Uma Nova Solução para *Visibility Culling* de Cenas Genéricas, baseada em Replicação, Heurísticas e Redução de *Draw Calls*

Yvens R. Serpa[1], **Mária Andréia F. Rodrigues (Orientadora)**[1]

[1]Programa de Pós-Graduação em Informática Aplicada (PPGIA)
Universidade de Fortaleza (UNIFOR)
Av. Washington Soares, 1321 – Bloco J(30) – Fortaleza – CE – Brazil

yvensre@gmail.com, andreia.formico@gmail.com

Resumo. *Aplicações gráficas com qualidade visual e níveis de interatividade cada vez maiores têm sido de fundamental interesse. Neste contexto, algoritmos de visibility culling restringem o processamento aos objetos efetivamente visíveis pelo observador, acelerando a visualização da cena. Contudo, as soluções do estado-da-arte ainda demandam alto custo computacional, não escalam em cenários complexos e são limitadas quanto à generalização. Em contraste, este trabalho apresenta RHView, uma solução inovadora genérica para cenas estáticas e dinâmicas, baseada em uma estrutura de particionamento espacial replicada e heurísticas. RHView usa heurísticas originais para estimativa de tempo de renderização e de balanceamento entre custo de processamento e precisão na remoção de triângulos, mantendo taxas interativas de quadros/s, mesmo em cenas com bilhões de triângulos. É a única solução ora existente voltada para a redução dos draw calls, um dos fatores de maior impacto no processamento gráfico. Testes sistemáticos mostraram que RHView pode ser até 2,8 vezes mais rápida que os algoritmos do estado-da-arte.*

Abstract. *Graphics applications with visual quality and increasing levels of interactivity have been of fundamental interest. Within this context, visibility culling algorithms restrict the processing to the objects actually visible by the observer, speeding up the scene visualization. However, state-of-the-art solutions still require a high computational cost, do not scale in complex scenarios and are limited in generalization. In contrast, this work presents RHView, an innovative generic solution for static and dynamics scenes, which is based on a replicated space-partitioning structure and heuristics. RHView uses novel heuristics for rendering time estimation and balance between processing cost and triangle removal accuracy, while maintaining interactive frame rates, even in scenes with billions of triangles. It is the only solution currently available to reduce draw calls, one of the factors that have the greatest impact on graphics processing. Systematic tests have shown that RHView can be up to 2.8 times faster than the state-of-the-art algorithms.*

1. Introdução

A geração de imagens com melhor qualidade visual é primordial nas aplicações gráficas (desde a indústria cinematográfica, a área de saúde e a de entretenimento), pois proporciona uma melhor experiência aos usuários e uma maior precisão nos dados sintetizados

por computador. Contudo, demanda um alto custo computacional, quase sempre, comprometendo as taxas de quadros/s ou *frames per second* (FPS) [Mattausch et al. 2008].

Os algoritmos de *visibility culling* destacam-se entre as soluções para restringir o processamento gráfico aos elementos visíveis ao observador [Cohen-Or et al. 2003] e, assim, garantir taxas de FPS interativas. Todavia, o processamento destes algoritmos pode comprometer o FPS gerado e, quanto maior a precisão destes, maior a tendência de aumentar as chamadas à API de renderização, ou *draw calls*, os quais possuem alto custo de processamento [Wloka 2003]. Os algoritmos mais tradicionais de *visibility culling* são: *View-Frustum Culling* (VFC) (verifica os elementos contidos no *frustum* de visualização); *Backface Culling* (BC) (identifica os elementos opostos ao observador); e *Occlusion Culling* (OC) (encontra os elementos ocultos por outros). Algoritmos de OC podem ser classificados em espaço 3D e em espaço de imagem [Robles Ortega et al. 2017]. Algoritmos de OC em espaço 3D projetam volumes de sombra a partir de objetos oclusores e classificam os contidos nestes volumes como invisíveis. Estes algoritmos são pouco flexíveis pois, para garantir eficiência, os volumes e oclusores devem ser convexos e com complexidade geométrica reduzida. Diferentemente, algoritmos de OC em espaço de imagem são mais genéricos. Identificam objetos ocultos e os renderizam em *buffers* auxiliares. Caso o *buffer* auxiliar não seja atualizado, o objeto testado não está visível. Esta renderização extra, porém, tem custo similar à renderização final e aciona *draw calls* adicionais [Hillesland et al. 2002]. Estes algoritmos são geralmente combinados a estruturas de particionamento espacial [Cohen-Or et al. 2003]. Através desta combinação, os algoritmos de *visibility culling* beneficiam-se de características específicas destas estruturas e aumentam seus níveis de granularidade. Diversas soluções em *visibility culling* têm sido propostas, mas a maioria não escala em cenários complexos e é limitada quanto à generalização, sendo aplicável a cenários específicos, com características particulares.

