Refresh Rate Modulation for Perceptually Optimized Computer Graphics

Jeffrey Smith, Thomas Booth and Reynold Bailey
Department of Computer Science, Rochester Institute of Technology, Rochester, New York, U.S.A.

Keywords: Ray-tracing, Eye-tracking, Real-time

Abstract: The application of human visual perception models to remove imperceptible components in a graphics system, has been proven effective in achieving significant computational speedup. Previous implementations of such techniques have focused on spatial level of detail reduction, which typically results in noticeable degradation of image quality. We introduce Refresh Rate Modulation (RRM), a novel perceptual optimization technique that produces better performance enhancement while more effectively preserving image quality and resolving static scene elements in full detail. In order to demonstrate the effectiveness of this technique, we have developed a graphics framework that interfaces with eye tracking hardware to take advantage of user fixation data in real-time. Central to the framework is a high-performance GPGPU ray-tracing engine. RRM reduces the frequency with which pixels outside of the foveal region are updated by the ray-tracer. A persistent pixel buffer is maintained such that peripheral data from previous frames provides context for the foveal image in the current frame. Applying the RRM technique to the ray-tracing engine results in a speedup of 3.2 (260 fps vs. 82 fps at 1080p) for the classic Whitted scene without secondary rays and a speedup of 6.3 (119 fps vs. 19 fps at 1080p) with them. We also observe a speedup of 2.8 (138 fps vs. 49 fps at 1080p) for a high-polygon scene that depicts the Stanford Bunny. A user study indicates that RRM achieves these results with minimal impact to perceived image quality. We also investigate the performance benefits of increasing physics engine error tolerance for bounding volume hierarchy based collision detection when the scene elements involved are in the user's periphery. For a scene with a static high-polygon model and 50 moving spheres, a speedup of 1.8 was observed for physics calculations.

SCIENCE AND TECHNOLOGY PUBLICATIONS

1 INTRODUCTION

Recent advances in consumer level parallel processing hardware have led to the feasibility of generating realistic computer graphics images at interactive rates. However, even with high-end hardware, computationally demanding rendering solutions such as ray-tracing must be heavily optimized to run in real-time.

A number of acceleration techniques have been proven effective in reducing rendering time for ray-tracing applications. These include the use of spatial data structures such as k-d trees (Wald and Havran, 2006) or bounding volume hierarchies (Goldsmith and Salmon, 1987). Despite these advances, computational resources are still dedicated to generating fine detail outside of the viewer's high acuity foveal region. These resources are wasted as the presence of such details does not impact the perceived quality of the scene due to reduced acuity in the peripheral

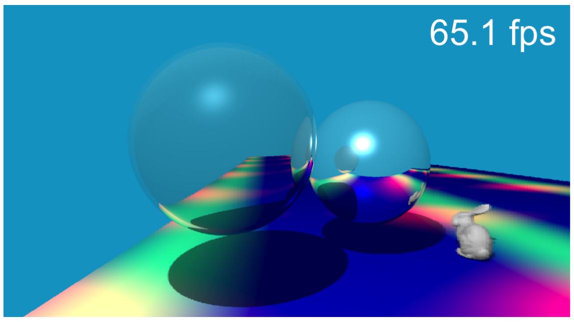


Figure 1: Ray-traced image generated using our perceptual optimization framework. The scene depicts a full ray-tracing solution, including reflected and refracted rays along with a high-polygon model. Our Refresh Rate Modulation technique results in 65.1 fps. For comparison, a full-resolution rendering of this scene has a frame rate of 17.6 fps.

region of the field of view. Raj et al. (Raj and Rosenholtz, 2010) noted however, that peripheral vision is not simply a blurred version of foveal vision, hence the traditional perceptual optimization approach of reducing spatial detail in the peripheral regions still results in a noticeable reduction in image quality.

In this paper, we introduce Refresh Rate Modulation (RRM), a novel perceptual optimization technique that produces better performance enhancement than spatial degradation techniques while more effectively preserving perceived image quality (see Figure 1). Similar to variable resolution approaches, RRM partitions the display area into two subregions that correspond to the foveal and peripheral portions of the user's field of view. However, instead of varying sampling frequency, RRM adjusts the rate at which pixels are updated by the ray-tracer. The foveal region is updated once per frame, and therefore shows the scene in full detail at all times. Pixels in the peripheral region are refreshed once every N frames, where N can be adjusted to strike a balance between performance and perceived output quality.

The result is a subtle fragmentation effect outside of the foveal region that does not decrease the perceived quality of the overall image, but significantly increases performance. If the scene remains still for N or more frames, all peripheral pixels are refreshed and a full-detail image of the entire viewing area is rendered.

