

Visualização de Terrenos Digitais Utilizando Técnicas Dependentes da Visão

Rafael Moreira Savelli
Roberto de Beauclair Seixas
Anselmo Antunes Montenegro
{savelli,rbs}@impa.br
anselmo@ic.uff.br

Resumo:

São apresentadas técnicas para melhorar a visualização interativa de terrenos digitais tri-dimensionais. Essas técnicas são aplicadas numa estrutura chamada *billboards* nos quais são utilizados com o objetivo de representar os diferentes tipos de vegetações nos típicos terrenos do interior do Brasil. Algumas dessas técnicas podem ser classificadas como sendo *dependente da visão* (ou *view-dependent*) pois são atualizadas em tempo real de acordo com a posição corrente da câmera no terreno.

Palavras chaves:

Visualização de Terrenos, Técnicas Dependentes da Visão e Texturas Multi-resolução.

Abstract:

Present some techniques to improve visualization of a digital featured terrain. We apply these techniques to the well-known *billboard* structure which are used to show different kinds of typical Brazilian vegetations. Some of these techniques can also be classified as being *view-dependent* because they are updated at real-time depending on the current camera position.

Keywords:

Terrain Visualization, View-dependent Techniques and Mult-resolution Textures.

Introdução:

Existem inúmeros problemas ligados à simulação e visualização de terrenos por meio de computadores. Vários desses problemas já foram resolvidos pela computação gráfica graças às muitas contribuições científicas de diversos pesquisadores e estudiosos da área espalhados por todo o mundo.

Entretanto, a demanda por utilizar terrenos cada vez maiores acaba por estimular não só a criação de hardwares mais velozes como algoritmos mais eficientes. Nesse momento surge um problema: quanto maior o terreno, maior tende a ser a quantidade de objetos que devem ser redesenhados interativamente resultando em uma visualização deficiente e com pouco realismo.

Na tentativa de minimizar esse problema, propomos algumas técnicas adaptativas que visam economizar de forma eficiente o tempo de cpu evitando assim a renderização desnecessária de objetos da cena. Mais do que isso, algumas técnicas são capazes ainda de levar em consideração a posição da câmera, fazendo um corte de processamento mais adequado sem prejudicar a qualidade e a interatividade da visualização.

Trabalhos Correlatos:

Existem na literatura diversos trabalhos envolvendo simulação e visualização de terrenos com vegetação. Diversas técnicas e métodos já foram apresentados na tentativa de resolver os mais diversos problemas ligados à forma de representação do terreno e dos objetos da cena. Mais do que isso, vemos também que existem trabalhos mais específicos tratando ambientes complexos onde os objetos da cena são meramente vegetação [1, 2, 3].

Este trabalho é baseado em resultados anteriores onde uma foto de satélite é utilizada como textura para ser aplicada no terreno [4]. Mais que isso, uma aplicação utiliza a própria imagem de satélite para identificaar as posições e classificar os tipos de vegetação para que possam ser inseridas na visualização 3D de forma semi-automática.

Terreno e Modelagem do Billboard:

No geral, algoritmos de visualização interativa de terrenos são muito complexos. E em função dessa complexidade o que se pode observar é que na última década, esse assunto tem recebido muita atenção de diversos pesquisadores de computação gráfica. Como consequência, diversas estratégias foram desenvolvidas. O mais interessante é que, de todas as estratégias, os trabalhos de Lindstrom e Pascucci [5,6] acabaram por fazer grandes contribuições, se tornando uma boa referência.

Os elementos básicos fundamentais na construção do terreno 3D é a combinação de uma malha de triângulos (dito *wireframe*) com uma textura georeferenciada aplicada a esta malha de triângulos conforme mostrada na Figura 1. Através da junção desses dois tipos de dados (*wireframe* e textura), conseguimos exibir algumas informações relativas a geografia do local como é o caso de montanhas e vales (conseguido pelo *wireframe*) e rios e lagos (conseguido pela textura georeferenciada colorida). O resultado final é um terreno 3D vazio que exibe somente elevações e alguns acidentes como rio e lagos [7].

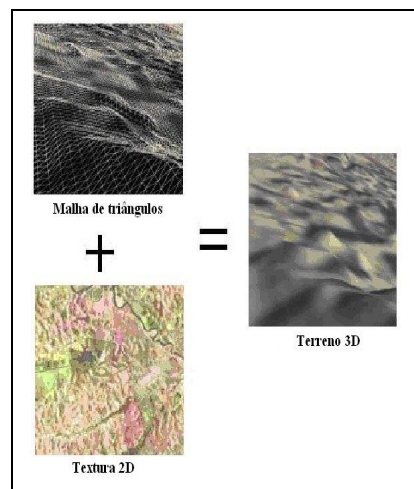


Figura 1: Composição do terreno.

