Universidade Federal dos Vales do Jequitinhonha e Mucuri
Faculdade de Ciências Exatas e Tecnológicas
Departamento de Computação

Bacharelado em Sistemas de Informação

Simulação e visualização de árvores tridimensionais em povoamentos florestais

Liliane Soares da Costa

Diamantina
27 de novembro de 2014

Universidade Federal dos Vales do Jequitinhonha e Mucuri
Faculdade de Ciências Exatas e Tecnológicas
Departamento de Computação

Liliane Soares da Costa

Simulação e visualização de árvores tridimensionais em povoamentos florestais

Trabalho apresentado ao Curso de Bacharelado em Sistemas de Informação do Departamento de Computação da Universidade Federal dos Vales do Jequitinhonha e Mucuri como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: *Prof. Dr. Alessandro Vivas Andrade*
Co-orientador: *Prof. Dr. Reynaldo Campos Santana*

Diamantina
27 de novembro de 2014

Esta folha deve ser subtituída pela cópia da folha de aprovação.

À minha família.

Agradecimentos

Agradeço a Deus, minha força maior, por sempre me acolher e propiciar momentos únicos e conquistas tão importantes. À minha mãe (*in memorian*) pela educação e amor a mim dedicados em vida, e ainda hoje por sentir sua proteção. Ao meu pai por me ensinar a ser uma pessoa batalhadora assim como ele foi e ainda é.

Aos meus oito irmãos, pela força da união, sempre batalhar e torcer pela vitória uns dos outros. Cada um tem um papel fundamental na minha vida. Agradeço especialmente a minha irmã Lenice, por ter se tornado uma mãe para mim, não medindo esforços para que meus sonhos sejam alcançados; e a minha irmã e melhor amiga Eliane, pelo seu amor, compreensão e companheirismo desde sempre.

Ao meu namorado André, por me acompanhar durante todo esse trajeto, me dando força, carinho e amor. Me ensinando e incentivando sempre. Obrigada por ser um dos grandes responsáveis pela minha formação.

Aos mestres responsáveis por me passar conhecimentos e desafios, especialmente aos professores Alessandro Vivas e Reynaldo Santana pela confiança no meu trabalho, pela oportunidade de realizá-lo, pelas ideias para melhorá-lo e por estarem sempre disponíveis quando precisei.

Aos amigos e colegas conquistados ao longo da vida, por ter me trazido aprendizado e momentos inesquecíveis.

Enfim, a todos que acreditaram nas minhas conquistas, meu sincero muito obrigada.

Grandes realizações são possíveis
quando se dá importância aos pequenos começos.
—LAO-TSÉ

Resumo

A visualização de paisagens naturais vem sendo muito utilizada como uma ferramenta para estudos das alterações ambientais e planos de gestão, especialmente em relação a ambientes florestais. Com o avanço dos recursos computacionais, o uso de ferramentas para criação de ambientes virtuais tridimensionais torna mais realista a visualização destas paisagens naturais. Assim, um software para simulação e visualização de floresta em três dimensões é uma ferramenta de grande utilidade para o setor florestal e para aplicações no meio acadêmico, auxiliando a tomada de decisões. Sendo assim, este trabalho apresenta o sistema desenvolvido, que tem como principal objetivo a visualização de florestas tridimensionais e a simulação de crescimento das árvores. Para a implementação do software é utilizada a linguagem de programação Java, pois esta é de fácil portabilidade entre diferentes sistemas operacionais. E para a criação dos ambientes virtuais tridimensionais é utilizada a API (*Application Programming Interface*) Java 3D, que através do uso de uma hierarquia de classes com alto nível de abstração, não exige do desenvolvedor conhecimento sobre detalhes de implementação de hardware gráfico ou algoritmos de computação gráfica. O sistema possibilita a renderização de diferentes modelos de árvores, cada qual com suas vantagens e desvantagens, procurando representar árvores de forma mais realista. A renderização de florestas no sistema se dá através de leitura de arquivo (planilhas do Excel no formato XLS). A partir dessa renderização, é possível, com modelos matemáticos, fazer a simulação do crescimento futuro da floresta com base na idade, sendo que os dados obtidos da simulação podem ser salvos em arquivo. O software possui uma interface amigável e intuitiva para facilitar a interação com o usuário, que pode visualizar a floresta por diversos ângulos, através de translação e rotação. Contudo, o sistema poderá ser uma ferramenta auxiliar a profissionais e acadêmicos das áreas florestais, auxiliando em estudos de dados e à tomada de decisão.

Palavras-chave: Visualização de paisagens. Florestas tridimensionais. Simulação de crescimento.

Abstract

The visualization of natural landscapes has been widely used as a tool for studies of environmental changes and management plans, particularly in relation to forest environments. With the advancement of computational resources, the use of tools for creating three-dimensional virtual environments becomes more realistic the visualization of these natural landscapes. Therefore, a software for simulation and visualization of forest in three dimensions is a very useful tool for the forest sector and for applications in academia, aiding decision making. Thus, this paper presents the developed system, which has the main objective to visualize three-dimensional trees and simulate growth of forest. To implement this software it was used the programing language JAVA because it is easily portable between different operating systems. To create the three-dimensional virtual environments it was used the Java 3D API (Application Programming Interface), which through the use of a hierarchy of classes with high level of abstraction, it does not require from the developer knowledge about graphics hardware implementation details, or algorithms of computer graphics. The system enables the rendering of different tree models, each one has its advantages and disadvantages, trying to represent trees more realistically. The rendering of forests in the system is through of the reading file (Excel spreadsheets in XLS format). From this rendering, it is possible, with mathematical models, to simulate future forest growth based on the age, and the obtained data from the simulation can be saved to file. The software has a friendly and intuitive interface to facilitate user interaction. The user can visualize the forest from different angles, using translation and rotation. Therefore, the system could be a tool to help professionals and academics in forest areas, assisting in studies date and decision making.

Keywords: Landcape visualization. Three-dimensional forests. Growth simulation.

Sumário

Lista de Figuras

Lista de Algoritmos

CAPÍTULO 1

Introdução

A rápida evolução vista no âmbito da tecnologia da informação e a sua utilização vem revolucionando a forma em que a sociedade vive e trabalha. As últimas décadas apresentaram uma crescente aceleração no processo de desenvolvimento de tecnologias nas mais diversas áreas, e a computação gráfica não poderia ficar alheia a este processo.

Atualmente a quantidade de informações geradas é extensa nos diferentes setores de trabalho. Essas informações necessitam ser analisadas e compreendidas para que possam produzir conhecimento, auxiliando assim a tomada de decisões. Dessa maneira, o uso de tecnologias de visualização com recursos de imagens tridimensionais como ferramenta de síntese está se tornando comum, o que permite transformar grande volume de dados em informações relevantes.

A *visualização* pode ser definida como o conjunto de métodos que permitem a interpretação de informações relevantes em conjuntos de dados provenientes de funções de difícil reconstrução, com o auxílio de técnicas de computação gráfica (Gattass, 2013).

A visualização tornou-se uma das mais poderosas ferramentas disponíveis para transformar os milhares de dados gerados diariamente em informações pertinentes, permitindo ao homem sintetizar e analisar grande volume de dados (McCormick *et al.*, 1987). No entanto, a visualização facilita a compreensão dos fenômenos descritos pelos dados, além da elaboração de hipóteses e da descoberta de conhecimento sobre esses fenômenos. Ela também é aplicada como forma eficaz de comunicação e como ferramenta de apoio em inúmeras áreas, como a medicina, engenharia, matemática e modelagem ambiental (Wagener e Kelleher, 2011; Ware, 2012).

Imagens tridimensionais podem ser classificadas de acordo com as seguintes características: pré-processamento, visualização, manipulação e análise (Udupa, 1999). Estas operações são altamente interdependentes. Por exemplo, alguma forma de visualização é essencial para facilitar as outras três classes de funcionamento. Da mesma forma, a definição do objeto através de um conjunto apropriado de operação de pré-processamento é vital para a visualização eficaz, manipulação e análise do sistema objeto. Usa-se o termo imagens tridimensionais (3D)

para se referir coletivamente a estas quatro classes de operações.

O objetivo principal de imagens tridimensionais é fornecer informações qualitativas e quantitativas sobre um objeto ou sistema de objetos a partir de imagens obtidas com várias modalidades.

A percepção da importância de tecnologias de imagens tridimensionais possibilita investimentos por parte de indústria e comércio. Portanto, o uso dessas tecnologias já se aplica na maioria das áreas, por exemplo, na visualização de paisagens – área utilizada no desenvolvimento desse trabalho.

De acordo com Wang *et al.* (2006), a visualização de paisagens é uma ferramenta muito mais poderosa e eficiente para facilitar o entendimento de sistemas complexos, quando a compara com os formatos de comunicação verbal ou numérico e a visualização de dados.

A visualização tridimensional de florestas e paisagens tem sido uma ferramenta muito útil na compreensão da dinâmica de florestas e paisagens para assessorar várias práticas de gerenciamento e planejamento (Sheppard e Salter, 2004). A aparência das paisagens e dos stands individuais, após as operações de colheita é fundamental para a aceitação pública (McGaughey, 1997). Sistemas de visualização de floresta podem ser utilizados para o estudo da estrutura das florestas, dinâmica, transformações de paisagens e planos regionais (Xie *et al.*, 2010).

A visualização tridimensional combinada com modelos de simulação pode mostrar as consequências imprevistas de nossas atividades em processos ecológicos no espaço e no tempo (Lim e Honjo, 2003). Tais simulações permitem a criação de um modelo simplificado, onde várias situações sobre um determinado assunto podem ser exploradas, oferecendo assim a possibilidade do pesquisador desenvolver hipóteses, testá-las, analisar resultados e refinar os conceitos. A visualização tridimensional proporciona uma visão mais abrangente, detalhada e realista em tempo real (Silva e Araújo, 2008).

Diante disso, este trabalho apresenta um sistema de visualização tridimensional e simulação de crescimento de árvores em povoamento, desenvolvido em uma parceria do curso de Sistemas de Informação com a Engenharia Florestal da Universidade Federal dos Vales do Jequitinhonha e Mucuri.

1.1 Objetivos

O objetivo geral deste trabalho foi desenvolver um software de visualização para ser utilizado como sistema auxiliar na tomada de decisões no manejo da floresta.

Para atingir este objetivo, foram definidos os seguintes objetivos específicos:

- implementar um ambiente para visualização de árvores 3D, utilizando os modelos estatísticos implementados, a partir de dados de parcelas permanentes;
- desenvolver uma interface de usuário amigável e intuitiva para manipulação do ambiente de visualização 3D;
- simular o crescimento de árvores individuais em função da idade, apresentando o resultado graficamente;

1.2 Estrutura do trabalho

No Capítulo 2 são apresentados conceitos comuns da Computação Gráfica, tecnologias para a criação de aplicações gráficas tridimensionais e exemplos de aplicações na visualização de paisagens florestais.

No Capítulo 3 são apresentados conceitos sobre as ferramentas utilizadas no desenvolvimento deste trabalho: a linguagem Java e a API Java 3D, responsável pelo desenvolvimento e apresentação de imagens tridimensionais. Também é apresentado o funcionamento desta API com maiores detalhes, assim como a sua utilização e seus principais componentes.

O Capítulo 4 apresenta a metodologia utilizada na criação dos modelos 3D e do sistema proposto. É feita a descrição do programa, suas funcionalidades e o modo como foram implementadas. São apresentados trechos do código do sistema sobre os pontos relevantes, sendo que esses trechos são fundamentais para que haja o entendimento da lógica utilizada no programa.

Por fim, no Capítulo 5 são apresentadas as conclusões obtidas com o desenvolvimento deste trabalho e recomendações para aprimoramentos futuros.

Não faz parte do escopo desta pesquisa avaliar as ferramentas de visualização e simulação existentes, nem a avaliação de qual API de computação gráfica é mais eficiente, e sim,

apenas desenvolver um sistema em conjunto com a Engenharia Florestal e que atenda as suas necessidades, auxiliando no processo de tomada de decisão.

CAPÍTULO 2

Referencial Teórico

Tecnologia de recursos gráficos vem se tornando cada vez mais um item indispensável em áreas variadas. Sendo assim, a Computação Gráfica e as suas ferramentas são largamente utilizadas no intuito de trazer facilidade, rapidez e qualidade na manipulação de informação. Na visualização de paisagens florestais, exemplos mostram benefícios da aplicação de computação gráfica na tentativa de renderizar paisagens próximas a paisagens reais, facilitando assim o entendimento da dinâmica do crescimento das florestas.

