houses*
- *Even windows on cars & buildings can be painful*
- *Restrictive interaction between objects"*

*"Order-**independent** transparency is must going forward*

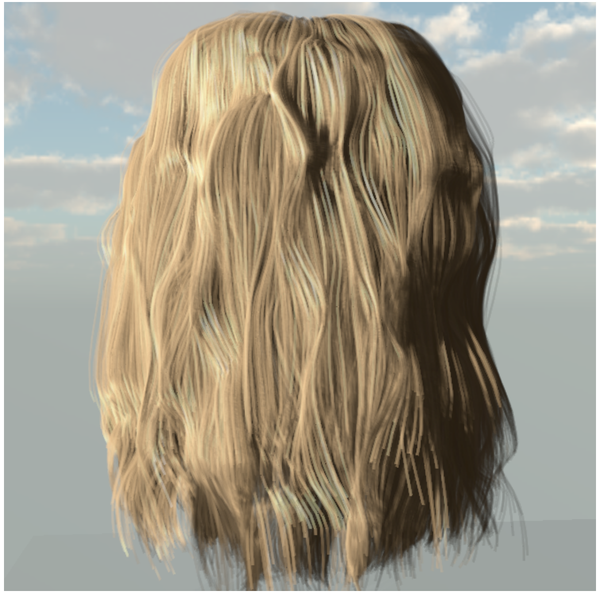
- *Big challenge! Gradual process"*

"Five Major Challenges in Interactive Rendering", SIGGRAPH 2010
Johan Andersson – DICE/EA