Este trabalho, síntese da dissertação de mestrado detalhada em [Rebouças Serpa 2017], apresenta RHView, uma solução inovadora e genérica para aplicações gráficas em geral, contendo cenas estáticas e dinâmicas [Serpa and Rodrigues 2018]. RHView foca na remoção de triângulos e no uso eficiente de memória, além de ser (até onde tem conhecimento os autores deste trabalho) a única voltada para a redução do número de *draw calls*. RHView compreende os algoritmos de VFC, BC e OC, especialmente adaptados para prover uma maior sinergia e interagir com uma nova estrutura de particionamento espacial baseada em replicação, a RHOctree. Além disso, novas heurísticas também foram formalizadas para a estimativa do tempo de renderização, *visibility culling*, nível de detalhamento dos objetos ou Level-of-Detail (LoD), coerência espacial e coerência temporal.

2. Estimativa de Tempo de Renderização

Desempenho é um fator crítico para a maioria das aplicações gráficas, porém, identificar os fatores que mais impactam no tempo de renderização é tarefa não trivial. Autores como [Wimmer and Wonka 2003] propuseram modelos de estimativa de tempo de renderização. Todavia, o custo dos *draw calls*, como apontado por [Wloka 2003], não é levado em consideração nestas fórmulas. A partir desta motivação, em [Rebouças Serpa 2017] as fórmulas propostas por [Wimmer and Wonka 2003] foram revisitadas, testadas e analisadas detalhadamente. Como resultado, propôs-se um novo modelo, baseado em dois custos principais (um generalizado de fragmentos e vértices renderizados e outro de *draw calls*),

como mostra a Eq. 1: $T(V, N) = c'_1 * V + c'_3 * N$. Onde c'_1 é a constante do custo generalizado de vértices e fragmentos; V, o total de vértices renderizados; c'_3, a constante para o custo de 1 *draw call*; e N, o número de *draw calls* efetuados. RHView foi projetada a partir da análise dos componentes da Eq. 1, sendo composta pelos algoritmos de *visibility culling* (VFC e OC) e pela nova estrutura RHOctree, avançando o estado-da-arte na área.

3. RHView

RHView contém o algoritmo VFC, de simples implementação e baseado na interseção de objetos 3D, na forma de volumes envoltórios. Diferentemente, o algoritmo de OC é uma das técnicas mais complexas e custosas em termos de processamento sendo, portanto, recomendado em situações específicas para evitar o comprometimento do desempenho da aplicação. Para decidir quando usá-lo, RHView estima o custo de renderização da cena aplicando a Eq. 1. Além disso, o OC da nossa solução é implementado em espaço de imagem e usa Occlusion Queries (OQs) [Mattausch et al. 2008], as quais podem processar múltiplos objetos simultaneamente. RHView tem uma gerência diferenciada de OQs para controlar o tempo de espera e requisição de resultados, reduzindo estagnações [Mattausch et al. 2008]. Além disso, os resultados das OQs são usados como entrada para a heurística que determina o LoD dos objetos visíveis.

O uso de LoD para simplificar os objetos dinâmicos ajuda a diminuir o total de triângulos, já que, em algumas situações, mesmo com algoritmos eficientes de *visibility culling*, o total de triângulos visíveis ainda pode ser muito alto. RHView usa heurísticas e os resultados das OQs para estimar o melhor LoD para cada objeto, buscando garantir resultados visuais satisfatórios. A RHOctree é uma variação da Octree tradicional, que mantém cópias da malha dos objetos em seus nós, permitindo diferentes níveis de granularidade. As malhas armazenadas na RHOctree são reordenadas para serem usadas por uma adaptação do algoritmo BC de Zhang [Zhang and Hoff 1997]. A RHOctree reduz eficientemente o número de *draw calls*: (1) usando a replicação de malhas para diminuir os *draw calls* feitos para renderizar cada nó, renderizando nós intermediários completamente visíveis (cujos nós filhos estão totalmente visíveis); (2) executando a ordenação das malhas, para que o BC de Zhang faça, no máximo, 2 *draw calls* por nó; e (3) agrupando malhas que compartilham das mesmas propriedades (*shaders*, texturas, etc.), em malhas únicas, para que sejam renderizadas com um único *draw call*.

