

Constant-Time Light Refraction Simulation

Chase Mayer¹, Ulf Assarsson²

¹mayerchase559@gmail.com

¹Harvard-Westlake School, Los Angeles, California, USA

²Chalmers University of Technology, Gothenburg, Sweden



Ground Truth



Our Method



Error

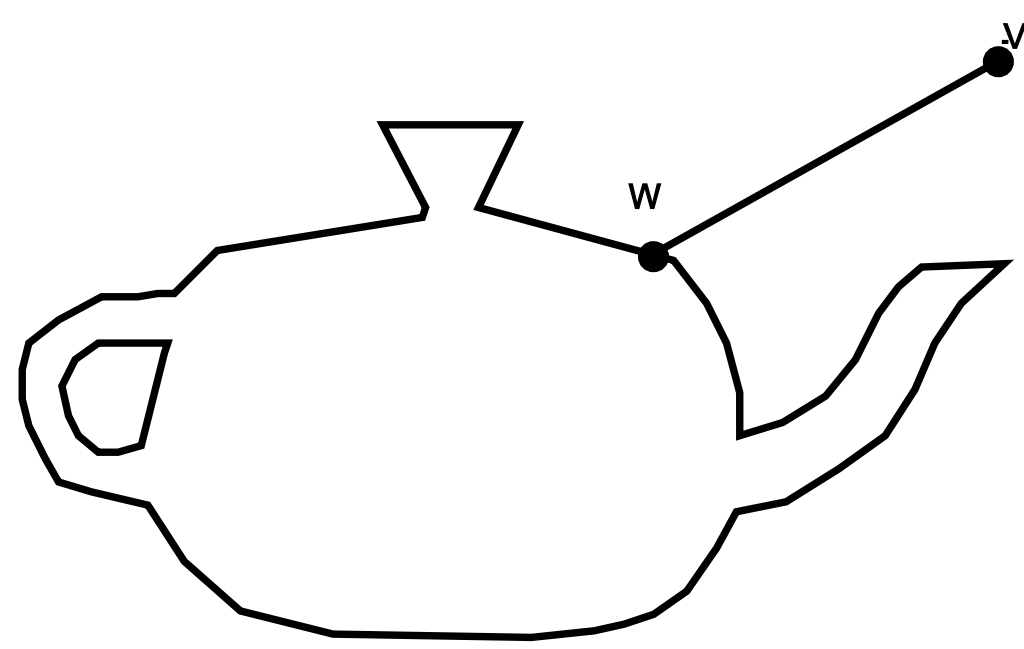
Double-sided refractions: light bends when it enters the buddha and exits the same. Our method handles the two refraction steps with almost the same fidelity as screen-space raymarching, but runs 38% faster.

We propose a fast, constant-time solution for refraction effects by generating a planar approximation for relevant geometry in screen-space. Our method is well-suited for real-time applications. It can also be applied and paired alongside other refraction algorithms, such as two-sided approaches and caustics simulations, as a replacement for raymarching. A ray can be refracted in constant-time, irrespective of scene complexity and screen resolution.