Motivation

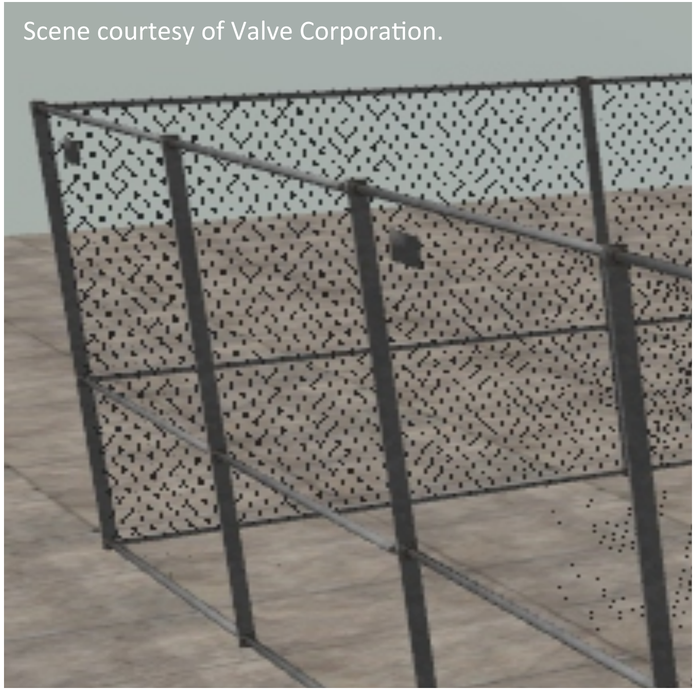


Alpha-Blending



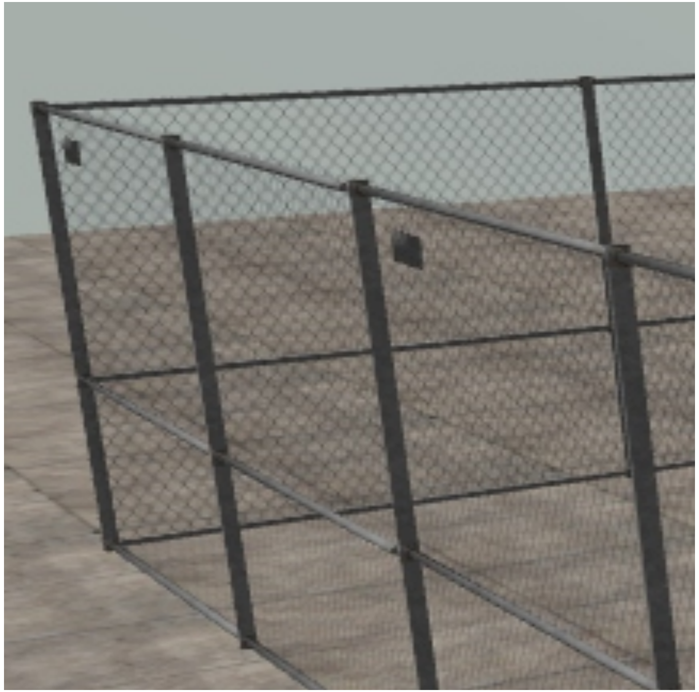
Adaptive Transparency

Motivation



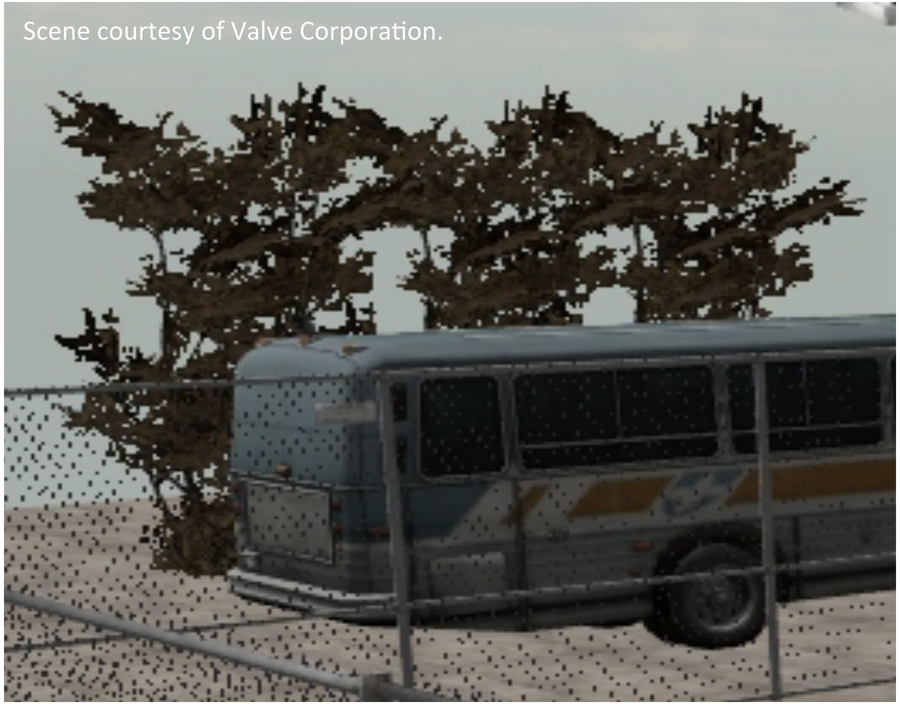
Scene courtesy of Valve Corporation.

