UNIVERSIDADE FEDERAL DO PARANÁ

JULIANO SIL WASILUK
VIKTOR HANS VON SCHMALZ

SIMULADOR DE COLETA DE DADOS FLORESTAIS PARA A DISCIPLINA DE INVENTÁRIO FLORESTAL DO CURSO DE ENGENHARIA FLORESTAL

CURITIBA
2013

JULIANO SIL WASILUK
VIKTOR HANS VON SCHMALZ

SIMULADOR DE COLETA DE DADOS FLORESTAIS PARA A DISCIPLINA DE INVENTÁRIO FLORESTAL DO CURSO DE ENGENHARIA FLORESTAL

Trabalho apresentado à disciplina de Trabalho de Conclusão de Curso, do curso de Tecnologia em Análise e Desenvolvimento de Sistemas, Setor de Educação Profissional e Tecnológica da Universidade Federal do Paraná.

Orientador: Professor Mestre Rafael Romualdo Wandresen

CURITIBA
2013

RESUMO

A proposta para este trabalho é o desenvolvimento de um simulador tridimensional para a coleta de dados de uma floresta, parte essencial para a formação de profissionais de um curso de engenharia florestal. Utilizando como principais recursos a ferramenta Unity3D para a construção de um ambiente virtual e a tecnologia Java para integração do ambiente com um servidor web, o desenvolvimento do simulador florestal é detalhado desde a sua concepção, visando justificar o projeto em vista das necessidades da criação desta ferramenta de ensino até o seu desenvolvimento, consistindo na implementação do software. Por fim, será apresentado o funcionamento do sistema desenvolvido, o qual consiste em um conjunto de páginas web onde o usuário, após efetuar o seu cadastro e autenticação, acessará um ambiente virtual, podendo então interagir com uma floresta de pinheiros tridimensional por meio de um modelo representando um engenheiro. Desta forma, poderão ser efetuadas uma série de medidas, com o uso de ferramentas específicas, que resultarão no levantamento de um inventário florestal. Os dados obtidos ao final do processo serão exportados no formato de planilha eletrônica para que o usuário, a partir das informações colhidas dentro do simulador, possa efetuar os seus cálculos e estimar a quantidade de volume de madeira da região, objetivo final do processo.

Palavras-chave: Simuladores, inventário florestal, Unity3D.

ABSTRACT

The proposal for this project is the development of a three-dimensional simulator for the data sampling process of a forest, which is essential for the graduation of professionals in a forest engineering course. Using as main resources Unity3D, for the construction of a virtual environment, and Java technology, for the integration of the environment with a Web server, the development of the forest simulator is detailed from its conception, in order to justify the project, in light of the need to create it as a learning tool, to its development, consisting of the actual implementation of the software. Finally, the complete system's functionality will be presented, which consists of a group of web pages where the user, after signing up and logging in, may access a virtual environment, then being able to interact with a three-dimensional pine forest through a model of an engineer. Then, several measurements may be made, with specific tools, which will result in a complete forest inventory process. The data obtained at the end of the process will be exported in a table format so that the user, based on the information collected in the simulator, may calculate an estimate of the volume of wood in the area, which is the final objective of the process.

Keywords: Simulators, forest inventory, Unity3D.

SUMÁRIO

1 INTRODUÇÃO

Atualmente, muitas atividades têm sido reforçadas e/ou facilitadas pela utilização de simuladores virtuais. Segundo Amory (2001), jogos virtuais e simuladores oferecem uma excelente oportunidade para imersão em ambientes de aprendizado, apoiando práticas educacionais contemporâneas. Com o crescimento das tecnologias necessárias para seu desenvolvimento, sua implementação vem se tornando cada vez mais viável, ou mesmo necessária, em termos de adaptação ao mercado. O aprendizado e a prática educacional devem combinar os elementos interativos e interessantes de jogos com um design de sistema instrucional e educacional, incluindo componentes motivacionais, interativos e de aprendizado (Quinn, 1997).

Por outro lado, nas últimas décadas, a comunidade desenvolvedora de aplicativos de realidade virtual tem baseado seu desenvolvimento em trabalhos anteriores de simulação virtual, gráficos 3D interativos e interfaces de usuário (Durlach & Mavor, 1995). Desta forma, foram desenvolvidas novas tecnologias, muito mais amplas, abrangendo tais áreas em conjunto, trazendo ênfase à ciência de desenvolvimento de simuladores virtuais e originando a atual área de pesquisa (Zyda, 2005).

1.1 CONTEXTUALIZAÇÃO

O processo atual de treinamento para coleta de dados florestais, na disciplina de Inventário Florestal do curso de Engenharia Florestal, envolve pesquisa de campo, exigindo dos alunos que realizem atividades práticas da disciplina. Várias medidas são obtidas de diversas árvores de uma região específica, havendo inclusive a necessidade de cubagem de algumas árvores (corte das árvores) para que possa ser calculado o volume estimado de madeira da região.

O Simulador Florestal visa facilitar esse processo, disponibilizando aos alunos uma floresta virtual completa, com medidas de árvores carregadas de um banco de dados, para que aqueles possam aprender e treinar os cálculos necessários para obtenção do volume de madeira de uma região sem a necessidade de realizar tais ações em florestas reais. No simulador, os alunos controlarão um engenheiro

florestal, o qual pode selecionar e medir árvores em uma parcela delimitada da floresta. As medidas feitas serão salvas pelo sistema.

1.2 JUSTIFICATIVA DO PROJETO

Este trabalho é parte integrante da pesquisa de doutorado do professor Rafael Romualdo Wandresen, que é aluno regularmente matriculado no programa de pós graduação em Engenharia Florestal da UFPR, área de concentração Manejo Florestal, sob a linha de pesquisa Processamento de Imagens e Sistemas de Informação Espaciais Aplicadas aos Recursos Naturais, e está sob orientação do professor Dr. Henrique Soares Koehler.

Segundo Wandresen (2010), em seu projeto de doutorado, a aplicação de um software específico para o ensino das disciplinas voltadas ao Inventário Florestal justifica-se para melhorar a qualidade do ensino das disciplinas.

Como parte integrante da pesquisa de doutorado de Wandresen (2010), este trabalho justifica-se por possibilitar a simulação da etapa de coleta de dados de uma parcela de uma floresta.

Segundo Sanquetta *et al.* (2009), existem diversas medidas que são tomadas em inventários florestais. Estas medidas podem ser obtidas diretamente com equipamentos de medição ou indiretamente por meio de soluções geométricas ou trigonométricas. Neste trabalho serão simuladas três medidas: altura, diâmetro na altura do peito (DAP) e o processo de obtenção de medidas para cubagem da árvore derrubada.

O processo de coleta de dados florestais exige um treinamento específico – sendo este o objetivo da disciplina de Inventário Florestal. Para facilitar o processo de treinamento, bem como possibilitar aos alunos uma experiência mais real da coleta de dados florestais, disponível de qualquer lugar, foi desenvolvido um simulador.

1.3 OBJETIVOS

1.3.1 Objetivo Geral

Desenvolver um simulador do processo de coleta de dados de uma parcela de uma floresta para os alunos da disciplina de Inventário Florestal, do curso de Engenharia Florestal, utilizando a engine de jogos Unity3D. O simulador deverá operar dentro de um sistema *web* e os dados medidos deverão ser exportados para um arquivo do tipo planilha, tal como o Microsoft Excel.

1.3.2 Objetivos Específicos

1. Análise de requisitos iniciais e planejamento do software a ser gerado
2. Familiarização com a ferramenta Unity3D
3. Desenvolvimento dos modelos 3D a serem utilizados
4. Criação e configuração do banco de dados a ser utilizado pelo simulador e estabelecimento da conexão
5. Criação e configuração de um servidor e de páginas web.
6. Desenvolvimento dos scripts em C Sharp para instanciação dos modelos em cena e implementação de cada funcionalidade
7. Correção de bugs
8. Implementação do simulador

2 FUNDAMENTAÇÃO TEÓRICA

2.1 DESENVOLVIMENTO DE JOGOS

O processo de desenvolvimento de jogos envolve três fatores principais: história ou propósito, arte, e software (Zyda, 2005). Ao desenvolver um jogo, a equipe de desenvolvimento combina estes três fatores de modo a gerar um produto finalizado. A equipe de design define os propósitos, ou objetivos, do jogo, e mesmo a sua história (elemento presente na grande maioria dos jogos sérios da atualidade). A equipe de arte traz todos os elementos gráficos ao jogo, definindo sua aparência e a sensação dos futuros jogadores. A equipe de programação desenvolve o código que irá implementar as funcionalidades necessárias para a realização do propósito do jogo, previamente definido pela equipe de design (Zyda, 2005).

O Simulador Florestal tem como propósito a amostragem de dados florestais, traz como arte as próprias ferramentas, modeladas, o engenheiro, o terreno, as árvores e os diversos outros elementos visuais nele presentes (detalhados no capítulo 4), e tem como software as classes programadas (Engenheiro, Arvore, Terreno, Colecoes, entre outras – mais detalhes nos diagramas no apêndice) e todas as funcionalidades implementadas, como as de realizar medidas e delimitar espaço.

Atualmente, o desenvolvimento de jogos vem encarando uma série de desafios. De modo que novas tecnologias (como *engines* de jogos) vão sendo produzidas e atualizadas, o tempo de treinamento para seu uso se torna mais e mais extenso, aumentando conseqüentemente o custo e a duração do projeto (Zyda, 2005). Além deste fator, há também a crescente demanda do mercado por gráficos e funcionalidades equivalentes ou melhores que os dos jogos atuais, visando utilizar ao máximo estas novas tecnologias.

A tecnologia utilizada para o desenvolvimento do Simulador Florestal foi o Unity3D. O Unity3D é uma engine de jogo – ou seja, um software que traz funcionalidades básicas para o desenvolvimento de um jogo, entre elas um *Renderer* (um programa/componente responsável por gerar imagens a partir de modelos tridimensionais), um *motor de física* (componente responsável por aplicar condições de física, como colisões e gravidade, a objetos dentro da *engine* de jogo),

apoio a som, animações, programação em determinadas linguagens e gerenciamento de memória.

As linguagens de programação para desenvolvimento de scripts no Unity3D são C# (C Sharp) e JavaScript (as linguagens são detalhadas em 3.5 – materiais). Os scripts nessas linguagens podem ser acionados pela engine e manipular as variáveis relacionadas aos objetos instanciados em uma cena (uma cena é um contexto criado pela engine para a instanciação de objetos, como um terreno, um personagem, uma câmera, etc).

Engines de jogo como o Unity3D visam reutilizar grandes quantidades de código básico, permitindo o desenvolvimento de jogos e simuladores em um nível mais alto, com custos reduzidos.

2.2 JOGOS SÉRIOS E SIMULADORES

A definição de “Jogo (Virtual) Sério”, segundo Zyda (2005), é melhor descrita como “uma atividade mental, realizada com um computador, a qual segue determinadas regras e padrões, tendo como finalidade o lazer, a recreação ou a competição por prêmios, ao mesmo passo que envolve elementos de aprendizado, ou seja, transferência de conhecimento.” Para Zyda, o objetivo principal de um jogo sério ainda reside no lazer, assim como jogos comuns. Já Michael e Chen (2006) definem “jogos sérios” como “jogos que não possuem o lazer, ou entretenimento, como objetivo primário”. Apesar de tal definição, os autores ressaltam que isto não significa que jogos sérios não possam, de alguma forma, promover o lazer; pelo contrário, eles defendem que, idealmente, jogos sérios deveriam conter um nível de lazer para que os jogadores se envolvam mais com o processo.

O grande diferencial de jogos sérios em relação a jogos comuns é que, além dos elementos básicos (história ou propósito, arte, e software), aqueles possuem o elemento adicional da *pedagogia* (Zyda, 2005). Alguns dos fatores que também diferenciam jogos sérios de jogos comuns são a diferença de foco (jogos sérios focam-se na resolução de problemas enquanto jogos comuns focam-se em experiências ricas), a simulação da comunicação (em jogos sérios, a comunicação deve ser realista – muitas vezes, imperfeita – enquanto em jogos comuns a comunicação é normalmente perfeita), e a utilização de fatores randômicos que afetem o percurso do jogador (em jogos sérios, é aconselhável evitar seu uso, pois o

foco está no aprendizado do jogador, trazendo maior importância às consequências das ações do mesmo ao invés de um fator dependente de sorte) (Michael & Chen, 2006).

Um exemplo de jogo sério, atualmente utilizado pelo exército americano como um simulador efetivo para treinamento, é o *America's Army* (Zyda, 2005). Inicialmente implementado por um sargento de Fort Benning, o jogo provê treinamento real para recrutas em diversos de seus cenários. Zyda e Bennett (2002) baseiam-se no sucesso educacional obtido pelo jogo para propor o desenvolvimento de um jogo sério altamente imersivo visando a educação em ciências ou matemática, denominado "First-Person Education" – nome originado da expressão "First-Person Shooter", a qual é a definição para jogos do gênero do *America's Army*.

Tendo isto em vista, o Simulador Florestal pode ser caracterizado como um jogo sério, visto que apresenta as características de um jogo (uma imersão em um ambiente virtual, controlado por um computador, com um propósito) e possui uma finalidade voltada para o aprendizado.

2.3 OBJETOS DE APRENDIZAGEM

Por definição, objeto de aprendizagem é todo recurso autoconsistente (ou seja, independente de outros recursos para seu funcionamento) utilizado em um processo de ensino (Mason, 1998). Beck (2001) define um objeto de aprendizagem como "qualquer recurso digital que possa ser reutilizado para o suporte ao ensino". Segundo Beck, a ideia principal de um objeto de aprendizagem é a divisão do conteúdo educacional para seu posterior reuso em contextos diversos – conceito semelhante ao princípio do paradigma de programação Orientado a Objetos.

O Simulador Florestal é um exemplo de objeto de aprendizagem, visto que é um simulador virtual – para ser utilizado por alunos – independente de outros recursos que também possam ser utilizados dentro do mesmo contexto, com o principal objetivo voltado para o aprendizado. O Simulador Florestal visa simular o processo de amostragem de uma área florestal. Tal amostragem pode ser feita em uma área fixa ou em uma área variável (Sanquetta *et al.*, 2009). O método representado no simulador é o de área fixa.

2.4 MÉTODO DE AMOSTRAGEM DE ÁREA FIXA FLORESTAL

Neste método de amostragem, é determinada uma área fixa da floresta sobre a qual ocorrerá o processo de amostragem. A seleção das árvores a serem medidas depende diretamente do tamanho da área selecionada e da frequência (distância) das árvores nela (Sanquetta *et al.*, 2009). O método de área fixa é o mais antigo e utilizado em inventários florestais, pois é simples de utilizar e possui muitas possibilidades de uso. A área selecionada pode ser definida em diversas formas, sendo as mais utilizadas as circulares, quadradas, retangulares ou em grupos (conglomerados). A forma da área representada no Simulador Florestal é retangular.

As unidades de amostra retangulares possuem dimensões de base e altura (B x H) e podem ser de város tamanhos, geralmente entre 1000 m^2 e 10000 m^2. Um tamanho popularmente utilizado é 2500 m^2, sendo a área equivalente à quarta parte de um hectare (Sanquetta *et al.*, 2009).

As principais vantagens da utilização do método de área fixa, em relação aos demais, são a simplicidade/praticidade de estabelecer uma área para amostragem, a facilidade de manutenção e correlação entre medidas sucessivas de uma mesma árvore em períodos de tempo extensos e a possibilidade de obter os valores de estimativa na unidade da amostra medida de forma direta, sem exigir conversões e aplicações de fórmulas mais complexas (Sanquetta *et al.*, 2009).

2.5 MEDIDAS DAS ÁRVORES

Em inventários florestais, há diversas medidas a serem realizadas em árvores. A ciência da medição destas grandezas é denominada *dendrometria* – *dendron,* do Grego, significando “árvore”, e *metria*, do Latim, significando “medida” (Machado & Figueiredo Filho, 2003). Estas medidas podem ser realizadas diretamente (manualmente), com ferramentas, ou indiretamente, com soluções geométricas ou trigonométricas, ou mesmo através de imagens (Sanquetta *et al.*, 2009). As principais medidas realizadas nas árvores são o DAP (Diâmetro à Altura do Peito), a área basal/transversal, a altura (podendo ser comercial ou total), a distância, a idade, o volume e a espessura da casca. As medidas representadas no Simulador Florestal são do DAP e da altura da árvore.

2.5.1 DAP (Diâmetro à Altura do Peito)

O diâmetro é uma medida extremamente utilizada em inventários florestais, devido ao fato de que pode ser medido de forma simples e direta, e, a partir do mesmo, pode-se calcular diversas outras grandezas da árvore, como a sua área transversal ou seu volume (Sanquetta *et al.*, 2009). O ponto padrão de altura para medida do diâmetro de uma árvore é de 1,30 m, ou seja, a altura do peito – assim denominando a grandeza de DAP – Diâmetro à Altura do Peito.

Para a realização da medida do DAP de uma árvore, podem ser usadas diversas ferramentas. As mais comuns são a suta, podendo ser mecânica ou eletrônica, a fita métrica e a régua de Biltmore (Sanquetta *et al.*, 2009) – uma régua com graduações especiais para medida do DAP de uma árvore, diferentes das de uma régua comum, a qual é graduada uniformemente. A ferramenta utilizada no simulador é uma suta mecânica (ver figura 1).

Figura 1: Suta mecânica. Trata-se de uma régua graduada com duas barras paralelas, sendo uma imóvel e outra móvel. Fonte: http://www.eloforte.com/v2/components/com_virtuemart/shop_image/product/Suta_mecanica002.jpg

2.5.2 Altura

A altura de uma árvore é uma grandeza fundamental para o cálculo do seu volume. Trata-se da distância linear entre o nível do solo e seu ápice. Uma árvore possui uma altura comercial e uma altura total – a primeira limitando-se ao ponto onde o caule da árvore atinge um diâmetro mínimo para uso (ou alguma outra

limitação para uso comercial, como galhos, bifurcações ou tortuosidades), e a segunda incluindo sua copa inteira (Sanquetta *et al.*, 2009).

O processo de medida da altura de uma árvore pode ser realizado de forma direta, por meio de vara trena ou mesmo visualmente (procedimento não recomendado devido à sua imprecisão), ou de forma indireta, através de instrumentos denominados *hipsômetros* (Sanquetta *et al.*, 2009). Há hipsômetros baseados em princípios geométricos ou trigonométricos, bem como o hipsômetro Vertex, o qual utiliza impulsos ultrassônicos para determinar a distância da árvore, e possibilita a medida da altura de uma árvore de até 999 metros, a uma distância de até 30 metros, com precisão e de forma prática. Há também o clinômetro eletrônico, um aparelho extremamente compacto, pesando aproximadamente 50 gramas, o qual mede a altura da árvore e o ângulo do seu ápice em relação a um plano horizontal sem necessidade de uma distância fixa da árvore (Sanquetta *et al.*, 2009). A ferramenta utilizada no simulador é um hipsômetro Vertex III (ver figura 2).

Figura 2: Hipsômetro Vertex III. Fonte: http://www.gisiberica.com/hips%F3metros/VERTEX4.jpg

3 METODOLOGIA

3.1 MODELO DE PROCESSO DE ENGENHARIA DE SOFTWARE

O processo de engenharia de software utilizado para o desenvolvimento do projeto foi baseado no RUP (Rational Unified Process), utilizando as suas fases principais. O RUP é um processo iterativo de desenvolvimento de software, no qual o projeto é dividido em quatro fases: concepção, elaboração, construção e transição. Em cada fase, há uma quantidade de iterações – quantas forem necessárias para atingir os objetivos. Abaixo estão definidas cada uma das fases, com seus principais objetivos, bem como uma referência ao projeto do Simulador Florestal.

3.1.1 Concepção

A fase de Concepção é caracterizada por definir o que será desenvolvido com o projeto, bem como uma justificativa para tal. Nela, deve ser definido o escopo aproximado do projeto, uma visão básica dos casos de uso do sistema, bem como devem ser identificados os riscos e estimados os custos do projeto, para que possa ser traçado um cronograma básico para o mesmo. É a fase mais curta do projeto, visando apenas uma especificação geral do que será produzido.

No projeto do Simulador Florestal, foi definida a ideia de desenvolver um simulador para o processo de coleta de dados florestais na disciplina de Inventário Florestal, tendo como justificativa principal facilitar o processo de treinamento dos alunos. Os primeiros requisitos foram levantados, para gerar uma visão geral do que seria produzido, e foram definidos alguns casos de uso e os principais riscos para o projeto (ver WBS no apêndice – página 59).

3.1.2 Elaboração

Na fase de Elaboração, são levantados os requisitos do sistema em detalhe, bem como é definida uma arquitetura para o sistema. Vários diagramas são criados nesta fase para a representação do sistema, como o Diagrama de Casos de Uso, o Diagrama de Classes Conceitual, e o Diagrama Entidade-Relacionamento. Em

suma, nesta fase é criado um plano completo para o desenvolvimento do software – um projeto do software, incluindo um cronograma realista e a especificação clara dos principais requisitos e casos de uso do sistema – para que este possa começar a ser desenvolvido.

No projeto do Simulador Florestal, foram definidas a grande maioria das funcionalidades do simulador nesta fase. Os diagramas iniciais foram feitos e um cronograma básico foi traçado para que o sistema pudesse ser desenvolvido de maneira organizada.

3.1.3 Construção

É na fase de Construção que o sistema propriamente dito é desenvolvido – o código é gerado. É a fase mais longa do projeto, necessitando de várias iterações para seu término. Em cada iteração, uma funcionalidade é desenvolvida e testada, assim acrescentando novas funcionalidades ao sistema de forma cíclica. É também nesta fase que são desenvolvidos os diagramas de implementação – representações mais próximas do próprio código do sistema.

No projeto do Simulador Florestal, cada iteração durou de uma a duas semanas, nas quais cada funcionalidade (caso de uso) foi sendo implementada. As funcionalidades do simulador, em detalhe, estão especificadas no tópico 3.7.6.

3.1.4 Transição

A fase de Transição tem como objetivo a implementação do sistema e seu uso pelos usuários finais. Nesta fase, é utilizado o feedback recebido pelos usuários para que ocorram refinamentos no sistema em cada iteração. O treinamento de usuários, se necessário, também ocorre nesta fase do projeto.

A fase de transição do Simulador Florestal se deu com a apresentação e validação do sistema com o professor orientador.

3.2 PLANO DE ATIVIDADES

O plano de atividades para o desenvolvimento deste projeto foi dividido em cinco partes, segundo a divisão das quatro etapas principais do RUP: Concepção,

Elaboração, Construção e Transição, bem como uma parte adicional referente ao gerenciamento. O diagrama WBS, no apêndice (página 59), ilustra a divisão detalhada das partes do projeto, bem como das tarefas específicas realizadas em cada uma.

A primeira etapa, a concepção do trabalho, teve como principais atividades o levantamento de requisitos do sistema (tanto da parte web quanto do simulador), a definição dos principais casos de uso (gerando um diagrama de Casos de Uso preliminar simplificado), uma estimativa de riscos mais prominentes, e a seleção de ferramentas para o desenvolvimento de cada parte do sistema (servidor, páginas web, simulador, modelos 3D).

Logo em seguida, iniciou-se a etapa da elaboração. Nesta, foi desenvolvida toda a documentação do sistema (Diagrama de Casos de Uso, Diagramas de Classes, Diagramas de Sequência, DER – detalhados no apêndice), num primeiro momento de forma superficial, e logo em seguida aprofundados e concluídos. Também foram desenvolvidos protótipos para o ambiente virtual do simulador, as páginas web e as interfaces do usuário dentro do simulador.

A próxima etapa do projeto foi a mais longa – a construção. Nesta foi desenvolvido o sistema em si. Foi subdividida em sete partes: desenvolvimento de modelos tridimensionais, animações, servidor, banco de dados, páginas web, sons e scripts para implementação de funcionalidades específicas. Estes foram classificados, a princípio, em “básicos” e “avançados”, sendo desenvolvidos em ordem crescente de complexidade.

A quarta etapa do projeto foi a transição, na qual ocorreu a integração do simulador, ou seja, a finalização e junção de todos os seus elementos e funcionalidades e o refinamento de elementos gráficos nele presentes. Tal integração foi dividida em três partes, detalhadas no diagrama WBS (página 59). Nesta etapa também foram gerados pacotes de instalação, foi concluída toda a documentação final do projeto (diagramas e parte textual, bem como sua ligação, através do desenvolvimento deste documento) e foram realizados testes com o sistema.

3.3 PLANO DE RISCOS

#	Condição	Data Limite	Consequência	Ação	Responsável por Monitorar	Probabilidade	Impacto	Classificação
1	Dificuldade de utilização de tecnologia	01/09/2013	Atraso nas atividades, necessidade de treinamento	Realização de testes com a tecnologia para aprendizado de uso	Grupo	Alta	Médio	6
2	Cronograma não realista	Final do projeto	Impossibilidade de seguir o cronograma	Considerar o tempo necessário para realização das atividades e folga para o imprevisto	Grupo/ Orientador	Média	Alto	6
3	Defeito na máquina contendo o software	Último dia do projeto	Perda de todo o trabalho e necessidade de refazer	Realizar backups em locais separados para restauração	Grupo	Baixa	Alto	5
4	Falta/saída de um membro da equipe	Último dia do projeto	Atraso nas atividades, redistribuição de responsabilidades		Grupo/ Orientador	Muito Baixa	Muito Alto	5
5	Falhas de comunicação	30/11/2013	Atraso no cronograma, possível perda de progresso	Realizar reuniões semanais e trocar e-mails regularmente	Grupo/ Orientador	Média	Médio	5
6	Excesso de mudança de requisitos	12/11/2013	Atraso no cronograma, erro no desenvolvimento	Especificar requisitos semanalmente em reuniões	Grupo/ Orientador	Média	Médio	5

Tabela de Classificação de Riscos

	Impacto Muito Baixo	Impacto Baixo	Impacto Médio	Impacto Alto	Impacto Muito Alto
Probabilidade Muito Baixa	1	2	3	4	5
Probabilidade Baixa	2	3	4	5	6
Probabilidade Média	3	4	5	6	7
Probabilidade Alta	4	5	6	7	8
Probabilidade Muito Alta	5	6	7	8	9

Riscos com classificação menor ou igual a 3 são de criticidade baixa. Riscos com classificação 4, 5 ou 6 são de criticidade média, e riscos com classificação maior ou igual a 7 são de criticidade alta.

3.4 RESPONSABILIDADES

O projeto foi separado em dois grandes módulos: o simulador e o servidor. Apesar dos dois membros da equipe manterem constante visão e organização sobre todas as partes do projeto, as responsabilidades foram divididas da seguinte forma:

-Viktor: Simulador – Modelagens de objetos tridimensionais, animações, sonoplastia, delimitação espacial e scripts para manipulação dos modelos e componentes criados.

-Juliano: Servidor – Desenvolvimento do servidor, páginas *web*, criação e gerenciamento do banco de dados e scripts para integração do simulador com o servidor.

Além do software, existe também a parte da documentação. Esta também foi revisada por ambos os membros da equipe, porém nela também houve uma divisão clara de responsabilidades:

-Viktor: Produção textual e embasamento teórico

-Juliano: Diagramas e plano de atividades

3.5 MATERIAIS

3.5.1 Unity3D

O Unity3D é uma engine de jogos (ver definição em 2.3). Nele, há uma variedade de ferramentas para construção de elementos que poderiam ser encontrados em jogos ou simuladores, como personagens, câmera, luz, terreno, e objetos com formas pré-definidas (como cubos, esferas, etc). Há também elementos da natureza, como árvores, moitas, grama, entre vários outros.

O Unity3D possui uma grande quantidade de scripts nativos, para execução de funcionalidades básicas, como controle de um personagem em terceira pessoa. Estes scripts são feitos em JavaScript (ver tópico 3.6.2.2). O Unity3D vem com um editor de scripts chamado MonoDevelop, onde cada script pode ser alterado e novos

scripts podem ser feitos. Scripts também podem ser criados com qualquer editor de texto e futuramente importados no Unity3D, mas a ferramenta MonoDevelop oferece um bom suporte durante a programação, a evidência de erros de compilação, e auto-completamento de nomes de variáveis ou métodos chamados para conveniência do programador. Scripts no Unity3D podem ser escritos em JavaScript (tópico 3.5.2.2) ou em C Sharp (tópico 3.5.2.1).

Na interface do Unity3D, há duas telas (na esquerda da interface) e três abas: Hierarchy, Project e Inspector. A tela superior corresponde a uma visão da cena sendo editada – melhor descrita como uma “câmera” somente para o editor – não é uma câmera presente na cena; apenas serve para que o editor visualize os objetos em cena. Já a tela inferior corresponde à visão da câmera principal da cena – esta sim é uma câmera real, e será o ponto de visão durante a execução do jogo.

A aba Hierarchy contém todos os elementos instanciados em cena. Nela pode-se visualizar cada objeto. Objetos podem também se agrupar formando uma espécie de hierarquia. Em termos de orientação a objetos, esta hierarquia apenas estende para a classe filha a posição e a rotação da classe mãe, de forma que, por exemplo, uma chave inglesa, que configura um objeto, na mão de um homem, que configura outro objeto, sempre acompanhará os movimentos da mão deste homem.

A aba Project contém todas as pastas do projeto. Nela estão organizados os recursos utilizados no projeto inteiro, como modelos, scripts e sons. Na aba Inspector, pode-se visualizar cada variável associada a um objeto ao selecioná-lo. Como o próprio nome da aba já sugere, ele serve para a “inspeção” de um elemento e suas características. Segue na página seguinte a interface principal do Unity3D.

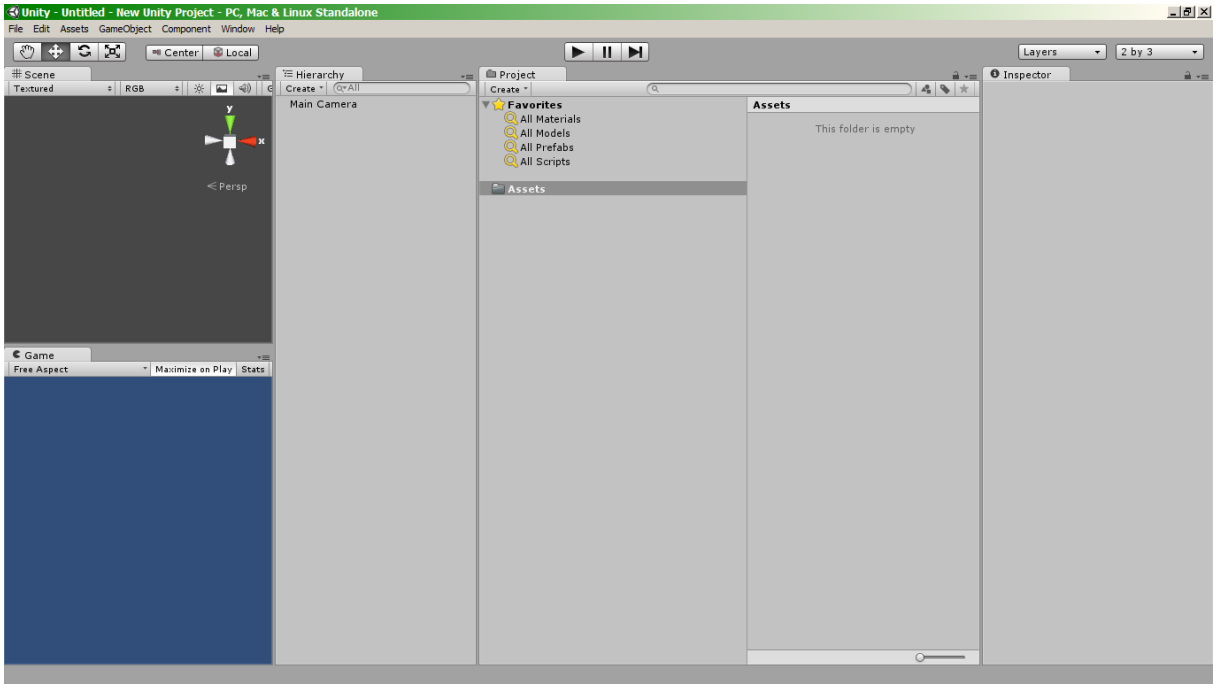

Figura 3: Interface principal do Unity3D.

O desenvolvimento do Unity3D, a grosso modo, resume-se em inserir objetos dentro de um espaço inicialmente vazio, que define uma cena, e programar um ou mais comportamentos a estes objetos. Cada objeto possui uma posição no espaço que são definidos com base no sistema de eixos cartesianos. Ângulos de rotação para os três eixos também são levados em consideração, totalizando seis variáveis para demarcação de um objeto na cena.

Praticamente tudo o que é inserido dentro da cena é um objeto. Objetos no Unity3D são denominados *GameObjects*. Eles podem ser modelos tridimensionais, criados pelo *designer* e importados para a *engine*, câmeras ou iluminação, que não possuem necessariamente uma representação poligonal. Também podem ser objetos abstratos que simplesmente carregam algum script para exercer alguma funcionalidade, não sendo necessariamente visíveis.

O que diferencia um *GameObject* de outro são objetos anexados a ele chamados de *Components*. Um *component* do tipo *Light* (luz) referenciado em um *GameObject* dará propriedades de iluminação ao objeto. Já um *component* de *physics* (física) fará com que o objeto esteja sujeito às leis da física, respondendo a forças externas, colidindo com demais objetos e assim por diante. Um *component* obrigatório a qualquer *GameObject* é o *Transform*. Sua utilidade está em definir as seis variáveis de localização espacial de um objeto na cena. Há uma infinidade de

components no Unity3D, de forma a contemplar os mais variados tipos de elementos presentes no mundo real.

Scripts desenvolvidos também são *components.* Se um *GameObject* for pensado como uma classe de um programa orientado a objetos, seria justo dizer que os scripts seriam os métodos desta classe, pois são eles que alteram as propriedades de tudo o que for relativo aquele objeto. De fato, *GameObjects* são classes, assim como *components.* Entretanto esta percepção não está explicita ao desenvolvedor pois toda a manipulação no editor é visual. Quando um *GameObject* é inserido dentro de uma cena, ele passará a ser uma instância da classe do *GameObject* e terá as propriedades e atributos definidos por uma série de componentes e poderá ter seu comportamento alterado durante a execução da cena, tal como foi programado, por meio dos *components* do tipo script.

Um projeto do Unity3D pode ser composto de mais de uma cena. Cenas novas geralmente são carregadas quando determinadas condições programadas dentro de um script são atendidas. Por exemplo, se em um projeto a passagem para dentro de um castelo exige o uso de uma chave, a cena do interior do castelo só irá carregar quando um personagem estiver próximo à porta de entrada e portando a chave necessária.

Com os objetos dentro de uma cena e com seus scripts definidos, o próximo passo é testar. Como a compilação do projeto pode ser um pouco demorada, é possível rodar a aplicação dentro do próprio editor do Unity3D. Clicando em um ícone (*play*) o projeto poderá ser testado e será exibido na tela inferior esquerda do editor, que é chamada de "*Game*". Caso ocorra algum erro em algum script, uma janela chamada "*console*" informará que há um erro, informando também a natureza da falha, o script em que ele se encontra e a linha.

A etapa final do desenvolvimento é a compilação do projeto. O Unity3D possui a vantagem de gerar executáveis para as mais diversas plataformas. É possível rodar um mesmo projeto em versões *desktop* para os sistemas operacionais Windows, Linux e Mac OS, na *Web* por meio de um *plugin* (para as plataformas Windows e Mac OS), em dispositivos móveis (Android, iOS, BlackBerry e Windows Phone 8), e finalmente, para consoles (Playstation 3, XBox 360 e Wii), embora para estes últimos sejam necessárias licenças adicionais.

A versão utilizada para o desenvolvimento do simulador é a 4.2.0f4. O manual do Unity3D pode ser encontrado no link abaixo:

http://docs.unity3d.com/Documentation/Manual/index.html

3.5.2 Linguagens de programação

O Unity3D permite o desenvolvimento de scripts em duas linguagens de programação: C Sharp (C#) e JavaScript. Para o projeto, a linguagem predominante que foi utilizada foi o C Sharp por conta da abordagem fortemente orientada a objetos e sintaxe semelhante com outras linguagens já aprendidas pela equipe, como C e Java.

Outro fator importante pela escolha do C Sharp foi o fato de que o JavaScript é implementado no Unity3D apenas em termos de sintaxe, e não das bibliotecas padrão da linguagem. Na *engine*, independente da linguagem escolhida para *scripting* , o programador acabará tendo que utilizar o *framework* Mono, implementação livre do .NET. Então, para evitar confusões e frustrações de tentar utilizar funções já conhecidas no JavaScript e inexistentes no Unity3D foi utilizado o JavaScript apenas quando foram utilizados scripts nativos da *engine* escritos nesta linguagem.