2.1 Computação Gráfica

A Computação Gráfica é uma área da Ciência da Computação que se dedica ao estudo e desenvolvimento de técnicas e algoritmos para a geração, manipulação e análise de imagens através do computador (Manssour, 2006). Esta é uma das áreas de maior expansão e importância que possibilita o desenvolvimento de trabalhos multidisciplinares, estando presente em quase todas as áreas do conhecimento humano.

O nicho mais conhecido da Computação Gráfica é o entretenimento, como jogos eletrônicos e efeitos especiais do cinema. No entanto, ela se aplica à diversas outras áreas. tais como: engenharia, com a simulação de eventos físicos e químicos; geoprocessamento, para geração de dados relacionados a regiões; medicina, na análise de exames como tomografia, radiografia e cirurgias com realidade aumentada, entre outras áreas (Martins *et al.*, 2009).

A Computação Gráfica foi dividida em três grandes áreas: síntese de imagens, processamento de imagens e visão computacional (Gattass, 2013).

- Síntese de imagens: considera as representações visuais de objetos criados pelo computador a partir de especificações geométricas e visuais dos seus componentes.
- Processamento de imagens: envolve técnicas de transformação de imagens visando me-

lhorar as características visuais.

- Visão Computacional: área que procura obter a especificação dos componentes de uma imagem a partir de sua representação visual.

2.1.1 Ferramentas

Existem diversas tecnologias para a criação de aplicações gráficas tridimensionais, como OpenGL (Shreiner, 2010), VRML (Walsh e Bourges-Sevenier, 2001) e Java 3D (Sowizral *et al.*, 1998). Nas sub-seções seguintes serão apresentadas estas tecnologias de maneira breve.

2.1.1.1 OpenGL

OpenGL (*Open Graphics Library*) é largamente utilizado para a programação gráfica em 3D. As maiores vantagens desta biblioteca são: estabilidade, segurança e portabilidade. O padrão OpenGL começou a ser desenvolvido em 1992 pela empresa Silicon Graph, sendo oferecido para diversas plataformas como: Unix, OS/2, MS Windows e Linux. Seu código pode ser colocado juntamente com códigos em linguagens C, C++, Object Pascal, Java, ADA e Fortran (Bicho *et al.*, 2002; Pinheiro, 2006).

OpenGL é uma API multiplataforma e multilinguagem que possibilita a criação de aplicações gráficas computacionais em duas ou três dimensões (Tufte, 2001). Através desta ferramenta é possível desenvolver aplicativos tridimensionais realistas e complexos que podem ser utilizados para simulações que auxiliem no processo de aprendizado e tomada de decisão.

OpenGL possui sua especificação aberta, com uma biblioteca de rotinas gráficas e de modelagem, ou seja, uma API utilizada para o desenvolvimento de aplicações de Computação Gráfica, tais como jogos e sistemas de visualização oferecendo um controle simples e direto, permitindo ao programador a especificação de objetos e as operações necessárias para a criação de imagens gráficas de alta qualidade (Tufte, 2001).

2.1.1.2 VRML

VRML (*Virtual Reality Modeling Language*) é uma linguagem de descrição de cenas 3D na forma de texto. Assim, por meio de qualquer processador de textos, um desenvolvedor

pode gerar um mundo virtual (Cardoso, 2003). Ela permite a Computação Gráfica 3D interativa na Web através de uma linguagem textual para descrição de cenas e ambientes interativos em 3D (Lim e Honjo, 2003). Dessa maneira, fornece construções para a especificação da forma, tamanho e aparência de objetos, assim como de sua localização na cena 3D. Os navegadores interpretam essa descrição e a exibem na tela.

O VRML tem diversas funções como sombreamento, projeção e mapeamento de texturas. Apesar de ser uma linguagem para Web ela pode ser rodada localmente, já que todo o processamento da cena 3D é feita pelo próprio navegador.

Para visualizar documentos VRML, é necessário uma aplicação *helper* ou um *plug-in*. Um *helper* consiste em um programa que entende o conteúdo e o formato destes outros tipos de informações, e um *plug-in*, por sua vez, é um programa que permite visualizar informações que não sejam HTML dentro da janela do navegador (Manssour, 2000).

2.1.1.3 Java 3D

A API (*Application Programming Interface*) Java 3D é usada no desenvolvimento de ambientes virtuais 3D, como é o da aplicação apresentada nos próximos capítulos deste trabalho. Esta API possibilita o desenvolvimento de aplicações gráficas, fornecendo as ferramentas para criar, visualizar e manipular geometria 3D. É possível aplicar os conceitos de programação orientada a objetos nas aplicações, e os elementos da cena tridimensional são organizados em uma estrutura de dados chamada de grafo de cena.

O Capítulo 3 é dedicado à descrição desta API.

2.2 Aplicação de computação gráfica na visualização de paisagens florestais

A visualização de paisagens naturais vem sendo muito utilizada como uma ferramenta para estudos das alterações ambientais e planos de gestão, especialmente em relação a ambientes florestais (Tang e Bishop, 2002). Diversos trabalhos presentes na literatura apresentam como essas ferramentas podem ser utilizadas, especialmente os sistemas de visualização 3D de florestas, os quais são utilizados em tarefas como as simulações de florestas reais, crescimento de florestas, propagação de fogo, entre outras.

A visualização de paisagem é uma ferramenta muito útil na prática de gestão e na compreensão da dinâmica de paisagens. Pesquisadores e gestores estão fazendo, cada vez mais, o uso de ferramentas de modelagem computacional e ferramentas de visualização de paisagens que lhes permitam avaliar melhor as práticas de colheita, verificar variações existentes na floresta e comunicar o impacto de mudanças ambientais (Meitner *et al.*, 2005; Tao e Jian-hua, 2009; Uusitalo e Orland, 2001; Wang *et al.*, 2006). Essas ferramentas fornecem também a oportunidade de visualizar as opções de gestão antes das atividades serem executadas e podem ajudar na redução de dificuldades de comunicação e compreensão da complexidade das florestas.

A interação humana com a paisagem modifica as variáveis florestais no decorrer do tempo. Diante disso, softwares e métodos de visualização foram desenvolvidos para recriar paisagens naturais e ilustrar mudanças na cobertura da terra utilizando dados temporais reais (Dunbar, 2003).

House *et al.* (1998), em um trabalho envolvendo diferentes áreas, apresentam uma ferramenta de visualização altamente realista que simula uma floresta existente. O objetivo era determinar o nível de detalhes necessário nas visualizações geradas por computador, de forma a demonstrar que o ambiente virtual pode ser semelhante ao real. No entanto, para que tal nível de detalhes seja possível, é necessário um complexo trabalho de modelagem, o que requer muitos recursos computacionais, como memória e processamento gráfico.

Outro sistema para simulação de cenários naturais é apresentado por Shihong *et al.* (2010). O sistema construído consegue reproduzir árvores tridimensionais, casas na região e paisagens. Os troncos das árvores são modelados por cilindros, com a aplicação de uma textura para deixá-los semelhantes às árvores reais. Em comparação ao trabalho de House *et al.* (1998), o uso de um simples cilindro perde em nível de detalhe para um modelo mais complexo, no entanto, permite a representação de regiões maiores, com mais árvores. As folhas são modeladas por várias faces com uma textura de folhagens com fundo transparente. Para modelar as arquiteturas antigas, como as casas, são utilizadas apenas formas geométricas básicas, como cubos e cilindros. Para tornar o modelo mais real, é trabalhado também a simulação da luz e do céu.

Foi desenvolvida por Falcão *et al.* (2006) uma ferramenta de visualização real de florestas para produzir simulações dinâmicas de paisagens florestais. A ênfase do trabalho é sobre as estratégias para lidar com a complexidade geométrica de áreas florestais de grande porte. As árvores são representadas por painéis cruzados (Figura 2.1), por meio destes a árvore pode

ser visualizada de qualquer posição que o observador estiver. Esta técnica foi escolhida por ser adequada na representação da maioria das espécies e por demandar menos recursos computacionais se comparado à modelagem de objetos 3D (formado por polígonos) para representação das árvores.

Figura 2.1 Representação de árvores 3D a partir de painéis cruzados. (Adaptado de Falcão *et al.* (2006)).

Com o objetivo de demonstrar o potencial de visualização através de computador como uma ferramenta para analisar e comunicar resultados de estudos de mudanças florestais, Dunbar (2003) utiliza uma combinação de técnicas de sensoriamento remoto, geoprocessamento e técnicas de visualização 3D para construir paisagens realistas e animações representando a mudança da cobertura florestal.

A renderização de paisagens de florestas com grande número de árvores é uma tarefa complexa devido ao grande número de polígonos utilizados na representação de florestas. Para melhorar o desempenho, a partir da ausência de sistemas específicos para visualização e simulação simultânea de forma eficiente, Lim e Honjo (2003) desenvolveram um sistema com esse objetivo. Nesse trabalho os autores testam a criação de modelos de árvores tridimensionais através do uso de polígonos complexos para modelar uma árvore real, polígonos básicos (cilindro e cone), e uso de imagens no formato GIF com transparência. O sistema gerado possibilitou a renderização de milhares a dezenas de milhares de árvores, permitindo "caminhar" pelo modelo tridimensional e simular alterações naturais e artificiais.

Bao *et al.* (2009) propõe um sistema que utilizam um modelo híbrido para representação de árvore, o tronco e os galhos são representados por polígonos e as folhas por cruzamento de painéis dando um efeito mais realístico em relação ao uso de apenas polígonos básicos.

Com isso foi obtido um melhor desempenho em relação ao uso de polígonos complexos para modelagem das árvores.

Para possibilitar a interação do usuário com o sistema, Fan *et al.* (2011) desenvolveram uma ferramenta para simular a evolução de florestas. Para possibilitar uma visualização rápida e realista da paisagem florestal virtual, eles utilizam uma estratégia em que são adicionadas imagens de textura de diferentes espécies de árvores sobre a geometria simplificada de um modelo de árvore. Esse modelo é constituído por dois cilindros, tronco e copa cilíndrica.

Yongjian *et al.* (2009) abordam a simulação de árvores em faixas etárias diferentes e visualização em tempo real. Colocam intervalos para as características de cada espécie, onde árvores de diferentes idades têm diferentes números de ramificação. Porém, a técnica utilizada demanda muito tempo e recursos para renderizar grandes cenas.

Assim como no trabalho de Yongjian *et al.* (2009), Xie *et al.* (2010) desenvolveram um sistema que permite a visualização em tempo real possibilitando a interação do usuário com o sistema, demonstrando que ferramentas para visualização 3D são promissoras no apoio a decisão no manejo florestal, permitindo uma análise do impacto visual.

Com intuito diferente dos trabalhos apresentados até agora que tem o foco na representação de visualização mais realistas, Kose *et al.* (2008) propõe um sistema de visualização que simula a propagação do fogo em florestas no decorrer do tempo. Esse sistema pode facilitar o trabalho de bombeiros, pois pode indicar onde o fogo estará em um dado momento facilitando o controle do mesmo. Ele também pode calcular a propagação do fogo em uma paisagem com diferentes condições e produzir simulações realistas usando a informação geográfica em uma determinada região.

Com similaridades a estes trabalhos apresentados, o atual trabalho tem como objetivo o desenvolvimento de um ambiente que possibilita a renderização de florestas tridimensionais a partir de dados reais de florestas, permitindo a simulação do crescimento e a geração de relatórios com dados importantes que variam com a idade das árvores.

No sistema desenvolvido é utilizado os modelos propostos de forma que o usuário possa definir o nível de detalhes em que deseja que a visualização 3D seja gerada, possibilitando que ele defina um *trade-off* entre o tamanho da floresta renderizada e o nível de detalhes, além de permitir que o sistema também possa ser executado em computadores modestos em relação a memória e processamento gráfico.

CAPÍTULO 3

Ferramentas Utilizadas

Para a implementação do software foi utilizado a linguagem de programação Java (Java SE 7) e a API (*Application Programming Interface*) Java 3D, que tem o propósito de expandir os recursos da biblioteca padrão do Java para criação de aplicações que utilizam recursos gráficos tridimensionais.

3.1 Java

O Java foi apresentado pela *Sun Microsystems* em maio de 1995 (Deitel e Deitel, 2005). Ele foi criado com o objetivo de ser utilizado no desenvolvimento de aplicativos embarcados, mas acabou ganhando maior destaque em outras áreas, como o desenvolvimento de aplicações Desktop e aplicações Web (Varejão, 2004). Atualmente o Java é umas das linguagens mais utilizadas no desenvolvimento de software (TIOBE Software, 2014), como apresentado no gráfico da Figura 3.1.