Alpha-Test



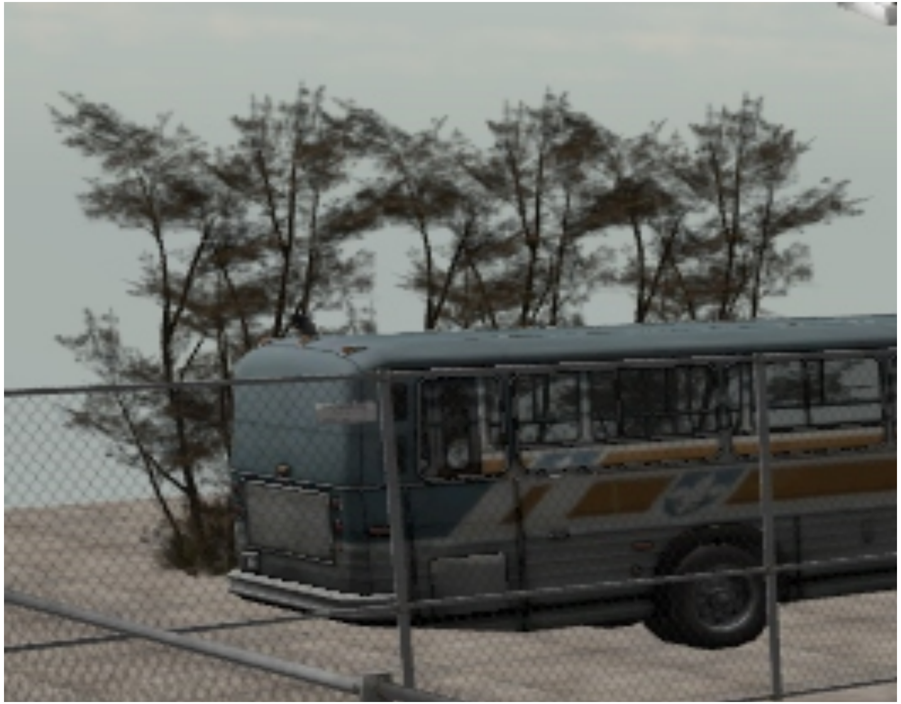
Adaptive Transparency

Motivation



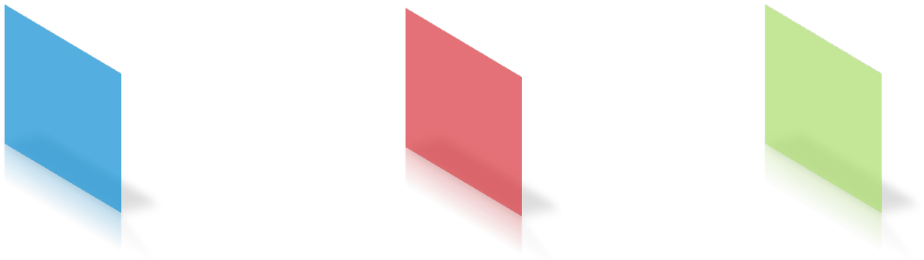
Scene courtesy of Valve Corporation.

Alpha-Test

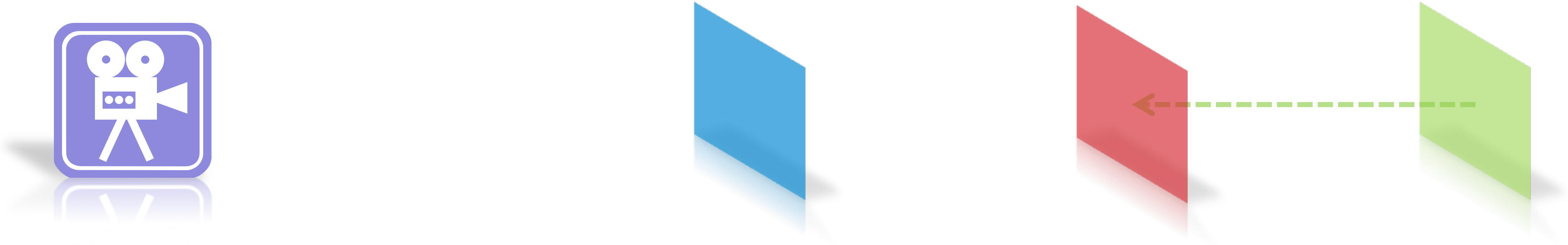


Adaptive Transparency

Alpha-Blending

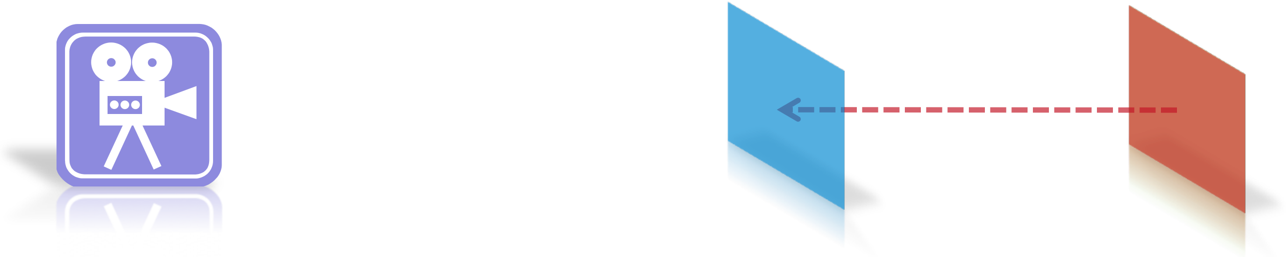


Alpha-Blending



Alpha-Blending

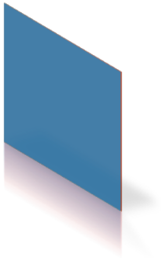
Alpha-Blending



Alpha-Blending

Alpha-Blending

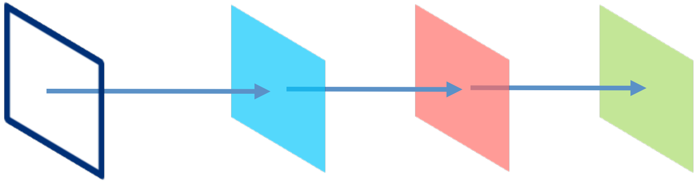
- Fast and stable/predictable performance
- No additional storage required