Para o lado do servidor e páginas web, optou-se pelo uso da tecnologia Java. A escolha foi feita para facilitar a integração e suporte com o servidor do professor orientador, que foi escrito em Java, e também pela facilidade da equipe com a linguagem, tanto na sua programação quanto nas ferramentas oferecidas para o desenvolvimento, como a IDE Eclipse.

3.5.2.1 C Sharp

A linguagem C Sharp é uma linguagem multiparadigma (pode ser utilizada seguindo uma grande variedade de paradigmas, porém é mais comumente orientada a objetos) desenvolvida pela Microsoft. Sua finalidade é a de ser uma linguagem de programação orientada a objetos simples e de propósito geral. Sua sintaxe possui uma grande semelhança com a linguagem Java, apesar de ser definida pelos seus desenvolvedores como “uma incrementação da linguagem C++”.

O Unity3D utiliza uma implementação livre do *Framework* proprietário da Microsoft .NET denominado Mono, permitindo o uso do C Sharp em praticamente qualquer sistema operacional (MONO, 2013).

3.5.2.2 JavaScript

A linguagem JavaScript é uma linguagem de programação interpretada, comumente utilizada em aplicativos web. Sua execução ocorre ao lado do cliente (client-side), em oposição a linguagens que executam ao lado do servidor (como, por exemplo, a linguagem PHP).

O Unity3D utiliza o JavaScript em uma implementação própria com uso de bibliotecas ao estilo .NET, não disponibilizando a maioria das funcionalidades nativas da linguagem, como aquelas costumeiramente utilizadas em páginas web. Em todo caso, quando o Unity3D se comunica com a página web que o hospeda em um projeto rodando na web, ele faz chamadas a funções escritas em JavaScript que estão na página. Neste caso, a linguagem obedece sua implementação padrão.

3.5.2.3 Java

Java é uma linguagem de programação orientada a objetos amplamente utilizada, pois além de ser gratuita, possui alto desempenho e é multiplataforma (JAVA, 2013). A linguagem executa em uma máquina virtual chamada JVM (Java Virtual Machine) que converte os bytecodes em linguagem de máquina. É esta camada que garante que uma aplicação Java seja capaz de ser executada em qualquer sistema operacional sem a necessidade de adaptar o código fonte.

A especificação Java EE (Java Enterprise Edition) traz suporte à tecnologia Java para a web, proporcionando o desenvolvimento de páginas dinâmicas (jsp - JavaServer Pages), a criação de servlets (traduzido como servidorzinho), que são classes especiais que estendem a funcionalidade de um servidor e por isso são capazes (mas não se resumem a somente) tratar e responder à requisições feitas por meio do protocolo HTTP, entre outros recursos. No projeto Simulador Florestal as páginas web possuem código Java e toda a comunicação entre o banco de dados e o simulador é feita com o auxílio dos servlets. Esta comunicação será descrita em detalhes no capítulo 4.7.

3.5.3 SketchUp

O Google SketchUp foi o software utilizado para a criação dos modelos em 3D presentes no simulador (detalhados em 3.6.4 – Modelagem). Há diversas outras ferramentas disponíveis para o mesmo propósito, como Blender, Maya, 3DSMax ou ZBrush – porém, o SketchUp foi selecionado para uso devido a sua interface mais prática que a dos demais, e a possibilidade de exportação para os formatos utilizados pelo Unity3D. Esta escolha foi fundamental para a otimização do tempo para o desenvolvimento do projeto.

As funcionalidades básicas de um editor de modelos em 3D são de desenho de formas geométricas básicas (como quadrados ou círculos), bem como linhas e pontos, e a ferramenta de extrusão – elevação de uma face na terceira dimensão – de modo a possibilitar a construção manual de um objeto em 3D. Há também a possibilidade de texturização do objeto, após a modelagem da sua forma, e por fim, o modelo criado é exportado para o formato .FBX – formato de um objeto em 3D reconhecido pelo Unity3D. O SketchUp possui todas estas funcionalidades, e por ser um software leve e de fácil aprendizagem, não apresentou nenhum problema durante sua execução e possibilitou o desenvolvimento de diversos modelos utilizados no simulador sem consumir demasiado tempo de produção.

3.5.4 MySQL

MySQL é um popular sistema gerenciador de banco de dados de código aberto disponível para as mais variadas plataformas. É fácil e simples de utilizar, possuindo alto desempenho e é amplamente utilizado na *web* (MYSQL, 2013a). Sites populares, como Youtube, Twitter e Facebook utilizam este banco de dados (MYSQL, 2013b).

3.5.5 JSON

JSON (JavaScript Object Notation - Notação de Objetos JavaScript) é um formato leve para troca de dados computacionais. É fácil para que humanos possam ler e escrever e é rápido para o computador gerar e decodificar. Apesar de possuir "JavaScript" em seu nome, é independente de linguagem de programação e

qualquer linguagem que tenha uma biblioteca para tal função é capaz de interpretar o formato (JSON, 2013).

É utilizado para troca de dados entre plataformas diferentes, e em geral é construído com base em pares de chave da forma "nome" e "valor". Um valor também pode ser uma lista de vários outros objetos que também podem ser composto por pares de chaves, e assim por diante, gerando estruturas complexas de acordo com a demanda da aplicação que utilize a tecnologia. No projeto Simulador Florestal toda a troca de dados entre o simulador e o servidor web é feita por meio de transporte de estruturas codificadas no formato JSON. Esta comunicação será detalhada no capítulo 4.

Foram usadas duas bibliotecas para a manipulação dos JSONs, a LitJSON para C Sharp, chamada dentro dos scripts do simulador e a SimpleJSON, usada no servidor web.

3.5.6 Eclipse

Eclipse é uma IDE *open source* desenvolvida em Java inicialmente pela IBM em 2001 utilizada para a edição de códigos de linguagens de programação. Apesar de amplamente utilizada para o desenvolvimento em Java, possui suporte para as mais diversas linguagens, como C,C++ e PHP. Hoje é mantida pela organização sem fins lucrativos *Eclipse Foundation* e fornece versões gratuitas da IDE para os mais variados fins (ABOUT, 2013).

Para este projeto foi utilizado o E*clipse Java EE IDE for Web Developers* na versão *Luna Release* para o desenvolvimento do servidor e páginas *web.*

3.5.7 Monodevelop

Monodevelop é uma IDE gratuita designada para a edição de códigos C Sharp e outras que utilizam *Frameworks* baseados em .NET. É um software multiplataforma com versões para Windows, Linux e Mac OS sendo possível portar com facilidade projetos desenvolvidos com o Microsoft Visual Studio (MONODEVELOP, 2013).

É a IDE padrão que é instalada com o Unity3D, com o propósito de servir de editor para os scripts da *engine*. Para este projeto foi utilizada a versão 2.8.2.

3.5.8 Apache Tomcat

Apache Tomcat é um software open source que é utilizado para a implementação de Java Servlets e de JavaServer Pages (APACHE, 2013). Pertence a uma categoria de servidores chamados de *servlet containers*.

A versão utilizada para o desenvolvimento do Simulador Florestal é a versão 7.0.

3.6 DESENVOLVIMENTO DO PROJETO

Segue abaixo um fluxograma ilustrando, passo a passo, o desenvolvimento do projeto. Logo em seguida, são detalhadas as etapas de forma textual.

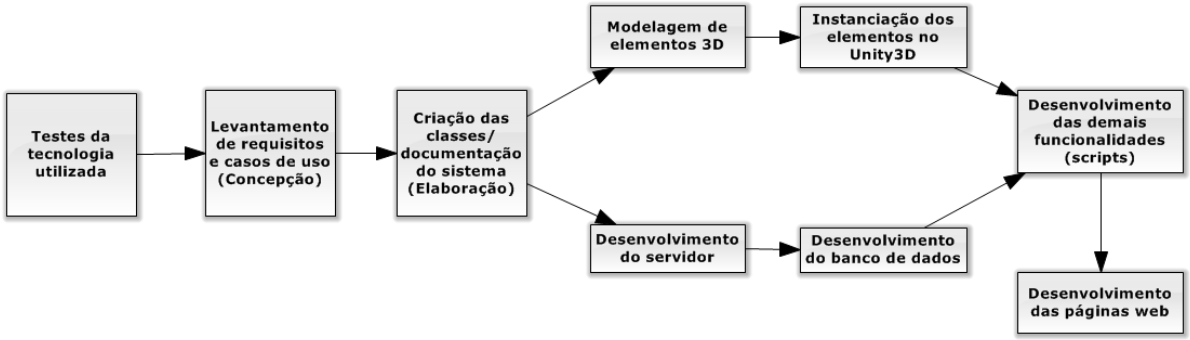

Figura 4: Fluxograma do desenvolvimento do projeto.

3.6.1 Testes da tecnologia utilizada

Em um primeiro momento, foi necessário realizar testes com a tecnologia Unity3D, para adaptação à interface e aprendizado dos seus elementos mais básicos. Foram criadas diversas cenas, com terrenos diferentes e animações simples – como um cubo girando no ar – bem como scripts com funcionalidades dependentes de ações do usuário, como, por exemplo, um portão que abre ou fecha com o clique do usuário.

Ambientados com a *engine*, a equipe realizou uma série de testes para verificar se o Unity3D teria a capacidade de se comunicar com um servidor externo e

após esta verificação foram efetuados testes simples de integração e troca de dados entre aplicações do Unity3D com *servlets*.

O próximo passo foi a procura de *parsers* para JSON que funcionassem dentro do Unity3D, o que levou um tempo considerável de pesquisa e finalmente foram testadas as possibilidades do projeto poder funcionar dentro de uma página web.

Após a validação e domínio destas funcionalidades o projeto foi iniciado com uma reunião entre a equipe de desenvolvimento e o professor orientador para o levantamento dos requisitos do sistema.

3.6.2 Levantamento de requisitos e casos de uso

O levantamento dos requisitos iniciais foi realizado em reunião com o professor orientador, que atuou na maior parte do tempo como cliente, em vista de que a intenção do simulador é de ser integrado em seu projeto de doutorado (Wandresen, 2010). As funcionalidades básicas do sistema foram descritas e discutidas em reunião, resultando nos casos de uso do sistema.

Os casos de uso foram alterados algumas vezes no decorrer do projeto devido à mudanças que o cliente (professor orientador) desejou. Essas alterações, por mais leves que parecessem, se refletiram em alterações em vários dos artefatos produzidos.

3.6.3 Criação das classes do sistema

O primeiro passo para desenvolvimento do sistema foi a definição das classes presentes no mesmo. Pelos diagramas de classes de análise (ver apêndice), pôde-se definir as relações básicas entre os elementos do simulador e do servidor, assim criando a base para o desenvolvimento do sistema.

As classes presentes no simulador desenvolvidas pela equipe são: Engenheiro, Terreno, Arvore, Colecoes e GUIUtils.

3.6.4 Modelagem

Os elementos que tiveram que ser modelados foram as três ferramentas do inventário do personagem, o anel para seleção da árvore, a esfera delimitadora do terreno, o resto da árvore ao ser cortada, o toco da árvore, e as duas fitas para marcação das árvores medidas. A árvore em si e o personagem controlado são modelos nativos do próprio Unity3D. Como são modelos extremamente complexos, optou-se por utilizá-los no lugar de desenvolver novos modelos. Porém, suas texturas foram alteradas para melhor representarem um engenheiro florestal e uma árvore da espécie *Pinus taeda*.

As ferramentas modeladas foram uma suta, um hipsômetro e uma motosserra. A suta é como uma régua com delimitadores móveis, possuindo um aspecto de “F”, e serve para medir o DAP (diâmetro) das árvores (ver tópico 2.5.1 – DAP). O hipsômetro é como uma pequena câmera, por meio da qual é medida a altura de uma árvore. A motosserra é a popular ferramenta utilizada para corte de objetos – no caso do simulador, para corte da árvore.

O toco e o resto da árvore foram obtidos por meio da importação da árvore do Unity3D para o Google Sketchup e sua subdivisão em duas partes, exportando então dois modelos separados para o Unity3D.

3.6.5 Instanciação de modelos

Uma vez importados os modelos, eles foram instanciados em cena. O modelo do engenheiro é instanciado por meio do script Terreno, e as árvores e esferas delimitadoras são instanciadas no script Coleções (ver Diagrama de Classes do Simulador no apêndice). A partir deste momento, os objetos são instanciados nos scripts Engenheiro e Arvore. As ferramentas são instanciadas no script Engenheiro, e herdam sua posição da mão direita do engenheiro. Já o anel para seleção das árvores e as fitas para sua marcação são instanciados no script Árvore.

3.6.6 Desenvolvimento das funcionalidades

3.6.6.1 Sistema de inventário

A primeira funcionalidade a ser implementada foi um sistema de inventário (ferramentas na mão do personagem). Foi necessário desenvolver uma função para mudar a ferramenta na mão do personagem, para que este possa selecionar uma ferramenta para realizar algum tipo de medida.

3.6.6.2 Seleção de árvores

A segunda funcionalidade a ser desenvolvida foi a de seleção de árvores, para que o usuário possa selecionar a árvore a ser medida.

3.6.6.3 Medidas

Logo em seguida, foram implementados os sistemas de realização de medidas nas árvores. Ao selecionar uma ferramenta e medir uma árvore, variáveis diferentes são atualizadas na árvore em específico, marcando que nela determinadas medidas foram realizadas.

3.6.6.4 Animações

Uma vez realizadas as instanciações dos modelos no simulador e implementadas as funcionalidades para inventário, seleção de árvores e realização de medidas, foram desenvolvidas as animações para realização de cada medida.

O Unity3D traz uma série de animações nativas para o personagem, como a de andar, correr e ficar parado. Foram criadas como animações novas a de medir DAP, medir altura e cortar a árvore.

Cada parte do personagem é tratada como um objeto separado, porém herda de uma parte mais central (por exemplo, a mão herda do braço, o braço herda do tórax, etc.). Para realização das animações, foram definidas rotações para cada parte em dados momentos no tempo, gerando um arquivo de animação que pode ser executado através de comandos via script.

Outro fator crucial para a execução de animações é o posicionamento do personagem. Para isto, foram criados scripts para orientação do engenheiro para a árvore selecionada, bem como de reposicionamento do engenheiro para otimização das animações.

3.6.6.5 Delimitação espacial

A parcela da floresta destinada ao usuário medir deve ser delimitada, para que o usuário não saia e se perca no mapa. Esta delimitação é feita através de esferas azul claras, que cercam a área desejada e impedem que o personagem as atravesse.

O método para instanciação destas esferas foi feito na classe Colecoes. A grande questão no desenvolvimento desse script foi a quantidade dinâmica de linhas e colunas de árvores reais a serem medidas, bem como seu espaçamento também dinâmico, visto que o espaçamento das esferas deve sempre ser o mesmo. Tais condições implicam que o script de instanciação das esferas deve trabalhar diretamente com estas variáveis.

As esferas são instanciadas a um metro de distância umas das outras, em quatro iterações, gerando um retângulo, delimitando a área com árvores a serem medidas. Tendo acesso à quantidade de linhas e colunas de árvores, bem como seu espaçamento em duas dimensões, é instanciada a quantidade correta de esferas e delimitada sempre a área correta.

3.6.6.6 Melhoria do ambiente

Após a implementação de todas as funcionalidades essenciais, ainda houve a necessidade de melhorar o ambiente, para tornar o cenário mais realista. Foram acrescentados leves aclives e declives no terreno, moitas e outros enfeites no terreno, bem como instanciadas árvores fora da zona delimitada por esferas, apenas para enfeite. Porém, estas árvores de enfeite não poderiam se associar ao script Arvore, pois nesta estão implementados scripts de medida, e não pode-se realizar medidas em uma árvore de enfeite. Por tal motivo, os modelos instanciados fora da área delimitada são objetos distintos que apenas se assemelham as árvores alvo das medidas. O script para sua instanciação foi feito na classe Colecoes.

3.6.6.7 Sonoplastia

Uma das últimas funcionalidades a serem implementadas no simulador foram os sons. Há dois sons ambientes (pássaros e vento), bem como sons para passos do personagem na grama, uso da motosserra, a árvore caindo, e a voz do engenheiro ao realizar medidas do DAP ou da altura de uma árvore.

4 APRESENTAÇÃO DO SOFTWARE

Neste capítulo será explicado como instalar o sistema a partir do CD de instalação, haverá o detalhamento com imagens das páginas web do sistema e no âmbito do simulador serão formalmente apresentadas todas as funcionalidades do sistema, em detalhes, bem como algumas imagens do simulador em funcionamento para melhor explicação de determinados recursos. Também será explicado o procedimento necessário para o uso do simulador, a forma de obter posteriormente os dados medidos e, por fim, um detalhamento de como é feita a integração do simulador desenvolvido com a tecnologia Unity3D com um servidor Java.

4.1 INSTALAÇÃO E CONFIGURAÇÃO DO AMBIENTE DE DESENVOLVIMENTO

Tanto a instalação quanto a configuração do ambiente de desenvolvimento será descrita para que seja feita dentro do ambiente Windows, embora a instalação também funcione no ambiente Linux, que não será descrita em virtude da existência de um grande número de distribuições do sistema, inviabilizando uma descrição precisa de instalação para cada caso. Em todo caso, bastará para o usuário de Linux instalar os softwares descritos da forma que sua distribuição permitir e seguir os passos tal como é feito no ambiente Windows.

Antes de mais nada, dois *softwares* são necessários: o servidor de aplicação Apache Tomcat e o sistema gerenciador de banco de dados MySQL. As versões utilizadas são a versão 7.0.47 para o Apache Tomcat e 5.6.14 para o MySQL. Os softwares podem ser baixados nos sites http://tomcat.apache.org/download-70.cgi e http://dev.mysql.com/downloads/. Também é importante ter o kit de desenvolvimento Java (JDK) instalado, este podendo ser baixado no site da Oracle, em http://www.oracle.com/technetwork/pt/java/javase/downloads/jdk7-downloads-1880260.html.

Para a configuração do ambiente de desenvolvimento, dois outros softwares necessitam ser instalados. O primeiro é a IDE Eclipse, disponível em http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/keplersr1 e a *engine* Unity3D, disponível em http://portuguese.unity3d.com/unity/download/. As versões utilizadas são o *Luna Release* e 4.2.0f4 respectivamente.

A instalação do Java JDK, Eclipse e do Unity3D são simples e não necessitam muita explicação, entretanto, ao instalar o Apache Tomcat, para evitar transtornos de configuração, deve-se atentar para definir que a porta padrão do serviço seja a de número 8080. Esta é a porta *default* utilizada pela aplicação, em todo caso. Quanto ao MySQL, o usuário e senha utilizados são a palavra 'root' para ambos, mas é possível alterar a configuração do simulador, como será mostrado mais adiante. É também importante verificar se as portas não estão em uso por outros aplicativos. Em caso afirmativo, recomenda-se encerrar estes processos ao utilizar o simulador ou então mudar o número da porta dos outros aplicativos.
O CD de instalação contém os seguintes itens:

-Uma pasta intitulada Simulador
-Uma pasta intitulada Documentação
-Um arquivo de extensão .war chamado TCCSimuladorFlorestal
-Uma pasta chamada TCCSimuladorFlorestal
-Um arquivo de extensão .sql chamado baseFlorestal
-Um arquivo de extensão .xml chamado crossdomain
-Um arquivo de extensão .pdf chamado leia-me

Primeiro, com o MySQL instalado, é necessário criar uma base de dados chamada florestal. Com isso, a importação da base de dados usada no projeto, que está no arquivo baseFlorestal.sql, deve ser feita para dentro da base recém criada.

Com a base de dados já configurada corretamente, o próximo passo é abrir o arquivo TCCSimuladorFlorestal.war e alterar, caso necessário as configurações da base de dados para o projeto caso a senha e usuário do MySQL tenham nomes diferentes de 'root'. Para isso, utiliza-se um programa tal como o Winrar para abrir o TCCSimuladorFlorestal.war e então ir para o diretório WEB-INF/classes/util. Feito isso, haverá um arquivo chamado BD.properties que poderá ser aberto com qualquer editor de textos. Dentro dele, os valores para usuário e senha podem ser alterados para refletir a configuração atual do MySQL de quem for instalar o sistema. Caso não seja usada senha para o MySQL, basta deixar o campo db.pwd sem nenhum argumento.

A seguir, o *deploy* (instalação) do arquivo TCCSimuladorFlorestal.war deve ser feito no Apache Tomcat. Isto é feito em duas etapas. A primeira é copiar o arquivo para dentro da pasta *webapps* que está dentro de onde o Apache Tomcat foi instalado (por exemplo: C:\Program Files\Apache Software Foundation\Tomcat 7.0). O segundo passo é copiar para a pasta ROOT que está dentro da pasta *webapps* o arquivo crossdomain.xml. Este arquivo é necessário para que o executável gerado pelo Unity3D seja capaz de se comunicar com outros domínios. Feito a cópia dos dois arquivos o servidor Apache Tomcat deve ser parado e inicializado novamente.

Se a instalação for bem sucedida, será possível acessar o sistema digitando o seguinte endereço no navegador web: http://localhost:8080/TCCSimuladorFlorestal/.
A configuração do ambiente de desenvolvimento consiste no uso da IDE Eclipse e da *engine* Unity3D. Com a IDE é possível editar os dados do servidor Java e com o Unity3D os *scripts*, modelos tridimensionais, terreno e etc. Embora a configuração do servidor funcione para Linux, não será possível configurar um ambiente de desenvolvimento para este sistema operacional, pois não há versões do Unity3D disponíveis no momento para tal plataforma.

Com o Eclipse instalado e aberto, o projeto do servidor, que é a pasta TCCSimuladorFlorestal deverá ser importado por meio da opção import dentro do menu file da IDE. Feito isso o projeto aparecerá no Project Explorer do Eclipse. Em seguida, o próximo passo é configurar o Apache Tomcat para funcionar dentro da IDE, isso é feito clicando com o botão direito do mouse sobre o projeto e em seguida em new e em other, onde será possível adicionar um servidor, escolhendo para isso a opção server. Será requisitado o diretório onde o Apache Tomcat está instalado e ao término o servidor estará adicionado ao projeto. Deve-se ter cuidado para parar o servidor caso ele esteja rodando, pois a partir de então o Eclipse passará a gerenciar a inicialização do servidor.

O último passo é abrir o projeto do simulador no Unity3D. Com o Unity3D aberto deve-se ir no menu File e em seguida em new project. Abrirá uma janela e a pasta Simulador do CD de instalação deverá ser selecionada. O Unity3D irá recarregar e após isso deve-se ir em file novamente e então escolher a opção open scene. Na janela que abrirá o arquivo simuladorFlorestalTerreno.unity deverá ser selecionado. Feito isso, a cena será carregada e poderá ser editada. Quando um *script* do simulador for aberto, ele será carregado na IDE MonoDevelop, que é

instalada junto com o Unity3D. Os *scripts* estão dentro da pasta monobehaviour da pasta Simulador no seguinte caminho: Simulador\Assets\Simulador Florestal\scripts.

Alterações feitas dentro do Unity3D para o projeto não irão se refletir no lado do servidor. É necessário construir o executável do Unity3D teclando no ambiente o atalho ctrl+B e escolhendo a forma de construção do projeto para o tipo web. Feito isso o Unity3D compilará um arquivo com extensão .unity dentro de uma pasta e uma página de formato .html que será o contêiner do arquivo. A página .html deve ser descartada, o arquivo .unity deverá ser renomeado para simulador.unity e ele deverá ser sobrescrito ao arquivo de mesmo nome dentro do projeto aberto no Eclipse.

4.2 PÁGINAS WEB

O simulador é executado dentro de um sistema web. Ele é constituído de uma página onde o usuário poderá fazer *login* (acesso ao sistema) ou, caso ainda não tenha, efetuar o seu cadastro.

Após um cadastro ou *login* bem sucedido, o usuário será encaminhado para a página principal do sistema. Nesta página algumas informações gerais são exibidas e é possível navegar para outras páginas ou fazer *logout* (sair do sistema) por meio de um menu.

As páginas acessíveis pelo menu são a página “Créditos’, onde é mostrado o nome da equipe e do professor orientador e a página “Simulador” onde o ambiente virtual é carregado para que o usuário possa fazer as medidas. Nesta página também há um botão para exportar os dados medidos no formato de planilha eletrônica.

Tentativas de acesso ao sistema sem que ocorra uma validação de dados por meio da página de *login* são barradas, havendo encaminhamento para uma página de acesso negado. Em tentativas de acesso a páginas não existentes dentro do sistema também haverá redirecionamento para uma página específica que informará o usuário do erro. Ver figuras na próxima página.

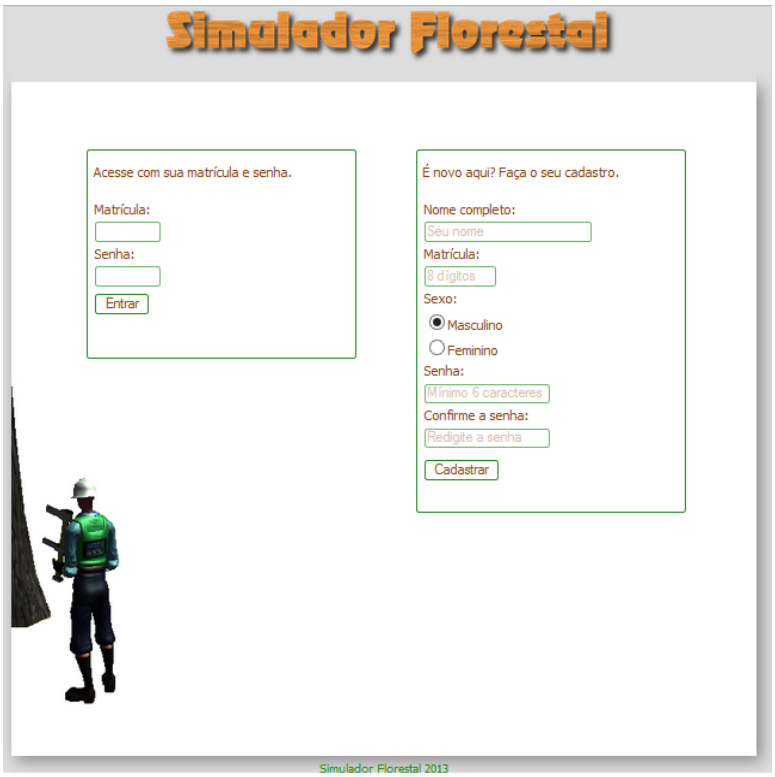

Figura 5: Página de login e cadastro.

Figura 6: Página principal.

Figura 7: Página de créditos.

Figura 8: Página do simulador.

Figura 9: Página de erro 404.

Figura 10: Página de acesso negado.

4.3 USO DO SIMULADOR

O simulador executa via web, por meio de uma página. O usuário – aluno da disciplina de Inventário Florestal – realiza *login* no sistema, caso já seja cadastrado, senão, realiza o cadastro na mesma página de *login* e ao selecionar “Simulador” no menu, logo após ser redirecionado para a página “Principal”, é redirecionado novamente, para a página do simulador onde este é carregado. Dentro do simulador será exibido um menu com informações sobre o ambiente e comandos básicos para controle, e clicando com o mouse no botão “Iniciar”, o menu desaparecerá e o usuário estará a partir de então no controle do engenheiro florestal que poderá realizar as medidas nas árvores da floresta ao seu redor.

Um detalhe importante é que o simulador será executado na página web por meio de um *plugin* que será baixado gratuitamente no site do Unity3D. Este *plugin* é compatível com as plataformas Windows e Mac OS, porém, ainda não há versão para Linux.

4.4 FUNCIONALIDADES DO SIMULADOR

4.4.1 Objetivo

O objetivo do usuário do simulador é realizar medidas necessárias nas árvores em uma região delimitada de uma floresta, de modo a viabilizar a estimativa do volume de madeira da região. As medidas são de três tipos: diâmetro de altura do peito (DAP), que deve ser feita em todas as árvores, altura, que é opcional e cubagem, que envolve o corte da árvore e deve ser feita em pelo menos 5 árvores. A natureza opcional da medida da altura deriva do fato de que há uma variedade de métodos para estimar o volume de madeira da floresta e que nem todos eles utilizam os dados de altura da árvore. Se no passo seguinte à coleta de dados (não contemplado neste trabalho), o aluno de engenharia florestal preferir utilizar métodos envolvendo a altura para os seus cálculos, ele deverá fazer estas medidas, além do DAP e da cubagem das árvores.

O número total de árvores da parcela (região em que um engenheiro florestal tipicamente delimita) é de 50 árvores e quando a quantidade mínima de medidas

forem feitas, os dados poderão ser recuperados no formato de planilha eletrônica para uso posterior.

4.4.2 Árvores

Cada uma das árvores do simulador possui medidas carregadas de um banco de dados, com dados reais (ver tabela arvore do Diagrama Entidade-Relacionamento, no apêndice, página 88). No banco, há um total de 296 árvores e no momento em que um usuário é cadastrado são escolhidas 50 deste total de forma aleatória para constituir a parcela do simulador de onde serão feitas as medidas. A espécie de árvore utilizada é a *Pinus taeda*, espécie bastante cultivada para uso comercial.

4.4.3 Ferramentas de medida

As ferramentas do engenheiro florestal são as seguintes: suta, hipsômetro e motosserra.

A suta é como uma régua, com delimitações perpendiculares móveis, dando uma forma de “F” à ferramenta. É utilizada para medir o DAP (diâmetro da altura do peito) de uma árvore. O hipsômetro é como uma câmera (um dispositivo eletrônico de fácil portabilidade com um visor), e tem a finalidade de medir a altura de uma árvore se utilizando para isso de relações de trigonometria calculadas pelo aparelho. A motosserra é uma ferramenta de maior porte, popularmente utilizada para o corte de materiais. No simulador, ela é utilizada para o corte de árvores da qual derivará os dados de cubagem. Cubagem é um processo onde a árvore é seccionada em diversas partes e as medidas da altura e diâmetro destas partes são medidos.

4.4.4 Comandos

O personagem pode caminhar com as setas direcionais no teclado, bem como com as teclas A-S-D-W – configuração de locomoção altamente popularizada por jogos atuais para computadores. Para circular no inventário, utilizam-se os botões Q e E. Tal configuração é extremamente conveniente, para que o usuário possa trocar facilmente de ferramentas enquanto ele utiliza A-S-D-W para andar.

A tecla “P” pausa a simulação, exibindo ao usuário um menu com os objetivos a serem atingidos e total de medidas realizadas, uma lista com os comandos, e a opção de continuar, que faz o menu desaparecer (podendo também teclar novamente a tecla “P” para mesmo fim). A tecla “C” muda a câmera do jogo, oferecendo ao usuário opções diferentes de visão durante a simulação. A tecla “M” mede uma árvore selecionada com a ferramenta em mãos.

4.4.5 Sistema de inventário

O sistema de inventário do engenheiro o permite circular entre as ferramentas disponíveis (suta, hipsômetro e motosserra) através de duas teclas. O método responsável por esta funcionalidade está na classe Engenheiro.

O engenheiro é instanciado com suas mãos vazias, e quando o usuário apertar a tecla “E”, é instanciada uma suta em sua mão direita. Há uma variável no script Engenheiro controlando qual ferramenta está na mão do engenheiro, assim sendo utilizada como referência para efetuar medidas e animações corretamente. Ao apertar a tecla “E” novamente, a suta é destruída e no seu lugar é instanciado um hipsômetro. A variável indicadora da ferramenta atual é então atualizada. Se o usuário apertar a tecla “Q”, o hipsômetro é destruído e a suta é instanciada novamente; já se for apertada a tecla “E”, aparecerá uma motosserra na sua mão. Basicamente, a tecla “E” incrementa a variável indicadora da ferramenta atual, enquanto a tecla “Q” a decrementa. Obviamente, para garantir o ciclo é necessária uma verificação – se a variável estiver em seu valor máximo (3 – motosserra), e for apertada a tecla “E” para incrementá-la, ela então é zerada e as mãos do engenheiro voltam a estar vazias. O mesmo acontece quando a tecla “Q” é apertada e as mãos do engenheiro estiverem vazias (a variável sendo igual a zero) – a variável é então atualizada para o valor 3 e a motosserra é instanciada.

4.4.6 Seleção de árvores

O método para seleção de árvores está na classe “Árvore”. Quando o usuário clica em uma árvore, é instanciado um anel ao seu redor e uma janela no canto superior esquerdo da tela com as opções para medir a árvore com o instrumento em mãos ou fechar a janela.

4.4.7 Medidas

Duas medidas podem ser obtidas das árvores antes da sua cubagem (uma para cada ferramenta) – o DAP (diâmetro na altura do peito) e a altura. Quando o DAP é medido, ele aparece na janela no canto superior esquerdo da tela, e uma fita vermelha é instanciada na árvore, indicando que seu DAP já foi medido. Já quando a altura é medida, uma fita azul é instanciada na árvore, e a medida também é exibida na janela. A terceira medida que pode ser realizada é a cubagem ou seja, o corte da árvore. Para realizá-la, basta estar com a motosserra em mãos e clicar em "Medir". Uma árvore cubada não pode mais ser selecionada.

4.5 IMAGENS DO SIMULADOR

Ao entrar na página do simulador, este é inicializado. É exibido o menu inicial do simulador, onde o usuário pode verificar algumas informações pessoais, os dados obtidos das árvores e a lista de comandos para controle do modelo de engenheiro florestal.

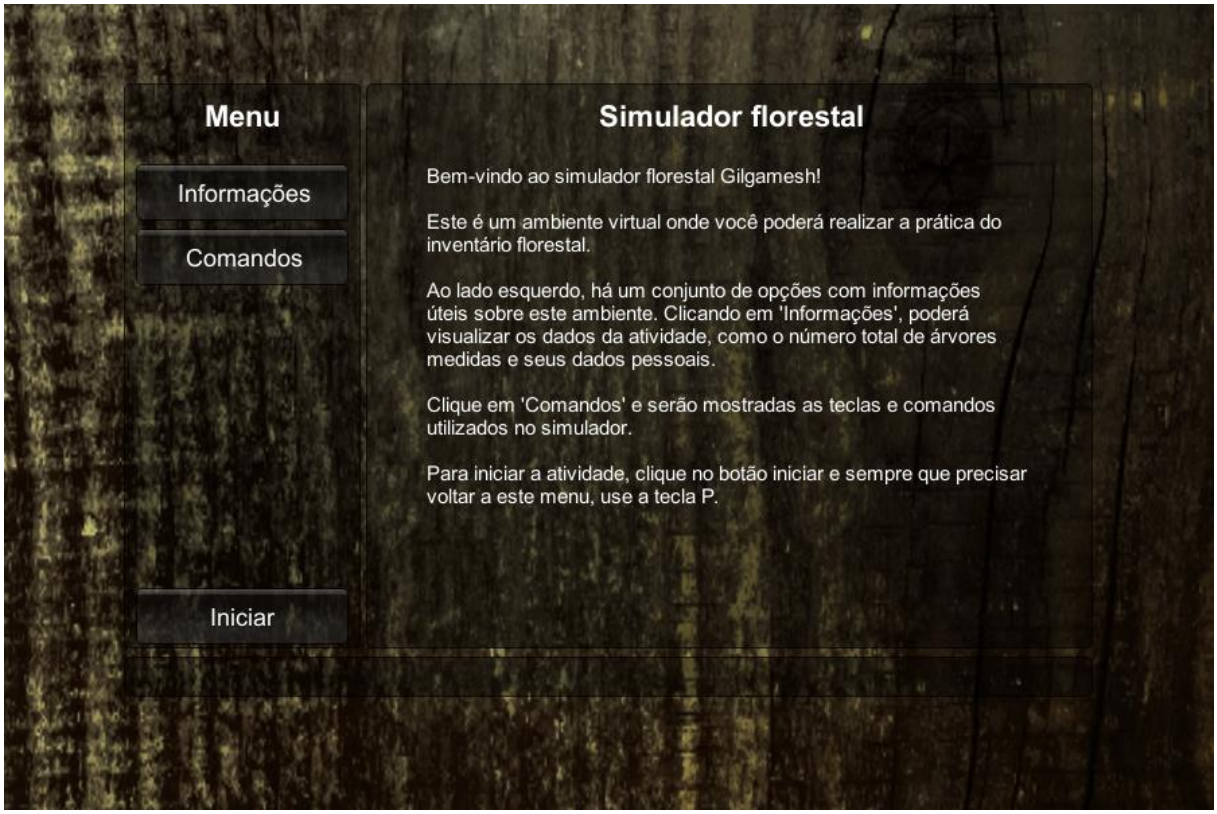

Figura 11: Tela inicial do simulador.