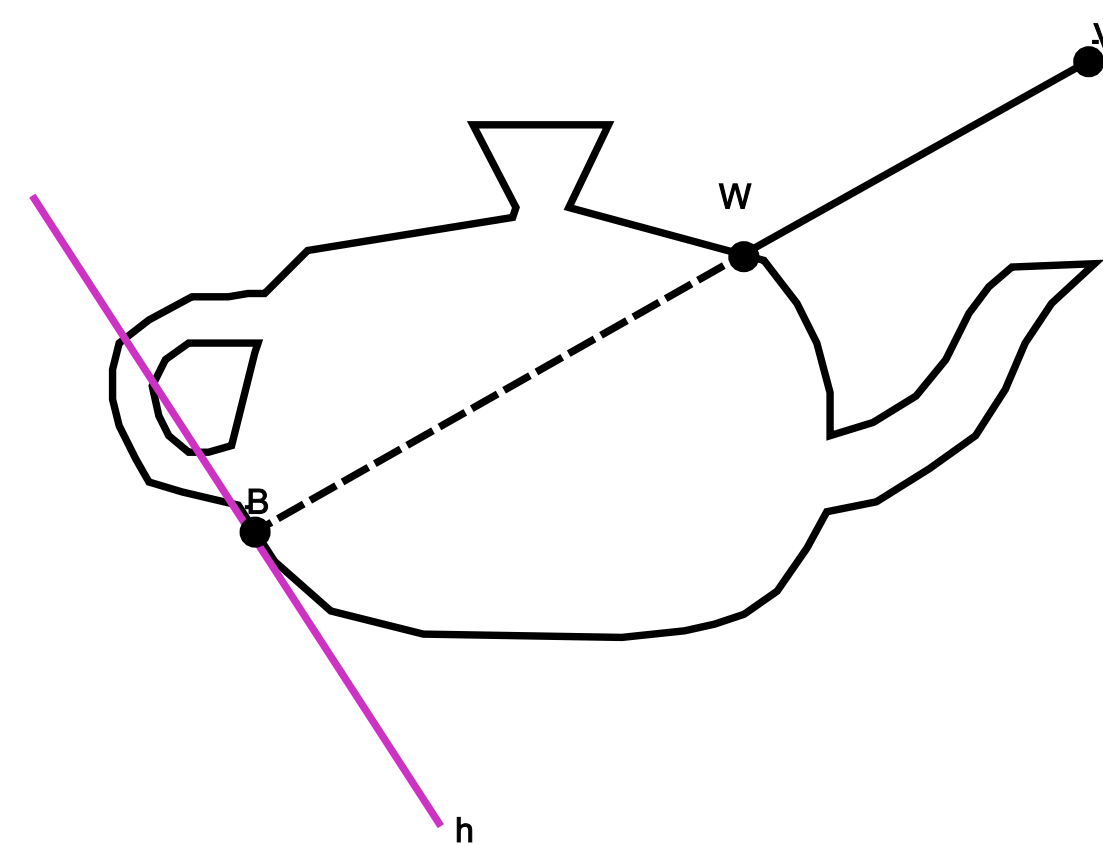


A transparent sphere refracts a painting. Here, our method is equivalent to raytracing.

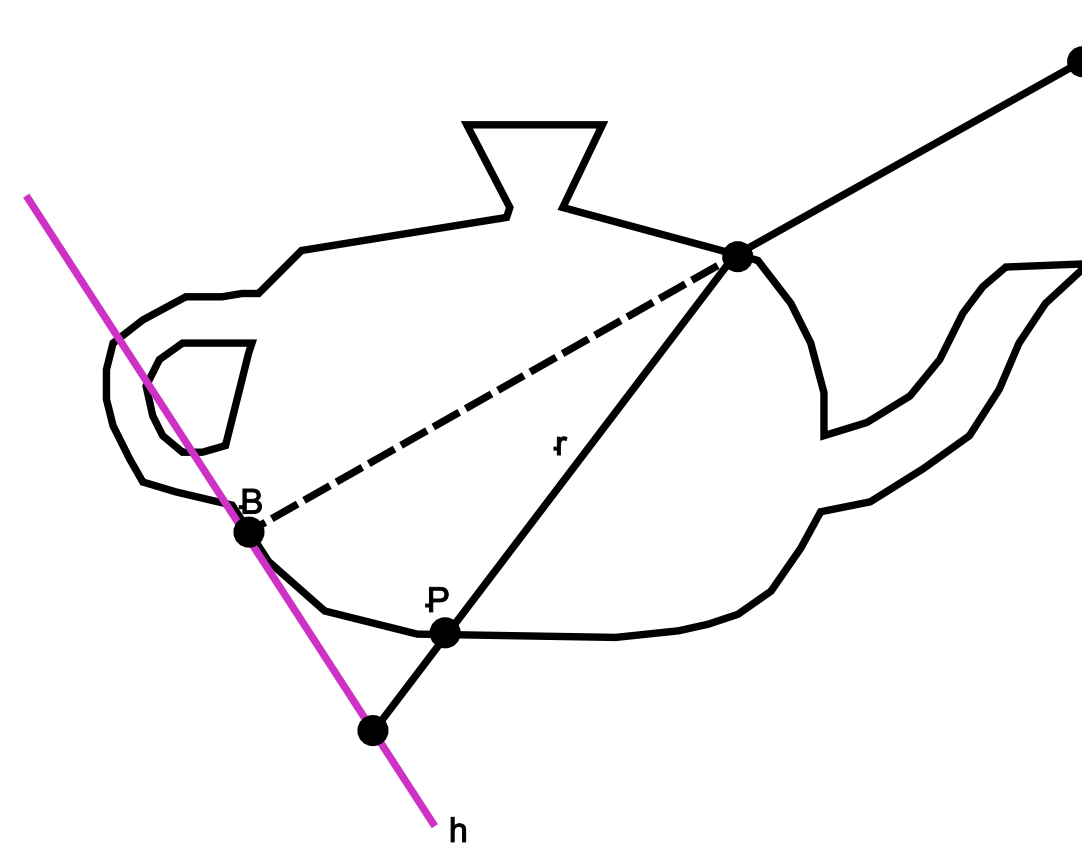
New method can match screen-space raymarched *refractions* while running 38% faster!