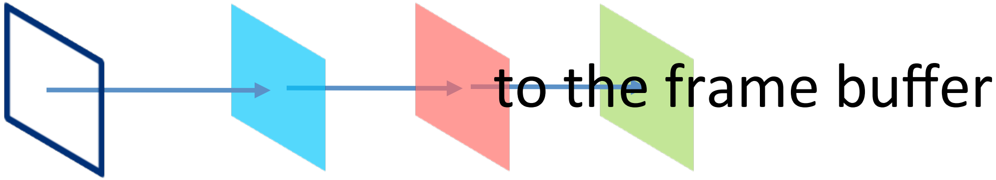
$$C_0 = \alpha_0 c_0$$
$$C_n = \alpha_n c_n + (1 - \alpha_n) C_{n-1}$$

A-buffer*

1) Render fragments color and depth in per-pixel lists



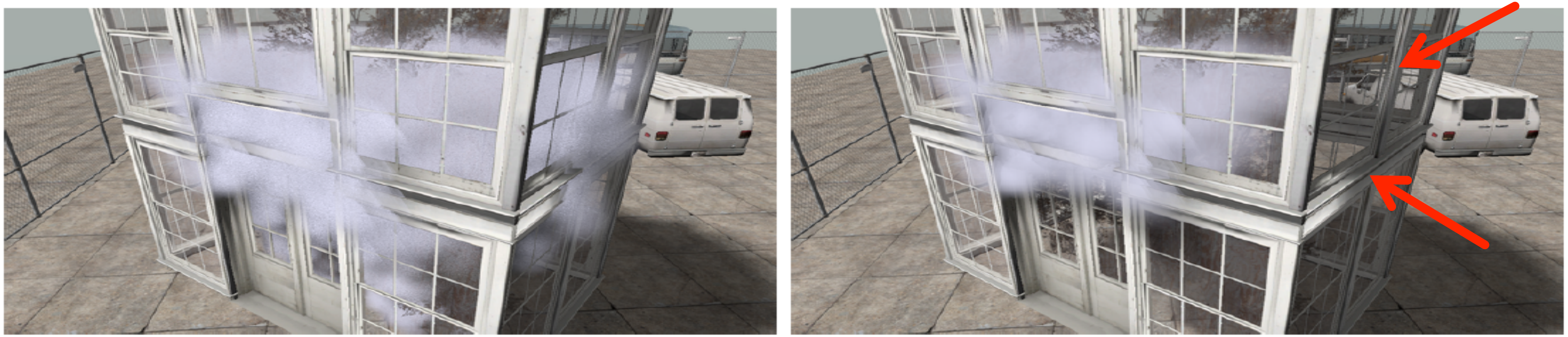
2) Per-pixel sort and composite fragments



*[Carpenter 1984] [Yang et al. 2009]

A-buffer Limitations

- Poor & unstable performance, memory BW limited
- Unbounded memory requirements



Scene courtesy of Valve Corporation.

The Ideal Real-Time OIT Method

- High image quality

- High and stable performance

- Bounded memory usage

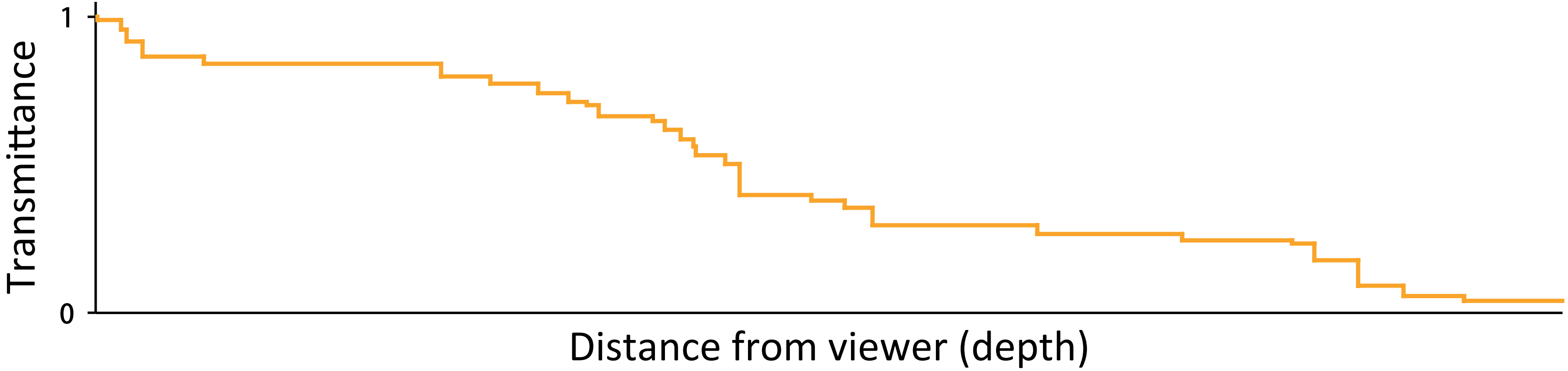
Alternative Compositing Method

- pixel color* = $\sum_{fragments} c_i \alpha_i vis(z_i)$

*[Sintorn et al. 2009]