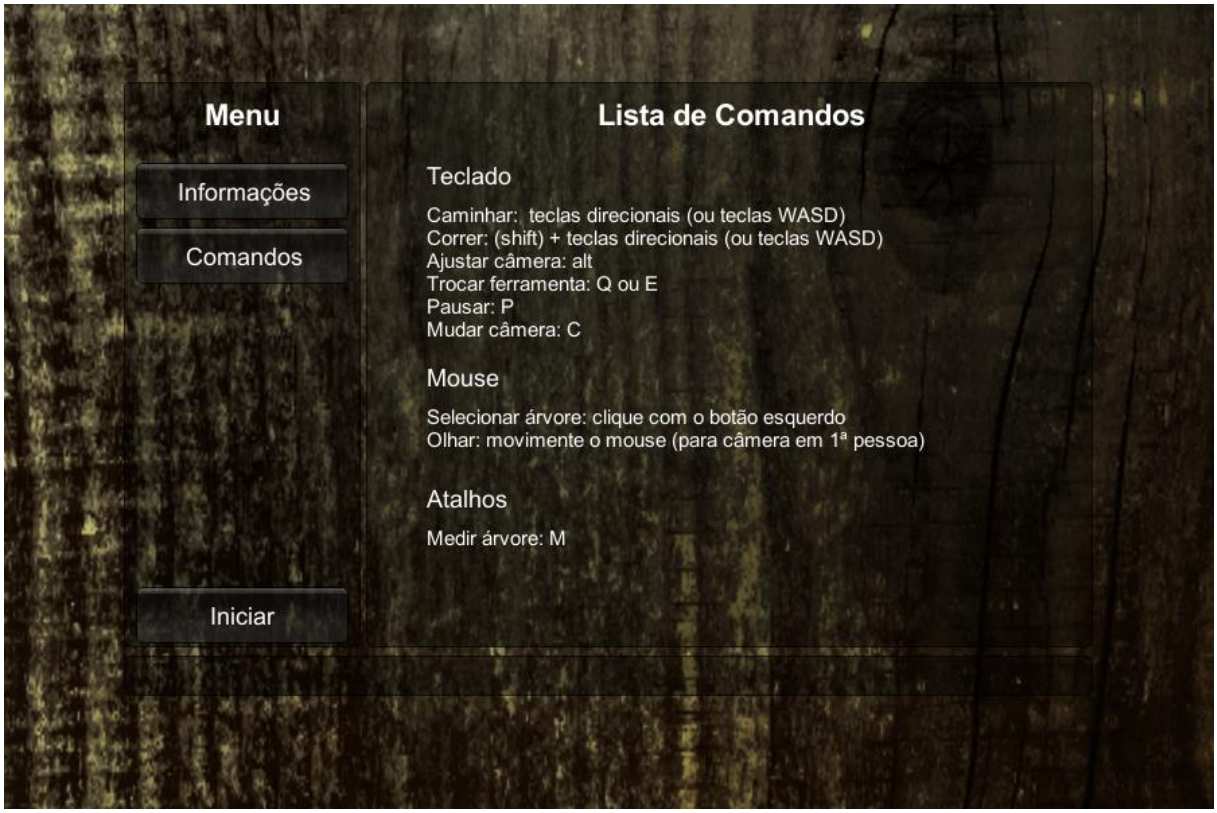

Figura 12: Comandos.

Figura 13: Informações.

O engenheiro é então posicionado onde ele estava na última vez que os dados foram gravados, ou se for o primeiro acesso, ele é posicionado em um local qualquer definido no código do simulador.

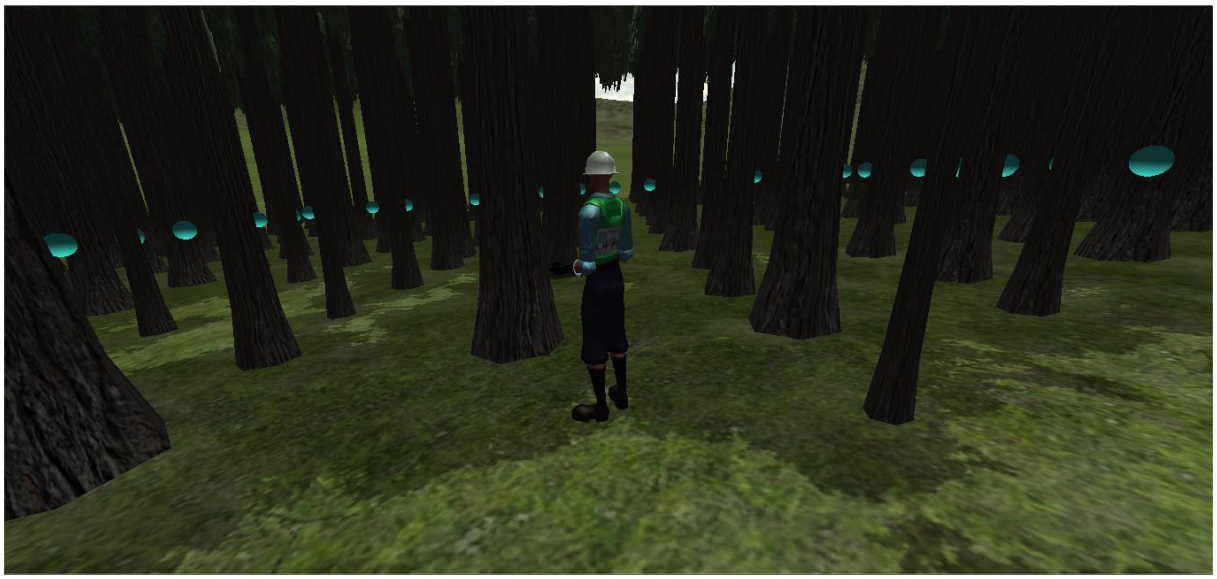

Figura 14: Engenheiro na floresta.

A partir deste momento, o usuário pode locomover o engenheiro pela floresta, para poder selecionar uma árvore e medi-la. Para selecionar uma árvore, basta clicar nela.

Figura 15: Árvore selecionada.

O usuário deve também selecionar uma ferramenta para que as medidas possam ser realizadas. Através das teclas “Q” ou “E”, o usuário circula pelo seu inventário e seleciona uma ferramenta.

Figura 16: Suta em mãos.

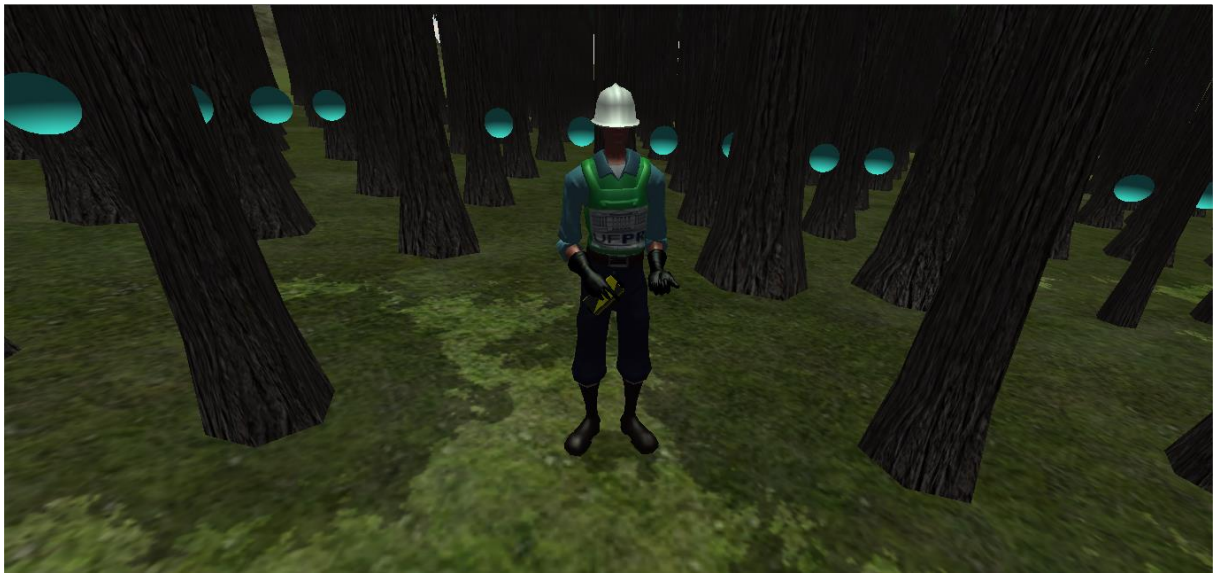

Figura 17: Hipsômetro em mãos.

Figura 18: Motosserra em mãos.

Selecionada a ferramenta, deve-se selecionar uma árvore para extrair a medida dela, para isso deve-se clicar em “Medir”, na janela no canto superior esquerdo da tela, ou mesmo apertar a tecla “M” para realizar uma medida na árvore selecionada.

Cada medida realizada na árvore é indicada com uma fita ao redor da mesma. Uma fita vermelha indica que o DAP da árvore foi medido, enquanto uma fita azul indica que a altura da árvore foi medida.

Figura 19: Medindo DAP da árvore.

Figura 20: Medindo altura da árvore.

Uma vez realizadas as medidas de DAP e altura da árvore, esta pode então ser cubada (cortada). O usuário seleciona a motosserra e clica em “Medir” para cortá-la.

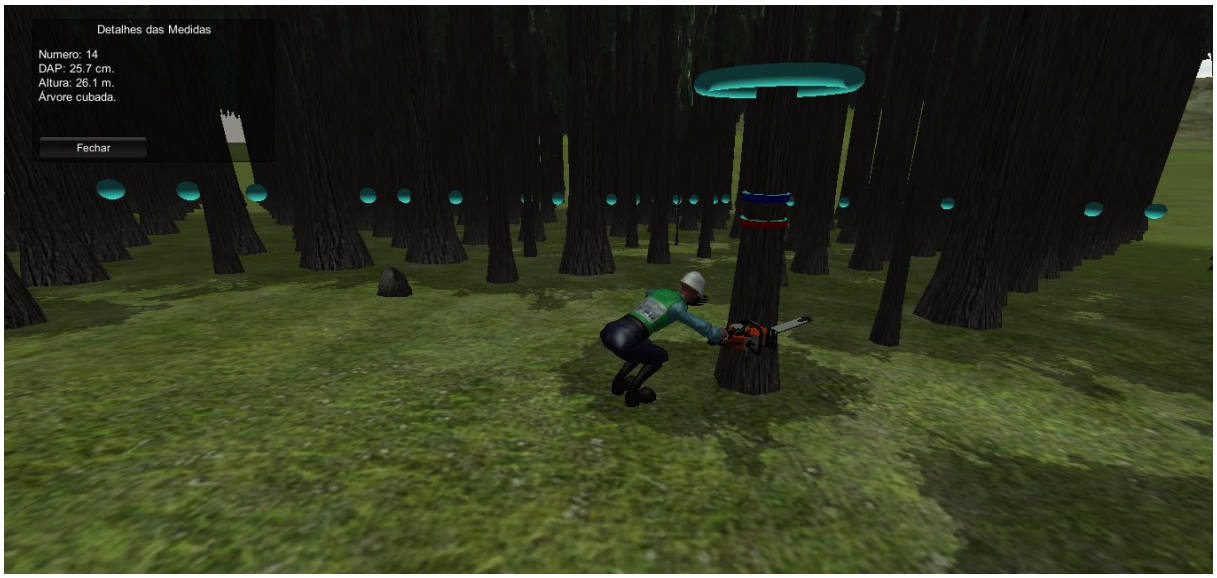

Figura 21: Cortando a árvore.

Figura 22: Árvore caindo.

A área cujas árvores devem ser medidas pelo usuário está delimitada por esferas azul claras, espaçadas de um em um metro. Estas esferas não permitem que o engenheiro florestal saia do espaço determinado.

Figura 23: Esferas delimitadoras.

Há também a opção de alterar a câmera para posições mais convenientes ao usuário, através da tecla “C”. Há cinco modos diferentes para a câmera: modo padrão, modo próximo, câmera em primeira pessoa, câmera em visão superior e câmera fixa, que ficará travada, deixando de acompanhar os movimentos do engenheiro, lembrando uma câmera de segurança. Quando a câmera está em

primeira pessoa, o usuário pode movê-la com o mouse, permitindo que ele olhe em direções diferentes.

Figura 24: Câmera em modo padrão.

Figura 25: Câmera em modo próximo.

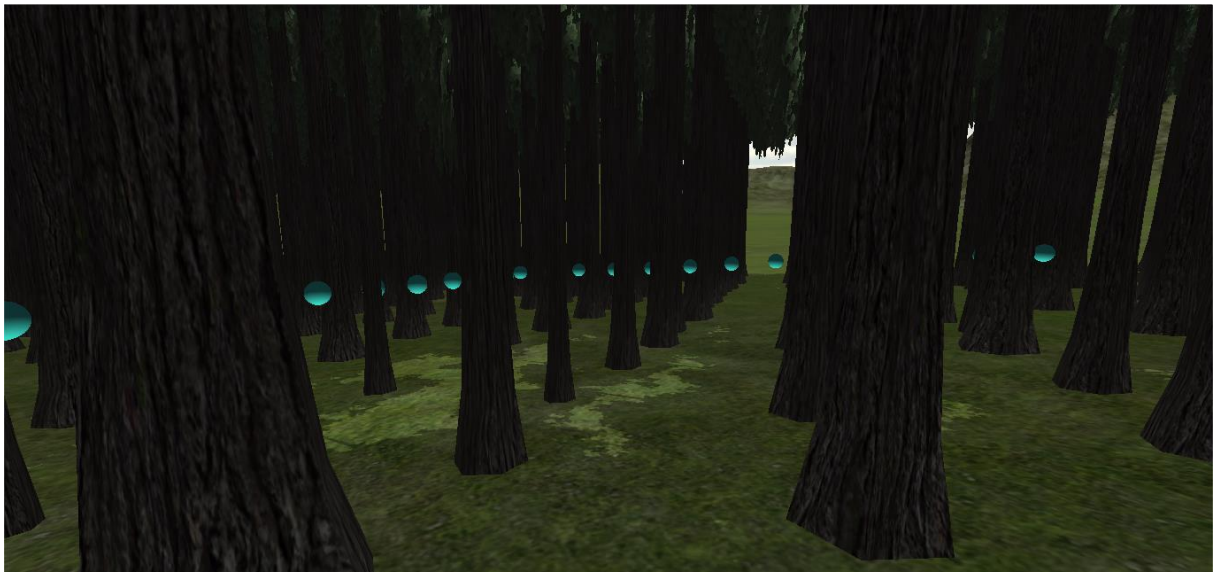

Figura 26: Câmera em primeira pessoa.

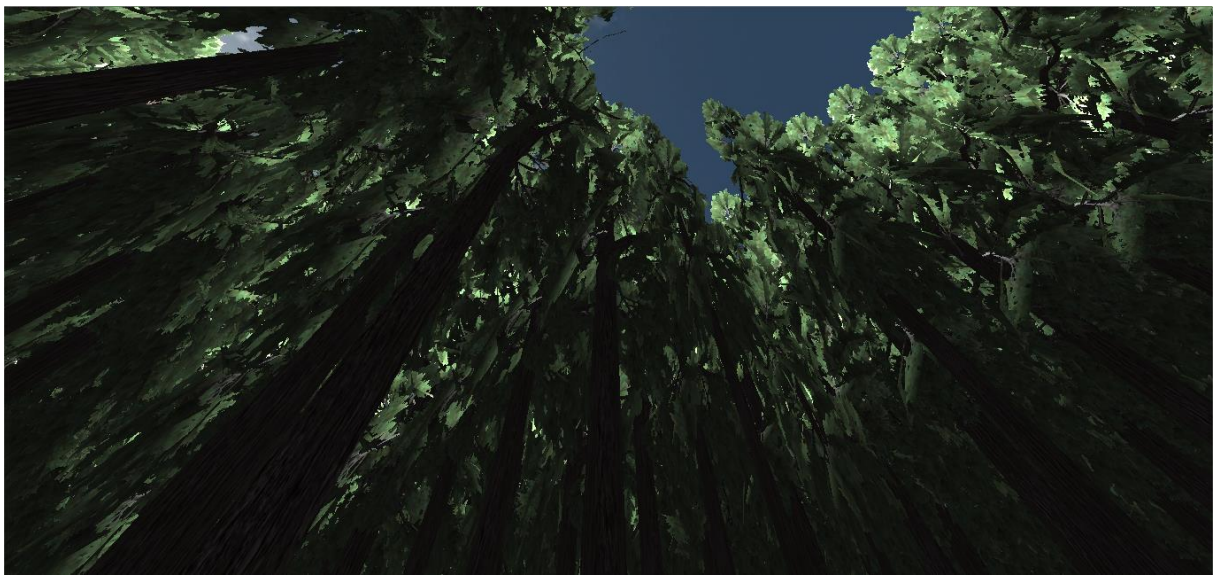

Figura 27: Câmera em primeira pessoa olhando para cima.

Figura 28: Câmera em visão superior.

4.6 EXPORTAÇÃO DOS DADOS DAS MEDIDAS

A partir do momento que o usuário mediu pelo menos o DAP de todas as árvores e cubou 5 árvores, um botão na parte inferior da página web do simulador ficará disponível para que os dados das medidas sejam exportados. Clicando no botão um arquivo será baixado e depois disso, da próxima vez que o usuário entrar no simulador, ele não poderá mais fazer medidas e só terá a opção de reiniciar o processo, o que acarretará na perda de todas as suas medidas feitas e restauração de suas configurações inicias, como posição no terreno e dinheiro.

A planilha baixada será composta por duas abas. A primeira, intitulada ‘Arvores medidas’ trará os dados de DAP e altura de cada uma das árvores. Se a altura de alguma árvore não for medida, obviamente será retornada um campo em branco para a altura. Na segunda aba, intitulada ‘Informações de cubagem’, será exibida as informações de cada uma das seções de cada árvore que foi cortada no processo. O formato do arquivo da planilha é .xls, podendo ser aberto em um software como o Microsoft Excel. Esta opção foi sugestão do professor orientador já que, segundo ele, planilhas eletrônicas são uma das ferramentas mais utilizadas pelos engenheiros florestais.

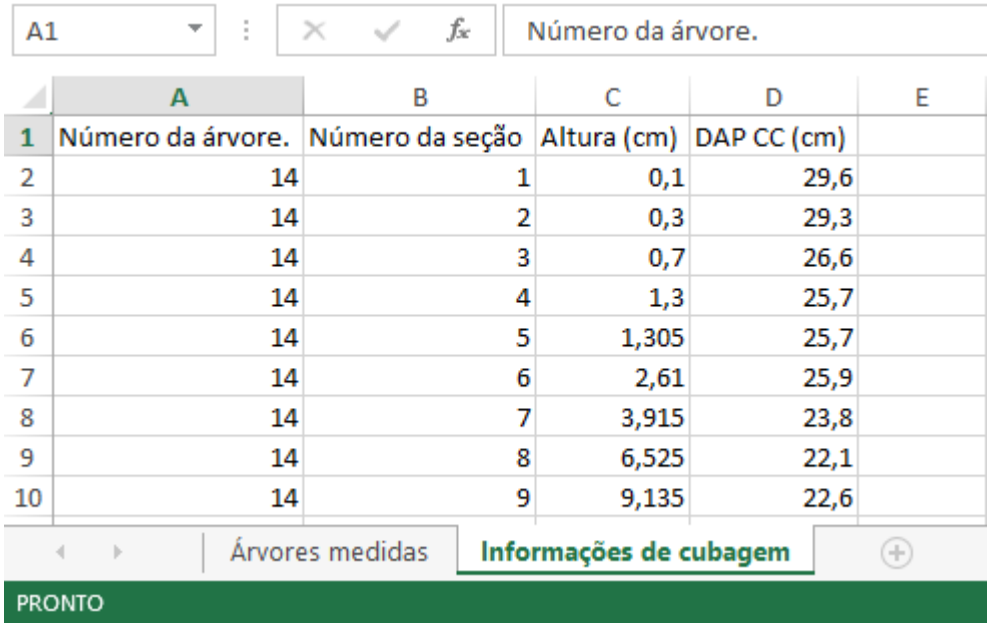

Figura 29: Detalhes da aba ‘Informações de cubagem’ de uma planilha gerada pelo sistema.

4.7 INTEGRAÇÃO DO SIMULADOR COM SERVIDOR

A integração do servidor ocorre quando há a necessidade de ler e gravar informações no banco de dados. O lado do servidor trata essa parte utilizando para isso *servlets* que são classes especiais escritas em Java, que neste caso, respondem à solicitações via protocolo HTTP.

Do lado do simulador há métodos dentro dos *scripts* que fazem conexão com os *servlets* acessando-os por meio de seus endereços HTTP e podendo passar também parâmetros via método POST.

Quando o simulador chama um *servlet*, ele retornará uma string com o conteúdo gerado pelo *servlet*. O conteúdo dependerá do *servlet* e do parâmetro passado a ele. Em qualquer caso, o formato do conteúdo será uma estrutura do tipo JSON descrita no capítulo 3 (seção 3.5.5).

O servidor será acessado quando o simulador é iniciado, solicitando os dados do usuário do sistema, dados estes recuperados por meio de variáveis de sessão presentes no sistema web e também solicitando o conjunto de árvores do aluno. O simulador receberá estas duas strings no formato JSON e fará o *parse* (quebra) da estrutura recuperando os dados passados e os distribuindo para os objetos presentes na cena do simulador, caracterizando cada árvore e também o engenheiro.

Cada vez que o engenheiro florestal faz uma medida de uma árvore, o simulador construirá um JSON com os dados das medidas da árvore e os dados do engenheiro e passará para um *servlet* estes JSONs e então, do lado do servidor, objetos são instanciados para que métodos presentes nestes objetos façam o *parse* dos dados e gravem na base de dados as medidas atualizadas e os dados atuais do engenheiro, como sua posição atual no terreno e sua quantia em dinheiro, gasta a cada medida feita.

Quando um JSON é passado para o lado do servidor, este geralmente responderá com uma mensagem que será recebida ao simulador, dependendo da mensagem haverá a indicação se a transação foi efetuada com sucesso ou não. Desta forma que duas tecnologias diferentes presentes neste sistema se comunicam de forma simples e eficiente.

5 CONSIDERAÇÕES FINAIS

O software obtido ao final deste trabalho foi um simulador tridimensional bastante fiel ao trabalho real executado por engenheiros ao fazerem um inventário florestal. Este sistema foi integrado em um ambiente web e os dados do simulador eram lidos e gravados com o auxílio de um servidor desenvolvido pela equipe.

5.1 DIFICULDADES

Para que o resultado final pudesse ser obtido, foi necessário à equipe que novas tecnologias fossem dominadas pelos membros, principalmente o Unity3D.

No início do projeto praticamente cada nova funcionalidade que precisasse ser desenvolvida foi realizada somente após muita pesquisa em foruns, vídeos e tutoriais, pois a documentação oficial da engine não é tão clara e está longe de ser completa.

Entre as tarefas mais demoradas, que causaram um relativo atraso no andamento do projeto, foram o desenvolvimento de métodos que fizessem o engenheiro florestal instanciar as ferramentas em sua mão, as rotinas para sincronizar as animações e a busca por um parser de json compatível com o Unity3D.

A modelagem dos elementos tridimensionais foi outro desafio, pois foi exigido um aprendizado do zero das técnicas e ferramentas para a construção dos modelos. Também houve uma dificuldade em elaborar a documentação de forma que representasse da melhor forma possível o Unity3D. Em pesquisa por exemplos na Internet, praticamente nada foi encontrado a respeito de UML aplicada à tecnologia e os diagramas foram construídos com base em muita pesquisa dentro de uma documentação tortuosa e testes com o software.

Por último, destaca-se o aprendizado sobre como é difícil atender as reais necessidades de um cliente, que no caso, foi representado pelo professor orientador. Algumas vezes requisitos eram alterados ou então, mesmo que mantidos, era exigido que algum detalhe fosse modificado, o que resultaria em muitas horas de programação. Por conta de alguns atrasos até foi cogitado em

reunião cancelar as animações do engenheiro, porém, arriscando o cronograma as animações foram feitas em tempo e em qualidade aceitável.

5.2 TRABALHOS FUTUROS

Com a versão para este trabalho de conclusão de curso entregue, resta aos membros dar suporte para eventuais falhas que possam vir a apresentar no sistema quando utilizado pelo professor orientador em seu projeto.

Também cogita-se o desenvolvimento de uma modalidade em forma de jogo para o sistema que trará dificuldades maiores para a coleta de dados e também bastante diversão. Irá se tratar de um modo ao estilo survival horror (horror de sobrevivência), inspirado em filmes B de terror. O objetivo seria em fazer todas as medidas enquanto o engenheiro tentaria desviar de ataques de engenheiros zumbis, lobisomens e outras entidades sobrenaturais que fariam esforços consideráveis para arrancarem os recursos e a vida do personagem.

5.3 CONCLUSÃO

Para concluir, os desafios encontrados tornaram o desenvolvimento do sistema bastante motivador e a cada versão parcial do software concluída, com novas funcionalidades e aumento gradual da qualidade, acabaram trazendo satisfação tanto à equipe quanto ao cliente. A experiência adquirida com o Unity3D e com a modelagem de elementos tridimensionais também foram de grande proveito para uma equipe que sempre teve o interesse em conhecer as etapas de desenvolvimento de jogos digitais.

REFERÊNCIAS

AMORY, A. ***Building an Educational Adventure Game****: Theory, Design and Lessons*. Journal of Interactive Learning Research, 2001

QUINN, C. N. *Designing Educational Computer Games*. Amsterdam: Elsevier Science, 1994

ZYDA, M. *From Visual Simulation to Virtual Reality to Games*. Wiley Press, 2005

DURLACH, N. & MAVOR, A. ***Virtual Reality****: Scientific and Technological Challenges*. National Academy Press, 1995

ZYDA, M. & BENNETT, D. *The Last Teacher.* U.S. Dept. of Commerce: 2020 Visions, 2002

MICHAEL, D. & CHEN, S. ***Serious Games****: Games that Educate, Train and Inform*. Boston, MA: Thomson Course Technology, 2006

MASON, R. *Globalizing Education*. London: Routledge, 1998

BECK, R. J. ***Learning Objects****: What?* University of Wisconsin: Milwaukee, 2001

SANQUETTA, C., WATZLAWICK, L., DALLA CÔRTE, A., FERNANDES, L. & SIQUEIRA, J. ***Inventários Florestais****: Planejamento e Execução*. Curitiba: Multi-Graphic Gráfica e Editora, 2009

WANDRESEN, Rafael Romualdo. Ensino de Inventário Florestal por meio de Software Especializado. Projeto de Tese de Doutorado apresentado ao programa de Pós-Graduação em Engenharia Florestal da UFPR. 2010.

MONO: Cross platform, open source .NET development framework. Disponível em: <http://www.mono-project.com/Main_Page>. Acesso em: 18/11/2013.
ORACLE e Java | Tecnologias | Oracle BR. Disponível em: <http://www.oracle.com/br/technologies/java/overview/index.html>. Acesso em: 18/11/2013.

MYSQL - Recursos | O Banco de Dados de Código Aberto Mais Popular. Disponível em <http://www.oracle.com/br/products/mysql/resources/index.html>. Acesso em: 18/11/2013.

MYSQL :: Top 10 Reasons to choose MySQL for Web-Based-Applications. Disponível em <http://www.mysql.com/why-mysql/white-papers/top-10-reasons-to-36 choose-mysql-for-web-based-applications/> . Acesso em: 18/11/2013.

JSON. Disponível em: <http://www.json.org/index.html>. Acesso: 19/11/2013.
ABOUT the Eclipse Foundation. Disponível em: <http://www.eclipse.org/org/>. Acesso: 19/11/2013.

MONODEVELOP - MonoDevelop. Disponível em: <http://monodevelop.com/>. Acesso: 19/11/2013.

APÊNDICE – DOCUMENTAÇÃO DO SOFTWARE

DOCUMENTAÇÃO DO PROJETO

Diagrama WBS (dividido em quatro partes)

WBS – Parte 1 (Gerenciamento e Concepção)

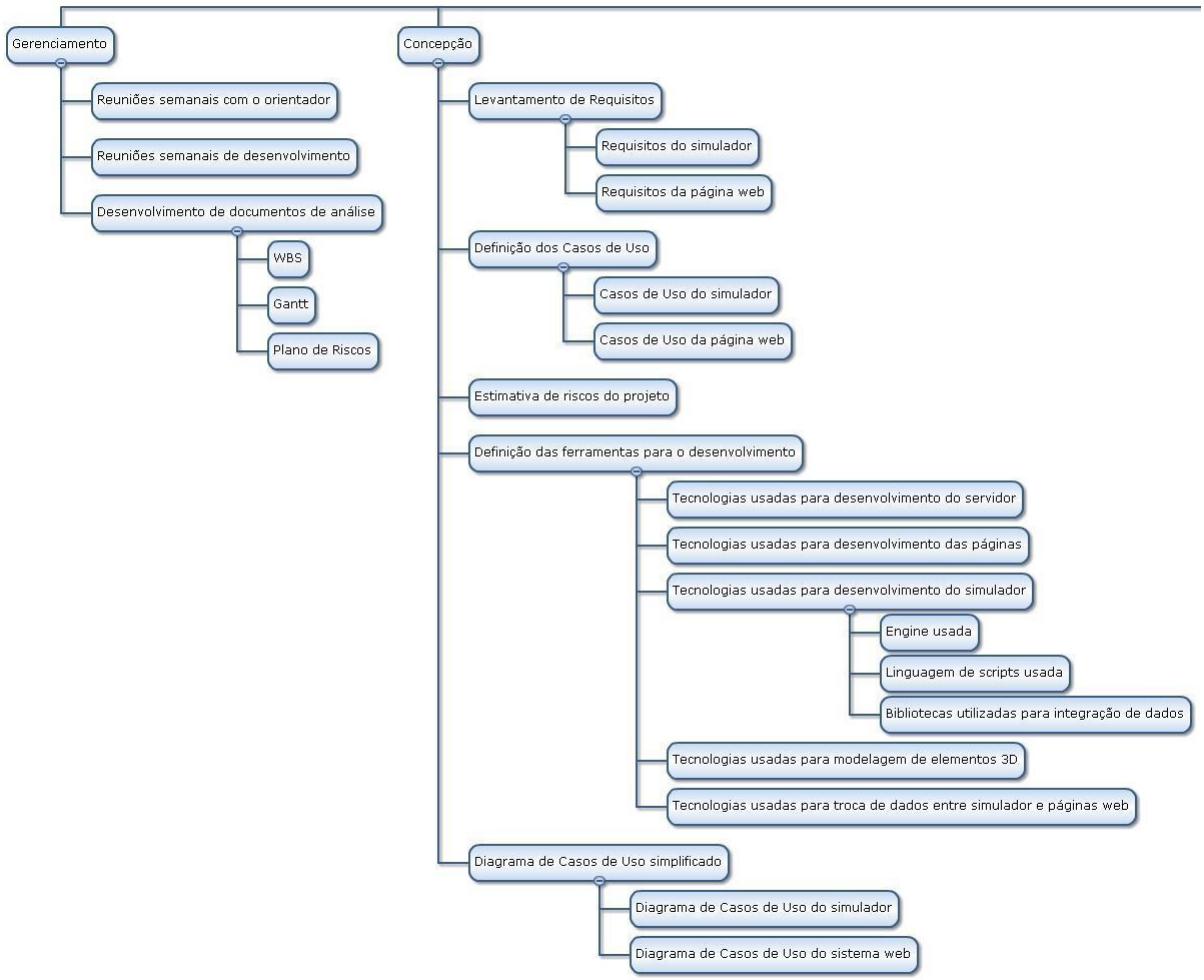

Figura 30: Diagrama WBS (parte 1).

WBS – Parte 2 (Elaboração)

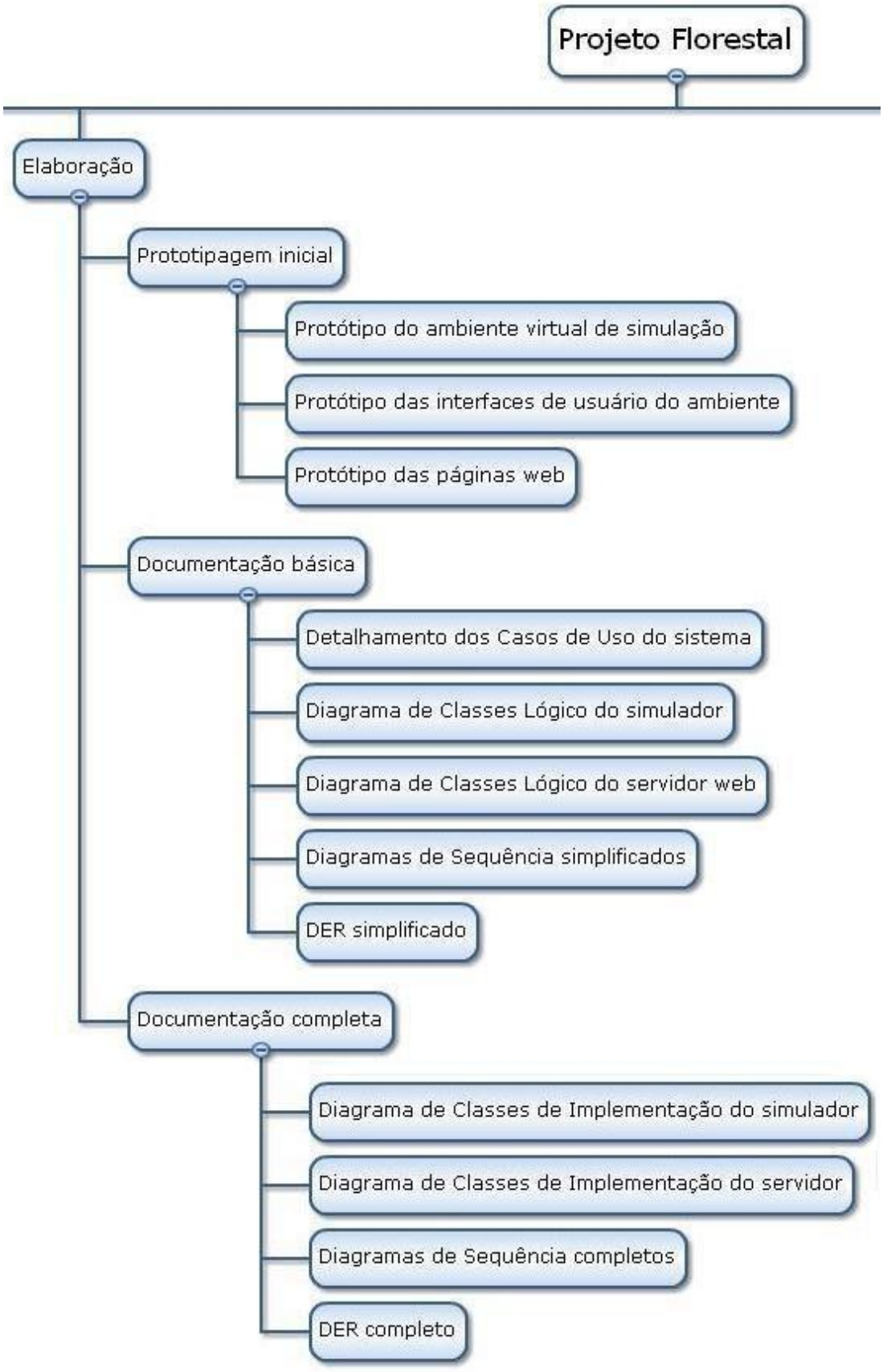

Figura 31: Diagrama WBS (parte 2).