Comparando, o número de *draw calls* costuma ser alto nas demais soluções existentes pelos seguintes motivos: suas estruturas renderizam nós folhas, cuja cardinalidade é ordens de vezes maior para cada nível de profundidade; seus algoritmos aumentam a granularidade da renderização, como o BC de Zhang tradicional, realizando, no mínimo, 7 *draw calls* por malha; e devido ao fato de não agruparem ou renderizarem objetos com as mesmas propriedades de uma única vez. A Figura 1 mostra o algoritmo completo de *visibility culling* da RHView. Inicialmente, usa o VFC para testar a visibilidade dos objetos dinâmicos. O tempo de renderização destes é estimado pela Eq. 1 e, caso maior que 0,016s (equivalente a 60 FPS), o módulo de OC é acionado. O módulo de OC renderiza os oclusores da RHOctree usando a adaptação do algoritmo de Zhang e é encerrado quando os objetos dinâmicos são testados, determinando se os mesmos são visíveis ou não. Objetos visíveis são classificados em LoD conforme suas respectivas distâncias ao observador e os fragmentos gerados nos testes. Em seguida, a RHOctree, a partir do nó raiz, verifica o estado dos nós. Caso um nó esteja totalmente visível, é renderizado e seus nós filhos

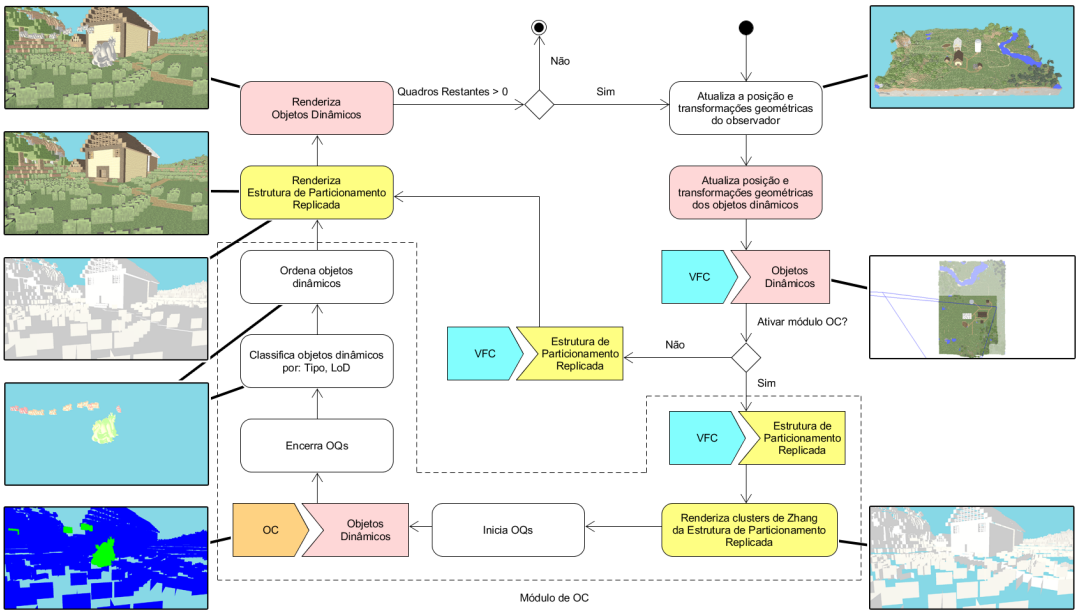

Figura 1. Visão geral da RHView.

são ignorados. Caso contrário, se existirem nós folhas, estes são avaliados. Finalmente, os objetos dinâmicos em LoD são renderizados. A complexidade do algoritmo da RHView é, no pior caso, igual a $2 * (\frac{\#N}{\#Tris})^{\frac{7}{8}} + \#Objs$, sendo: $\#N$ o total de nós folhas da RHOctree, $\#Tris$ o total de triângulos da cena e $\#Objs$ o total de objetos dinâmicos.

4. Testes e Resultados

RHView foi testada sistematicamente em diferentes cenários, compostos por uma cena (uma malha geométrica maior e estática), acrescida de objetos dinâmicos (mudando de posição ao longo da execução e, portanto, sujeitos a transformações geométricas). Diferentes *walkthroughs* foram definidos para cada cenário e executados sequencialmente para todos os algoritmos. Em particular, as soluções do estado-da-arte usadas no estudo comparativo foram: CHC++ [Mattausch et al. 2008] e Nested Grid [Bittner et al. 1998].