Within our framework, physics calculations may also be optimized through the use of real-time perceptual data. While the center of the field of view is able to detect errors in physical phenomena with high accuracy, the periphery is less well-equipped to do so (O'Sullivan et al., 1999). This means that collision error tolerances can be significantly increased in regions outside of the fovea without reducing the perceived quality of motion (so long as penetrative errors are avoided). Many physics engines utilize acceleration structures for polygonal meshes that result in a series of successive calculations for collision detection between two objects. If the collision algorithm is modified to return a collision several layers earlier, computation for collisions with the mesh terminate early and a computational speedup occurs.

The remainder of this paper is organized as follows: background and related work are presented in Section 2. The design of our perceptually optimized rendering framework is described in Section 3. Performance results and a user study to gauge the perceptibility of our novel Refresh Rate Modulation technique are presented in Section 4. In section 5, the paper concludes with a summary of the contributions and potential avenues of future research.

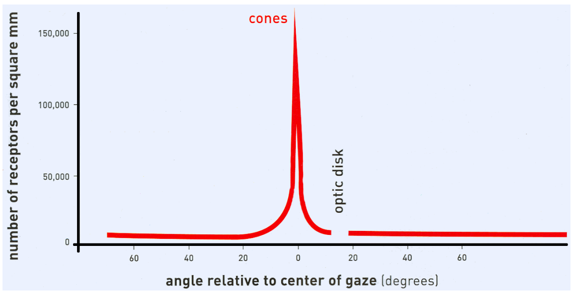


Figure 2: Distribution of cones in the retina. Adapted from (Livingstone, 2002). Cones are densely packed in the center of gaze (fovea) and the density of cones falls off rapidly as angle from the center of gaze increases. The distribution of cones directly affects visual acuity. Visual acuity is highest in the center of gaze and falls off rapidly as angle from the center of gaze increases.

2 BACKGROUND AND RELATED WORK

The human retina contains a large number of interconnected receptor cells that intercept incoming photons and output electrical signals to the visual cortex. Cone cells provide color vision at high illumination levels, and are responsible for detail-oriented visual tasks. The distribution of cone cells across the retina is nonuniform (see Figure 2). This results in two distinct regions within the field of view: the central, high acuity fovea, and the outer, low acuity periphery. The visual system reorients the eye an average of three times per second via saccadic movement and integrates the information gathered at each fixation point to produce a high-detail perceived image of the environment.

Computer graphics models that take advantage of this property of the human visual system for computational speedup or data compression have been proposed and implemented with positive results. Geisler and Perry's (Geisler and Perry, 1999) foveal pyramid approach partitions an existing image into distinct regions based on distance from the current region of interest. The resolution in each of the regions is reduced with a series of low-pass filters, resulting in a multi-resolution image that has full detail in the region of interest and decreases in resolution moving away from this region. This algorithm achieves a 3x reduction in the amount of data required to represent an image. Levoy and Whitaker (Levoy and Whitaker, 1990) implemented a spatially adaptive ray-tracing system that incorporates real-time fixation data from an eye tracker to produce a multi-resolution rendered image. Their 3D mip-map based algorithm results in a nonuniform sampling distribution across the image plane, with considerably higher ray density in the foveal region. After a five minute preprocessing step, they observed a 4.6x speedup when rendering a 256 x 256 x 109 voxel magnetic resonance scan.

Certain features of a scene, such as edges, abrupt changes in color, and sudden movement tend to attract involuntary user attention. Low-level saliency models determine which regions of a scene exhibit these features, and can be used as an alternative to eye tracking when locating regions of interest. Cater et al. (Cater et al., 2003) applied a saliency model that includes knowledge of a viewer's visual task in order to render a scene with high resolution in regions of interest and lower resolution elsewhere. Spatial level of detail variation can also be realized through adaptive subdivision of three dimensional polygon meshes. Reddy (Reddy, 2001) developed a system that renders terrain geometry in high detail at the fixation point and a simplified mesh outside of the foveal region. This is accomplished by recursively subdividing the mesh, with regions outside of the fovea terminating earlier than those within. For a terrain model with 1.1 million triangles, the perceptual optimization achieved a 2.7x improvement in rendering time. While no eye tracking hardware was used for this model (fixation was bound to the center of the image), Reddy emphasizes the need for such technology to produce an accurate perceptually based system. A more general-purpose method for adaptive subdivision was proposed and implemented by Murphy and Duchowski (Murphy and Duchowski, 2001). It converts a full-polygon mesh to a variable level of detail mesh through spatial degradation according to visual angle. An eye tracker is used to determine which portion of the mesh to render in full detail while the remainder is rendered using the degraded mesh. For a 268,686 polygon Igea mesh, applying this technique allowed for near-interactive frame rates (20 - 30 fps), while frame rate for the full resolution model was too low to measure.