WBS – Parte 3 (Construção)

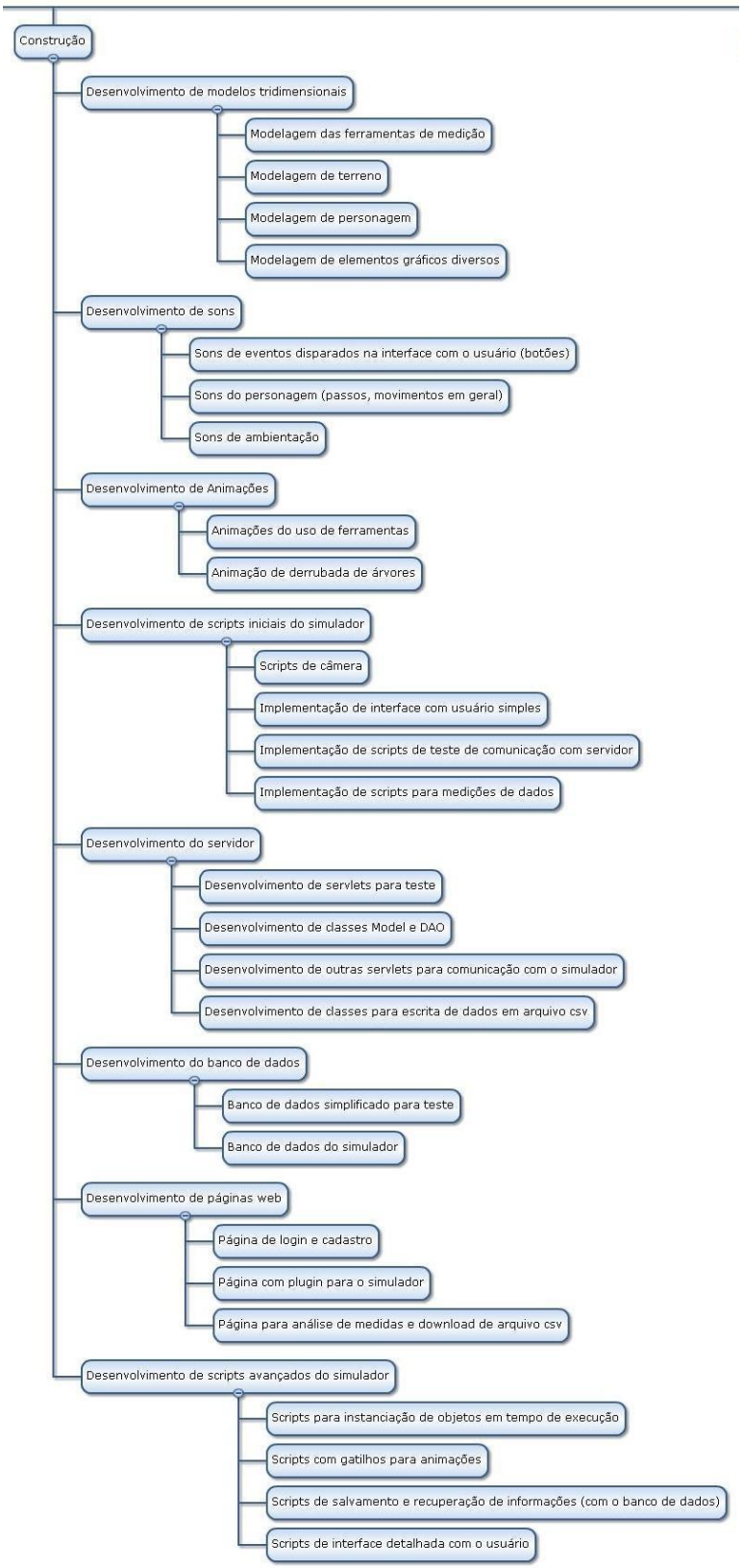

Figura 32: Diagrama WBS (parte 3).

WBS – Parte 4 (Transição)

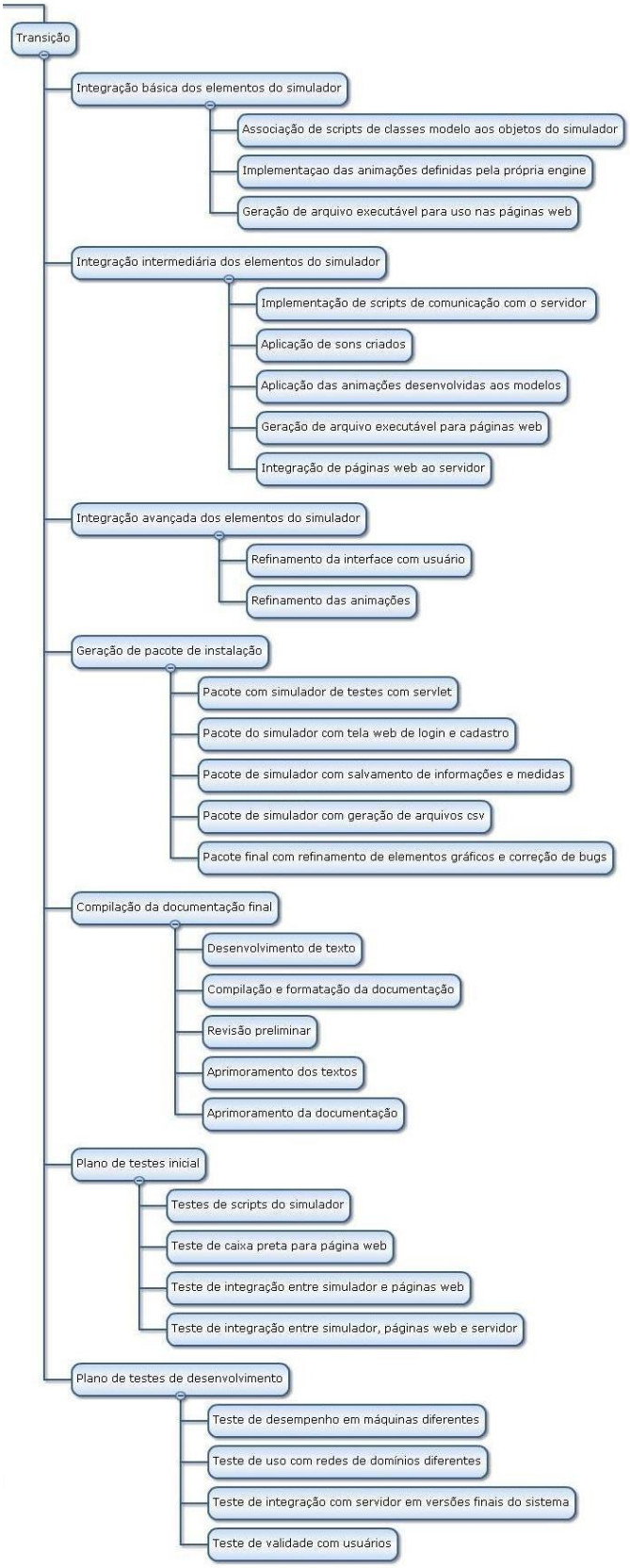

Figura 33: Diagrama WBS (parte 4).

Tabela de Gantt (separada em 3 partes)

Tabela de Gantt – Parte 1 (Concepção e Elaboração)

1		**Projeto Florestal**	180 days	**8/5/13 8:...**	**12/6/13 ...**		
2		**Concepção**	9 days	**8/5/13 8:...**	**8/9/13 1:...**		
3		**Levantamento de requisitos**	1 day	**8/5/13 8:...**	**8/5/13 1:...**		
4		Requisitos do simulador	1 day	8/5/13 8:0...	8/5/13 1:0...		Viktor
5		Requisitos da página web	1 day	8/5/13 8:0...	8/5/13 1:0...		Juliano
6		**Definição dos Casos de Uso**	1 day	**8/6/13 1...**	**8/6/13 5:...**	**3**	
7		Casos de Uso do simulador	1 day	8/6/13 12:...	8/6/13 5:0...		Viktor
8		Casos de Uso da página web	1 day	8/6/13 12:...	8/6/13 5:0...		Juliano
9		Estimativa de riscos do projeto	1 day	8/7/13 8:0...	8/7/13 1:0...	3	Juliano
10		**Definição das ferramentas para o desenvolvimento**	1 day	**8/8/13 8:...**	**8/8/13 1:...**	**3**	
11		Tecnologias usadas para desenvolvimento do servidor	1 day	8/8/13 8:0...	8/8/13 1:0...		Juliano
12		Tecnologias usadas para desenvolvimento das páginas	1 day	8/8/13 8:0...	8/8/13 1:0...		Juliano
13		**Tecnologias usadas para desenvolvimento do simulador**	1 day	**8/8/13 8:...**	**8/8/13 1:...**		
14		Engine usada	1 day	8/8/13 8:0...	8/8/13 1:0...		Viktor
15		Linguagem de scripts usada	1 day	8/8/13 8:0...	8/8/13 1:0...		Viktor
16		Bibliotecas utilizadas para integração de dados	1 day	8/8/13 8:0...	8/8/13 1:0...		Viktor
17		Tecnologias usadas para modelagem de elementos 3D	1 day	8/8/13 8:0...	8/8/13 1:0...		Viktor
18		Tecnologias usadas para troca de dados entre simulador e páginas web	1 day	8/8/13 8:0...	8/8/13 1:0...		Juliano
19		**Diagrama de Casos de Uso simplificado**	1 day	**8/9/13 8:...**	**8/9/13 1:...**	**6**	
20		Diagrama de Casos de Uso do simulador	1 day	8/9/13 8:0...	8/9/13 1:0...		Juliano
21		Diagrama de Casos de Uso do sistema web	1 day	8/9/13 8:0...	8/9/13 1:0...		Juliano
22		**Elaboração**	24 days	**8/12/13 ...**	**8/27/13 ...**	**2**	
23		**Prototipagem inicial**	21 days	**8/12/13 ...**	**8/26/13 ...**		
24		Protótipo do ambiente virtual de simulação	21 days	8/12/13 8:...	8/26/13 1:...		Viktor
25		Protótipo das interfaces de usuário do ambiente	7 days	8/12/13 8:...	8/15/13 1:...		Juliano
26		Protótipo das páginas web	7 days	8/12/13 8:...	8/15/13 1:...		Juliano
27		**Documentação básica**	7 days	**8/12/13 ...**	**8/15/13 ...**		
28		Detalhamento dos Casos de Uso do sistema	1 day	8/12/13 8:...	8/12/13 1:...		Juliano
29		Diagrama de Classes Lógico do simulador	1 day	8/12/13 8:...	8/12/13 1:...		Juliano
30		Diagrama de Classes Lógico do servidor web	1 day	8/12/13 8:...	8/12/13 1:...		Juliano
31		Diagramas de Sequência simplificados	2 days	8/14/13 12...	8/15/13 1:...	29;30	Juliano
32		DER simplificado	1 day	8/13/13 8:...	8/13/13 1:...		Juliano
33		**Documentação completa**	14 days	**8/19/13 ...**	**8/27/13 ...**	**27**	
35		Diagrama de Classes de Implementação do simulador	7 days	8/19/13 8:...	8/22/13 1:...		Juliano
36		Diagrama de Classes de Implementação do servidor	7 days	8/19/13 8:...	8/22/13 1:...		Juliano
37		Diagramas de Sequência completos	4 days	8/26/13 8:...	8/27/13 5:...	35;36	Juliano
38		DER completo	3 days	8/19/13 8:...	8/20/13 1:...		Juliano

Figura 34: Tabela de Gantt (parte 1).

Tabela de Gantt – Parte 2 (Construção)

39		**Construção**	77 days	**9/2/13 8:...**	**10/24/13...**	**22**	
40		**Desenvolvimento de modelos tridimensionais**	14 days	**9/2/13 8:...**	**9/10/13 ...**		
41		Modelagem das ferramentas de medição	7 days	9/2/13 8:0...	9/5/13 1:0...		Viktor
42		Modelagem de terreno	3 days	9/2/13 8:0...	9/3/13 1:0...		Viktor
43		Modelagem de personagem	1 day	9/2/13 8:0...	9/2/13 1:0...		Viktor
44		Modelagem de elementos gráficos diversos	8 days	9/5/13 8:0...	9/10/13 5:...	42	Viktor
45		**Desenvolvimento de sons**	7 days	**9/23/13 ...**	**9/26/13 ...**	**40**	
46		Sons de eventos disparados na interface com o usuário (botões)	7 days	9/23/13 8:...	9/26/13 1:...		Viktor
47		Sons do personagem (passos, movimentos em geral)	7 days	9/23/13 8:...	9/26/13 1:...		Viktor
48		Sons de ambientação	7 days	9/23/13 8:...	9/26/13 1:...		Viktor
49		**Desenvolvimento de animações**	14 days	**10/15/13...**	**10/24/13...**	**40**	
50		Animações do uso de ferramentas	7 days	10/15/13 1...	10/18/13 5...		Viktor
51		Animação de derrubada de árvores	7 days	10/21/13 8...	10/24/13 1...	50	Viktor
52		**Desenvolvimento de scripts iniciais do simulador**	7 days	**9/2/13 8:...**	**9/5/13 1:...**	**40**	
53		Scripts de câmera	6 days	9/2/13 12:...	9/5/13 1:0...		Juliano
54		Implementação de interface com usuário simples	6 days	9/2/13 8:0...	9/4/13 5:0...		Juliano
55		Implementação de scripts de teste de comunicação com servidor	6 days	9/2/13 12:...	9/5/13 1:0...		Juliano
56		Implementação de scripts para medições de dados	6 days	9/2/13 12:...	9/5/13 1:0...		Juliano
57		**Desenvolvimento do servidor**	14 days	**9/9/13 8:...**	**9/17/13 ...**		
58		Desenvolvimento de servlets para teste	14 days	9/9/13 8:0...	9/17/13 5:...		Juliano
59		Desenvolvimento de classes Model e DAO	14 days	9/9/13 8:0...	9/17/13 5:...		Juliano
60		Desenvolvimento de outras servlets para comunicação com o simulador	14 days	9/9/13 8:0...	9/17/13 5:...		Juliano
61		Desenvolvimento de classes para escrita de dados em arquivo csv	7 days	9/9/13 12:...	9/12/13 5:...	56	Juliano
62		**Desenvolvimento do banco de dados**	7 days	**9/23/13 ...**	**9/26/13 ...**		
63		Banco de dados simplificado para teste	2 days	9/23/13 12...	9/24/13 1:...		Juliano
64		Banco de dados do simulador	3 days	9/25/13 12...	9/26/13 5:...	63	Juliano
65		**Desenvolvimento de páginas web**	7 days	**9/30/13 ...**	**10/3/13 ...**		
66		Página de login e cadastro	7 days	9/30/13 12...	10/3/13 5:...	62	Juliano
67		Página com plugin para o simulador	7 days	9/30/13 12...	10/3/13 5:...		Juliano
68		Página para análise de medidas e download de arquivo csv	7 days	9/30/13 12...	10/3/13 5:...	61	Juliano
69		**Desenvolvimento de scripts avançados do simulador**	21 days	**10/10/13...**	**10/24/13...**	**52**	
70		Scripts para instanciação de objetos em tempo de execução	21 days	10/10/13 8...	10/24/13 1...		Viktor
71		Scripts com gatilhos para animações	21 days	10/10/13 8...	10/24/13 1...		Viktor
72		Scripts de salvamento e recuperação de informações	21 days	10/10/13 8...	10/24/13 1...	62	Juliano
73		Scripts de interface detalhada com o usuário	21 days	10/10/13 8...	10/24/13 1...		Juliano

Figura 35: Tabela de Gantt (parte 2).

Tabela de Gantt – Parte 3 (Transição e Gerenciamento)

74		**Transição**	50 days	**10/28/13...**	**12/2/13 ...**	**39**	
75		**Integração básica dos elementos do simulador**	3 days	**10/28/13...**	**10/29/13...**		
76		Associação de scripts de classes modelo aos objetos do simulador	1 day	10/28/13 1...	10/28/13 5...		Juliano
77		Implementaçao das animações definidas pela própria engine	1 day	10/29/13 8...	10/29/13 1...	76	Viktor
78		Geração de arquivo executável para uso nas páginas web	1 day	10/29/13 1...	10/29/13 5...	77	Juliano
79		**Integração intermediária dos elementos do simulador**	2 days	**11/1/13 ...**	**11/1/13 ...**	**75**	
80		Implementação de scripts de comunicação com o servidor	1 day	11/1/13 8:...	11/1/13 12...		Juliano
81		Aplicação de sons criados	1 day	11/1/13 8:...	11/1/13 12...		Viktor
82		Aplicação das animações desenvolvidas aos modelos	1 day	11/1/13 8:...	11/1/13 12...		Viktor
83		Geração de arquivo executável para páginas web	1 day	11/1/13 12...	11/1/13 5:...	80;81;82	Juliano
84		Integração de páginas web ao servidor	2 days	11/1/13 8:...	11/1/13 5:...	83	Juliano
85		**Integração avançada dos elementos do simulador**	7 days	**11/4/13 ...**	**11/7/13 ...**	**79**	
86		Refinamento da interface com usuário	7 days	11/4/13 12...	11/7/13 5:...		Juliano
87		Refinamento das animações	7 days	11/4/13 12...	11/7/13 5:...		Viktor
88		**Geração de pacote de instalação**	7 days	**11/11/13...**	**11/14/13...**	**85**	
89		Pacote com simulador de testes com servlet	2 days	11/11/13 1...	11/12/13 1...		Juliano
90		Pacote do simulador com tela web de login e cadastro	1 day	11/12/13 1...	11/12/13 5...	89	Juliano;Viktor
91		Pacote de simulador com salvamento de informações e medidas	1 day	11/13/13 8...	11/13/13 1...	90	Juliano;Viktor
92		Pacote de simulador com geração de arquivos csv	1 day	11/14/13 1...	11/14/13 5...	91	Juliano;Viktor
93		Pacote final com refinamento de elementos gráficos e correção de bugs	2 days	11/14/13 8...	11/14/13 5...	92	Juliano;Viktor
94		**Compilação da documentação final**	7 days	**11/18/13...**	**11/21/13...**		
95		Desenvolvimento de texto	3 days	11/18/13 8...	11/19/13 1...		Viktor
96		Compilação e formatação da documentação	3 days	11/18/13 8...	11/19/13 1...		Juliano
97		Revisão preliminar	1 day	11/20/13 1...	11/20/13 5...	95;96	Juliano;Viktor
98		Aprimoramento dos textos	3 days	11/20/13 8...	11/21/13 1...	97	Viktor
99		Aprimoramento da documentação	3 days	11/20/13 8...	11/21/13 1...	97	Juliano
00		**Plano de testes inicial**	3 days	**11/25/13...**	**11/26/13...**	**88**	
01		Testes de scripts do simulador	1 day	11/25/13 1...	11/25/13 5...		Viktor
02		Teste de caixa preta para página web	1 day	11/25/13 1...	11/25/13 5...		Juliano
03		Teste de integração entre simulador e páginas web	1 day	11/26/13 8...	11/26/13 1...	101;102	Juliano
04		Teste de integração entre simulador, páginas web e servidor	1 day	11/26/13 1...	11/26/13 5...	103	Juliano
05		**Plano de testes de desenvolvimento**	4 days	**11/28/13...**	**12/2/13 ...**	**100**	
06		Teste de desempenho em máquinas diferentes	4 days	11/28/13 1...	12/2/13 1:...		Juliano;Viktor
07		Teste de uso com redes de domínios diferentes	4 days	11/28/13 1...	12/2/13 1:...		Juliano;Viktor
08		Teste de integração com servidor em versões finais do sistema	4 days	11/28/13 1...	12/2/13 1:...		Juliano;Viktor
09		Teste de validade com usuários	4 days	11/28/13 1...	12/2/13 1:...		Juliano;Viktor
10		**Gerenciamento**	180 days	**8/5/13 8:...**	**12/6/13 ...**		
11		Reuniões semanais com o orientador	180 days	8/5/13 8:0...	12/6/13 5:...		Juliano;Viktor
12		Reuniões semanais de desenvolvimento	179 days	8/5/13 12:...	12/6/13 5:...		Juliano;Viktor
13		**Desenvolvimento de documentos de análise**	1 day	**8/5/13 8:...**	**8/5/13 1:...**		
14		WBS	1 day	8/5/13 8:0...	8/5/13 1:0...		Viktor
15		Gantt	1 day	8/5/13 8:0...	8/5/13 1:0...		Viktor
16		Plano de Riscos	1 day	8/5/13 8:0...	8/5/13 1:0...		Viktor

Figura 36: Tabela de Gantt (parte 3).

DIAGRAMAS UML

Abaixo está descrita a documentação do Simulador Florestal por meio de diagramas UML e diagrama entidade e relacionamento para a base de dados. Os diagramas foram separados em duas categorias: diagramas de análise, derivados a partir do levantamento de requisitos e diagramas de implementação, que são a representação de como o software foi desenvolvido. No que tange aos diagramas de implementação também foram inseridos variados diagramas feitos de acordo com a documentação do Unity3D de forma a situar o leitor sobre como os objetos do simulador são criados.

Também foram anexados a este apêndice as consultas realizadas à base de dados e as estruturas JSON utilizadas para comunicação do Unity3D com o servidor Java.

Diagramas de Análise

De forma a manter a organização e a legibilidade, o diagrama de casos de uso foi dividido em duas partes. A primeira representa em geral as funcionalidades da tela do sistema web e a segunda parte as funcionalidades do simulador desenvolvido no Unity3D. Como salientado nas pré-condições dos casos de uso da segunda parte, as funcionalidades do simulador estendem a partir do caso de uso UC05 - Entrar no simulador. Seguem as imagens na página seguinte.

Diagramas de Casos de Uso

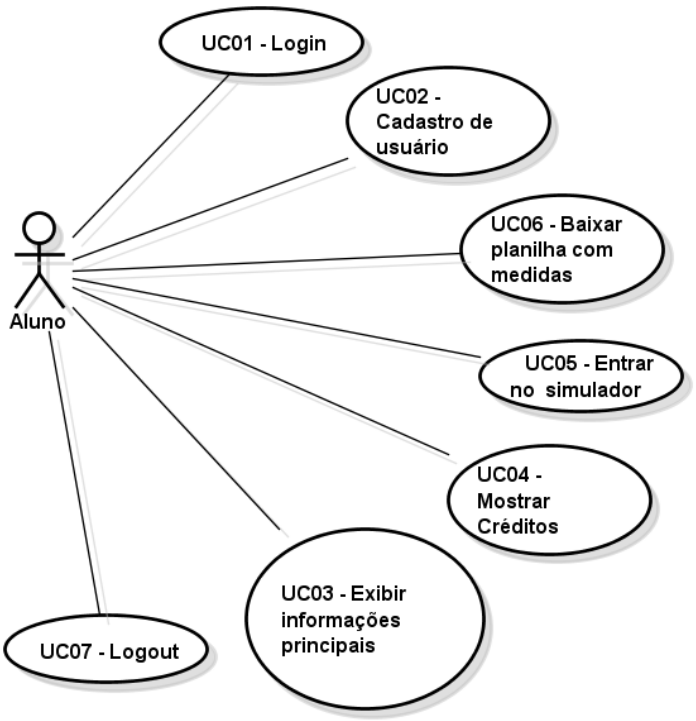

Figura 37: Diagrama de Casos de Uso do sistema web.

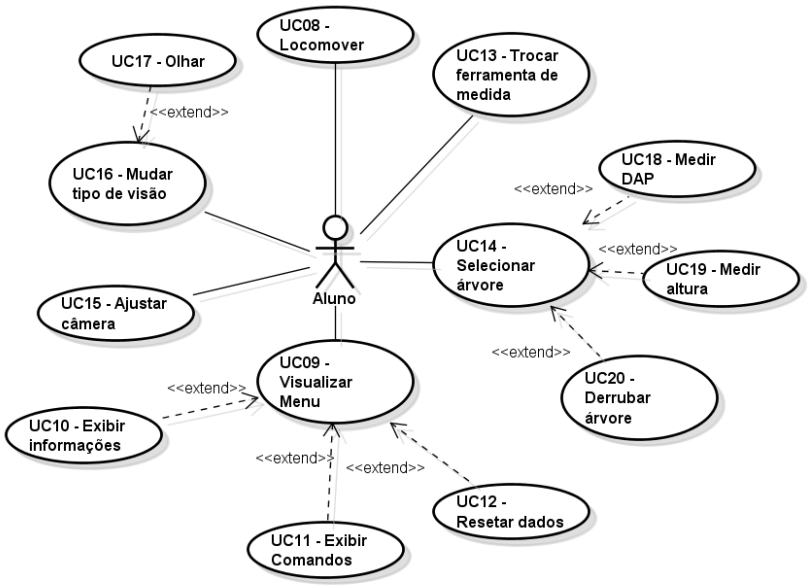

Figura 38: Diagrama de Casos de Uso do simulador.

Descrição dos Casos de Uso

UC01 – Login

Descrição: Trata da autenticação do usuário no sistema. É feito com a entrada e a validação dos dados da matrícula e da senha previamente cadastrados pelo aluno.

Pré-condições: O caso de uso UC02 (Cadastro de usuário) deve ter sido executado corretamente uma vez.

Pós-condições: Após o login o usuário é encaminhado para a página principal do sistema - UC03 (Exibir informações principais) e os seus dados serão acessíveis em qualquer página enquanto não executar o caso de uso UC07 (Logout) .

Cenário principal: O aluno acessa a página inicial (index) e então digita no campo matrícula do formulário de acesso ao sistema um valor composto por 8 dígitos, em seguida ele digita a sua senha no campo senha, composta de pelo menos 6 caracteres. Por fim o aluno clica no botão entrar.

Cenários alternativos

-Se o login não existir ou se a senha estiver incorreta, quando o aluno clicar no botão entrar é retornada na própria página uma mensagem o informando que o login ou a senha estão incorretos.
-Se um dos campos a serem preenchidos estiver em branco, ao clicar no botão entrar é retornada na própria página uma mensagem informando que todos os campos devem ser preenchidos.
-Se o aluno tentar acessar a página principal (executar UC03) digitando o endereço da página no navegador é aberta uma página o informando que o acesso foi negado e que é preciso fazer o login ou se cadastrar, e nesta tela há um link para a página inicial.

UC02 – Cadastro

Descrição: O aluno cadastra os seus dados por meio de um formulário para que possa começar a usar o sistema.

Pré-condições: Nenhuma.

Pós-condições: Após o login o usuário é encaminhado para a página principal do sistema - UC03 (Exibir informações principais) e os seus dados serão acessíveis em qualquer página enquanto não executar o caso de uso UC07 (Logout). São sorteadas 50 árvores do banco de dados que irão compor a floresta ao qual o aluno fará as medidas no simulador.

Cenário principal: Na página inicial do sistema (index), o aluno digita no formulário de cadastro o seu nome, de no máximo 60 caracteres, seu número de matrícula, composto por 8 dígitos, (ignorando os caracteres 'GRR' iniciais do padrão de matrícula da UFPR, embora estes sejam concatenados quando os dados forem persistidos) escolhe o sexo por meio de dois radio buttons (masculino ou feminino), digita uma senha de no mínimo 6 e no máximo 18 caracteres e confirma a escolha digitando a senha novamente no campo para confirmação.

Por último o aluno clica no botão cadastrar. O sistema então se encarregará de sortear cinquenta árvores aleatórias que constam na base de dados e que formarão a floresta a ser utilizada no simulador.

Cenários alternativos

-Caso o aluno deixe pelo menos um dos campos em branco na página de matrícula, ao clicar no botão cadastrar a página exibirá uma mensagem informando que todos os campos devem ser digitados.
-Caso a matrícula digitada seja um número já cadastrado anteriormente, ao clicar no botão cadastrar, uma mensagem é mostrada na mesma página informando que já há um cadastro com aquele número, solicitando outro diferente.

-Caso a matrícula não tenha 8 caracteres numéricos, é mostrada uma mensagem na mesma página o informando que a matrícula deve ser numérica e constituída de 8 caracteres numéricos.
-Caso a senha criada pelo usuário seja diferente do que a inserida no campo de confirmação de senha, ao clicar no botão cadastrar é informado em uma mensagem na própria página que ambas as senhas devem ser iguais.
-Caso a senha criada tenha menos que seis caracteres, ao clicar em cadastrar aparecerá uma mensagem afirmando que a senha deve ter pelo menos seis caracteres.
-Caso o banco de dados não esteja funcionando, ao clicar em cadastrar será mostrada uma mensagem informando da indisponibilidade do serviço.

UC03 - Informações principais

Descrição: Trata-se da página principal do sistema após o aluno ter feito login ou se cadastrado. Apresenta informações básicas sobre o sistema bem como menus de navegação.

Pré-condições: O caso de uso UC01 (Login) ou UC02 (Cadastro de usuário) deve ter sido executado.

Pós-condições: Não há.

Cenário principal: Ao fazer login, cadastro ou clicando em 'Principal' no menu, é exibida uma página contendo informações gerais sobre o sistema, informando o usuário sobre do que se trata o sistema e como é feito o acesso ao simulador.

Cenários alternativos

-Caso o usuário não tenha uma sessão válida e tente acessar essa página de alguma forma, o sistema carregará uma página informando que o acesso foi negado.

UC04 - Mostrar Créditos

Descrição: Uma página informando o nome dos desenvolvedores e do professor orientador.

Pré-condições: O caso de uso UC01 (Login) ou UC02 (Cadastro de usuário) deve ter sido executado.

Pós-condições: Não há.

Cenário principal: Clicando em 'Créditos' no menu principal das páginas do sistema é exibida uma página de créditos, que informa a natureza do projeto e a lista dos responsáveis por ele, que são a equipe de desenvolvimento do trabalho de conclusão de curso e o professor orientador.

Cenários alternativos

-Caso o usuário não tenha uma sessão válida e tente acessar essa página de alguma forma, o sistema carregará uma página informando que o acesso foi negado.

UC05 - Entrar no simulador

Descrição: Este caso de uso aborda o carregamento do ambiente de simulação, que ocorre dentro de uma página do sistema.

Pré-condições: O caso de uso UC01 (Login) ou UC02 (Cadastro de usuário) deve ter sido executado.

Pós-condições: Não há.

Cenário principal: Ao clicar em 'Simulador' no menu, é aberta a página do simulador, onde o ambiente virtual é carregado. Dentro do ambiente, depois que ele é carregado, é exibida uma janela de boas vindas com um menu (UC09) contendo as opções: informações (UC10), comandos (UC11) e iniciar. A interface é a mesma do

menu normal do simulador (UC09), excetuando a existência do botão ‘Iniciar’ no lugar do botão ‘Continuar’ . Clicando em iniciar o aluno deve passar a ter controle de um modelo tridimensional de engenheiro florestal em uma floresta. A mata deve ser composta por um terreno poligonal que representará uma parcela (área delimitada para que sejam feitas as medidas). Na parcela são carregadas as árvores dispostas de forma a conter 5 árvores em cada uma das 10 colunas. O espaçamento das árvores é de 2 metros entre cada coluna e 3 metros entre cada linha. As árvores detectam colisões e respondem a elas, de forma que o engenheiro florestal não a atravesse como se fosse um fantasma.

Ao redor das árvores, para delimitar a região, esferas com detectores de colisão são colocadas para impedir que o engenheiro florestal saia da região das árvores com medidas. Finalmente, do lado exterior da parcela, além das esferas, são carregadas algumas fileiras de árvores que não possuem comportamento algum, servindo apenas para preencher a floresta e dar uma noção de continuidade.

Abaixo da janela do simulador é exibido um texto informando o procedimento de uso do simulador, com os comandos essenciais para uso do ambiente e abaixo do texto há um botão para download das amostras. O botão por padrão é desabilitado e muda o seu estado para habilitado apenas com o simulador completamente carregado na página e quando o usuário atingiu a meta mínima de medidas determinadas.

Cenários alternativos

-Se o *plugin* Unity Web Player não estiver instalado no browser, aparecerá um ícone padrão do Unity para a página de download do *plugin*, que é externa ao sistema. O *plugin* uma vez instalado, funcionará bastando fazer o recarregamento da página.

-Caso o aluno não tenha uma sessão válida e tente acessar essa página de alguma forma, o sistema carregará uma página informando que o acesso foi negado.

-Caso o aluno já tenha feito o download da planilha o simulador após seu carregamento informará que ele não poderá mais fazer medidas. Será mostrada uma opção para resetar os dados (UC12), caso o aluno queira repetir o processo das medições.

UC06 - Baixar planilha com medidas

Descrição: As medidas feitas no simulador são exportados para uma planilha eletrônica e acessíveis para download.

Pré-condições: Os casos de uso UC01 (Login) ou UC02 (Cadastro de usuário) e UC05 (Entrar no simulador) devem ter sido executados. Devem ter sido feitas as medidas de DAP de todas as árvores e a cubagem de pelo menos 10% das árvores.

Pós-condições: Após o download da planilha o acesso à floresta dentro do simulador é vetado, impossibilitando o aluno de fazer mais medidas.

Cenário principal: Na página simulador, após o ambiente tridimensional carregar, o botão para download será habilitado a partir do momento em que as pré-condições sejam satisfeitas. Então o usuário clica no botão 'Download da planilha' e o browser se encarregará de gerenciar o download do arquivo, que é no formato xml e possui duas tabelas: uma com as medidas de DAP e altura feitas para cada árvore, identificada pelo seu número, que varia de 1 a 50 e outra com as seções de cada árvore cubada, que são compostas por um diâmetro, uma altura, um número e o número da árvore a qual cada seção pertence.

Cenários alternativos

-Caso o simulador não esteja carregado, o botão ficará desabilitado.
-Caso o usuário tente acessar a planilha de qualquer forma que não seja como descrita no cenário principal, não tendo atingido as medidas necessárias, haverá o encaminhamento para a página de acesso negado.

UC07 – Logout

Descrição: O aluno encerra as atividades dentro do sistema.

Pré-condições: O caso de uso UC01 (Login) ou UC02 (Cadastro de usuário) deve ter sido executados.

Pós-condições: O navegador deverá carregar a página inicial do sistema (index).

Cenário principal: Após ter feito login ou cadastro no sistema e o usuário estar dentro da página 'Principal', 'Simulador' ou 'Créditos', ao clicar no link 'Logout' no menu a sessão do usuário é finalizada.

Cenários alternativos: Não há

UC08 – Locomover

Descrição: As formas como o modelo de engenheiro florestal pode se locomover na floresta.

Pré-condições: Os casos de uso UC01 (Login) ou UC02 (Cadastro de usuário) e UC05 (Entrar no simulador) devem ter sido executados. A tela de menu do simulador (UC09) deve estar fechada.

Pós-condições: Não há.

Cenário principal: Dentro do simulador, na floresta, o engenheiro florestal deve se locomover para as direções norte, sul, leste e oeste segurando as setas do teclado, a saber: cima, baixo, direita e esquerda, ou então, as teclas W, S, D ou A respectivamente. Movimentos diagonais também devem ser possíveis combinando duas teclas direcionais. São as seguintes combinações: esquerda e cima, direita e cima, esquerda e baixo e direita e baixo. Deve ser possível agilizar os movimentos segurando as teclas de direção em conjunto com a tecla shift esquerda, fazendo o engenheiro correr para a direção escolhida. Parando de segurar shift, o engenheiro voltará a apenas caminhar. Parando de segurar uma tecla direcional, caso se esteja segurando mais do que uma das setas, o engenheiro andará para a direção da tecla que ainda esteja pressionada. Caso nenhuma tecla direcional seja pressionada, mesmo que shift seja pressionado, o engenheiro deverá ficar parado.

Cenários alternativos

-Se uma direção oposta a outra for segurada simultaneamente, por exemplo segurar as setas cima e baixo ou esquerda e direita, o engenheiro ficará parado.
-Caso sejam seguradas mais de duas teclas direcionais, o engenheiro responderá apenas à ação das duas últimas teclas que foram pressionadas.
-Apesar de pouco usual não é vetada a combinação de teclas do tipo ‘seta’ com as teclas WASD. Caso segure-se, por exemplo, as teclas W e a seta para direita, o modelo de engenheiro caminhará para o nordeste e caso segure-se W e seta para cima, que representam ambas uma mesma função, o comando deve ser interpretado como se fosse apertada uma tecla só.

UC09 - Visualizar menu

Descrição: Exibição das informações de simulação dentro do ambiente virtual.

Pré-condições: Os casos de uso UC01 (Login) ou UC02 (Cadastro de usuário) e UC05 (Entrar no simulador) devem ter sido executados e a tela de menu do simulador deve estar fechada.

Pós-condições: O caso de uso UC10 (Exibir informações) é exibido.

Cenário principal: Teclando a tecla P, o simulador será pausado e será mostrado o menu de simulação. Os movimentos do engenheiro serão travados, não será possível selecionar árvores, fazer medidas ou qualquer ação além de controlar o menu do simulador com o mouse. Ao passar o mouse sobre cada um dos botões, é mostrada abaixo da tela uma pequena descrição sobre o que cada um dos botões faz. Clicando novamente em P ou clicando no botão Voltar, o menu desaparece e todos os recursos do engenheiro são habilitados novamente.

Os botões presentes são: Informações, que executa UC10 (Exibir informações) , Comandos, que executa UC11 (Exibir comandos), e Voltar, que fecha a tela e libera o controle do engenheiro florestal.

Cenários secundários

-Ao iniciar a simulação, aparecerá por padrão o menu, porém somente quando o jogo é iniciado aparecerá uma mensagem de boas vindas com primeiras instruções (UC05). Ao pausar durante a simulação, a tela padrão é a de informações (UC10).

UC10 - Exibir informações

Descrição: Informações sobre o aluno e suas medidas são exibidas dentro do simulador.

Pré-condições: Os casos de uso UC01 (Login) ou UC02 (Cadastro de usuário) , UC05 (Entrar no simulador) e UC09 devem ter sido executados.

Pós-condições: Não há.

Cenário principal: No menu de pausa ao clicar em informações, é exibido o nome do aluno, seu GRR, dinheiro disponível, o número de árvores com DAP, altura, e cubagem medidas, além das quantidades mínimas para cada uma das medidas. É mostrado também o dinheiro disponível e se a planilha com os dados está disponível ou não.

Cenários alternativos: Não há.

UC11 - Exibir comandos

Descrição: É exibido a lista dos comandos usados dentro do ambiente virtual.

Pré-condições: Os casos de uso UC01 (Login) ou UC02 (Cadastro de usuário) , UC05 (Entrar no simulador) e UC09 devem ter sido executados.

Pós-condições: Não há

Cenário principal: No meu de pausa ao clicar em 'Comandos' é exibido os comandos para controlar o engenheiro pela floresta e fazer as medidas. Os comandos são:

-Locomover: setas do teclado ou teclas WASP;
-Correr: Locomover-se segurando a tecla shift;
-Ajustar câmera: tecla Alt esquerda;
-Alterar visão da câmera: tecla C;
-Pausar: tecla P;
-Medir: tecla M;
-Selecionar árvore: clique sobre uma árvore próxima;
-Olhar: movimento com o mouse, quando em visão em primeira pessoa.