Outra consideração importante inclui a técnica de billboard para representar no terreno outros elementos como, por exemplo, vegetação. Por intermédio de billboards, é possível adicionar no terreno árvores, gramas e outros tipos de vegetação.

Um billboard é uma estrutura geométrica na qual mapeamos uma textura simples. Ele serve para aumentar o realismo e o desempenho quando comparado com a tradicional modelagem gráfica. Na prática, objetos complexos são desenhados como sendo uma textura planar que por sua vez é mapeada em uma geometria simples. Além disso, essa estrutura deve estar sempre orientada de modo a ficar de frente pro observador. A transformação consiste tipicamente em uma rotação no eixo z para orientar o objeto de modo a ficar de frente para o observador [8]. Para o caso da imagem ser uma árvore, um objeto relativamente simétrico, o efeito de uma rotação axial faz com que todo quadrilátero gire, girando também a árvore. A Figura 2 mostra um esquema típico de billboard.

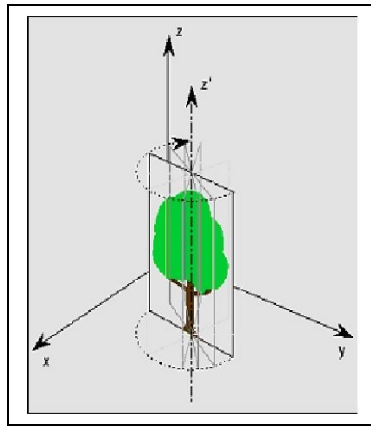


Figura 2: Um billboard esquemático.

No âmbito de melhorar o realismo, uma quarta componente chamada *alpha-channel* foi adicionada à componente rgb da imagem do billboard. Esta componente é responsável por dizer, em cada pixel, o quão opaco ou transparente a textura deve ser. Na prática, a imagem rgba tende a ser transparente nas bordas e opacas no objeto em si, dando a idéia de estar totalmente inserido no contexto da cena [9]. Como exemplo, podemos observar a Figura 3 que mostra um grupo de 3 billboards.

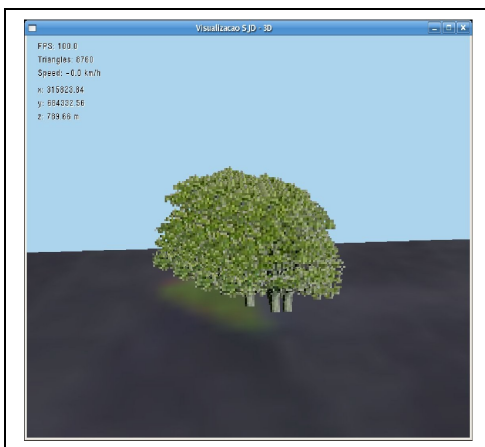


Figura 3: Exemplo de billboards simples.



Figura 4: Exemplo de sprites.

Comparativamente foram feitas experiências utilizando *sprites* para representar objetos da cena. O *Sprite* é parecido com *billboard* entretanto não existe rotação. Ao invés disso, os *sprites* sempre aparecem aos pares com duas estruturas colocadas perpendicularmente entre si na forma de uma cruz. O mesmo grupo de árvores mostrada na Figura 3 como *billboards* agora pode ser observado com *sprites* na Figura 4. Essa tentativa visava economizar processamento com rotação do *billboard*, mas os resultados práticos mostraram exatamente o oposto, contrariando as nossas expectativas. Pelo que foi observado, redesenhar um *sprite* a cada frame é mais computacionalmente custoso do que redesenhar um *billboard*. Acreditamos que para o computador é mais fácil calcular a matriz de rotação a cada frame do que redesenhar duas texturas estáticas. Como pudemos observar na tabela 1, o resultado foi quase o mesmo para ambas estratégias quando o número de vegetação no terreno era relativamente baixo. Entretanto, à medida que mais árvores foram sendo adicionadas ao terreno, a quantidade de quadros por segundos (ou *frame-per-second*) para o *sprite* cai de forma acentuada.

Número de objetos:	Quadros por segundo:	
	<i>Billboard:</i>	<i>Sprite:</i>
50	40.0	40.0
100	40.0	40.0
200	33.3	25.0
300	28.6	20.0
400	25.0	16.7

Tabela 1: Desempenho das estratégias *sprite* e *billboard*.