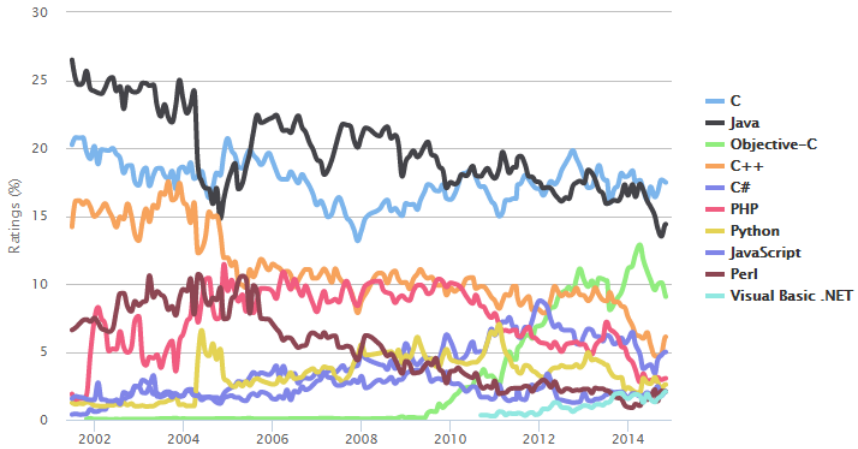


Figura 3.1 Ranking de popularidade das linguagens de programação (Fonte: www.tiobe.com).

A linguagem de programação Java possui uma especificação técnica que padroniza a sintaxe e a semântica. Além disso, disponibiliza uma API base que determina as classes e interfaces para o desenvolvimento de sistemas em Java. A especificação técnica e a API são responsáveis por definir a linguagem de programação Java (Liang, 2009).

Java é uma linguagem de alto nível que é caracterizada por ser simples, orientada a objetos, dinâmica e multiplataforma. Ela se tornou uma linguagem bem difundida pelo conceito de portabilidade, ou seja, o código Java compilado em uma arquitetura pode ser executado em qualquer outra arquitetura.

Essa portabilidade é possível porque o compilador Java converte o código-fonte em *bytecodes*, que são executados pela Máquina Virtual Java (JVM, do inglês *Java Virtual Machine*). Os *bytecodes* são instruções independentes de plataforma, podendo ser executados em qualquer plataforma que contenha a JVM (Deitel e Deitel, 2005). A JVM é específica para cada plataforma, pois ela é a responsável por converter os *bytecodes* em código de máquina específico para a arquitetura no momento que o programa escrito em Java é executado. Diferente de linguagens de programação como a linguagem C e C++, onde código fonte é compilado para código de máquina específico de uma plataforma em que o programa será executado. A Figura 3.2 ilustra esse processo de transformação do código-fonte escrito em Java para código de máquina.

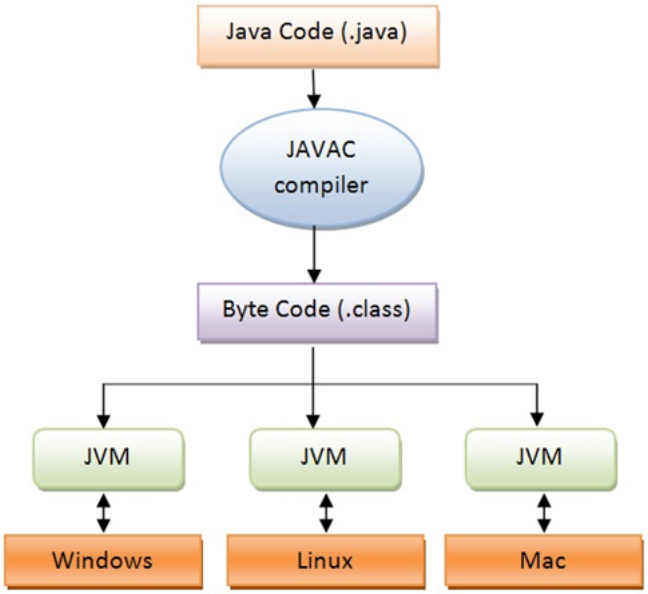


Figura 3.2 Esquema de Funcionamento da JVM (Adaptado de Romanato (2013)).

Códigos escritos em um determinado sistema operacional podem ser executados em um outro. Sendo assim, não é necessário a preocupação em qual sistema operacional a aplicação irá executar, tipo de máquina, e configurações. A aplicação roda sem o envolvimento com o sistema operacional, depende apenas da comunicação com a máquina virtual.

A popularidade do Java proporcionou o surgimento de conjuntos de códigos responsáveis por prover uma solução para uma família de problemas, auxiliando o desenvolvimento de sistemas nessa linguagem, que são as APIs e *frameworks*.

Neste trabalho, para o desenvolvimento e apresentação de imagens tridimensionais de florestas, foi utilizada a API Java 3D, que fornece uma hierarquia de classes para o desenvolvimento de sistemas gráficos tridimensionais.

3.2 Java 3D

A API Java 3D possibilita a criação de aplicações gráficas tridimensionais. Ela oferece recursos de alto nível para criação e manipulação de geometrias 3D. Com ela é possível renderizar ambientes virtuais de grandes escalas de maneira eficiente (Oracle, 2014). As cenas são representadas através de grafos, e os detalhes de sua visualização são gerenciados automaticamente (Manssour, 2003).

Ela é uma API orientadada a objetos e permite ao desenvolvedor programar de maneira mais clara e intuitiva através do seu conjunto de classes e métodos para a programação 3D, não exigindo conhecimento sobre detalhes de implementação de hardware gráfico ou de algoritmos de computação gráfica.

O conjunto de classes oferecidas pelo Java 3D possibilita, além de construir uma cena 3D a partir de um programa, que esta seja carregada de um arquivo externo. Este conjunto de propriedades dá ao Java 3D grande flexibilidade, fazendo dela uma plataforma viável para diferentes aplicações gráficas.

As suas classes estão organizadas em pacotes. Dentre eles, pode-se destacar:

- javax.media.j3d: contém as classes que formam o núcleo da API Java 3D, como as classes para definição de polígonos, texturas e transformações.
- com.sun.j3d.utils: contém vários subpacotes que agrupam classes utilitárias para auxiliar

o desenvolvimento de aplicações com a API Java 3D. Por exemplo, geometrias básicas pré-definidas (com.sun.j3d.utils.geometry) e interação com o usuário (com.sun.j3d.utils.behavior).

Os elementos gráficos construídos pela aplicação são objetos conectados entre si através de uma estrutura do tipo árvore denominada grafo de cena.

3.2.1 Grafo de cena

O modelo de programação baseado em grafo de cena fornece um mecanismo simples e flexível para representação e renderização de cenas. O grafo de cena contém a descrição completa para renderização do universo virtual, que inclui os dados geométricos, atributos e a informação necessária para visualizar a cena a partir de um ponto de vista específico.

Dessa forma, para criar uma aplicação com o Java 3D é necessário construir o grafo de cena. A Figura 3.3 ilustra um grafo de cena básico de uma aplicação baseada no Java 3D.

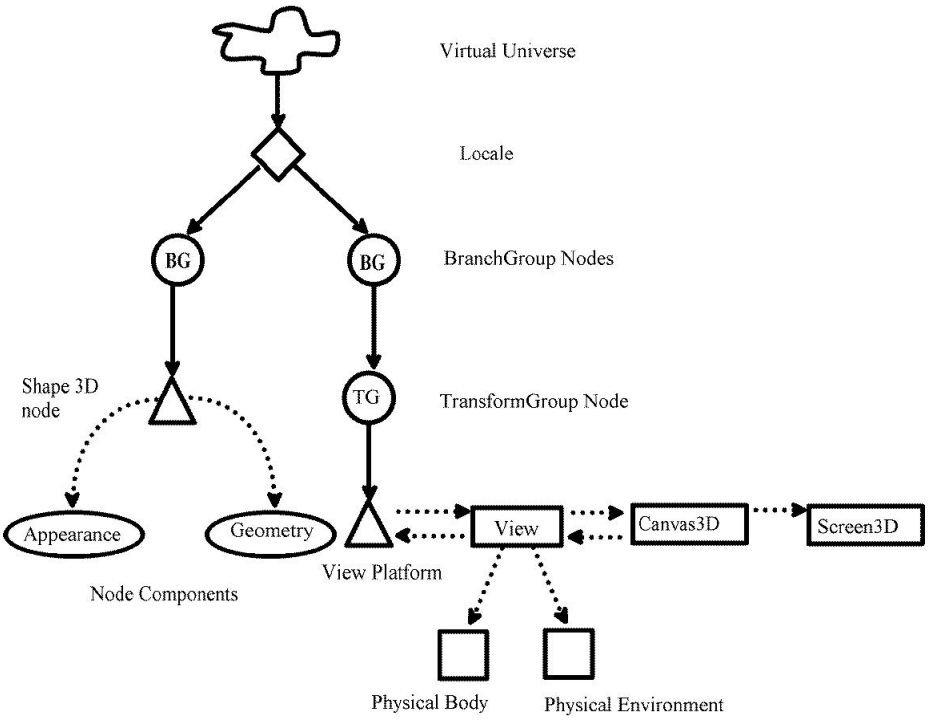


Figura 3.3 Um grafo de cena simples da API Java 3D (Bicho *et al.*, 2002).

O primeiro passo na codificação de uma aplicação Java 3D é definir um universo virtual, que representa o espaço tridimensional onde podem ser inseridos diversos elementos, como

objetos e suas geometrias, luzes, comportamentos, sons, entre outros. O universo virtual é responsável pela configuração de um ambiente mínimo para executar um programa Java 3D.

O Java 3D organiza o universo virtual usando o conceito de agrupamento, isto é, um nó mantém uma combinação de outros nós de modo a formar um componente único (Bicho *et al.*, 2002). Os arcos na Figura 3.3 representa o relacionamento entre os nós do grafo de cena. Os relacionamentos podem ser: (i) por referência (arcos pontilhados), que associa um objeto com o grafo de cena; e (ii) por herança (arcos contínuos), onde um nó do tipo grupo pode ter um ou mais filhos, mas apenas um pai. Um nó do tipo folha não pode ter filhos.

No grafo de cena da Figura 3.3, os nós do tipo grupo são identificados por círculos, os nós do tipo folha por triângulos e os demais objetos por retângulos. Cada nó possui um conjunto de atributos que podem, ou não, influenciar seus nós conectados.

O nó raiz é o primeiro nó do grafo e todos os outros nós estão ligados a ele direta ou indiretamente. O nós internos possuem várias propriedades, sendo o uso mais comum o de representar transformações 3D (rotação, translação e escala). Os nós folha contêm, geralmente, a representação geométrica de um objeto.

Os grafos de cena diminuem o trabalho de se desenvolver aplicações gráficas de alto desempenho, pois ele é o responsável por gerenciar toda a parte gráfica, encapsulando todas as tarefas de baixo nível necessárias para renderizar a cena. Sendo assim, não é necessário que o desenvolvedor se preocupe com a configuração em que irá rodar a aplicação, focando apenas no desenvolvimento.

O universo virtual, representado pela classe *VirtualUniverse*, é o objeto que representa o nó raiz do grafo de cena. Este objeto define um universo e possui pelo menos um objeto *Locale*, que é responsável por fornecer um sistema de coordenadas e pela especificação do ponto de referência no universo virtual, servindo como raiz dos sub-grafos de um grafo de cena (Manssour, 2003).

Para o universo virtual da Figura 3.4, seria possível criar dois objetos da classe *Locale*, um representando a posição da Terra ou um local próximo e outro representando a posição do planeta ou um local próximo em uma galáxia distante.

Cada *Locale* possui um número de nós filhos do tipo *BranchGroup*, que é o nó raiz de cada ramificação de um grafo de cena (Bouvier, 1999).