UC12 - Resetar dados

Descrição: Atualiza os dados do aluno no que tange à sua posição no cenário, seu dinheiro e suas medidas. Estes dados retornarão para valores iniciais, iguais aqueles do momento em que acabou de efetuar o cadastro.

Pré-condições: Os casos de uso UC01 (Login) ou UC02 (Cadastro de usuário) , UC05 (Entrar no simulador) e UC06 devem ter sido executados.

Pós-condições: É executado o caso de uso UC05 (Entrar no simulador).

Cenário principal: Ao clicar em 'Resetar dados' no menu de pausa, aparecerá uma janela de confirmação e feito isso, todas as medidas do usuário serão resetadas, bem como seu dinheiro e posição inicial no cenário, de forma que ele possa recomeçar a simulação. Os valores seguintes valores deverão ser atualizados:

-Dinheiro para 2000;
-Estado para 0;
-PosicaoX para 0.0;
-PosicaoY para 0.0;
-PosicaoZ para 0.0;

Cenários alternativos

-Caso o servidor esteja fora do ar, aparecerá uma mensagem informando que os dados não puderam ser resetados.

UC13 - Trocar instrumento

Descrição: Altera a ferramenta que está nas mãos do engenheiro florestal.

Pré-condições: Os casos de uso UC01 (Login) ou UC02 (Cadastro de usuário) , UC05 (Entrar no simulador) devem ter sido executados.

Pós-condições: Não há.

Cenário Principal: O engenheiro inicia a simulação com nada nas mãos. Quando ele clicar nas teclas Q ou E uma das ferramentas aparecerá na mão do engenheiro florestal e aparecerá uma mensagem na tela com o nome da ferramenta que foi selecionada. A ordem das ferramentas é "Nenhuma ferramenta", "Suta", "Hipsômetro" e "Motosserra". Clicando a tecla E a troca das ferramentas será nessa ordem até que volte para "Nenhuma ferramenta", fechando um ciclo. Caso aperte a tecla Q, o ciclo ocorrerá de forma contrária, ou seja, de "Nenhuma ferramenta" passará para "Motosserra" e assim por diante.

Cenários alternativos: Não há.

UC14 - Selecionar árvore

Descrição: O engenheiro seleciona uma das árvores da floresta.

Pré-condições: Os casos de uso UC01 (Login) ou UC02 (Cadastro de usuário) , UC05 (Entrar no simulador) devem ter sido executados.

Pós-condições: Não há.

Cenário principal: O aluno ao selecionar uma árvore clicando nela, a árvore deve ser circundada com um objeto que indique a seleção e um quadro deve aparecer informando os dados sobre o pinheiro, que são o seu número, e os valores das medidas de DAP e altura, caso feitas. Clicando em outra árvore esta passa a ser selecionada e é exibido os dados desta, e assim por diante.

No quadro também há um botão de ‘Medir’ e um botão de ‘Fechar’. O segundo fecha a janela e desseleciona a árvore destruindo o objeto que representa a seleção.

Cenários alternativos

-Caso o engenheiro esteja muito longe da árvore selecionada (cinco metros) ele não poderá ser capaz de selecioná-la. Caso o simulador esteja pausado, isto é, mostrando o menu de opções (UC09), a seleção de árvores será desabilitada, o mesmo procedimento vale para a visão em primeira pessoa (UC16).

UC15 - Ajustar câmera

Descrição: Ajusta a câmera para que se posicione para as costas do engenheiro florestal.

Pré-condições: Os casos de uso UC01 (Login) ou UC02 (Cadastro de usuário) e UC05 (Entrar no simulador). O modo de câmera não deve estar em primeira pessoa nem em modo fixo.

Pós-condições: Não há.

Cenário principal: Quando a câmera estiver em modo de visão próxima, visão normal ou vista de cima, é possível ajustar a posição dela para que fique posicionada nas costas do engenheiro clicando a tecla alt. Caso o engenheiro não faça nenhum movimento por cerca de dez segundos, a câmera deverá se ajustar automaticamente.

Cenários alternativos

-Caso a visão seja em primeira pessoa ou no modo de câmera fixa, não acontecerá nenhuma alteração na câmera.

UC16 - Mudar visão

Descrição: Altera a visão do engenheiro entre diferentes modos.

Pré-condições: Os casos de uso UC01 (Login) ou UC02 (Cadastro de usuário) e UC05 (Entrar no simulador).

Pós-condições: O UC17 é habilitado quando a visão é em primeira pessoa.

Cenário principal: A câmera em modo normal é a visão em terceira pessoa posicionada nas costas do engenheiro há uma distância de 3 metros dele. Caso que clique na tecla C, a visão passará para visão próxima, onde a distância entre a câmera e o engenheiro passa a se situar a 1 metro mais próximo a ele. Teclando C novamente a vista passa a ser em primeira pessoa, onde o usuário passará a ver o cenário como se fosse pela visão do próprio engenheiro, porém, sem capacidade para locomoção, seleção de árvores, ou realizar medidas. Clicando C novamente, a câmera se posicionará em 3 metros de altura do engenheiro, caracterizando uma vista de cima. Teclando C mais uma vez a câmera é travada e não acompanhará o movimento do modelo. Finalmente, teclando C novamente, a câmera deverá voltar para a visão normal.

Cenários alternativos: Não há.

UC17 – Olhar

Descrição: O modelo do engenheiro olha para o cenário em visão de primeira pessoa.

Pré-condições: Os casos de uso UC01 (Login) ou UC02 (Cadastro de usuário), UC05 (Entrar no simulador) e UC16 (Mudar visão) em modo de visão em primeira pessoa devem ter sido executados.

Pós-requisitos: Não há.

Cenário principal: Movimentando o mouse o modelo de engenheiro olhará para a direção onde a seta do mouse estiver. Devem ser possíveis movimentos que permitam que o engenheiro olhe para os lados, para cima e para baixo.

Cenários alternativos: Não há.

UC18 - Medir DAP

Descrição: A medida do DAP é obtida de uma árvore.

Pré-condições: Os casos de uso UC01 (Login) ou UC02 (Cadastro de usuário) , UC05 (Entrar no simulador) e UC14 (Selecionar árvore) devem ter sido executados.

Pós-condições: A planilha pode ser baixada (UC06) quando todas as árvores tiverem seus dados obtidos e 5 árvores forem derrubadas.

Cenário principal: Com a ferramenta suta em mãos o aluno clica no botão medir ou aperta a tecla M, então o engenheiro florestal mede o DAP da árvore. É descontado R$ 1,00 e os dados da posição, dinheiro, estado do engenheiro a as medidas feitas são gravadas no banco de dados. A árvore é marcada com uma fita ao seu redor informando que o DAP dela já foi medido. A informação sobre o número de árvores com DAP medidos na tela de ‘Informações’ é atualizada.

Cenários alternativos

-Quando todos os DAPs forem medidos e 10% das árvores forem cubadas uma mensagem aparecerá na tela informando que a planilha com os dados das medidas estão disponíveis para download. Na página do simulador o botão para download da

planilha é habilitado. O estado do engenheiro deverá mudar. É apenas um valor numérico que informará que todas as medidas já foram feitas, servindo para habilitar o botão de download da planilha.
-Se o DAP já foi medido uma mensagem deve ser mostrada avisando que esta medida já foi feita.
-Caso o engenheiro florestal não tenha pelo menos R$ 1,00 uma mensagem deverá avisar que não há mais dinheiro para fazer medidas.

UC19 - Medir altura

Descrição: A altura de uma árvore é medida.

Pré-condições: Os casos de uso UC01 (Login) ou UC02 (Cadastro de usuário) , UC05 (Entrar no simulador) e UC14 (Selecionar árvore) devem ter sido executados.

Pós-condições: Não há.

Cenário principal: Com a ferramenta hipsômetro em mãos o aluno clica no botão medir ou aperta a tecla M, então o engenheiro florestal deverá medir a altura da árvore. É descontado R$ 5,00 e os dados da posição, dinheiro, estado do engenheiro a as medidas feitas são gravadas no banco de dados. A árvore é marcada com uma fita ao redor dela informando que a altura já foi medida. A informação sobre o número de árvores com alturas medidas na tela de ‘Informações’ é atualizada.

Cenários alternativos

-Se a altura já foi medida uma mensagem deve ser mostrada avisando que esta medida já foi feita. A medida não é feita.
-Caso o engenheiro florestal não tenha pelo menos R$ 5,00 uma mensagem deverá avisar que não há mais dinheiro para fazer medidas. A medida não é feita.

UC20 - Derrubar árvore

Descrição: Uma árvore é derrubada para obtenção dos dados de cubagem.

Pré-condições: Os casos de uso UC01 (Login) ou UC02 (Cadastro de usuário) , UC05 (Entrar no simulador), UC14 (Selecionar árvore) e UC18 (Medir DAP) devem ter sido executados.

Pós-condições: A planilha pode ser baixada (UC06) quando todas as árvores tiverem seus dados obtidos e 10% das árvores forem derrubadas.

Cenário principal: Com a ferramenta motosserra em mãos o aluno clica no botão medir ou aperta a tecla M, então o engenheiro florestal derruba . É descontado R$ 80,00 e os dados da posição, dinheiro, estado do engenheiro a as medidas feitas são gravadas no banco de dados. A árvore desaparece do cenário. A informação sobre o número de árvores derrubadas (cubadas) na tela de ‘Informações’ é atualizada. Este caso de uso também marcará a altura como medida, em vista da possibilidade de ter este valor ao cubar uma árvore.

Cenários alternativos

-Quando todos os DAPs forem medidos e 5 árvores forem cubadas uma mensagem aparecerá na tela informando que a planilha com os dados das medidas estão disponíveis para download. Na página do simulador o botão para download da planilha é habilitado.
-Caso não o engenheiro florestal não tenha pelo menos R$ 80,00 uma mensagem deverá avisar que não há mais dinheiro para fazer medidas.

Diagrama de Classes de Análise

Segue abaixo o diagrama de classes de análise do projeto. Este diagrama independe de tecnologia usada, porém já pode-se adiantar que as classes Aluno, Medida e SecaoArvore serão implementadas do lado do servidor. A classe Arvore possuirá tanto representação do lado do servidor quanto do lado do simulador e as demais classes serão integrantes do simulador.

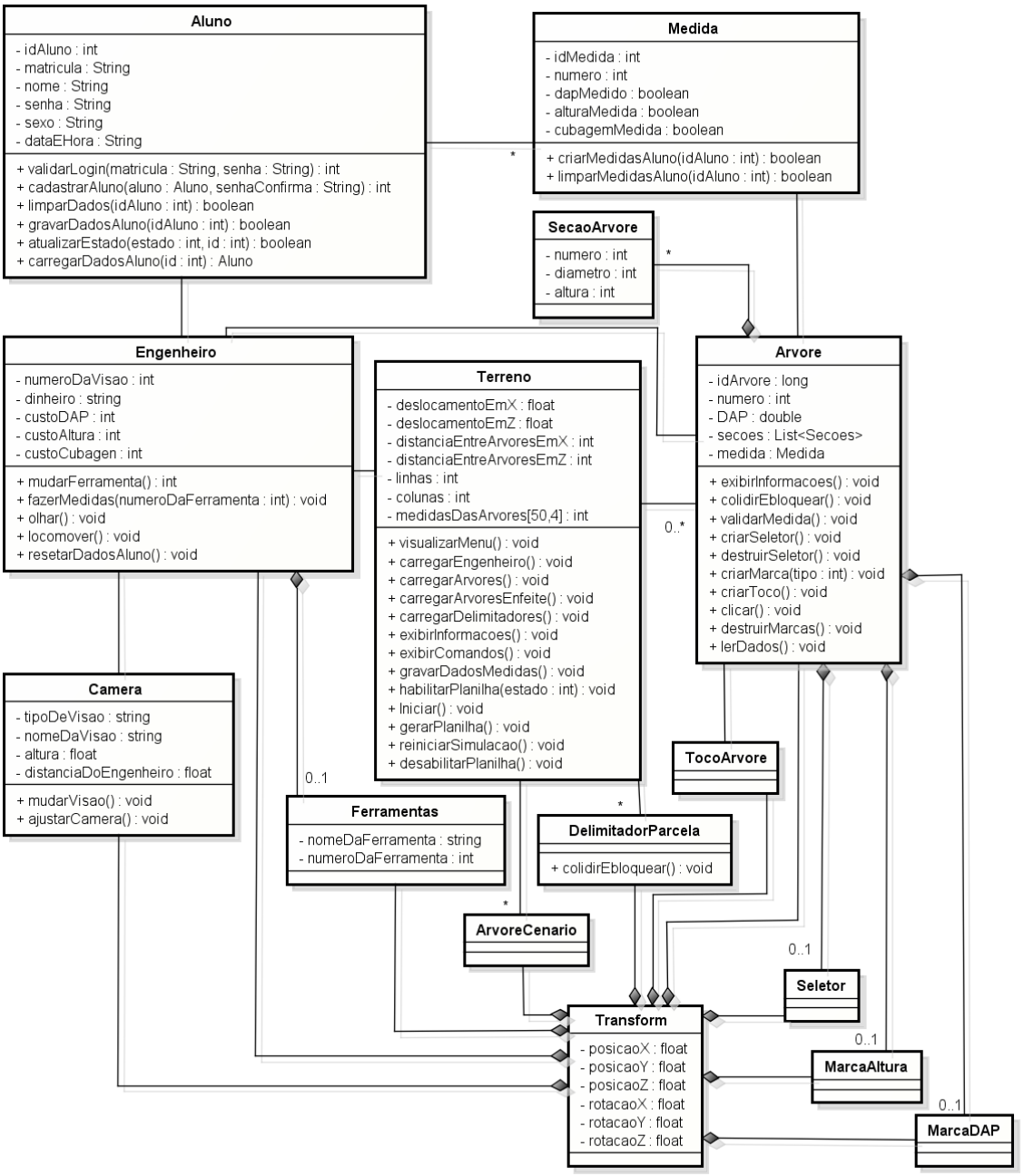

Figura 39: Diagrama de Classes de Análise.

Diagrama Entidade-Relacionamento

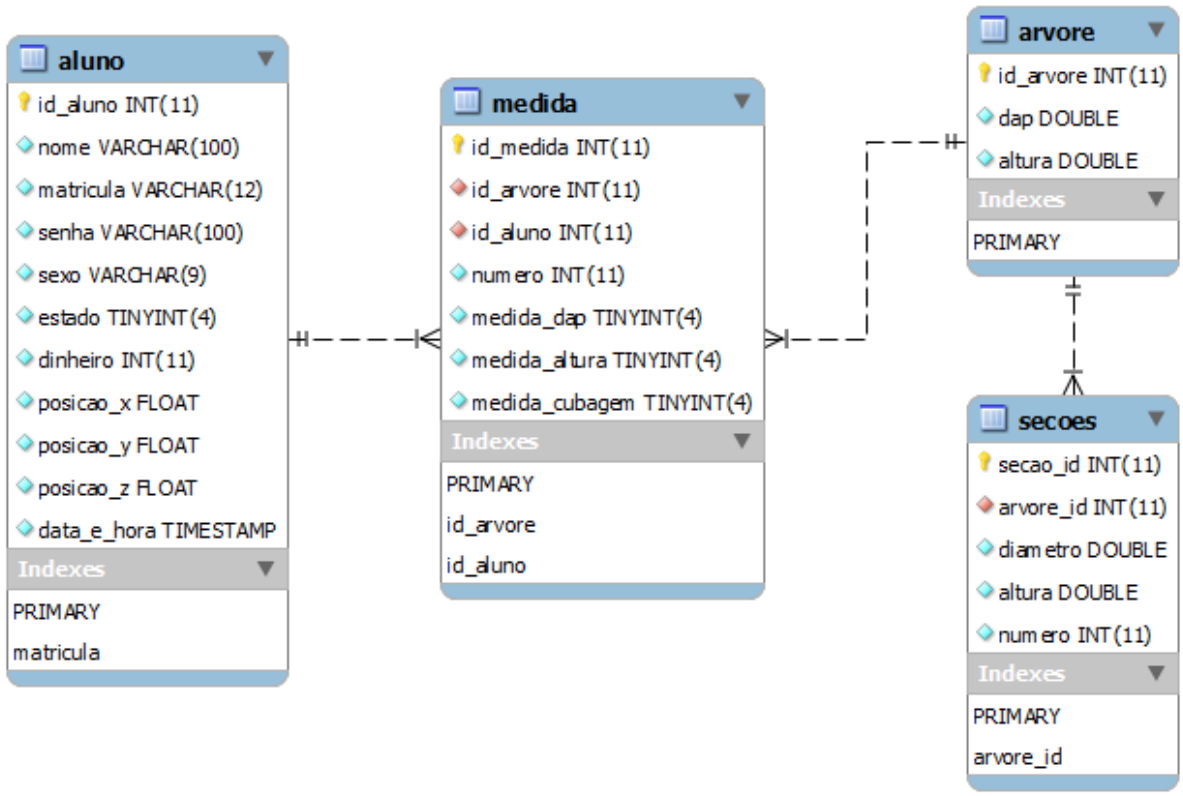

Figura 40: Diagrama Entidade-Relacionamento.

Diagramas de Sequência de Análise

Diagrama de Sequência de Análise – Login

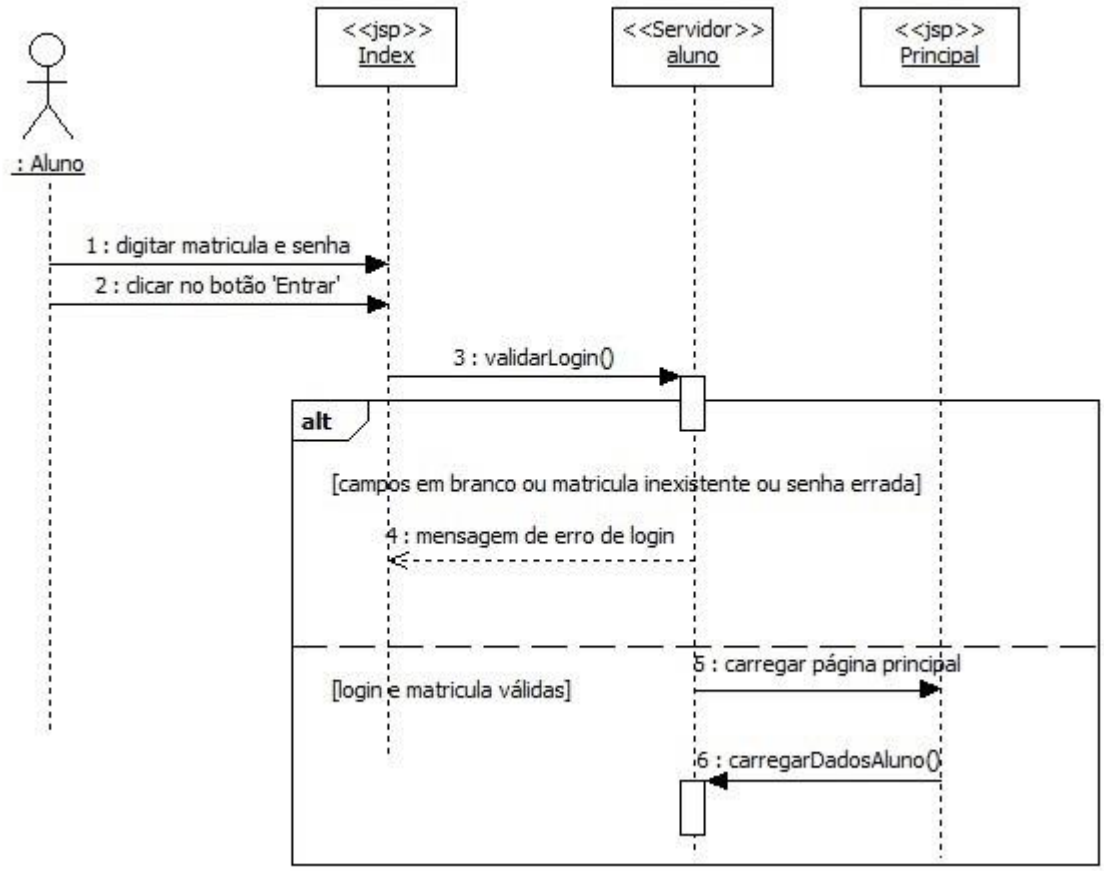

Figura 41: Diagrama de Sequência de Análise para a funcionalidade de login no sistema.

Diagrama de Sequência de Análise – Cadastro

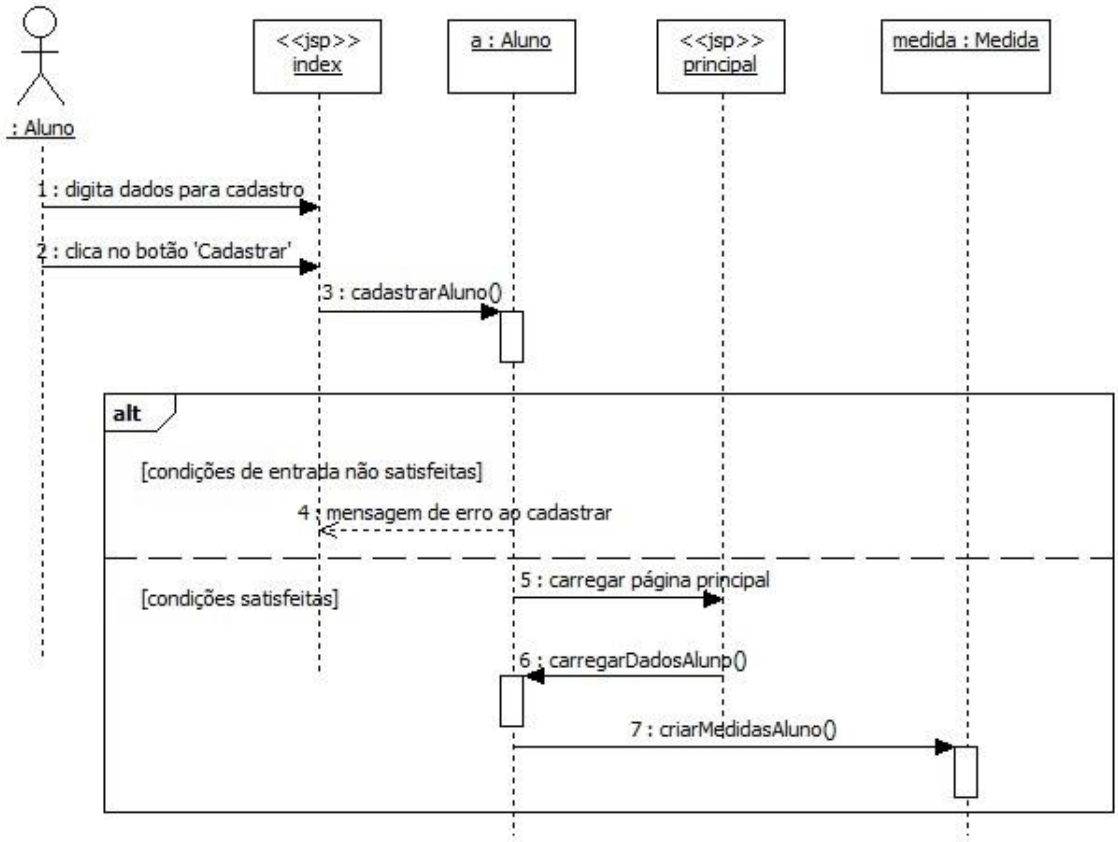

Figura 42: Diagrama de Sequência de Análise para a funcionalidade de cadastro no sistema.

Diagrama de Sequência de Análise – Carregamento do Simulador

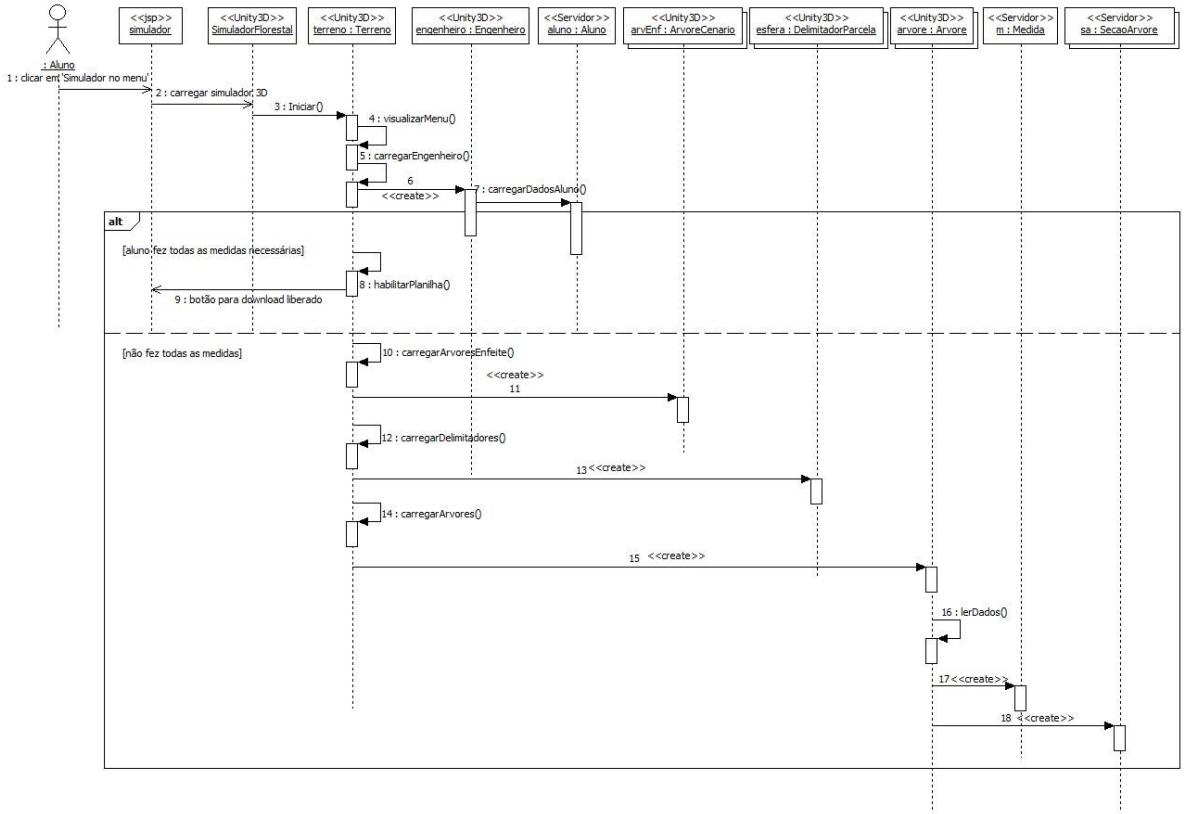

Figura 43: Diagrama de Sequência de Análise que descreve o carregamento do simulador e seus elementos.

Diagrama de Sequência de Análise – Selecionar Árvore

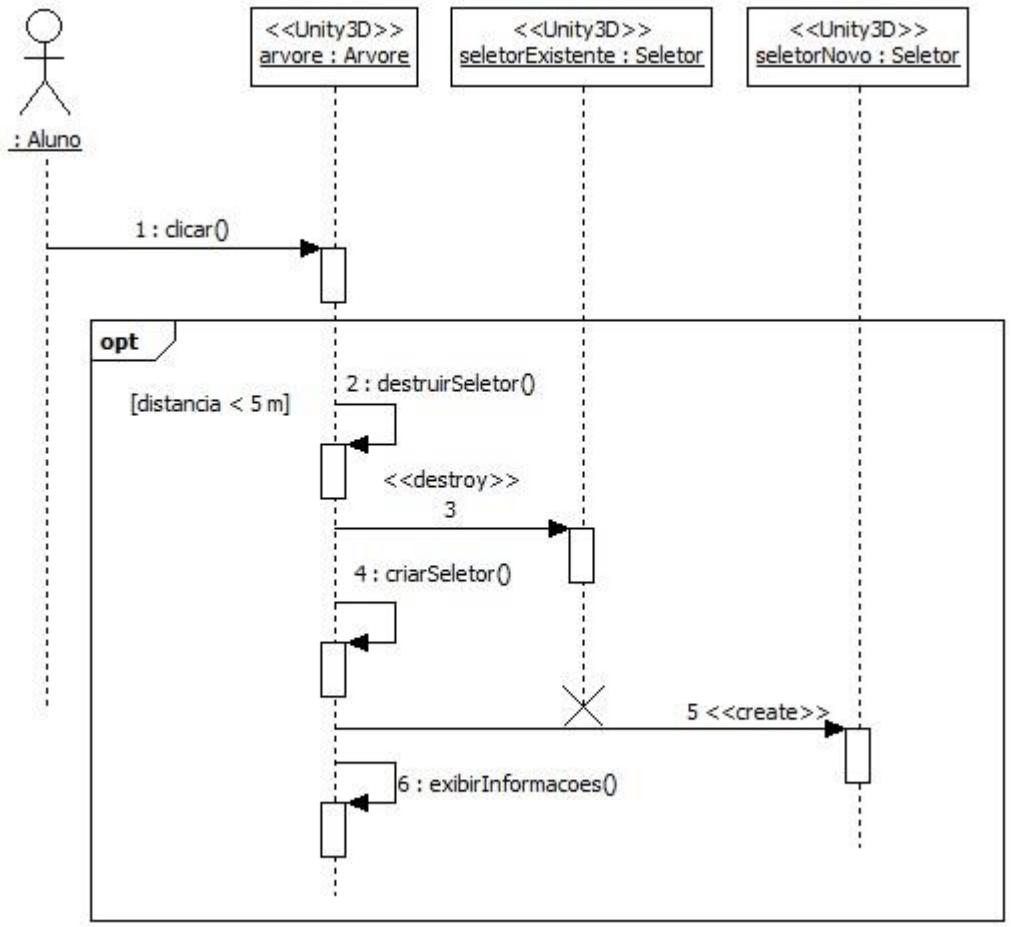

Figura 44: Diagrama de Sequência de Análise para a funcionalidade de selecionar uma árvore.

Diagrama de Sequência de Análise – Medir DAP

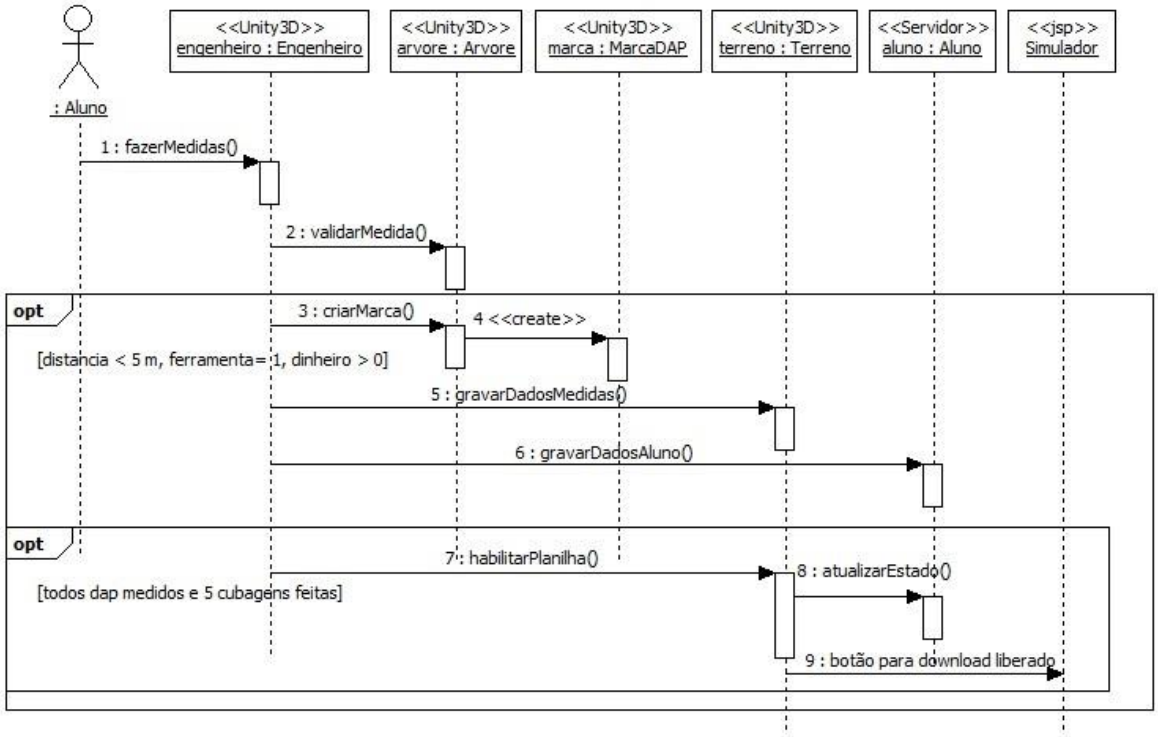

Figura 45: Diagrama de Sequência de Análise para a funcionalidade de medida do DAP de uma árvore.

Diagrama de Sequência de Análise – Medir Altura

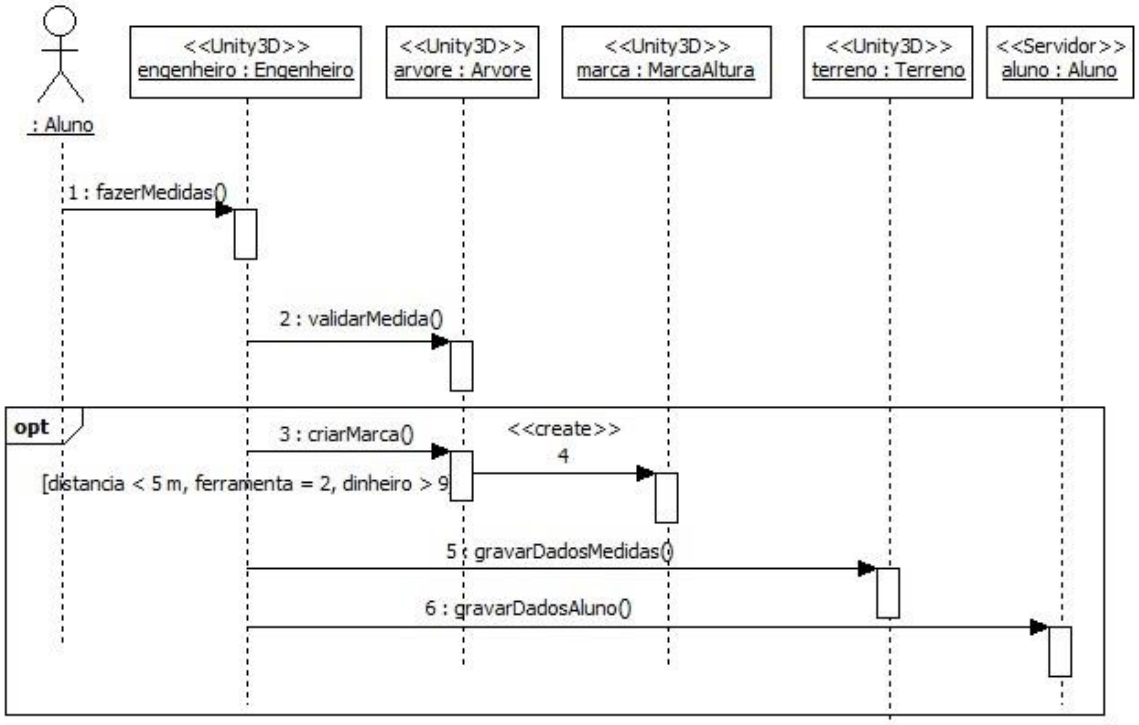

Figura 46: Diagrama de Sequência de Análise para a funcionalidade de medida de altura de uma árvore.