Para estudar o comportamento das estruturas tradicionais e das soluções Nested Grid e CHC++, frente à RHView, foi executado um *walkthrough* em uma cena (*Rungholt*) de alta complexidade geométrica e densidade de triângulos, com aproximadamente 6 milhões de triângulos. Os resultados da Tabela 1 mostram que RHView tem desempenho superior frente às estruturas tradicionais e não tradicionais, principalmente, devido ao número reduzido de *draw calls*. No caso específico das estruturas tradicionais, RHView também obtém uma maior precisão na remoção de triângulos. *Nested Grid* gastou um tempo de processamento bastante superior ao das outras abordagens, devido ao uso intensivo de OC na remoção de nós invisíveis, aumentando os *draw calls* ao custo da precisão. Já a CHC++, durante todo o *walkthrough*, mantém-se com tempo de processamento bastante próximo ao da RHView. Similarmente à *Nested Grid*, CHC++ também realiza OC para a remoção de nós invisíveis, porém, com coerência temporal, o que não aumenta significativamente os *draw calls*, mas impacta no tempo de processamento gasto.

Para avaliar as soluções em cenários dinâmicos, um *walkthrough* foi gerado para uma cena de média complexidade (*Fazenda*), contendo cerca de 500 mil triângulos. Nesta

Tabela 1. Resultados para as cenas *Rungholt* e *Fazenda*. Valores médios ao longo do *walkthrough* para Tempo, FPS, Vértices, *Draw Calls* e Objetos Visíveis.

	Rungholt				Fazenda				
Estrutura	**Tempo (s)**	**FPS**	**Vértices**	**Draw Calls**	**Tempo (s)**	**FPS**	**Vértices**	**Draw Calls**	**Objetos Visíveis**
Octree	0,0866	11,53	5.406.270,99	195.831,18	-	-	-	-	-
Nested Grid	0,0137	72,86	582.542,38	1.197,29	0,0291	34,31	88.982.068,27	184,10	33,90
CHC++	0,0030	329,46	1.019.734,06	152,57	0,0281	35,54	89.495.474,06	37,16	33,98
RHView	0,0017	570,06	4.012.392,68	98,46	0,0255	39,11	84.198.522,83	15,52	31,83
RHView (LoD)	-	-	-	-	0,0105	95,08	31.952.939,25	15,52	31,83

cena, foram adicionados 310 objetos (*Stanford Dragon*), cada um com 871.199 triângulos, totalizando um cena com 270.536.526 triângulos. A Tabela 1 mostra que todas as abordagens, com exceção da RHView com LoD, obtiveram tempos médios superiores a 0,016s. O uso de LoD pela RHView garantiu taxas interativas e reduziu significativamente o total de triângulos visíveis, sem prejudicar a qualidade dos resultados. Mesmo sem LoD, a RHView alcança resultados superiores em relação às demais, inclusive, na remoção de triângulos. A Eq. 1 foi usada nesta cena, levando-se em consideração os vértices dos objetos dinâmicos e os *draw calls* adicionais para a renderização destes. Os resultados sem o LoD mostram que o tempo calculado pela Eq. 1 foi bastante próximo ao tempo real, com erro médio de $0,05$s. Para a RHView com LoD, o erro médio foi de $\pm 0,02$s. Animações com os resultados estão disponíveis em `https://tinyurl.com/y8qoqck3`.

5. Conclusão

Este trabalho apresentou RHView, uma nova solução para *visibility culling*, baseada em uma estrutura de particionamento espacial replicada e heurísticas. Uma extensa análise dos componentes que influenciam o tempo de renderização foi feita, possibilitando a definição de heurísticas e equações para a estimativa de tempo de renderização. A partir da equação de estimativa de tempo de renderização, uma estrutura de particionamento espacial de alto desempenho, baseada na redução dos *draw calls* e integrada a mecanismos de coerência temporal, foi proposta e implementada. Testes sistemáticos foram conduzidos em cenas com 500 mil a 1,5 bilhão de vértices para validar a solução proposta, comparando-a tanto com soluções tradicionais de *visibility culling*, quanto a soluções do estado-da-arte (*Nested Grid* e CHC++). Os resultados mostram que RHView apresenta melhores tempos de processamento, assim como reduzido número de *draw calls*, em comparação às demais soluções. Em cenários dinâmicos, RHView apresentou ainda uma maior precisão na remoção de objetos dinâmicos invisíveis, menor número de vértices renderizados, sendo até 2,8 vezes mais rápida que o CHC++.