While many perceptual optimization techniques have shown positive results, existing methods are not well-suited for application in a subtle, perceptually optimized real-time computer graphics architecture. Multi-resolution display models tend to produce noticeable image degradation; according to Levoy and Whitaker (Levoy and Whitaker, 1990), "users are generally aware of the variable-resolution structure of the image". In addition, the nonuniform pixel distribution produced by the multi-resolution approach tends to exhibit poor coherency with regards to the Single Instruction, Multiple Data (SIMD) paradigm employed by modern GPUs. Considering the current trend towards massively parallel computing architectures, this is a major drawback. Adaptive subdivision comes with a similar drawback; transitioning between the full-detail mesh and the spatially degraded mesh produces motion that is very perceptible to the user's peripheral vision. Task-level saliency models offer excellent computational speedup and low noticeability. However, they are not applicable to the general case, where the user task may be complex and regions of interest are not guaranteed to be consistent or easily identifiable. Furthermore, automatic prediction of attention regions has been shown to be unreliable (Marmitt, 2002). Our perceptually optimized rendering framework leverages the difference in acuity between the foveal and peripheral regions of the field of view to provide computational speedup while more effectively preserving perceived image quality compared to spatial degradation techniques.

3 SYSTEM DESIGN

3.1 Ray-Tracing Framework

Ray-tracing is a well-established method for rendering three-dimensional scenes (Whitted, 1980). The algorithm models the approximate path of light in reverse, flowing from the camera to objects in the scene. When a light ray intersects an object, the associated pixel is filled with the color of the object at that point. For reflective and refractive objects, additional rays are spawned recursively at the point of intersection.

The ray-tracing algorithm is computationally intensive, which has historically prevented it from being used for real-time applications. Approximately 75% of the time required to render simple scenes is allocated to computing ray-object intersections, with this number increasing for scenes with a large number of objects. A performance speedup can be realized by reducing the number of intersection tests per ray or the overall number of rays computed. Our system is built on a basic ray-tracing framework, and is designed to reduce the number of rays that need to be computed by taking advantage of the differences in visual acuity between the foveal and peripheral vision. It also includes a number of traditional optimizations that reduce the number of intersections per ray as well as the time required to compute each intersection.

The bounding volume hierarchy (BVH) is one effective method of organizing scene object data to reduce ray-object intersection calculations per ray. Each polygon in a mesh is encapsulated within a bounding volume; we use a sphere, which has a relatively low intersection cost. This set of volumes is

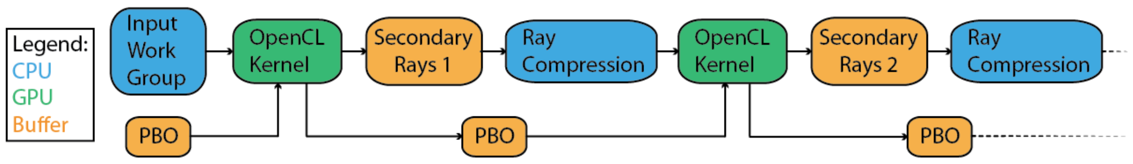


Figure 3: Secondary ray processing. Instead of using recursion or a secondary ray stack, multiple passes are made with the same OpenCL kernel.

paired up and encapsulated within larger volumes until only one volume remains. This volume now contains a hierarchy that represents all geometry in the mesh. Ray intersections on the entire mesh are performed using the hierarchy. If a ray intersects the top level bounding sphere, its children are recursively checked for intersection. The BVH scheme eliminates all but one sphere intersection test for the majority of rays that do not actually intersect the mesh. In order to send the bounding volume hierarchy to an OpenCL kernel, it must be flattened into a one-dimensional array. The flattening algorithm described by Thrane (Thrane et al., 2005) serves as a basis for our method. The bounding volume hierarchy is traversed depth-first, and a traversal index is assigned to each node to indicate traversal order. Each node is also given an escape index, which indicates where to go next if its child nodes are to be ignored. Leaf nodes require a third index that corresponds to the mesh triangle that they encapsulate. Once these values are assigned, all bounding volumes can be placed in a linear array in the order of their traversal indices. The flattened BVH and triangle data are written to a file and loaded directly on subsequent executions to accelerate program startup.

General purpose GPU acceleration has emerged as a compelling means of reducing intersection computation time for high performance ray-tracing systems. Our framework places all ray-tracing logic, including ray calculation, intersection tests, texturing and shading, in a single OpenCL kernel for execution on the GPU. An array of work unit structures holds input and output data for the kernel. Each work unit corresponds to a group of 4 onscreen pixels (a pixel block), and contains screen coordinate position and ray data along with other information. The kernel accepts input work units with either precomputed ray data or raw screen coordinates.