O grafo de cena ilustrado na Figura 3.3 possui dois sub-grafos cujas raízes são objetos do tipo *BranchGroup*. O objetivo destes objetos é agrupar nós relacionados através de alguma

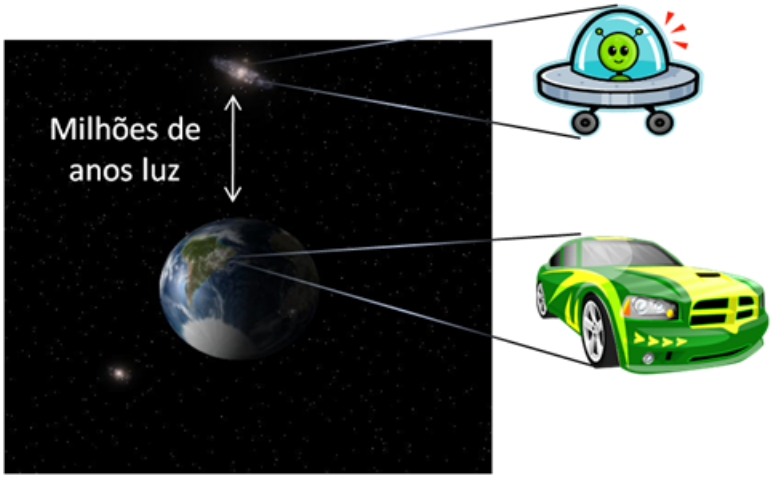


Figura 3.4 Universo virtual com um carro na Terra e uma nave espacial em um planeta de uma galáxia distante. (Fonte: Java Magazine 83, 2010)

associação comum ou através de um conjunto de características.

Existem duas categorias diferentes de sub-grafos que geralmente são incluídos em um grafo de cena, incluídos no *Locale*. A primeira, representada pela sub-árvore da esquerda no grafo de cena da Figura 3.3, descreve o conteúdo do universo virtual (*content branch*), tais como geometrias, aparências, comportamentos, localizações, sons e luzes. A segunda, representada pela subárvore da direita, especifica os parâmetros de controle da visualização da cena (*view branch*), tal como direção de observação.

O *view branch* possui um *TransformGroup* que é usado para especificar a posição, orientação e escala dos objetos geométricos no universo virtual (Manssour, 2003), que por sua vez, possui um nó *ViewPlatform* responsável pela visão final do usuário dentro do universo. Os objetos que compõem este nó são: *View* (com as ramificações *PhysicalBody* e *PhysicalEnvironment*), *Canvas3D* e *Screen3D*.

O objeto *View* armazena todas as informações necessárias para renderizar o grafo de cena, além de coordenar todo o processo de renderização. Ele está associado às classes: *PhysicalBody*, que possui informações sobre características físicas do usuário, como a localização dos olhos e a distância entre as pupilas; e *PhysicalEnvironment*, que gerencia informações sobre o local do ambiente físico do usuário, dispositivos de áudio e sensores de movimento.

O objeto *Canvas3D* garante a apresentação gráfica da cena enquadrada em uma janela de visualização, pois agrupa todos os parâmetros relacionados a essa janela e, finalmente, *Screen3D* possui as propriedades físicas da tela (Bouvier, 1999).

A API Java 3D fornece um grande número de classes para especificação, posicionamento e visualização de objetos gráficos, porém, uma das classes mais importantes é a *SimpleUniverse*, responsável pela configuração de um ambiente mínimo para executar um programa Java 3D. Ela fornece as funcionalidades necessárias para a maioria das aplicações. Quando uma instância de *SimpleUniverse* é criada, automaticamente são criados todos os objetos necessários para o *view branch*, tais como *Locale*, *ViewPlatform* e *View*.

No *content branch* os nós folha são objetos do tipo *Shape3D*. O *Shape3D* refere-se a dois objetos: *Geometry*, que fornece a forma geométrica do objeto e *Appearance*, que descreve a aparência do objeto, ou seja, sua cor, textura, características de reflexão, entre outras.

A aparência de uma forma geométrica (*Appearance*), também pode ser definida por uma imagem. Para isso é preciso definir o posicionamento da textura na geometria.

As transformações geométricas, por exemplo, rotação e translação, são especificadas através de uma instância de *Transform3D*. Objetos da classe *TransformGroup* especificam uma transformação através de um objeto *Transform3D*, que é aplicado a todos os seus filhos. As transformações aplicadas aos objetos são acumulativas.

Além da textura dos objetos, para gerar imagens com realismo, é necessário o uso de algumas técnicas que permitem reproduzir no computador a aparência dos objetos do mundo real. Entre essas técnicas encontram-se os efeitos de iluminação.

No Java 3D a classe abstrata Light define um conjunto de parâmetros comum para todos os tipos de fonte de luz. Estes parâmetros são usados para definir a cor da luz e a sua região de influência. As suas subclasses são *AmbientLight*, *DirectionalLight* e *PointLight* (Manssour, 2003), que definem diferentes formas de iluminação (Figura 3.5).

Dessa forma, escrever um programa em Java 3D requer basicamente a seguinte sequência de passos (Bouvier, 1999):

1. Criar um objeto *Canvas3D*;
2. Criar um objeto *VirtualUniverse*;
3. Criar um objeto *Locale* e anexá-lo ao *VirtualUniverse*;
4. Construir um grafo *view branch*:
   (a) Criar um objeto *View*;
   (b) Criar um objeto *ViewPlatform*;

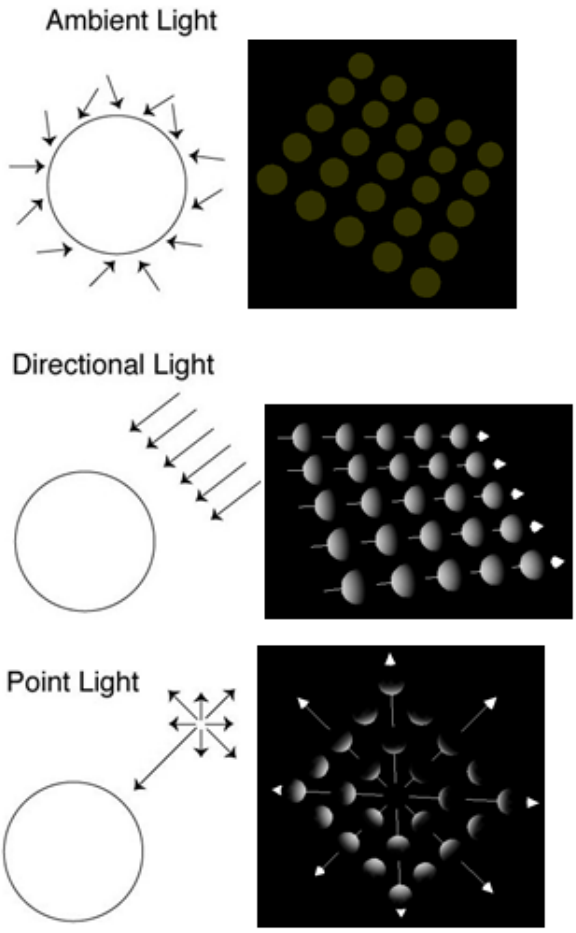


Figura 3.5 Formas de iluminação

(c) Criar um objeto *PhysicalBody*;

(d) Criar um objeto *PhysicalEnvironment*;

(e) Anexar os objetos criados em (b), (c) e (d) ao objeto *View*.

5. Construir um ou mais grafos *content branch*;

6. Compilar o(s) *branch graph(s)*;

7. Inserir os sub-grafos ao nó *Locale*.

3.2.2 Sistemas de coordenadas

Um objeto do tipo *Locale* permite a representação precisa de coordenadas em um universo virtual tão grande quanto o universo real. As posições dos objetos são representadas em relação aos seus respectivos locais através de números reais. Após a definição dos objetos *Locale* de um universo virtual, deve-se criar e posicionar os diferentes elementos que o compõem.

O sistema de coordenadas utilizado no Java 3D para o posicionamento de elementos no universo virtual é baseado no sistema de coordenadas de mão direita, que determina um sistema cartesiano tridimensional conforme o representado na Figura 3.6.

(a) Sistema cartesiano tridimensional (b) Regra da mão direita

Figura 3.6 Sistema de coordenadas de mão direita utilizado no Java 3D (Adaptado de Luque e Silva (2010))

O nome do sistema deve-se ao fato de que, com a mão direita, pode-se definir tanto a posição dos eixos quanto a forma como se dá a rotação em torno dos eixos. Das diversas formas de definição da posição dos eixos, talvez a mais fácil de visualizar seja através da regra da mão direita.

Quando esta mão é colocada da forma apresentada na Figura 3.6b, o polegar aponta para o eixo X, o indicador para o eixo Y e o dedo médio para o eixo Z, sendo que os dedos apontam para o sentido positivo dos eixos (Luque e Silva, 2010).

A relação entre esses eixos e um dispositivo de saída pode ser observada na Figura 3.7, com o eixo X apontando para a direita, o eixo Y para cima e o eixo Z para fora do dispositivo.

A posição de um ponto no espaço tridimensional pode ser representada, portanto, através de três coordenadas (x, y, z), sendo que a interseção entre os três eixos é a origem (0, 0, 0). Uma unidade de coordenada corresponde, por padrão, a um metro. A Figura 3.8 representa o sentido de rotação em cada um dos eixos.

Figura 3.7 Relação entre os eixos cartesianos tridimensionais e um dispositivo de saída (Adaptado de Luque e Silva (2010)).

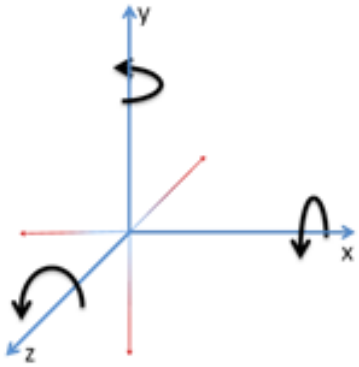


Figura 3.8 Sentido de rotação dos eixos (Adaptado de Luque e Silva (2010)).

CAPÍTULO 4

Sistema Desenvolvido

Este capítulo apresenta o sistema desenvolvido, que tem como intuito possibilitar a visualização e simulação de florestas tridimensionais. As seções seguintes apresentam os aspectos relevantes para a construção do sistema, o público que o mesmo irá atender e as suas principais funcionalidades.

4.1 Sistema para visualização e simulação de florestas

Uma floresta em condições tropicais varia entre 7 a 30 anos para concluir um ciclo econômico de crescimento (Santana, 2013). Ao longo desse período, as atividades de manejo florestal devem ser planejadas. Assim, prever o estado da floresta em condições futuras é importante para a tomada de decisões. Nesse contexto, tecnologias para simulações de condições complexas são importantes.

O manejo florestal compreende a condução sustentável da floresta para produção de produtos florestais garantindo a qualidade e a quantidade de forma economicamente viáveis. Para acompanhar a evolução de uma floresta é necessário criar mecanismos para avaliação da produtividade. Ferramentas de visualização também podem ser utilizadas para auxiliar a tomada de decisões de gerenciamento.

O sistema desenvolvido tem como principal recurso a visualização de florestas tridimensionais. Além disso, ele possibilita a simulação do crescimento e a estimativa de alguns parâmetros da floresta. Esta ferramenta poderá ser utilizada por profissionais do setor florestal e discentes de graduação e pós-graduação no estudo sobre crescimento e produção em diferentes cenários.

As funcionalidades do sistema são apresentadas no diagrama de casos de uso na Figura 4.1. O sistema permitirá que o usuário carregue os dados da floresta a partir de uma planilha eletrônica, simule o crescimento da floresta, visualize estimativas de volume e biomassa, salve

dados simulados em arquivo, gere relatórios em PDF com as informações da floresta, configure dados de visualização e simulação. Além disso, o sistema permite ao usuário interagir com o ambiente, visualizando a floresta por diversos ângulos.