Os valores da tabela 1 foram obtidos através da visualização do terreno de *Itaoca, ES* em uma computador intel pentium 4 1.6 Ghz com 512 MB de memória RAM e 256 MB de placa de vídeo modelo GeForce FX5200.

Técnicas Dependentes da Visão:

Por razões e tendências óbvias, sabemos que quanto maior o terreno maior será também o número de objetos a serem desenhados. Com isso, dependendo do tamanho do terreno, desenhar todos os objetos pode ser uma tarefa bastante complicada. Assim, na tentativa de contornar esse tipo de problema, alguns artifícios e truques tiveram que ser adicionados à aplicação com o objetivo de deixar mais tempo de cpu disponível para a renderização sem comprometer ao mesmo tempo a qualidade da visualização. Esses artifícios e truques são chamados de técnicas dependentes da visão (*view-dependent*).

O que foi feito foram alguns métodos adaptativos que, dependendo da posição do observador na cena, a aplicação exibe para este observador somente o indispensável para uma boa visualização naquele extato ponto de vista.

Nas próximas sub-seções nós mostraremos os artifícios e as técnicas dependentes da visão ambos utilizados na visualização de terrenos tri-dimensionais.

Alturas aleatórias:

Como as vegetações são lidas por meio de imagens de satélites, fica difícil de saber ao certo qual é a altura de cada árvore. Por esse motivo, precisamos fazer uma estimativa da altura média de uma árvore e fazer pequenas variações em torno dessa média. Essas variações são obtidas de forma aleatória, mas sempre respeitando um limite superior e inferior.

A ideia principal desta técnica consiste em deixar a visualização com um aspecto um pouco mais realístico visto que na natureza podemos encontrar, por exemplo, árvores da mesma espécie porém com altura distintas.

Como sabemos, implementar essa técnica é bem simples e de baixo custo para o processamento de cpu, pois as alturas dos billboards são atribuídas de forma aleatória logo no início do carregamento da aplicação. Por esse motivo, apenas um pequeno esforço inicial se faz necessário. Uma vez feito isso, pouco ou nada afeta o restante da visualização onde o usuário está em constante interação. A Figura 5 mostra como alguns billboards com essa técnica aparecem inseridos em uma típica cena da visualização 3D.

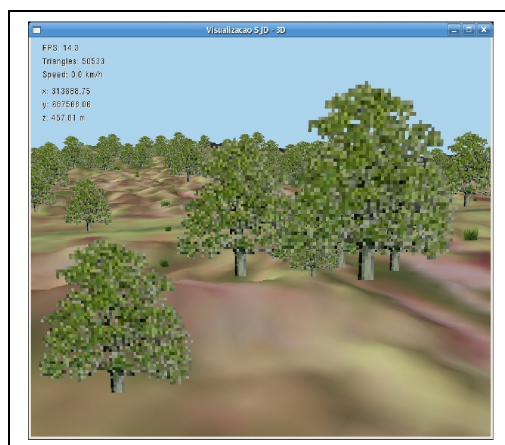


Figura 5: Alturas randômicas.

Clusterização de Billboards:

Outra consideração é o fato de que, quanto mais distante do observador, menos detalhes podem ser observados na vegetação. Isso é uma causa natural da nossa visão e foi feito descartando algumas vegetações relativamente distante do observador, poupando assim, tempo de renderização.

Na Figura 6 podemos identificar 3 grandes grupos de vegetação com diferentes distâncias entre cada um desses grupos e o observador (usuário da aplicação). O grupo 1 está o mais próximo possível do observador bem na parte inferior da figura. Já o grupo 2 se encontra mais ou menos no meio da interface à uma distância intermediária do observador. Por fim, o grupo 3 que se encontra na parte extrema superior da janela do usuário e justamente na maior distância possível do observador. Olhando individualmente para cada grupo, note como as densidades variam de um grupo para outro. O grupo mais distante e por isso menos observado também tem poucos detalhes quando comparado com o grupo mais próximo. Os resultados obtidos através dessa técnica foram realmente bastante satisfatórios.