6. Publicações em Conferências e Periódicos

Duas publicações em veículos qualificados, diretamente relacionadas ao mestrado, foram: [Serpa et al. 2016] e [Serpa and Rodrigues 2018]. Além destas, outras sete publicações relacionadas, relevantes para o conhecimento técnico e científico e importantes para a capacitação em aplicações gráficas a taxas interativas foram: [Rodrigues et al. 2014], [Rodrigues et al. 2015a], [Rodrigues et al. 2015b], [Macedo et al. 2015], [Rodrigues et al. 2016], [Rodrigues et al. 2017] e [Fonteles et al. 2018].

Referências

Bittner, J., Havran, V., and Slavik, P. (1998). Hierarchical Visibility Culling with Occlusion Trees. In *Proceedings of the 1998 CGI*.

Cohen-Or, D., Chrysanthou, Y. L., Silva, C. T., and Durand, F. (2003). A Survey of Visibility for Walkthrough Applications. *IEEE TVCG*, 9(3):412–431.

Fonteles, J. H., Serpa, Y. R., Rodrigues, M. A. F., et al. (2018). Gesture-Controlled Interactive Musical Game to Practice Hand Therapy Exercises and Learn Rhythm and Melodic Structures. In *Proceedings of the 2018 SeGAH*. Aceito para publicação.

Hillesland, K., Salomon, B., et al. (2002). Fast and Simple Occlusion Culling using Hardware-Based Depth Queries. *Chapel Hill: University of North Carolina*.

Macedo, D. V., Rodrigues, M. A. F., and Serpa, Y. R. (2015). Desenvolvimento de Aplicações Gráficas Interativas com a Unreal engine 4. *RITA*, 22(2):181–202.

Mattausch, O., Bittner, J., and Wimmer, M. (2008). CHC++: Coherent Hierarchical Culling Revisited. In *CGF*, volume 27, pages 221–230.

Rebouças Serpa, Y. (2017). Uma Nova Solução para *Visibility Culling* de Cenas Genéricas, baseada em Replicação, Heurísticas e Redução de *Draw Calls*. Master's thesis, PPGIA, UNIFOR. Fortaleza-CE, Brasil.

Robles Ortega, M., Ortega, L., and Feito, F. (2017). Efficient Visibility Determination in Urban Scenes Considering Terrain Information. In *TSAS*, volume 3, pages 10:1–10:24.

Rodrigues, M. A. F., Macedo, D. V., Pontes, H. P., et al. (2016). A Serious Game to Improve Posture and Spinal Health while Having Fun. In *Proc.of the 2016 SeGAH*, pages 1–8. IEEE.

Rodrigues, M. A. F., Macedo, D. V., and Rebouças, Y. S. e. a. (2014). Combatendo a Halitose Um Serious Game Multiplataforma em Saúde Bucal. *Anais do SBGames, SBC*, pages 210–219.

Rodrigues, M. A. F., Macedo, D. V., and Rebouças, Y. S. e. a. (2015a). Um Serious Game em Saúde Bucal Contra o Tártaro. *Anais do SBGames, SBC*, pages 699–702.

Rodrigues, M. A. F., Macedo, D. V., and Serpa, Y. R. e. a. (2015b). Beyond Fun: an Interactive and Educational 3D Traffic Rules Game Controlled by Non-Traditional Devices. In *Proceedings of the ACM SAC 2015*, pages 239–246. ACM.

Rodrigues, M. A. F., Serpa, Y. R., Macedo, D. V., et al. (2017). A Serious Game to Practice Stretches and Exercises for a Correct and Healthy Posture. *ECJ*, pages 1–25.

Serpa, R. Y., Serpa, R. Y., and Rodrigues, M. A. F. (2016). A Comparative Study on a Novel Drawcall-Wise Visibility Culling and Space-Partitioning Data Structures. *Anais do SBGames, SBC*, pages 36–43.

Serpa, Y. R. and Rodrigues, M. A. F. (2018). A Draw Call-Oriented Approach for Visibility of Static and Dynamic Scenes with Large Number of Triangles. *The Visual Computer Journal*, pages 1–15. Disponível em: https://tinyurl.com/y8oo7kx4.

Wimmer, M. and Wonka, P. (2003). Rendering Time Estimation for Real-Time Rendering. In *Proceedings of the 2003 EGSR*, pages 118–129. Eurographics.

Wloka, M. (2003). Batch, Batch, Batch: What Does It Really Mean? *Presentation at GDC*. Disponível em: http://tinyurl.com/glo7o7k.

Zhang, H. and Hoff, III, K. E. (1997). Fast Backface Culling Using Normal Masks. In *Proceedings of the 1997 I3D*, pages 103–ff. ACM.