Secondary ray processing is accomplished with several consecutive kernel calls (see Figure 3). This allows OpenCL to reconfigure work distribution for each reflection or refraction step in order to maintain GPU saturation. This contrasts with the more traditional stack-based approach, where processing elements that do not generate secondary rays sit idle until those with secondary rays finish computation. Secondary ray output for most scenes is very sparse, so the CPU-side host program compresses the output array before initiating the next kernel pass. Secondary work units represent only 1 onscreen pixel, since pixel blocks may overlap reflective/refractive and non-reflective/refractive surfaces.

3.2 Perceptual Optimization

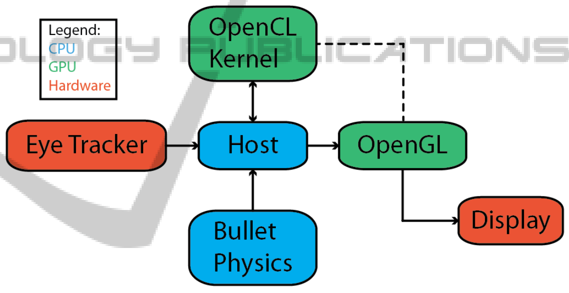


Figure 4: Runtime data flow. Our framework is composed of several subsystems that work in tandem to produce perceptually optimized computer graphics.

Figure 4 shows an overview of our perceptually optimized framework. The host serves as a central hub and handles initialization and interprocess communication. It also maintains object and camera positional data, and manages the Refresh Rate Modulation cycle. The OpenCL kernel contains the full rendering algorithm, which writes to a persistent pixel buffer that is shared between OpenCL and OpenGL in GPU memory. OpenGL is responsible for writing this buffer to the display each frame and initiating the rendering process for the next frame. Eye tracking hardware provides real-time fixation data in the form of X and Y screen coordinates, which is passed from the host to the OpenGL kernel to update foveal position. If physics functionality is enabled, the open-source Bullet Physics engine drives positional data for all scene objects.

The eye-tracker used in this project is a SensoMotoric Instruments iView X Remote Eye-Tracking Device operating at 250 Hz with gaze position accuracy $< 0.5°$. While the data it provides is quite accurate, like all eye-trackers it does exhibit some degree of noise. Using raw fixation data detracts from the user experience, because peripheral vision is extremely sensitive to motion (McKee and Nakayama, 1984). To rectify this issue, an auxiliary smoothing filter has been placed between the eye tracker and the host. Figure 5 shows a photograph of our setup.

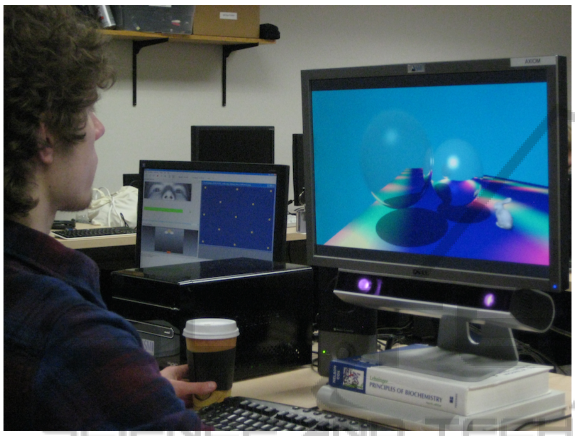


Figure 5: Photograph of our setup. User fixation is monitored and used as input to our perceptual framework. The eye-tracking hardware is fixed to the bottom of the screen.

3.2.1 Refresh Rate Modulation

The primary contribution of this work is the Refresh Rate Modulation technique for perceptually adaptive level of detail adjustment. The display is split into two segments: the foveal region and the peripheral region. The foveal region is a small inset pixel group that corresponds to the high acuity region of the human visual system, and is updated once every frame. The work group that represents the foveal region is dense, and features one work unit for each pixel group in the region. The peripheral region takes up the remainder of the image, and lies within the lower-detail portion of the user's field of view. This region is updated only once every N frames, which results in a substantial computational speedup. The work group that represents this region is sparse, and features one work unit for every N pixel blocks in the region. Each work unit in both regions computes for four consecutive pixels to maximize SIMD coherency and improve performance. Figure 6 shows how the display area is separated into foveal and peripheral regions for a user fixating in the center of the screen. Note that a circular foveal region, while more perceptually accurate, would eliminate much of this coherency and significantly reduce speedup.