Visibility Function

- Models light absorption
- Product of step functions (thin blockers)

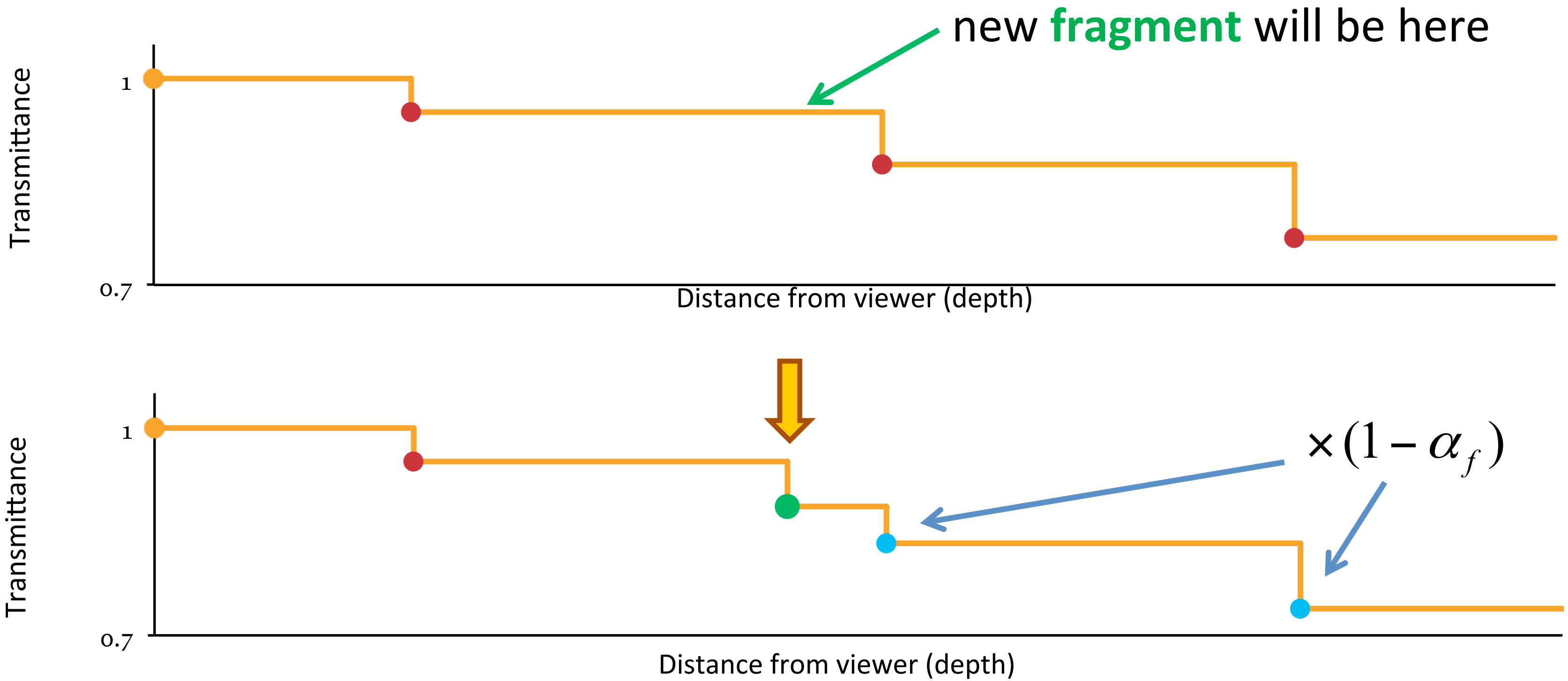


Adaptive Transparency

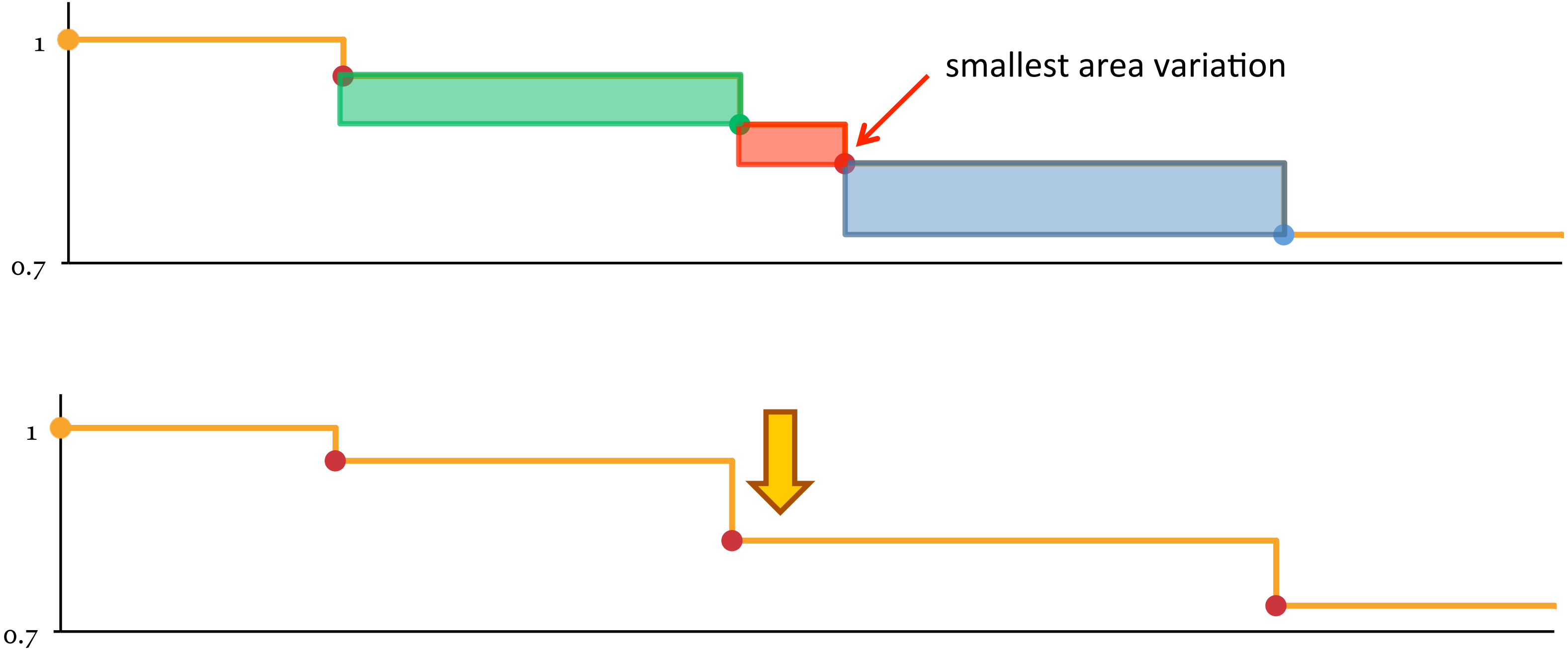
- Fixed size per-pixel visibility representation
  - Store up to N step functions
  - Lossy compression

- Render transparent geometry twice
  1. Build per-pixel visibility function
  2. Evaluate $\sum_{fragments} c_i \alpha_i vis(z_i)$

- To add a fragment ***f*** to we multiply all **nodes** located behind it by $(1-\alpha_f)$



- To compress visibility we remove the node that generates the smallest area variation



GPU Implementation

- Store visibility in the frame buffer?
  - Data update cannot be mapped to DX11 blend modes
  - No RMW operations on the frame buffer
- Store visibility in a Read/Write buffer (UAV)?
  - Cause data races

Proof-of-Concept Implementation

1) Render transparent fragments to per-pixel lists
   - Same as A-buffer implementation

2) For each pixel: build an approximate visibility function and use it to composite all transparent fragments
   - Full-screen pass guarantees atomicity

Results

Scene courtesy of Valve Corporation.

SMOKE scene
21 ms - 10.6 MFragment
Max fragment per pixel: 312
30x faster than A-buffer
2.5x faster than Stoc. Transp.

Model courtesy of Cem Yuksel.

HAIR scene
48 ms - 15.0 MFragment
Max fragment per pixel: 663
40x faster than A-buffer
2x faster than Stoc. Transp.

Scene courtesy of Valve Corporation.

FOREST scene
8 ms - 6.0 MFragment
Max fragment per pixel: 45
7x faster than A-buffer
2x faster than Stoc. Transp.

Results

- Up to 40x faster than A-buffer

- High image quality and scalable performance
  - Easy to trade-off IQ for performance by tuning node count

- Works on any type of transparent geometry
  - Foliage, particles, hair, glass, etc.