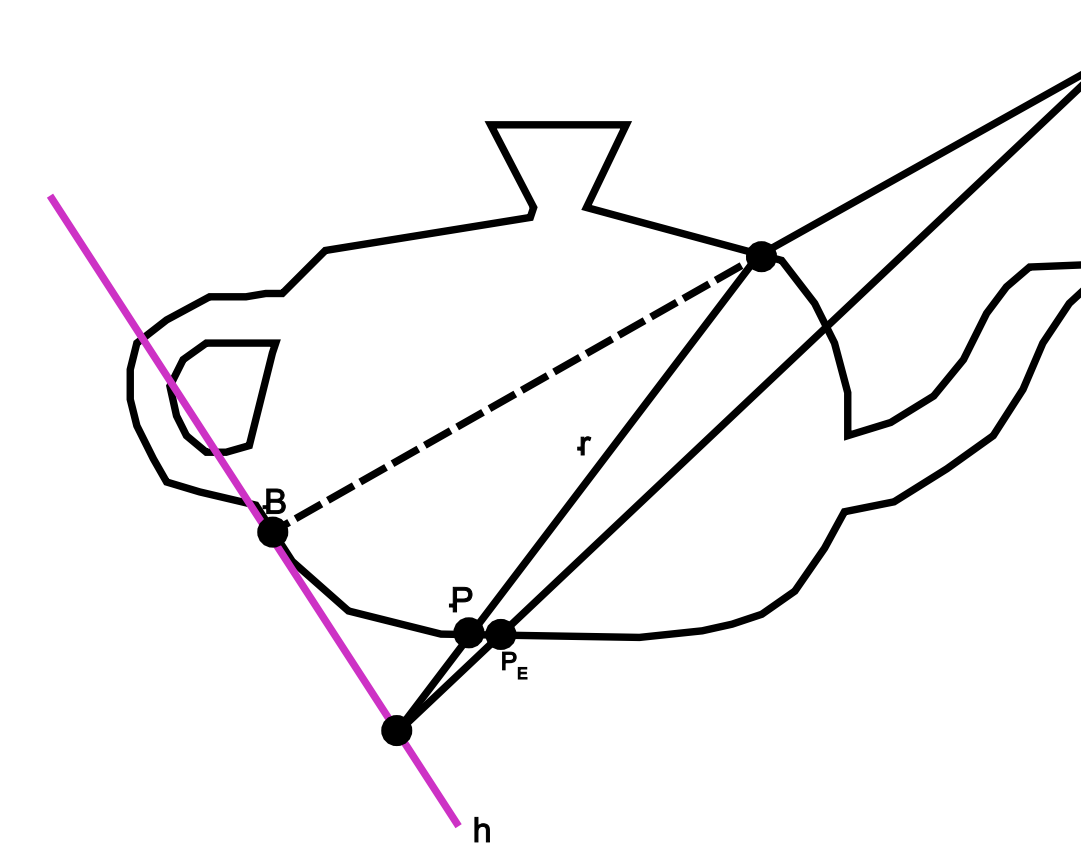
Step 1: Positioned at V , we look at a transparent object at point W .



Step 2: Sampling screen buffers, we find B , the point directly behind W . We generate a planar approximation h for the geometry (see below).



Step 3: Refract the view direction, yielding ray r . Intersect r with plane h in order to approximate the true intersection point P .



Step 4: Cast the intersection point into screen space and sample, yielding point P_e as our approximation for P . Notice how close they are!