Figure 7 illustrates the Refresh Rate Modulation technique. A single work unit is processed at each

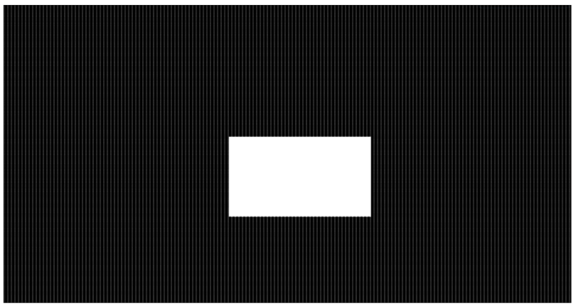


Figure 6: The display area is segmented into a dense inner region (fovea) and a sparse outer region (periphery). White pixels represent work units computed during a single frame.

frame, while the rest of the work units within the cycle group maintain data from previous frames. When the red marker is reached, a full cycle is complete and rendering for the next frame begins again at the green marker. Work units in the foveal region undergo a full cycle each frame, so they are updated in real-time. Units in the peripheral region undergo a full cycle only once every N frames.

Applying the Refresh Rate Modulation technique leads to an effect in which the portion of the display that is viewed by the fovea is rendered in crisp detail, while the rest of the display is subtly fragmented. This fragmentation only occurs when the camera or scene elements are in motion; if scene movement ceases, the display naturally resolves a full-detail image after N frames with no additional overhead. This results in a full-resolution rendering after less than half a second for applications with a real-time frame rate.

The RRM technique avoids the post-processing step that is required by traditional spatial degradation techniques to reduce visible pixelation, and thereby avoids a great deal of processing overhead. Refresh order within the cycle groups can be adjusted on the fly to adjust fragmentation style for scene content and movement (e.g. horizontal, serpentine, scattered), and work group size may be decreased to reduce fragmentation in high-motion scenes or increased to maximize performance. Movement of the foveal region is accomplished within the GPU kernel by simply adding the current fixation position to each work unit position. This allows the entire input work group GPU memory buffer to remain unchanged throughout execution, and avoids costly CPU to GPU memory operations.

A persistent OpenGL pixel buffer object is maintained and shared between OpenGL and OpenCL for efficient memory management. Pixel buffer objects allow pixel data to be stored in high-performance graphics memory on the GPU, and enable fast data transfer to and from the graphics card through direct memory access (DMA) without CPU involvement.

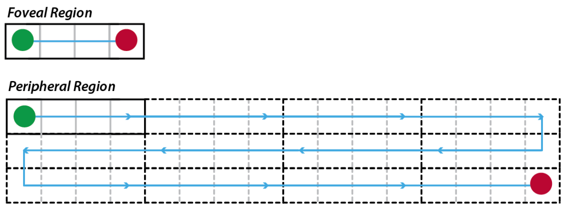


Figure 7: Refresh rate modulation. Work units within the foveal region are updated every frame for real-time rendering, while those in the peripheral region are updated only once every N frames for significant computational speedup.

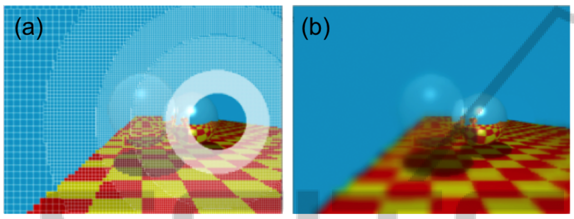


Figure 8: Spatial degradation results in less optimal performance and requires a costly blurring effect that is still noticeable to peripheral vision. (a) Whitted scene with grid to reveal multi-resolution layout (b) Whitted scene with blur.

Since the buffer is not flashed at the beginning of each frame, peripheral data from previous frames can be leveraged to provide meaningful context for the real-time contents of the foveal region.

A more conventional multi-resolution ray-tracing framework was developed near the beginning of this project, and motivated development of the RRM technique. Analyzing performance differences between execution on the CPU and GPU yielded valuable insight regarding GPU characteristics as well as bottlenecks in the algorithm itself. Figure 8 shows a sample image from this original framework with a grid feature enabled to highlight the various resolution levels. While results for this implementation were quite promising on the CPU and on an older commodity GPU, performance was less than impressive on a newer high-performance GPU. There were two primary reasons for this: complex CPU-side work group management between frames, and irregular work groups that are not well-suited to SIMD. In addition, this technique leads to pixilation even when there is no movement in the scene, which in turn requires a post-processing blur effect to remove visible seams. While post-processing effects can be achieved easily with the OpenGL Shading Language (GLSL), frequent switching between OpenCL and GLSL contexts on the GPU is costly and results in massive performance drop. These observations led to a focus on minimizing work group management, simplifying the work group layout and achieving acceptable perceived image quality without post-processing.