Future Work

- Investigate bounded memory implementations
  - Per-pixel locks? New frame-buffer format?

- Better visibility data compression
  - Reduce MSAA impact on memory requirements

Acknowledgements

- Special thanks to Craig Kolb, Matt Pharr, Charles Lingle and Elliot Garbus for supporting this work.

- Jason Mitchell and Wade Schinn at Valve Software for the assets from Left-for-Dead-2

- Cem Yuksel (Cornell University) for the hair model

Q&A

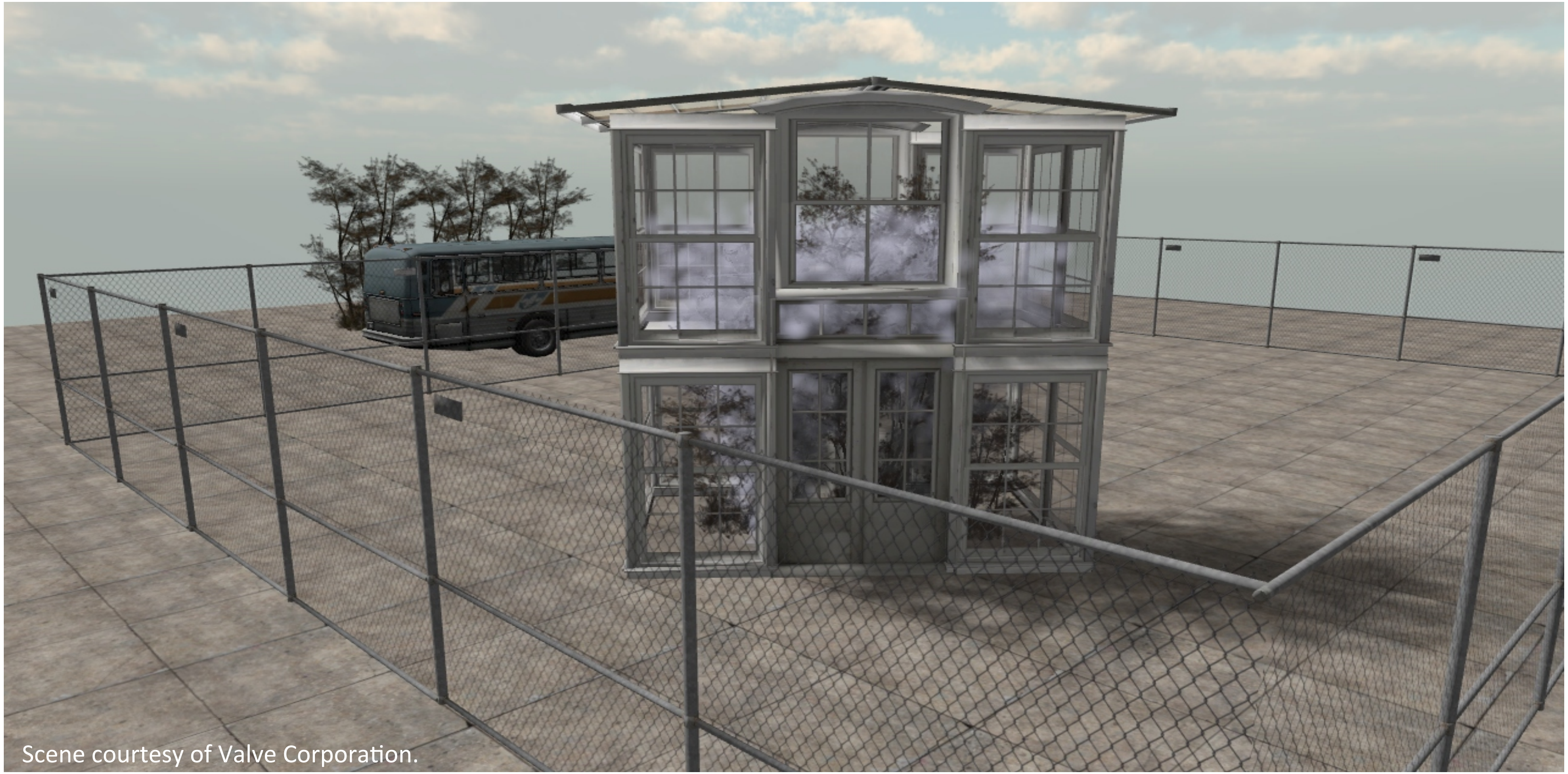
pre-print: http://intel.ly/at_hpg11
source code: http://intel.ly/aoit_gdc

twitter: **@marcosalvi**
e-mail: marco.salvi@intel.com
blog: http://pixelstoomany.wordpress.com