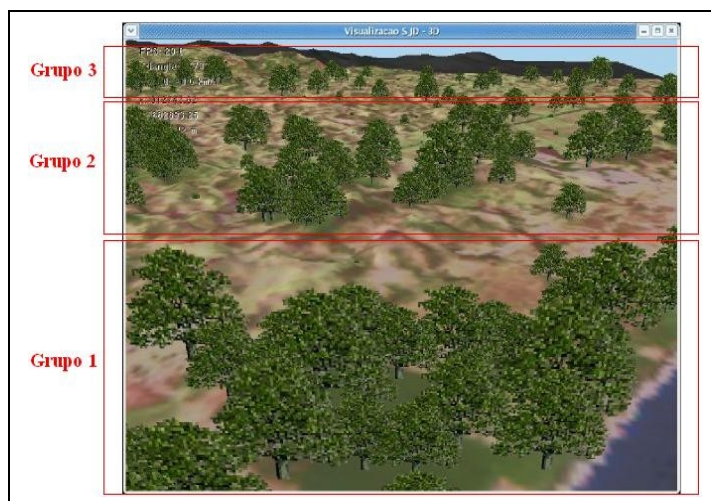


Figura 6: Billboard clustering.

Para conseguirmos tal efeito, aplicamos um conceito conhecido como LOD (do inglês Level-Of-Detail). Essa técnica é utilizada para simplificar modelagens conforme descrita nos trabalhos [10,11], mas aqui ela foi adaptada para realizar descartes de alguns *billboards* que estão suficientemente distantes do observador. A idéia é fazer isso de modo progressivo e sem prejudicar a visualização. Para isso, dividimos a parte do terreno que de fato é visualizada pelo observador em três regiões distando d_1 , d_2 e d_3 do observador conforme mostrada na Figura 7. A região mais próxima do observador exibe 100% dos elementos. A região intermediária exibe apenas 80% de

toda vegetação presente nessa região. Já a terceira e última região exibe somente 50% dos elementos. A partir de uma certa distância maior do que d_3 do observador nenhuma vegetação é mostrada.

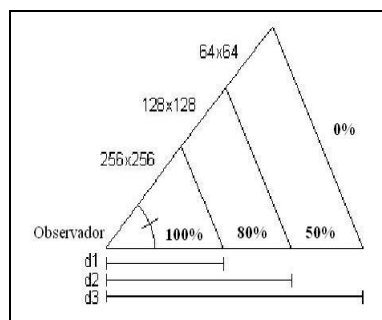


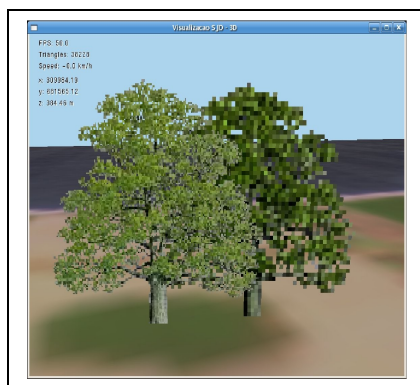
Figura 7: Distribuição dos billboards multi-textura em função da distância.

Billboards com Multi-resolução:

Ainda com a tentativa de melhorar as taxas interativas, passamos a trabalhar com multi-textura nas imagens utilizadas pelos billboards. Conforme podemos ver na Figura 8, existem duas árvores com duas resoluções diferentes. A da esquerda tem 256x256 pixels de resolução. Já a da direita tem apenas 64x64.

A idéia consiste em economizar processamento de cpu através da substituição automática e interativa da textura de cada billboard da cena. Essa troca é feita de modo a atribuir texturas com resoluções mais altas para aqueles billboards mais próximos do observador e resoluções mais baixas para aqueles mais distantes.

Para o nosso estudo de caso, utilizamos três resoluções diferentes para as texturas dos billboards tal como descrito no trabalho [2]. Ou seja, utilizamos a resolução de 64x64, 128x128 e a 256x256 pixels. A Figura 7 mostra como fizemos para distribuir os billboards multi-texturas em função das distâncias d_1 , d_2 e d_3 .



**Figura 8: Árvores com texturas diferentes.
A esquerda tem 256x256 e a direita tem 64x64.**

Resultados finais:

Cada técnica dependente da visão vista até agora contribui de alguma forma para baixar o processamento de cpu sem prejudicar a visualização. Entretanto, cada técnica em separado não faz uma mudança significativa já que seus ganhos são modestos e limitados. A diferença maior se percebe no momento em que se combina todas essas técnicas.

Começamos por combinar as técnicas de clusterização de billboard com a de multi-textura. Isso foi razoavelmente simples pois uma vez dividido o campo de visão do observador em áreas conforme já mostradas na Figura 8, definir um percentual de clusterização para cada área foi

simples

Para agregar a técnica de alturas randômicas não foi difícil. Aplicamos em todas as regiões e em todos os *billboards* igualmente, respeitando somente o limite mínimo e máximo em torno da média para cada espécie de vegetação utilizada na visualização.