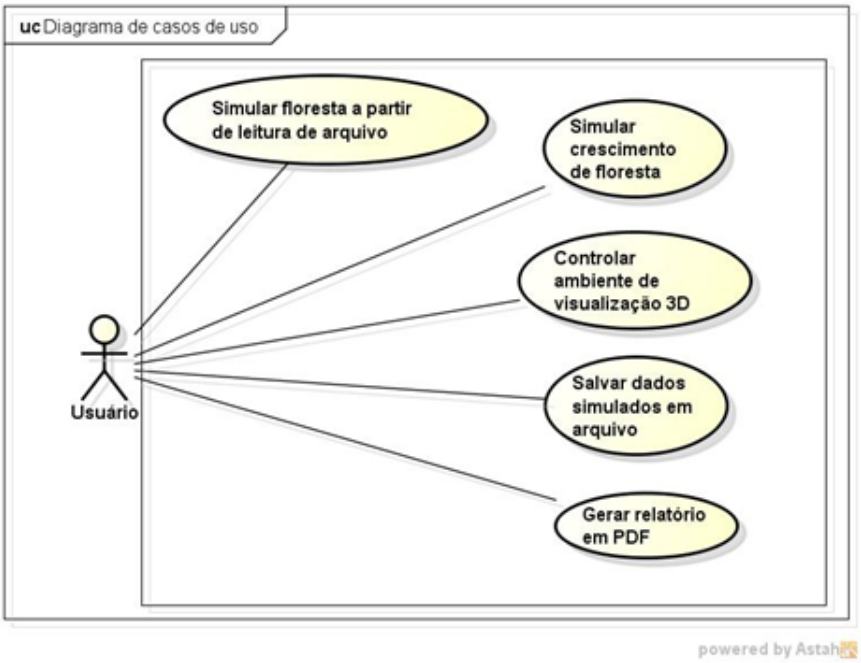


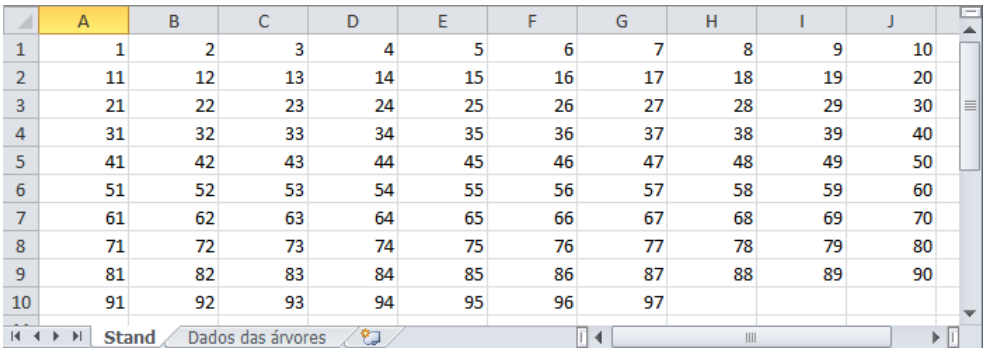
Figura 4.1 Diagrama de casos de uso das funcionalidades implementadas.

As subseções seguintes apresentam as características e funcionalidades do sistema, além de detalhes de sua implementação que sejam considerados relevantes.

4.2 Leitura de dados

Para possibilitar a visualização tridimensional da floresta, é necessário que o usuário organize os dados das árvores em uma planilha no formato XLS (planilha do *Microsoft Office Excel*), a qual deve possuir duas folhas.

A primeira folha da planilha deve conter a forma como as árvores estão dispostas em um *grid*, a ordem em que elas serão posicionadas no processo de renderização. Cada posição nesse *grid* é preenchida com o identificador único da árvore. A Figura 4.2 apresenta um exemplo desta primeira folha.

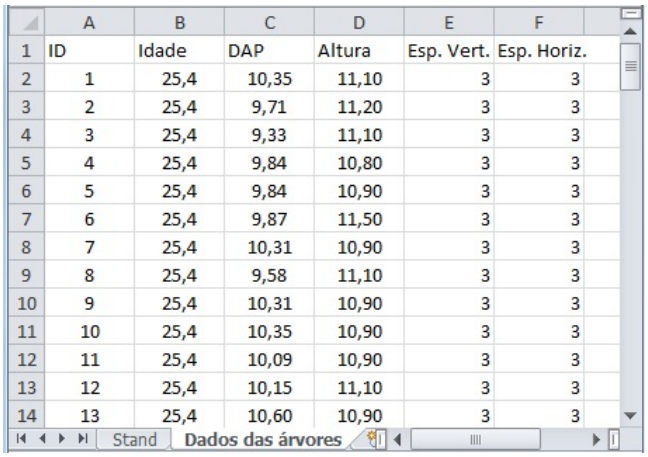


	A	B	C	D	E	F	G	H	I	J
1	1	2	3	4	5	6	7	8	9	10
2	11	12	13	14	15	16	17	18	19	20
3	21	22	23	24	25	26	27	28	29	30
4	31	32	33	34	35	36	37	38	39	40
5	41	42	43	44	45	46	47	48	49	50
6	51	52	53	54	55	56	57	58	59	60
7	61	62	63	64	65	66	67	68	69	70
8	71	72	73	74	75	76	77	78	79	80
9	81	82	83	84	85	86	87	88	89	90
10	91	92	93	94	95	96	97			

Stand | Dados das árvores

Figura 4.2 Exemplo da primeira folha da planilha.

A segunda folha da planilha deverá conter informações das árvores, onde cada linha contém os dados de uma única árvore, exceto a primeira linha, que é o cabeçalho indicando o conteúdo de cada coluna. As colunas devem ser organizadas na seguinte ordem: identificado único (ID), idade, diâmetro na altura do peito (DAP), altura, espaçamento vertical e espaçamento horizontal. A Figura 4.3 apresenta um exemplo da segunda folha.



	A	B	C	D	E	F
1	ID	Idade	DAP	Altura	Esp. Vert.	Esp. Horiz.
2	1	25,4	10,35	11,10	3	3
3	2	25,4	9,71	11,20	3	3
4	3	25,4	9,33	11,10	3	3
5	4	25,4	9,84	10,80	3	3
6	5	25,4	9,84	10,90	3	3
7	6	25,4	9,87	11,50	3	3
8	7	25,4	10,31	10,90	3	3
9	8	25,4	9,58	11,10	3	3
10	9	25,4	10,31	10,90	3	3
11	10	25,4	10,35	10,90	3	3
12	11	25,4	10,09	10,90	3	3
13	12	25,4	10,15	11,10	3	3
14	13	25,4	10,60	10,90	3	3

Stand | Dados das árvores

Figura 4.3 Exemplo da segunda folha da planilha.

A partir desses dados o sistema é capaz de criar o modelo tridimensional da floresta e estimar alguns parâmetros.

4.3 Visualização de floresta 3D

A maior parte da janela da aplicação é dedicada à visualização da floresta, onde cada árvore é renderizada de acordo com as informações lidas a partir do arquivo e posicionadas da forma como é definida no *grid*. A Figura 4.4 apresenta uma visão geral da janela principal do sistema.

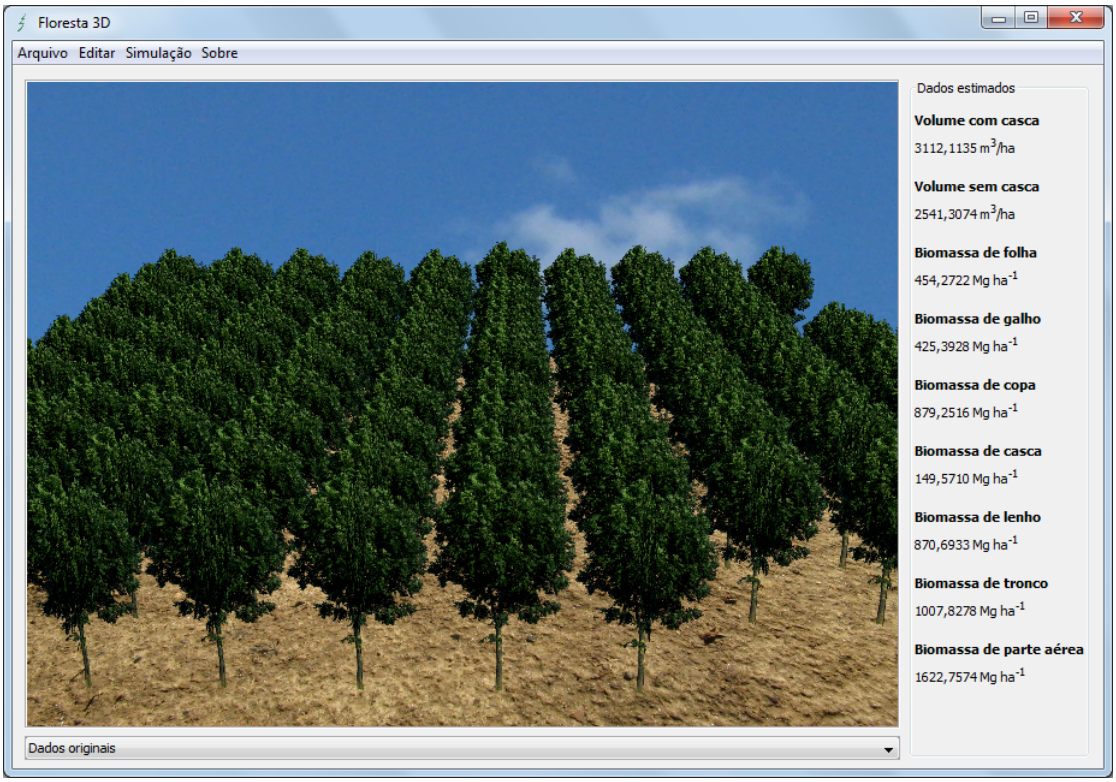


Figura 4.4 Janela principal do sistema.

As árvores são posicionadas sobre um terreno plano, apesar de que, em um cenário real esta situação nem sempre é verdadeira. Porém, foi implementado dessa forma para simplificação do desenho gerado. Para desenhar um terreno irregular seria necessário a criação de uma figura complexa, com muitos pontos e faces, o que tornaria a renderização do cenário mais lenta. A visualização por parcela pode ser precisa.

Assim que o modelo tridimensional é gerado, o sistema permite que usuário controle a cena a partir do mouse ou teclado. Dessa forma, a floresta poderá ser observada de todos os ângulos. Os controles de cena oferecidos são a translação e rotação.

Ainda na janela principal, alguns parâmetros da floresta podem ser visualizados. Estes valores são o somatório do resultado obtido de cada árvore.

4.3.1 Modelos 3D das árvores

As árvores podem ser desenhadas de várias formas, desde um nível de detalhes mais básico a modelos mais elaborados. A seguir, os modelos de renderização de árvore implementados neste trabalho são descritos.

4.3.1.1 Modelo básico

Neste modelo, a árvore é constituída por duas figuras geométricas: um cilindro que representa o tronco da árvore e um cone que representa a copa. Em cada uma dessas figuras é utilizada uma textura para dar a aparência de tronco e folhas. Para obter uma aparência mais real a textura utilizada no cilindro é criada a partir de uma imagem de um tronco real, porém, para a copa uma aparência mais real não é possível devido a forma limitada do cone, que é apenas preenchido com a cor verde. A Figura 4.5 apresenta uma árvore criada por este modelo sob diferentes ângulos de visões.

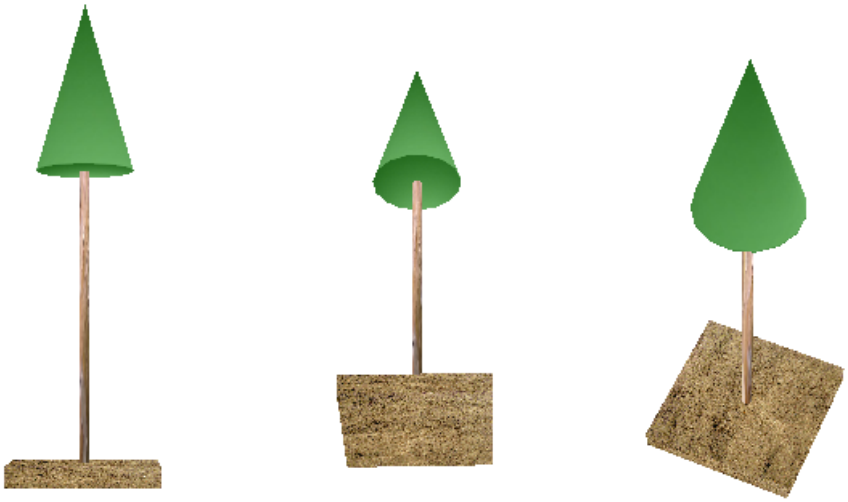


Figura 4.5 Árvore do Modelo básico.

A vantagem do modelo básico é a sua simplicidade, sendo necessário poucos recursos

computacionais (memória RAM, processamento gráfico) para renderizar florestas. Com isso é possível gerar florestas grandes de maneira rápida. No entanto, esse modelo deixa a desejar na criação de visualização mais realista. A Figura 4.6 apresenta uma floresta gerada utilizando este modelo.