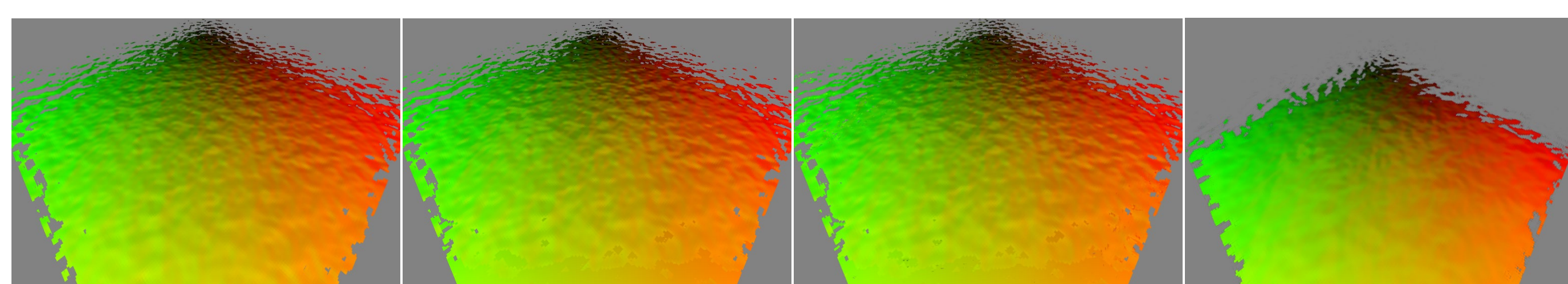
Step 5: iteratively refine the approximation, using P_e as the location to sample screen buffers and generate a new plane. Continue until desired quality is reached -- usually only one or two iterations.

How do we generate the planar approximation? There are two methods.

Method 1: Normal-based
Sample the scene depth and normal behind point W and generate a plane. The normal map may be mipmapped -- a more blurred mip level gives a more complete representation of the scene for drastic refraction effects, while a sharper level is more accurate for subtle effects.

Method 2: Depth-based
We can omit the normal buffer and work with just the scene depth. We need three points to define a plane: we begin with point B directly behind W . Next, we travel along ray r for distance d , where d is the distance between B and W . The third point can be generated using the screen-space derivative of either preceding point. To account for all geometry in between these three estimates, we sample all three points using the mipmap level that encompasses the entire region these three points span.

RESULTS



Ground Truth Ours (normal-based) Ours (depth-based) GPU Gems Method

Unlike single-sample methods like that presented in GPU Gems [4], our methods produce displacements similar to the ground truth. Errors and rendering time shown to the right.

Interested in more details? Scan the QR codes.



In-depth explanation



See it in action!

Performance and Accuracy of Refraction Algorithms (Images Shown to the Left)

Algorithm	Time (ms)	MSE from Raytracing
Single Sample	0.33	0.0057
Our Method (normal-based)	0.37	0.00094
Our Method (depth-based)	0.52	0.00091

Measurements were taken using OpenGL on an RTX 3070 with 14 GB of memory.

REFERENCES

- [1] Charles de Rousiers, Adrien Bousseau, Kartic Subr, Nicolas Holzschuch, and Ravi Ramamoorthi. 2012. Real-Time Rendering of Rough Refraction. *IEEE Transactions on Visualization and Computer Graphics* 18, 10 (2012), 1591–1602. doi:10.1109/TVCG.2011.282
- [2] Honglei Liu, Honglei Han, and Guangzhei Fei. 2020. Two-phase real-time rendering method for realistic water refraction. *Virtual Reality and Intelligent Hardware* 2, 2(2020).
- [3] Manuel M. Oliveira and Maicon Brauwere. 2007. Real-time refraction through deformable objects. In *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games (Seattle, Washington) (I3D '07)*. Association for Computing Machinery, New York, NY, USA, 89–96. doi:10.1145/1230100.1230116
- [4] Tiago Sousa. 2005. Generic Refraction Simulation. In *GPU Gems 2*, Matt Pharr (Ed.). Addison-Wesley, Chapter 19. <https://developer.nvidia.com/gpugems/gpugems2/part-ii-shading-lighting-and-shadows/chapter-19-generic-refraction-simulation>
- [5] Unity Technologies. 2025. Refraction in HDRP. <https://docs.unity3d.com/Packages/com.unity.render-pipelines.high-definition@10.1/manual/Refraction-in-HDRP.html>. Accessed: May 2025.
- [6] Chris Wyman. 2005. Interactive image-space refraction of nearby geometry. In *Proceedings of the 3rd International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia (Dunedin, New Zealand) (GRAPHITE '05)*. Association for Computing Machinery, New York, NY, USA, 205–211. doi:10.1145/1101389.1101431