Diagrama de Sequência de Análise – Cubagem

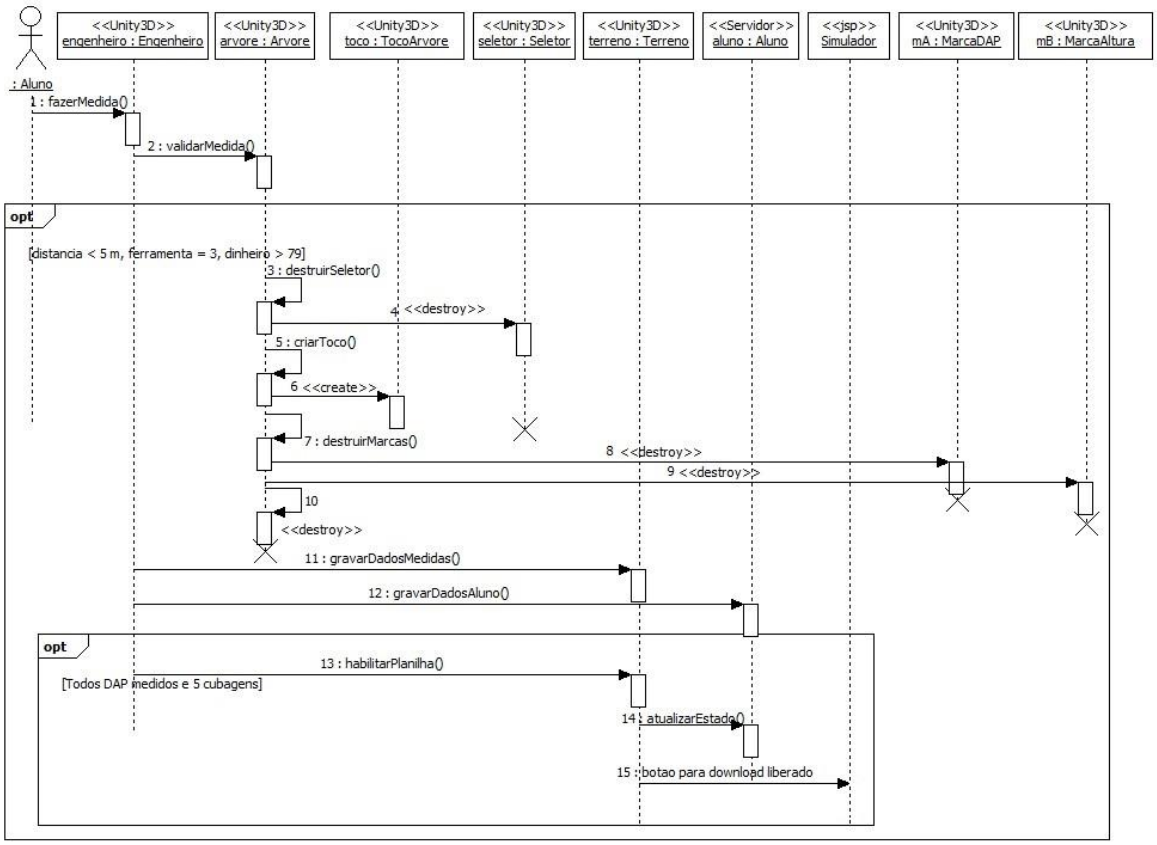

Figura 47: Diagrama de Sequência de Análise para a funcionalidade de cubagem de uma árvore.

Diagrama de Sequência de Análise – Resetar Dados

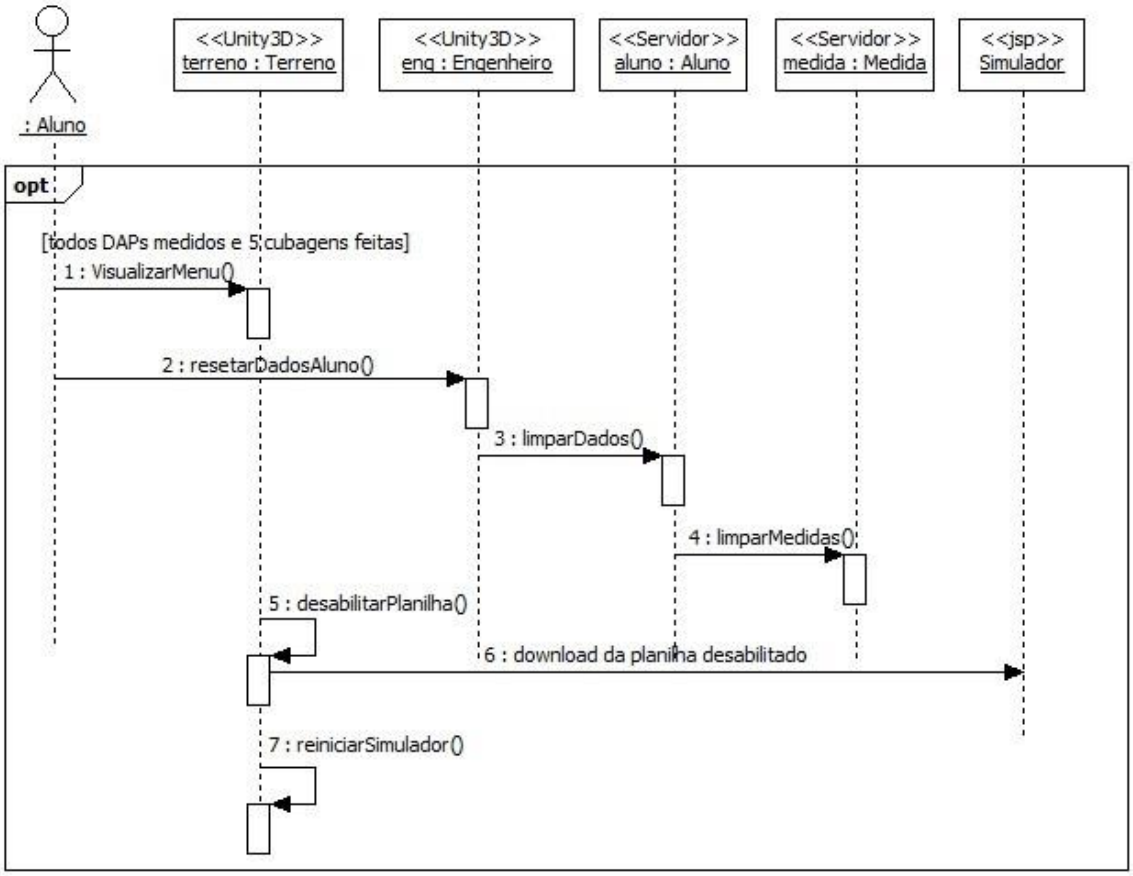

Figura 48: Diagrama de Sequência de Análise para a funcionalidade de resetar os dados.

Diagramas de Implementação

Aqui serão mostrados os diagramas de classe e os diagramas de sequência tal como foram implementados no sistema. O diagrama de classes do servidor é composta de apenas um diagrama, seguindo a orientação a objetos padrão, já que foi desenvolvido em Java. Já o diagrama de classes para a parte do simulador, construída com a tecnologia Unity3D será criado a partir de uma série de diagramas que abordarão a forma como o Unity3D funciona.

Em primeiro lugar, os elementos que constituem um projeto do Unity3D são dispostos em um cenário de forma gráfica; as classes que definem os objetos não são possíveis de serem alteradas pelo desenvolvedor. Pode-se afirmar que os objetos do cenário são instâncias de uma classe, da mesma forma como prega a metodologia orientada a objetos. Porém, cada classe de objetos para atender os mais diversos fins, são compostas por diversas outras, denominadas *components*.

Quando um objeto é desenvolvido para uso no simulador, por exemplo, uma árvore ou um engenheiro, criamos um objeto vazio, reconhecido no Unity3D como um *GameObject*. Este objeto vazio já carrega consigo um *component* do tipo *Transform*, que informa o posicionamento do objeto dentro de uma cena (uma região do cenário do projeto). Em seguida, adicionamos diversos *components* diferentes para caracterizar o objeto da melhor forma que o represente e então, adicionamos os scripts que definirão o seu comportamento personalizado. Os scripts, que são escritos pelos desenvolvedores, também são *components*. Ao seguir essas etapas temos uma composição de classes que define o que no Unity3D é chamado de *Prefab*. *Prefabs* são objetos prontos para o uso que podem ser instanciados em tempo de execução ou arrastados para dentro do cenário para que já sejam carregados automaticamente quando o projeto é executado. Estas estruturas serão consideradas as classes implementadas e documentadas a seguir.

Os diagramas seguintes refletirão os passos para a elaboração de cada um dos *Prefabs* utilizados no simulador. Primeiro serão mostradas as classes principais que formam um objeto no Unity3D, as classes *GameObject* e sua mãe *Object*. Em seguida as classes dos *components* utilizados no projeto serão retratadas (há muitas outras além das usadas no simulador), bem como um diagrama de classes dos *scripts* desenvolvidos em C Sharp. A partir disso, diagramas de classes para cada

Prefab serão gerados e finalmente, o Diagrama de Classes de Implementação e o de sequência serão mostrados, construídos a partir dos *Prefabs*.

Por último, deve ser informado que as classes foram criadas com o auxílio da documentação do Unity3D disponível no link abaixo.

http://docs.unity3d.com/Documentation/ScriptReference/

Cada uma das classes foi montada seguindo as descrições contidas no site. A documentação oficial não fornece diagramas, mas auxilia listando os atributos, métodos e classes mãe de cada classe. Seguem os diagramas na página seguinte.

Diagrama de Classes de Implementação do servidor

Abaixo segue o Diagrama de Classes de Implementação do servidor. A forma como o servidor foi construído segue os padrões de projeto MVC e DAO.

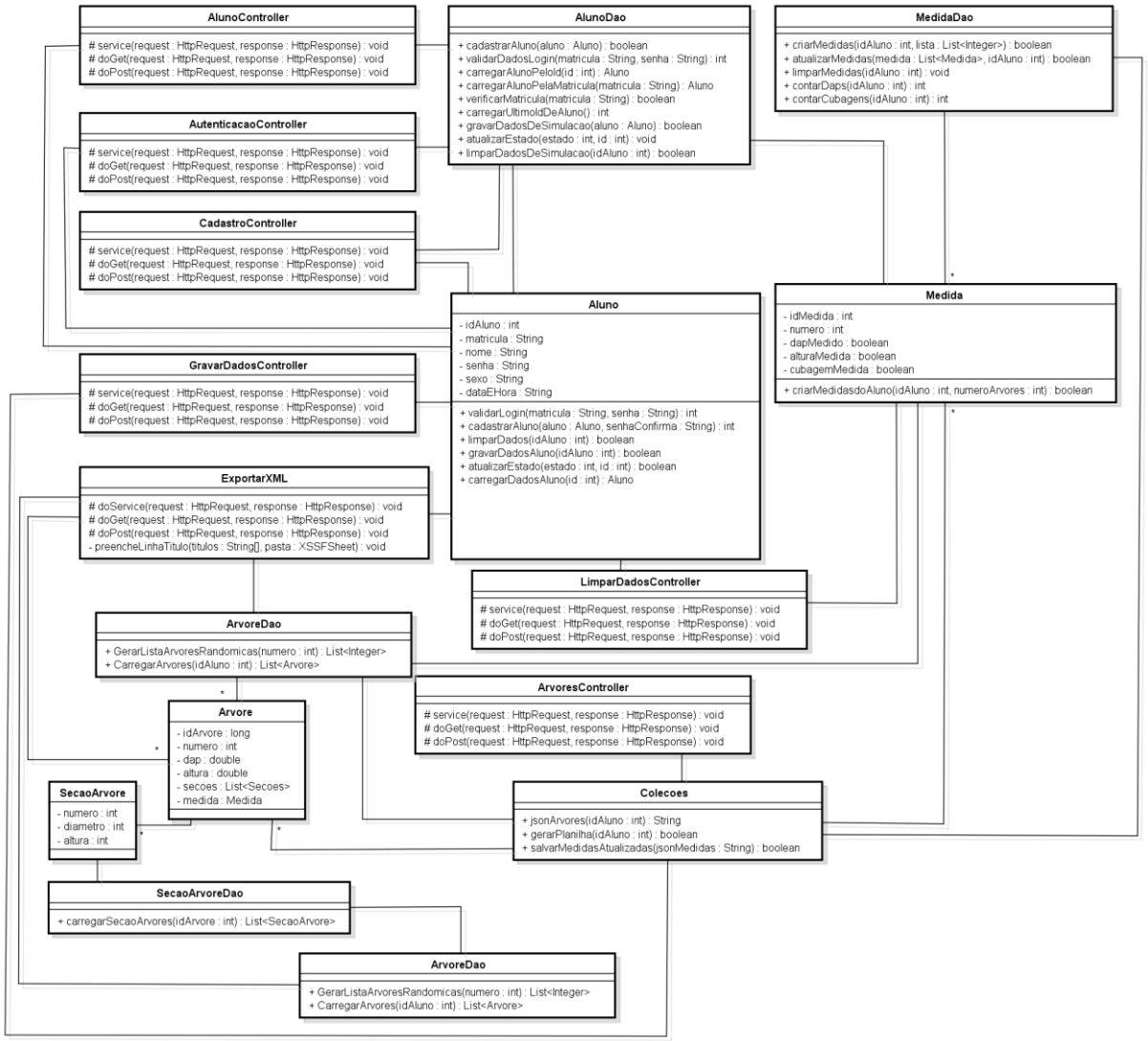

Figura 49: Diagrama de Classes de Implementação do servidor Java.

Classes *Object, GameObject* e *Component*

Estas três classes são a base para qualquer objeto do Unity3D. O diagrama abaixo mostra suas relações. Note que Object é o topo da hierarquia das classes do Unity3D. Um detalhe curioso é que ao desenvolver scripts, para acessar a classe Object do C Sharp é necessário utilizar o *namespace* System para distinguir qual classe está sendo tratada.

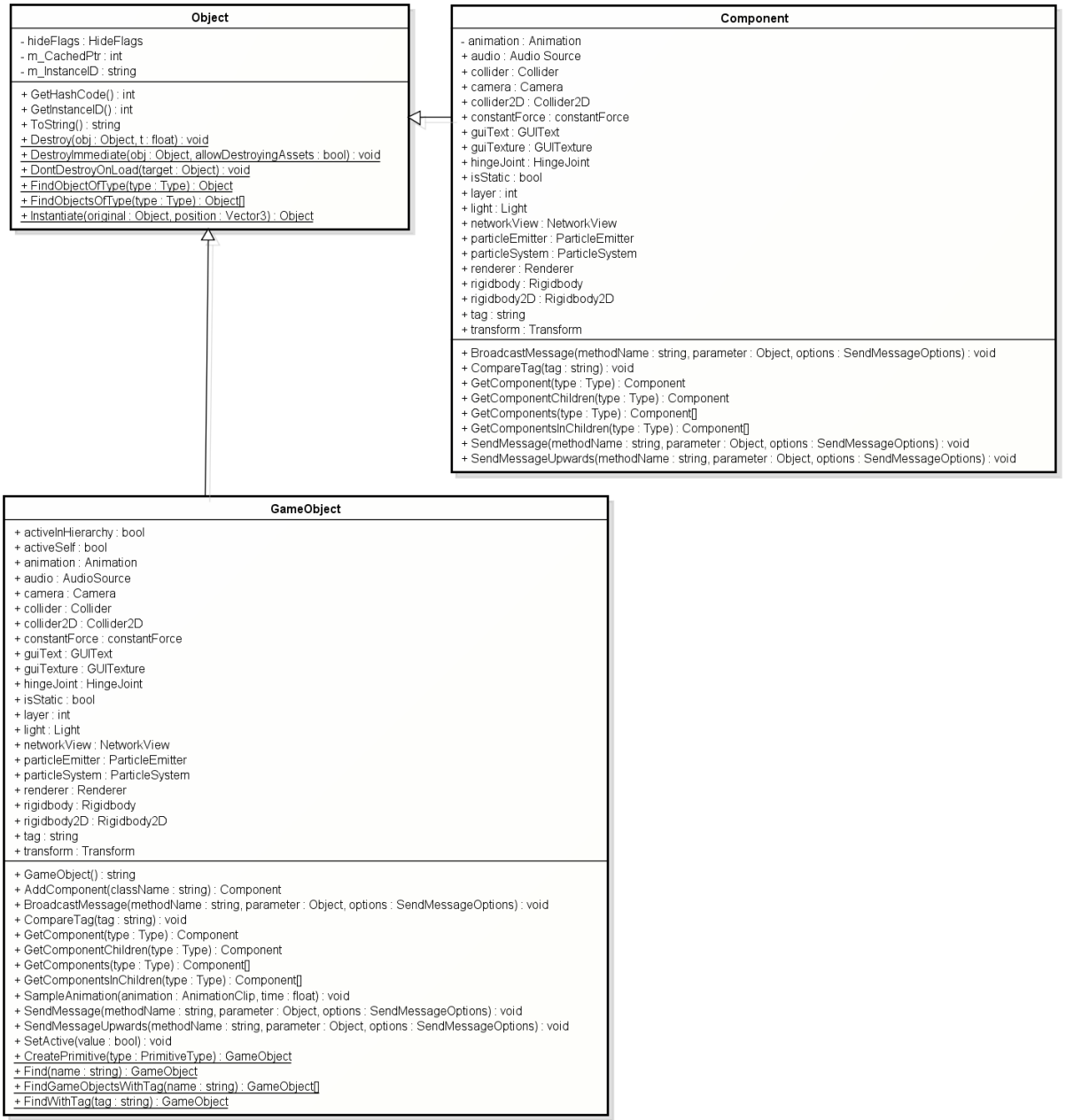

Figura 50: Diagrama para classes *Object*, *GameObject* e *Component*.

Classes *Behaviour* e *MonoBehaviour*

Behaviour herda de *Component* e é classe mãe de várias classes que são *Component*s e que necessitem do comportamento fornecido por esta classe: a capacidade dela poder ser desabilitada ou habilitada em tempo de execução por meio de scripts. Uma das classes que estendem de *Behaviour* é a classe *MonoBehaviour*, mostrada em conjunto no diagrama abaixo. *MonoBehaviour* é um

component da qual todos os scripts são filhos. Scripts em JavaScript estendem automaticamente, porém ao escrever os códigos em C Sharp é necessário estender explicitamente a classe.

Figura 51: Classes Behaviour e MonoBehaviour.

Classes *Transform*, *Light* e *Camera*

Estas três classes são *components* praticamente obrigatórios em qualquer cena no Unity3D. *Transform* sempre é anexado quando um novo *GameObject* é criado, pois nesta classe estão os métodos e atributos de posicionamento do objeto em uma cena. Cada objeto obrigatoriamente deve estar em algum lugar, mesmo que ele não seja visível. *Light* garante que uma cena seja iluminada quando associada em algum *GameObject* e *Camera* é a visão fornecida ao usuário da cena de um projeto do Unity3D, análogo às câmeras de televisão.

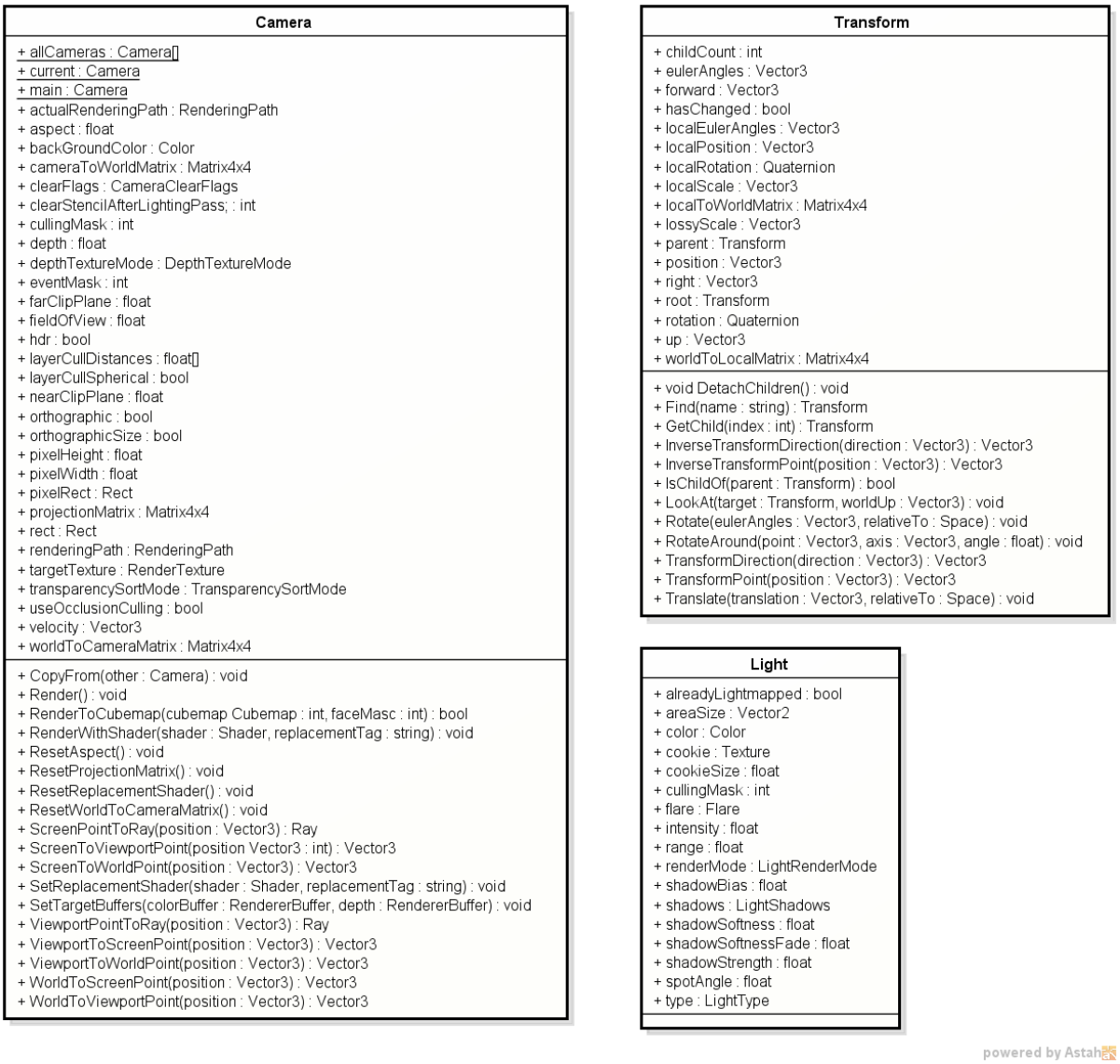

Figura 52: As Classes *Light*, *Camera* e *Transform*.

Classes *Collider* e *Rigidbody* - *components* de física

Quando há a necessidade de fazer com que um objeto detecte um choque com um outro para impedir que eles não atravessem um ao outro, é utilizado *components* do tipo *Collider*. Vários tipos destes herdam de *Collider* e se diferenciam no formato da área que deve ser detectada a colisão. Uma bola de futebol usaria um *SphereCollider* pois se ajustaria perfeitamente às dimensões esféricas da bola, por exemplo.

Rigidbody é utilizado quando há a necessidade de tratar forças que atuam sobre um objeto, como gravidade, torque e outras, além de variáveis físicas do objeto, como massa e também restrições ao movimento do objeto.

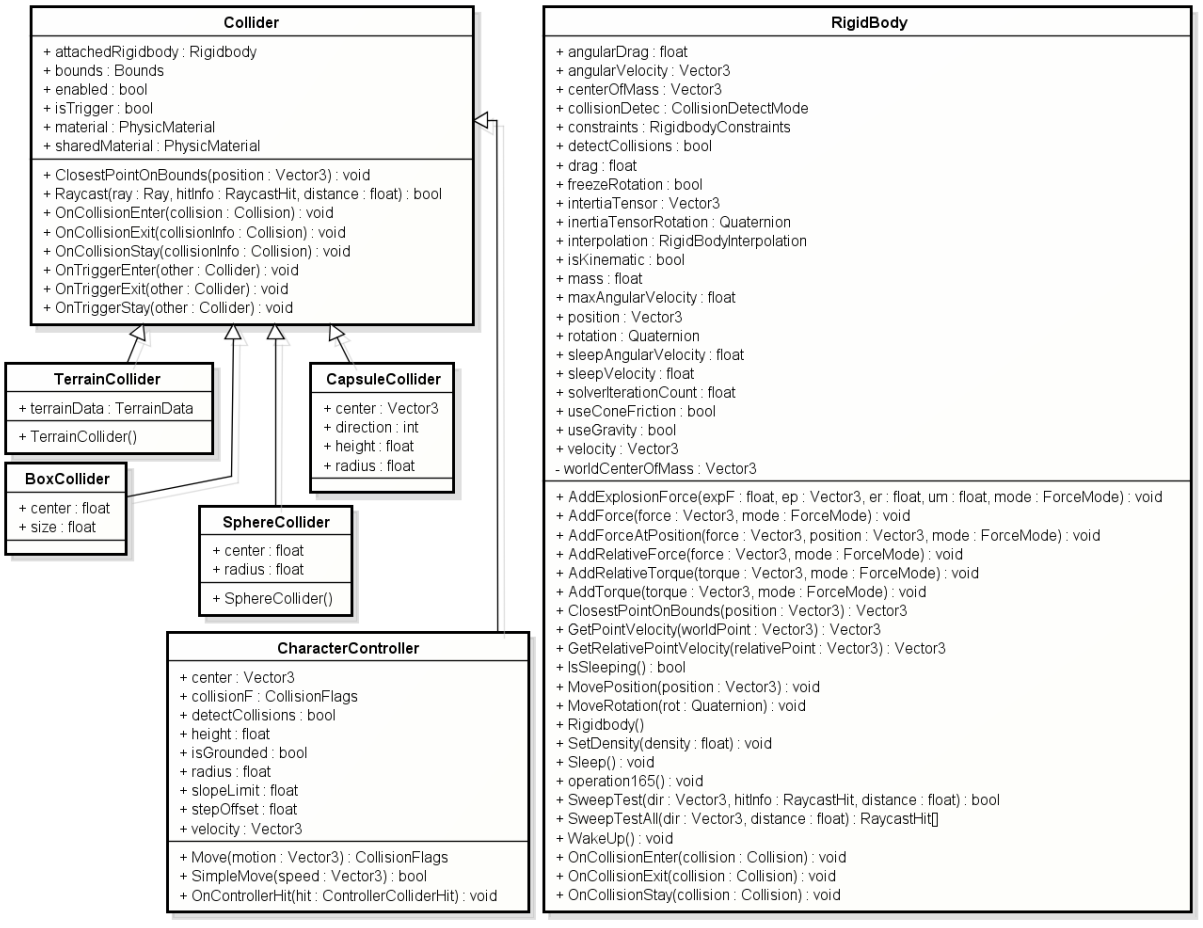

Figura 53: Classes *Collider* e suas filhas e classe *Rigidbody*.

Classes *AudioSource* e *AudioListener*

Cada elemento de áudio tocado dentro de uma cena é associado a um *component AudioSource* que fornece diversos ajustes para como o som deve ser executado, tal como o seu volume, tempo de duração, se é executado em *loop* ou se e como implementa-se o efeito Doppler nele (este efeito é detectado quando um objeto movimenta-se em relação à fonte de som, causando uma distorção em na frequência do áudio, resultando em um som mais agudo quando a fonte se aproxima do objeto e mais grave quando há um afastamento).

AudioListener é usado para que o som possa ser ouvido pelo usuário. É associado a apenas um objeto da cena e, por exemplo, *AudioSources* mais próximos são ouvidos com um som mais alto do que *AudioSources* mais distantes do objeto que possui o *AudioListener* associado.

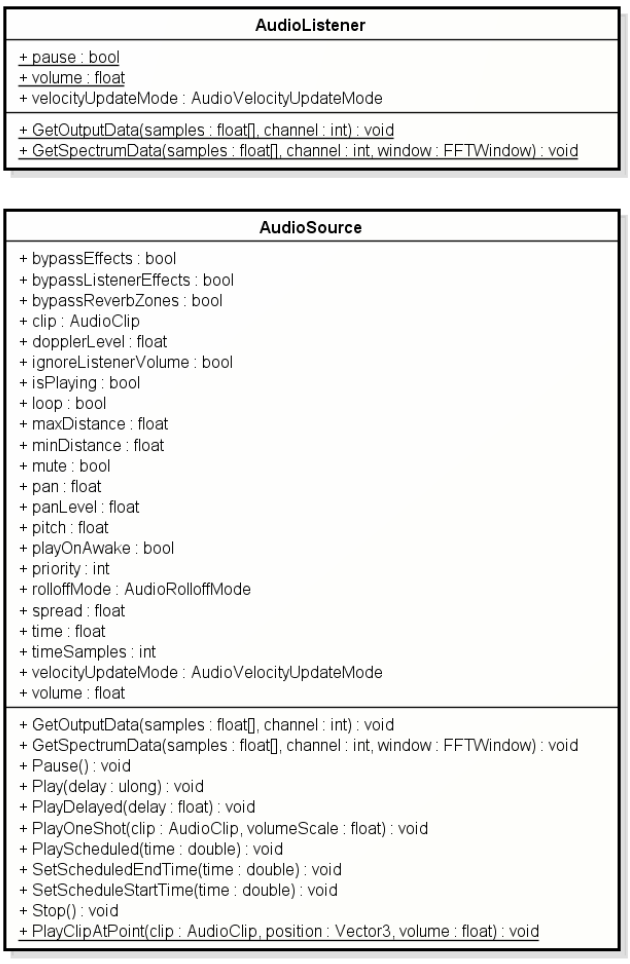

Figura 54: Classes *AudioSource* e *AudioListener*.

Classe *Animation*

Quando um objeto do cenário como um engenheiro florestal caminha pelo cenário, o movimento do seu corpo é feita por meio de animações. Estas animações são feitas utilizando um editor do Unity3D onde cada membro do corpo do modelo é rotacionado de determinada forma à escolha do desenvolvedor em cada intervalo de tempo. Esta animação é salva em um arquivo e anexada ao *component Animation* que gerencia de que forma a animação será executada, semelhante ao *component AudioSource*.

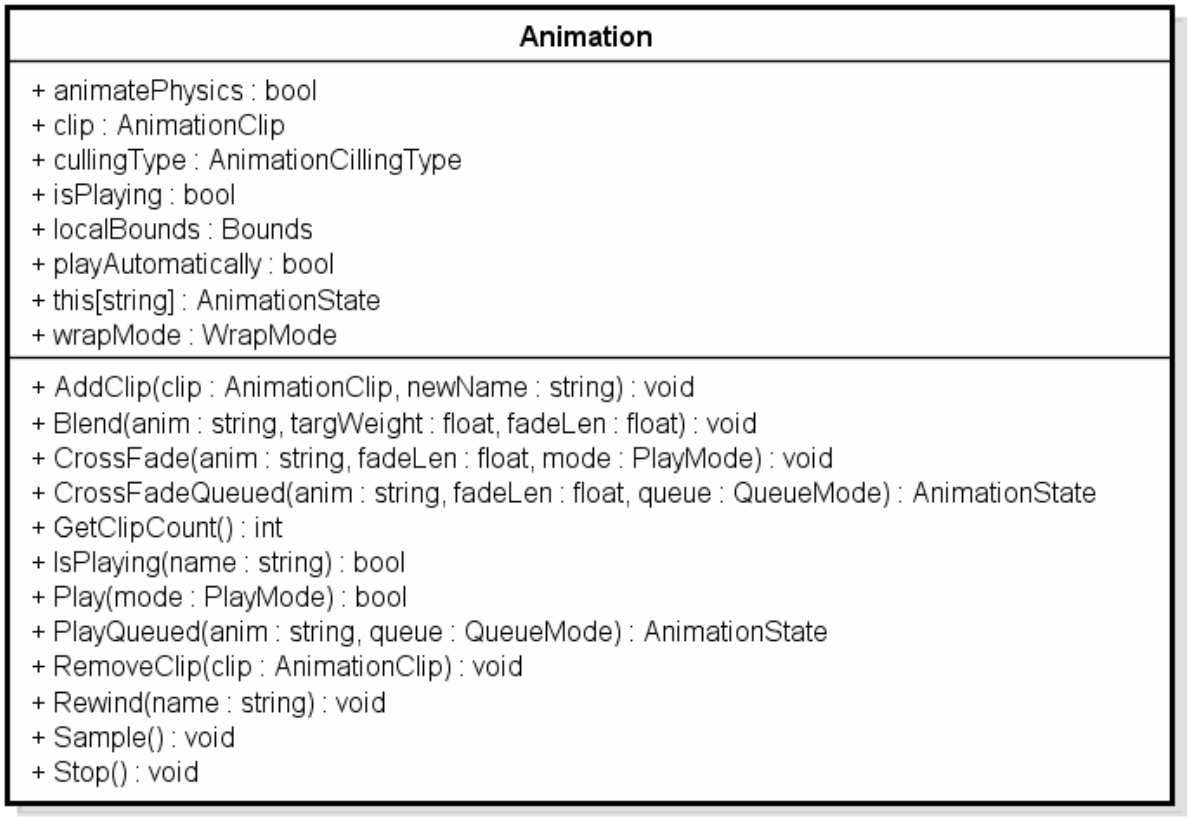

Figura 55: Classe *Animation*.

Diagrama de classes dos *scripts* desenvolvidos

Os *scripts* são à forma com que os objetos de uma cena se comportam da forma que o desenvolvedor projeta. Estes *components* tem a capacidade de acessar os métodos e atributos dos demais *components* de um *prefab*, instanciar outros e até mesmo destruir *prefabs* da cena. Também neles que são escritos métodos para comunicação com servidor externo e muitos outras possibilidades.

Eles podem ser descritos com um diagrama de classes normal pois possuem relacionamentos semelhantes ao padrão da orientação a objetos. É importante ressaltar que a forma com que um *prefab* se relaciona com outros de uma cena é feita por meio de *scripts*, caso contrário não haveria coesão entre cada objeto disposto em um cenário. Segue o diagrama na página seguinte.

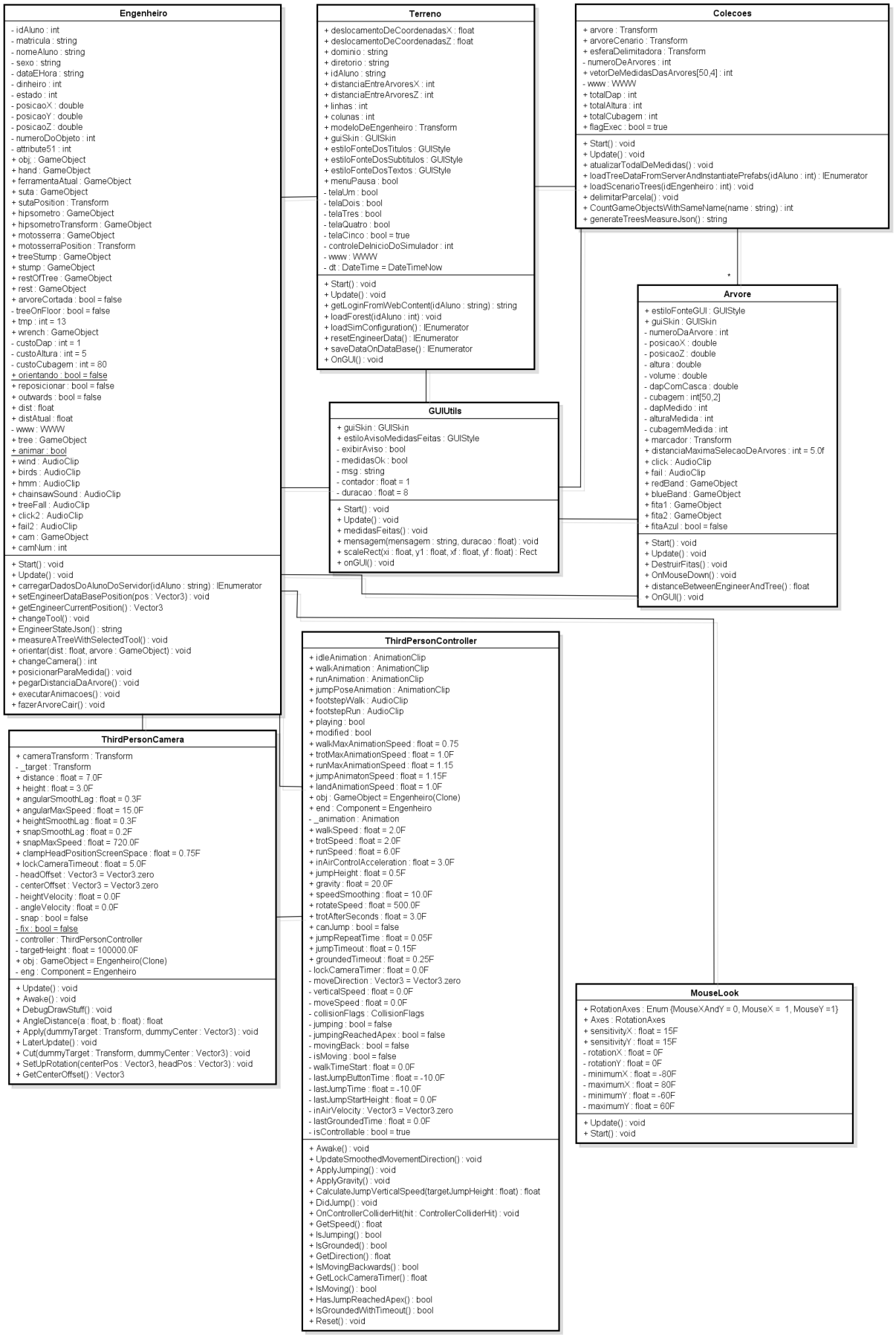

Figura 56: Diagrama de classes dos *scripts* do Simulador Florestal.

Diagramas de classes dos *prefabs*

Definidos os *components* e classes essenciais, agora é possível montar cada *prefab* que representa as classes para os objetos do simulador. Por uma questão de legibilidade, serão mostradas apenas as relações entre as classes definidas anteriormente, o que não é um grande problema pois os atributos e métodos já foram mostrados.