3.3 Perceptually Optimized Collision Detection

In order to demonstrate the flexibility of our framework, we have integrated a perceptual collision detection subsystem that renders realistically moving geometry using our ray-tracing engine and the open-source Bullet Physics Library. O'Sullivan (O'Sullivan et al., 1999) showed that interruptible collision detection can significantly reduce the time required for physics calculations while maintaining plausible scene motion. Our system takes advantage of the bounding volume hierarchy based collision detection system that Bullet Physics provides for static polygonal meshes. A performance improvement is gained by using only the top level of the BVH for collisions, which is subtle enough to be imperceptible to the periphery.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

4 RESULTS

The quality of a perceptual optimization technique must be assessed in two ways: computational speedup, and perceptual subtlety. The RRM approach requires a specialized metric for computational performance, since the onscreen position of the dense foveal region can have an effect on frame rate for scenes with nonuniform complexity. To account for this, we render one frame for each possible foveal position and average the results to produce a representative overall frame rate. The perceptual subtlety of the effect was measured during a small user study. Test subjects were instructed to look at the full resolution scene. RRM was then enabled, and subjects rated how noticeable the effect was on a scale from 1 to 10 (1 is not noticeable, 10 is very noticeable).

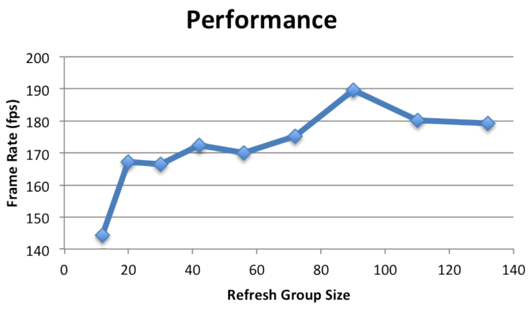


Figure 9: Performance of RRM for different refresh group sizes. Increasing group size reduces the number of pixels that must be updated each frame and improves performance.

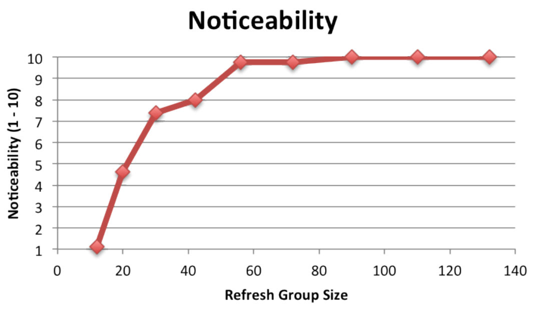


Figure 10: Results of user study. Increasing group size causes RRM to degrade perceived image quality.

Figure 11 shows the computational speedup that RRM achieves for the classic Whitted scene with and without secondary ray effects, as well as a high polygon scene that features a 5000 polygon model of the Stanford Bunny. The Whitted scene with secondary rays offers the best speedup since computation for a large portion of both primary and secondary rays is avoided, while the other scenes remove only primary rays. Our framework handles high polygon scenes very well with RRM enabled, achieving a frame rate that is considerably higher than real-time. All tests were performed on a system with a 3.6GHz Intel Core i7 processor and a Radeon HD 7970 GPU.

The perceptual study also produced favorable results, with an average noticeability rating of 1.13 for a group with 5 participants (4 males, 1 female, ages 18-27). This indicates that RRM achieves excellent computational speedup with almost no impact to the perceived quality of rendered images. The high-polygon scene with an irregular floor texture was used for the study to maximize geometric and color variety. After subjects rated the scene for the default 3x4 refresh group dimensions, group size (or N, see Figure 7) was increased to gauge the impact of different peripheral refresh rates on the noticeability and performance of RRM. As illustrated by Figure 9, increasing N from the default of 12 to a maximum of 132 increases frame rate by up to 50 fps. This performance increase has an associated perceptual cost; the participants indicated that the effect is noticeable (greater than 5 on the noticeability scale) to the periphery for values of N larger than 20 (see Figure 10). Figure 12 shows the high polygon scene with N = 132 to illustrate the perceptual impact of large values of N.