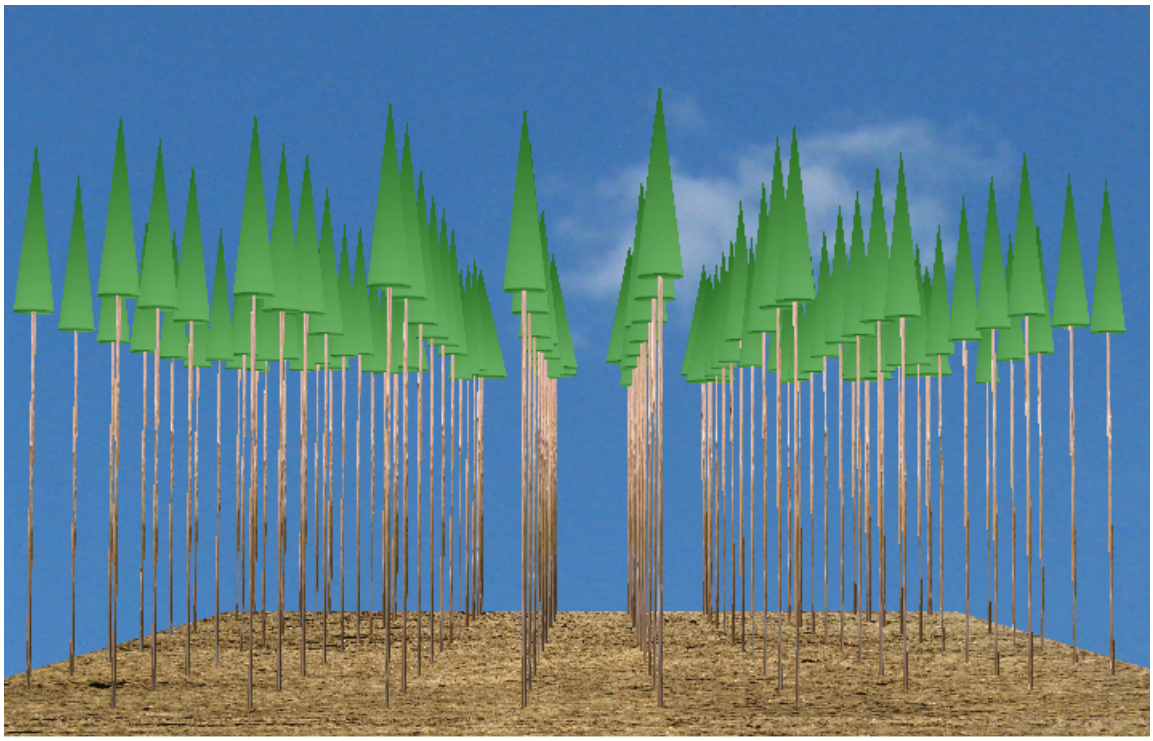


Figura 4.6 Floresta do Modelo básico.

4.3.1.2 Modelo painel 1

No modelo painel 1, as árvores são representadas por painéis cruzados, como apresentado no trabalho de Falcão *et al.* (2006). O uso dessa estratégia permite a criação de grandes florestas utilizando poucos recursos tridimensionais e um bom nível de realismo. A Figura 4.7 apresenta uma árvore gerada por este modelo. Como pode ser observado, este modelo apresenta um efeito indesejado quando as árvores são observadas por cima, já que os painéis são cruzados apenas na vertical. A Figura 4.8 apresenta uma floresta completa com o uso desse modelo.

Apesar do uso de painéis cruzados permitir cenários realistas, criar ou encontrar figuras que se adaptem bem para esse modelo não é tão simples, mesmo que seja possível a edição

Figura 4.7 Árvore Modelo painel 1.

Figura 4.8 Floresta Modelo painel 1.

de uma imagem comum para a criação de uma imagem com transparência, deixando visível apenas a parte desejada. Isso é devido a copa da árvore que faz com esta tarefa seja trabalhosa.

Além da dificuldade da transparência, se a espécie tiver um tronco com curva, o painel

cruzado não gera um bom resultado com apresentado na Figura 4.9.

Figura 4.9 Tronco com curva.

4.3.1.3 Modelo painel 2

O modelo painel 2 também utiliza painéis cruzados, como no modelo anterior, porém, o tronco e a copa são criados separadamente. Neste modelo, o tronco é criado a partir de uma geometria plana com as bordas fazendo a curvatura do tronco. O afilamento do tronco é calculada a partir do modelo de afilamento ou *taper* (Garay, 1979):

$$\frac{d}{dap} = \beta_1 \left\{ 1 + \beta_2 Ln \left[1 - \beta_3 \left(\frac{h}{H} \right)^{\beta_4} \right] \right\} + \varepsilon \tag{4.1}$$

onde,

- h: altura em um ponto qualquer do fuste, em metros;
- H: altura total, em metros;
- d: diâmetro na altura h, em metros;
- dap: diâmetro a 1,30 metros de altura, em centímetros;

- β_1, β_2, β_3, β_4 são parâmetros a serem estimados por regressão;
- ε: erro aleatório.

A Figura 4.10 apresenta o exemplo da geometria gerada. Quanto maior o número de pontos utilizados na criação da geometria do tronco, melhor será a qualidade da curvatura. Após a criação, a textura de tronco é aplicada sobre essa geometria. Com isso, duas cópias dessa geometria são cruzadas, como é feito no modelo painel 1.

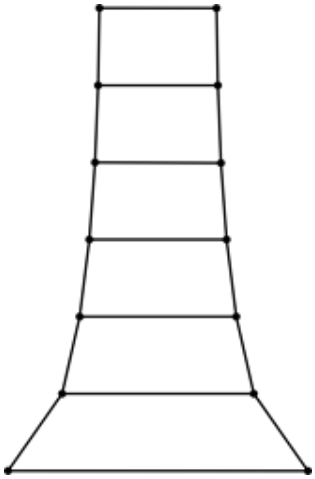


Figura 4.10 Geometria gerada para formação do tronco no Modelo painel 2.

A criação da copa é feita de maneira similar ao modelo painel 1, utilizando imagens com transparência. Porém a imagem contém apenas a copa da árvore. Devido a isso, o efeito indesejado do modelo painel 1 ao visualizar a imagem por cima pode ser contornado. Para isso, além dos dois painéis cruzados verticalmente, é utilizado também um painel na horizontal. A Figura 4.11 apresenta o desenho de uma árvore gerada com esse modelo vista por diferentes ângulos. Na Figura 4.12 é visualizada uma floresta inteira gerada com este modelo.

4.3.1.4 Modelo híbrido 1

No modelo híbrido 1, o tronco das árvores é representado por um cilindro, como no modelo básico, e a copa representada da mesma forma que o modelo painel 2. Este modelo é simples como o modelo básico. Porém, possibilita uma visualização mais realista pelo fato da copa ser feita por painéis cruzados.

Figura 4.11 Árvore do Modelo painel 2.

O uso de cilindro no desenho do tronco, comparando se com o tronco do modelo painel 2, evita alguns efeitos indesejados quando visualizado por alguns ângulos específicos, como apresentado na Figura 4.13. No entanto, perde-se o afilamento do tronco.

A Figura 4.14 apresenta uma árvore gerada por este modelo e vista por diferentes ângulos. Já na Figura 4.15 pode ser observada uma floresta completa.

4.3.1.5 Modelo híbrido 2

O modelo híbrido 2 é o modelo de árvore mais elaborado implementado no sistema. Ele apresenta visualizações mais realistas, porém é também o mais complexo e mais pesado. Assim como nos modelos painel 2 e híbrido 1, a copa é feita por painéis cruzados. Pois essa é a estratégia mais realista para o desenho da copa. O diferencial desse modelo está no formato

Figura 4.12 Floresta Modelo painel 2.

Figura 4.13 Efeito indesejado do tronco modelo painel 2.

cilíndrico do tronco com afilamento, ou seja, ele traz as vantagens do modelo hibrido 1 e painel 2 sem os efeitos indesejados.

Na Figura 4.16 pode ser visualizada uma árvore gerada por esse modelo a partir de diferentes ângulos. Como pode ser observado ela não apresenta nenhum efeito indesejado dos modelos anteriores. Na Figura 4.17 poder ser vista uma floresta completa gerada com o modelo híbrido 2.

Figura 4.14 Árvore do Modelo híbrido 1.

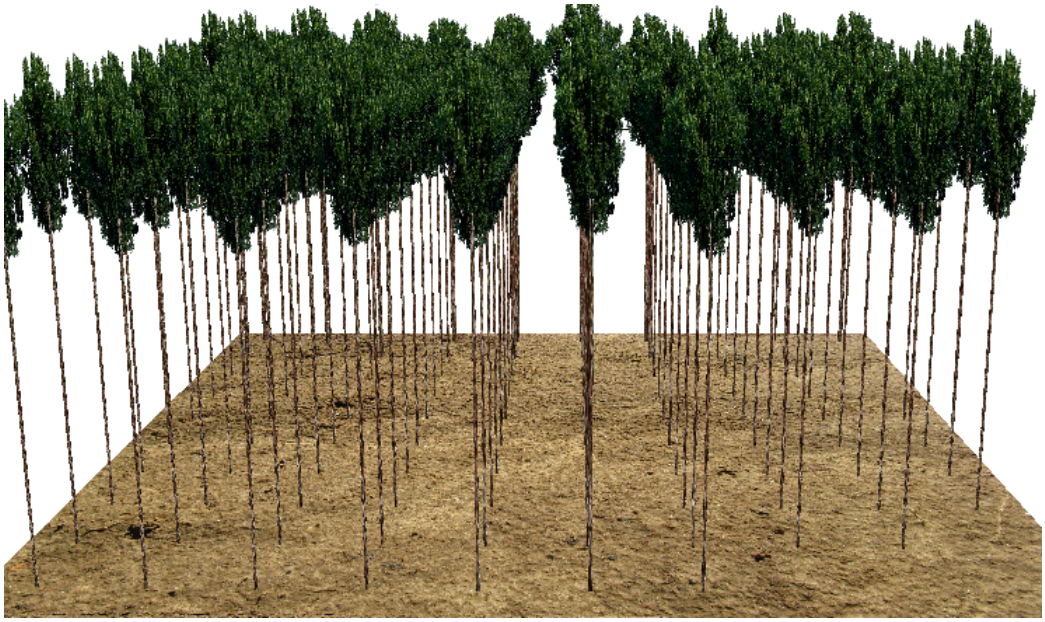


Figura 4.15 Floresta Modelo híbrido 1.

Apesar desse modelo gerar a figura mais realista, ele implica em vários cálculos para a geração da geometria do tronco, o que pode tornar o desenho de florestas muito grandes uma tarefa lenta. A Figura 4.18 apresenta a forma da geometria gerada para o tronco deste modelo.

Figura 4.16 Árvore do Modelo híbrido 2.

Quanto maior o número de pontos utilizados na vertical (n_v), melhor é o efeito do afilamento do tronco, e quanto maior o número de pontos na circunferência (n_c) para cada nível de altura, melhor é o efeito cilíndrico. No exemplo apresentado na Figura 4.18, n_v é igual a 7 e n_c é igual a 8.

O número total de pontos que devem ser calculados para a criação de um único tronco é dado por $n_v \times n_c$. Quanto maior o nível de detalhes desejados, maior o número de pontos a serem calculados.

O cálculo da curvatura do tronco é feita da mesma forma que no modelo painel 2. O Algoritmo 4.1 abaixo apresenta a lógica para o cálculo dos pontos dessa geometria.