O último diagrama desta seção representará vários *prefabs*, o que pode parecer estranho já que uma motosserra não se parece em nada com uma suta. Isso ocorre porque a parte da renderização gráfica de cada objeto, como suas texturas e polígonos é feita de forma invisível ao desenvolvedor, e não se associando a nenhum tipo de *component*. O que resta de diferente neste aspecto seria somente o nome do *prefab*. Os tipos de components usados são os mesmos.

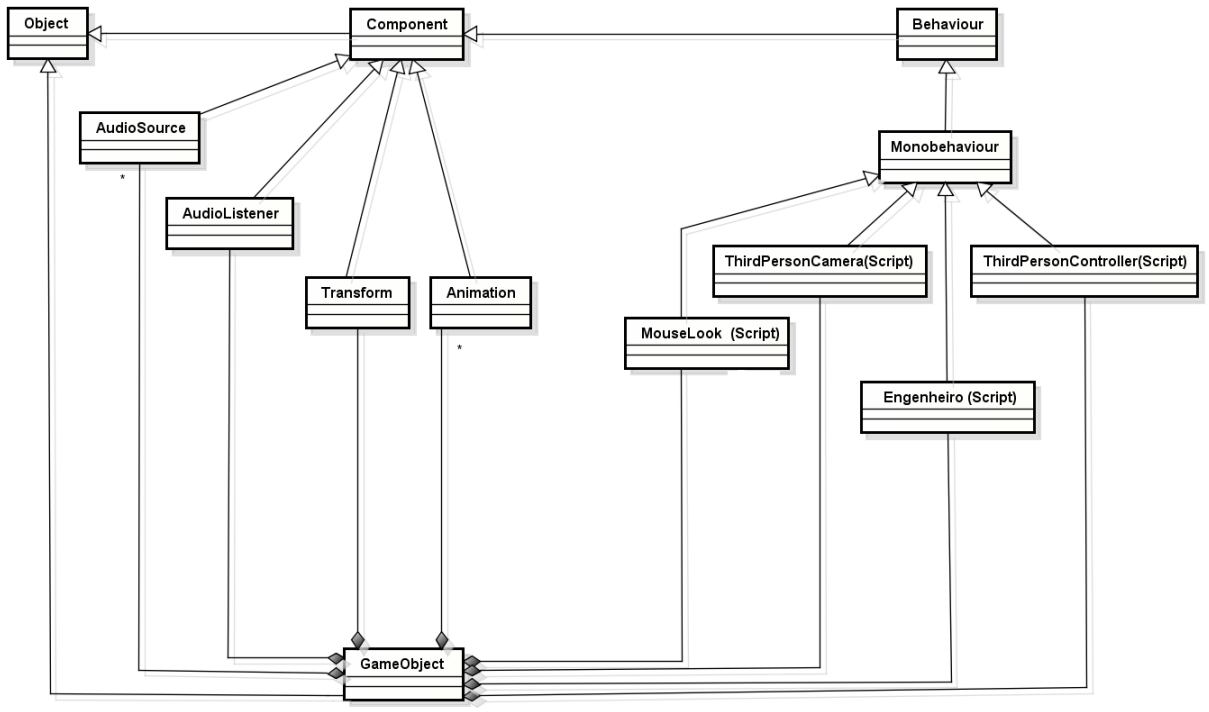

Figura 57: Diagrama de classes do *prefab* Engenheiro.

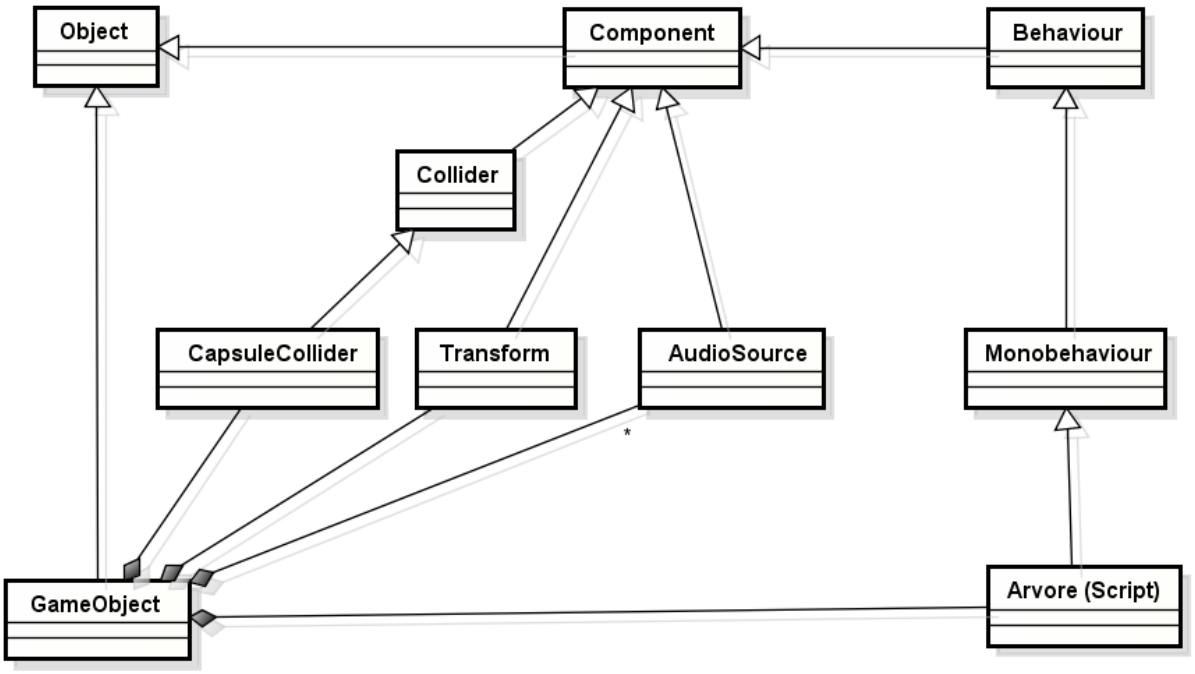

Figura 58: Diagrama de classes do *prefab* Arvore.

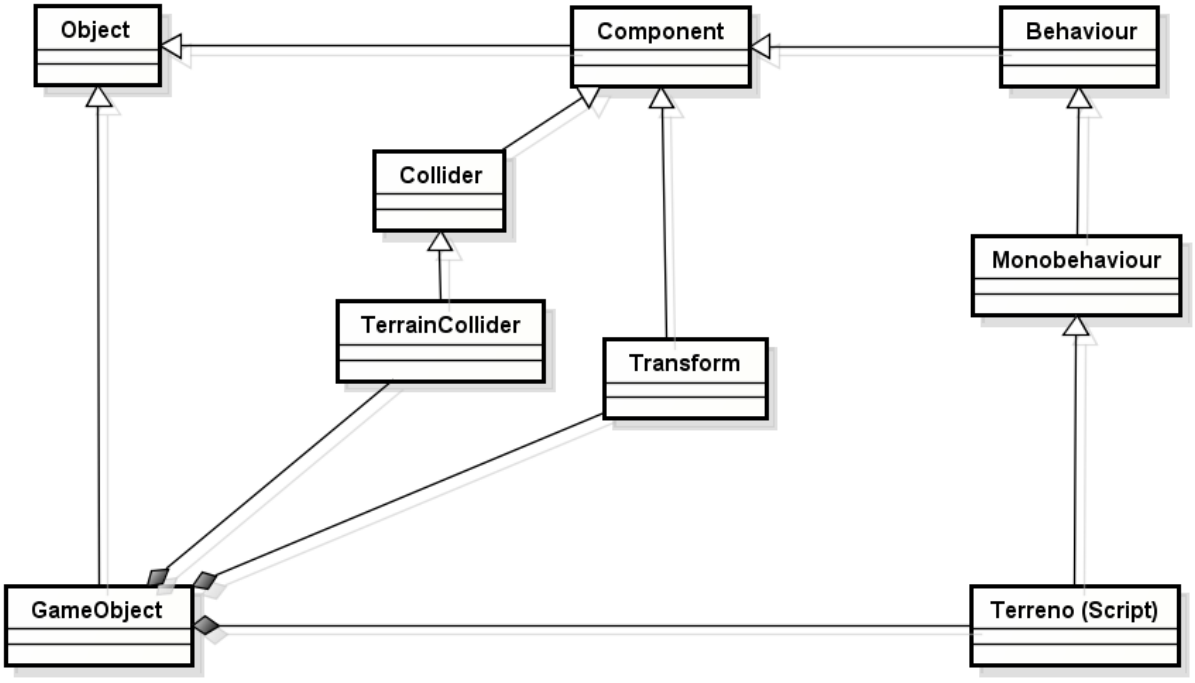

Figura 59: Diagrama de classes do *prefab* Terreno.

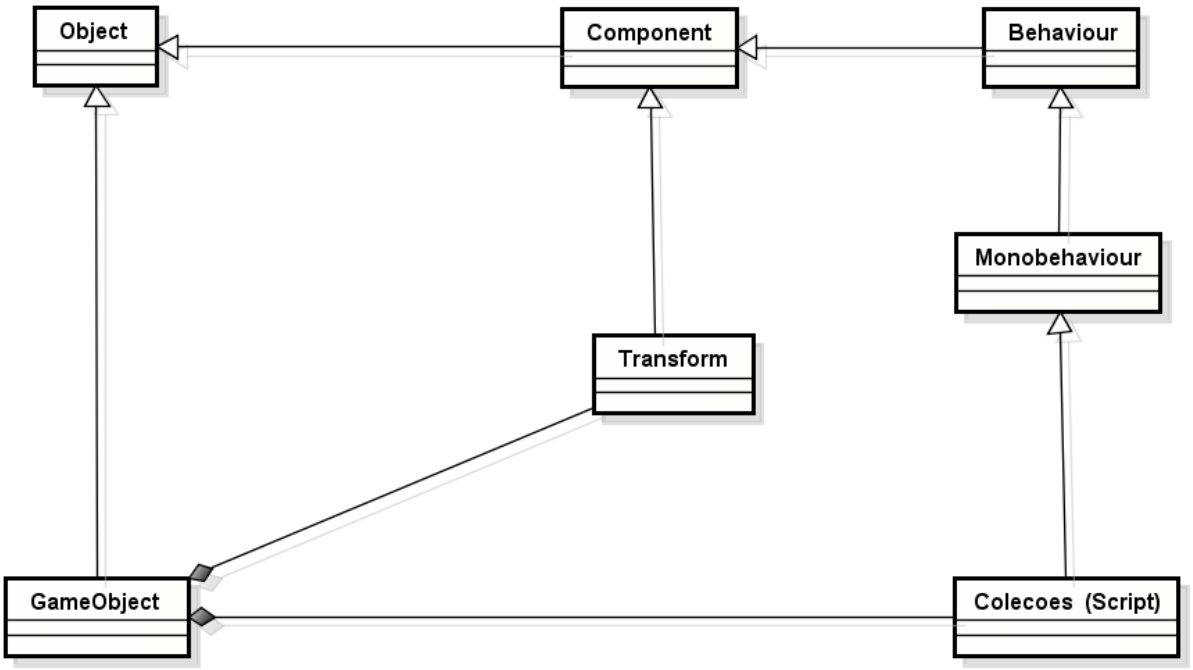

Figura 60: Diagrama de classes do *prefab* Colecoes.

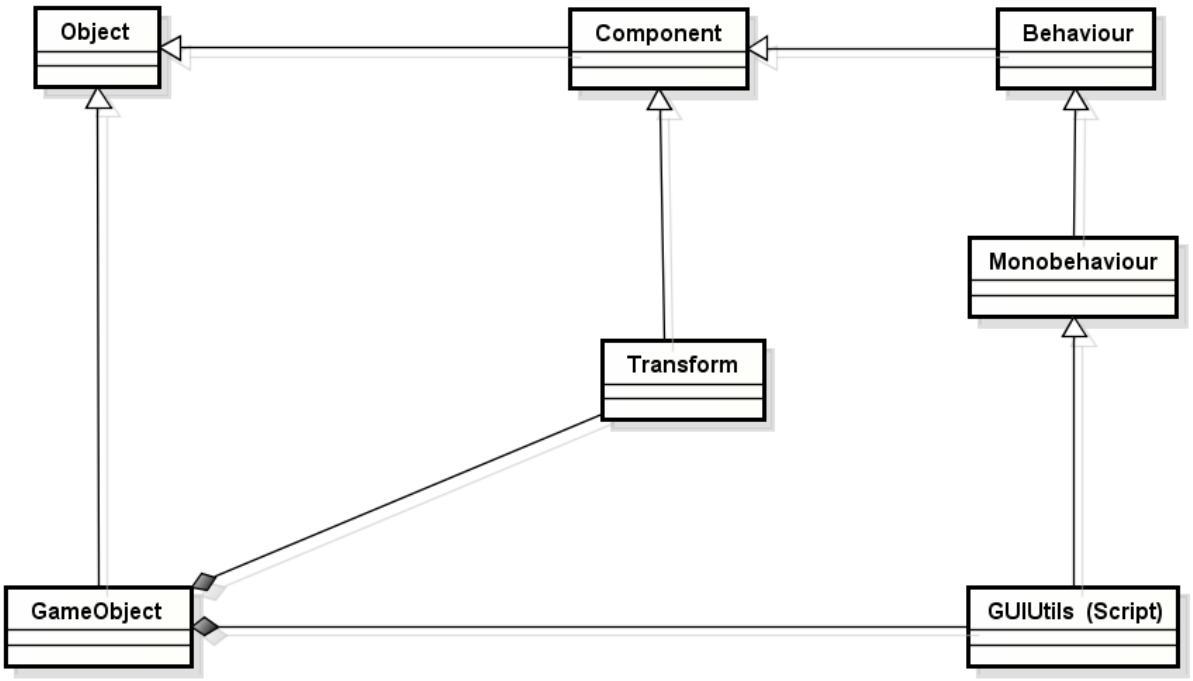

Figura 61: Diagrama de classes do *prefab* GUIUtils.

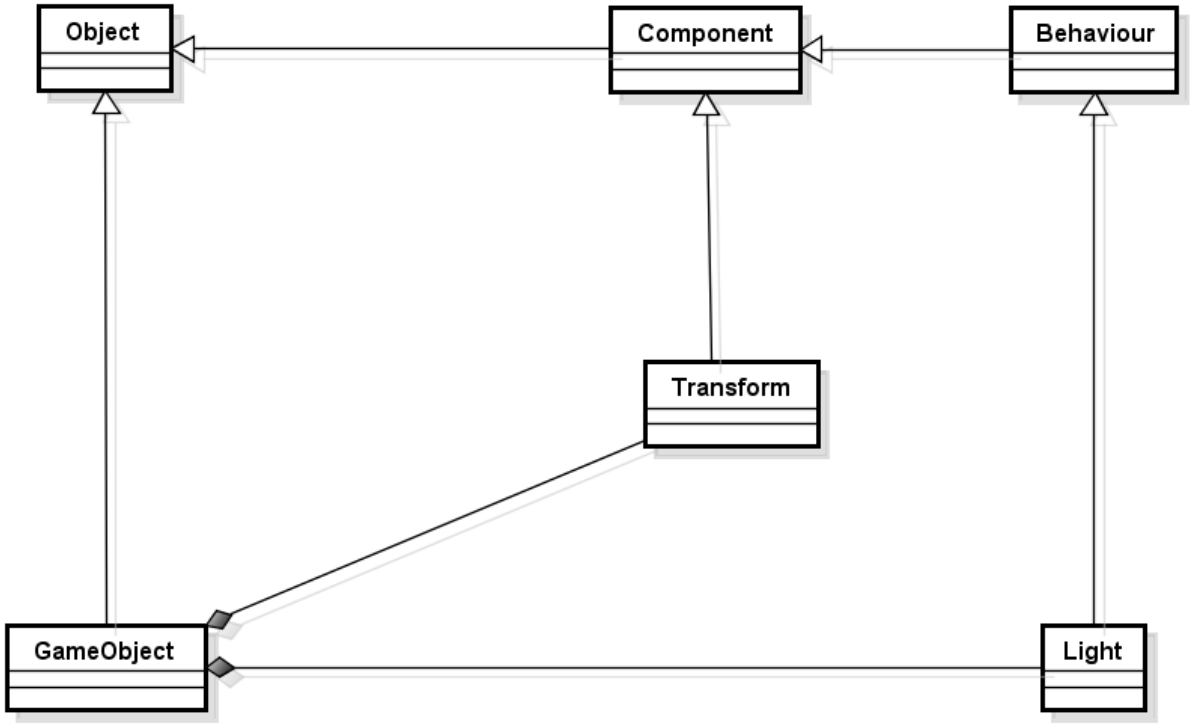

Figura 62: Diagrama de classes do *prefab* LuzDirecional.

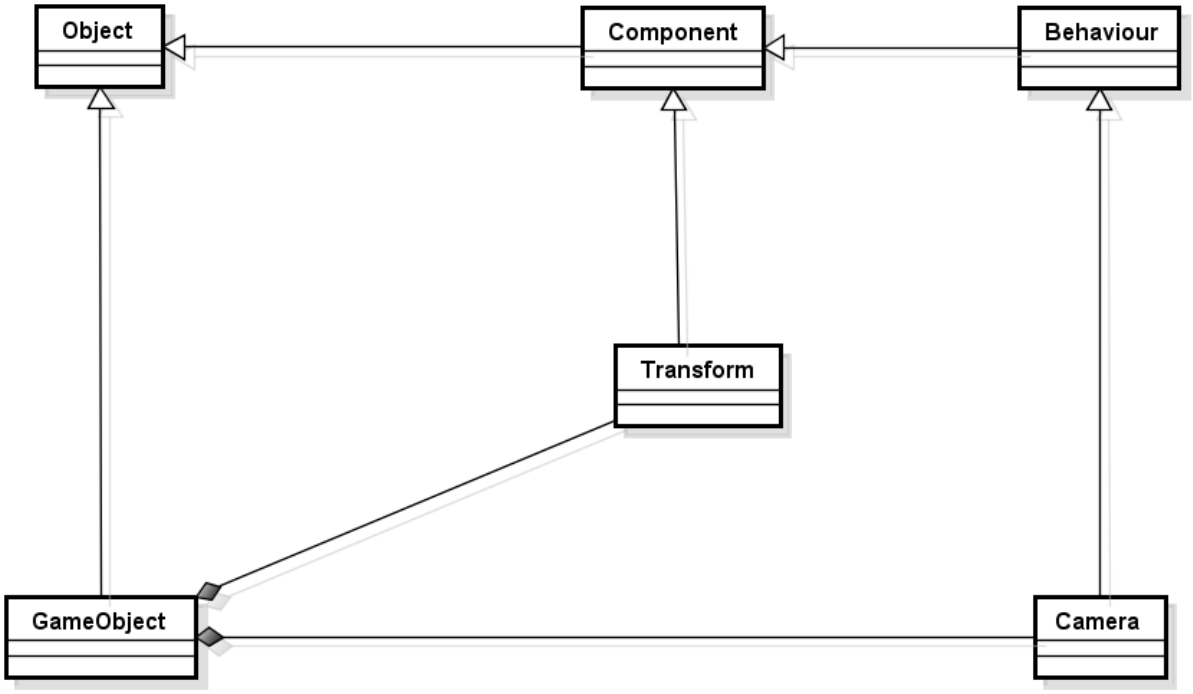

Figura 63: Diagrama de classes do *prefab* Camera.

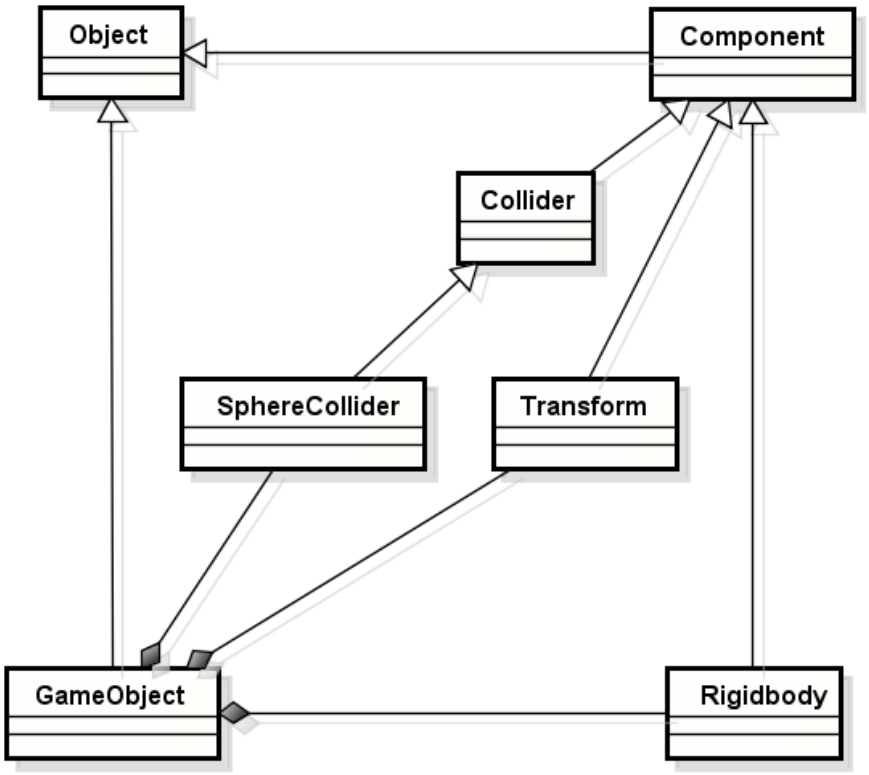

Figura 64: Diagrama de classes do *prefab* Sphere.

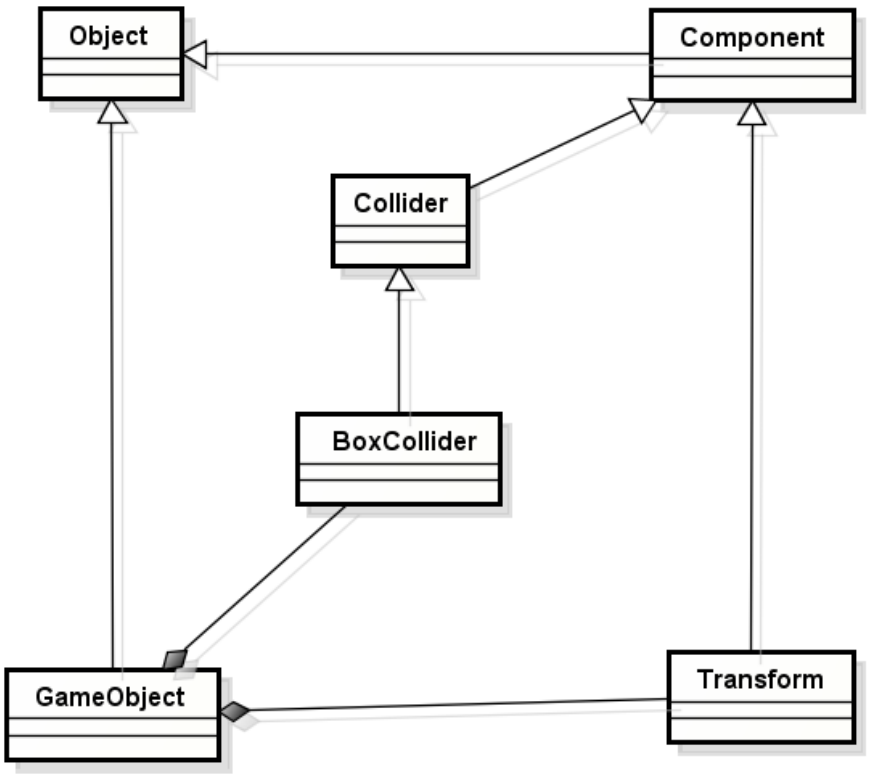

Figura 65: Diagrama de classes do *prefab* TreeStump.

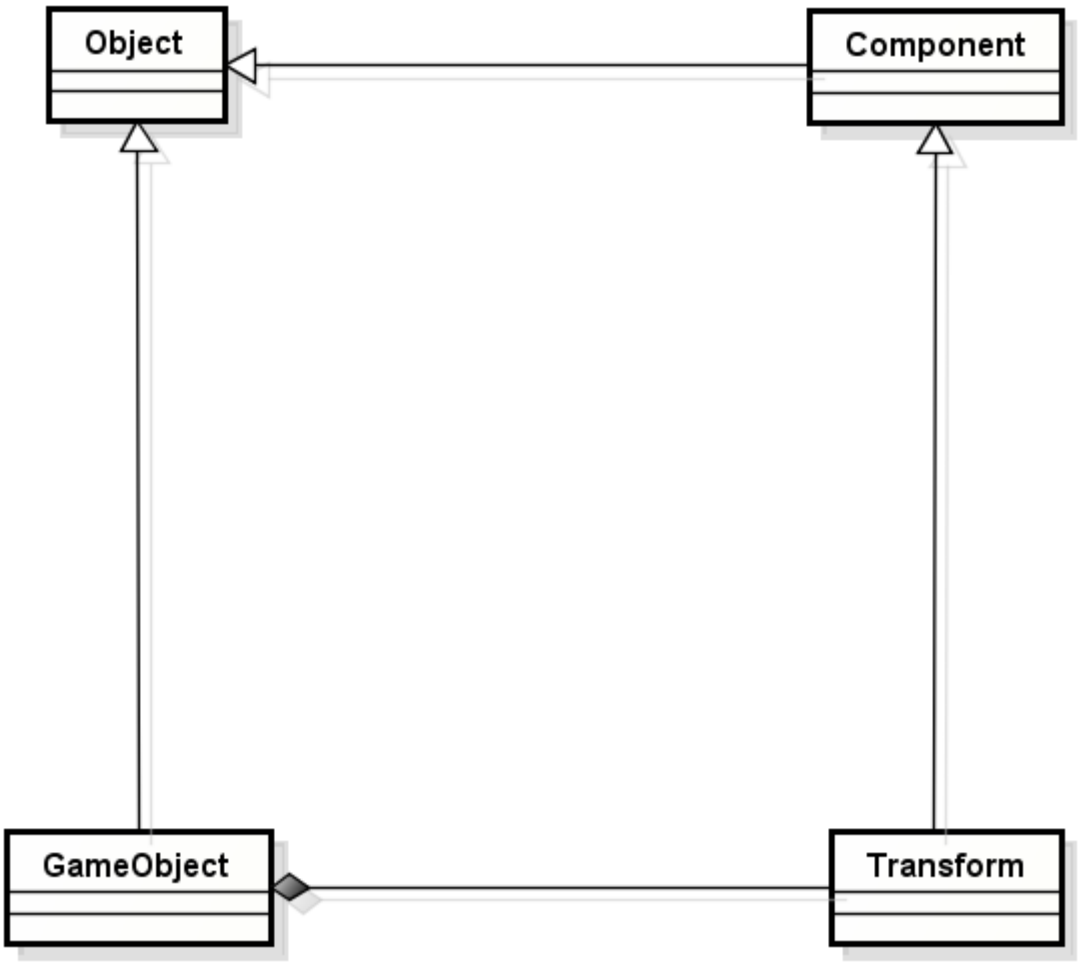

Figura 66: Diagrama de classes para os *prefabs* Suta, Motosserra, Hipsometro, ArvoreCenario, MarcaDap, MarcaAltura e Marcador.

Diagrama de Classes de Implementação do simulador

Definidos todos os prefabs, temos um conjunto de classes que se relacionam entre si. Estes relacionamentos são mostrados no diagrama abaixo. Novamente, por questões de legibilidade os métodos e atributos foram omitidos e podem ser consultados em seções anteriores deste documento.

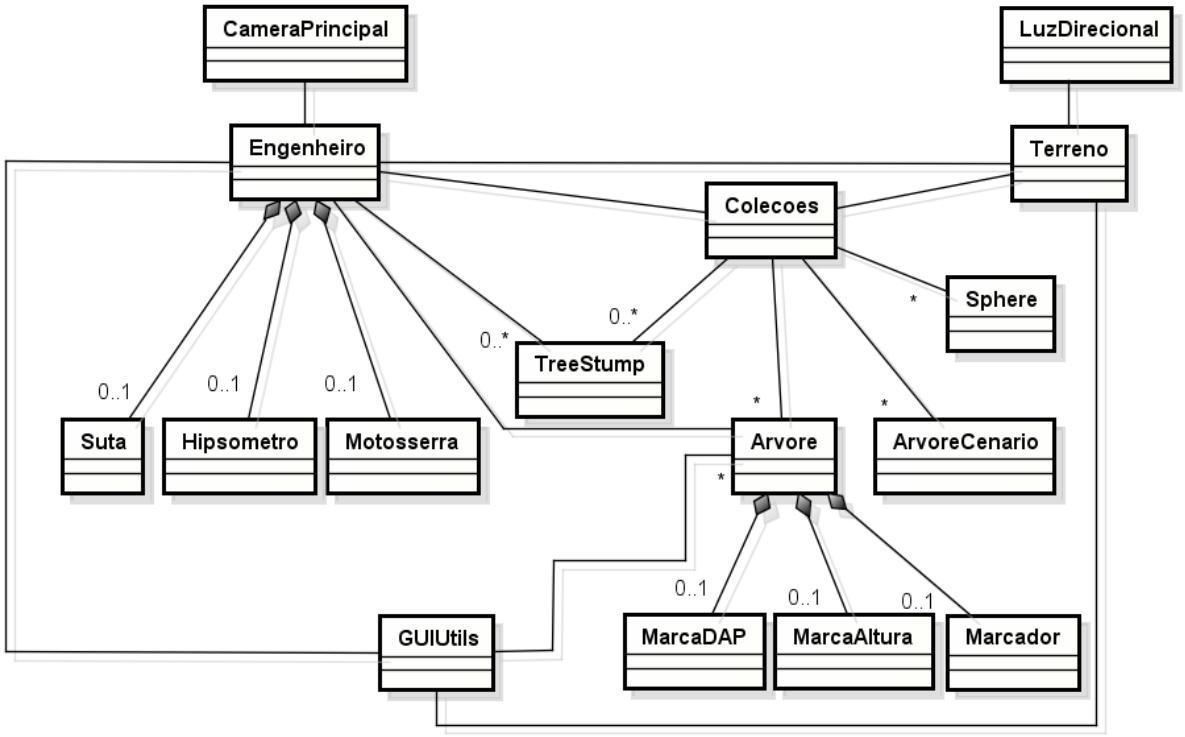

Figura 67: Diagrama de Classes de Implementação do simulador.

Diagramas de sequência de implementação

Os diagramas de sequência abaixo foram criados a partir das classes de implementação do simulador e das classes de implementação do servidor. Os métodos chamados pelo simulador são sempre os dos *scripts* associados a cada objeto, pois a partir destes métodos que podemos acessar variáveis e métodos de *component* e classes que formam um *prefab*.

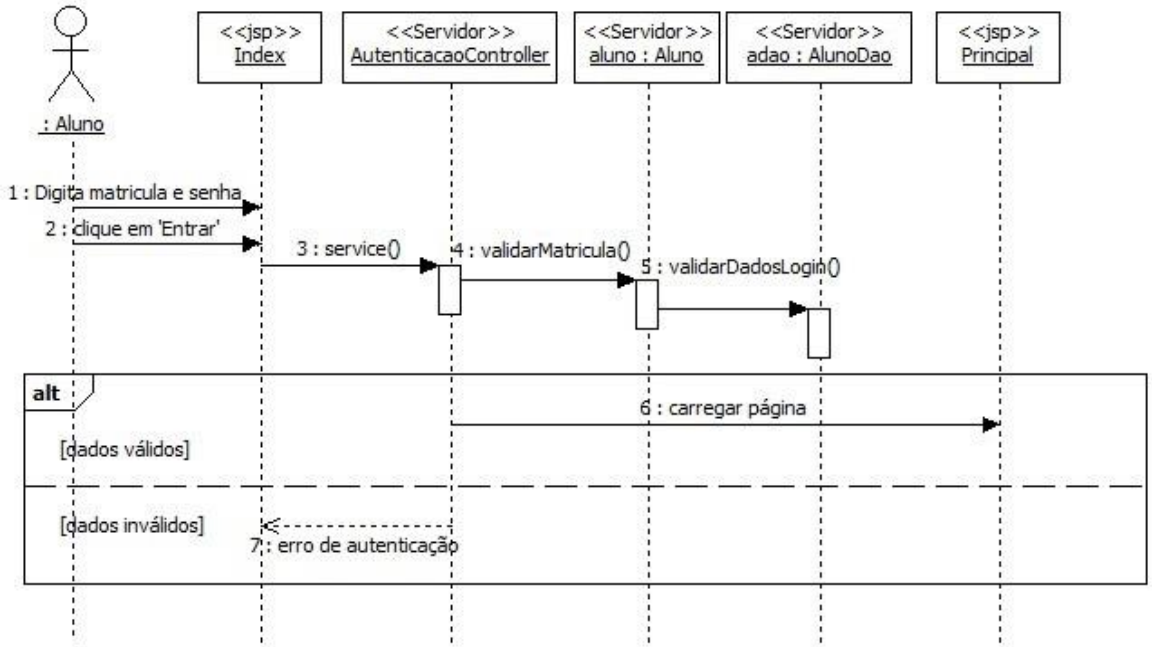

Figura 68: Diagrama de sequência de implementação da funcionalidade login.

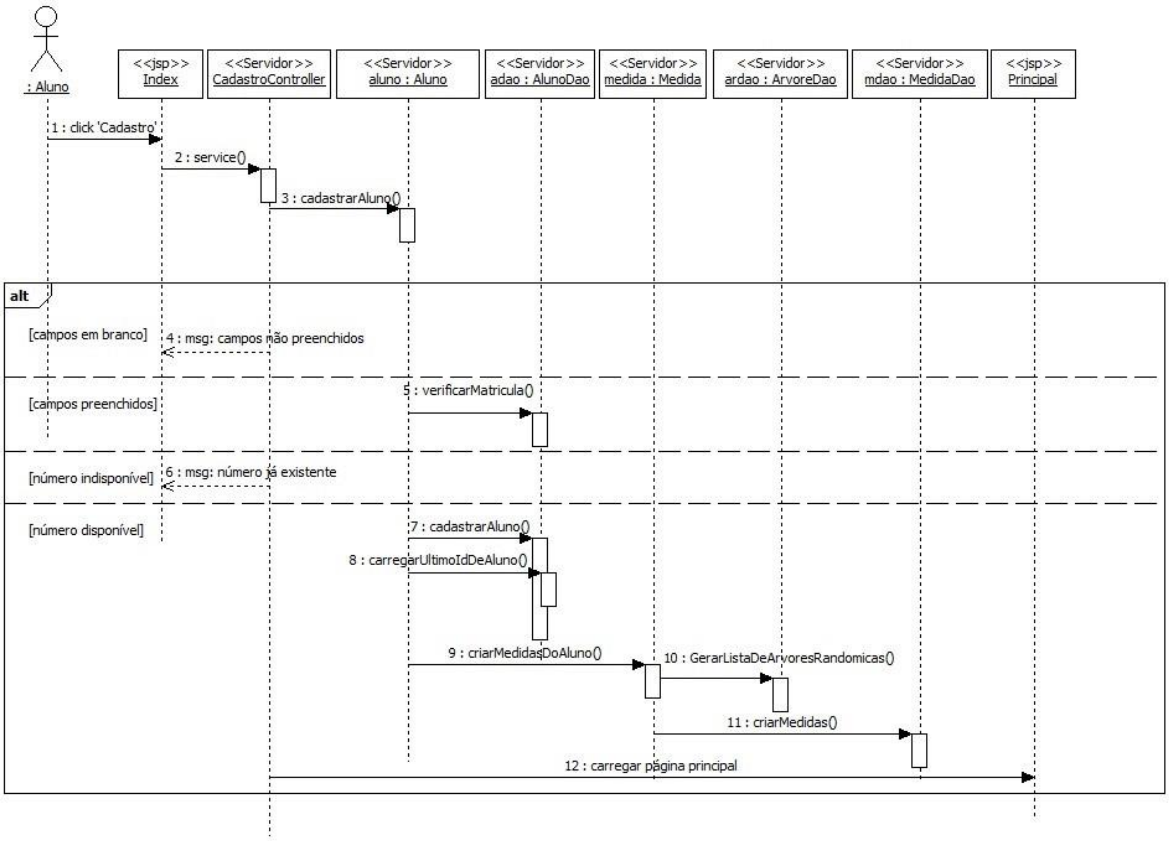

Figura 69: Diagrama de sequência de implementação da funcionalidade cadastro.

Pela sua extensão, o diagrama a seguir para a funcionalidade de carregar os dados do simulador foi dividido em quatro partes. Segue a primeira parte na página seguinte.