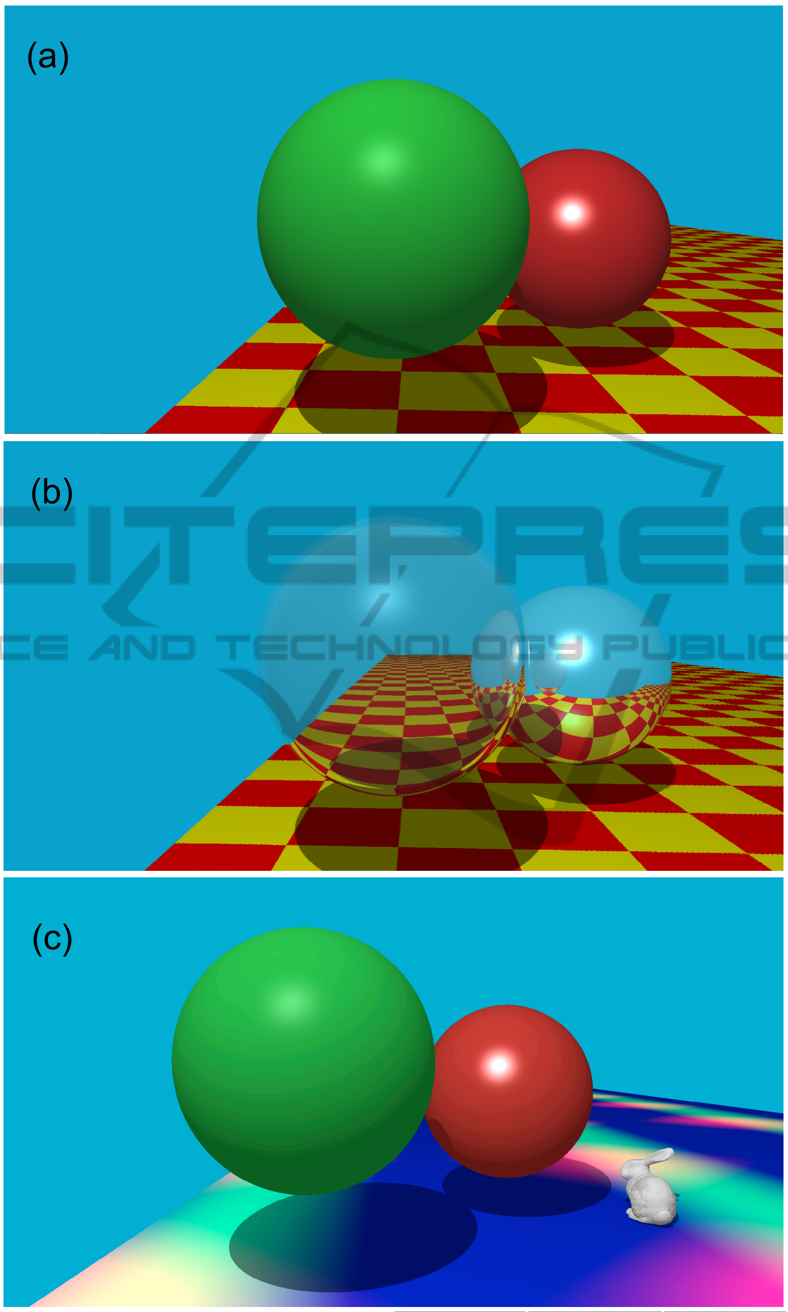
The perceptually optimized collision detection system decreases the physics calculation time for the scene shown in Figure 13 from 2.01ms to 1.15ms for a speedup of 1.8. Physics calculation times were measured over a span of 1000 frames and averaged to ensure that they are indeed representative of the optimization.

5 CONCLUSION

Through the implementation and testing of our perceptually optimized computer graphics framework, we have demonstrated the viability of the novel Refresh Rate Modulation (RRM) optimization technique. Our RRM technique partitions the viewing area into two subregions based on a model of human visual perception, and reduces the refresh rate in the outer region for up to a 6x speedup in our tests. A user study indicates that RRM achieves this computational speedup with almost no effect on perceived image quality. We have also shown that the framework is extensible to other perceptually optimization techniques by incorporating an interruptible collision detection algorithm that drives positional data for the ray-tracing engine.

Some work remains to more fully realize the potential of our current system. The collision detection system should be more fully integrated with the ray-tracing engine such that the optimization is engaged via fixation instead of being manually toggled as it is now. The brute force bounding volume hierarchy construction algorithm should also be made more efficient to allow for the representation of higher polygon models in a reasonable amount of time. Our performance results suggest that a model with significantly more than 5000 polygons could be rendered in real-time with RRM enabled. In addition, a reduction routine on the GPU could be used to remove empty secondary rays in place of the CPU method that is currently in place. While reduction is not well-suited to stream processors, it would avoid a number of expensive GPU-to-CPU memory operations. Finally, refresh order and group size could be adjusted in real-time to perceptually compensate for dynamic scene content. This might be accomplished through the use of visual energy functions, as described by Avidan and Shamir (Avidan and Shamir, 2007).

Our framework can be extended to include a variety of perceptual optimizations. Adaptive subdivision, the simplification of polygonal meshes outside of the foveal region, is a prime candidate for our framework as it has been proven effective in providing computational speedup. Supersampling within the foveal region to improve perceived image quality while only modestly increasing computational load is another possibility. We envision that our framework will be used as a testbed for future perceptual graphics techniques.



SCITEPRESS
SCIENCE AND TECHNOLOGY PUBLICATIONS

(d)

	Full Resolution	Rate Modulation	Speedup
Whitted Scene (no secondary rays)	82 fps	260 fps	3.2
Whitted Scene (with secondary rays)	19 fps	119 fps	6.3
High Polygon Scene	49 fps	138 fps	2.8

Figure 11: Performance results for selected scenes rendered at 1080p resolution. (a) Whitted scene without secondary rays. (b) Whitted scene with secondary rays. (c) Scene containing a 5000 polygon model of the Stanford Bunny. (d) Performance results for each scene at full resolution and with RRM enabled. A refresh group size of 12 is used for RRM.