O resultado final pode ser observado com grande realismo e com taxas interativas apreciáveis. As Figuras 9, 10 e 11 mostram como ficaram as visualizações após a junção de todas essas técnicas.

Trabalhos Futuros:

Um dos mais óbvios trabalhos futuros consiste em continuar pesquisando outras formas dependentes da visão e ir incorporando ao que já fora feito até então. A idéia é estar sempre melhorando as taxas interativas sem comprometer o realismo como um todo.

Outro possível trabalho poderia ser através do aprimoramento das técnicas já discutidas aqui. Um exemplo disso é o fato de que quando o billboard ultrapassa uma distância máxima, a aplicação simplesmente deixa de redesenhá-lo e ele desaparece da aplicação. Mesmo estando relativamente longe do observador, essa ação às vezes pode ser drástica de mais. Uma outra abordagem seria fazer com que esses billboards que estão na iminência de desaparecer fosse sumindo aos poucos com *fade*, por exemplo. Isso faria uma retirada bem mais suave do elemento da cena.

Conclusão:

A primeira conclusão deste trabalho foi o fato de que billboards funcionam mais eficientemente do que sprites. Através da experimentação, vimos que é computacionalmente mais fácil calcular a matriz de rotação para cada billboard do que redesenhar sprites. Vimos também que, se for muito pequeno o número de objetos na cena, as duas estratégias funcionam bem. Mas se o número de objetos crescerem, sprites podem gerar certas complicações.

A segunda conclusão refere-se a importância de elementos com estratégias dependentes da visão em terrenos tri-dimensionais mesmo com todo o recente avanço nos recursos em *hardware* disponível hoje em dia. Percebemos que, com a necessidade de se trabalhar cada vez mais com grandes terrenos, espaço para desperdícios de processamento passam a ser não mais tolerados.

Bibliografia:

- [1] Lluch, J., Camahort, E., Vivo, R.: An image based multiresolution model for interactive foliage rendering. *Journal of WSCG'04* 12(3) (2004) 507–514.
- [2] Aleks Jakulin. “Interactive Vegetation Rendering with Slicing and Blending”. *Eurographics 2000 Short Pre-sentations*, August 2000.
- [3] Andreas Dietrich, Carsten Colditz, Oliver Deussen, and Philipp Slusallek. “Realistic and Interactive Visualization of High-Density Plant Ecosystems”. In *Natural Phenomena 2005, Proceedings of the Eurographics Workshop on Natural Phenomena*, pages 73–81, 2005.
- [4] Savelli, R. M.; Seixas, R. B. “Semi-Automatic Detection of Vegetations in Digital Satellite Images for Building 3D Terrains”. *The Sixth Iasted International Conference On Visualization Imaging And Image Processing*, (0531-046): 494-498, August 2006.
- [5] P. Lindstrom, V. Pascucci, “Visualization of Large Terrains Made Easy”, *Proceedings of IEEE Visualization*, San Diego, California, October, 2001, 363-370.

[6] P. Lindstrom, V. Pascucci, Terrain Simplification Simplified: “A General Framework for View-Dependent Out-of-Core Visualization”, IEEE Transactions on Visualization and Computer Graphics, May 8, 2002.

[7] E. Poyart, P. Frederick, R.B. Seixas, M. Gattass, “[Simple Real-Time Flight Over Arbitrary-Sized Terrains](#)”, pre-print submitted to Workshop Brasileiro de GeoInformática, 2002.

[8] McReynolds T. and Blythe D., “Advanced graphics programming techniques using OpenGL”. SIGGRAPH, 1998.

[9] T. D. M. Woo, J. Neider, “OpenGL Programming Guide: The Official Guide to learning OpenGL”. Addison-Wesley Professional, 1999.

[10] Bradford Chamberlain, Tony DeRose, Dani Lischinski, David Salesin and John Snyder. “Faster rendering of complex environments using a spatial hierarchy”. In *Proceedings of Graphics Interface '96*, May 1996.

[11] Thomas A. Funkhouser and Carlo H. Séquin. “Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments”. In *Computer Graphics Proceedings, Annual Conference Series*, pp. 247-254, August 1993.