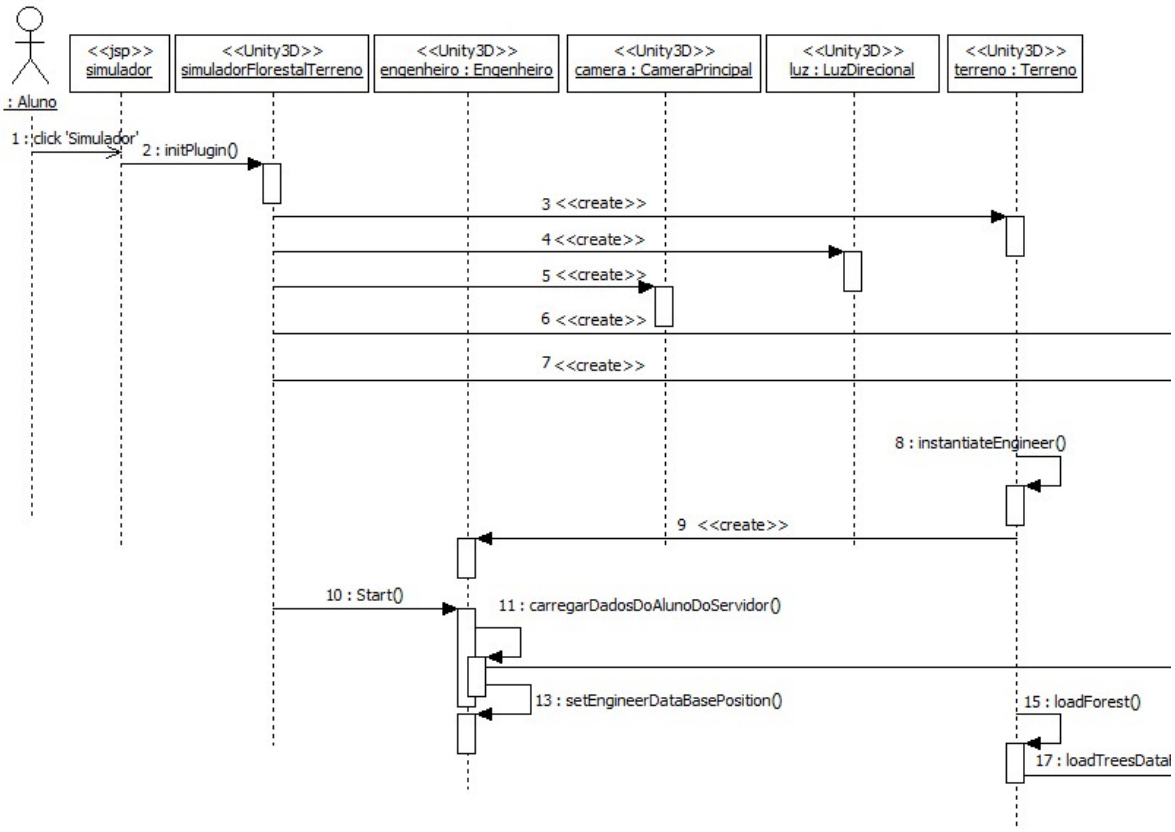

Figura 70: Diagrama de sequência de implementação da funcionalidade carregar dados do simulador - parte 1.

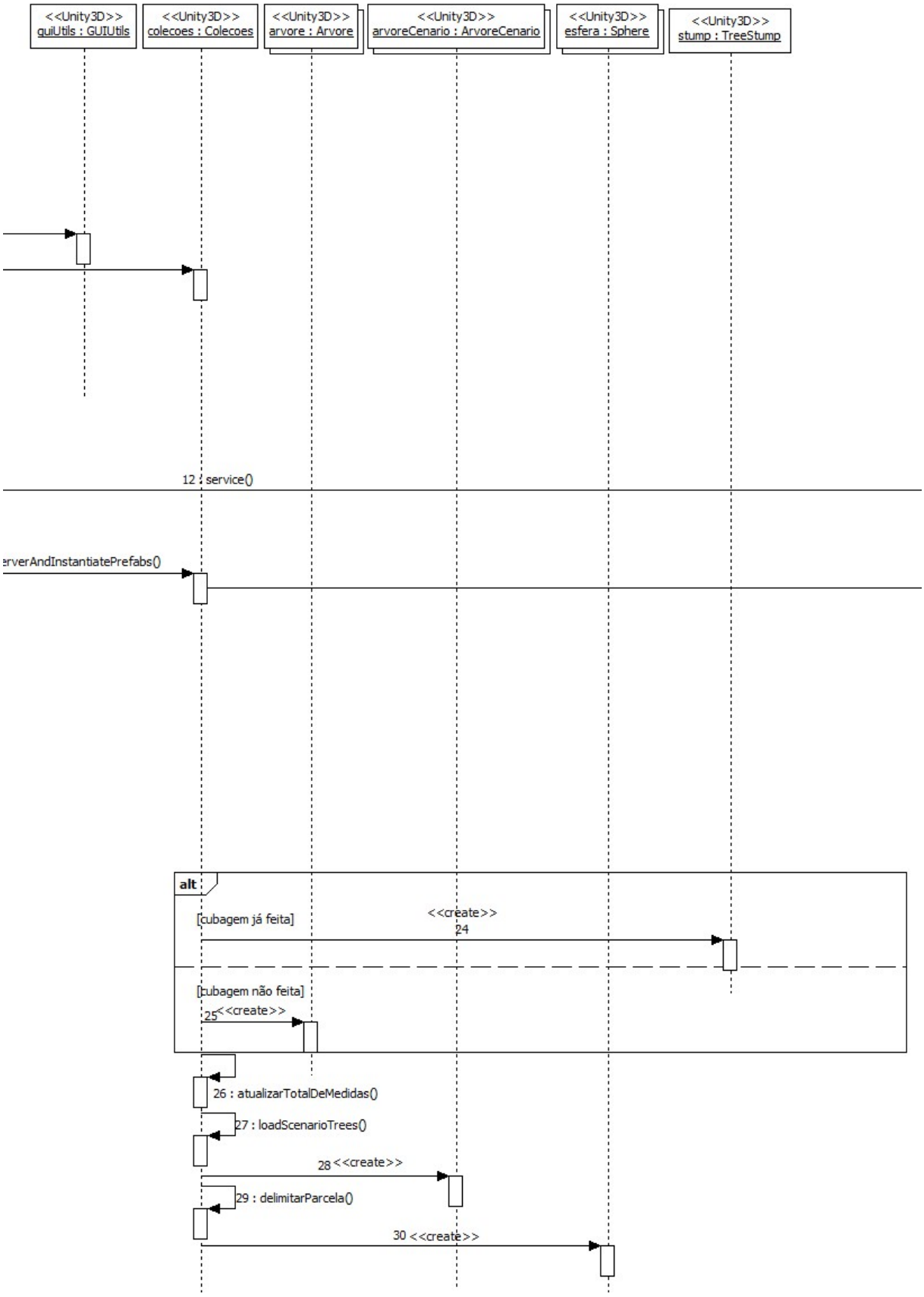

Figura 71: Diagrama de sequência de implementação da funcionalidade carregar dados do simulador - parte 2.

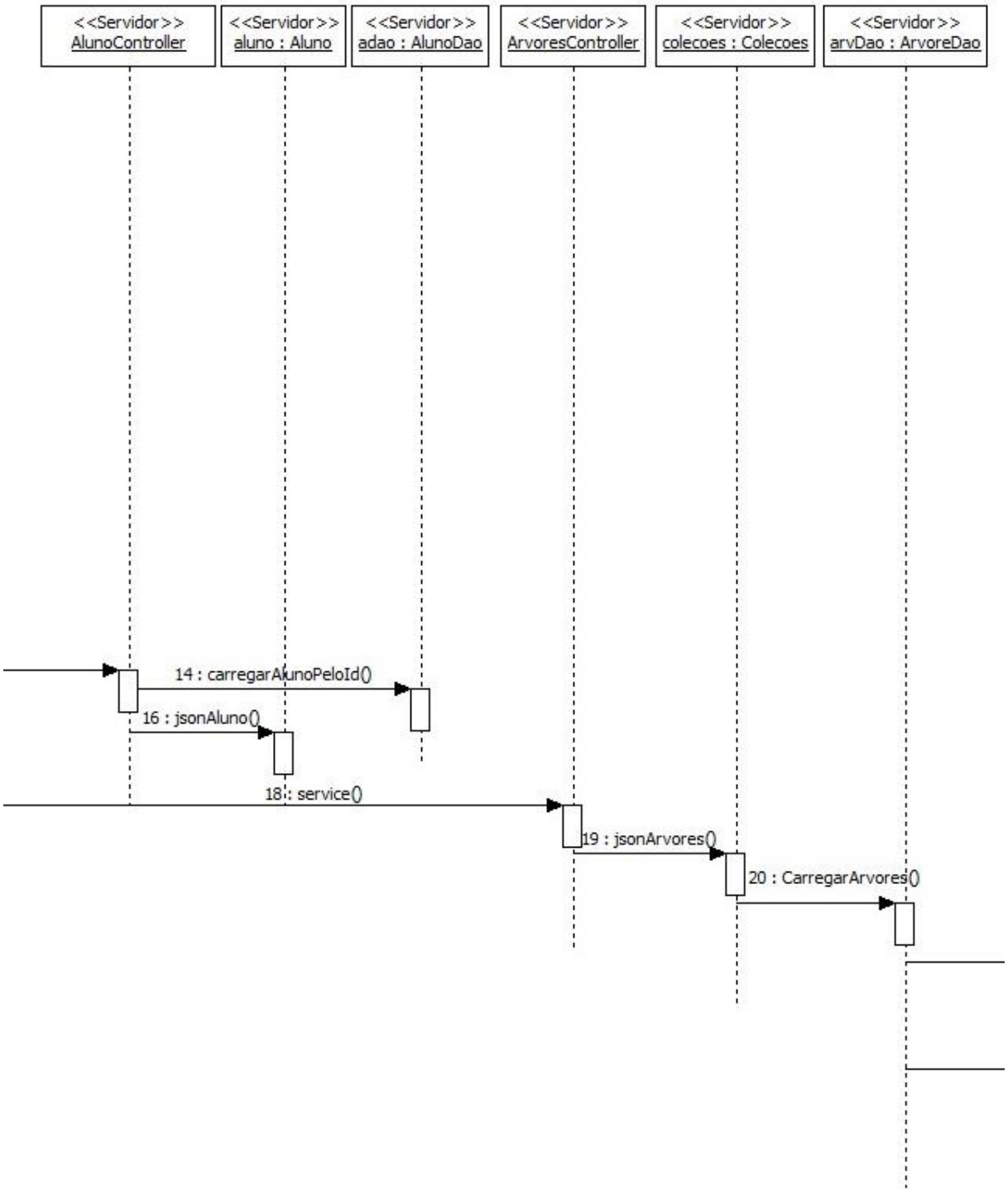

Figura 72: Diagrama de sequência de implementação da funcionalidade carregar dados do simulador - parte 3.

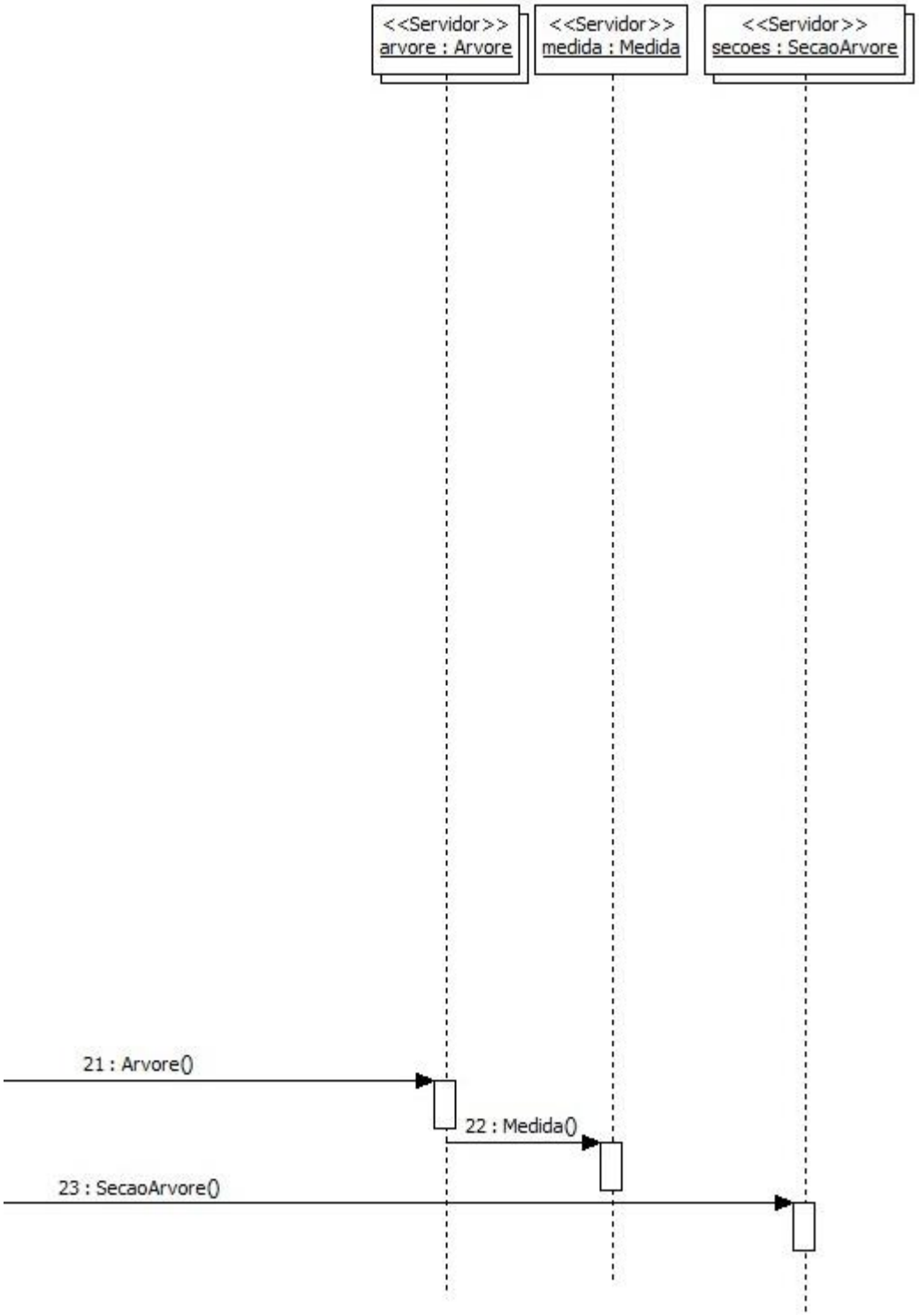

Figura 73: Diagrama de sequência de implementação da funcionalidade carregar dados do simulador - parte 4.

Selecionar Árvore

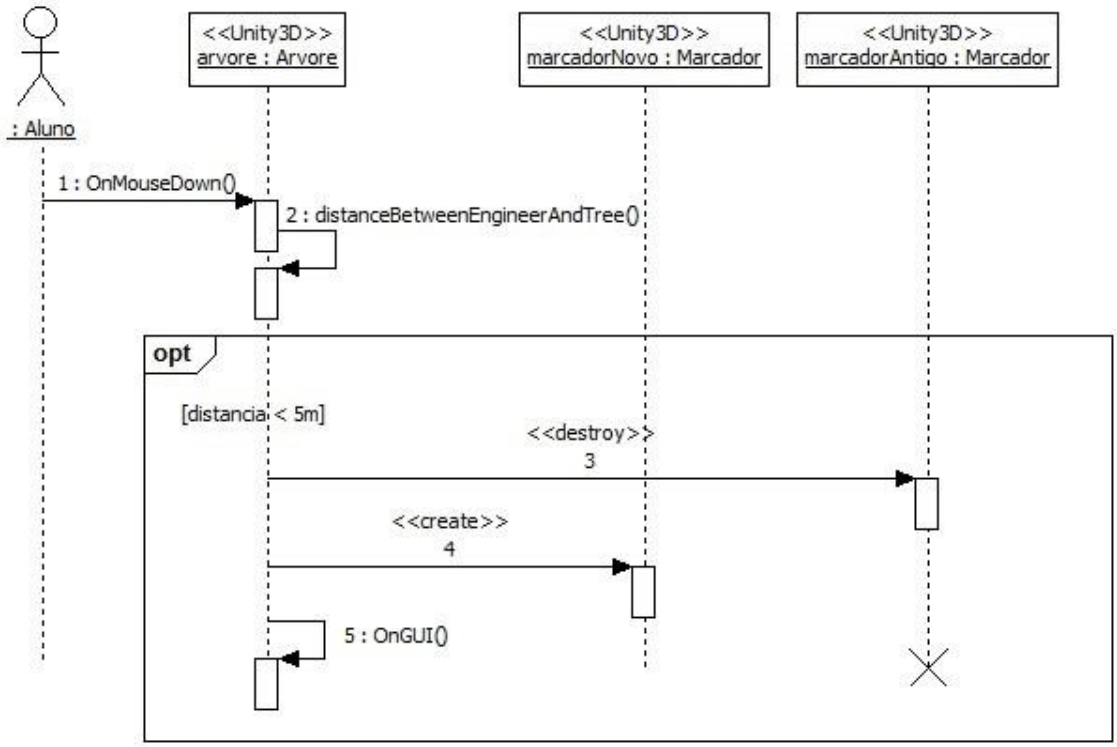

Figura 74: Diagrama de sequência de implementação da funcionalidade de seleção de árvore.

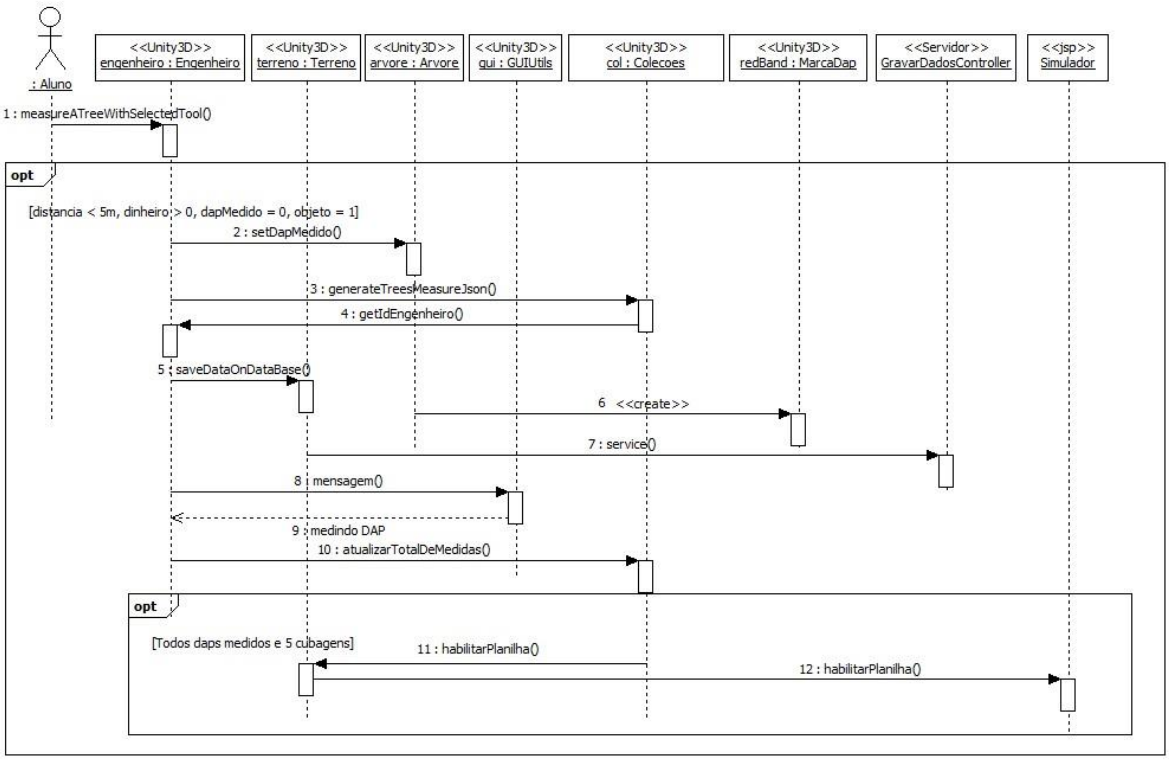

Figura 75: Diagrama de sequência de implementação da funcionalidade de medir DAP.

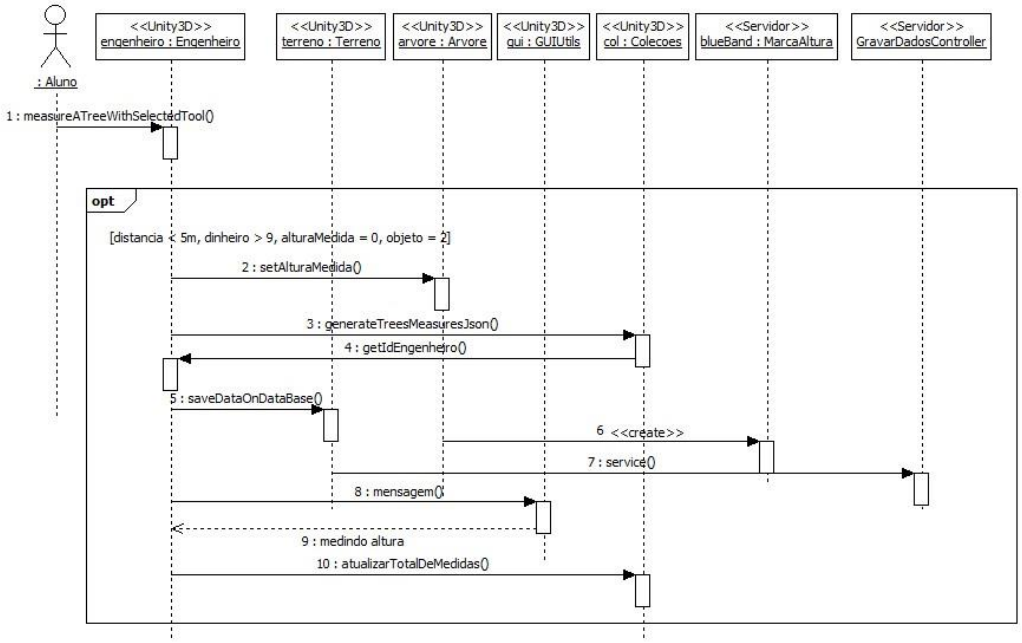

Figura 76: Diagrama de sequência de implementação da funcionalidade de medir altura.

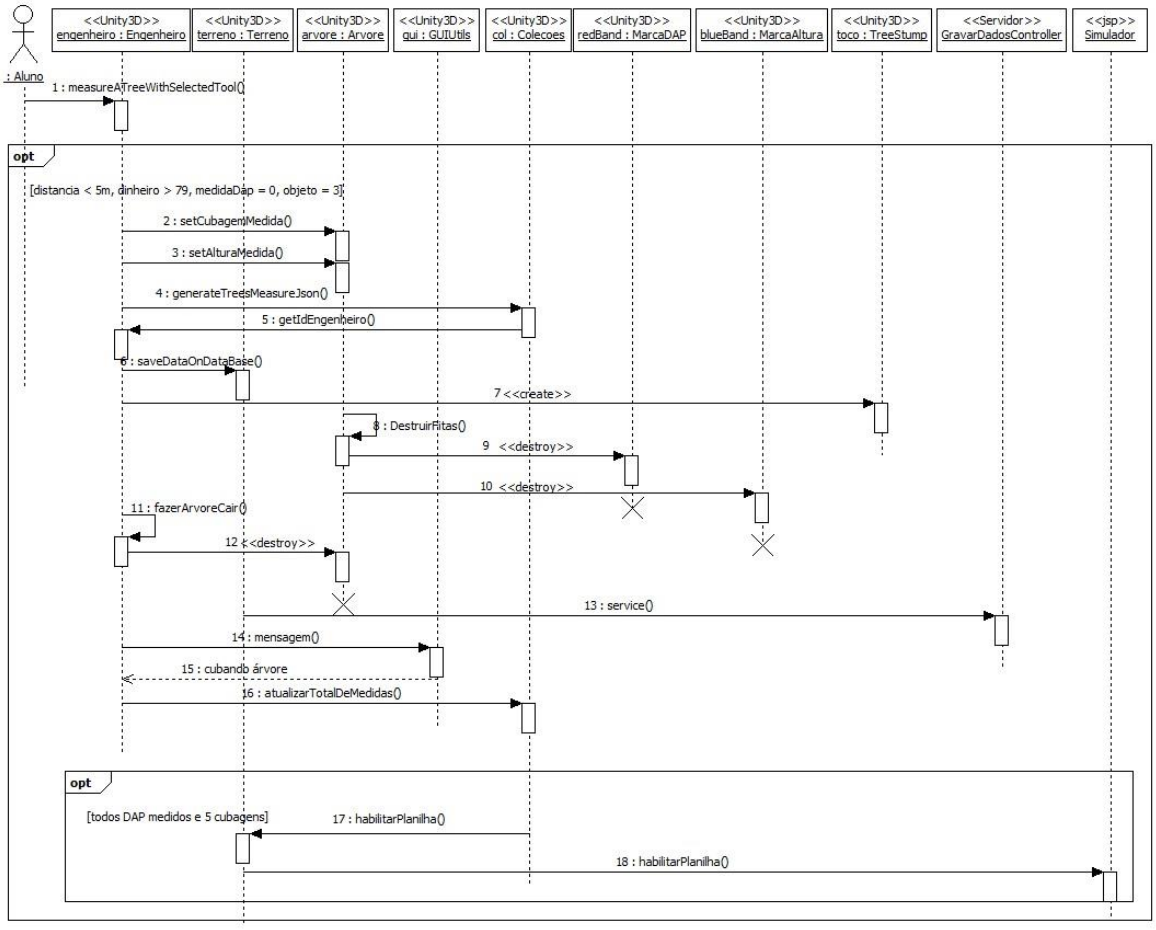

Figura 77: Diagrama de sequência de implementação da funcionalidade de realizar cubagem.

Os três diagramas anteriores, referentes às medidas, terminam com o diagrama abaixo, que faz a persistência no banco de dados por meio do servidor.

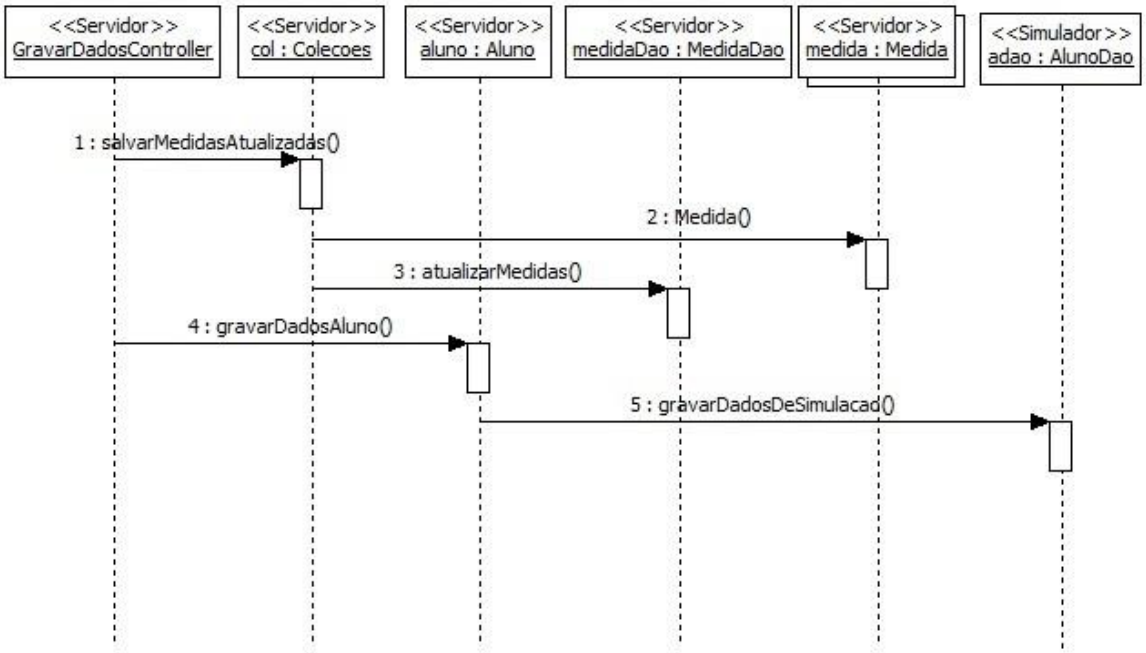

Figura 78: Diagrama de sequência de implementação da funcionalidade de gravar dados.

Resetar Dados

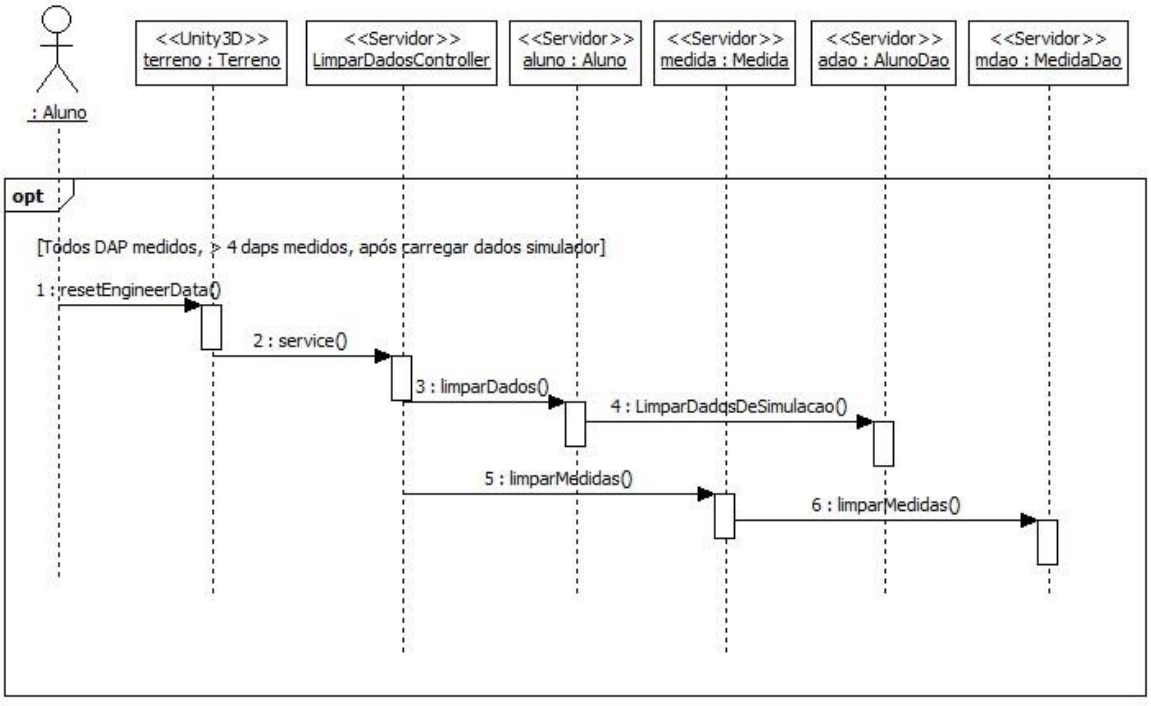

Figura 79: Diagrama de sequência de implementação da funcionalidade de resetar dados.

COMANDOS SQL DO SISTEMA

Cadastro de aluno: INSERT INTO aluno (id_aluno, nome, matricula, senha, sexo, estado, tipoUsuario, dinheiro) VALUES(0, ?, UPPER(?), ? ,?, 0, 0, 2000)

Validação de dados de login: SELECT count(*) as count FROM aluno WHERE matricula = ? AND senha = ?

Carregar aluno: SELECT * FROM aluno WHERE id_aluno = ?

Verificar matrícula: SELECT * FROM aluno WHERE matricula = ?

Ler último id de aluno cadastrado: SELECT id_aluno FROM aluno ORDER BY id_aluno DESC LIMIT 1

Gravar os dados de simulação: UPDATE aluno SET estado = 1 , dinheiro = ? , posicao_x = ? , posicao_y = ? , posicao_z = ? WHERE id_aluno = ?

Atualizar estado: UPDATE aluno SET estado = ? WHERE id_aluno = ?

Limpar dados de simulação: UPDATE aluno SET estado = 0 , dinheiro = 2000 , posicao_x = 0.0 , posicao_y = 0.0 , posicao_z = 0.0 WHERE id_aluno = ?

Contar número de árvores na base de dados: SELECT COUNT(id_arvore) as count FROM arvore

Gerar lista de árvores randômicas: SELECT DISTINCT id_arvore FROM arvore ORDER BY RAND() LIMIT 0, ?

Carregar árvores: SELECT a.id_arvore, b.numero, a.dap, a.altura, b.medida_dap, b.medida_altura, b.medida_cubagem FROM arvore a INNER JOIN medida b ON a.id_arvore = b.id_arvore WHERE b.id_aluno = ?

Criar medidas: INSERT INTO medida (id_medida, id_arvore, id_aluno, numero) VALUES(0, ?, ? ,?)

Atualizar medidas: UPDATE medida SET medida_dap = ? , medida_altura = ? , medida_cubagem = ? WHERE numero = ? and id_aluno = ?

Limpar medidas: UPDATE medida SET medida_dap = 0 , medida_altura = 0 , medida_cubagem = 0 WHERE id_aluno = ?

Carregar sessões das árvores: SELECT numero, diametro, altura FROM secoes WHERE arvore_id = ?

JSONS USADOS PARA TROCA DE INFORMAÇÕES ENTRE SIMULADOR E SERVIDOR

Exemplo de resposta do servlet AlunoController ao simulador

{ "idAluno":"7", "nome":"Gilgamesh", "matricula":"GRR20130777", "sexo":"masculino", "estado":"4", "dinheiro":"519", "posicaoX":"1245.5", "posicaoY":"7.73398", "posicaoZ":"1284.07", "dataEHora":"2013-11-29 02:57:50" }
Exemplo de resposta do servlet ArvoresController ao simulador (apenas uma árvore de um total de 50) { "alunoID":"7", "listaArvores":[{ "id":"43", "numero":"1", "coordenadaX":"0.0", "coordenadaY":"0.0", "dapcc":"28.8", "dapsc":"0.0", "altura":"26.2", "dapccMedida":"false", "dapscMedida":"false", "alturaMedida":"false", "cubagemFeita":"false", "secoes":[{ "numero":"1", "altura":"0.1", "diametro":"34.0" }, { "numero":"2", "altura":"0.3", "diametro":"32.0" }, { "numero":"3", "altura":"0.7", "diametro":"30.5" }, { "numero":"4", "altura":"1.3", "diametro":"28.8" }, { "numero":"5", "altura":"1.31", "diametro":"28.9" }, { "numero":"6", "altura":"2.62", "diametro":"27.2" }, { "numero":"7", "altura":"3.93", "diametro":"26.0" }, { "numero":"8", "altura":"6.55", "diametro":"22.9" }, { "numero":"9", "altura":"9.17", "diametro":"21.7" }, { "numero":"10", "altura":"11.79", "diametro":"19.7" }, { "numero":"11", "altura":"13.1", "diametro":"20.0" }, { "numero":"12", "altura":"14.41", "diametro":"19.4" }, { "numero":"13", "altura":"17.03", "diametro":"16.6" }, { "numero":"14", "altura":"19.65", "diametro":"13.8" }, { "numero":"15", "altura":"22.27", "diametro":"9.1" }, { "numero":"16", "altura":"24.89", "diametro":"4.1" }] }] }

Exemplo de JSONs enviados para o servlet GravarDadosController

{ "idAluno":"7", "estado":"1", "dinheiro":"509", "posicaoX":"1248.858", "posicaoY":"7.565121", "posicaoZ":"1282.299", }

para gravar dados de um aluno e

{ "alunoID":7, "listaArvores":[{ "id":"1", "dapccMedida":"false", "dapscMedida":"false", "alturaMedida":"false", "cubagemFeita":"false" }, { "id":"2", "dapccMedida":"false", "dapscMedida":"false", "alturaMedida":"false", "cubagemFeita":"false" }, { "id":"3",

"dapccMedida":"false", "dapscMedida":"false", "alturaMedida":"false", "cubagemFeita":"false" }, { "id":"4", "dapccMedida":"false", "dapscMedida":"false", "alturaMedida":"false", "cubagemFeita":"false" }, { "id":"5", "dapccMedida":"true", "dapscMedida":"false", "alturaMedida":"false", "cubagemFeita":"false" }, { "id":"6", "dapccMedida":"false", "dapscMedida":"false", "alturaMedida":"false", "cubagemFeita":"false" }, { "id":"7", "dapccMedida":"false", "dapscMedida":"false", "alturaMedida":"false", "cubagemFeita":"false" }, { "id":"8", "dapccMedida":"false", "dapscMedida":"false", "alturaMedida":"false", "cubagemFeita":"false" }, { "id":"9", "dapccMedida":"false", "dapscMedida":"false", "alturaMedida":"false", "cubagemFeita":"false" }, { "id":"10", "dapccMedida":"false", "dapscMedida":"false", "alturaMedida":"false", "cubagemFeita":"false" },] }

O trecho acima serve para gravar as medidas feitas (mostrando apenas 10 das 50).