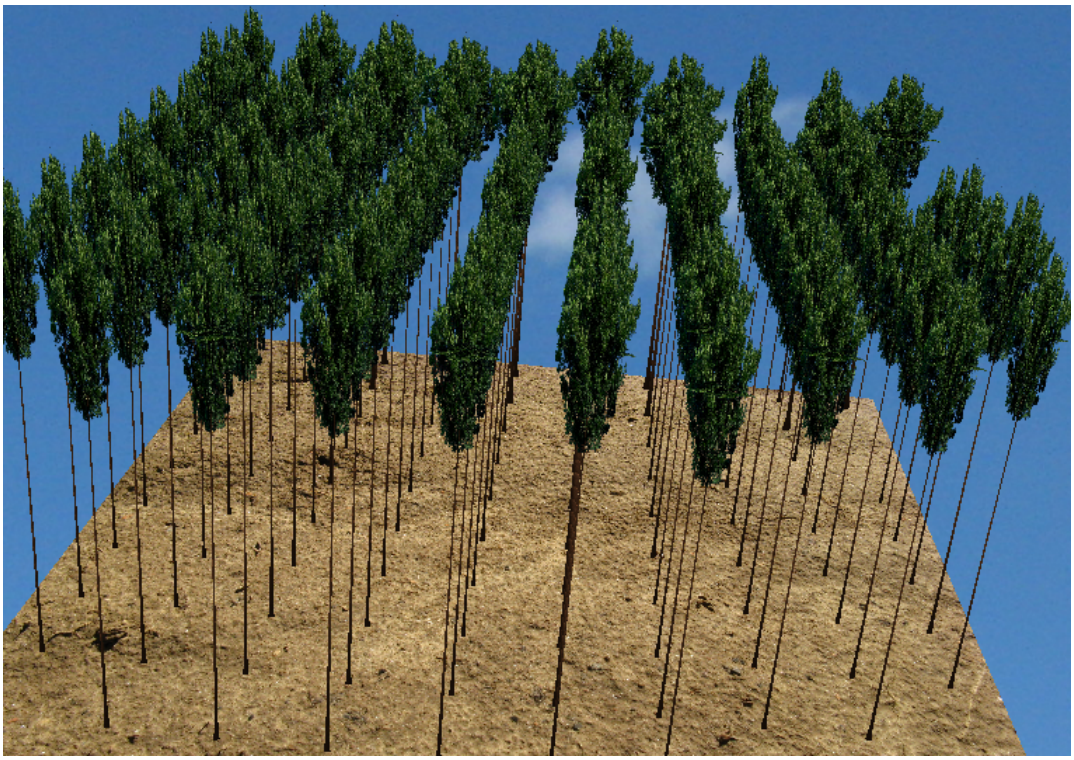


Figura 4.17 Floresta Modelo híbrido 2.

$\Delta h \leftarrow arvore.altura/(n_v - 1)$;
$\Delta c \leftarrow 360/n_c$;
para $h = 0$ **até** $n_v - 1$ **faça**
  $y \leftarrow \Delta h \times h$;
  $diametro \leftarrow calcularDiametro(y, arvore.altura, arvore.dap)$;
  para $c = 0$ **até** n_c **faça**
    $\alpha \leftarrow \Delta c \times c$;
    $x \leftarrow cos(\alpha) \times (diametro/2)$;
    $z \leftarrow sen(\alpha) \times (diametro/2)$;
  fim
fim

Algoritmo 4.1: Lógica para o cálculo dos pontos da geometria do modelo híbrido 2.

4.4 Simulação de crescimento

Além da visualização de florestas a partir da leitura de dados de arquivo, o sistema possibilita simular o crescimento dessas floresta com base na idade. Ao renderizar uma floresta, o usuário pode visualizar o crescimento futuro da mesma, possibilitando a comparação visual

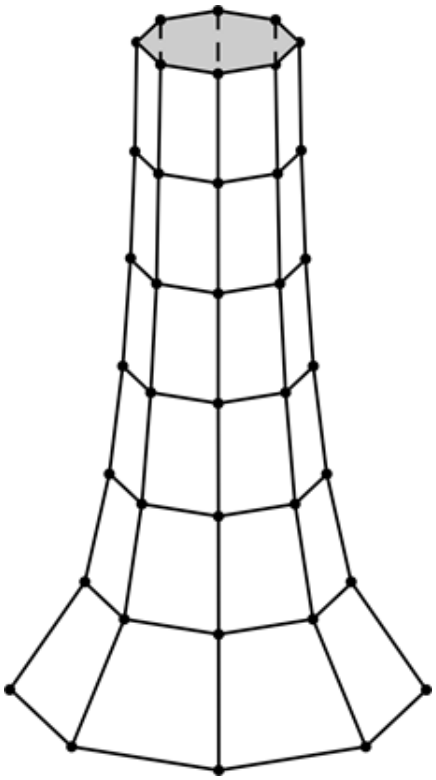


Figura 4.18 Geometria gerada para formação do tronco no Modelo híbrido 2.

entre florestas, e não apenas dados numéricos.

A simulação do crescimento é realizada a partir da estimativa da altura e do DAP da árvore em uma idade desejada. As funções que estimam a altura e o DAP das árvores são apresentadas nas equações (4.2) e (4.3) (Pienaar e Shiver, 1981), respectivamente:

$$Ht_2 = Ht_1 e^{-\theta_1 (I_2^{\theta_2} - I_1^{\theta_2})} + \varepsilon \tag{4.2}$$

$$dap_2 = dap_1 e^{-\theta_1 (I_2^{\theta_2} - I_1^{\theta_2})} + \varepsilon \tag{4.3}$$

onde,

- Ht_1: é a altura atual da árvore em metros;
- Ht_2: é altura estimada da árvore em metros;
- dap_1: é diâmetro na altura do peito atual;
- dap_2: é diâmetro na altura do peito estimado;

- I_1: é a idade atual da árvore em meses;
- I_2: é a idade para qual se deseja estimar a nova altura Ht_2;
- θ_1 e θ_1 são parâmetros que devem ser definidos de acordo com observações reais de forma a melhor ajustar a função ao crescimento real. Os valores destes parâmetros são diferentes para o cálculo da altura e DAP.
- ε: é uma variável de erro que descreve variações no valor estimado em relação ao valor observado.

As Figuras 4.19 e 4.20 ilustram uma mesma floresta com idade de 15 meses e 60 meses (após a simulação).

Figura 4.19 Árvores com 15 meses.

4.5 Estimativa de parâmetros da floresta

Tão importante quanto a visualização tridimensional é a estimativa de algumas informações das florestas (Santana, 2009). Sendo assim, o sistema calcula e apresenta as seguintes informações:

- volume com casca
- volume sem casca

Figura 4.20 Árvores com 60 meses.

- biomassa de folha
- biomassa de galho
- biomassa de copa
- biomassa de casca
- biomassa de lenho
- biomassa de tronco
- biomassa de parte aérea.

Tais informações permitem um melhor conhecimento a respeito da floresta estudada, já que a visualização é apenas um dos recursos utilizados no estudo e na tomada de decisões. Para

o cálculo dessas informações é necessário a configuração de alguns coeficientes para ajuste dos modelos estatísticos, como acontece na estimativa de altura e de DAP. Para isso, o sistema permite que estes coeficientes sejam definidos pelo usuário (Figura 4.21).

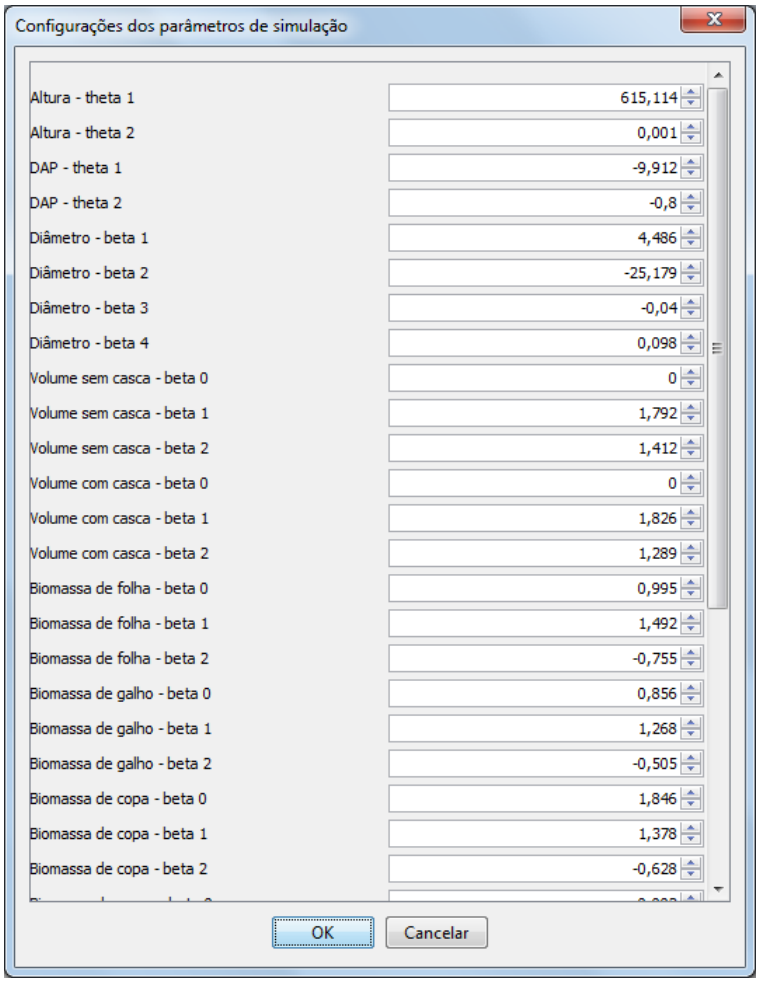


Figura 4.21 Configuração dos coeficientes dos modelos.

4.6 Geração de relatórios

Para facilitar a análise de dados gerados, é possível exportar os dados apresentados no sistema para relatórios em formato PDF. Estes relatórios possibilitam a consulta e estudos a respeito das informações geradas após o encerramento do programa.

Este relatório em PDF contém os valores estimados para as variáveis da floresta apre-

sentadas de maneira resumida, como na janela principal (somatório do valor de cada árvore), e os valores destas mesmas variáveis para cada árvore.

Além do relatório em PDF, é possível também salvar em planilha XLS os dados das árvores após a simulação de crescimento. Isso possibilita que estes dados sejam posteriormente carregados no sistema para visualização, e também possam ser utilizados para comparação numérica.

4.7 Detalhes de implementação

Um dos critérios considerados na escolha da linguagem Java para o desenvolvimento do sistema é a possibilidade de utilizar o mesmo código compilado em uma plataforma em outras, já que o código fonte é compilado para um código intermediário denominado bytecode. No entanto, a API Java 3D depende de recursos nativos do sistema, o que de certa forma o torna depende da plataforma. Por exemplo, para a execução no Windows a API Java 3D é executada sobre o DirectX, e para o Linux ela roda sobre o OpenGL. Para isso, a API Java 3D utiliza bibliotecas em código nativo de cada arquitetura. Apesar dessa característica, a linguagem Java é flexível o suficiente para que, em tempo de execução, seja possível identificar a arquitetura que o sistema está sendo executado e carregar as bibliotecas nativas necessárias. Com isso, essa dependência se torna transparente para o usuário final.

O código apresentado a seguir (Figura 4.22) é executado no início do sistema, e é o responsável por identificar a arquitetura e carregar as bibliotecas nativas necessárias.

Uma outra característica relevante é a estratégia para carregar as texturas utilizadas no sistema. Cada textura deve ser carregada uma única vez, e utilizada sempre a mesma instância quando necessária. Para conseguir isso, todas as texturas utilizadas no sistema foram concentradas em uma única classe de forma que, em qualquer ponto do sistema que elas pudessem ser acessadas.

Essa estratégia leva a um ganho de desempenho muito significativo, possibilitando a renderização de um maior número de árvores quando comparada a uma estratégia em que cada vez que é necessária uma nova instância da textura é criada.

Para exemplificar o ganho de desempenho, considere que uma árvore utilize duas instâncias de textura, uma para o tronco e uma para a copa. Se cada uma dessas texturas carregar

```
String dirBibliotecas = null;
if(sistemaOperacional.toLowerCase().contains("windows") && arquitetura.contains("x86")) {
    dirBibliotecas = "lib/j3d/windows-x86/";
}
else if(sistemaOperacional.toLowerCase().contains("windows") && arquitetura.contains("amd64")) {
    dirBibliotecas = "lib/j3d/windows-amd64/";
}
else if(sistemaOperacional.toLowerCase().contains("linux") && arquitetura.contains("x86")) {
    dirBibliotecas = "lib/j3d/linux-x86/";
}
else if(sistemaOperacional.toLowerCase().contains("linux") && arquitetura.contains("amd64")) {
    dirBibliotecas = "lib/j3d/linux-amd64/";
}
else if(sistemaOperacional.toLowerCase().contains("mac")) {
    dirBibliotecas = "lib/j3d/mac/";
}
```

Figura 4.22 Identificação de arquitetura para carregamento das bibliotecas adequadas.

um arquivo de imagem de 32 KB, para desenhar uma floresta com mil árvores será necessário carregar na memória $32 \times 1000 \times 2$ KB, que é igual a 64000 KB, o que equivale a 62,5 MB, se utilizada a estratégia de criar uma instância da textura sempre que ela é necessária. Se cada textura for criada uma única vez e utilizada sempre a mesma instância quando necessária, apenas 64 KB serão carregados na memória.

CAPÍTULO 5

Considerações Finais

O trabalho realizado teve como objetivo o desenvolvimento de um sistema para a visualização de florestas tridimensionais e a simulação do crescimento de árvores, de forma a facilitar a tomada de decisões no manejo florestal.