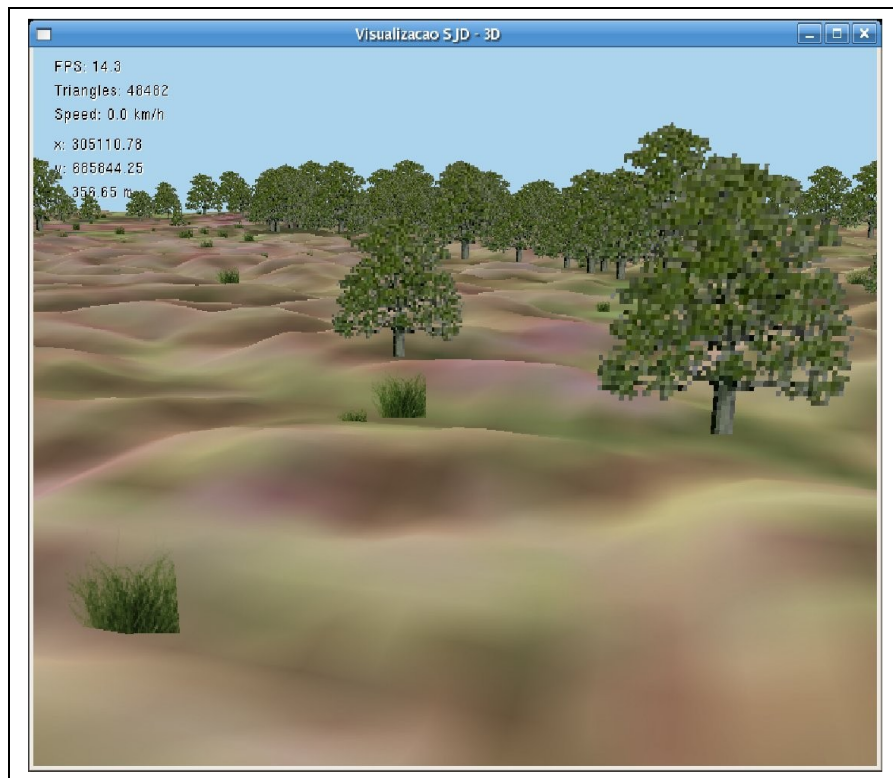


Figura 10: Resultado final com todas as técnicas dependentes da visão apresentadas nesse trabalho.

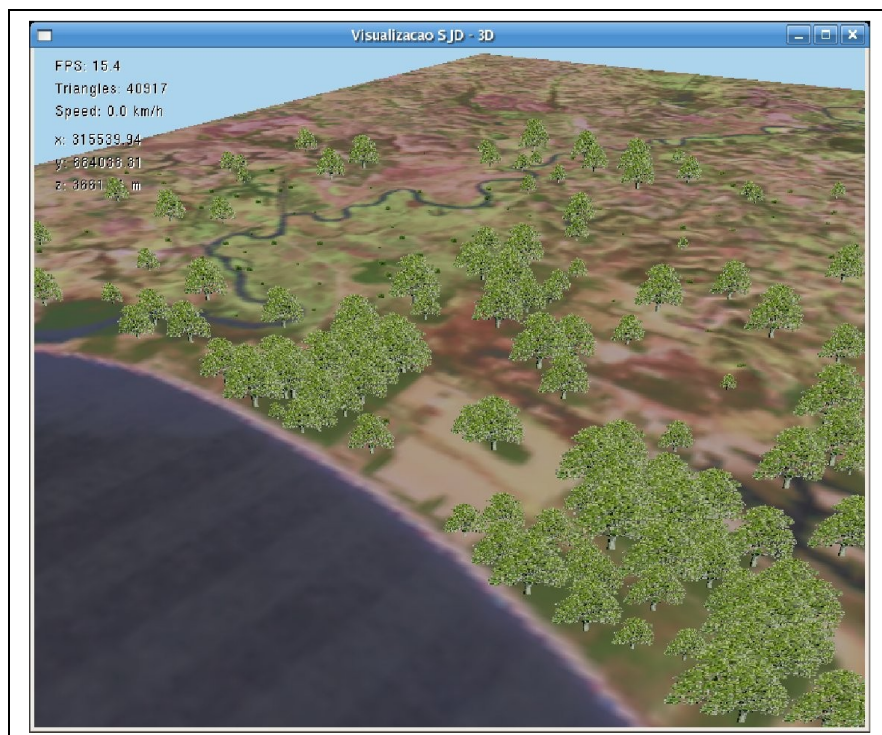


Figura 11: Resultado final com todas as técnicas dependentes da visão apresentadas nesse trabalho.