Bibliography

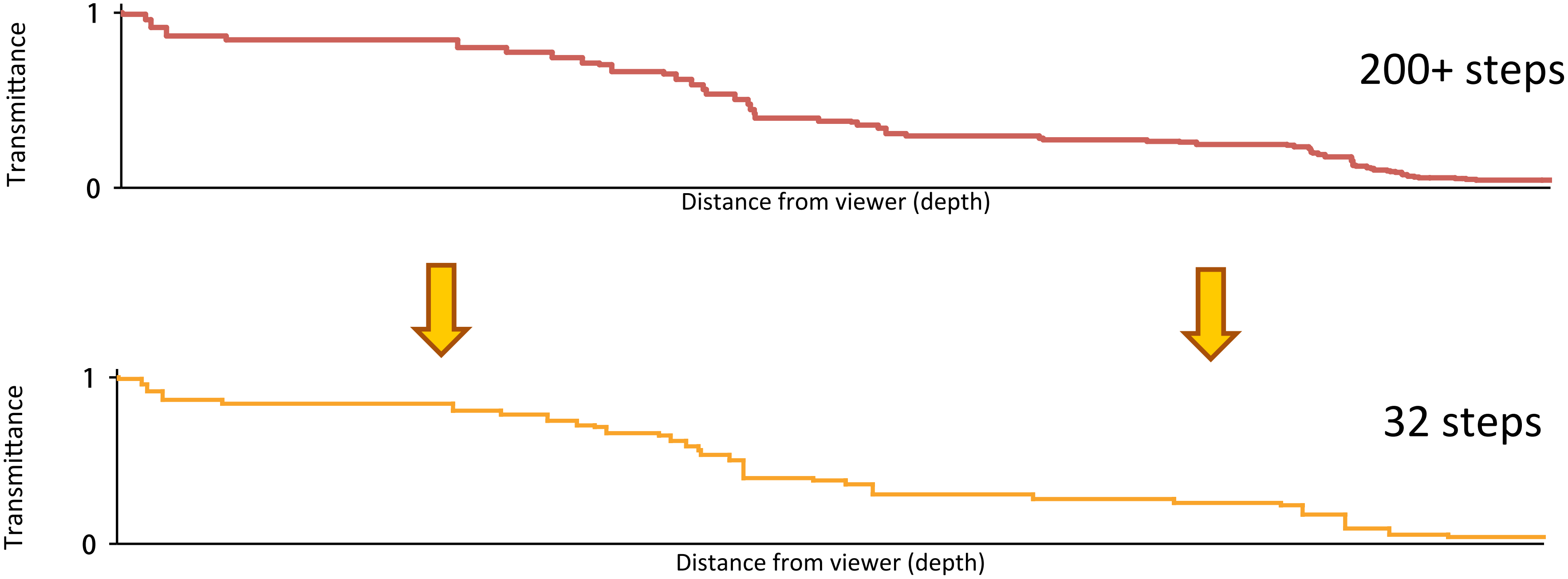
- PORTER, T., AND DUFF, T. 1984. *“Compositing digital images.”* SIGGRAPH Comput. Graph. 18, 3, 253–259.
- CARPENTER, L. 1984. “*The A -buffer, an antialiased hidden surface method*.” SIGGRAPH Comput. Graph. 18, 3, 103–108.
- SINTORN, E., AND ASSARSON, U. 2009. “*Hair self shadowing and transparency depth ordering using occupancy maps.*” In I3D ’09: Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games, 67–74
- PATNEY, A., TZENG, S., AND OWENS, J. D. 2010. *“Fragment parallel composite and filter.”* Computer Graphics Forum (Proceedings of EGSR 2010) 29, 4 (June), 1251–1258.
- YANG, J., HENSLEY, J., GR¨U 547 N, H., AND THIBIEROZ, N. 2010. *“Real-time concurrent linked list construction on the gpu.”* Computer Graphics Forum (Proceedings of EGSR 2010) 29, 4 (June),
- SALVI, M., VIDIMCE, K., LAURITZEN, A., AND LEFOHN, A. 2010. *“Adaptive volumetric shadow maps.”* Computer Graphics Forum (Proceedings of EGSR 2010) 29, 4 (June), 1289–1296.

Backup



Scene courtesy of Valve Corporation.

- Idea: Save bandwidth by working with an approximate visibility function


Transmittance
Distance from viewer (depth)
200+ steps
Transmittance
Distance from viewer (depth)
32 steps