Para o desenvolvimento do sistema, foi utilizada a linguagem Java e a API Java 3D, que facilita o desenvolvimento de aplicações com recursos tridimensionais, oferecendo diversas funcionalidades com um alto nível de abstração, descartando assim a necessidade de domínio do campo da Computação Gráfica. Além da linguagem de programação Java ter a vantagem da portabilidade, possibilitando que um mesmo sistema seja utilizado em diferentes sistemas operacionais, há também a simplicidade na integração dos ambientes 3D com interface com o usuário.

A ferramenta desenvolvida é capaz de gerar, conforme especificações: (i) a visualização do ambiente florestal tridimensional, a partir de arquivos de dados numéricos; (ii) a simulação do crescimento de florestas, a partir de modelos matemáticos, possibilitando comparações de florestas com diferentes idades; (iii) a geração de relatórios com dados das florestas originais e simuladas; (iv) a total interação do usuário com o ambiente, permitindo a ele navegar pelo cenário gerado e visualizar todos os pontos do espaço renderizado pela aplicação.

O sistema possibilita que o usuário escolha o modelo de renderização de árvore mais adequado baseado nos recursos computacionais que ele tem disponível e um nível de detalhes desejado. Além disso, ele é flexível o suficiente para permitir que florestas, independente da espécie, sejam simuladas. Já que os parâmetros dos modelos matemáticos utilizados na simulação podem ser facilmente configurados.

As funcionalidades oferecidas pelo sistema podem contribuir e facilitar o trabalho de profissionais e estudantes de áreas florestais.

Além dos avanços alcançados com esta aplicação, é importante que trabalhos futuros sejam realizados com intuito de dar continuidade ao desenvolvimento e aprimoramento do software implementado, aumentando funcionalidades que possam trazer contribuição na tomada de

decisões no manejo florestal.

Sugestões de trabalhos futuros são: (i) a implementação do terreno com relevo, que permite modelar a geografia da localidade da floresta; (ii) a possibilidade do usuário carregar as imagens que quer utilizar como textura na renderização da floresta, e assim permitir que a visualização das árvores seja mais semelhantes com espécies reais.

Referências Bibliográficas

Bao *et al.*(**2009**) Guanbo Bao, Xiaopeng Zhang, Wujun Che, e Marc Jaeger. Billboards for tree simplification and real-time forest rendering. Em *Third International Symposium on Plant Growth Modeling, Simulation, Visualization and Applications*, páginas 433–440. Citado na pág. 9

Bicho *et al.*(**2002**) A. L. Bicho, L. G. Silveira, A. J. A. Cruz, e A. B. Raposo. Programação gráfica 3D com OpenGL, Open Inventor e Java 3D. *Revista Eletrônica de Iniciação Científica*, 2(1). Citado na pág. xi, 6, 14, 15

Bouvier(**1999**) Denis J. Bouvier. *Getting started with Java 3D API*. Sun Microsystems, Inc. Citado na pág. 15, 16, 17

Cardoso(**2003**) Alexandre Cardoso. VRML - a web em 3D, 2003. URL `http://www.alexandre.eletrica.ufu.br/cg/livro_VRML_SVR03.pdf`. Citado na pág. 7

Deitel e Deitel(**2005**) H. M. Deitel e P. J. Deitel. *Java: como programar*. Pearson Prentice Hall, 6 edição. Citado na pág. 11, 12

Dunbar(**2003**) Matt D. Dunbar. 3D visualization for the analysis of forest cover change. Em *Proceedings ISPRS Commission IV Joint Workshop: Challenges in Geospatial Analysis, Integration and Visualization II*. Citado na pág. 8, 9

Falcão *et al.*(**2006**) A. O. Falcão, M. P. dos Santos, e J. G. Borges. A real-time visualization tool for forestecosystem management decision support. *Computers and Eletronics in Agriculture*, páginas 3–12. Citado na pág. xi, 8, 9, 26

Fan *et al.*(**2011**) Jing Fan, Xin xin Guan, e Ying Tang. The dynamic simulator of forest evolution. Em *2011 IEEE International Symposium on VR Innovation (ISVRI)*, páginas 225–229. Citado na pág. 10

Garay(1979) Lorenzo Garay. *Tropical forest utilization system, VIII, A taper model for entire stem profile including buttressing*. Seatlhe. Coll. Forest. Resour., Inst. Forest Prod. Univ. Wash. 64p. (contrib. 36). Citado na pág. 28

Gattass(2013) Marcelo Gattass. Notas de aula em computação gráfica, 2013. URL http://webserver2.tecgraf.puc-rio.br/~mgattass/. Citado na pág. 1, 5

House *et al.***(1998)** D. H. House, G. S. Schmidt, S. A. Arvin, e M. Kitagawa-DeLeon. Visualizing a real forest. *IEEE Computer Graphics and Applications*, 18(1):12–15. Citado na pág. 8

Kose *et al.***(2008)** K. Kose, N. Grammalidis, E. Yilmaz, e E. Cetin. 3D forest fire propagation simulation. Em *3DTV Conference: the true vision - capture, transmission and display of 3D video*, páginas 369–372. Citado na pág. 10

Liang(2009) D. Liang. *Introduction to Java Programming*. Pearson Prentice Hall. Citado na pág. 12

Lim e Honjo(2003) En-Mi Lim e Tsuyoshi Honjo. Three-dimensional visualization forest of landscapes by VRML. *Landscape and Urban Planning*, 63:175–186. Citado na pág. 2, 7, 9

Luque e Silva(2010) L. Luque e R. R. Silva. Desenvolvimento 3D em Java, 2010. URL http://www.devmedia.com.br/desenvolvimento-3d-em-java-java-magazine-83/18017. Acessado em 11 de outubro de 2012. Citado na pág. xi, 19, 20

Manssour(2000) I. H. Manssour. Introdução à VRML 2.0, 2000. URL http://www.inf.pucrs.br/manssour/VRML/index.html. Citado na pág. 7

Manssour(2003) I. H. Manssour. Introdução ao Java 3D. *RITA*, 10:71–96. Citado na pág. 13, 15, 16, 17

Manssour(2006) I. H. Manssour. Introdução à computação gráfica. *RITA*, XIII(2). Citado na pág. 5

Martins *et al.***(2009)** L. B. Martins, P. B. Abdelhay, R. I. A. Rezende, R. A. Kaiser, e T. R. Rangel. Computação gráfica - componentes e requisitos. Relatório técnico, Universidade Federal do Rio de Janeiro. Citado na pág. 5

McCormick *et al.***(1987)** Bruce H. McCormick, Thomas A. DeFanti, e Maxine D. Brown. *Visualization in scientific computing*, volume 21 of *Computer Graphics*. ACM Press. Citado na pág. 1

McGaughey(1997) Robert J. McGaughey. Visualizing forest stand dynamics using the stand visualization system. Em *Proceedings of the 1997 ACSM/ASPRS Annual Convention and Exposition*. Citado na pág. 2

Meitner *et al.***(2005)** M. J. Meitner, R. Gandy, e S. R. J. Sheppard. Reviewing the role of visualization in communicating and understanding forest complexity. Em *Proceedings of Ninth International Conference on Information Visualisation*, páginas 121–128. Citado na pág. 8

Oracle(2014) Oracle. Java SE desktop technlogies, 2014. URL http://www.oracle.com/technetwork/java/javase/tech/index-jsp-138252.html. Acessado em 17 de novembro de 2014. Citado na pág. 13

Pienaar e Shiver(1981) L. V. Pienaar e B. D. Shiver. Survival functions for site prepared slash pine plantations in the flat woods of georgia northern florida. *Southern Journal Forestry*, 5 (2):50–62. Citado na pág. 35

Pinheiro(2006) Cássio David Borralho Pinheiro. Sistema de realidade virtual para suporte ao ensino em redes de computadores. Dissertação de Mestrado, Universidade Federal do Pará. Citado na pág. 6

Romanato(2013) Allan Romanato. Entenda como funciona a Java virtual machine (JVM), 2013. URL http://www.devmedia.com.br/entenda-como-funciona-a-java-virtual-machine-jvm/27624. Acessado em 14 de agosto de 2014. Citado na pág. xi, 12

Santana(2013) Reynaldo Campos Santana. Framework para visualização de florestas 3D. Relatório técnico, Universidade Federal dos Vales do Jequitinhonha e Mucuri. Citado na pág. 21

Santana(2009) Wilma Michele Santos Santana. Crescimento, produção e propriedades da madeira de um clone de eucalyptus grandis e e. urophylla com enfoque energético. Dissertação de Mestrado, Programa de Pós-graduação em Ciência e Tecnologia de Madeira, Universidade Federal de Lavras. Citado na pág. 36

Sheppard e Salter(2004) S. R. H. Sheppard e J. D. Salter. The role of visualization in forest planning. Em *Encyclopedia of Forest Sciences, Oxford, UK*, páginas 486–498. Academic Press/Elsevier. Citado na pág. 2

Shihong *et al.***(2010)** Chen Shihong, Hou Shuang, e Zhai Haijuan. Implementation of virtual forest park interactive walkthrough system. Em *2nd International Conference on Computer and Automation Engineering*, páginas 212–214. Citado na pág. 8

Shreiner(2010) Dave Shreiner. *OpenGL programming guide: the official guide to learning OpenGL, versions 3.0 and 3.1*. Addison-Wesley Professional. Citado na pág. 6

Silva e Araújo(2008) W. L. Silva e R. H. C. Araújo. Visualização síncrona de processos com o OpenGL. *Revista Eletrônica FAINOR*. Citado na pág. 2

Sowizral *et al.***(1998)** Henry Sowizral, Dave Nadeau, Michael Deering, e Mike Bailey. Introduction to programming with with Java 3D, 1998. URL `http://www.sdsc.edu/~nadeau/Courses/Siggraph98java3d/`. Citado na pág. 6

Tang e Bishop(2002) H. Tang e I. D. Bishop. Integration methodologies for interactive forest modeling and visualization systems. *The Cartographic Journal*, 39:27–35. Citado na pág. 7

Tao e Jian-hua(2009) Wang Tao e Gong Jian-hua. Forest reconstruction using point cloud data of airborne LIDAR. Em *Proceedings of International Conference on Management and Service Science*, páginas 1–4. Citado na pág. 8

TIOBE Software(2014) TIOBE Software. TIOBE index for November 2014, 2014. URL `http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html`. Acessado em 17 de novembro de 2014. Citado na pág. 11

Tufte(2001) E. R. Tufte. *The visual display of quantitative information*. Graphics Press. Citado na pág. 6

Udupa(1999) Jayaram K. Udupa. Three-dimensional visualization and analysis methodologies: a current perspective. *Imaging & Therapeutic Technology*, 19(3). Citado na pág. 1

Uusitalo e Orland(2001) Jori Uusitalo e Brian Orland. Virtual forest management: possibilities and challenges. *International Journal of Forest Engineering*, 12(2). Citado na pág. 8

Varejão(2004) Flávio Miguel Varejão. *Linguagens de programação: conceitos e técnicas*. Elsevier. Citado na pág. 11

Wagener e Kelleher(2011) T. Wagener e C. Kelleher. Ten guidelines for effective data visualization in scientific publications. *Environmental Modelling & Software*, 26(6):822–827. Citado na pág. 1

Walsh e Bourges-Sevenier(2001) Aaron E. Walsh e Mikael Bourges-Sevenier. *Core Web 3D*. Prentice Hall PTR, 1 edição. Citado na pág. 6

Wang *et al.*(2006) Xianli Wang, Bo Song, Jiquan Chen, Thomas R. Crow, e Jacob J. LaCroix. Challenges in visualizing forests and landscapes. *Journal of Forestry*, 104(6):316–319. Citado na pág. 2, 8

Ware(2012) Colin Ware. *Information visualization: perception for design*. Morgan Kaufmann, 3 edição. Citado na pág. 1

Xie *et al.*(2010) Xiaokui Xie, Zhenggang Liu, Dongkai Su, Limin Dai, Xinchuang Wang, e Guang Qi. Forest landscape 3-D visualization from China's National Forest Inventory Spatial Database. Em *4th International Conference on Bioinformatics and Biomedical Engineering*, páginas 1–4. Citado na pág. 2, 10

Yongjian *et al.*(2009) Huai Yongjian, Zeng Xi, Yu Peng, e Li Jingli. Real-time rendering of large-scale tree scene. Em *4th International Conference on Computer Science Education*, páginas 748–752. Citado na pág. 10