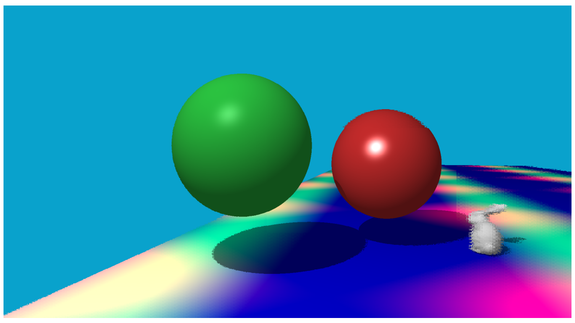


Figure 12: Perceptual impact of increasing the refresh group size to N = 132. The fragmentation effect is very noticeable even to peripheral vision.

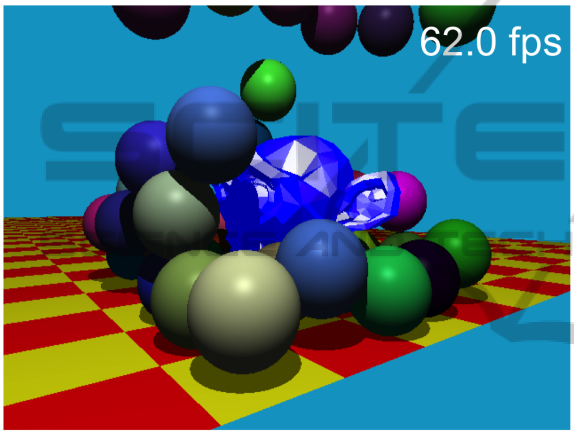


Figure 13: High polygon mesh with 50 moving spheres. Applying perceptual optimization to the physics model results in a physics calculation speedup of 1.8.

ACKNOWLEDGEMENTS

This material is based on work supported be the National Science Foundation under Award No. IIS-0952631. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

Avidan, S. and Shamir, A. (2007). Seam carving for content-aware image resizing. In *ACM SIGGRAPH 2007 papers*, SIGGRAPH '07, New York, NY, USA. ACM.

Cater, K., Chalmers, A., and Ward, G. (2003). Detail to attention: exploiting visual tasks for selective rendering. In *Proceedings of the 14th Eurographics workshop on Rendering*, EGRW '03, pages 270–280, Aire-la-Ville, Switzerland. Eurographics Association.

Geisler, W. S. and Perry, J. S. (1999). Variable-resolution displays for visual communication and simulation. *SID Symposium Digest of Technical Papers*, 30(1):420–423.

Goldsmith, J. and Salmon, J. (1987). Automatic creation of object hierarchies for ray tracing. *IEEE Comput. Graph. Appl.*, 7(5):14–20.

Levoy, M. and Whitaker, R. (1990). Gaze-directed volume rendering. *SIGGRAPH Comput. Graph.*, 24(2):217–223.

Livingstone, M. (2002). *Vision and Art: The Biology of Seeing*. Harry N. Abrams, Inc.

Marmitt, G. (2002). *Modeling Visual Attention in VR: Measuring the Accuracy of Predicted Scanpaths*. Clemson University.

McKee, S. and Nakayama, K. (1984). The detection of motion in the peripheral visual field. *Vision Research*, 24(1):25–32.

Murphy, H. and Duchowski, A. T. (2001). Gaze-contingent level of detail rendering. In *Proceedings of Eurographics 2001*, Manchester, UK.

O'Sullivan, C., Radach, R., and Collins, S. (1999). A model of collision perception for real-time animation. In *Proc. 1999 Conference on Computer Animation and Simulation - Eurographics Workshop (EGCAS*, pages 67–76. Springer.

Raj, A. and Rosenholtz, R. (2010). What your design looks like to peripheral vision. In *Proceedings of the 7th Symposium on Applied Perception in Graphics and Visualization*, APGV '10, pages 89–92, New York, NY, USA. ACM.

Reddy, M. (2001). Perceptually optimized 3D graphics. *IEEE Comput. Graph. Appl.*, 21(5):68–75.

Thrane, N., Simonsen, L. O., and Ørbæk, P. (2005). A comparison of acceleration structures for gpu assisted ray tracing. Technical report.

Wald, I. and Havran, V. (2006). On building fast kd-trees for ray tracing, and on doing that in O(N log N). In *Proceedings of the 2006 IEEE Symposium on Interactive Ray Tracing*, pages 61–70.

Whitted, T. (1980). An improved illumination model for shaded display. *Commun. ACM*, 23(6):343–349.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS