UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA
DEPARTAMENTO DE FÍSICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENSINO DE CIÊNCIAS EXATAS

MARCO ANTONIO SPIROPULOS GONÇALVES

O *easy java simulations* como ferramenta de ensino e aprendizagem

SÃO CARLOS

2011

O *EASY JAVA SIMULATIONS* COMO FERRAMENTA DE ENSINO E APRENDIZAGEM

UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA
DEPARTAMENTO DE FÍSICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENSINO DE CIÊNCIAS EXATAS

O *EASY JAVA SIMULATIONS* COMO FERRAMENTA DE ENSINO E APRENDIZAGEM

MARCO ANTONIO SPIROPULOS GONÇALVES

Dissertação apresentada ao Programa de Pós-Graduação em Ensino de Ciências Exatas da Universidade Federal de São Carlos, como parte dos requisitos para obtenção do título de Mestre em Ensino de Ciências Exatas.

Orientador: Prof. Dr. Nelson Studart Filho

São Carlos – SP
2011

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

G635ej

Gonçalves, Marco Antonio Spiropulos.
O Easy Java Simulations como ferramenta de ensino e aprendizagem / Marco Antonio Spiropulos Gonçalves. -- São Carlos : UFSCar, 2012.
194 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2011.

1. Física - estudo e ensino. 2. Professores - formação. 3. Simulação por computador. I. Título.

CDD: 530.07 (20ª)

Banca Examinadora:

NStudart

Prof. Dr. Nelson Studart Filho
DM - UFSCar

Carlos E. Aguiar

Prof. Dr. Carlos Eduardo Magalhães de Aguiar
IF - UFRJ

Profa. Dra. Ducinei Garcia
DF - UFSCar

Dedico este trabalho:

Ao professor que desejar modificar a sua prática de ensino e estudar uma ferramenta de simulação que poderá ser a base para o Ensino de Física.

Aos filhos Marcella, Renato, Enrico e Giordano.

À minha esposa Thaís. Obrigado por tudo!

Vivemos neste mundo para nos esforçar a aprender sempre, para esclarecer uns aos outros, por meio de troca de ideias, e para nos aplicar todos os dias e cada vez mais às ciências e às artes.

Wolfgang Amadeus Mozart

AGRADECIMENTOS

Ao orientador Prof. Dr. Nelson Studart Filho pela dedicação e preocupação em desenvolver instrumentos de ensino que possam ser utilizados por professores do Ensino Médio.

Ao Prof. Dr. Carlos E. Aguiar pelas observações e sugestões apresentadas sobre o EJS durante o Workshop Internacional sobre Objetos de Aprendizagem no Ensino de Ciências e Matemática (WIOA 2010) em São Carlos (SP).

Aos professores do Programa de Pós-Graduação em Ensino de Ciências Exatas (PPGECE) pelas sugestões, discussões e instigações durante as aulas das disciplinas do programa.

Ao meu avô Jorge Martins Spiropulos (*in memoriam*).

RESUMO

O objetivo deste trabalho é apresentar a ferramenta de simulação *Easy Java Simulations* (EJS), criada por Francisco Esquembre, e mostrar ao professor do Ensino Médio, através de alguns exemplos e de um guia básico sobre o EJS, como ele pode usá-la para preparar atividades de aprendizagem diferenciadas aos seus alunos. Faz-se uma análise da potencialidade da ferramenta EJS e apresenta-se todo o procedimento básico e necessário para a construção de uma simulação, desde o modelo matemático usado para modelar a evolução do sistema físico em estudo, até a preparação da interface gráfica que permite a visualização dos elementos gráficos. Como produto final da dissertação do Mestrado Profissional, apresenta-se um guia básico do EJS para aquele professor que estiver disposto e motivado em aprender os rudimentos do EJS e aplicá-los em sala de aula. Espera-se que este trabalho seja um convite para o professor melhorar a sua prática profissional, rompendo-se com uma formação ambiental, de caráter repetitivo, e que, a partir de um trabalho contínuo de estudo e de reflexão, ele possa tornar o Ensino de Física no Ensino Médio eficiente, já que uma parcela significativa dos alunos demonstra pouco, ou nenhum, interesse pelo estudo de Física.

Palavras-chave: EJS. *Easy Java Simulations*. Ensino de Física. Formação de Professores. Simulação de Física.

ABSTRACT

The aim of this dissertation is to introduce a simulation tool called *Easy Java Simulations* (EJS), created by Francisco Esquembre, and also to provide guidelines to Physics teacher at the High School level in how it could be used by promoting differentiated learning activities for his students. An analysis of the strength of the tool EJS is performed and it is introduced the whole basic and necessary procedure for constructing a simulation, from the mathematical modeling of the evolution of the physical system until the elaboration of the interface that allows the visualization of the graphic elements. As a final product of the dissertation, it is presented an introductory guide to EJS for all teachers interested in learning and applying it within the classroom. It is intended that this work could motivate the teacher to improve his professional practice, breaking up with the actual classroom environment, based essentially on lectures and recitation classes, and through a continuous work of study and reflection, he could turn Physics teaching more efficient at the High School, since a significant portion of the students has shown little or none interest in studying Physics.

Keywords: EJS. Easy Java Simulations. Physics Teaching. Teacher Preparation. Physics Simulation.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Durante as conversas informais entre os professores de Física do Ensino Médio, principalmente aquelas que ocorrem durante os intervalos de aulas, ouve-se com frequência reclamações do tipo: “os alunos não se esforçam em aprender Física”, “eles não conseguem prestar atenção na explicação de um conteúdo”, “não têm capacidade de aprender Física, pois a matéria é para poucos” ou, ainda, “os alunos envolvem-se facilmente em brincadeiras, gerando indisciplina na sala de aula”. Por sua vez, os alunos se queixam que a Física “é uma matéria chata”, “não serve para nada”, “tem muitas fórmulas para serem decoradas” e “não se relaciona com o dia-a-dia”. Diante dessas reclamações é inevitável indagar: “Por que uma parcela significativa dos nossos alunos do Ensino Médio não aprende Física?”

Para responder à indagação, procura-se neste trabalho oferecer subsídios para o professor de Física de Ensino Médio melhorar sua prática docente e, assim, responder parcialmente a indagação acima. Entende-se por docência de qualidade: aquisição de conhecimentos teóricos sobre as abordagens de ensino e aprendizagem, conhecimentos da matéria a ser ensinada, capacidade de preparar atividades, saber orientar trabalhos para serem desenvolvidos por alunos e elaborar avaliações.

Esse trabalho de dissertação oferece alguns subsídios teóricos e sugestões para o educador refletir sobre a sua ação pedagógica, bem como recursos mínimos para ele desenvolver atividades de simulação junto aos seus alunos.

Acreditamos que simulações planejadas e estruturadas, desenvolvidas pelo próprio professor, isto é, aquele professor que conhece e trabalha junto aos seus alunos, possam contribuir com eficiência, e significativamente, na aprendizagem da Física. Espera-se que no decorrer do processo de ensino e aprendizagem, os alunos modifiquem as suas ideias pré-concebidas e considerem a Física como uma disciplina que realmente contribui para a melhoria das condições de vida das pessoas e no desenvolvimento tecnológico das mais diferentes áreas.

O desenvolvimento desse trabalho está relacionado com a nossa experiência, enquanto professor de Física do Ensino Médio, na revisão bibliográfica sobre as metodologias de ensino, das abordagens de aprendizagens utilizadas e no desenvolvimento das atividades realizadas por professores do Ensino Médio em suas práticas profissionais. Faz-se, também, uma análise sobre a possibilidade do uso de simulações via a ferramenta *Easy Java*

Simulations, abreviadamente EJS. Nela é apresentada a ferramenta EJS, suas potencialidades e como o professor deve proceder para criar suas próprias simulações de Física.

Como produto final da dissertação de um Mestrado Profissional, apresentamos um guia básico do EJS para aquele professor do Ensino Médio que estiver disposto e motivado em aprender os rudimentos de EJS e da linguagem de programação *Java*. O título desse guia é "Procedimentos básicos para a criação de simulações de com o *Easy Java Simulations*". O objetivo desse texto (produto final da dissertação) é apresentar de forma didática a ferramenta EJS, sua potencialidade, algumas das propriedades dos elementos gráficos e como deve ser a estrutura da simulação. A sua elaboração e desenvolvimento é fruto da nossa leitura e interpretação do livro de Esquembre (2005), da análise do tutorial apresentado na página da *web* (http://www.um.es/fem/Ejs/), dos estudos dos exemplos que estão no próprio livro e daqueles que estão disponíveis em pastas ao se fazer o *download* do programa EJS. Algumas simulações elementares apresentadas nesse texto servirão de "roteiros" de estudos para os professores conhecerem as possibilidades e potencialidades de EJS. Acreditamos que os exemplos funcionarão como um ponto de partida para que professores, ou pessoas interessadas, dediquem-se em criar simulações através do EJS.

Essa dissertação poderá contribuir para a formação de equipes interdisciplinares (alunos, professores, cientistas da computação, pedagogos, etc.) com o objetivo de elaborar, planejar, desenvolver e executar simulações que tenham real valor pedagógico para o ensino da Física.

Espera-se, nesse contexto, que a dissertação seja um convite para se romper com a visão de que ensinar Física é algo simples e que o professor possa exercer uma docência diferenciada ao romper com o simples giz, lousa, livro didático e problemas matemáticos inventados sobre as leis físicas!

2 MODELAGEM E SIMULAÇÃO NO ENSINO DE FÍSICA NO ENSINO MÉDIO

Os computadores são elementos que afetam a forma como a sociedade se organiza, as relações de trabalho e como se aprende. Particularmente nas ciências (Física, Química, Biologia, etc.) muitos dos *hardwares* e *softwares* apresentam aplicações específicas como análise numérica, manipulação simbólica, coleta e análise de dados, simulação e visualização.

No ensino de ciências, o uso de computadores tem se apresentado como uma ferramenta que pode auxiliar na construção do conhecimento (CORTE[1] e TAYLOR[2] citados por VEIT; TEODORO, 2002, p. 87) através dos chamados laboratórios virtuais. Entretanto, Medeiros e Medeiros (2002, p. 77) apontam que "desde o início do século XX, várias ondas tecnológicas inovadoras têm assolado a educação com promessas e perspectivas mirabolantes".

Compartilhamos com o pensamento de Veit e Teodoro, citados acima, e o objetivo desse trabalho é ilustrar a potencialidade de um *software* educacional, no caso o EJS, como uma ferramenta que possa auxiliar no ensino e na aprendizagem da Física. Acreditamos que o professor seja o elemento fundamental do processo ensino/aprendizagem e, assim, ele precisa dominar os conhecimentos que permitam elaborar atividades diferenciadas. Dentre elas, particularmente, as atividades experimentais via laboratório virtual.

2.1 MODELAGEM E SIMULAÇÃO

Nas Ciências da Natureza, Engenharia, Economia e mesmo na Psicologia, os cientistas, ou pessoas envolvidas no processo de produzir conhecimento científico, tentam em geral descrever ou modelar o comportamento de um sistema (ou fenômeno) através da linguagem matemática. Segundo Chaves e Sampaio (2007, p. 16):

> O termo modelo é empregado na Ciência com dois significados. Em um deles, modelo é algo intermediário entre leis fenomenológicas e teoria científica. No outro, modelo é uma idealização simplificada de um sistema complexo, cujo comportamento supostamente reproduz na essência o do sistema complexo cujo entendimento é almejado.

---

[1] CORTE, E. de; VERSCHAFFEL, L.; LOWYCK, J. Computers and learning. In: HUSÉN, T. N. **Education:** The Complete Encyclopedia. Oxford: Pergamon Press, 1998. 1 CD.
[2] TAYLOR, R. P. **The computer in the school:** tutor, tool, tutee. New York: Teachers' College Press, 1980.

Neste contexto, a Lei da Gravitação de Newton, por exemplo, é uma forma de representar a interação gravitacional entre dois corpos através de um modelo matemático, contudo sem explicar o que é a gravidade. Através da equação matemática que expressa essa lei, pode-se representar no papel, ou no computador, o comportamento dos corpos que interagem entre si. Essa representação simplificada de um sistema, como a interação entre dois corpos mantendo-se suas características essenciais, é feita através de entes matemáticos como funções, tabelas e figuras geométricas.

Basicamente, a descrição matemática do modelo consiste em:

a. Identificar as variáveis que são responsáveis por variações, ou mudanças, do sistema;
b. Levantar hipóteses sobre o sistema.

Nas hipóteses podem ser consideradas leis empíricas aplicáveis ao sistema. A estrutura matemática dessas hipóteses, que se constitui no modelo matemático do sistema, é muitas vezes uma equação diferencial ou um sistema de equações diferenciais (ZILL; CULLEN, 2008, p. 13). Um modelo matemático aceitável do sistema deve apresentar uma solução consistente com o comportamento observado do sistema. As soluções podem ser exatas, aproximadas, numéricas ou, ainda, a equação diferencial pode não apresentar uma solução por falta de fundamentos físicos, matemáticos ou ambos (MACHADO, 2004).

Existem modelos que estabelecem relações entre grandezas físicas e o tempo, considerado como variável independente. A solução matemática da equação diferencial que descreve esse tipo de sistema, também denominado de sistema dinâmico, representa o estado do sistema. Isso significa que para valores apropriados do tempo, os valores da variável dependente descrevem o sistema no passado, presente e futuro (ZILL; CULLEN, 2008, p. 14). Esses modelos aparecem na maior parte dos conteúdos de Física dos currículos do Ensino Médio e Universitário.

Embora a modelagem do fenômeno possa ser efetuada sem a utilização do computador, o seu uso permite aos estudantes resolverem problemas difíceis e que demandam tempo. Além do mais, a combinação entre modelagem computacional, teoria e atividades experimentais de laboratório pode levá-los a alcançar um nível de compreensão que não poderia ser alcançado de outra forma (CHRISTIAN; ESQUEMBRE, 2007). Para esses pesquisadores, a modelagem computacional demanda:

1. A descrição e a análise do problema.
2. A identificação das variáveis e os algoritmos utilizados.
3. A implementação da plataforma específica para *hardware* e *software*.
4. A execução da implementação e análise dos resultados.

5. Refinamento e generalização dos métodos computacionais empregados.
6. Apresentação dos resultados.

Para esses pesquisadores, uma simulação computacional é a implementação de um modelo que permite testá-lo com diferentes condições iniciais e com o objetivo de aprender sobre o comportamento do modelo. A aplicabilidade dos resultados da simulação para o sistema físico real depende do quanto o modelo descreve a realidade. Esquembre (2005, p. 10) salienta que:

> Las simulaciones son programas de computador que contienen un modelo de un sistema o proceso físico y que están dedicados a la visualización gráfica de este. Además, el programa invita a los estudiantes a explorar e interactuar con el sistema modificando su estado, cambiando parámetros y observando el resultado de esta manipulación.[3]

Na nossa visão, as simulações computacionais são ferramentas que podem auxiliar no entendimento de sistemas complexos.

2.2 MODELAGEM E SIMULAÇÃO NO ENSINO DE FÍSICA

A apresentação de visualizações atrativas de fenômenos naturais estudados no Ensino Médio pode ser útil para facilitar a compreensão dos alunos ou, ainda, tornar mais agradável o estudo dos princípios relacionados a eles. Se tal visualização for efetuada via simulação, então existe um valor agregado que permite ao aluno desempenhar um papel ativo no processo educativo. Esse valor está relacionado com a possibilidade de interação entre aluno e máquina.

Como salientado na introdução deste trabalho, ensinar Física nas escolas de Ensino Médio não parece tarefa fácil para os professores. Medeiros e Medeiros (2002) apontam algumas razões para essas dificuldades: a quantidade de conceitos envolvidos, alta dose de abstração relacionada com alguns dos conceitos, ferramenta Matemática distante da compreensão do aluno, o uso da ferramenta Matemática como parte essencial ao ensino dos conceitos físicos. Veit e Teodoro (2002), nessa linha de pensamento, salientam que para muitos estudantes, a Física se resume em decorar fórmulas de origem e finalidades desconhecidas.

---

[3] As simulações são programas de computador que contêm um modelo de um sistema ou processo físico, dedicados à visualização gráfica deste. Além disso, o programa convida os estudantes a explorar e interagir com o sistema modificando seu estado, alterando parâmetros e observando o resultado desta manipulação.

Surge a questão: Como desenvolver um Ensino de Física mais próximo da realidade da produção científica e tecnológica, rompendo-se com um ensino tradicional em que professores escrevem fórmulas no quadro-negro e os alunos resolvem problemas numéricos inventados a partir dessas fórmulas?

Trabalhos recentes sobre Ensino de Física mostram que a modelagem e a simulação trazem benefícios significativos na aprendizagem, desde que usadas de forma coerente e com estrutura pedagógica consistente. Neste sentido, como apontam Veit e Teodoro (2002), a introdução de modelagem no processo ensino/aprendizagem possibilita uma melhor compreensão dos conteúdos da Física ao mesmo tempo em que contribui para o desenvolvimento cognitivo do aluno. Alunos interagindo com uma simulação computacional obtêm melhor entendimento dos sistemas reais, pois essa ferramenta permite efetuar a exploração de conceitos, testes de hipóteses e explicações de ideias (ESQUEMBRE, 2005). Em função disso, os alunos são obrigados a modificarem seus modelos mentais através da comparação entre respostas obtidas e esperadas.

Mas que ferramenta de modelagem e simulação utilizar?

Vários programas de computador voltados ao Ensino de Física apareceram nos últimos anos. Dentre as categorias existentes, destacam-se aquelas voltadas às simulações e modelagens. Para Esquembre (2005, p. 10), esses programas podem ser usados para:

> Poner a disposición del aula un currículo basado en problemas de mundo real.
> Proporcionar guías y herramientas para mejorar el aprendizaje.
> Dar a los estudiantes y a los profesores más oportunidades para la realimentación, la reflexión y la revisión.
> Construir comunidades locales y globales que incluyan a los profesores, administradores, estudiantes, padres y científicos en activo.
> Expandir las oportunidades para la formación de los profesores.[4]

Particularmente, as simulações são úteis quando a experiência original for impossível de ser realizada por alunos em sala de aula, ou quando os experimentos forem perigosos ou, ainda, quando os equipamentos forem de custo elevado. Pode-se salientar, também, o experimento de evolução lenta ou o extremamente rápido (MEDEIROS; MEDEIROS, 2002). Esses autores salientam, ainda, que a interatividade está vinculada ao

[4] Pôr à disposição da aula um currículo baseado em problemas do mundo real. Proporcionar guias e ferramentas para melhorar a aprendizagem. Dar aos estudantes e aos professores mais oportunidades para a realimentação, a reflexão e a revisão. Construir comunidades locais e globais que incluam os professores, administradores, estudantes, pais e cientistas em atividade. Expandir as oportunidades para a formação dos professores.

fato do programa criado permitir diferentes animações através da seleção, via *input*, de parâmetros selecionados pelo estudante.

Gaddis[5], citado por Medeiros e Medeiros (2002), defende a ideia de que simulações computacionais vão além das simples animações. Elas englobam uma vasta classe de tecnologias, do vídeo à realidade virtual, que podem ser classificadas em certas categorias gerais baseadas fundamentalmente no grau de interatividade entre o aprendiz e o computador. Ele, também, fez amplo levantamento das principais justificativas de se utilizar a ferramenta de simulação, a saber:

1. reduzir o 'ruído' cognitivo de modo que os estudantes possam concentrar-se nos conceitos envolvidos nos experimentos;
2. fornecer um *feedback* para aperfeiçoar a compreensão dos conceitos;
3. permitir aos estudantes coletarem uma grande quantidade de dados rapidamente;
4. permitir aos estudantes gerarem e testarem hipóteses;
5. engajar os estudantes em tarefas com alto nível de interatividade;
6. envolver os estudantes em atividades que explicitem a natureza da pesquisa científica;
7. apresentar uma versão simplificada da realidade pela destilação de conceitos abstratos em seus mais importantes elementos;
8. tornar conceitos abstratos mais concretos;
9. reduzir a ambiguidade e ajudar a identificar relacionamentos de causas e efeitos em sistemas complexos;
10. servir como uma preparação inicial para ajudar na compreensão do papel de um laboratório;
11. desenvolver habilidades de resolução de problemas;
12. promover habilidades do raciocínio crítico;
13. fomentar uma compreensão mais profunda dos fenômenos físicos;
14. auxiliar os estudantes a aprenderem sobre o mundo natural, vendo e interagindo com os modelos científicos subjacentes que não poderiam ser inferidos através da observação direta;
15. acentuar a formação dos conceitos e promover a mudança conceitual.

Diante de tanto otimismo sobre as possibilidades vantajosas do uso das simulações no Ensino de Física, é necessário, também, observar e identificar as suas

[5] GADDIS, B. **Learning in a virtual lab:** distance education and computer simulations. Doctoral Dissertation – University of Colorado, 2000.

desvantagens (MEDEIROS; MEDEIROS, 2002). A utilização de modelagem e simulação em sala de aula equivale, sem dúvida alguma, à quebra de paradigma baseado em aulas expositivas e tradicionais. Entretanto, críticos da sua utilização argumentam que elas são modelos simplificados e aproximados da realidade. Nesse sentido, eles salientam a diferença entre as experiências através do experimento real e através do experimento virtual (simulação computacional). Se tal diferença não for percebida, as simulações e modelagens podem comunicar concepções contrárias àquelas que o educador pretende mostrar aos alunos. Edward[6] , citado por Medeiros e Medeiros (2002), observou, com alunos em sala de aula, que as simulações mostravam-se menos efetivas do que experimentos reais. Pode-se, então, indagar: Por quê?

Uma boa simulação pode comunicar muito mais que imagens estáticas apresentadas na lousa e em livros didáticos. Inferir-se, entretanto, que as simulações são comparáveis aos experimentos reais é um equívoco, pois dotar de realismo uma simulação é tarefa extremamente difícil e complicada. A simulação mostra, então, segundo os seus críticos, situação que não está de acordo com a realidade. Se a modelagem não incluir os limites de validade e se determinados fatores forem negligenciados, a simulação irá proporcionar danos maiores que aquelas imagens estáticas.

Medeiros e Medeiros (2002) mostram como os estudantes são ludibriados pela beleza e pelo fascínio da realidade virtual em certas simulações. Isso decorre das simplificações exageradas e dos equívocos produzidos pelo programador durante a criação da simulação. O valor da simulação está condicionado ao modelo, isto é, à teoria utilizada na sua construção. Na elaboração das simulações computacionais, aquele que a estiver preparando, deve ter o cuidado e a atenção à modelagem que lhe dá suporte, isto é, na aplicação das leis que regem o fenômeno estudado, aos conceitos inerentes na construção da teoria e o domínio de validade. Assim, o valor da simulação dependerá de seu papel heurístico e não apenas do algoritmo utilizado para a sua programação ou do efeito ilustrativo que ele proporciona. Uma simulação mal planejada poderá fazer com que o aluno não perceba a desconexão com a situação real. É preciso, então, por parte do professor ou daquele que estiver efetuando o seu planejamento, ter em mente que a simulação é uma imitação da realidade, que está vinculada a um modelo matemático específico e que poderá imitar determinados aspectos da realidade e não a sua total complexidade.

---

[6] EDWARD, N. An evaluation of student perceptions of screen presentations in computer-based laboratory simulations. **European Journal of Engineering Education,** v. 22, p. 143-151, 1997.

No nosso entender, as simulações devem ser pensadas como instrumentos de pesquisa especulativa e que permitam ao estudante um “afastamento” das manipulações e métodos matemáticos, como é notoriamente disseminado nos cursos introdutórios de Física. A simulação bem estruturada e sua utilização planejada pelo professor contribuem como etapa intermediária à abstração. Ela não pode ser vista como o atalho relativo às idas e vindas dos raciocínios exploratórios, dos erros experimentais e dúvidas encontradas em situação real. Todas essas dúvidas e incertezas fazem parte da nossa aprendizagem (“do como aprendemos”) na vida real e a aprendizagem não é apenas um processo de construção individual, pessoal, desconectado das interações sociais e das ferramentas envolvidas no processo de aprendizagem (VEIT; TEODORO, 2002).

A introdução de simulação e modelagem em sala de aula depende de mudanças estruturais relacionadas aos métodos pedagógicos, treinamento de professores, conhecimento das expectativas desses mesmos professores, postura pedagógica de coordenadores e de direção. Sem a intervenção de um professor não há engajamento dos alunos e de outros professores.

Acreditamos que o uso de modelagens e simulações em sala de aula proporcionará uma aprendizagem além da simples memorização e as atividades decorrentes deverão desenvolver habilidades que permitirão o envolvimento cognitivo dos alunos, de forma que o conhecimento possa ser construído e reconstruído. No sentido de se favorecer a aprendizagem construtivista, Veit e Teodoro (2002, p. 88) salientam que a modelagem pode:

> Elevar o nível do processo cognitivo, exigindo que os estudantes pensem num nível mais elevado, generalizando conceitos e relações;
> Exigir que os estudantes definam suas ideias mais precisamente;
> Propiciar oportunidades para que os estudantes testem seus próprios modelos cognitivos, detectem e corrijam inconsistências.

2.3 ORIENTAÇÕES DOS PARÂMETROS CURRICULARES NACIONAIS (PCN)

O Ensino de Física deve promover o desenvolvimento de competências, habilidades integradas pela interdisciplinaridade e da contextualização. Esses são os princípios gerais estabelecidos pelas Diretrizes Curriculares Nacionais para o Ensino Médio (BRASIL, 1999), cujo objetivo é o de conferir unidade ao ensino das diferentes disciplinas de uma área do conhecimento e permitir a articulação dos professores das correspondentes disciplinas.

Ao tratar das tecnologias, particularmente ao uso dos computadores em sala de aula, existe uma defasagem em relação a sua utilização científica e os PCNEM (Parâmetros Curriculares Nacionais para o Ensino Médio) preconizam que os computadores sejam incorporados no processo ensino/aprendizagem. Através do seu uso, professores poderão adotar em sala de aula um currículo baseado em problemas reais, proporcionar ferramentas que auxiliam a aprendizagem, desenvolver atividades reflexivas e expandir as oportunidades relacionadas à formação de professores, particularmente na área de Ciências da Natureza, Matemática e suas Tecnologias:

> A aprendizagem de concepções científicas atualizadas no mundo físico e natural e o desenvolvimento de estratégias centradas na solução de problemas e finalidade da área, de forma a aproximar o educando do trabalho de investigação científica e tecnológica, como atividade institucionalizada de produção de bens e serviços. (BRASIL, 1999, p. 33)

Como mencionado no início dessa seção, os objetivos curriculares devem ser focados em competências e habilidades ao invés de serem focados nos conteúdos específicos que são tratados nas diferentes disciplinas. Outra característica apontada é a de uma educação com ambição formativa, tanto em termos da natureza das informações tratadas, dos procedimentos e atitudes envolvidas como em termos das habilidades, competências e valores desenvolvidos (BRASIL, 1999, p. 207).

Dessa forma, justifica-se a utilização de modelagem e simulação computacional no Ensino de Física para alunos do Ensino Médio, pois elas auxiliariam diretamente no desenvolvimento das habilidades e competências dos alunos na área de Ciências da Natureza, Matemática e suas Tecnologias. Além disso, esse trabalho visa contribuir para a formação do professor de Física do Ensino Médio, como se verá no capítulo seguinte.

3 NECESSIDADES FORMATIVAS DO PROFESSOR

A escola é uma instituição criada pelo homem e tem papel fundamental na elaboração da visão de mundo e na construção do ser humano. Como instituição social, ela tem a tarefa de transmitir as formas de entendimento culturalmente estabelecidas em determinados períodos da história, consolidar certas ideias e posições em relação ao trabalho, ao trabalhador e ao cidadão. Cumpre-lhe, também, a função de sistematizar a transmissão das experiências bem sucedidas da humanidade e adaptá-las às necessidades atuais, com o objetivo de preparar as novas gerações. Dessa forma, o currículo escolar deve materializar esse ideal, propiciando formas eficientes de aquisição das experiências anteriores, crescimento do indivíduo, desenvolvimento de sua autonomia e a comunicação entre os indivíduos.

Pode-se, então, levantar as seguintes questões: A escola tem cumprido o papel de transmissora de uma tradição que auxilia o indivíduo a adquirir uma visão de mundo adequada ao seu desenvolvimento, seja pessoal, social ou intelectual? O indivíduo tem se beneficiado, em seu cotidiano, do que é ensinado na escola?

Adquirir uma visão de mundo e do que é ensinado na instituição escolar está associado ao progresso da sociedade.

O Ensino de Física, em particular, deve ser pensado como parte de um conhecimento científico a ser transmitido pela escola, submetido às necessidades de uma educação geral que permita ao indivíduo entender e interagir no mundo em que vive. Neste contexto, o professor é absolutamente necessário para gerar renovação no ensino. Entretanto, existem diferenças entre o objetivo proposto pelo currículo escolar e a prática pedagógica do professor. Não basta estruturar cuidadosamente um currículo se o professor não estiver preparado adequadamente para aplicá-lo (CRONIN-JONES, 1991). Esse problema não se resolve proporcionando ao professor instruções mais detalhadas através de cursos, manuais, programas de “capacitação” ou “reciclagem”. É necessário fazer uma revisão da formação do professor, incluindo o conhecimento sobre as diversas abordagens do processo ensino/aprendizagem. Nesse ponto, em particular, quando se solicita a um professor de Física em exercício, ou em formação, que expresse a sua opinião sobre a importância de se conhecer as diferentes abordagens, a resposta, em geral, é pífia e, também, não inclui conhecimentos sobre a pesquisa na área de Ensino de Física.

A falta, ou o pouco, de conhecimento do professor de Física em relação às contribuições da pesquisa em Didática pode ser interpretado como resultado da visão, obtida

na maioria dos cursos de Licenciatura em Física, de que o ensino é algo simples para o qual basta um bom conhecimento da matéria, alguma prática e alguns complementos psicopedagógicos (DUMAS-CARRÉ et al., 1990). A formação do professor de Física é vista como a transmissão de um conjunto de conhecimentos e destrezas que são insuficientes na preparação dos alunos e do próprio professor (BRISCOE, 1991).

O professor de Física necessita de uma formação adequada e, muitas vezes, não tem consciência de suas insuficiências. Essas carências não devem ser interpretadas como incapacidades, pois proporcionando aos professores um trabalho reflexivo, de debate e aprofundamento, constata-se uma produção de excelente nível. Deve-se, então, orientar o trabalho de formação do professor como uma pesquisa dirigida, que contribua de forma funcional e efetiva na transformação de suas concepções iniciais. O que se deve "saber" e "como saber" por parte do professor de Física implica em exercer uma docência de qualidade (GIL-PÉREZ, 1991).

Para Robilotta e Babichak (1997), ensinar física é difícil, pois a disciplina apresenta características peculiares e que abrangem o conhecimento das ideias abstratas sobre o mundo natural, a educação, a matemática, a psicologia, etc.

Para romper com a visão de que ensinar é fácil, são apresentados alguns temas, que poderão servir como início de reflexão, relacionados às necessidades formativas do professor de Física (CARVALHO; GIL-PÉREZ, 1995):

- conhecimentos teóricos sobre os processos ensino/aprendizagem.
- conhecimentos sobre a matéria a ser ensinada.
- preparação de atividades.
- orientação de trabalhos a serem desenvolvidos pelos alunos.
- avaliação.

Nesse sentido, coloca-se a seguinte questão: Quais conhecimentos os professores precisam adquirir? Essa questão é diferente daquela desenvolvida até recentemente em que se tentava identificar as características de um bom professor de Física, isto é, nas diferenças entre um bom ou mau professor. Assim, o novo questionamento supera concepções do tipo "um professor é ou nasce como tal".

Em geral, esse conjunto de problemas, relacionados com as necessidades formativas do professor de Física, costuma gerar um mal-estar entre os professores. Eles questionam, por exemplo, se é possível, um só indivíduo, possuir tantos conhecimentos diversificados. Naturalmente é impossível, mas conhecer a pesquisa que está sendo desenvolvida no Ensino de Física é exigência do trabalho docente.

O professor não deve se sentir vencido por um conjunto muito grande de saberes. O trabalho docente deve ser uma tarefa coletiva, desde a preparação das aulas até a avaliação. A complexidade da atividade docente, desta forma, deixa de ser vista como um obstáculo à eficiência e passa a ser um convite ao rompimento com a inércia de um ensino monótono e sem perspectivas. O trabalho docente desenvolvido como um trabalho coletivo, de inovação, pesquisa e formação permanente iria contribuir muito para o desenvolvimento do potencial criativo dos alunos. O professor deve ser a imagem viva do “aprender a aprender” (DEMO, 1993, p.128), sendo o núcleo da educação.

4 A FERRAMENTA *EASY JAVA SIMULATIONS* (EJS)

Os computadores podem ser utilizados no ensino como uma ferramenta para diversos propósitos: tornar leituras mais atrativas, visualizar imagens, realizar visitas virtuais aos museus, ouvir músicas, assistir documentários, auxiliar no entendimento de conceitos, etc. Entretanto, eles ainda não são usados efetivamente por professores das instituições públicas ou privadas. Isso é decorrência, talvez, do receio desses professores de não dominarem os recursos relacionados ao uso do computador, ou de não saber utilizar e controlar tecnologias multimídia, ou, ainda, de não terem à disposição produtos que sejam eficientes para aquilo que se pretende ensinar. Dentre esses produtos destacam-se as ferramentas de simulação e animação para o ensino de Ciências.

Uma possibilidade para o Ensino de Física, em particular, é apresentar uma ferramenta que permita ao professor criar suas próprias simulações. Durante o processo de criação da simulação, o professor obteria uma nova visão do fenômeno que ele está tentando explicar e, ao mesmo tempo, poderia desenvolver nos alunos o entusiasmo sobre o uso da simulação. Outra possibilidade é deixar que os alunos criem e desenvolvam suas próprias simulações. Essa participação do aluno no processo de aprendizagem é denominada de modelagem construtiva pelos pesquisadores em Educação. A vantagem dessa possibilidade no processo de ensino/aprendizagem está relacionada com o fato de o estudante fazer ciência de forma exploratória e construtiva, prática essa esquecida pela maioria dos professores.

Criar uma simulação exige, por parte da pessoa envolvida no processo, esforço, tempo, completo entendimento do fenômeno que se quer simular e, além do mais, conhecimento sobre as técnicas da linguagem de programação computacional. O resultado final é a resposta desejada sobre a evolução do comportamento do fenômeno a ser mostrado no monitor do computador. Assim, a proposta dos autores de *Easy Java Simulations*, abreviadamente EJS, "é permitir ao professor criar suas próprias simulações científicas em Java de forma rápida e fácil" (ESQUEMBRE, 2005, p. XII).

O programa EJS e as informações para o seu uso, que podem ser obtidos na *web* (http://www.um.es/fem/Ejs/), estão direcionados aos alunos de ciências, professores ou pesquisadores que tenham o conhecimento básico em programação de computadores, mas que não possuem o tempo necessário para criar uma simulação gráfica completa. A ideia principal dos autores de EJS é a de que ele foi criado e desenvolvido para permitir a qualquer pessoa, envolvida no processo da simulação, concentrar o máximo de seu tempo em escrever e refinar

algoritmos, segundo modelos científicos, e dedicar o menor tempo nas técnicas de programação.

O objetivo deste capítulo é o de apresentar a estrutura básica de uma simulação, de modo que ela possa ser corretamente traduzida para o programa de computador, e instruir o professor, ou o aluno, como utilizar o EJS para a criação de simulações de acordo as suas ideias e concepções. O capítulo de forma alguma esgotará as possibilidades de EJS, mas servirá como ponto de partida para que o professor, ou um aluno, possa apreciar a estruturação de uma simulação e, o mais importante, como produzir a sua visualização.

No nosso entendimento, o texto tutorial sobre o uso do EJS apresentado na página da internet, com o endereço citado anteriormente, pode deixar o usuário confuso e os exemplos distribuídos junto com o programa são, na maioria, complexos. Da mesma forma, o livro de Esquembre (2005), em língua espanhola, não é fácil de ser encontrado no mercado brasileiro e a sua estrutura é semelhante àquela apresentada na página da *internet*. Consideramos, também, o fato de existirem muitas pessoas (professores e alunos) que não possuem o conhecimento de outros idiomas (inglês ou espanhol).

Salientamos, também, que Figueira (2005) faz uma apresentação das características e aplicações da ferramenta EJS e cita, de forma muito breve, as possibilidades e as vantagens do uso de simulações no Ensino de Física.

Como produto final dessa dissertação, o leitor encontrará um breve texto que servirá de orientação sobre o uso do EJS, conforme citamos na Introdução dessa dissertação. Esse texto que estará disponível na página do PPGECE-UFSCar não está completo! Ele será periodicamente atualizado com as informações relacionadas às técnicas de criação de simulações e, também, com exemplos.

4.1 O *EASY JAVA SIMULATIONS*

O EJS é um *software* desenvolvido por Francisco Esquembre, da Universidade de Murcia, Espanha, com o propósito de se criar uma simulação de computador discreta, isto é, um programa que reproduz com propósitos pedagógicos, ou científicos, um fenômeno natural através da visualização dos diferentes estados que ele possa assumir. Cada um desses estados pode ser descrito por um conjunto de variáveis que se modificam com o tempo devido à iteração do algoritmo considerado.

Como um programa que auxilia na criação de simulações científicas, o EJS foi concebido por professores de ciências para professores, estudantes de ciências e, ainda

àqueles que se preocupam em simular um fenômeno sem se aterem aos aspectos técnicos necessários à construção da simulação. EJS fornece uma estrutura conceitual e ferramentas simplificadas que permitem ao professor, e aluno, gastar um tempo maior na descrição do modelo do fenômeno que se quer simular. O resultado final, que é automaticamente gerado pelo EJS, pode ser considerado uma criação de um programador profissional em termos de eficiência e sofisticação.

A escolha da linguagem *Java* é justificada pelos autores de EJS em função de sua aceitação internacional entre os usuários da internet e pelo fato de ser suportada por várias plataformas, permitindo obter um produto independente, de alto desempenho e que pode ser compartilhado via internet. O EJS pode ser usado como um programa em diferentes sistemas operacionais, distribuído via *internet* e rodar em páginas do tipo HTML. Em particular, o EJS cria *applets* em *Java* independentes, que podem ser utilizados em diferentes plataformas e serem visualizados com a utilização de diferentes *browsers*. Além do mais, ele permite a leitura de um conjunto de dados através da *net* e as simulações podem ser controladas usando-se *scripts* através de páginas HTML.

O EJS é uma ferramenta pedagógica em si, pois existe uma riqueza educacional no processo de se criar uma simulação. Com ele, por exemplo, um professor poderá orientar seus alunos na criação de simulações sobre um determinado tópico de estudo, como oscilações, conservação da energia, etc. A ferramenta EJS estará, assim, auxiliando o estudante a expor e explicar as suas concepções sobre um determinado fenômeno. Usado em grupos, os estudantes estarão desenvolvendo e aprimorando a capacidade de discutir e comunicar sobre ciência.

4.2 “ATIVAÇÃO” E “EXECUÇÃO” DO EJS

Após a instalação do programa, é possível deixar o ícone console do EJS (*EjsConsole*) visível no *desktop* do *Windows*. Ao clicar no ícone (duplo clique sobre o ícone com o uso do *mouse*) observa-se o aparecimento de uma caixa mostrando o progresso de abertura do programa.

Na finalização do processo de abertura do EJS, o usuário observará uma janela, denominada de janela de interface, onde ele efetuará todos os procedimentos para a criação de uma simulação de Física. Basicamente, durante a preparação de uma simulação, o usuário seguirá um conjunto de operações padronizadas e que serão descritas ao longo desse trabalho.

A Figura 1 mostra a janela da interface. Observa-se uma região com os botões *Description* (descrição), *Model* (modelo) e *View* (janela de visualização), cada um correspondendo às três partes da simulação que será criada com o EJS. Cada botão tem uma cor característica e funciona um de cada vez ao ser selecionado. Abaixo dessa barra de botões (linha) existe uma área vazia com a mensagem convidativa *"Click to create a description page"* (clique para criar página de descrição).

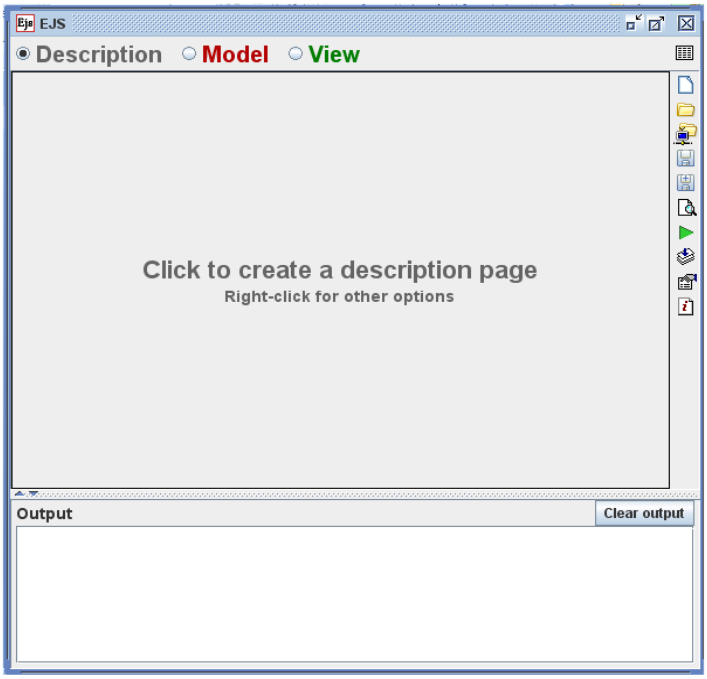


Figura 1 – Janela da interface do *EJS*.

Fonte: Easy Java Simulations

Na parte inferior da janela está a área de mensagens (O*utput*) que o EJS utiliza para mostrar as mensagens correspondentes das ações realizadas durante o processo de compilação ou execução da simulação.

A parte lateral direita da janela apresenta uma série de ícones que podem ser facilmente reconhecidos. Por exemplo, o ícone  representa "Novo" e sua função é a de permitir criar um novo documento, ou seja, a simulação.

Na pasta "Abrir" (ícone ), existem vários exemplos desenvolvidos por professores e pesquisadores de instituições diferentes e que podem ser utilizados como roteiro de estudo para se criar uma simulação. Esses exemplos tentam mostrar como desenvolver uma simulação, e, ao mesmo tempo, permitem observar como elas ocorrem. Entretanto, a maioria dos exemplos, aí contidos, é sofisticada e "perde-se muito tempo" em decifrá-los. O aconselhável é aprender inicialmente a construção de uma simulação e dominadas as tarefas iniciais, estudar as simulações contidas nesta pasta.

4.3 A ESTRUTURA DE UMA SIMULAÇÃO

Muitas simulações computacionais de fenômenos naturais são descritas através do paradigma *model-control-view*. Este paradigma considera a simulação constituída de três partes:

a. *Model* (modelo): O modelo descreve o fenômeno a ser estudado em termos das variáveis e das relações entre essas variáveis. As variáveis são caracterizadas como grandezas de estado e podem assumir determinados valores dentro do intervalo de validade do fenômeno simulado. As relações entre variáveis correspondem às leis que governam o fenômeno e são expressas por algoritmos escritos em uma linguagem de programação.
b. *Control* (controle): O controle define certas ações que o usuário pode alterar durante a simulação.
c. *View* (visão ou janela de visualização): A visão mostra a representação dos diferentes estados que o fenômeno pode assumir.

Essas três partes estão relacionadas entre si. O modelo afeta a visão, pois uma mudança no estado do modelo pode alterar a configuração gráfica visualizada pelo usuário. O controle altera o modelo, pois as ações de controle podem alterar os valores das variáveis do modelo. Finalmente, a janela de visualização afeta o modelo e o controle, pois a interface gráfica (a representação visual para o usuário) pode conter componentes que permitem o usuário modificar as variáveis ou efetuar as ações pré-definidas.

Entretanto, o EJS apresenta uma simplificação desse paradigma através da supressão da parte relativa ao controle (*Control*) e integrando as suas funções na visão (via janela de visualização) e no modelo (*Model*). Essa simplificação permite o usuário interagir com a simulação. Para essa tarefa ocorrer adequadamente, ao se programar o modelo é necessário especificar de que forma as ações realizadas pelo usuário sobre os componentes de controle da janela de visualização ou dos componentes gráficos afetarão os valores das variáveis de estado do modelo, durante a simulação. Esse assunto será analisado na parte relativa à “construção” da janela de visualização (*View*).

O painel mostrado na Figura 1 resume a simplificação do paradigma apontado no parágrafo anterior. Assim, a definição de um laboratório virtual com o uso do EJS se estrutura nas seguintes partes:

a. *Description* (Descrição): Nessa página é possível incluir roteiros, resumos, descrição de experimento, questionários, etc.

b. *Model* (Modelo): Modelo dinâmico cuja simulação via interatividade é a base do laboratório virtual.
c. *View* (Visão): Interface entre o usuário e o modelo que proporciona a representação visual do comportamento dinâmico do modelo e os mecanismos para o usuário poder interagir com o modelo durante a simulação.

As partes descritas acima mostram que os computadores são ferramentas que permitem interatividade, isto é, o usuário pode alterar a lógica de um programa através de certas ações como: ao clicar num botão do *mouse*, ou arrastar um elemento da simulação com uso do próprio *mouse*, ou via alteração de valores com o uso do teclado do computador, ou através de outros tipos de periféricos do computador, ou, ainda, na própria janela de visualização. Dessa forma, a janela de visualização pode ser utilizada para controlar a simulação.

Em resumo: criar uma simulação com o EJS consiste em definir o modelo, criar a correspondente janela de visualização, estabelecer as conexões necessárias para a correta visualização do fenômeno simulado e inserir as ações que permitam a interação apropriada entre o usuário e a janela de visualização. Essa separação explícita das partes reforça conceitualmente o papel central do modelo e da simulação.

4.4 ESCREVENDO UMA INTRODUÇÃO

Ao criar uma simulação, é conveniente incluir uma descrição textual sobre como utilizá-la. Isso será apropriado para aquelas pessoas que utilizarão a simulação. Pode-se, também, apresentar um resumo teórico do fenômeno a ser simulado ou, ainda, criar listas de exercícios e de atividades para os alunos. Como o EJS permite criar mais de uma página, é importante diferenciar os seus nomes. Dessa forma, é possível escrever uma introdução, um resumo teórico, criar páginas com atividades diferentes ou páginas com atividades para os alunos desenvolverem.

Admitindo-se que a janela do EJS esteja ativada, o usuário deve selecionar o botão *Description* (Descrição) e, a seguir, clicar com o botão direito (ou o esquerdo, dependendo da configuração adotada no painel de controle do *Windows*) do *mouse* no painel central. Assim, uma caixa de diálogo aparece no monitor para o usuário atribuir um nome à página que deseja criar. Finalizado esse procedimento, o usuário observa uma janela de edição de texto com ícones que são facilmente reconhecidos por suas funções. Entretanto, se o

usuário passar a seta (do *Windows*) sobre um determinado ícone, ele irá obter a informação sobre a sua função através de uma pequena caixa de diálogo.

Ao clicar na região central da janela, o usuário escreverá aquilo que tem em mente, ou seja, um resumo teórico, uma atividade, a descrição de como funciona a simulação, o objetivo da simulação, etc. Em cada página é possível inserir uma figura, uma fórmula, ou uma tabela. Se preferir, o usuário poderá escrever o texto em outro editor, como o *Word*, e importá-lo para a página que está sendo desenvolvida.

A Figura 2 mostra um exemplo de página pronta. Nela pode-se observar a existência de três páginas (abas): Definição do MRU, Descrição do MRU e Atividades. Ao ativar cada uma delas, o usuário encontrará uma série de informações necessárias ao entendimento da simulação e as tarefas propostas. Claro que a concepção dessas páginas depende do objetivo que se pretende atingir!

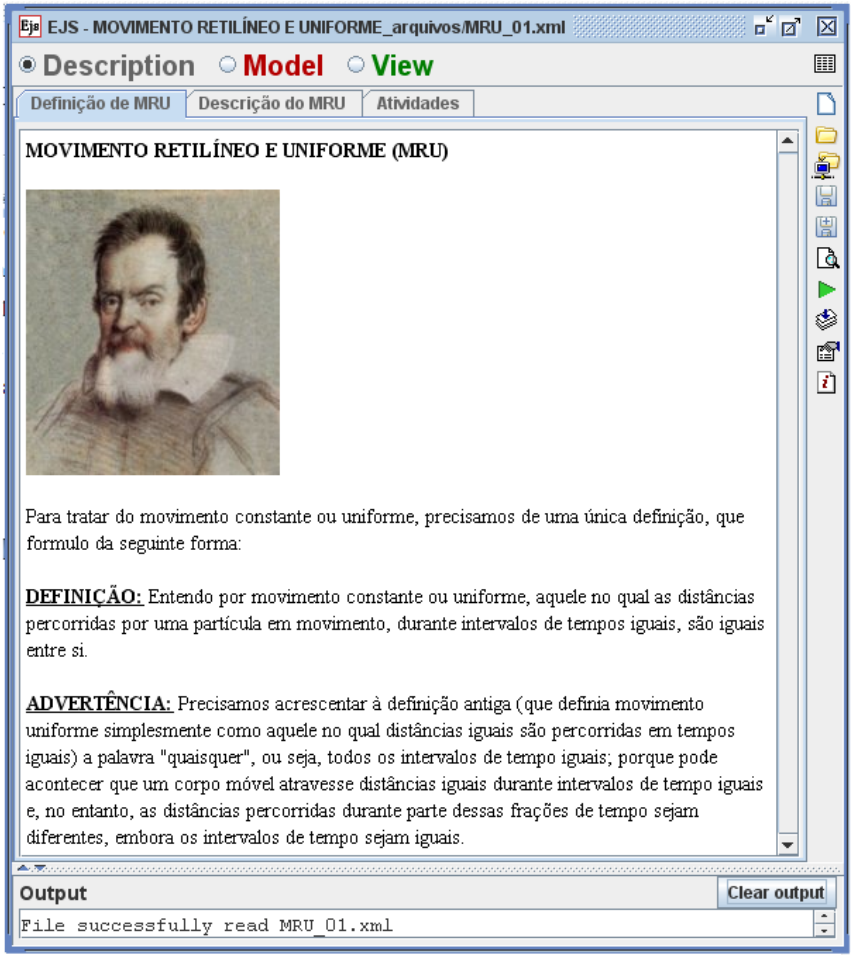


Figura 2 – Exemplo de página criada em *Description* para a simulação sobre MRU.

Fonte: Easy Java Simulations

Salientamos que a página apresentada na Figura 2 faz parte da simulação MRU_01.xml, de nossa autoria. O objetivo dessa simulação é o de aprender os recursos do EJS relacionados com o editor de textos, a escrita de equações diferenciais, a disposição dos elementos gráficos, como eles são conectados para se ter um efeito desejado pelo usuário e

qual o efeito final apresentado na janela principal de visualização. O texto apresentado é cópia, *ipsis litteris*, das páginas 214 e 215 do livro de Hawking (2005).

4.5 CONSTRUINDO MODELOS COM O EJS

4.5.1 DEFINIÇÃO DO MODELO

A parte mais importante da simulação é o modelo (*Model*). Nessa fase definem-se as grandezas relevantes, os valores iniciais de algumas delas e as regras (leis físicas, por exemplo) que governam a mudança dos valores dessas grandezas. A palavra grandeza é usada para indicar as variáveis de estado (valores que descrevem a evolução do fenômeno e são expressas por derivadas), os parâmetros (valores que permanecem constantes) e variáveis algébricas (aquelas que não são expressas por derivadas e não permanecem constantes). A palavra grandeza será usada como sinônimo de variável. A variável pode representar uma grandeza que seja constante ou cujo valor se altera em relação ao tempo, por exemplo.

Para iniciar o modelo, o usuário deve inicialmente selecionar o botão *Model* na barra de ferramentas e, logo a seguir, uma nova página é apresentada (Figura 3). Observa-se, abaixo da barra de ferramentas principal (*Description*, *Model*, *View*), outra barra com as palavras em vermelho: *Variables*, *Initialization*, *Evolution*, *Fixed relations* e *Custom*. Ao lado de cada uma dessas palavras, o correspondente botão de ativação. Na parte central da janela está escrito o convite para se criar as variáveis de estado que intervêm no fenômeno a ser simulado.

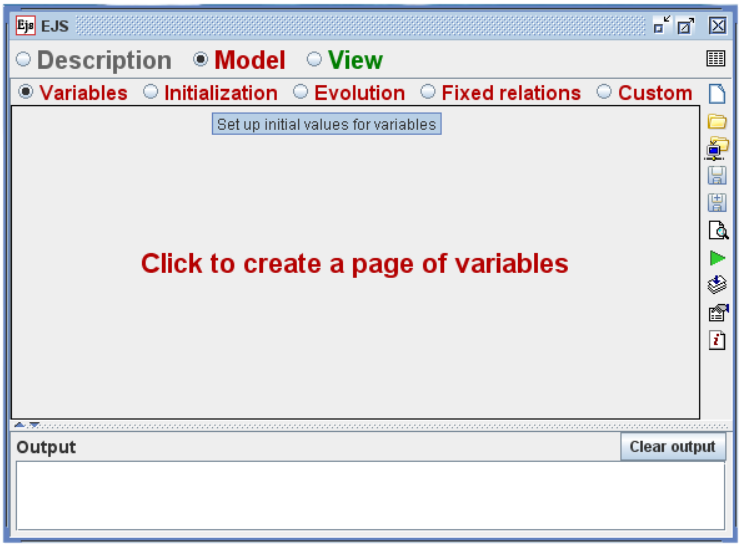


Figura 3 – Ativação da janela *Model*.

Fonte: Easy Java Simulations

O estado de um modelo é representado pelos valores das suas variáveis. Normalmente, atribuem-lhes nomes significativos como velocidade, aceleração, massa, força, etc.

Para se especificar o modelo de uma simulação é necessário caracterizar as grandezas do fenômeno, estabelecer os valores iniciais do estado inicial do sistema, as equações que determinam a evolução do sistema e as equações de vínculo, que apresentam grandezas que dependem dos valores das grandezas modificadas durante a evolução.

4.5.2 O ESTADO INICIAL DO MODELO

A primeira tarefa ao se construir a simulação é definir as variáveis que intervêm no modelo. Esse procedimento é crucial para a evolução da simulação. Escolhidas e definidas as variáveis, atribuem-se os valores iniciais ao estado inicial do sistema.

Para exemplificar essa tarefa, considere o problema massa-mola que é estudado por alunos do Ensino Médio e Superior. Nesse problema, um corpo de massa *m*, preso na extremidade livre de uma mola de massa desprezível, comprimento *L* e constante elástica *k*, é colocado em movimento. Admite-se que a mola obedeça à lei de Hooke e, por simplicidade, os efeitos dissipativos são desprezados. O objetivo do problema é o de simular o movimento do sistema.

Pela Segunda Lei de Newton, o movimento do sistema é modelado pela equação diferencial:

$$\frac{d^2x}{dt^2} + \frac{k}{m} \cdot (x - L) = 0 \qquad (4.5.1)$$

As grandezas físicas envolvidas no modelo e representadas na equação acima são:

x – posição da extremidade da mola presa ao corpo,

t – tempo,

k – constante elástica da mola,

L – comprimento da mola no estado relaxado,

m – massa do corpo preso na extremidade livre da mola.

Aqui reside uma das dificuldades de EJS! Ao escrever a equação diferencial do movimento do sistema massa-mola, pode-se pensar que apenas as grandezas que figuram naquela equação são as grandezas que interferem no fenômeno. Entretanto, não é tão simples assim. O EJS é um programa de computador e, sendo assim, ao simular um fenômeno físico o usuário deve ter em mente qual é o resultado final da simulação. Apesar de não ser necessário ao usuário conhecer as técnicas de programação, ou ter que escrever o programa em *Java*, alguns detalhes são obrigatoriamente necessários conhecer. Dentre eles:

- O EJS segue as regras de execução de um algoritmo escrito em linguagem *Java*. A evolução do fenômeno ocorre em função do tempo e torna-se necessário caracterizar uma grandeza que representa a variação do tempo. Além das grandezas representadas acima, utiliza-se, também, a grandeza *dt*. Essa grandeza representa o incremento de tempo para calcular a posição do corpo, ou sua velocidade, num determinado instante *t*.
- Em geral, o produto final da simulação é uma animação ou um gráfico. Para isso, o usuário deverá considerar algumas outras grandezas que permitirão a evolução correta da simulação. Algumas dessas grandezas podem estar relacionadas, por exemplo, com a posição de determinados elementos (objetos) que aparecerão na janela de visualização ao representar a evolução da simulação.

Continuando com o exemplo do sistema massa-mola descrito anteriormente, ao clicar na região central da janela, *Click to create a page of variables*, correspondente a *Variables* (vide Figura 3, p. 35), o usuário poderá especificar numa tabela característica o tipo de variável e atribuir alguns valores, como valores iniciais de estado, parâmetros, etc. Da mesma forma que em *Description*, o usuário pode criar quantas páginas forem necessárias.

As variáveis, que intervêm no fenômeno a ser simulado, podem ser do tipo *integer* (inteiro), *double* (real de dupla precisão), *string* (cadeia de caracteres), etc. Por enquanto, para o exemplo que estamos utilizando, as variáveis do tipo *double*. Observa-se nessa nova janela (Figura 4) uma parte central contendo quatro colunas com as designações *Name*, *Initial value*, *Type* e *Dimension*, cada uma correspondendo, respectivamente, ao nome que será atribuído à variável (letra ou palavra), ao valor inicial (se necessário), o tipo (se inteiro, real, etc.) de variável e a sua dimensão (para variáveis do tipo vetor ou matriz).

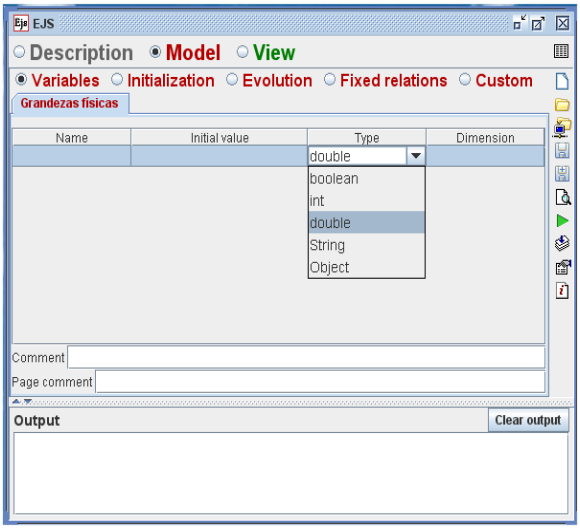


Figura 4 – Atribuição dos valores iniciais das variáveis.

Fonte: Easy Java Simulations

O EJS fornece a opção para se alterar o tipo de variável diretamente na coluna *Type* ao lado da palavra *double*. Ao se clicar do lado direito dessa palavra, com o botão contrário do *mouse* (Figura 4), aparece uma coluna com as seguintes opções: *boolean*, *int*, *double*, *String* e *Object*.

Para preencher uma linha

com o nome, o valor inicial, etc., basta clicar na linha e automaticamente uma nova linha é criada. Isso ocorre sucessivamente para cada variável que é colocada em sua respectiva linha.

Após a inserção das variáveis é possível, ainda, editar essa página. Clicando-se sobre uma linha, em qualquer parte dela, com o botão "contrário" do *mouse*, é possível efetuar as alterações desejadas através de uma coluna que aparece com as seguintes opções: *Insert before* (inserir linha antes), *Insert after* (inserir linha depois), *Move up* (mover a linha para cima), *Move down* (mover a linha para baixo), *Cut* (recortar), *Copy* (copiar), *Paste* (colar) e *Delete* (apagar).

A Figura 5 ilustra a janela *Variables* completamente preenchida com a opção, só para exemplificar, de se efetuar uma alteração desejada. No final da página, no campo *Output*, visualiza-se a mensagem decorrente da compilação. Observa-se, também, o campo denominado *Comment*, no qual é possível fazer uma descrição da variável, e o campo *Page comment*, para se escrever um comentário da página. Esses campos são importantes para se efetuar uma documentação da simulação, de forma que qualquer pessoa possa entender o que representa a variável e qual a finalidade da página apresentada na simulação.

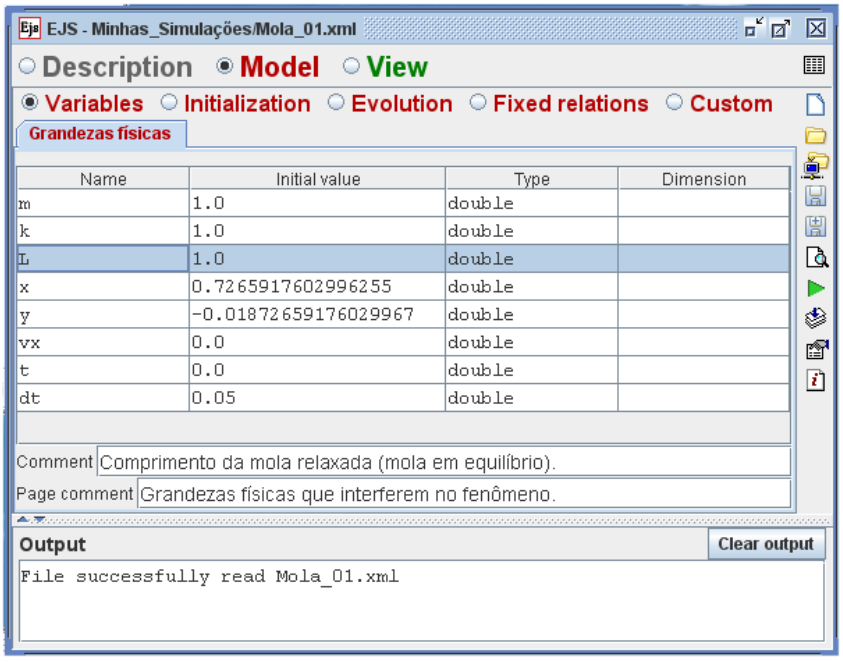


Figura 5 – Janela *Variables* com as grandezas físicas do fenômeno.

Fonte: Easy Java Simulations

4.5.3 EQUAÇÕES DE EVOLUÇÃO E VÍNCULO

Um sistema físico pode evoluir automaticamente do estado atual, representado pelos valores x_1, x_2,..., x_n para um novo estado representado pelos valores x'_1, x'_2, ..., x'_n, simulando, assim, o desenvolvimento do fenômeno com o tempo. As equações que regem tal transição, de um estado para outro estado posterior, são denominadas de equações de evolução. Genericamente elas podem ser escritas como um sistema de uma ou mais equações da forma:

$$x'_i = f_i(x_1, x_2, ..., x_n) \qquad (4.5.2).$$

Algumas vezes, as leis que governam essa evolução têm uma formulação direta, como no caso de sistemas discretos, da forma:

$$x_{n+1} = f\left(x_n\right) \qquad (4.5.3).$$

Em outras situações, as equações de evolução derivam dos métodos numéricos discretos de modelos contínuos descritos por equações diferenciais ordinárias.

Entretanto, é importante salientar que na maioria das vezes se requer uma conjunção de técnicas e decisões lógicas para se produzir um algoritmo sofisticado e que possa descrever de forma mais próxima do real o fenômeno que se pretende simular.

Simular a evolução de um sistema em relação ao tempo consiste em realizar uma computação do estado atual de valores do modelo x_1, x_2, ..., x_n para o novo conjunto de valores x'_1, x'_2, ..., x'_n, tomando-se, por sua vez, esse novo conjunto como o novo estado do modelo e repetindo continuamente esse processo, enquanto a simulação estiver acontecendo.

Além das equações associadas à evolução automática do sistema, é necessário descrever as reações às mudanças forçadas por agentes externos, isto é, mudanças impostas diretamente pelo usuário.

A mudança de uma dada variável causada pelo usuário pode afetar outras variáveis que estão relacionadas a ela (como já citado no Capítulo 4.3). Essas relações, que permitem tal alteração, denominadas de vínculos (*Fixed relations*), são escritas por uma ou mais equações da forma:

$$x_i = g_i\left(x_1, x_2, ..., x_{i-1}, x_{i+1}, ..., x_n\right) \qquad (4.5.4).$$

Uma variável específica só pode aparecer em um dos lados da equação acima. Se em um dado instante, uma variável do lado direito de um vínculo mudar, então a equação é avaliada e a variável do lado esquerdo é modificada.

Ao se escrever um modelo, é conveniente identificar claramente quais são as equações que correspondem à evolução e quais correspondem ao vínculo. Um critério útil,

porém nem sempre válido, é examinar a equação utilizada para computar o novo valor de uma variável. Se este valor depender do valor atual da mesma variável, então essa é uma equação de evolução.

4.5.4 INICIALIZAÇÃO DAS VARIÁVEIS E EQUAÇÕES

Assim que as variáveis estiverem caracterizadas, os valores iniciais atribuídos e as equações de evolução e de vínculo especificadas, diz-se que o modelo está finalizado. Ao executar (rodar) a simulação, as variáveis são criadas, os seus valores iniciais estipulados na inicialização e as equações de vínculo avaliadas. Diz-se que o modelo está em seu estado inicial e a simulação espera pelo passo (etapa) da evolução ou pela interação do usuário. No primeiro caso, as equações de evolução são avaliadas e, imediatamente depois, as equações de vínculo. Dessa forma é atingido um novo estado do modelo em um novo instante de tempo. No segundo caso, quando o usuário altera o valor de uma variável, as equações de vínculo são avaliadas e obtém-se um novo estado do modelo no mesmo instante de tempo.

No exemplo do sistema massa-mola, descrito no Capítulo 4.5.2, já foram identificadas as variáveis de estado e inseridas na página correspondente a *Variables* (reveja a Figura 5, p. 38).

Para continuar na elaboração da página *Model*, o usuário deve clicar em *Initialization* e, novamente, uma nova página irá aparecer. Repetindo-se os procedimentos anteriores, o usuário fornece o nome da página, clica-se em *OK* e aparece outra página onde será escrito o código *Java* para se inicializar a simulação. O código em *Java*, nessa página, será executado uma única vez no início da simulação e na mesma ordem em que as abas são mostradas (se existirem outras abas). No exemplo que estamos analisando, os valores iniciais já estão mostrados na Figura 5 e não há necessidade de uma página de inicialização.

A próxima tarefa é escrever as equações que permitem a evolução temporal do sistema. Para isso, o usuário deve clicar em *Evolution* e a Figura 6 mostra o aspecto da interface que aparece no monitor do computador. Pode-se observar que este painel é mais elaborado. Diferente dos painéis anteriores, a parte central é dividida em dois subpainéis. O superior é para se criar uma nova página, onde será escrita a equação de evolução em código *Java*, e o inferior é para se criar uma equação diferencial ordinária (*ODE-Ordinary diferential equation*). Isso corresponde ao fato de que EJS permite que a evolução do modelo possa ser descrito de duas formas diferentes e que são complementares.

Em princípio, escrever uma equação de evolução consiste simplesmente em transcrevê-las em sentenças do código *Java*. Entretanto, muitas equações de evolução derivam de uma resolução numérica de sistemas de equações diferenciais. Isso só é possível àqueles que estão familiarizados com as técnicas de programação e conhecem os métodos numéricos de resolução que permitem a escrita em código *Java*.

Aqui reside uma das vantagens do EJS. Ele inclui a possibilidade de se introduzir as equações em um editor para esse fim e, automaticamente, gerar o código associado para resolver as equações utilizando os algoritmos de resolução mais empregados. O usuário pode, então, clicar na parte superior para começar com uma área em branco e escrever o código *Java*, ou clicar na parte inferior para iniciar com o editor de equações diferenciais.

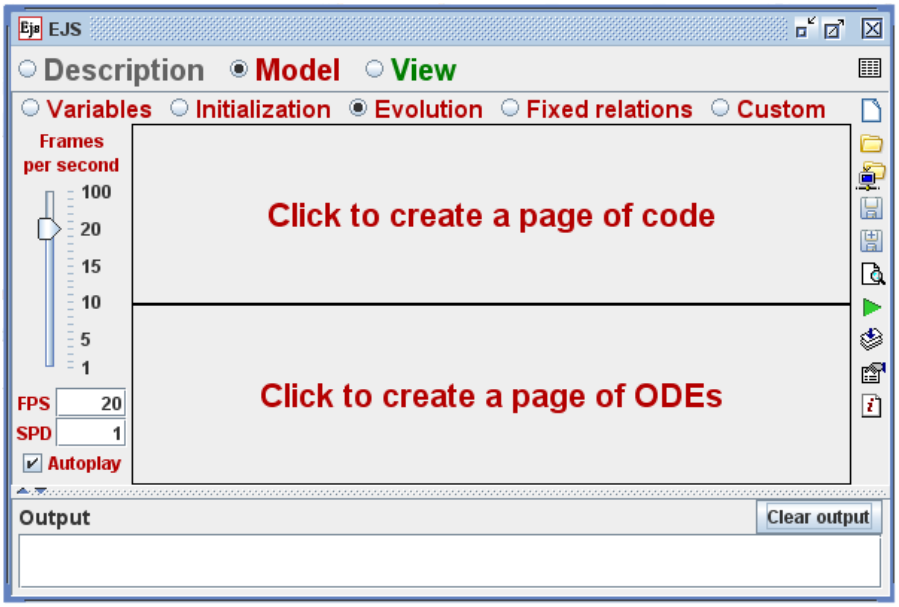


Figura 6 – A janela *Evolution* para as equações de evolução do sistema.

Fonte: Easy Java Simulations

No lado esquerdo do painel mostrado na Figura 6, observa-se um botão deslizante que modifica o valor do chamado *Frames per second* (quadros por segundo), seguido por um texto não editável de rótulo *FPS* e um *Box* de checagem onde se lê *Autoplay*. Por *default*, quando uma simulação é executada, a evolução se inicia automaticamente. Isso pode ser visto pela seleção *Autoplay* do *check Box*. Entretanto, algumas vezes torna-se necessário alterar esse comportamento. Por enquanto, considera-se que o usuário irá manipular a simulação na ordem apresentada para completar certas tarefas, antes de iniciar a evolução. Dessa forma, o usuário necessita desabilitar a opção *Autoplay*. A simulação será

mostrada, mas a sua evolução não irá acontecer. Consequentemente, deve-se orientar o usuário, aquele que irá utilizar a simulação para estudo, o que fazer para iniciar a evolução. Isso pode ser realizado, com descrito anteriormente, em páginas específicas do painel *Description*.

Outra forma de iniciar a simulação, é o de incluir um botão na janela de visualização da simulação com a ação associada ao método pré-definido *_play( )*. Esse tipo de procedimento será abordado no Capítulo 5 (Criando uma janela de visualização).

Acima da caixa *Autoplay* está o controle de velocidade da simulação. Normalmente espera-se que o computador execute uma simulação com rapidez e repetindo a evolução dos passos sem interrupção ou atraso. Entretanto, como os computadores tornam-se cada vez mais rápidos, é necessário forçá-los a interromper a execução entre passos sucessivos. Caso contrário, não seria possível visualizar o que está acontecendo. Este é o propósito do cursor associado a *Frames per second* (*FPS*) – quadros por segundo.

O *FPS* representa o número aproximado de passos de evolução que irá ocorrer no intervalo de tempo de um segundo, as se executar a simulação. O valor mínimo é um (1) e o valor máximo é cem (100).

Como mencionado anteriormente, as equações de evolução podem ser escritas através do código *Java*. É possível adicionar quantas páginas de código forem necessárias, da mesma forma e com a mesma utilidade das páginas de inicialização descritas anteriormente.

Se existirem as equações diferenciais ordinárias (EDO), essas podem ser resolvidas pelos métodos apresentados em livros de Cálculo Numérico. Para isso, efetua-se o mesmo procedimento descrito anteriormente, ou utilizar o editor para as EDO(s).

Para iniciar o editor de EDO(s), o usuário deve clicar na região mostrada na Figura 6 onde está escrito "*Click to create a page of ODEs*". Aparece, então, uma pequena caixa onde deve ser escrito o nome da página e, ao clicar em *OK*, automaticamente aparece uma janela, mostrada na Figura 7, onde o usuário irá escrever o conjunto de equações diferenciais relacionadas ao fenômeno físico que se pretende simular.

Para se escrever a EDO, deve-se inicialmente escolher a variável independente. No exemplo, sistema massa-mola, utilizado nessa discussão, a variável independente é o tempo (*time*), cuja letra representativa é o *t*, como mostrado anteriormente na Figura 5 (Janela *Variables* completa). Isso pode ser feito de duas formas. A primeira é digitar a letra *t* no campo com o rótulo *Indep. Var.* mostrado na Figura 7, na próxima página.

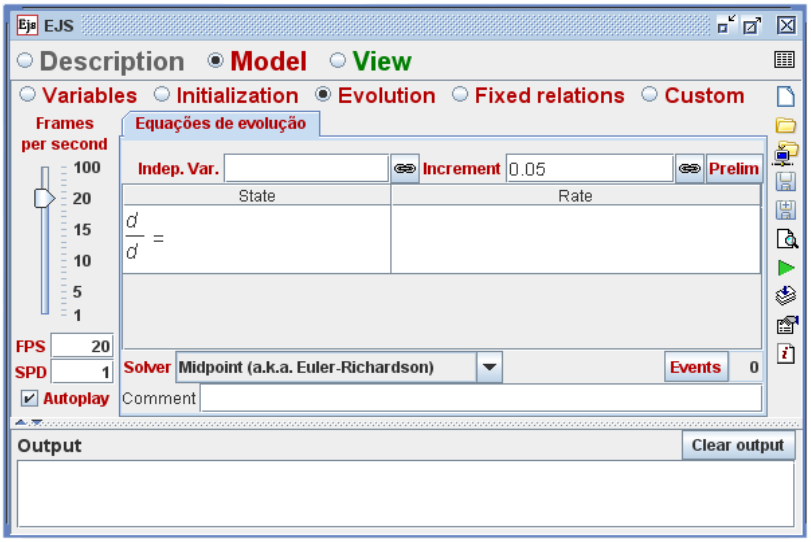


Figura 7 – Editor de EDOs ativado.

Fonte: Easy Java Simulations

A segunda forma é clicar no ícone "corrente", símbolo  , à direita desse campo. Imediatamente uma caixa de diálogo, denominada *List o suitable model variables*, aparece com a lista das variáveis que podem ser usadas nesse campo (Figura 8). Escolhe-se nessa caixa a variável desejada (linha em verde) e ao clicar em *OK*, a letra correspondente aparece no campo *Indep. Var.*

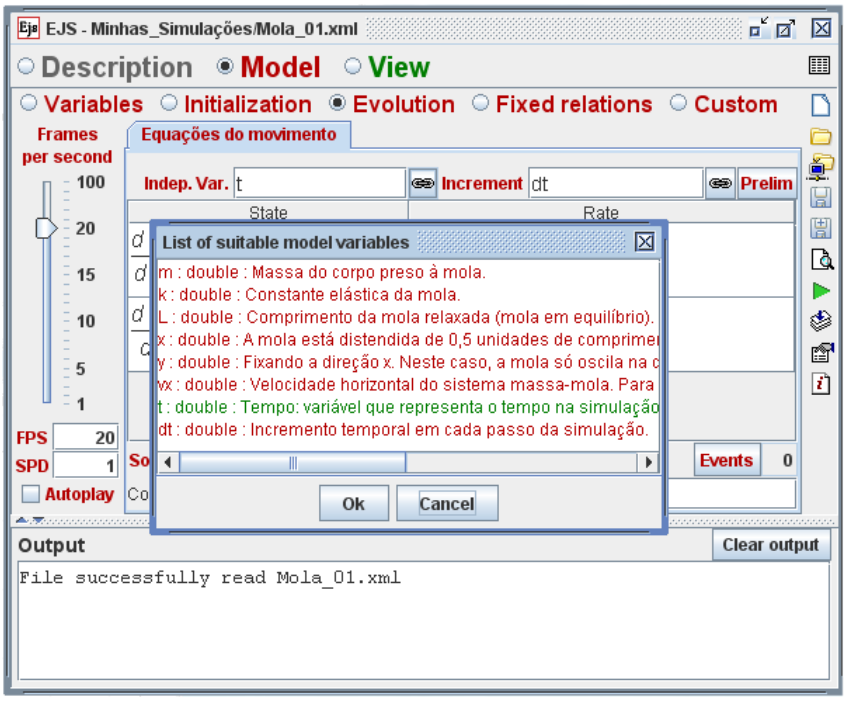


Figura 8 – Formas de se declarar a variável independente *t*.

Fonte: Easy Java Simulations

A próxima etapa é o de especificar o valor do incremento para a variável independente, o que corresponde ao "passo" (repetição computacional) da evolução. O passo pode ser uma variável da lista ou uma constante. No primeiro caso, pode-se usar um dos métodos descritos acima para indicar a escolha. Se o valor do incremento for constante, simplesmente digita-se o número no campo *Increment* (incremento). Ao se observar a Figura 7 (p. 43), nota-se o valor 0,05 que já foi declarado em *Variables*. Na Figura 8 (p. 43), entretanto, aparece o símbolo *dt*, cujo valor pode ser digitado diretamente ou escolhido da lista de variáveis. De qualquer forma, o efeito final é exatamente o mesmo.

Finalmente, para se escrever a EDO ou o sistema de equações diferenciais, deve-se clicar (duplo clique) na região onde aparece o símbolo $\frac{d}{d}=$, mostrado na Figura 7, abaixo da palavra *State*, e digitar a variável de estado desejada para cada uma das equações do sistema que se deseja derivar em relação à variável independente. Outra opção é a de clicar, nessa mesma região, com o botão contrário do mouse. Dessa forma, na tela do monitor aparecerá o menu *popup* com as opções: *Select a state variable*, *Insert before this equation*, *Insert after this equation* e *Remove this equation*. As três últimas opções permitem efetuar a edição da página, ou seja, inserir ou apagar uma equação. Escolhendo-se a opção *Select a state variable*, o menu *List o suitable model variables*, o mesmo da Figura 8, é ativado e o usuário seleciona a variável de interesse.

Na coluna da direita, abaixo do título *Rate*, é escrito (digitado) o valor, ou a variável, ou a expressão correspondente à taxa de variação temporal da variável dependente, isto é, a derivada. Para indicar cada taxa, o usuário digita o nome da variável (duplo clique na região), ou da expressão matemática correspondente, ou selecione-a através da lista de variáveis, como descrito anteriormente.

A Figura 9 mostra o resultado final da página correspondente ao exemplo sistema massa-mola. É importante salientar a forma de sistema de equações diferenciais que aparece na página. Isso acontece pelo fato de EJS não apresentar um método direto de resolução de equação diferencial ordinária de segunda ordem. Dessa forma a equação (5.5.1) é escrita como:

$$\begin{cases} \dfrac{dx}{dt} = vx \\ \dfrac{dvx}{dt} = -\dfrac{k}{m}\cdot\left(x - L\right) \end{cases} \qquad (5.5.5).$$

Observa-se, ainda na Figura 9, que no campo denominado *Solver* aparece o nome do método numérico utilizado na resolução da equação diferencial. O EJS fornece alguns métodos numéricos como os de Euler, Euler-Richardson, Runge-Kutta, etc., que podem ser visualizados ao se clicar na seta ao lado do campo onde aparece o nome do método numérico. As descrições específicas desses métodos podem ser encontradas em livros de Cálculo Numérico. Não é nosso objetivo, por enquanto, apresentá-los nesse trabalho.

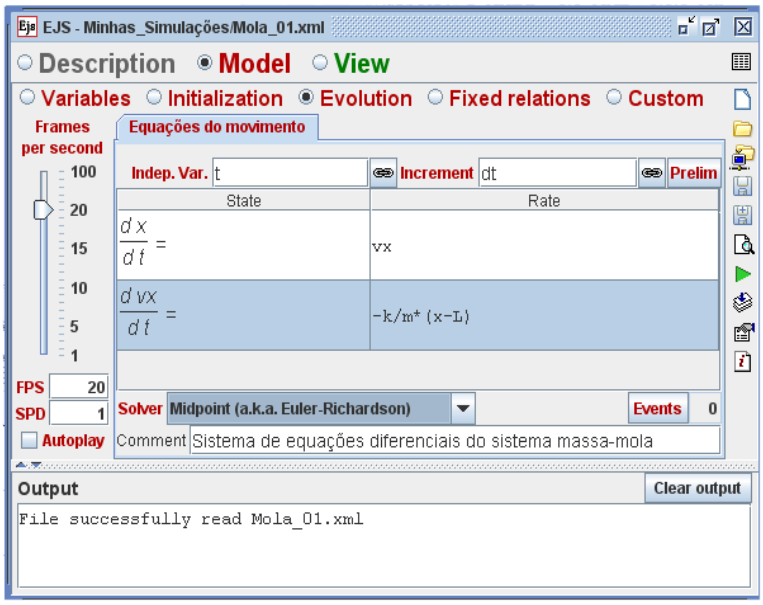


Figura 9 – Equações de evolução do Sistema massa-mola.

Fonte: Easy Java Simulations

4.5.5 AS PÁGINAS *FIXED RELATIONS* E *CUSTOM*

O painel *Fixed relations*, uma vez ativado, é utilizado para se escrever, em código Java, relações pré-determinadas entre variáveis do sistema físico. No exemplo que está sendo apresentado, não há necessidade de tais relações e, por isso, a página se apresenta em branco, como mostra a Figura 10. Repetimos, novamente, que o EJS permite simulações sofisticadas e para criá-las, é necessário que o usuário tenha bom conhecimento dessa ferramenta. O nosso objetivo é, repetindo-nos, o de apresentar a ferramenta àqueles (professores, alunos, pesquisadores, etc.) que estiverem motivados a criar uma simulação em função de sua própria ideia e objetivo que pretende atingir.

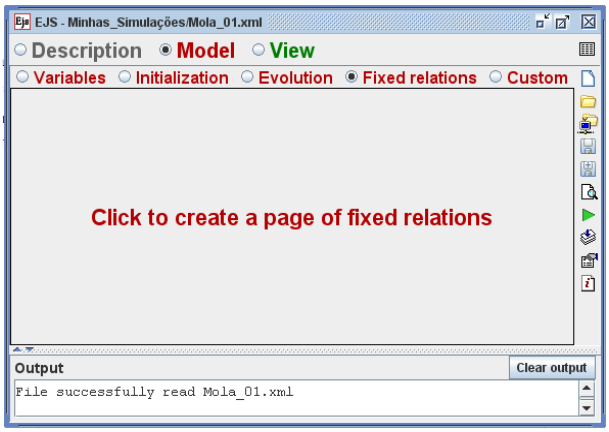


Figura 10 – O painel *Fixed relations*.

Fonte: Easy Java Simulations

Ao se ativar o botão correspondente a *Custom*, imediatamente aparece um painel onde o usuário terá a opção de escrever algumas funções ou sub-rotinas em código *Java*. Essa ferramenta é usada para tornar a escrita do programa mais elegante e criar controles de ação para a execução da simulação. No exemplo do sistema massa-mola não há necessidade dessas funções e a deixamos em branco, como mostrado na Figura 11.

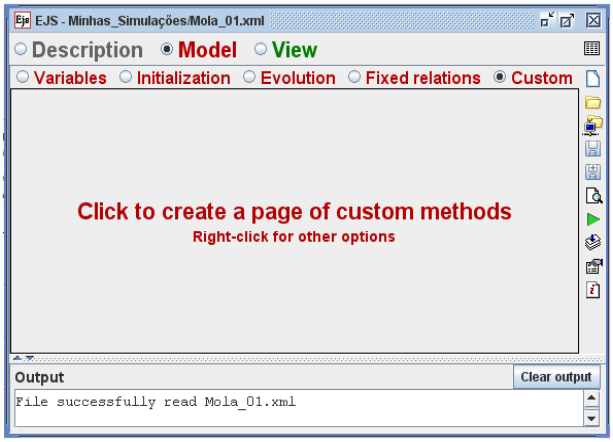


Figura 11 – O painel *Custom*.

Fonte: Easy Java Simulations

Como mencionado nas seções anteriores, ao se ativar o painel de *Fixed relations* ou *Custom*, através do clique do botão do *mouse* na região central do painel, aparece a caixa de diálogo onde o usuário deve digitar um nome. Ao clicar em *OK*, aparece a região onde será escrito o código em *Java*. Assim, a tarefa relacionada à *Model* está finalizada.

Com a prática e ao se elaborar simulações mais sofisticadas, muitos problemas irão aparecer! Este trabalho de dissertação está muito longe de ser um manual completo sobre EJS.

5 CRIANDO A JANELA DE VISUALIZAÇÃO

5.1 A INTERFACE GRÁFICA

A criação de uma interface gráfica, isto é, a janela que permite visualizar a evolução da simulação e a alteração de alguns valores através de ações do usuário, exige do programador muita dedicação, profundo conhecimento das técnicas avançadas de programação e tempo para se efetuar pesquisas nos manuais de referência, escolhendo, assim, as bibliotecas e rotinas gráficas adequadas à simulação que se pretende criar.

O estágio avançado das técnicas de programação permite conceber simulações com alto grau de visualização e interatividade. Entretanto, se a simulação for concebida com propósitos pedagógicos, como o EJS, torna-se necessário utilizar um programa que permita a visualização do fenômeno simulado e, também, a interação do usuário com a simulação. Essa é propriamente a terceira e última parte da elaboração da simulação.

O EJS utiliza uma biblioteca gráfica em código *Java* simplificada com o objetivo de se obter a maximização de seu uso por parte da pessoa que está preparando a simulação. Essa biblioteca se baseia no *Java´s Swing Toolkit* e nas ferramentas criadas por Wolfgang Christian, do Davidson College, e Francisco Esquembre da Universidade de Murcia. Ambos os conjuntos de ferramentas são suportadas por *Java 2* e seus *plug-in*, podendo ser visualizados pela maioria dos *browsers* modernos voltados a *web*.

Uma interface gráfica no EJS é criada através da construção de uma árvore onde são estruturados alguns elementos gráficos selecionados, como mostra a coluna esquerda da Figura 12 (p. 48). Um elemento gráfico pode ser considerado um tijolo da interface que ocupa uma área específica da tela do computador e realiza uma tarefa particular para o qual foi criado. Existem elementos de diferentes classes como painéis, rótulos, botões, flechas, partículas, etc., como mostrado na coluna direita da Figura 12.

A aparência gráfica de cada elemento é principalmente determinada por sua classe. Cada elemento tem uma classe própria, denominada de propriedade, que poderá ser modificada pelo usuário de acordo com as suas necessidades envolvidas na programação. Alguns elementos têm propriedades de um tipo especial, denominadas de ações, que especificam a resposta decorrente da interação do usuário com o elemento. Essa interação ocorre através do uso do *mouse* ou digitando “algo” (um número, por exemplo) diretamente no elemento que suporta essa ação, através do teclado do computador.

Os nomes dos elementos das classes são descritivos em função de seu uso natural. No texto elaborado como produto final dessa dissertação são apresentadas algumas

tabelas com as correspondentes descrições de alguns elementos que podem ser utilizados no EJS e agrupados por funcionalidade.

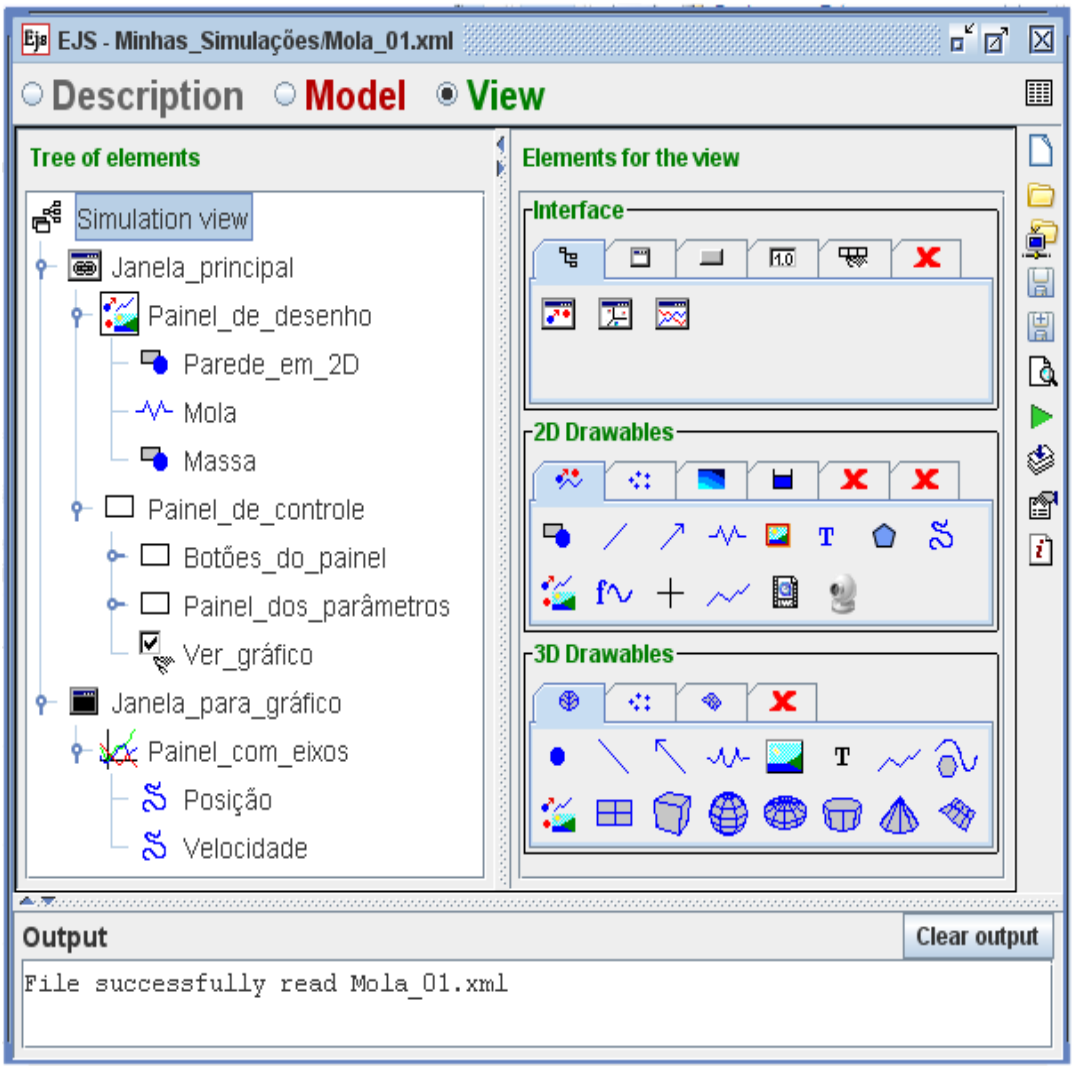


Figura 12 – Árvore estruturada com elementos selecionados – janela pronta.

Fonte: Easy Java Simulations

Sem dúvida, a parte importante, e mais complicada, na criação e preparação das simulações com o uso do EJS é aprender o que faz cada um desses elementos, como usá-los e personalizá-los. Essa é a parte que exige o maior tempo de dedicação na aprendizagem do EJS, para aqueles que não estão acostumados com as técnicas de computação orientada a objetos.

5.2 *LAYOUTS*

Um elemento da interface gráfica pode conter outros elementos gráficos. O primeiro tipo é denominado de “pai”, aquele que contém outros elementos gráficos, enquanto o segundo é denominado de “filho”, aquele que está contido. Essa nomenclatura de “pai” e “filho” é utilizada pelo próprio Esquembre (2005). Como um mesmo elemento pode ser “pai” e “filho”, torna-se necessário construir uma estrutura hierárquica na forma de árvore, onde esses elementos são colocados, para evitar uma confusão que poderá comprometer a execução da simulação.

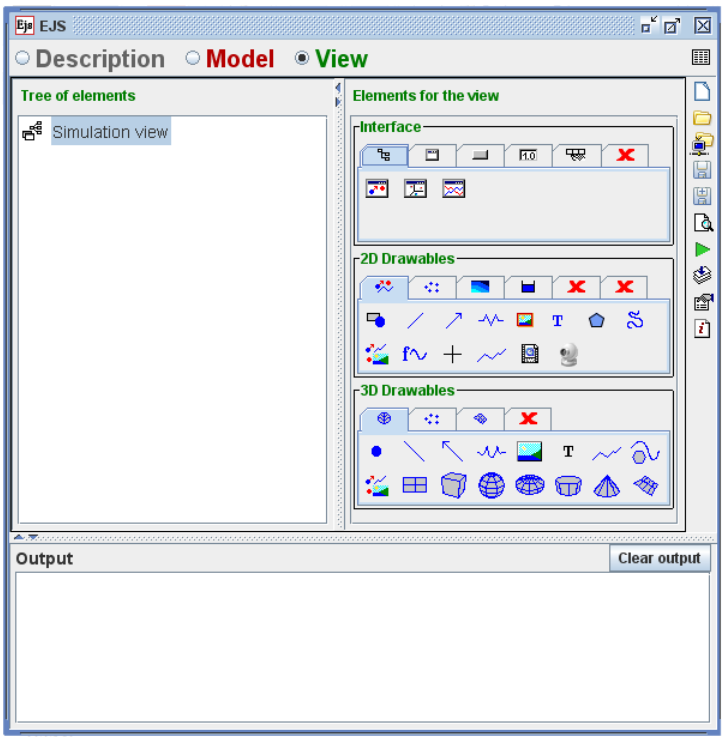


Figura 13 – Inicialização da janela de visualização – *View*.

Fonte: Easy Java Simulations

A Figura 13 mostra o estágio inicial da janela de visualização, logo após a seleção do botão *View*. Observa-se na coluna esquerda da janela o título *Tree of elements* (árvore de elementos) e abaixo uma região em branco com um ícone de rótulo *Simulation view* (visão da simulação). Isto constitui a raiz da árvore. Na verdade ele não é um elemento gráfico, mas serve como um elemento a partir do qual serão colocados outros elementos.

O primeiro “filho” na raiz da árvore é chamado de janela principal (*main window*), que é aquele que aparece em primeiro lugar na tela do monitor do computador e, também, que EJS inclui na página HTML ao gerar a simulação.

Existem famílias de *containers*, isto é, aquelas que guardam elementos básicos (*interface*) e outras que guardam os elementos de desenho (*drawables*). O primeiro tipo guarda outros *containers* e elementos básicos, que são utilizados para se criar o esqueleto da interface de uso da simulação. O segundo tipo é feito de *containers* especializados que só guardam elementos de desenho. Neste caso em particular, eles desempenham a função de "filhos". Esses *containers* e seus "filhos" de desenho são utilizados para as montagens das animações e construção de gráficos do fenômeno que pretende simular (vide Figura 13, coluna esquerda do painel *View*).

Uma propriedade importante dos *containers* do primeiro tipo (*containers* que guardam outros elementos) é o *layout*, responsável pela aparência da simulação na tela do monitor. Quando um "filho" é adicionado ao *container*, o "pai" estipula a sua posição e tamanho de acordo com o espaço disponível da demanda de outros "filhos" e de sua própria "polícia de guarda" (ESQUEMBRE, 2005). Isso é denominado de polícia de *layout*, ou simplesmente *layout*. Alguns *layouts* fornecem aos "filhos" a oportunidade de "escolher" a posição junto ao "pai", embora em muitos outros essa posição seja uma consequência da "ordem de nascimento do filho".

Os *layouts* são úteis para se controlar a aparência da interface, pois o usuário poderá mudar o tamanho da janela principal, o comprimento das fontes ou dos textos. Quando o tamanho do "pai" é alterado, automaticamente ocorre o redimensionamento de seus "filhos".

Segundo os autores de EJS, "depois de um pouco de prática" o uso dos *layouts* se torna natural! Nós, entretanto, não compartilhamos desse pensamento.

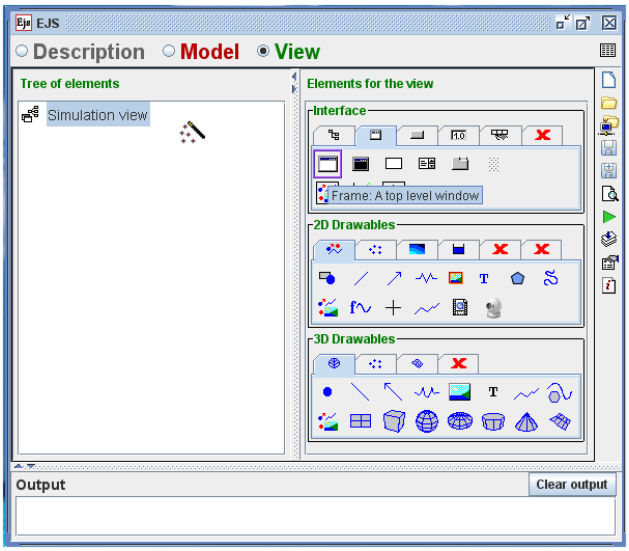


Figura 14 – Acrescentando os "pais" e "filhos" na árvore de elementos com o uso da *varinha mágica*.

Fonte: Easy Java Simulations

Para se adicionar um elemento na raiz da árvore, basta escolher o tipo de elemento que aparece na coluna *Elements for the view* e efetuar um clique do botão do *mouse* sobre este elemento. A cor de fundo deste ícone selecionado se altera, indicando que a seleção está ativada. Observa-se, também, que o cursor do mouse, que por *default* é uma seta, se altera em uma *varinha mágica*, mostrada na Figura 14. Leva-se, então, essa *varinha mágica* até o

elemento, do tipo *container*, na árvore de componentes, ou seja, clicando sobre o ícone *Simulation view*. Semelhante às ações descritas anteriormente surge na tela do monitor uma janela convidando o usuário a fornecer um nome ao elemento que está "sendo criado". O nome escolhido é Janela principal, como mostrado na Figura 15. Ao finalizar essa etapa, clicando-se em *OK*, o elemento aparece inserido na árvore (Figura 16) e uma nova janela, que será usada para mostrar todo o desenvolvimento da simulação, aparece na tela do monitor ao lado do painel de EJS, ou em outra posição que depende da configuração do computador (Figura 17).

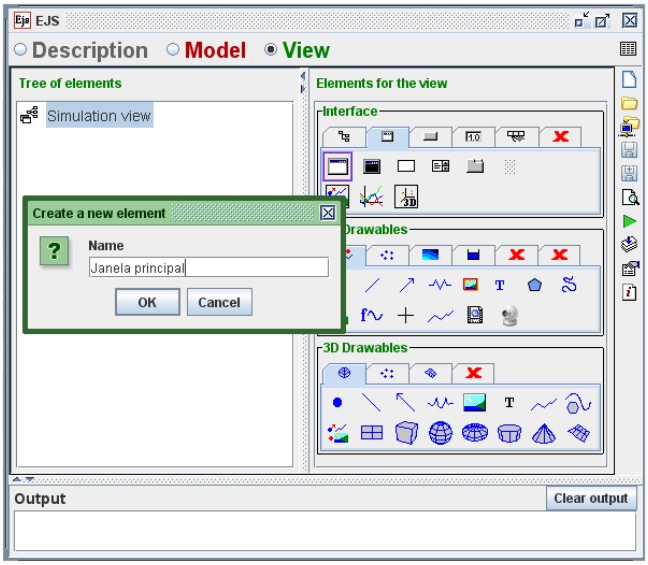


Figura 15 – Nome do elemento que está sendo inserido na árvore de elementos.

Fonte: Easy Java Simulations

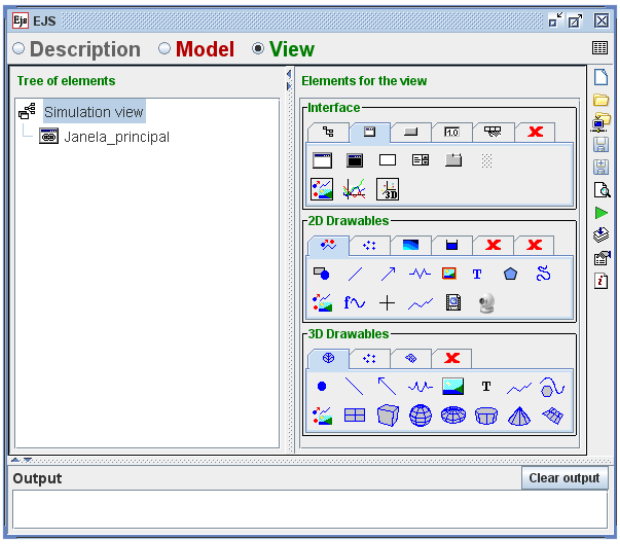


Figura 16 – Elemento "Janela_principal" adicionado na árvore de elementos.

Fonte: Easy Java Simulations

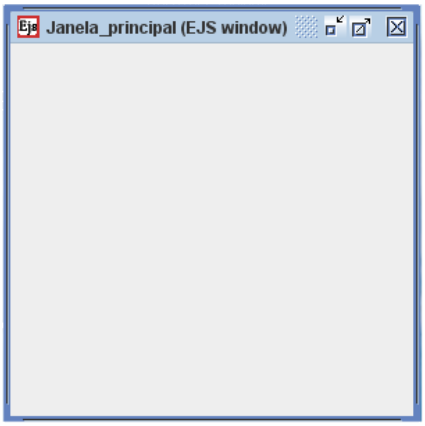


Figura 17 – Apresentação da "Janela_principal" na tela do monitor.

Fonte: Easy Java Simulations

Aparentemente, a tarefa de se criar uma janela de visualização é muito simples. Entretanto, é o usuário que deve fornecer os parâmetros relacionados com a posição dos elementos, ou seja, a aparência (*layout*) da simulação mostrada na tela do computador. Para

isso, faz-se necessário conhecer as propriedades dos elementos, algumas delas descritas no tutorial. Nas próximas seções, entretanto, estaremos apresentando a descrição de alguns dos elementos, de suas propriedades e como acessar a janela de propriedades para poder efetuar modificações que o usuário desejar. Por enquanto, destacamos que os *layouts* são de cinco tipos cujas funções são descritas abaixo, bem como a aparência de cada um deles na tela do monitor.

a. *Flow* – Distribui os "filhos" em uma linha, da esquerda para a direita, de forma similar às palavras escritas em um texto. Os "filhos" podem estar na esquerda, no centro ou na direita. A Figura 18 mostra a disposição de alguns botões de comando utilizando-se a opção *Flow*.

Figura 18 – *Layout* do tipo *Flow*.

Fonte: Easy Java Simulations

b. *Border* – Distribui os "filhos" em uma das cinco posições: *up* (em cima), *down* (em baixo), *left* (esquerda), *right* (direita) e *center* (centro), como mostra a Figura 19. Os elementos podem ter dimensões diferentes.

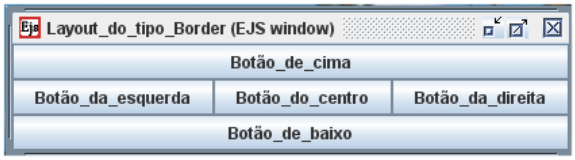


Figura 19 – *Layout* do tipo *Border*.

Fonte: Easy Java Simulations

c. *Grid* – Localiza os "filhos" em uma tabela retangular com linhas e colunas igualmente dimensionadas e de acordo com a indicação do usuário (Figura 20).

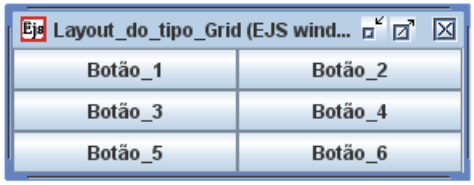


Figura 20 – *Layout* do tipo *Grid.*

Fonte: Easy Java Simulations

d. *Horizontal Box* – Sua função é similar ao *grid*, só que os "filhos" ficam dispostos em linha (Figura 21).

Figura 21 – *Layout* do tipo *Horizontal box*.

Fonte: Easy Java Simulations

e. *Vertical Box* – Sua função também é similar ao *grid*, com os "filhos" dispostos em uma coluna simples. Os "filhos" podem apresentar tamanhos diferentes (Figura 22).

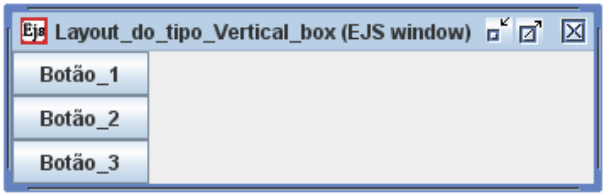


Figura 22 – *Layout* do tipo *Vertical box*.

Fonte: Easy Java Simulations

Em todos os *layouts* apresentados é possível alterar parâmetros que indicam a separação vertical e a horizontal dos "filhos", cor, dimensões, etc.

5.3 CRIANDO UMA JANELA DE VISUALIZAÇÃO

Criar uma janela de visualização, cujo objetivo é mostrar o desenvolvimento da simulação através de animação ou gráficos, consiste em gerar uma estrutura de elementos agrupados em forma de árvore de elementos, como discutido no Capítulo 5.1 e mostrado na Figura 12 (p. 48). Essa estrutura deve incluir:

a. Elementos de visualização do estado do modelo.
b. Elementos que permitem a interatividade do usuário com a simulação.
c. *Containers* que guardam todos os outros elementos.

Essa é a fase mais trabalhosa de todo o planejamento da simulação, pois o usuário deve saber o que faz cada elemento que aparece na coluna direita *Elements for the view*. Por qual começar?

5.3.1 *CONTAINERS*

Como mencionado da seção 5.2, existem elementos que são denominados de *containers*, ou seja, elementos que guardam outros elementos. Para identificá-los, deve-se acionar através do clique do *mouse* na segunda aba correspondente ao grupo *Interface*, como mostra a Figura 16 (p. 51). Verifica-se a existência de oito ícones, cujos nomes são: *Frame*, *Dialog*, *Panel*, *SplitPanel*, *TabbedPanel*, *DrawingPanel*, *PlottingPanel* e *DrawingPanel3D*. A função de cada um deles está descrita no texto referente ao produto final dessa dissertação.

Os *containers Panel*, *SplitPanel*, *TabbedPanel*, *DrawingPanel*, *PlottingPanel* e *DrawingPanel3D*, não podem ser adicionados na raiz, isto é, eles só podem ser adicionados nos *containers Frame* ou *Dialog*.

Para cada um desses elementos existe um conjunto de propriedades. Uma vez inserido na árvore, ao se clicar duas vezes sobre o ícone desejado, aparece na tela do monitor uma tabela onde é possível modificar os parâmetros (propriedades) relacionados com a aparência do elemento durante a execução da simulação. Existem muitos recursos! E como já dissemos anteriormente, alguns deles são apresentados no texto final da dissertação. A Figura 23 mostra a tabela de propriedades, já editada, para o *container Frame*.

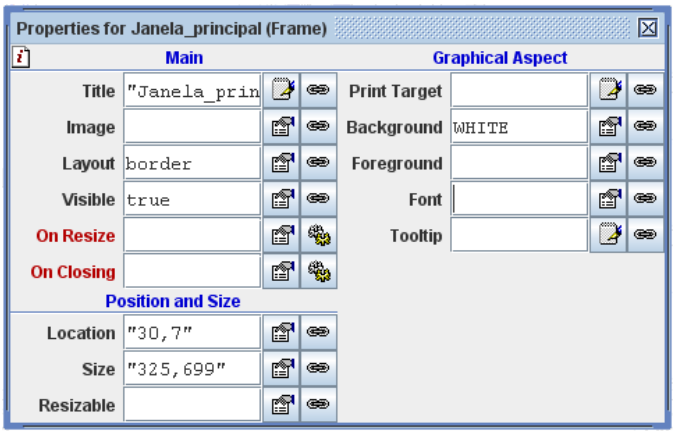


Figura 23 – A tabela de propriedades do *container Frame*.

Fonte: Easy Java Simulations

5.3.2 ELEMENTOS BÁSICOS

Esse grupo é composto por uma série de elementos de *Interface* que podem ser usados para decorar a janela, mostrar e editar as variáveis e, também, como um "gatilho" para produzir determinados tipos de ação. Os elementos básicos podem ser adicionados aos *containers* do primeiro tipo, mas não podem conter outros elementos.

No texto final da dissertação são mostrados alguns ícones dos grupos e dos elementos básicos pertencentes a cada grupo, e algumas funções correspondentes. Por enquanto, o usuário deve saber que existem famílias de elementos agrupados de acordo com funções semelhantes. Isso pode ser visualizado através das abas que aparecem na região *Interface*. Por exemplo, ao se ativar a terceira aba, *Buttons and decoration*, tem-se um conjunto de elementos relacionados, como mostrados na Figura 24.

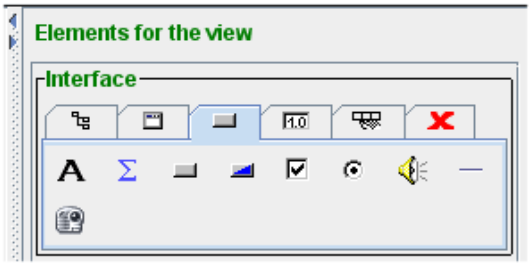


Figura 24 – Ativação da família correspondente a *Buttons and decoration*.

Fonte: Easy Java Simulations

Efeito semelhante é obtido ao se ativar a quarta aba (Figura 25) ou a quinta aba (Figura 26).

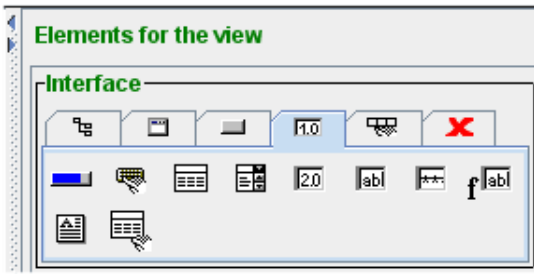


Figura 25 – Ativação da família correspondente a *Input and output*.

Fonte: Easy Java Simulations

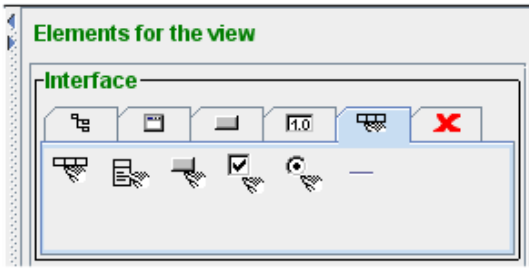


Figura 26 – Ativação da família correspondente a *Menu elements*.

Fonte: Easy Java Simulations

Analogamente ao que foi descrito anteriormente, uma vez inserido um elemento básico em um container, é possível através do duplo clique do *mouse* ativar a tabela referente à propriedade do elemento. A Figura 27 mostra a propriedade referente ao elemento *TwoStateButton* e se o usuário desejar, ele poderá efetuar as modificações necessárias.

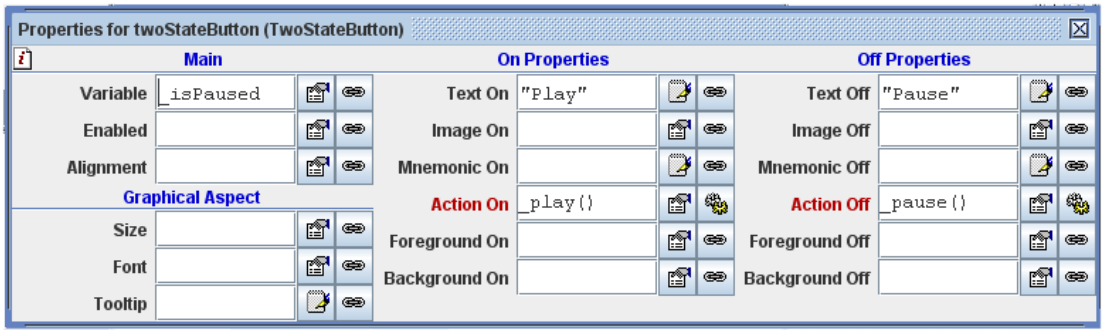


Figura 27 – Tabela referente à propriedade do elemento *TwoStateButton*.

Fonte: Easy Java Simulations

5.3.3 ELEMENTOS DE DESENHO – *2D DRAWABLES E 3D DRAWABLES*

O conjunto de elementos de desenho é a principal contribuição do *Open Source Physics* para o EJS. Ele consiste de uma série de elementos de visualização que podem ser incluídos em *containers* do segundo tipo, com a função de mostrar a animação gráfica e permitir a visualização dos diferentes estados do modelo. Essas animações gráficas podem ser simples ou muito sofisticadas, dependendo do grau de conhecimento de quem está criando a simulação, e incluem partículas (representadas por elipses ou retângulos), setas, imagens, linhas, campos vetoriais, etc. Os elementos de desenho também estão agrupados por

funcionalidade. Ativando-se cada aba dessas seções, observa-se o conjunto de elementos correspondentes e selecionado um deles, já inserido em um container da árvore de elementos, é possível ativar a tabela de propriedades e efetuar as alterações necessárias. O procedimento é semelhante ao descrito nas seções anteriores.

No texto elaborado como produto dessa dissertação são fornecidas funções e propriedades de alguns dos elementos contidos em *2D Drawables* e *3D Drawables*.

5.4 A JANELA DE VISUALIZAÇÃO DO SISTEMA MASSA-MOLA

O usuário já tem algumas condições de iniciar a construção da janela de visualização de sua simulação. Entretanto, ele deve ter claro em mente o que se deseja com a simulação. Assim, é aconselhável se fazer um esboço da janela de visualização, como planejar onde os elementos de animação devem se apresentar, se a simulação apresentará, ou não, gráficos, etc. Dessa forma, o usuário poderá criar janelas diferentes para mostrar ações diferentes, isto é, uma janela de visualização para mostrar a evolução da animação do fenômeno, outra (ou outras) para se mostrar o gráfico, etc. De fato podem ser acrescentadas quantas janelas o usuário desejar.

No exemplo do sistema massa-mola, será mostrada uma janela para evolução da animação e outra o gráfico da posição e velocidade, em função do tempo, do corpo que oscila com a mola. A Figura 12, p. 48, mostra a árvore de elementos finalizada para a simulação do sistema massa-mola. Cumpre-se, agora, justificar a escolha e especificar as ações decorrentes dela.

O primeiro elemento selecionado é o *container Frame* que está com a denominação Janela _principal. Como o nome indica, ela é a janela que irá mostrar a animação. Ela está dividida em duas regiões, uma onde fica a animação e outra reservada para se colocar os botões de controle. Deve-se tem em mente que essa disposição foi previamente planejada pelo usuário (autor da simulação).

A Figura 28 mostra o *Layout* referente à tabela das propriedades da Janela_principal (Figura 23, p. 54). Essa janela é acionada após se clicar sobre o ícone ao lado da caixa onde está escrita a palavra *border*.

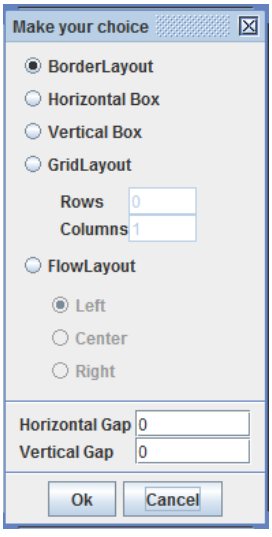


Figura 28 – *Layout* do *container Frame* (*Properties for* Janela_principal).

Fonte: Easy Java Simulations

O usuário deve se preocupar em observar quais são os parâmetros relacionados a cada uma das tabelas e procurar gravar quais são os efeitos decorrentes das mudanças desses valores.

O próximo passo é inserir no contêiner *Frame*, outro contêiner relacionado com o desenho (*DrawingPanel*). Seleciona-se, assim, o ícone correspondente, ativando-o, e através da varinha mágica, clica-se sobre o ícone Janela_principal que já está na coluna *Tree of elements*. Fornecemos um nome, no caso, Painel_de_desenho, e clica-se em *OK*. Um duplo clique sobre o elemento criado e, novamente, aparece uma tabela com as propriedades que o usuário irá modificar. Muitas das propriedades mostradas são dadas por *default*.

A Figura 29 mostra as propriedades do Painel_de_desenho, onde serão colocados os elementos que fazem parte da animação. Deseja-se mostrar neste painel uma mola, uma pequena esfera presa em uma de suas extremidades e a outra extremidade da mola presa a uma parede. Todos esses elementos são fornecidos pelo EJS. Basta efetuar as mesmas operações descritas anteriormente, isto é, selecionando-se o elemento desejado e "depositando-o" sobre o *container* Painel_de_desenho.

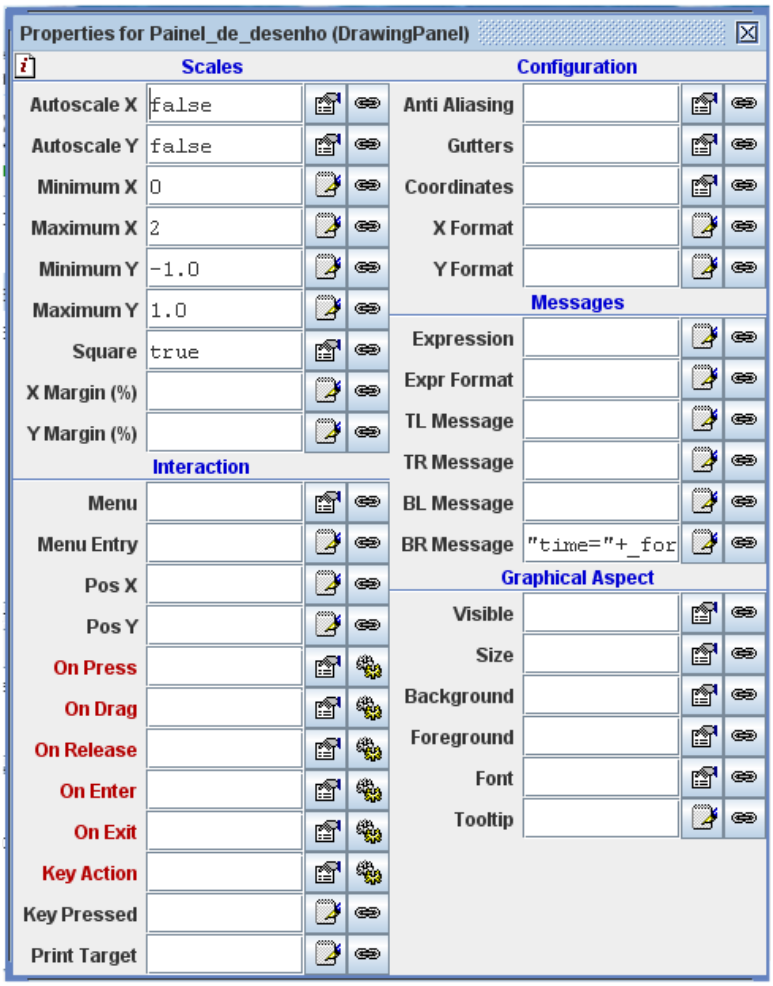


Figura 29 – Propriedades do Painel_de_desenho.

Fonte: Easy Java Simulations

Novamente, a Figura 12 (p. 48) orienta o leitor quanto à disposição dos elementos inseridos, enquanto as Figuras 30, 31 e 32 fornecem, respectivamente, as tabelas de propriedades relativas aos elementos Parede_em_2D, Mola e Massa, ora mencionados.

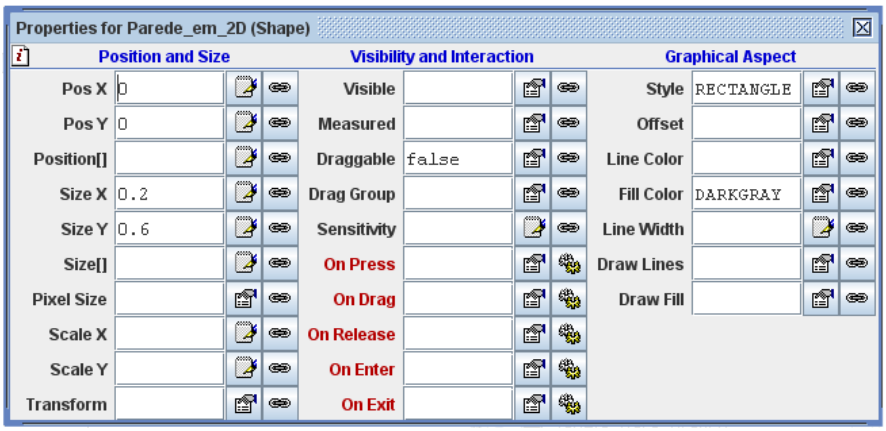


Figura 30 – Propriedades do elemento de desenho Parede_em_2D.

Fonte: Easy Java Simulations

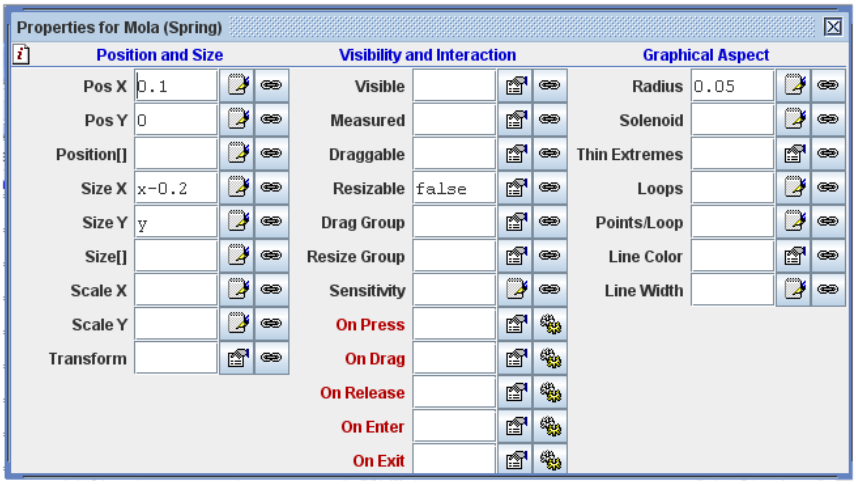


Figura 31 – Propriedades do elemento de desenho Mola.

Fonte: Easy Java Simulations

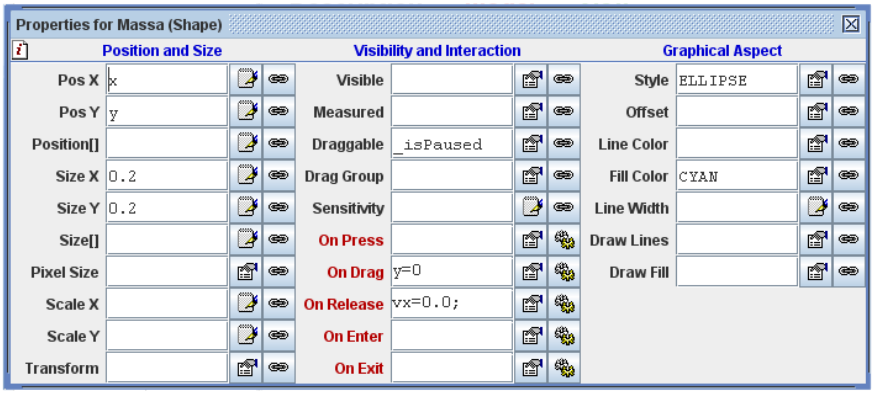


Figura 32 – Propriedades do elemento de desenho Massa.

Fonte: Easy Java Simulations

A terceira etapa da tarefa consiste em criar o *contaiiner* Painel_de_controle. Este contêiner irá guardar outros *containers* onde serão colocados os elementos que permitem a interação do usuário com a simulação. O processo de inserção dos elementos é análogo ao descrito nas seções anteriores e, de agora em diante, não repetiremos essa etapa de descrição.

Tomando-se como referência a Figura 12 (p. 48), observa-se a existência de dois outros *containers*: Botões_do_painel e Painel_de_parâmetros. Ao clicar nos círculos que estão do lado esquerdo dos correspondentes ícones, observamos os respectivos "filhos", mostrados em destaque na Figura 33.

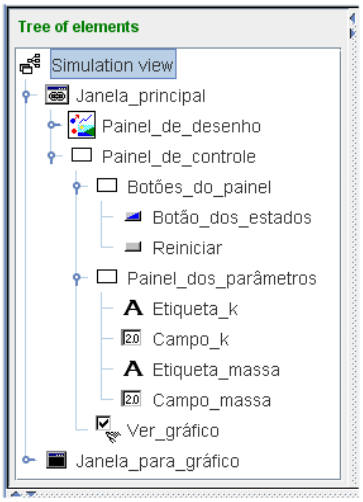


Figura 33 – Destaque dos elementos do Painel_de_controle.

Fonte: Easy Java Simulations

Ao se efetuar o duplo clique, com o botão do mouse, sobre o Painel_de_controle, o usuário irá visualizar as suas propriedades, como mostra a Figura 34. O mesmo procedimento pode ser realizado para visualizar as propriedades dos demais "filhos", mostradas nas Figuras 35 e 36, respectivamente.

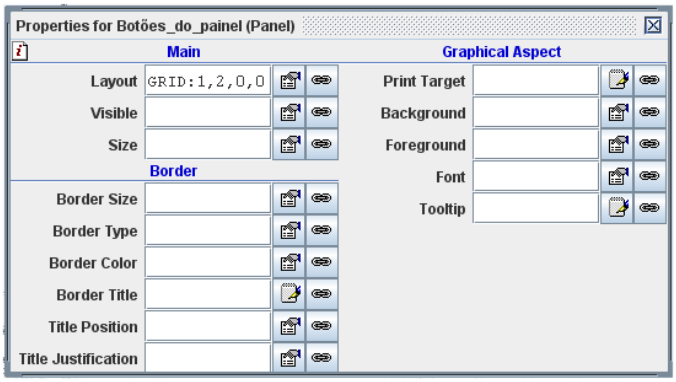


Figura 34 – Propriedades do *container* Botões_do_painel.

Fonte: Easy Java Simulations

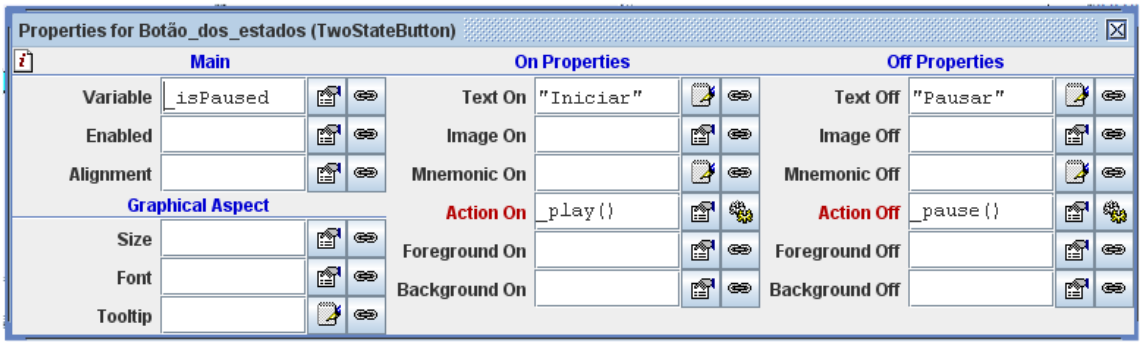


Figura 35 – Propriedades do Botão_dos_estados.

Fonte: Easy Java Simulations

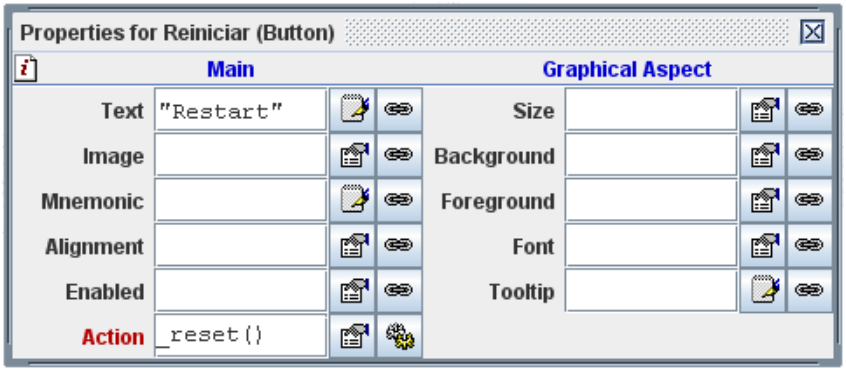


Figura 36 – Propriedades do botão Reiniciar.

Fonte: Easy Java Simulations

Chamamos a atenção do leitor às especificações mostradas nos diversos campos dessas tabelas. Alguns campos estão em branco, mostrando que não é necessário o seu preenchimento. Entretanto, o usuário poderá preenchê-los com valores apropriados e modificar a aparência desses elementos na janela de visualização da simulação.

Analogamente, pode-se obter a visualização das propriedades do Painel_dos_parâmetros, Figura 37, e de seus respectivos "filhos", mostradas nas Figuras 38, 39, 40 e 41, respectivamente. Novamente, observe os valores dos parâmetros.

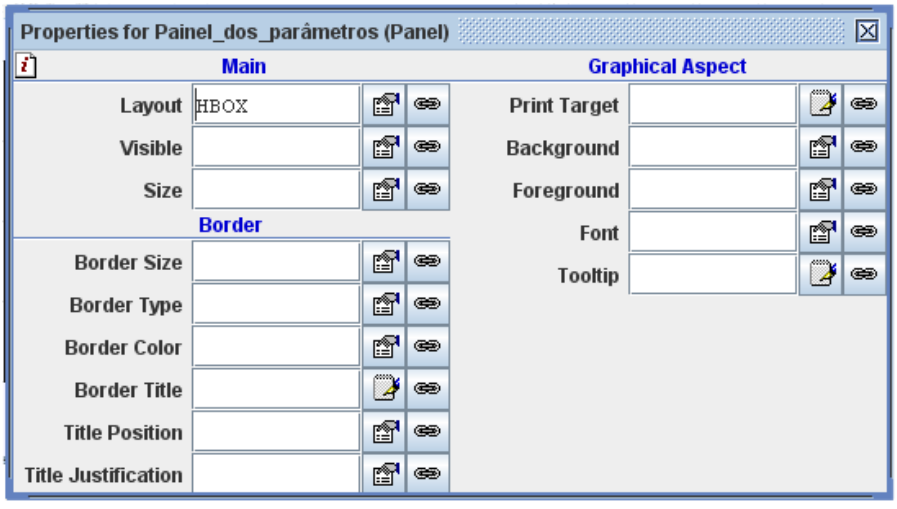


Figura 37 – Propriedades do *container* Painel_dos_parâmetros.

Fonte: Easy Java Simulations

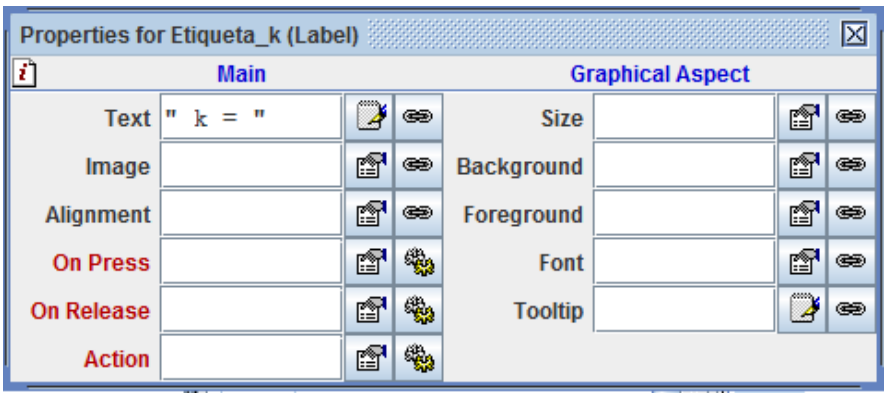


Figura 38 – Propriedades de Etiqueta_k.

Fonte: Easy Java Simulations

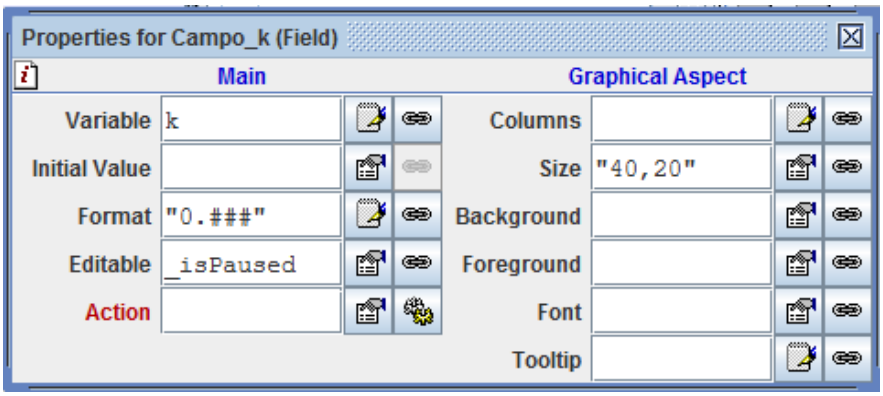


Figura 39 – Propriedades de Campo_k.

Fonte: Easy Java Simulations

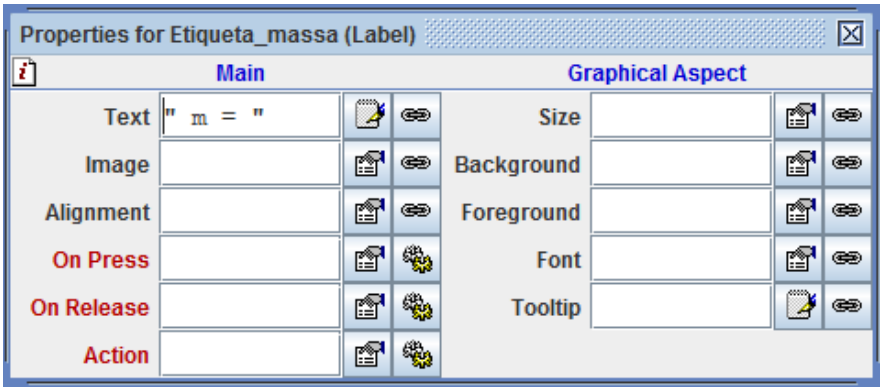


Figura 40 – Propriedades de Etiqueta_massa.

Fonte: Easy Java Simulations

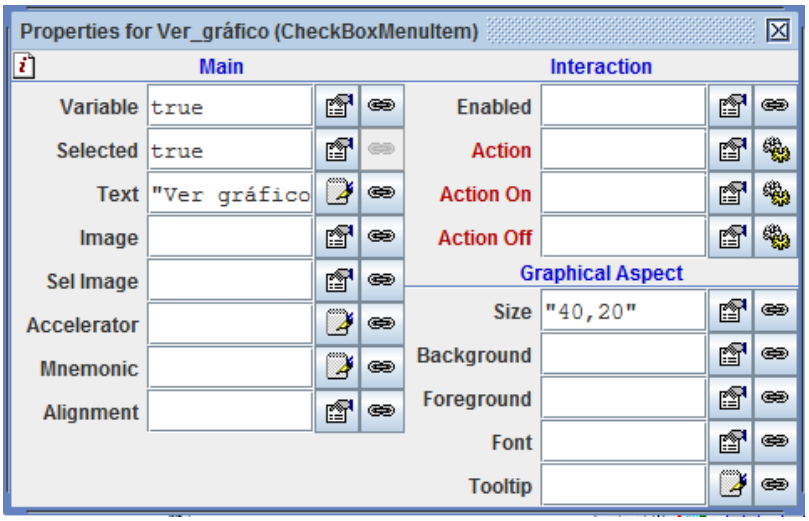


Figura 41 – Propriedades de Campo_massa.

Fonte: Easy Java Simulations

Para completar essa parte, acrescenta-se o elemento Ver_gráfico (vide a Figura 33, p. 59) que serve para ativar a janela referente ao gráfico da evolução do sistema massa-mola que será construído durante a execução da simulação. A Figura 42 mostra o conjunto de propriedades correspondente.

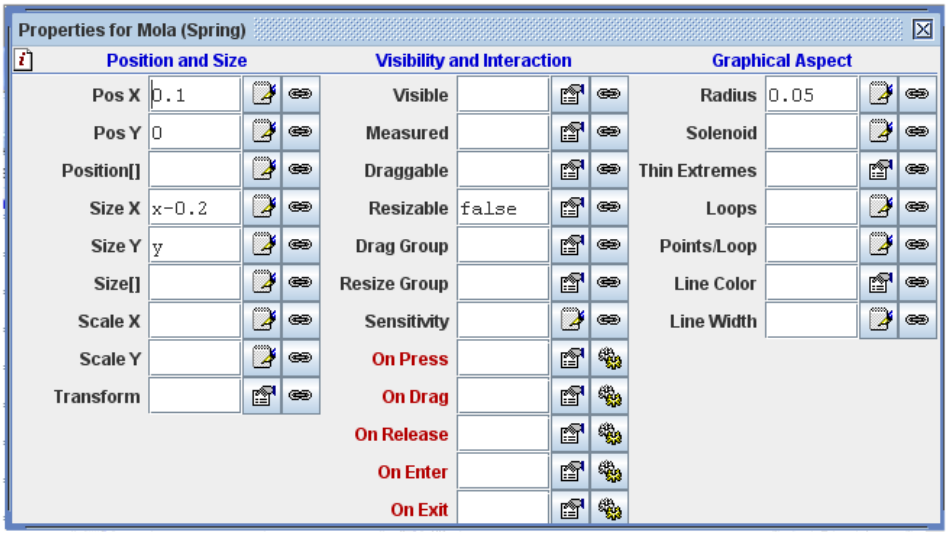


Figura 42 – Propriedades do elemento de desenho Mola.

Fonte: Easy Java Simulations

O resultado final de toda essa ação é mostrado na Figura 43, isto é, a janela de visualização principal da simulação. O usuário pode ativar a simulação através da tecla “Iniciar”. Pode, também, pausar a simulação, recomeçá-la através da mudança da posição da

esfera ou acionado o botão “Restart”. Essas ações demonstram a versatilidade de EJS, isto é, permitir a interatividade entre a simulação e o usuário. Observe que é possível alterar os valores das variáveis “k” e “m”. Basta digitar o valor desejado no correspondente campo e apertar a tecla *enter*.

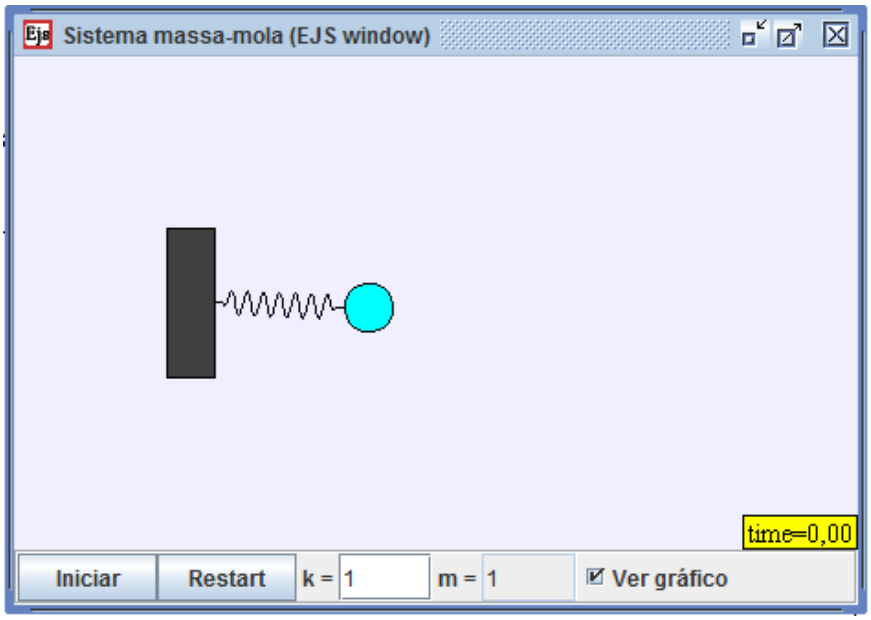


Figura 43 – Janela de visualização principal do EJS exemplificada para a simulação do Sistema massa-mola.

Fonte: Easy Java Simulations

Para se visualizar a evolução da variação da posição e da velocidade do corpo em função do tempo, é possível criar uma nova janela com a função de permitir a visualização dos gráficos. Novamente, recorre-se à Figura 12 (p. 48) para visualizar na coluna *Tree of Elements* o container *View* com o título Janela_para_gráfico. Adicio na-se a ele o container *PlottingPanel*, com o título Painel_com_eixos, que permite a EJS criar um sistema de eixos cartesianos.

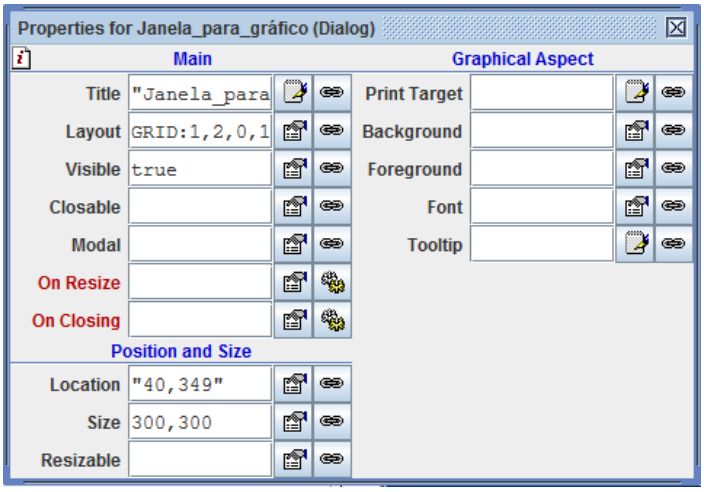


Figura 44 – Propriedades de Janela_para_gráfico.

Fonte: Easy Java Simulations

A Figura 44 permi te visualizar as propriedades da

Janela_para_gráfico, enquanto a Figura 45, a seguir, mostra as propriedades para Painel_com_eixos.

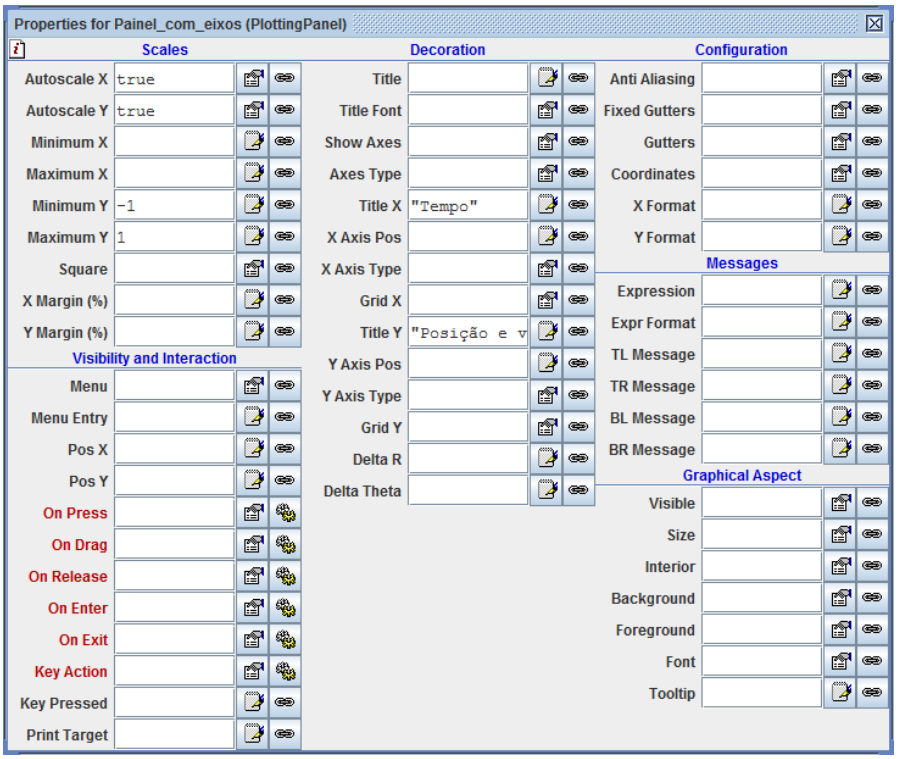


Figura 45 – Propriedades de Painel_com_eixos.

Fonte: Easy Java Simulations

Falta "informar" ao EJS o que deve ser representado no Painel_com_eixos, isto é, o gráfico da posição da extremidade livre da mola em função do tempo e a velocidade do corpo, preso a mola, em função do tempo. Isso é feito através da seleção do elemento *Trail*. A sua função é desenhar a trilha que representa as funções *x(t)* e *v(t)*. Na Figura 12, p. 48, este elemento aparece duas vezes: um para indicar a curva referente à posição em função do tempo, *x(t)*, e o outro para indicar a

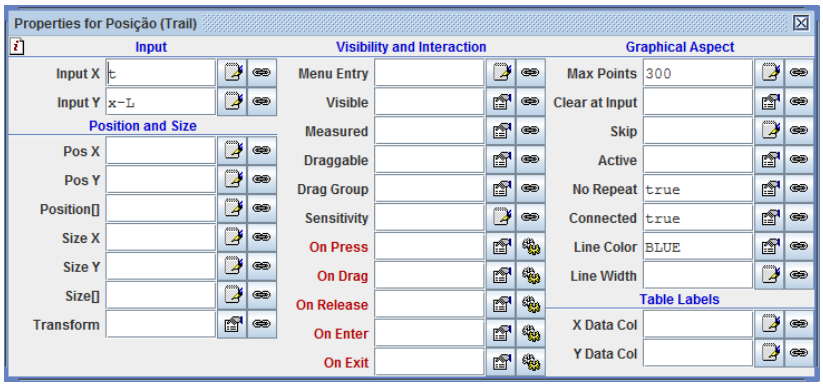


Figura 46 – Propriedades de Posição (*Trail*).

Fonte: Easy Java Simulations

curva referente à velocidade do corpo em função do tempo, *v(t)*. A Figura 46, na página anterior, e a Figura 47 mostram, respectivamente, as propriedades dos elementos Posição e Velocidade.

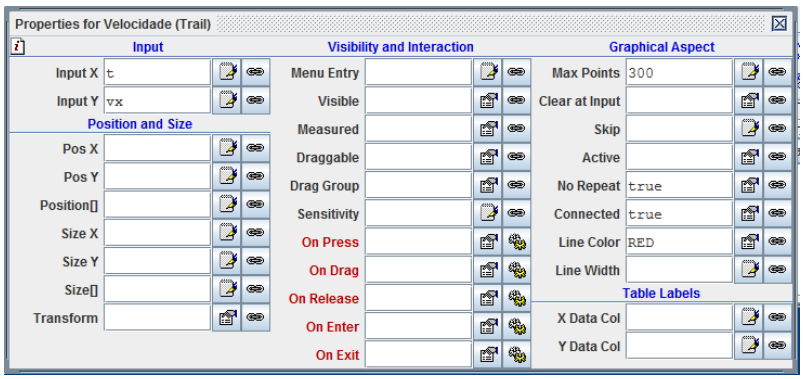


Figura 47 – Propriedades de Velocidade (*Trail*).

Fonte: Easy Java Simulations

Finalizada essa etapa, tem-se a janela de visualização da simulação com o correspondente gráfico (Figura 48). Dessa forma, dizemos que a simulação está finalizada e de acordo com a intenção de quem a elaborou.

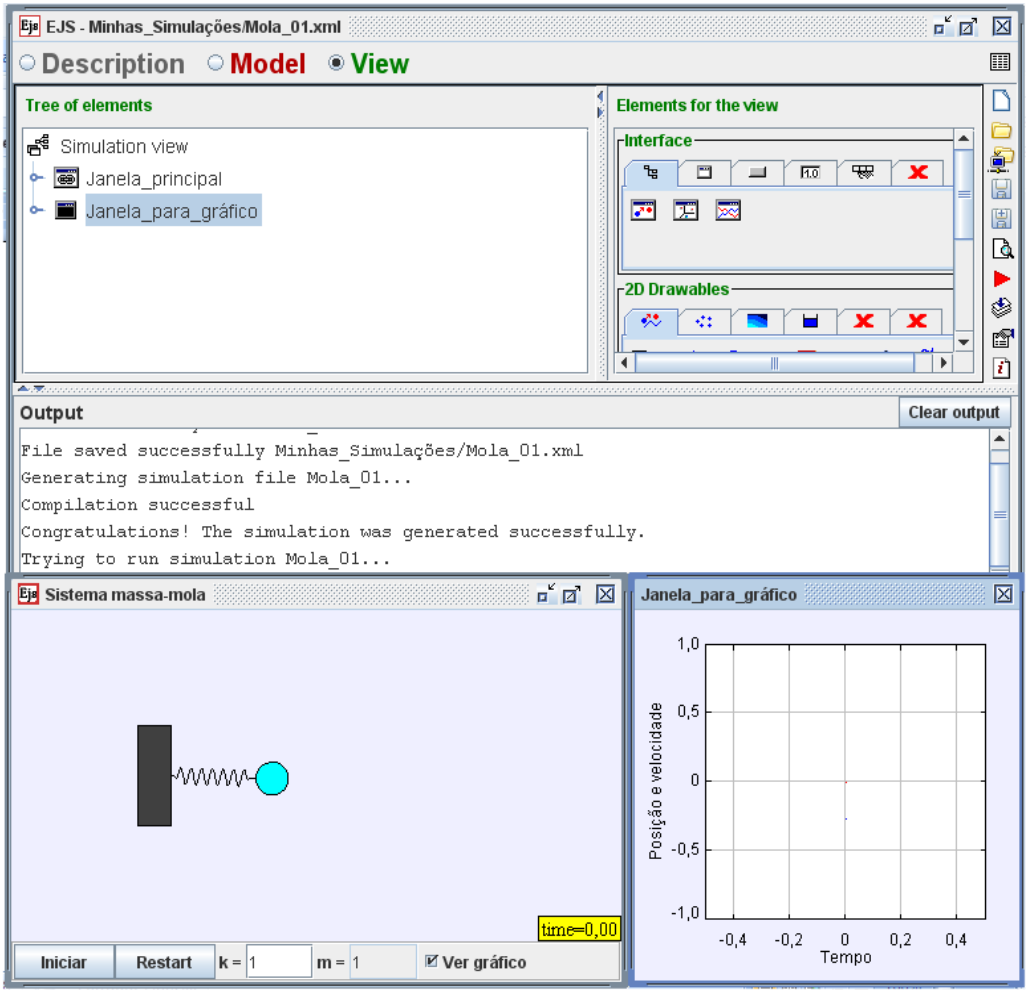


Figura 48 – Estágio inicial da simulação pronta, para o caso do Sistema massa-mola.

Fonte: Easy Java Simulations

Para se executar a simulação, clica-se no ícone da barra de tarefas. O EJS começa o processo de compilação e, se tudo estiver correto, após alguns instantes, tem-se a tela correspondente à Figura 48. Observa-se no campo *Output* a mensagem dizendo que a compilação foi bem sucedida. Caso contrário, o usuário deverá verificar o que diz a mensagem e tentar resolver o problema. Em geral é uma vírgula que aparece no lugar de um ponto, um tipo de variável, uma alteração de uma determinada propriedade e assim por diante.

O usuário deve iniciar a simulação clicando diretamente no botão "Iniciar". Dessa forma, ele começa a observar o movimento de oscilação do sistema massa-mola. Ao mesmo tempo, os gráficos de *x(t)* e *v(t)* aparecem na tela referente à Janela_para_gráfico. A Figura 49 mostra o estado do sistema no instante *"time = 12,20"* (observe a caixa em amarelo) obtido através da tecla *Print Screen*. Chamamos a atenção para as curvas correspondentes para a posição (em azul) e velocidade (em vermelho) em função do tempo.

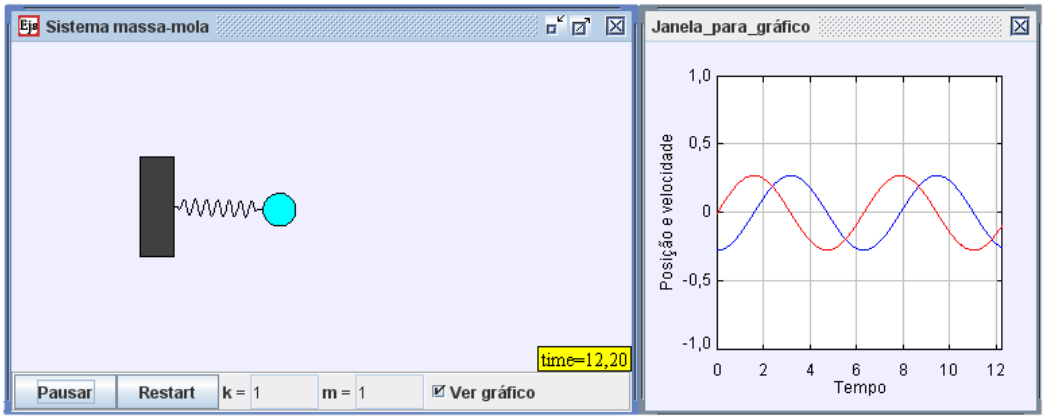


Figura 49 – Estado do Sistema massa-mola no instante 12,20 s.

Fonte: Easy Java Simulations

É importante salientar, novamente, que a aparência do desenvolvimento da simulação é fruto da ideia de quem a preparou. É possível efetuar diversas modificações nessa simulação. Por exemplo: mostrar janelas distintas para as representações gráficas, acrescentar outros elementos para a simulação apresentar a aparência de um experimento de laboratório, etc. Essa etapa, entretanto, deve ser efetuada após a aprendizagem dos recursos do EJS, isto é, como construir e estruturar a árvore de elementos, o que faz cada elemento, quais as propriedades associadas a cada um dos elementos e, finalmente, como efetuar as modificações nessas propriedades para se obter o resultado desejado. Isso significa que a elaboração da simulação, desde o planejamento até a sua execução e verificação se tal simulação é adequada, exige muitos conhecimentos da própria ferramenta EJS e tempo para prepará-la.

6 GRÁFICOS

No capítulo anterior apresentamos alguns recursos de EJS como: criar uma simulação “elementar”, inserir elementos para a visualização da evolução da simulação e inserir elementos relacionados à construção de gráficos.

Muitos dos conteúdos desenvolvidos no currículo de Física do Ensino Médio exigem a confecção de gráficos. Sem dúvida, o EJS permite ao professor elaborar aulas de determinados assuntos com a apresentação de gráficos. Dessa forma, o professor poderá explorar vários aspectos do fenômeno estudado, apresentado aos alunos os gráficos preparados com o EJS. A vantagem é que durante a apresentação do fenômeno, alguns parâmetros poderão ser alterados e os alunos imediatamente visualizarão os efeitos decorrentes dessas alterações. A partir dessa apresentação, o professor poderá propor aos alunos atividades que explorem diversos aspectos como: problemas com coletas de dados, apresentação de problemas que levem ao conflito cognitivo, elaboração e preparação de gráficos no papel e comparação com aqueles preparados pelo professor com o uso de EJS, comparação de resultados, etc.

Um exemplo de aplicação desse recurso de EJS está relacionado ao tópico "Queda Livre”. Em geral, nos livros de Ensino Médio é feita a apresentação desse tópico desprezando-se a resistência do ar. Muitos alunos têm a impressão de que o assunto “Queda Livre” é apenas um caso particular do estudo do “Movimento Retilíneo Uniformemente Variado”. Entretanto, o professor pode aproveitar o tema e apresentá-lo de forma a incluir o efeito da resistência do ar durante a queda do corpo, testar hipóteses sobre viscosidade do meio, forma geométrica do corpo que se desloca no meio, massa do corpo em queda, etc.

Utilizando-se os recursos do EJS para a confecção de gráficos, o professor poderá apresentar, por exemplo, os gráficos de velocidade e posição do corpo em função do tempo para o caso em que se despreza a resistência do ar e para aquele que se considera o atrito viscoso. Ao mesmo tempo, poderá alterar alguns dos parâmetros apresentados no parágrafo anterior. Dessa forma, o professor poderá confrontar os resultados obtidos e os alunos poderão visualizá-los imediatamente.

A Figura 50 mostra o estado inicial ($t = 0\,s$) do “Painel de controle” os gráficos da velocidade e da posição para o caso do corpo em queda e na ausência da resistência do meio viscoso. Observe os valores das variáveis *y* (altura em que o corpo é abandonado em relação ao solo), *m* (massa do corpo) e *b* (coeficiente de atrito viscoso) no “Painel de controle”.

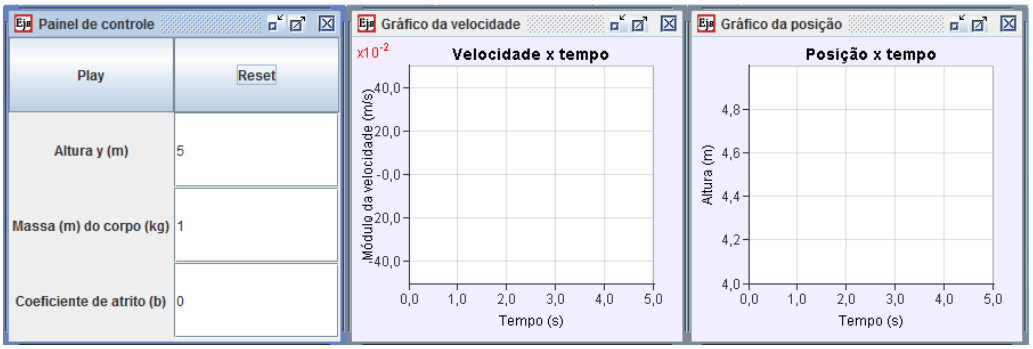


Figura 50 – Estado inicial da simulação de queda livre do corpo.

Fonte: Easy Java Simulations

Quando o usuário clicar no botão *"Play"*, a evolução do fenômeno se inicia e ele poderá visualizar a confecção dos gráficos.

A Figura 51 mostra o estado final da simulação. Observe que o professor (ou o aluno) poderá repetir o processo e verificar o efeito através da alteração dos valores da altura e da massa, mantendo-se o valor do coeficiente de atrito igual a zero.

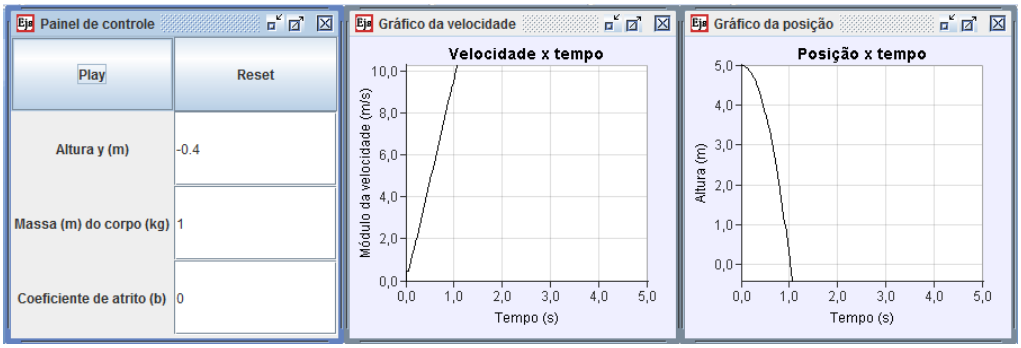


Figura 51 – Estado final da simulação de queda livre do corpo.

Fonte: Easy Java Simulations

A Figura 52, na próxima página, mostra o resultado final para o caso em que se considera o atrito viscoso ($b = 0{,}8\,\text{kg.s}^{-1}$). Chamamos a atenção do leitor para o valor numérico do coeficiente de atrito viscoso *b*. Ao criar uma simulação, o professor deverá se preocupar com o limite de validade da lei utilizada e se os valores inseridos na simulação são compatíveis com aqueles que já se encontram em tabelas. Para o exemplo apresentado, o valor numérico 0,8 foi inserido arbitrariamente. A nossa preocupação foi a de verificar se o programa e o efeito produzido funcionavam ou não! Essa é uma das vantagens do uso do EJS. O professor poderá refinar a simulação, incluir parâmetros, verificar o domínio de validade,

conferir resultados, verificar a consistência dos gráficos resultantes da simulação e analisar se a evolução dos elementos relacionados com a animação é compatível com o experimento real efetuado em laboratório.

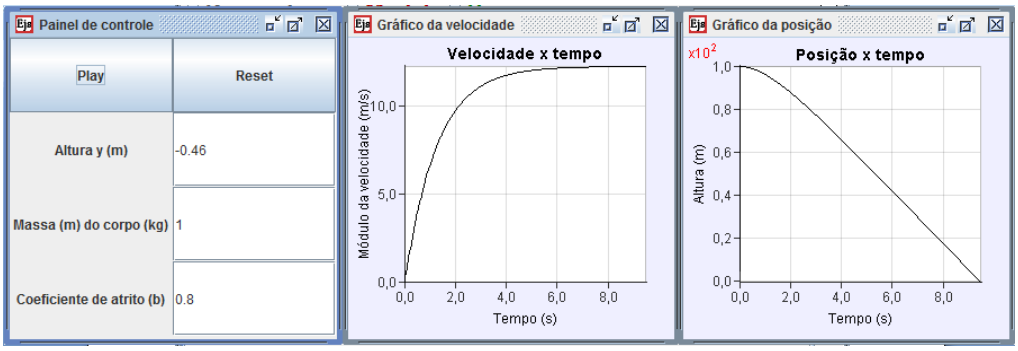


Figura 52 – Gráficos da simulação de queda do corpo no meio resistivo (estado final).

Fonte: Easy Java Simulations

O usuário (professor ou aluno) poderá repetir a simulação através da modificação das variáveis e observar os efeitos decorrentes. Estudos dirigidos, elaborados pelo professor, poderão ajudar os alunos a identificar as relações de causa e efeito e, ao mesmo tempo, tornar "mais concretos" alguns conceitos abstratos. Toda essa atividade poderá auxiliar o estudante a aprender sobre o mundo real, visualizando e interagindo com modelos científicos.

O EJS permite utilizar recursos em duas dimensões (2D) e em três dimensões (3D). Com esses recursos, o professor poderá explorar e diferenciar suas atividades em sala de aula.

Normalmente, nos cursos de Física do Ensino Médio são apresentados movimentos de partículas em uma dimensão. Entretanto, com o uso do EJS é possível apresentar e explorar os movimentos em 2D, em 3D e, ainda, modificar o ângulo de visualização da evolução de um determinado fenômeno. Por exemplo, é possível explorar o movimento de uma partícula eletrizada lançada em um campo magnético. Em tal simulação, o professor poderá alterar o ângulo de lançamento da partícula no interior do campo magnético e verificar as diferentes trajetórias descritas pela partícula. Claro que simulações mais elaboradas exigirão do usuário um grau de conhecimento mais aprofundado da ferramenta EJS.

A Figura 53 mostra a trajetória em 3D descrita por uma partícula. Para essa simulação em particular, foram inseridas as equações paramétricas $x(t)$, $y(t)$ e $z(t)$, ou seja, a simulação se resume a um problema simples de cinemática. Entretanto, é possível preparar

simulações em função das forças que atuam numa partícula e verificar o efeito produzido no seu movimento.

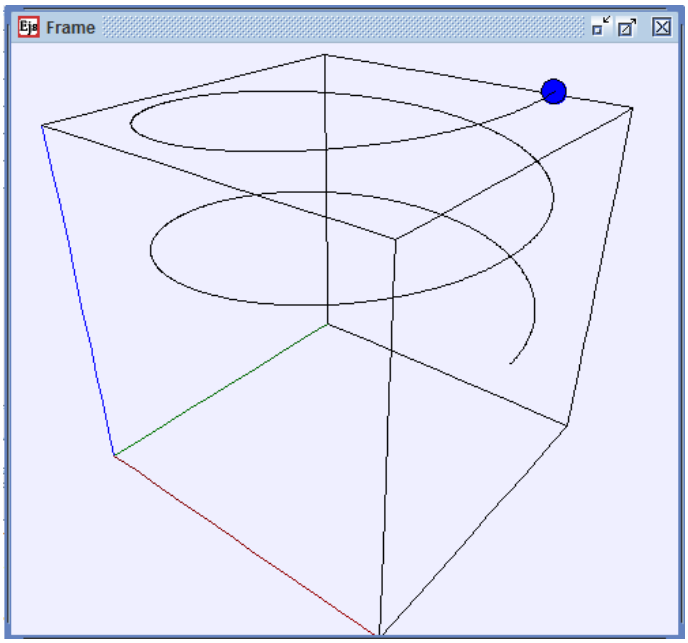


Figura 53 – Trajetória helicoidal descrita pela partícula, conhecidas as equações paramétricas do movimento.

Fonte: Easy Java Simulations

A Figura 54 mostra a mesma trajetória segundo outro ângulo de visualização.

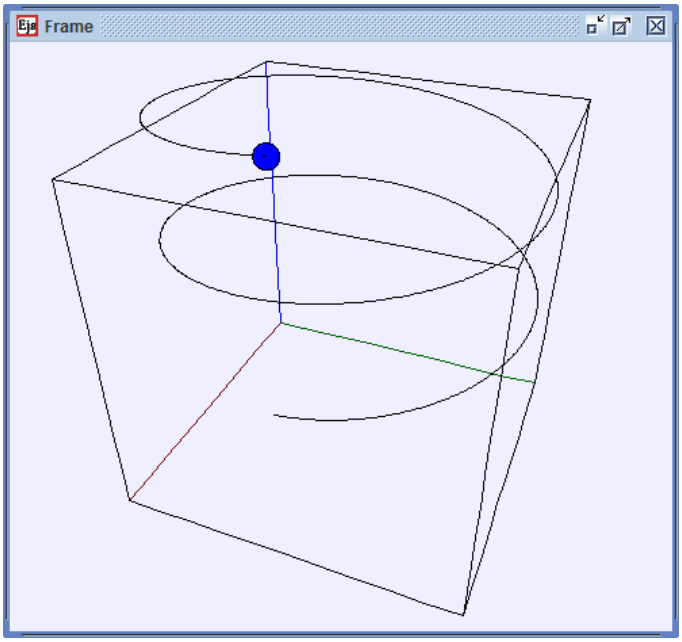


Figura 54 – Trajetória helicoidal da partícula segundo outro ângulo de visualização.

Fonte: Easy Java Simulations

Salientamos que a necessidade de se conhecer “alguns rudimentos de linguagem de programação” é uma condição para se planejar simulações com maior grau de complexidade e, dessa forma, obter uma resposta mais próxima da realidade. Assim, o EJS é uma ferramenta que pode ser considerada difícil de ser aprendida imediatamente. Torna-se, então, necessário prover o professor (aluno, usuário) com informações planejadas e orientações para que ele, sozinho, consiga planejar simulações em função da sua necessidade encontrada em sala de aula.

Nesse ponto, pode-se questionar: “Por que não utilizar uma ferramenta mais fácil de ser manipulada?”, “Por que não utilizar as simulações já prontas e disponíveis na *Internet*?”

Existem outras ferramentas disponíveis e que cumprem a função de modelagem. Por exemplo, o *Modellus* (http://modellus.fct.unl.pt/). Essa ferramenta dispensa o conhecimento de uma linguagem de programação e a sua sintaxe de escrita, é praticamente igual àquela utilizada na escrita matemática convencional. *Modellus* foi concebido como um *software* de modelagem e a ideia básica de seu projeto é a de permitir ao aluno se preocupar mais com o significado do modelo desenvolvido do que com as equações matemáticas. Assim, alunos e professores poderão realizar experiências com modelos matemáticos e controlar variáveis de tempo, distância, velocidade, analisar variação de uma função, preparar animações, resolver e criar exercícios. Aparentemente, tudo que o EJS permite fazer.

Um dos aspectos que distingue o EJS de outras ferramentas está na possibilidade de se efetuar simulações e representações em 3D, como aquelas mostradas nas Figuras 53 e 54 da página anterior. Outro aspecto relaciona-se com a possibilidade de se executar a simulação como um *applet*, utilizando para isso um navegador *web*, e, ainda, executá-la como uma aplicação *Java* independente.

O EJS não funciona como uma “caixa preta” e a sua utilização se justifica perante as simulações prontas e disponíveis na *Internet*. É interessante observar que muitas das simulações disponibilizadas não apresentam valor pedagógico e, muitas vezes, aquelas consideradas boas não atendem às expectativas do professor. Assim, como já mencionamos ao longo dessa dissertação, uma simulação desenvolvida com o EJS deixa de ser uma “caixa preta” e o professor (ou aluno, etc.) poderá efetuar as alterações e os ajustes necessários para melhorar os aspectos gráficos, computacionais ou pedagógicos da simulação.

7 O LABORATÓRIO VIRTUAL E CONSIDERAÇÕES FINAIS

Nessa dissertação, preocupamo-nos em demonstrar a necessidade de modificar o tipo de aula ministrada pelo professor de Física do Ensino Médio. Uma possibilidade é utilizar a ferramenta EJS. Entretanto, espera-se que ao longo da dissertação tenha ficado claro que essa ferramenta não é fácil de dominar em um intervalo de tempo pequeno. O professor precisará ter uma orientação adequada para que ele possa, enfim, planejar, elaborar ações e executar o tipo de simulação que ele tem em mente.

O professor, ao participar do ato de criação de uma simulação, estará adquirindo uma melhor compreensão do assunto a ser ensinado, pois ele mesmo irá se deparar com conflitos cognitivos. Sem dúvida, conhecendo os próprios conflitos cognitivos, isso irá permitir-lhe planejamentos de aulas mais elaboradas e, ao mesmo tempo, poderá propor atividades para os seus alunos compreenderem de forma diferenciada o assunto a ser estudado.

Como mencionado no Capítulo 2.2, ao se utilizar simulações no Ensino de Física para alunos do Ensino Médio, o professor estaria desenvolvendo uma atividade mais próxima da realidade da produção científica e tecnológica, rompendo com o ensino tradicional. Para isso, o professor deverá mudar suas concepções e ideias pré-concebidas. Perceber que somente com atividades colaborativas, interdisciplinares, ele poderá obter resultados mais valiosos para o Ensino de Física.

Dentro da proposta de se utilizar simulações no ensino, um aspecto crucial ao se desenvolver o laboratório virtual é o de permitir a interatividade. Esse é, ainda, um recurso pouco explorado como atividade pedagógica. É justamente a interatividade que permitirá o estudante controlar, alterar, modificar parâmetros e verificar os efeitos decorrentes de suas ações. O estudante (aluno) terá a oportunidade de obter respostas quantitativas e qualitativas proporcionadas pelo sistema simulado.

Mas, uma interatividade pedagogicamente rica vai além da simples capacidade de alterar valores de parâmetros, “rodar” algoritmos, analisar e comparar respostas ou, ainda, repetir um procedimento. A interatividade pedagogicamente valiosa deve permitir tudo isso e, também, proporcionar a visualização da evolução do sistema físico sob diferentes aspectos sistema ou mostrar a resposta, em tempo real, quando o usuário alterar um determinado parâmetro que governa a evolução do sistema que está sendo simulado. É justamente essa resposta imediata que iria ajudar o estudante a adquirir melhor entendimento e visão mais crítica das leis que governam o fenômeno físico simulado.

A combinação entre interatividade, visualização e gráficos fazem do EJS uma ferramenta especial que pode contribuir ao Ensino da Física. Ela merece ser estudada, mas, ao mesmo tempo, exige dedicação e tempo de estudo para o professor dominar uma parcela de sua potencialidade. Acreditamos que o professor deverá ver nesse estudo e dedicação, um investimento próprio e uma motivação para mobilizar outros professores a participarem de projetos de construção de laboratório virtual. EJS permite a formação de equipes interdisciplinares de professores para a elaboração de tais projetos.

Para finalizar, o trabalho desenvolvido nessa dissertação está longe de estar concluído. Esse trabalho mostra a possibilidade de atividades diferenciadas e com várias aplicações sob o ponto de vista pedagógico. Prosseguiremos na elaboração e atualização do texto sobre a ferramenta EJS, apresentado como produto dessa dissertação, na sua divulgação (via página do PPGECE-UFSCar) e no incentivo à formação de equipes de professores que possam estar engajados no planejamento de atividades de simulação com real valor pedagógico.

8 REFERÊNCIAS BIBLIOGRÁFICAS

BRASIL. Ministério da Educação. Secretaria de Educação Média e Tecnológica. **Parâmetros curriculares nacionais:** ensino médio**.** Brasília: Ministério da Educação, 1999. 364p.

BRISCOE, C. The dynamic interactions among beliefs, roles, methaphores and teaching practices: a case study of teacher change. **Science Education**, v. 75, n.2, p. 185-99, 1991.

CARVALHO, A. M. P.; GIL-PÉREZ, D. **Formação de professores de ciências**: tendências e inovações. 2. ed. São Paulo: Cortez, 1995. 120 p.

CHAVES, A.; SAMPAIO, J. F. **Física básica:** mecânica**.** Rio de Janeiro: LTC, 2007. v. 1.

CHRISTIAN, W.; ESQUEMBRE, F. Modeling physics with easy java simulations. **The Physics Teacher**, v. 45, p. 475-480, 2007.

CRONIN-JONES, L. L. Science teaching beliefs and their influence on curriculum implementation: two case studies. **Journal of Research in Science Teaching**, v. 38, n.3, p. 235-250, 1991.

DEMO, P. **Desafios modernos da educação.** 8. ed. Petrópolis, R.J: Vozes, 1999. 272 p.

DUMAS-CARRÉ, A.; FURIÓ, C; GARRET, R. Formación inicial del profesorado de ciencias en Francia, Inglaterra y Gales y España: análisis de la organización de los estudios y nuevas tendencias**. Enseñanza de las Ciencias**, v. 8, n. 3, p. 274-281, 1990.

ESQUEMBRE, F. **Creación de simulaciones interactivas en java.** Madrid: Pearson Educación, S. A., 2005. 352 p.

ESQUEMBRE, F. **Easy Java Simulations.** Disponível em: <http://www.um.es/fem/Ejs/>. Acesso em: 10 mar. 2011.

FIGUEIRA, J. S. Easy java simulations: modelagem computacional para o ensino de física. **Revista Brasileira de Ensino de Física**, v. 27, n. 4, p. 613-618, 2005.

GIL-PERÉZ, D. Qué han de saber y saber hacer los profesores de ciencias? **Enseñanza de las ciencias**, v. 9, n. 1, p. 69-77, 1991.

HAWKING, S. **Os gênios da ciência:** sobre os ombros de gigantes: as mais importantes ideias e descobertas da física e da astronomia. 2. ed. Rio de Janeiro: Elsevier, 2005. 1056 p.

MACHADO, K. D. **Equações diferenciais aplicadas à física.** 3. ed. Ponta Grossa: Editora UEPG, 2004. 598 p.

MEDEIROS, A.; MEDEIROS, C. F. Possibilidades e limitações das simulações computacionais no ensino de física. **Revista Brasileira de Ensino de Física**, v. 24, n. 2, p. 77-86, 2002.

ROBILOTA, R. M.; BABICHAK, C. C. Definições e conceitos em física. **Cadernos Cedes**, ano XVIII, n. 41, p. 35-45, jul. 1997.

VEIT, E. A.; TEODORO, V. D. Modelagem no ensino/aprendizagem de física e os novos parâmetros curriculares nacionais para o ensino médio. **Revista Brasileira de Ensino de Física**, v. 24, n. 2, p. 87-96, 2002.

ZILL, D. G.; CULLEN, M.R. **Equações diferenciais.** São Paulo: Pearson Makron Books, 2001. v. 1.

ANEXO

PROCEDIMENTOS BÁSICOS PARA SE CRIAR UMA SIMULAÇÃO DE FÍSICA COM O *EASY JAVA SIMULATIONS*

SUMÁRIO

1 CONCEITOS BÁSICOS DE MODELAGEM E SIMULAÇÃO

O objetivo deste trabalho é apresentar a ferramenta de simulação *Easy Java Simulations* (EJS) e mostrar ao professor do Ensino Médio como ele pode usá-la para preparar atividades diferenciadas aos seus alunos. Pela leitura do texto, fica clara a nossa preocupação em fazer uma exposição didática. Assim, mostra-se todo o procedimento básico e necessário para a construção de uma simulação, desde o modelo matemático utilizado para modelar a evolução do sistema físico em estudo, até a preparação da interface gráfica que permite a visualização dos elementos gráficos.

Iniciamos o texto com conceitos gerais sobre modelos, sistemas e o algoritmo de uma simulação. Eles são necessários para aqueles que não estão acostumados com as "técnicas de programação". A partir daí, pode-se apreciar as vantagens de se conhecer, e aprender, o EJS. Para isso, são apresentadas algumas simulações simples que podem servir como roteiro de estudo.

Alertamos, também, que o presente trabalho não esgota o tema. EJS exige tempo e dedicação! Assim, contamos com a participação de professores motivados e dispostos em aprender os seus rudimentos.

1.1 MODELOS E SISTEMAS

Um modelo pode ser uma representação de um sistema físico, por exemplo, desenvolvido com a finalidade de responder às perguntas específicas sobre o sistema estudado. Para a sua construção é necessário definir o sistema e quais são as perguntas. Um sistema, por sua vez, é qualquer objeto ou conjunto de objetos cujas propriedades desejam-se estudar. Assim, circuitos elétricos, espelhos esféricos e pêndulos simples são alguns exemplos de sistemas físicos.

Considerando-se o sistema pêndulo simples, pode-se levantar o seguinte conjunto de questões:

b) Qual é o período de oscilação do pêndulo?
c) Quais são as grandezas físicas (massa, comprimento do fio, amplitude, etc.) que interferem no período de oscilação?
d) Como o meio altera o período de oscilação do pêndulo?

Essas questões podem ser respondidas através de uma experiência, que é um processo de se obter (ou extrair) dados do sistema em estudo e sobre o qual se exerce algum tipo de influência externa. Tal procedimento é denominado de método experimental e ele

apresenta muitas vantagens, pois está alicerçado sobre sólidos fundamentos científicos. Entretanto, o método experimental apresenta limitações e alguns experimentos são difíceis, ou mesmo desaconselhável, de realizá-los. Dentre as limitações, destacam-se:

a) Elevado custo para a execução do experimento.
b) O experimento pode provocar danos àqueles que o estão realizando.
c) Tempo de realização do experimento (muito rápido ou muito lento).
d) Necessidade de se alterar as variáveis do sistema real ou, ainda, impossível de alterá-las no intervalo de tempo desejado.
e) O próprio sistema não existe fisicamente. Isso acontece quando se projeta um novo sistema e necessita-se saber, antes de construir o próprio sistema, qual será o seu comportamento.

Qualquer que seja o inconveniente apontado acima é possível, com o uso de um modelo adequado, efetuar o ensaio de um sistema em condições extremas e muitas vezes impraticável. Nesse sentido, a simulação é uma técnica que permite analisar diferentes sistemas submetidos a diferentes condições experimentais.

1.2 MODELOS MATEMÁTICOS

Os modelos são representações utilizadas pelos seres humanos para aprender e prever o comportamento de um sistema real. Os modelos podem ser mentais, verbais, físicos e matemáticos.

Os modelos mentais relacionam-se com a intuição e experimentação. Por exemplo, aprender a dirigir um automóvel consiste antecipadamente em desenvolver um modelo mental das propriedades da condução do automóvel.

Os modelos verbais são descritos através das palavras. Assim, a frase “ao pisar no acelerador do automóvel, então a sua velocidade aumenta” descreve o comportamento do sistema automóvel através do modelo verbal.

Os modelos físicos, tais como maquetes, são utilizados por engenheiros e arquitetos para testar e comprovar as propriedades estéticas, aerodinâmicas, estáticas, etc., dos sistemas que se pretende construir.

Finalmente, os modelos matemáticos utilizam relações entre quantidades (grandezas) que podem ser observadas no sistema e que são escritos através de relações matemáticas. Essas grandezas podem ser, por exemplo, a distância, a velocidade, a temperatura, etc. A maioria das leis naturais são modelos matemáticos e os cientistas

constroem modelos de uma porção da realidade para estudar as suas propriedades. Por exemplo, para o sistema resistência elétrica, a Lei de Ohm descreve a relação entre a diminuição do potencial elétrico (d.d.p.) e a intensidade da corrente elétrica do dipolo (resistor elétrico).

Em alguns casos, as relações matemáticas que descrevem um modelo são simples e podem ser resolvidos analiticamente. Outros modelos, entretanto, devem ser estudados com o auxílio de um computador que utiliza métodos numéricos. O experimento numérico realizado sobre o modelo matemático recebe o nome de simulação.

A seguir são descritas, resumidamente, as classificações mais comuns de modelos matemáticos utilizados em experimentos numéricos.

a) Modelo determinista versus modelo estocástico: No modelo determinista, todas as variáveis são conhecidas em cada instante, enquanto que no modelo estocástico alguma variável de entrada é aleatória. As variáveis do modelo calculadas a partir das variáveis aleatórias são, também, variáveis aleatórias.
b) Modelo estático versus modelo dinâmico: O modelo estático é uma representação do sistema em dado instante particular. Esse modelo serve para representar um sistema no qual a grandeza tempo não participa. O modelo dinâmico representa um sistema que evolui com o tempo.
c) Modelo de tempo contínuo versus modelo de tempo discreto: No modelo de tempo contínuo os seus valores de estado mudam continuamente, infinitas vezes, em um intervalo de tempo específico. Por exemplo, o nível de água em um reservatório. No modelo de tempo discreto, entretanto, os seus valores de estado se alteram somente em determinados instantes, isto é, suas variáveis de estado mudam de valor um número finito de vezes por unidade de tempo. A decisão de se realizar um modelo de tempo contínuo ou discreto depende do objetivo específico e do estudo do tipo de sistema.

Finalmente, é importante observar que qualquer modelo matemático está fundamentado em um conjunto de hipóteses. Quando as condições experimentais são tais que não satisfazem às hipóteses do modelo, este deixa de ser válido. Para evitar esse tipo de problema, a descrição do modelo deve ser acompanhada de uma documentação que indique o conjunto de experimentos válidos para tal modelo.

1.3 SIMULAÇÃO DE MODELOS DE TEMPO CONTÍNUO

O objetivo desse capítulo é o de explicar alguns aspectos fundamentais dos modelos de tempo contínuo e que servem de base para a compreensão, planejamento e execução de uma simulação interativa que utilize o EJS.

1.3.1 EQUAÇÕES E VARIÁVEIS

Equações são relações entre as grandezas físicas de um sistema e essas, por sua vez, são os elementos constituintes dos modelos matemáticos. As grandezas físicas são chamadas simplesmente de variáveis.

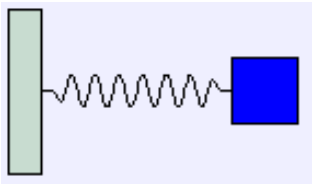


Figura 1 – O sistema massa-mola.

Considere, por exemplo, o sistema massa-mola (Figura 1). A massa do corpo é representada por *m*, o comprimento da mola em seu estado natural (relaxado) por *L* e a constante elástica da mola por *k*. Considera-se, em princípio, uma mola ideal.

Ao se deslocar o corpo de sua posição de equilíbrio, distendendo-se ou comprimindo-se a mola, este fica sujeito a uma força elástica que, para os nossos propósitos, obedece à lei de Hooke. Assim, ao se abandonar o corpo, este passa a oscilar em torno da posição de equilíbrio.

Desprezando-se os efeitos dissipativos, o sistema oscilará "eternamente" com período de oscilação constante. Para um sistema real, entretanto, a amplitude de movimento decresce com o tempo. Neste caso, deve-se considerar a força de amortecimento que pode ser devido à interação do corpo com o ar, ou à interação entre as superfícies de contato (superfície do corpo e superfície de apoio), ou, ainda, a ambos. Para o propósito desse capítulo, considera-se uma força de amortecimento de módulo proporcional ao módulo da velocidade. Assim, as forças que atuam no corpo em movimento são:

a) Força elástica: $\vec{F}_e = -k \cdot (x - L) \cdot \hat{i}$

b) Força de amortecimento: $\vec{F}_a = -b \cdot v \cdot \hat{i}$

Pela Segunda Lei de Newton, a posição do corpo em função do tempo é dada por:

$$\vec{F}_R = \vec{F}_e + \vec{F}_a \Rightarrow F_R = -k \cdot (x - L) - b \cdot v$$

A posição e a velocidade do corpo são calculadas em função da derivada em relação ao tempo, isto é, considerando-se que a derivada da posição em relação ao tempo é a

velocidade e que a derivada da velocidade em relação ao tempo é a aceleração do corpo. Logo, o modelo matemático para o sistema massa-mola é, também, constituído pelas equações:

$$F_R = m \cdot a$$

$$\frac{dx}{dt} = v$$

$$\frac{dv}{dt} = a \, .$$

Assim, o modelo matemático que descreve a relação entre as grandezas físicas relevantes do sistema é constituído por quatro equações (mostradas acima) e as grandezas físicas que aparecem nessas equações são as variáveis do modelo. Essas variáveis, no entanto, podem ser classificadas em parâmetros, variáveis de estado e variáveis algébricas.

Os parâmetros são as variáveis cujos valores permanecem constantes durante a simulação; as variáveis de estado são aquelas que aparecem derivadas em relação ao tempo; as variáveis algébricas são as demais variáveis que aparecem no modelo, isto é, são aquelas que não são constantes e nem aparecem derivadas em relação ao tempo. Assim, para o modelo do sistema massa-mola tem-se:

a) Parâmetros: m, L e k.
b) Variáveis de estado: x e v.
c) Variáveis algébricas: F_R e a.

1.3.2 ALGORITMO PARA A SIMULAÇÃO DE MODELOS DE TEMPO CONTÍNUO

Uma simulação computacional é um experimento numérico realizado com o uso de um modelo matemático. Tal experimento deve ser escrito em uma linguagem de programação e o conceito central da programação é o algoritmo. Programar é basicamente elaborar e escrever um algoritmo. Assim, de forma resumida, um algoritmo é a descrição do comportamento de um sistema (físico, químico, etc.) expresso em função de um número finito de ações bem definidas e com o pressuposto de que essas ações possam ser executadas.

A Figura 2, a seguir, mostra um algoritmo, em forma de fluxograma, para simular modelos de tempo contínuo.

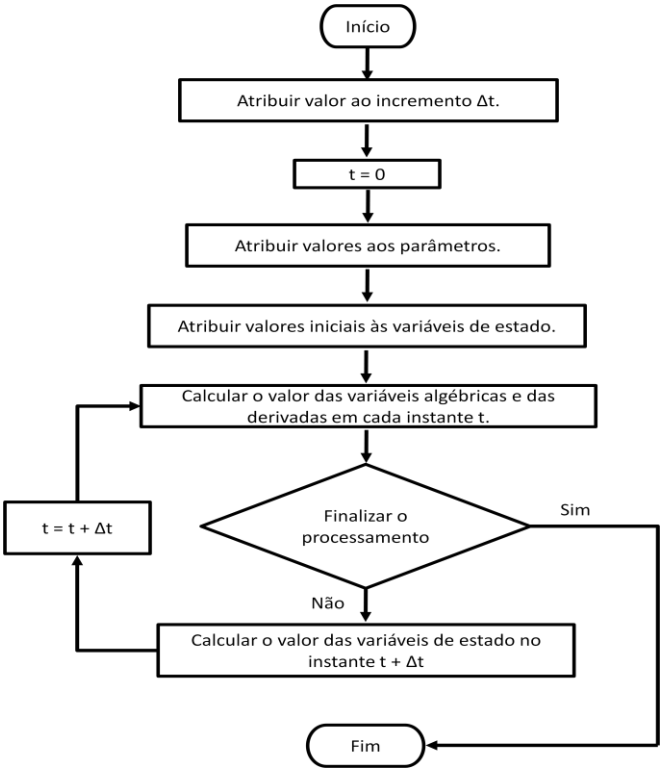


Figura 2 – Fluxograma para modelo de tempo contínuo.

A simulação tem início ao se atribuir os valores de Δt (incremento do tempo), t (tempo) e os valores dos parâmetros. Os valores dos parâmetros permanecem constantes durante o processamento.

As variáveis de estado são calculadas a partir da integração numérica de suas derivadas. Existem métodos específicos para se efetuar a integração, como, por exemplo, o método de Euler e o método de Runge-Kutta.

Os valores das variáveis de estado são calculados em cada instante de tempo a partir das variáveis de estado no instante inicial ($t = 0$) e dos valores dos parâmetros.

A condição de finalização da simulação depende do tipo de experimento numérico que se realiza. Por exemplo, a simulação pode finalizar quanto se alcançar um valor de t especificado ou, ainda, que uma determinada variável satisfaça uma condição imposta pelo usuário.

Finalmente, torna-se necessário salientar que o tamanho do passo de integração (incremento Δt) deve ser escolhido de forma a alcançar o compromisso entre a precisão e a carga computacional. Quanto menor o valor de Δt, menor será o erro cometido no cálculo das variáveis de estado durante a execução da simulação, porém, o tempo de execução será maior.

2 CONCEITOS BÁSICOS PARA SE DESENVOLVER UMA SIMULAÇÃO COM O USO EASY JAVA SIMULATIONS (EJS)

O EJS foi criado e desenvolvido pelo Prof. Francisco Esquembre com objetivos pedagógicos, isto é, o EJS é uma ferramenta que permite aos professores e alunos criarem de forma "simples" os seus laboratórios virtuais, sem que eles conheçam as técnicas sofisticadas de programação. Esse é o aspecto que torna EJS uma ferramenta especial! É possível encontrar muitas simulações na *internet* e poucas permitem visualizar os seus "segredos", isto é, a forma como foi efetuada a programação. Por outro lado, aquelas que permitem a sua visualização não proporcionam o código fonte, resultando ser acessível somente àqueles que possuem o conhecimento das técnicas de programação e de uma determinada linguagem de programação.

A documentação de EJS, os casos para estudo, *download* do programa (gratuito), etc., podem ser encontrados no site http://fem.um.es/Ejs.

Basicamente, EJS está desenhado para permitir a interação entre o usuário e o modelo desenvolvido durante o processo de simulação.

A descrição de como EJS deve realizar os cálculos das variáveis de estado, das variáveis algébricas ou, ainda, a evolução do processo deve ser feito através de fragmentos de códigos escritos na linguagem de programação *Java*. Entretanto, como foi dito no início desse item, não é necessário conhecer as técnicas de programação em geral e em *Java*, em particular.

Nos próximos capítulos, o leitor encontrará os procedimentos básicos para se criar uma simulação com o EJS. Alertamos, entretanto, que a apresentação está longe de ser um tutorial de EJS. O nosso objetivo é apresentar algumas técnicas e procedimentos que permitam ao professor (ou pessoas interessadas) começar a planejar e aplicar experiências de Física via simulações. Esse breve guia de procedimentos estará sendo modificado à medida que dominarmos as técnicas para a elaboração de simulações mais sofisticadas. Também serão acrescentadas as tabelas relacionadas às denominadas propriedades dos elementos gráficos.

2.1 "EXECUTANDO" O EJS

Realizado o *download* do programa EJS, é possível deixar o ícone console do EJS visível no *desktop* do *Windows*. Para isso, o usuário dever abrir o diretório onde foi

instalado o programa, clicar sobre o ícone do console EJS e arrastá-lo até o *desktop*. A aparência do ícone é mostrada na Figura 3.

Figura 3 – Ícone de atalho para EJS.

Para iniciar o programa, o usuário deve clicar nesse ícone e, assim feito, aparecerá na tela do monitor do computador uma caixa mostrando o progresso de abertura do programa, como indica a Figura 4.

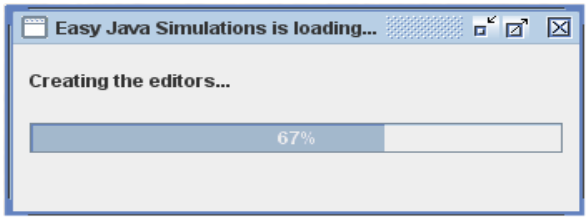


Figura 4 – Caixa de progresso de abertura do programa

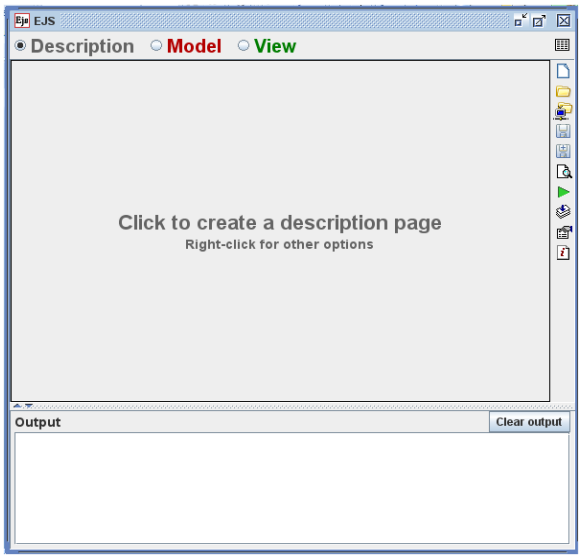


Figura 5 – Janela de interface do EJS com a opção *Description* ativada.

Completada a fase inicial, o usuário vê na tela do monitor a janela da interface do EJS, mostrada na Figura 5. No topo da janela aparecem os botões *Description*, *Model* e *View* correspondendo às três partes da simulação que será criada com o EJS. Cada botão tem uma cor característica e uma vez selecionada uma opção, aparecerá um círculo ao lado da palavra. Por exemplo, ao selecionar e ativar o botão *Description*, o círculo ao lado da palavra aparece preenchido, como mostrado na Figura 5. Abaixo dessa barra de botões existe uma área vazia com a mensagem convidativa *"Click to create a description page"* (Clique para criar uma página de descrição). Na parte inferior da janela está uma área de mensagens (*Output*) onde o EJS mostrará as mensagens correspondentes das ações realizadas durante o processo de compilação.

Ao selecionar o botão *Model*, a janela de visualização correspondente terá a aparência da Figura 6. Pode-se, então, observar um novo conjunto de botões que permitirá, uma vez selecionado um de cada vez, o usuário criar o modelo da simulação. Observa-se abaixo desse conjunto de botões a mensagem convidativa *"Click to create a page of variables"* (Clique para criar uma página de variáveis). Para cada botão selecionado, a janela terá uma aparência e mensagem correspondente à sua função. Isso será especificado no decorrer desse guia de procedimentos.

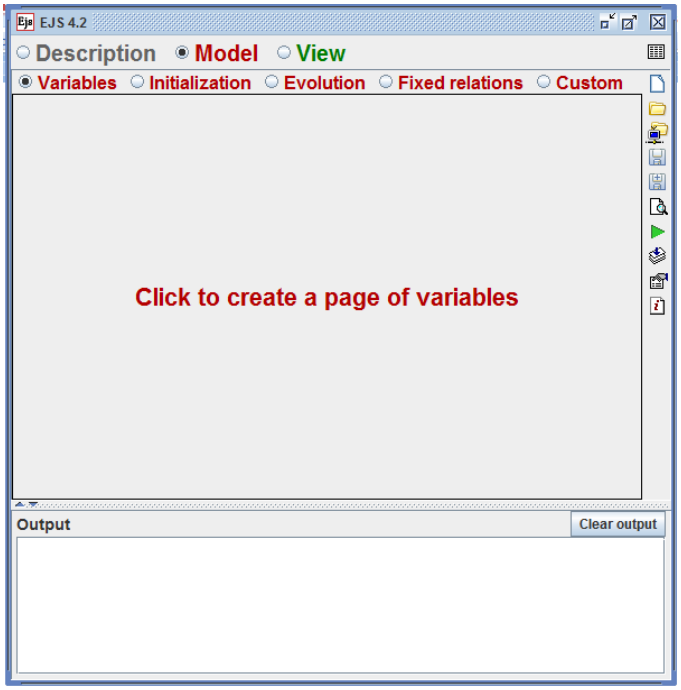


Figura 6 – Janela de interface do EJS com a opção *Model-Variables* ativada.

Ao ativar a opção *View*, uma janela com uma estrutura correspondente será mostrada. Nessa janela serão distribuídos os elementos responsáveis pela aparência da simulação. Como distribuir os elementos na janela de visualização da simulação, como ligá-los às variáveis do modelo, como criar gráficos, etc. será mostrado no item correspondente à descrição de *View*.

O leitor deve ter observado que na parte direita da janela de interface do EJS existe uma série de ícones. São ícones que podem ser facilmente reconhecidos, por serem usuais em aplicativos do *Windows*. Na tabela 1, a seguir, estão indicados os seus nomes e funções.

Salientamos que na pasta “abrir”, existem vários exemplos, disponíveis com a versão do programa, desenvolvidos por diferentes pesquisadores e que poderão ser usados como guia de estudo. Esses exemplos mostram como desenvolver a estrutura de uma simulação e, ao mesmo tempo, permitem ao usuário observar a sua evolução com o tempo durante a sua execução. Entretanto, a maioria desses exemplos é sofisticada e exigem um grau de conhecimento avançado sobre as técnicas de programação. Outras simulações, ainda, são muito específicas de uma área da tecnologia. Assim, o leitor poderá perder muito tempo em decifrá-las. Aconselhamos ao leitor seguir os “passos” aqui apresentados e aprender inicialmente a construção de uma simulação. Dominadas essas tarefas iniciais, então, o leitor poderá estudar as simulações disponíveis nessa pasta.

ÍCONE	NOME	FUNÇÃO
	Novo	Criar uma nova simulação
	Abrir	Carregar uma simulação preexistente
	Ler	Ler a partir de uma biblioteca virtual EJS
	Salvar	Salvar e guardar a simulação atual em um disco ou arquivo já existente
	Salvar como	Salvar e guardar a simulação atual num arquivo de nome diferente
	Procurar	Procurar um nome (*string*)
	Executar	Gerar e executar a simulação em uso
	Empilhar	Empilhar a simulação em uso
	Opções	Modificar algumas opções sobre a aparência e comportamento do EJS
	Informação	Mostra a informação relativa ao EJS.

TABELA 1 – Ícones e suas funções

2.2 A ESTRUTURA DE UMA SIMULAÇÃO

A metodologia usada na criação de laboratório virtual do EJS está baseada numa simplificação do paradigma *model-control-view*, descrito a seguir.

a. *Model* (Modelo): O modelo descreve o fenômeno a ser estudado em termos das variáveis (grandezas que figuram no fenômeno) e das relações matemáticas existentes entre essas variáveis. Tais relações são expressas por algoritmos de computadores.

b. *Control* (Controle): Define as ações que o usuário pode alterar durante a simulação.

c. *View* (Visão ou Janela de visualização): Mostra a representação dos diferentes estados que o fenômeno pode assumir.

Essas três partes estão inter-relacionadas. O Modelo afeta a Visão, pois esta deve mostrar a evolução dos valores das variáveis de estado. O Controle altera o Modelo, pois as ações exercidas pelo usuário podem modificar os valores das variáveis do Modelo. E, finalmente, a Visão afeta o Modelo e o Controle, pois a interface gráfica pode conter elementos que permitem o usuário alterar os valores de determinadas grandezas e, ainda, determinadas ações.

Entretanto, EJS apresenta uma simplificação desse paradigma através da supressão da parte relativa ao Controle e integrando as suas funções na Visão (via janela de visualização) e no modelo. Dessa forma, o usuário pode interagir com a simulação, através da interface gráfica, utilizando o *mouse* ou o teclado do computador. Para essa tarefa ocorrer adequadamente, ao se programar o modelo é necessário especificar de que forma as ações realizadas pelo usuário, durante a simulação, sobre os componentes de controle da janela de visualização ou dos componentes gráficos, afetarão os valores das variáveis de estado do modelo. Esse assunto será analisado na parte relativa à "construção" da janela de visualização.

O painel mostrado na Figura 5, p. 87, resume a simplificação do paradigma apontado no parágrafo anterior. Assim, a definição de um laboratório virtual com o uso do EJS se estrutura nas seguintes partes:

a. *Description* (Descrição): Nessa página é possível incluir roteiros, resumos, descrição de experimento, questionários, etc.
b. *Model* (Modelo): Modelo dinâmico cuja simulação via interatividade é a base do laboratório virtual.
c. *View* (Visão): Interface entre o usuário e o modelo que proporciona a representação visual do comportamento dinâmico do modelo e os mecanismos para o usuário poder interagir com o modelo durante a simulação.

2.3 O PAINEL *DESCRIPTION*

Uma vez selecionada a opção *Description* é possível incluir uma descrição textual de como operar a simulação. Pode-se, também, inserir páginas (quantas forem necessárias) para se apresentar resumos teóricos, fornecer listas de exercícios, questionários e atividades aos alunos.

Para inserir uma página nesse painel, o usuário deve clicar como o botão direito do *mouse* (ou esquerdo, dependendo da configuração usada no painel de controle do *Windows*) na parte central onde se lê *"Click to create a description page"*. Feito isso, uma caixa de diálogo aparece no painel e o usuário deverá, no campo *"Intro Page"*, digitar um nome em função do que se deseja criar (Figura 7). Por exemplo: "Resumo teórico", "Roteiro", "Lista de exercícios", etc.

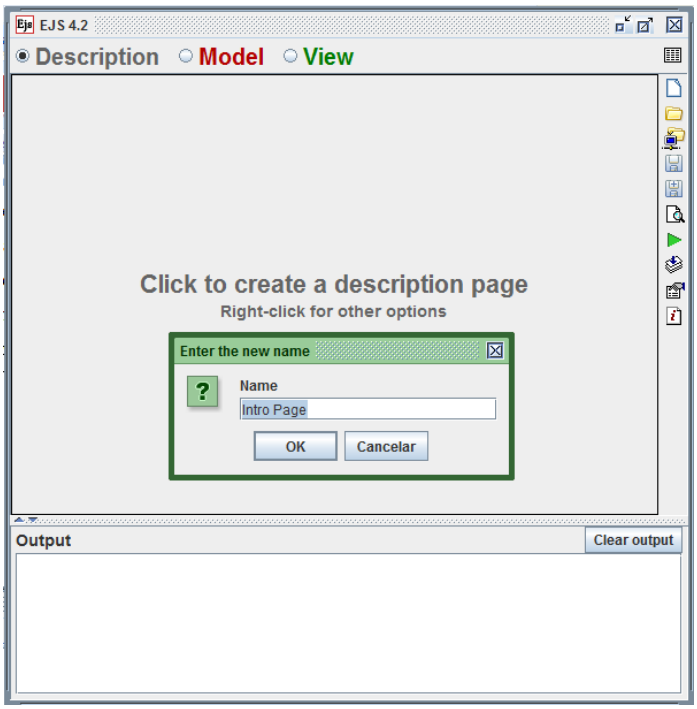


Figura 7 – Painel *Description* com a caixa para inserir um nome de página

Após digitar o nome da página e clicar em *OK*, observa-se o aparecimento da janela onde o usuário editará o texto em mente. Para isso, basta clicar na região central da janela e escrever (Figura 8). Observa-se que aparece uma barra com funções e ícones, da mesma forma que uma página do editor do *Word*. Essas funções são facilmente reconhecidas e, ao colocar a seta do mouse sobre um ícone, aparece uma pequena caixa que representa a sua função.

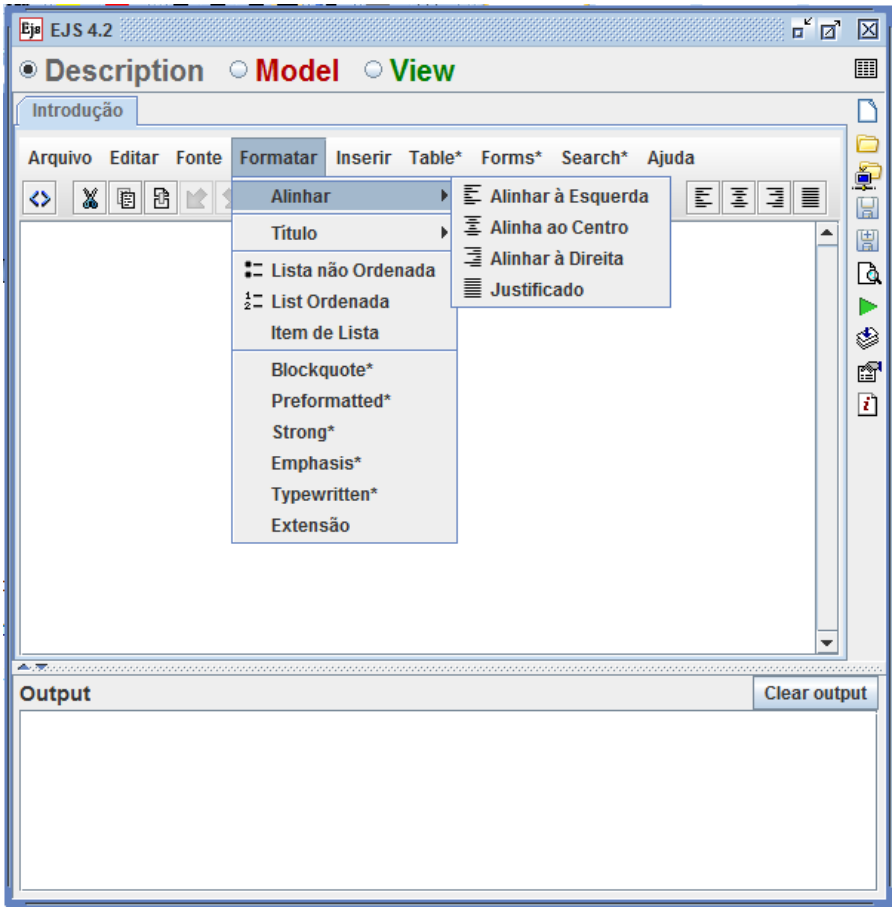


Figura 8 – Edição de uma página do painel *Description*.

É possível inserir, nessa página, figuras, fórmulas e tabelas. Se preferir, o usuário pode escrever o texto em outro editor de texto, como o *Word*, e importá-lo para a página que está sendo desenvolvida. As figuras devem ser salvas no formato *gif* no diretório onde as simulações estão sendo guardadas. As fórmulas podem ser editadas via *Equation* do *Word* ou, ainda, com o uso do *MathType*. Feito isso, deve salvar a equação como uma figura, também no formato *gif* e colocá-la do diretório da simulação em desenvolvimento. Finalizados esses procedimentos, o usuário insere a figura e a fórmula na página. A Figura 9 mostra uma página completa com texto, fórmula e figura. Observa-se, para o exemplo mostrado nessa figura, a existência de abas indicando, além da página inicial "Descrição do fenômeno", uma página de "Questões conceituais" e "Atividades". Ao ativar cada uma delas, o usuário encontrará uma série de informações necessárias ao entendimento da simulação e as tarefas propostas. Claro que a concepção dessas páginas depende do objetivo que se pretende atingir!

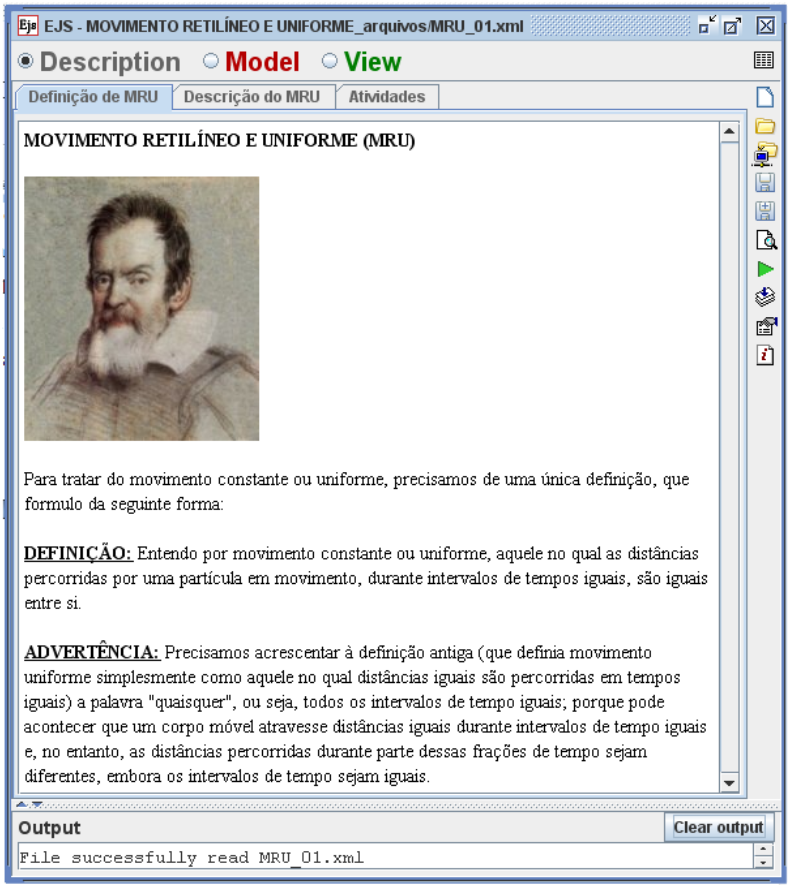


Figura 9 – Exemplo para uma página do painel *Description*.

2.4 COMPONENTES DO MODELO DESENVOLVIDO COM O EJS

Um modelo desenvolvido com o EJS deve apresentar o conjunto de grandezas relevantes do fenômeno simulado, os valores iniciais atribuídos para algumas delas e a escrita das regras (leis físicas) que governam a mudança dos valores das grandezas em função do tempo, por exemplo. Ao se escrever a simulação, a palavra grandeza pode representar uma constante ou uma variável. Na elaboração do modelo é possível, ainda, escrever os algoritmos que demonstram a ação interativa do usuário sobre a janela de visualização do laboratório virtual.

O procedimento usado por EJS para introduzir essas informações é formado pelas seguintes partes:

a) Declaração das variáveis: As variáveis e seus valores iniciais (se existirem) são declarados na página *Variables* do painel *Model*.

b) Inicialização das variáveis: Na página *Initialization* do painel *Model* são escritos os algoritmos necessários para inicializar algum tipo de variável ou conjunto de variáveis.
c) Evolução das variáveis: Na página *Evolution* do painel *Model* são escritos os algoritmos que descrevem como algumas variáveis evoluem com o tempo, por exemplo.
d) Relações fixas: Na página *Fixed relations* do painel *Model* são escritos os algoritmos (ou equações) que estabelecem relações entre as grandezas algébricas.
e) Métodos próprios: Na página *Custom* do painel *Model* são escritos os algoritmos que podem ser requisitados em qualquer ponto da descrição do modelo ou, ainda, da janela de visualização da simulação. A palavra método utilizada na linguagem Java é denominada em outras linguagens de programação função ou sub-rotina.

Nos próximos itens serão mostrados como proceder em cada uma das partes descritas acima.

2.5 O ALGORITMO DE SIMULAÇÃO DE EJS

Um algoritmo de simulação de EJS é estruturado de forma a mostrar a ordem em que EJS executa os diferentes painéis e as diferentes janelas dentro de cada painel. O fluxograma da Figura 10 mostra a sequência de tarefas que EJS realiza durante a execução da simulação de um laboratório virtual. Abaixo são descritas sucintamente tais tarefas realizadas por EJS:

a) Declaração das variáveis e atribuição de valores iniciais de algumas variáveis no painel *Variables*.
b) Execução dos algoritmos descritos no painel *Initialization*.
c) Execução dos algoritmos escritos no painel *Fixed relations*.
d) EJS "cria" a janela de visualização da simulação e a exibe na tela do monitor, estabelecendo as propriedades dos elementos gráficos e as variáveis do modelo, assim como a reação entre as ações do usuário sobre o modelo. Dessa forma, diz-se que o modelo se encontra em seu estado inicial e a janela de visualização reflete os valores das variáveis neste estado.
e) EJS examina se o usuário interage com a simulação através da janela de visualização, que pode ocorrer de duas formas:

e.1) O usuário interage com a janela de visualização, EJS executa os algoritmos associados à interação e, em seguida, ele executa os algoritmos do painel *Fixed relations*.

e.2) O usuário interage com a janela de visualização, EJS executa os algoritmos do painel *Evolution* e, a seguir, os algoritmos do painel *Fixed relations*.

f) EJS representa na janela de visualização do laboratório virtual o novo estado do modelo que foi avaliado no passo anterior.

g) EJS retorna ao passo 4.

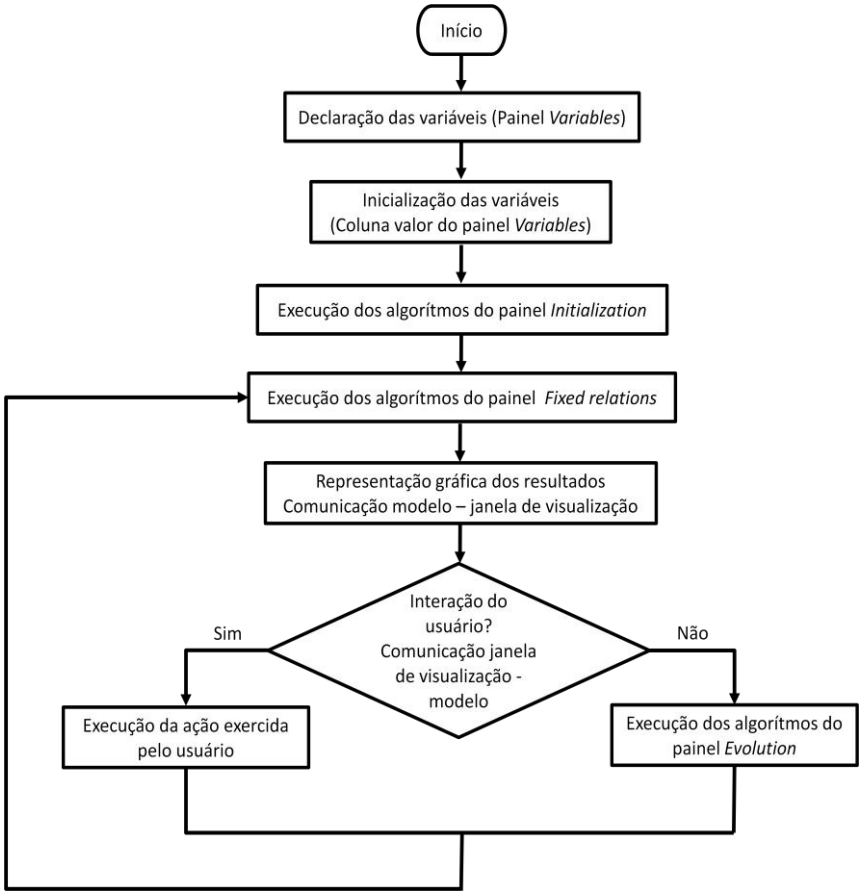


Figura 10 – Fluxograma do algoritmo da simulação com EJS.

Os valores característicos das grandezas físicas envolvidas num fenômeno podem mudar com o tempo por dois motivos:

a. Devido à dinâmica interna da simulação, denominada de evolução (E*volution*).
b. Devido às influências de agentes externos, isto é, da interação direta do usuário na simulação.

As alterações causadas diretamente por um deles podem provocar alterações indiretas em outras grandezas físicas (variáveis). Isto acontece se existirem grandezas cujos valores dependem explicitamente daquelas que foram modificadas. Dessa forma, diz-se existir

vínculos (*Fixed relations*) entre as variáveis. Os processos de mudança dos valores das variáveis são regulados por fórmulas matemáticas, isto é, algoritmos que descrevem as leis sob as quais as evoluções ocorrem, ou da dependência entre as grandezas.

Dependendo da habilidade de quem escreve o modelo, podem aparecer em um mesmo painel várias páginas. EJS executa-as seguindo a ordem em que elas aparecem, isto é, da esquerda para a direita.

Os algoritmos escritos em uma página são executados na ordem em que estão escritos, da parte superior da página para a parte inferior, da mesma forma como são executados os comandos de código de uma linguagem de programação.

2.6 A OPÇÃO *VARIABLES* DO PAINEL *MODEL*

Uma vez caracterizado o fenômeno que se pretende simular e caracterizadas as grandezas físicas que intervêm no fenômeno, o usuário deve declará-las e atribuir os seus valores iniciais (quando necessário). Para isso, uma vez ativado o botão *Variables* e após o usuário ter clicado na parte central do painel, aparece uma caixa onde se deve atribuir um nome ao conjunto de variáveis (*Var Table*) a ser declarado, como mostra a Figura 11. Da mesma forma que em *Description*, o usuário pode criar quantas páginas forem necessárias. Isso depende do grau de complexidade da simulação a ser criada, do grau de experiência e do conhecimento da ferramenta EJS daquele que a está desenvolvendo ou, ainda, por considerações didáticas no processo de criação da simulação.

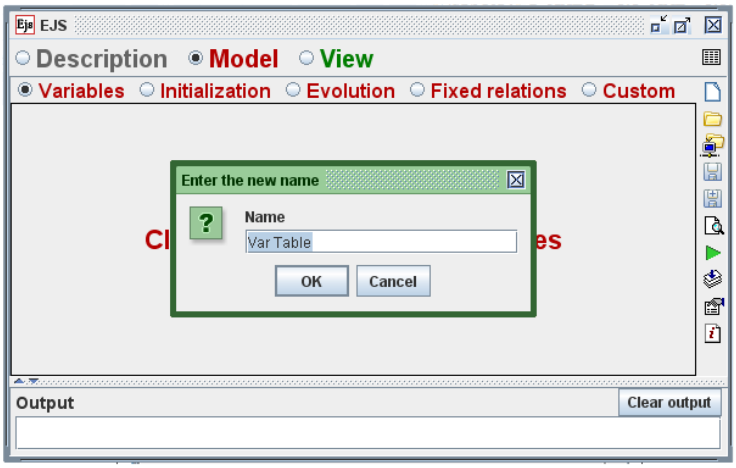


Figura 11 – Painel *Model* ativado para a criação da tabela de variáveis.

Realizada a tarefa anterior, uma nova janela, em forma de tabela, aparece na tela do monitor. Nessa tabela, como mostra a Figura 12 na página 97, o usuário pode observar as colunas com os títulos: *Name*, *Initial Value*, *Type* e *Dimension*. Os nomes das colunas indicam que o usuário deve inserir (declarar) um nome da variável na coluna *Name*, atribuir (ou não) um valor inicial à ela na coluna *Initial Value*, especificar o tipo de variável na coluna *Type* e a sua

dimensão na coluna *Dimension*, caso a variável seja um vetor ou uma matriz. Abaixo dessas designações, tem-se uma linha, em azul, onde será escrito o nome, o valor inicial, o tipo (que por *default* aparece *double*, isto é, real de dupla precisão) e a dimensão.

O EJS fornece a opção para se alterar o tipo de variável diretamente na coluna *Type* ao lado da palavra *double*. Ao se clicar do lado direito dessa palavra, com o botão contrário do *mouse* (Figura 12), aparece uma coluna com as seguintes opções: *boolean*, *int*, *double*, *String* e *Object*.

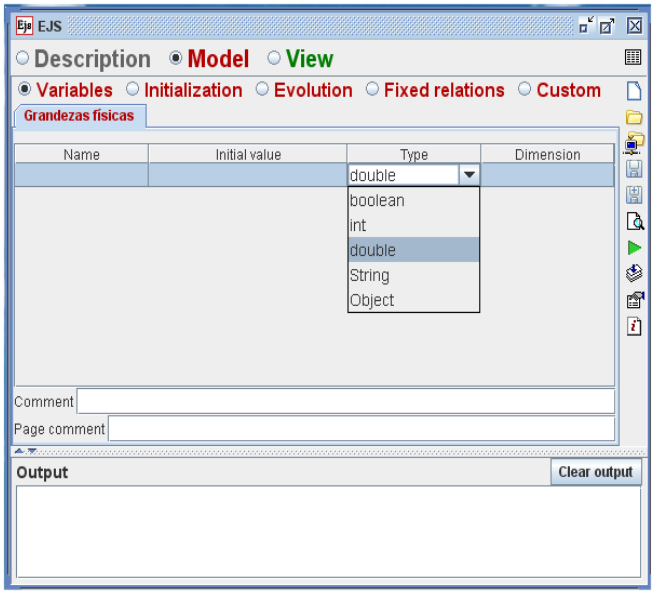


Figura 12 – Atribuição dos valores iniciais das variáveis.

Ao clicar na linha para preenchê-la com o nome, o valor inicial, etc., automaticamente uma nova linha é criada. Isso ocorre sucessivamente para cada variável que é colocada em sua respectiva linha.

Após a inserção das variáveis é possível, ainda, editar essa página. Pode acontecer, por exemplo, de o usuário ter que alterar a ordem de apresentação das variáveis, ou adicionar variáveis e apagar (*delete*) uma determinada linha. Clicando-se sobre uma linha, em qualquer parte dela, com o botão "contrário" do *mouse*, é possível efetuar as alterações desejadas através de uma coluna que aparece com as seguintes opções: *Insert before* (inserir linha antes), *Insert after* (inserir linha depois), *Move up* (mover a linha para cima), *Move down* (mover a linha para baixo), *Cut* (recortar), *Copy* (copiar), *Paste* (colar) e *Delete* (apagar). A Figura 13, p. 98, ilustra a janela *Variables* completamente preenchida com a opção, só para exemplificar, de se efetuar uma alteração desejada. No final da página, no campo *Output*, visualiza-se a mensagem decorrente da compilação.

Ao se escrever um programa de computador numa linguagem como *Java*, *C*, *C++*, *Fortran*, etc., o programador deve inserir comentários para que outras pessoas possam entender a lógica do programa. Na Janela *Variables*, observam-se dois campos relacionados com a ação de se documentar a simulação (Figura 13). O primeiro é o campo denominado *Comment*, onde a pessoa que estiver preparando a simulação faz uma descrição da variável; o segundo é o *Page comment*, onde se faz o comentário da página. Esses campos são usados para se efetuar a documentação do modelo como, o que representa a grandeza, se é uma

página de parâmetros, etc. Novamente, dependendo da sofisticação da simulação e da experiência do programador, mais páginas para *Variables* poderão ser acrescentadas.

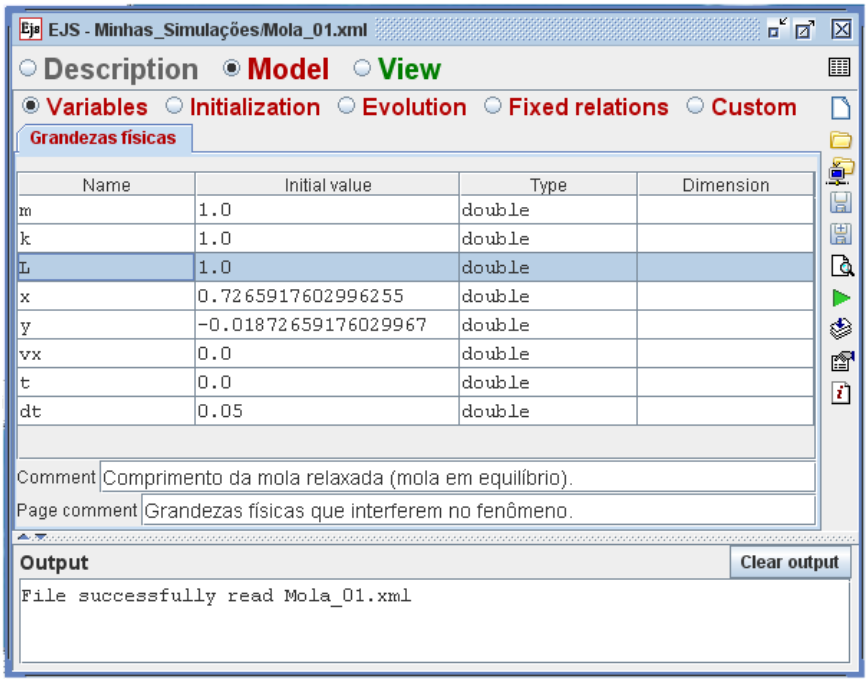


Figura 13 – Página *Variables* completa.

Para finalizar, salientamos que é preciso declarar, mas não inicializar, as variáveis do tipo algébricas e as derivadas das variáveis de estado.

2.7 A OPÇÃO *INITIALIZATION* DO PAINEL *MODEL*

Os valores iniciais de algumas variáveis podem ser dados diretamente na coluna *Initial Value* da tabela descrita no item anterior, especificando-se de forma inequívoca o estado inicial do sistema. Entretanto, dependendo da complexidade do fenômeno que se deseja simular, inicializações mais elaboradas serão necessárias. Nesse caso, o EJS permite criar uma página própria, como mostra a Figura 14 (p. 99), para se escrever códigos de inicialização, como por exemplo, ao se atribuir diferentes valores aos elementos de um arranjo, ou especificar o valor de uma variável que deve ser calculado inicialmente através de um algoritmo matemático, etc. Essas páginas são criadas no momento em que o botão *Initialization* é ativado e da mesma forma como descrito nos casos anteriores.

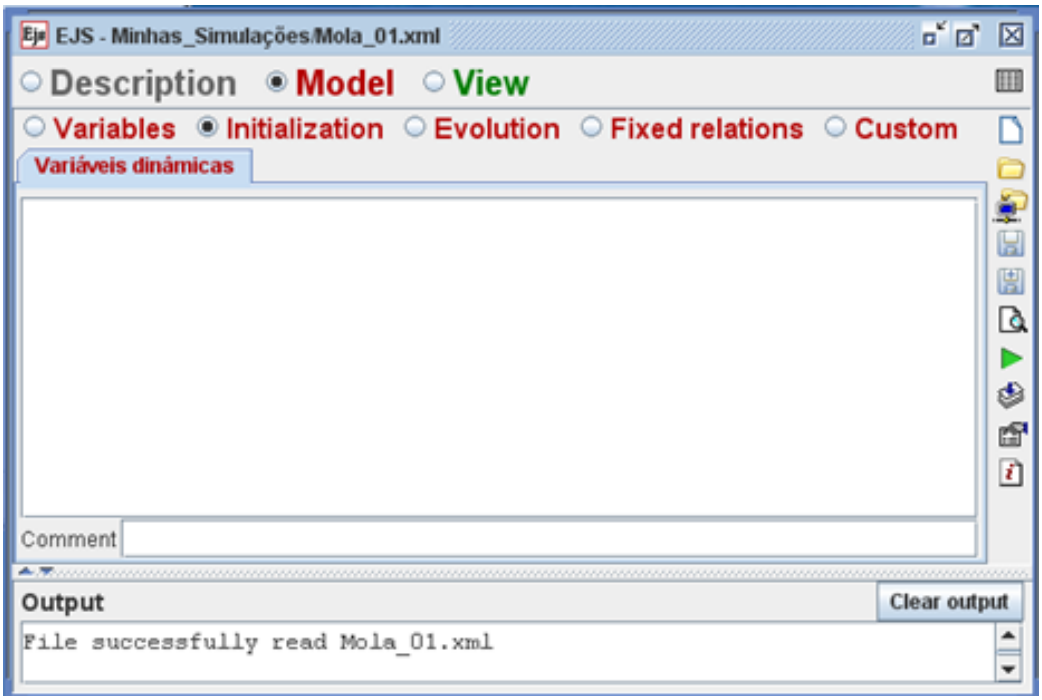


Figura 14 – Exemplo de página *Initialization* para a escrita do código Java.

Uma vez escrito o código em *Java*, nessa página, ele será executado uma única vez no início da simulação e na mesma ordem em que as abas são mostradas (se existirem outras abas). Esses valores podem ser especificados na página *Initialization*, como mostra a Figura 15 da próxima página. Com essa opção, o usuário apenas declara as variáveis na coluna *Name* que intervêm no modelo, sem atribuir seus valores iniciais, e efetua a atribuição na página *Initialization*.

Algumas vezes, porém, será preciso efetuar uma inicialização mais complexa, através de um cálculo prévio. Assim, o uso de uma página no painel *Initialization* será obrigatório e nela se escreverá o código *Java* necessário para completar a inicialização do estado do sistema.

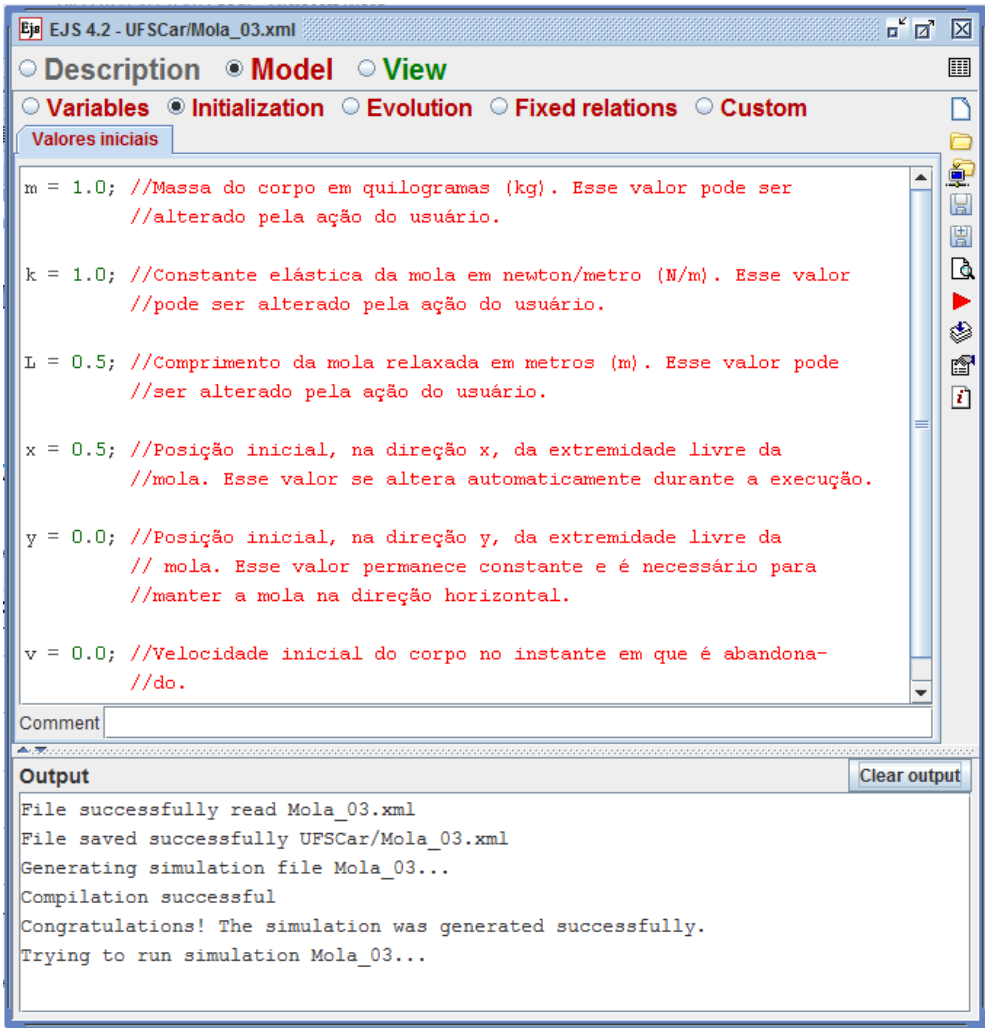


Figura 15 – Exemplo de uma página *Initialization* com a atribuição dos valores iniciais.

2.8 A OPÇÃO *EVOLUTION* DO PAINEL *MODEL*

A Figura 16, na página 101, mostra o painel *Evolution* quando selecionada pelo usuário. Ele se apresenta dividido em dois subpainéis. No superior aparece a mensagem *"Click to create a page of code"* em que o usuário deverá escrever o algoritmo em Java que deseja, isto é programar o próprio método de integração para calcular as grandezas (variáveis) de estado. Na parte inferior lê-se *"Click to create a page o ODEs"*, onde o usuário escreverá as equações diferenciais ordinárias (EDOs) que descrevem o modelo. Esta página é um assistente para a definição das EDOs, onde EJS gera automaticamente o código para integrar numericamente tais equações.

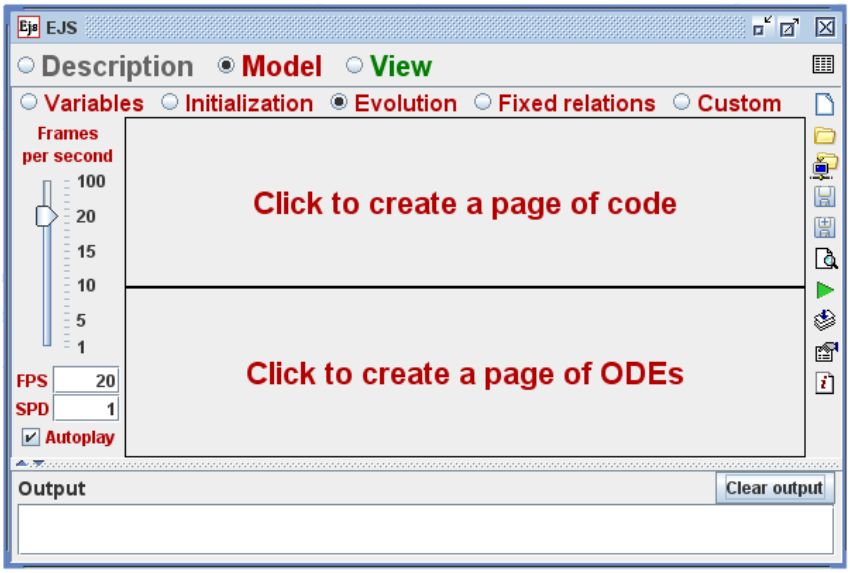


Figura 16 – O painel *Evolution* ativado.

No lado esquerdo do painel existe um cursor deslizante que modifica o valor do chamado *Frames per second* (quadros por segundo), seguido pelo rótulo FPS que mostra justamente o valor indicado pelo cursor deslizante. Logo abaixo, encontra-se o rótulo SPD (*Steps per display*) que indica o número de passos dados pelo modelo antes de atualizar a visão (janela de visualização). Esses valores podem ser introduzidos pelo usuário ao digitar o número desejado no campo correspondente e apertar a tecla *enter*.

De forma padronizada (*default)*, a evolução da simulação se inicia automaticamente. Isso é mostrado no *check box* ativado do *Autoplay*. Entretanto, se o usuário desejar, essa opção pode ser desabilitada ao clicar na caixa.

Como mencionado anteriormente, as equações de evolução podem ser escritas através de algoritmos de código em Java se o usuário desejar. Uma vez selecionada essa opção, o usuário poderá incluir o número de páginas necessárias para efetuar a programação. Porém, se o usuário desejar apenas escrever as EDOs, basta escrevê-las no campo mostrado na Figura 17 (p. 102) e escolher o método numérico para a sua resolução. O método numérico é mostrado no campo *Solver*. Basicamente, para qualquer método escolhido, EJS realiza as seguintes tarefas:

a) Calcula o valor das variáveis de estado, aplicando para isso o algoritmo correspondente ao método de integração escolhido pelo usuário. Se o valor atual da variável tempo é t e o passo de integração é Δt, então o valor calculado da variável de estado será o valor para o instante $t + \Delta t$.

b) Incrementa o valor da variável tempo em Δt.

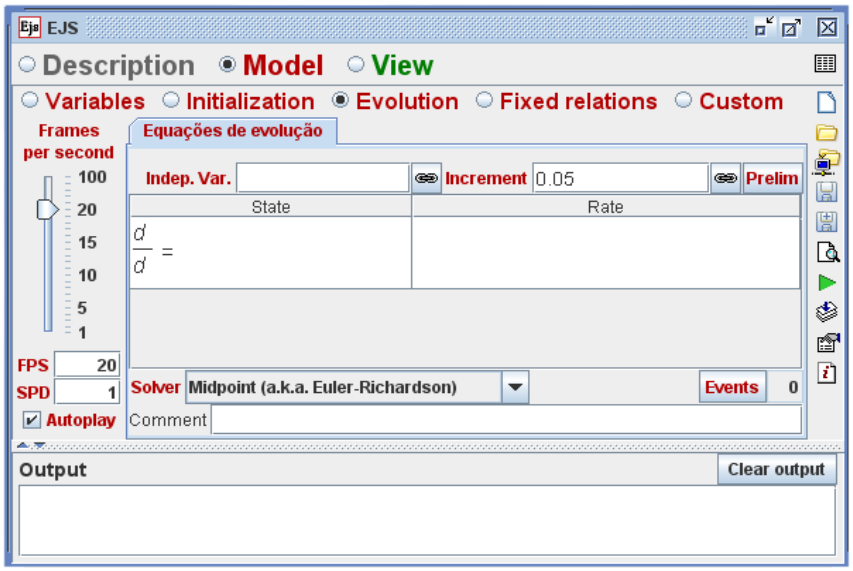


Figura 17 – Página de edição das EDOs ativada com o método numérico selecionado.

Para escrever a EDO, o usuário deve escolher a variável independente, em geral o tempo (*t*). Para isso, deve-se digitar diretamente a letra *t* (ou o nome da variável) no campo *Indep. Var.* Analogamente, deve-se digitar o símbolo (letra ou nome) representativo para o incremento no campo *Increment*. Alternativamente, pode-se clicar sobre ícone da corrente que aparece ao lado de cada campo mencionado anteriormente. Ao fazê-lo, aparece na tela uma caixa com a lista dos nomes das variáveis declaradas. Basta, então, dar um clique na variável desejada e o campo será automaticamente preenchido pela escolha do usuário. A Figura 18, na próxima página, resume o que foi descrito no último parágrafo.

Prosseguindo no processo de inserção da EDO, o usuário deve dar duplo clique na região onde aparece o símbolo $\frac{d}{d} =$ (veja a Figura 17), e digitar a letra (ou nome) da variável de estado. Outra forma é efetuar o que foi descrito no parágrafo anterior, isto é, clicar no campo com o botão contrário do *mouse* e selecionar na caixa de diálogo que aparece a variável desejada, como mostra a Figura 18. Já no campo *Rate*, o usuário digitar um valor, uma variável ou uma expressão correspondente à taxa de variação da grandeza de estado, isto será a derivada. A Figura 19, na próxima página, mostra o resultado final de todo esse procedimento.

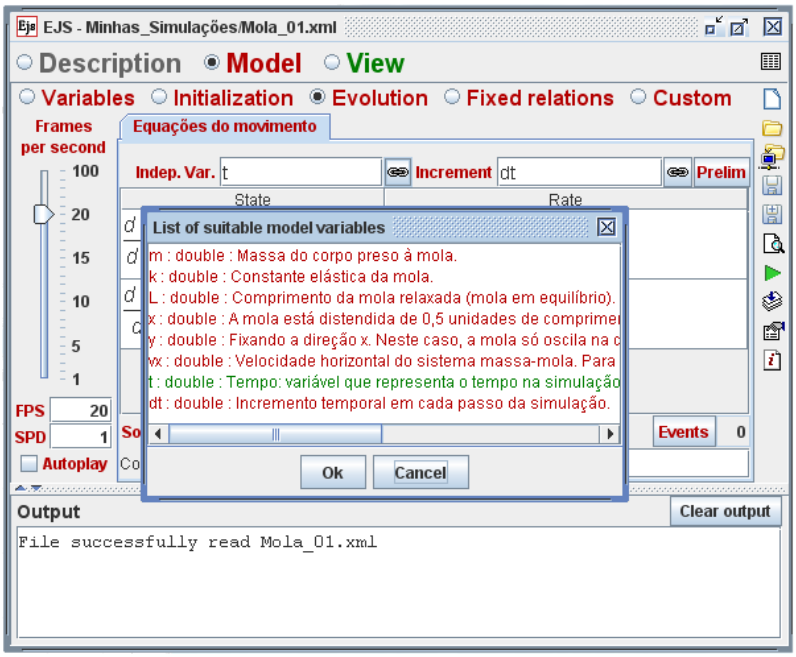


Figura 18 – Seleção da variável independente e do incremento.

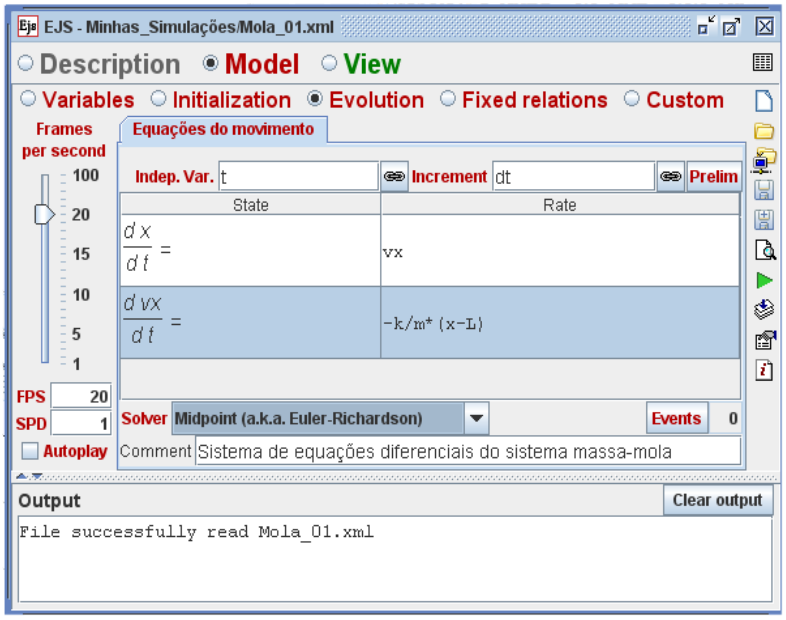


Figura 19 – Edição das equações diferenciais para um sistema massa-mola.

É importante salientar a forma de sistema de equações diferenciais que aparecem nessa página. Isso ocorre pelo fato de EJS não apresentar um método direto de resolução de equação diferencial ordinária de segunda ordem. Assim, a equação diferencial

ordinária $\frac{d^2x}{dt^2}+\frac{k}{m}.(x-L)=0$ para o sistema massa-mola, sem dissipação, deverá ser escrita como um sistema de equações diferenciais da forma:

$$\begin{cases}\frac{dx}{dt}=vx\\ \frac{dvx}{dt}=-\frac{k}{m}\cdot(x-L)\end{cases}$$

Alertamos o usuário que no painel *Evolution* só pode existir uma página com equações diferenciais.

2.9 AS OPÇÕES *FIXED RELATIONS* E *CUSTOM* DO PAINEL *MODEL*

Uma vez selecionada a opção *Fixed relations* e após nomear uma página (vide Figura 20), como descrito nas seções anteriores, o usuário pode escrever os algoritmos em código Java para calcular os valores das variáveis algébricas do modelo.

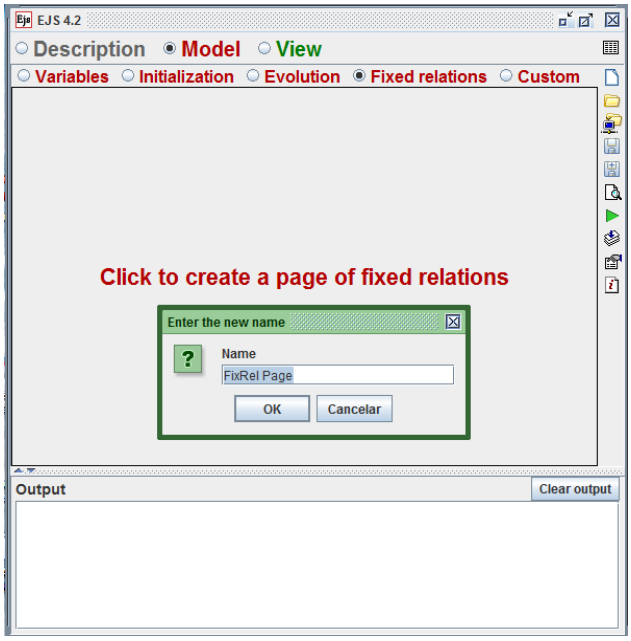


Figura 20 – Criando uma página de código no painel *Fixed relations*.

As equações usadas numa página do painel *Fixed relations* são aquelas que descrevem o comportamento do modelo durante a sua evolução com o tempo e, também, o comportamento no caso em que o usuário realiza qualquer mudança interativa do valor de alguma variável. Esse parágrafo mostra a diferença conceitual entre uma equação escrita no painel *Evolution* e da equação escrita no painel *Fixed relations*.

A Figura 21 mostra uma página escrita no painel *Fixed relations*. Pode-se observar o código em *Java* para o usuário selecionar uma opção (interação usuário-modelo) e equações algébricas exemplificadas para o caso de transformações de escalas termométricas.

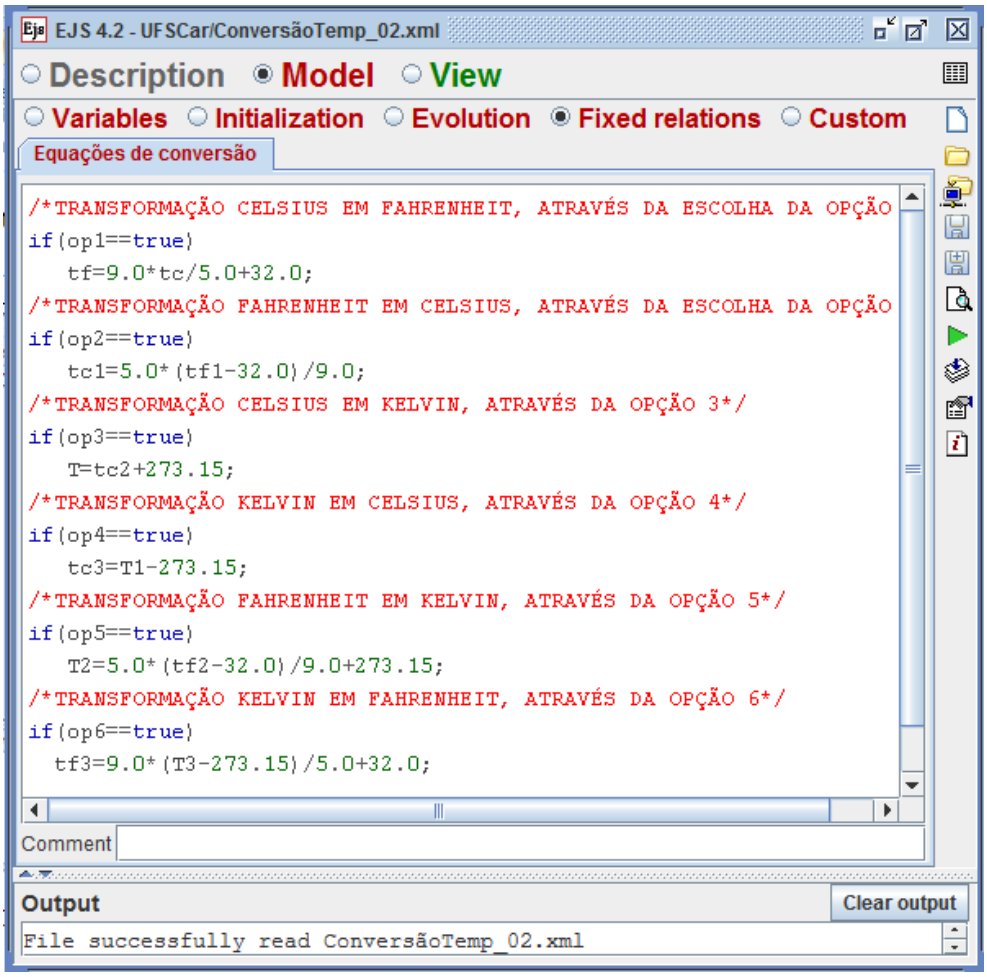


Figura 21 – Exemplo de uma página escrita em código *Java* do painel *Fixed relations*.

Ao criar uma página no painel *Custom*, o usuário poderá escrever em código *Java* todos os métodos necessários para a definição do modelo ou da visão (janela de visualização). Esses métodos são escritos para definir ações sobre o modelo, isto é, ações que são ativadas a partir da janela de visualização, como, por exemplo, quando o usuário acionar um botão, ou modificar um valor numérico de uma grandeza física.

Para completar essa seção, é possível realizar algumas ações como "habilitar/desabilitar" e "mostrar/esconder" páginas já preparadas. Isso é feito ao se clicar sobre o nome da página criada com o botão contrário do *mouse*. A Figura 22 mostra o *menu popup* resultante dessa ação.

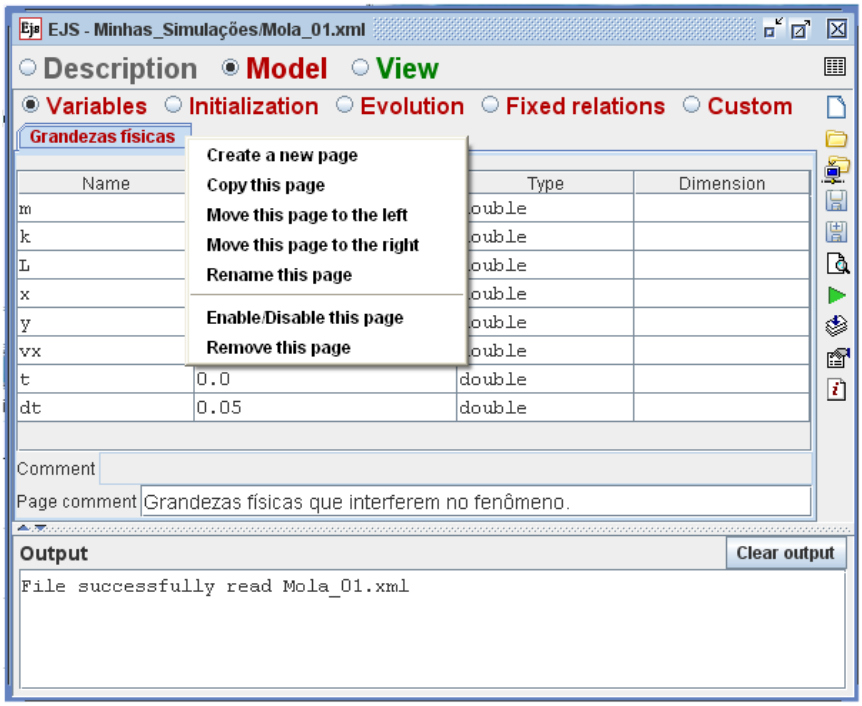


Figura 22 – *Menu popup* para a edição de páginas.

A opção *Enabling and Disabling* (habilitado e desabilitado) é útil para testar vários algoritmos para um dado problema e escolher qual é o melhor. Assim, o usuário pode escrever vários algoritmos em diferentes páginas e com a função citada, testar cada um deles. As páginas desabilitadas (*Disabled*) são facilmente reconhecidas através do sufixo *D* ao lado do nome da própria página.

Finalmente, a opção *Show and Hide* (mostrar e esconder) permite manter uma página escondida. Essa opção tem um propósito pedagógico ao se criar simulações para os estudantes. Para isso, é possível escrever uma página com códigos e configurar o EJS de forma a não mostrá-la. Páginas escondidas são invisíveis, mas ativas, e mostram aos estudantes uma parte simplificada do modelo. Mais tarde, quando os estudantes estiverem preparados, pode-se, então, tornar a página visível e mostrar toda a simulação a eles. As páginas proibidas (*Hidden Pages*) estão indicadas com o sufixo *H* após o nome da página.

3 "CONSTRUÇÃO" DE UMA JANELA DE VISUALIZAÇÃO (*VIEW*)

A janela de visualização de uma simulação virtual é a interface gráfica entre o usuário e o modelo. Através da manipulação dos controles inseridos nessa janela, o usuário pode controlar a execução da simulação e alterar os valores de algumas grandezas físicas presentes no fenômeno em estudo.

Basicamente, a janela de visualização deve seguir duas etapas. Na primeira etapa, constrói-se a árvore dos elementos, resultante da estruturação de alguns elementos gráficos selecionados pelo usuário. A segunda etapa consiste em editar as denominadas *propriedades* de cada elemento e que servem para mostrar a aparência (cor, forma, tamanho, etc.) de alguns elementos que fazem parte do laboratório virtual, ou para modificar os valores das variáveis mostradas na própria janela. É justamente essa última possibilidade que torna EJS uma ferramenta especial capaz de converter a simulação em uma visualização dinâmica do fenômeno físico.

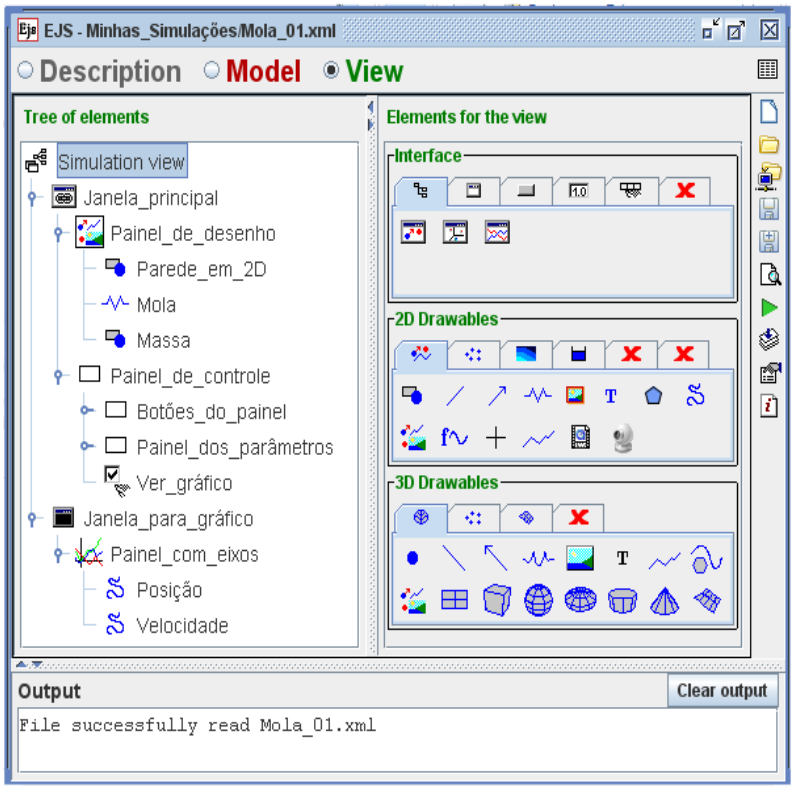


Figura 23 – O painel *View* com a árvore de elementos estruturada.

A Figura 23 mostra o painel *View* ativado. Na coluna da esquerda observa-se a árvore de elementos (*Tree of elements*), já estruturada em função da concepção daquele que criou a simulação. Na coluna da direita estão agrupados os diferentes elementos agrupados em classes (*Interface*, 2D Drawables, *3D Drawables*). Cada classe, por sua vez, apresenta conjuntos (representadas por diferentes abas com correspondentes ícones) que contêm elementos específicos como setas, linhas, imagens, etc.

Sem dúvida, a parte mais importante na criação e preparação da simulação com o EJS é conhecer cada elemento, aprender a função de cada um deles, como usá-los e

personalizá-los. Para isso é necessário editar as propriedades desses elementos, como dissemos no início desse item. Essas propriedades são valores internos do elemento que estabelece como é a sua aparência e como se comporta durante a execução da simulação na janela de visualização.

3.1 A ÁRVORE DE ELEMENTOS (*TREE OF ELEMENTS*)

Criar uma janela de visualização, cujo objetivo é mostrar o desenvolvimento da simulação através de animação ou gráficos, consiste em gerar uma estrutura de elementos agrupados em forma de árvore de elementos, como mostrada na coluna da esquerda da Figura 23 (p. 107). Essa estrutura deve incluir:

a. Elementos de visualização do estado do modelo.
b. Elementos para usar a interatividade entre usuário e simulação.
c. *Containers* que guardam todos os outros elementos.

A janela de visualização de um laboratório virtual compõe-se de elementos gráficos que podem estar inseridos em outros elementos, formando a estrutura da árvore. Dessa forma, diz-se que um elemento A é "pai" de um elemento B se A contém B. Assim, o elemento B é "filho" do elemento A. Um "pai" pode ter vários "filhos", porém um "filho" só pode ter um único "pai". Durante a construção da árvore de elementos, observa-se um enlace entre os elementos A e B.

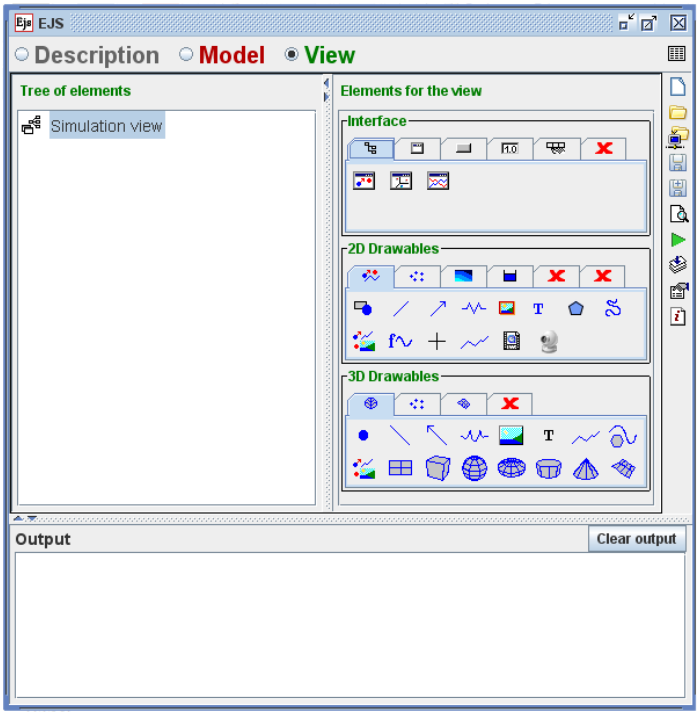


Figura 24 – Inicialização da janela de visualização – *View*.

A Figura 24 mostra o estágio inicial da janela de visualização, logo após a seleção do botão *View*. Observa-se na coluna esquerda da janela o título *Tree of elements* (árvore de elementos) e abaixo uma região em branco com um ícone de rótulo *Simulation view* (visualização da simulação). Isto constitui a raiz da árvore. Na verdade ele não é um

elemento gráfico, mas serve como um elemento a partir do qual serão colocados outros elementos.

O primeiro "filho" na raiz da árvore é chamado de janela principal (*main window*), que é aquele que aparece em primeiro lugar na tela do monitor do computador e, também, que EJS inclui na página HTML ao gerar a simulação.

Para se adicionar um elemento na raiz da árvore, basta escolher o tipo de elemento que aparece na coluna *Elements for the view* e efetuar um clique do botão do *mouse* sobre este elemento. A cor de fundo deste ícone selecionado se altera, indicando que a seleção está ativada. Observa-se, também, que o cursor do mouse, que por *default* é uma seta, se altera em uma *varinha mágica*, mostrada na Figura 25. Leva-se, então, essa *varinha mágica* até o elemento, do tipo *container*, na árvore de componentes, ou seja, clicando sobre o ícone *Simulation view*. Semelhante às ações descritas anteriormente surge na tela do monitor uma janela convidando o usuário a fornecer um nome ao elemento que está "sendo criado". O nome escolhido é Janela principal, como mostrado na Figura 26 na página 110.

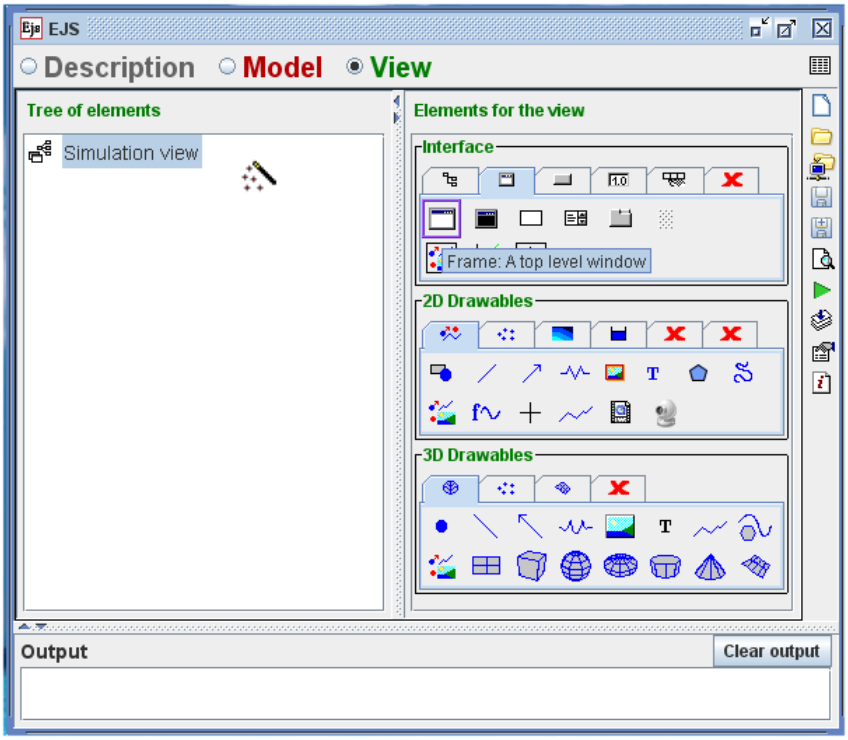


Figura 25 – Acrescentando os "pais" e "filhos" na árvore de elementos com o uso da *varinha mágica*.

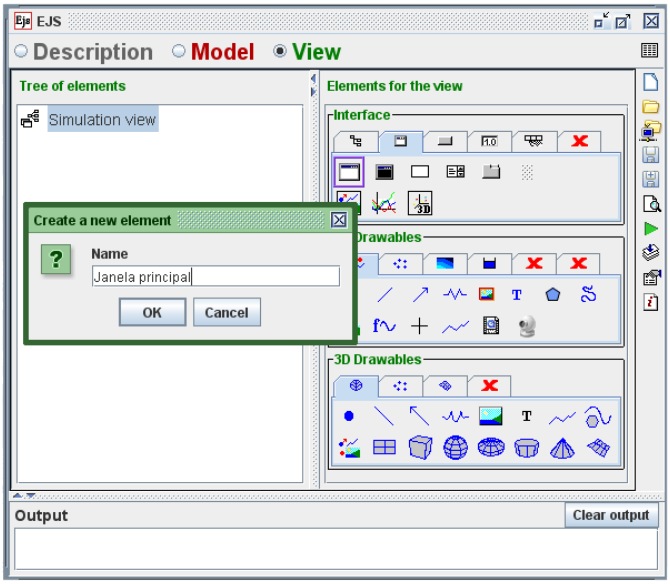


Figura 26 – Nome do elemento que está sendo inserido na árvore de elementos.

Ao finalizar a etapa anterior, clicando-se em *OK*, o elemento aparece inserido na árvore (Figura 27) e uma nova janela, que será usada para mostrar todo o desenvolvimento da simulação, aparece na tela do monitor ao lado do painel de EJS, ou em outra posição que depende da configuração do computador (Figura 28, p. 111).

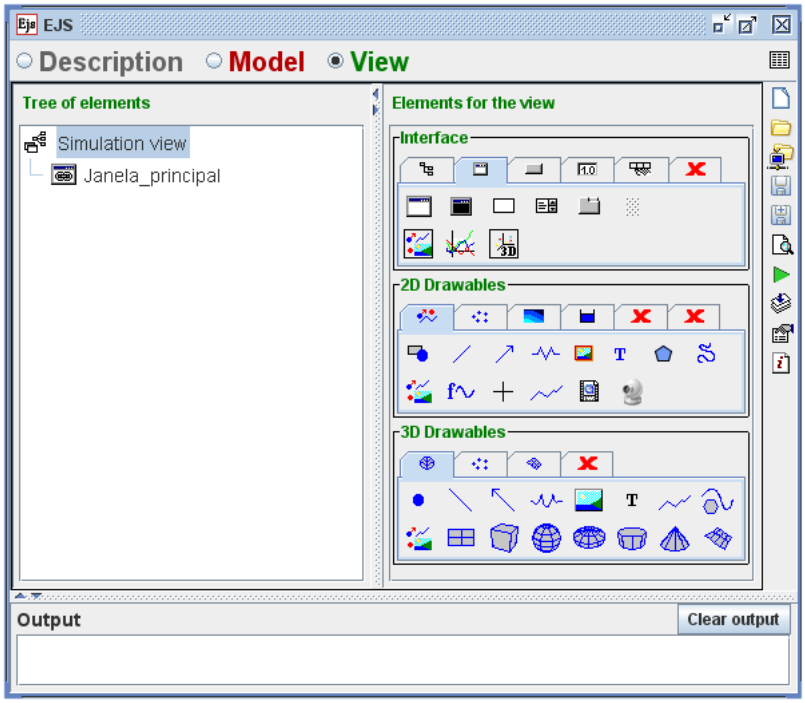


Figura 27 – Elemento “Janela_principal” adicionado na árvore de elementos.

Aparentemente, a tarefa de se criar a janela de visualização é muito simples. Entretanto, é o usuário que deve fornecer os parâmetros relacionados com a posição dos elementos, ou seja, a aparência (*layout*) da simulação mostrada na tela do computador. Para isso, faz-se necessário conhecer as propriedades dos elementos, algumas delas descritas no tutorial. Nas próximas seções, entretanto, estaremos apresentando a descrição de alguns dos elementos, de suas propriedades e como acessar a janela de propriedades para poder efetuar modificações que o usuário desejar. Por enquanto, destacamos que os *layouts* são de cinco tipos cujas funções são descritas abaixo, bem como a aparência de cada um deles na tela do monitor.

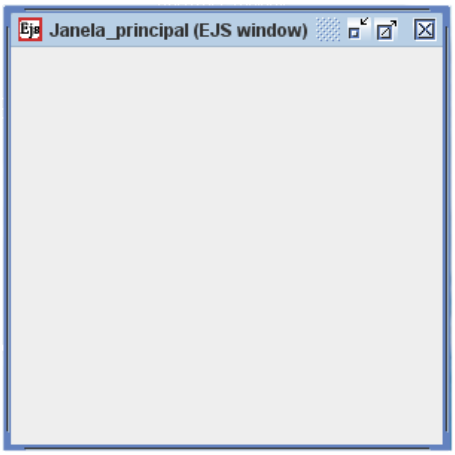


Figura 28 – Apresentação da Janela principal na tela do monitor.

f. *Flow* – Distribui os "filhos" em uma linha, da esquerda para a direita, de forma similar às palavras escritas em um texto. Os "filhos" podem estar na esquerda, no centro ou na direita. A Figura 29, a seguir, mostra a disposição de alguns botões de comando utilizando-se a opção *Flow*.

Figura 29 – *Layout* do tipo *Flow*.

g. *Border* – Distribui os "filhos" em uma das cinco posições: *up* (em cima), *down* (em baixo), *left* (esquerda), *right* (direita) e *center* (centro), como mostra a Figura 30. Os elementos podem ter dimensões diferentes.

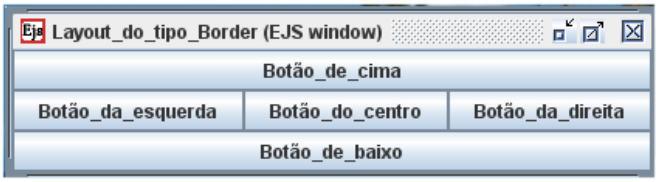


Figura 30 – *Layout* do tipo *Border*.

h. *Grid* – Localiza os "filhos" em uma tabela retangular com linhas e colunas igualmente dimensionadas e de acordo com a indicação do usuário (Figura 31).

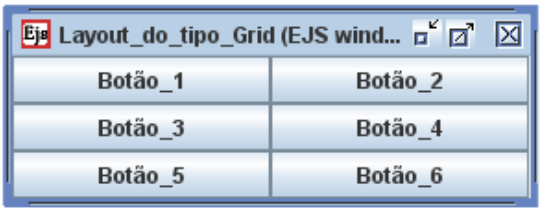


Figura 31 – *Layout* do tipo *Grid* com a opção três linhas por duas colunas.

i. *Horizontal Box* – Sua função é similar ao *grid*, só que os "filhos" ficam dispostos em linha (Figura 32).

Figura 32 – *Layout* do tipo *Horizontal box*.

j. *Vertical Box* – Sua função também é similar ao *grid*, com os "filhos" dispostos em uma coluna simples. Os "filhos" podem apresentar tamanhos diferentes (Figura 33).

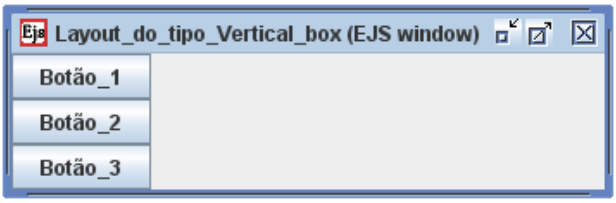


Figura 33 – *Layout* do tipo *Vertical box*.

Em todos os *layouts* apresentados é possível alterar parâmetros que indicam a separação vertical e horizontal dos "filhos", cor, dimensões, etc.

3.2 *CONTAINERS*

Como mencionado no item anterior, existem elementos que são denominados de *Containers*, ou seja, elementos que guardam outros elementos. Para identificá-los, selecione com o *mouse* na coluna *"Elements for the view"*, a segunda aba (*"Windows, containers, and drawing panels"*) correspondente ao conjunto de ícones denominado *"Interface"* (vide Figura 23, p. 107). Ao fazê-lo, o usuário verifica a existência de oito ícones, cada um com um nome e função descrita a seguir.

 **Frame –** Sua principal característica é o que fazer no ambiente Windows do sistema operacional do computador, como por exemplo, permitir minimizar ou maximizar a janela. Cada janela de visualização deve ter pelo menos um elemento deste tipo, no qual serão colocados outros elementos. Este primeiro elemento é geralmente chamado de janela principal (*main window*) e deve ser adicionado na raiz da árvore, isto é em *Simulation view*.

 **Dialog** – Janelas de diálogo são do tipo independente que não podem ser minimizadas, mas podem ser escondidas. Apresentam a habilidade de se manterem visíveis em cima das janelas que a antecedem na árvore de elementos, recurso útil em simulações com muitas janelas. Ele também deve ser adicionado na raiz da árvore (*Simulation view*).

 **Panel** – É o tipo de *container* mais básico e, por isso, o primeiro candidato a ser usado para agrupar outros elementos.

 **SplitPanel** – Este tipo de *container* apresenta duas áreas separadas cujas dimensões podem ser alteradas pelo usuário.

 **TabbedPanel** – Este *container* mostra somente um "filho", dentre vários, de cada vez, organizando-os através do uso de abas (*tabs*).

 **DrawingPanel** – *Container* básico para desenhos em duas dimensões.

 **PlottingPanel** – É um *container* que permite adicionar um sistema de eixos cartesianos ou um sistema de coordenadas polares.

 **DrawingPanel3D** – Versão em três dimensões do *container* DrawingPanel.

Os *containers Panel*, *SplitPanel*, *TabbedPanel*, *DrawingPanel*, *PlottingPanel* e *DrawingPanel3D*, não podem ser adicionados na raiz, isto é, eles só podem ser adicionados nos *containers Frame* ou *Dialog*. Para cada um deles existe um conjunto de propriedades.

Uma vez inserido o elemento na árvore e ao após clicar duas vezes sobre o ícone representativo, aparecerá na tela do monitor uma tabela onde é possível modificar os parâmetros relacionados com a aparência da simulação. A Figura 34 mostra a edição da propriedade para o container *Frame*.

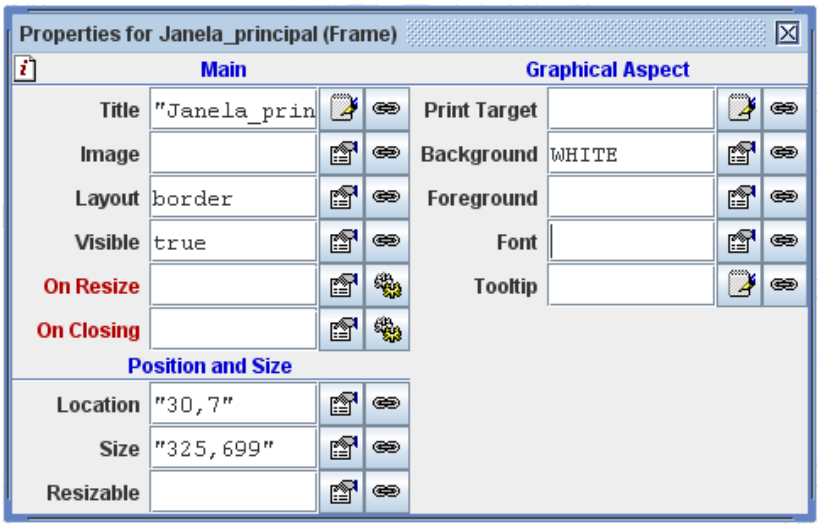


Figura 34 – A tabela de propriedades do *container Frame*.

3.3 ELEMENTOS BÁSICOS

Cada aba do grupo *Interface* corresponde a uma família de elementos com funções específicas. Por exemplo, a terceira aba (*Buttons ande decoration*) do conjunto *Interface* é formada por um conjunto de elementos que podem ser usados para decorar a janela, mostrar e editar as variáveis pertencentes ao modelo matemático (Figura 35).

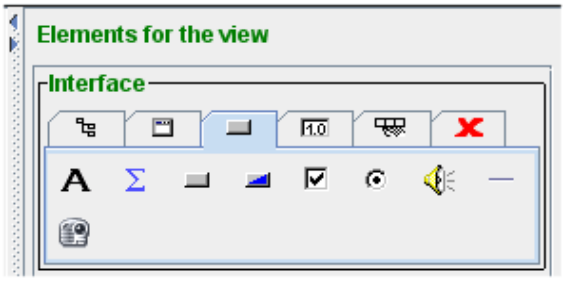


Figura 35 – Ativação da família correspondente a *Buttons and decoration*.

Efeito semelhante é obtido ao se ativar a quarta (Figura 36) ou a quinta (Figura 37) aba. Esses elementos apresentam, também, a função de "gatilho de ação", isto é, permitirão uma determinada intervenção do usuário durante a execução da simulação.

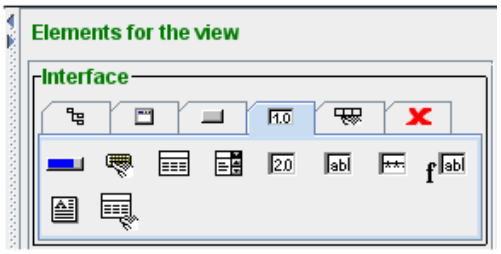


Figura 36 – Ativação da família correspondente a *Input and output*.

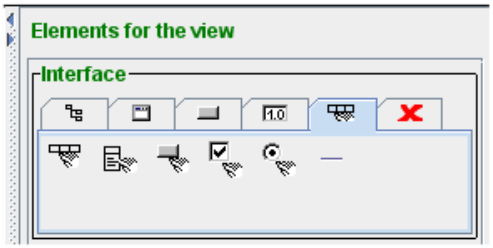


Figura 37 – Ativação da família correspondente a *Menu elements*.

Os elementos básicos podem ser adicionados aos *Containers* do primeiro tipo (descritos no item anterior), mas não podem conter outros elementos.

Analogamente ao que foi descrito anteriormente, uma vez inserido um elemento básico em um container, é possível através do duplo clique do *mouse* ativar a tabela referente à propriedade do elemento, como mostra a Figura 38 referente ao elemento *TwoStateButtom* e efetuar as modificações que o usuário desejar.

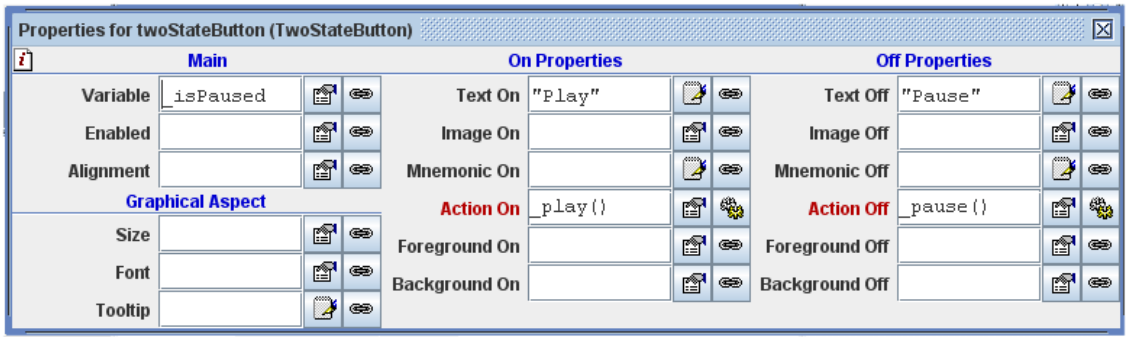


Figura 38 – Tabela referente à propriedade do elemento *TwoStateButtom*.

3.4 ELEMENTOS DE DESENHO – *2D DRAWABLES E 3D DRAWABLES*

O conjunto de elementos de desenhos (*2D Drawables* e *3D Drawables*) consiste de uma série de elementos de visualização que podem ser incluídos em *containers* do segundo tipo com a função de mostrar a animação gráfica e permitir a visualização dos diferentes estados do modelo. Essas animações gráficas podem ser simples ou muito sofisticadas, dependendo do grau de conhecimento de quem está criando a simulação, e incluem partículas (representadas por elipses ou retângulos), setas, imagens, linhas, campos vetoriais, etc. Os elementos de desenho também estão agrupados por funcionalidade. Ativando-se cada aba dessas seções, observa-se o conjunto de elementos correspondentes e selecionado um deles, uma vez inserido em um container da árvore de elementos, é possível ativar a tabela de propriedades e efetuar as alterações desejadas pelo usuário. O procedimento é semelhante ao descrito nas duas seções anteriores.

4 SIMULAÇÕES ELEMENTARES – CASOS DE ESTUDO

4.1 O SISTEMA MASSA-MOLA

O usuário tem condições de iniciar a construção da janela de visualização de sua simulação. Inicialmente ele deve fazer um esboço do que se deseja. É comum acrescentar animação e gráficos em uma mesma janela de visualização. Entretanto, aconselha-se criar janelas diferentes para mostrar ações diferentes, isto é, uma janela de visualização para mostrar a animação do fenômeno e outra (ou outras) para se mostrar o gráfico. De fato podem ser acrescentadas quantas janelas o usuário desejar.

No exemplo do sistema massa-mola, considera-se uma janela para demonstrar a animação e outra o gráfico. A Figura 23, na página 107, mostra a árvore de elementos finalizada. Cumpre-se justificar a escolha e especificar as ações decorrentes dela.

O primeiro elemento selecionado é o *container Frame* que está com a denominação Janela _principal. Como o nome indica, ela é a janela que irá mostrar a animação. Ela está dividida em duas regiões, uma onde fica a animação e outra reservada para se colocar os botões de controle. Deve-se tem em mente que essa disposição foi previamente planejada pelo usuário (autor da simulação). A Figura 39 mostra a tabela das propriedades da Janela_principal juntamente com a tabela correspondente a *Layout* após a sua ativação que é feita ao se clicar sobre o ícone  . O usuário deve se preocupar em observar quais são os parâmetros relacionados a cada uma das tabelas e procurar gravar quais são os efeitos decorrentes das mudanças desses valores.

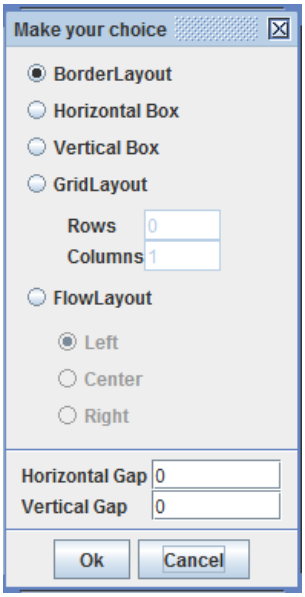


Figura 39 – *Layout* do container *Frame* (*Properties for* Janela_principal)

O próximo passo é inserir no *container* anterior outro relacionado com desenho (*DrawingPanel*). Para isso, seleciona-se o ícone correspondente, ativando-o, e através da varinha mágica, clica-se sobre o ícone Janela_principal que já está na coluna *Tree of elements*. Fornecemos um nome, no caso, Painel_de_desenho, e clica-se em OK. Um duplo clique sobre o elemento criado e, novamente, aparece uma tabela com as propriedades que o usuário irá modificar. Muitas das propriedades mostradas são padronizadas (Figura 40).

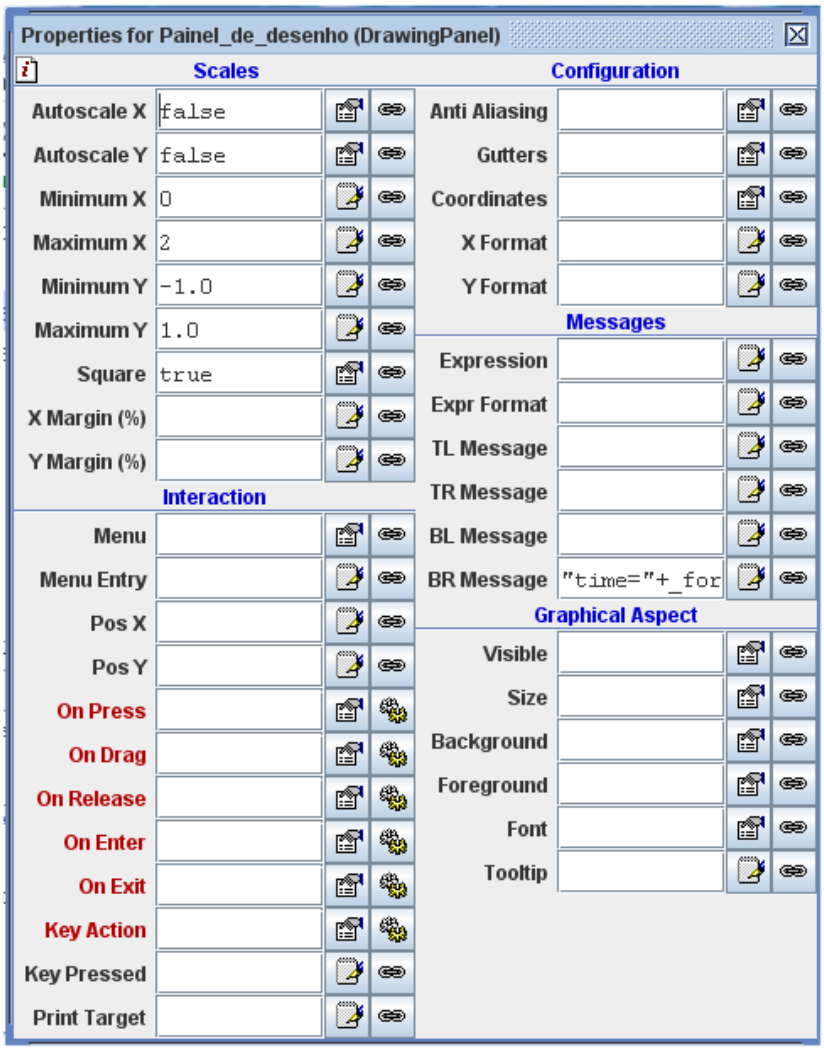


Figura 40 – Propriedades do Painel_de_desenho.

Neste painel são colocados os elementos de desenho que fazem parte da animação. Deseja-se mostrar simplesmente uma mola, uma pequena esfera presa em uma de suas extremidades e a outra extremidade da mola presa a uma parede. Todos esses elementos são fornecidos pelo EJS. Basta efetuar as mesmas operações descritas anteriormente, isto é, selecionando-se o elemento desejando e "depositando-o" sobre o contêiner Painel_de_desenho. A Figura 23 orienta o leitor quanto à disposição dos elementos inseridos, enquanto as Figuras 41, 42 e 43 fornecem as tabelas de propriedades relativas aos elementos Parede_em_2D, ora mencionados.

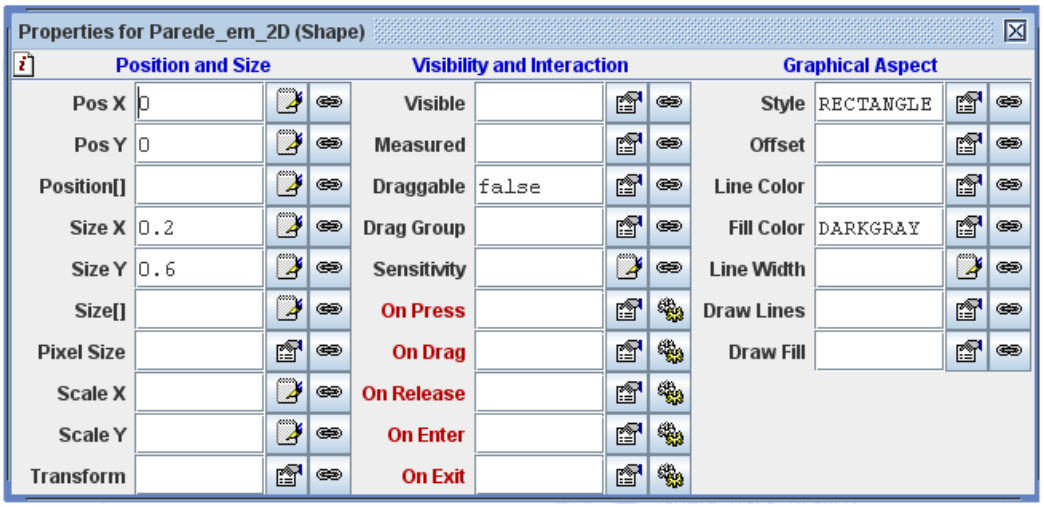


Figura 41 – Propriedades do elemento de desenho Parede_em_2D.

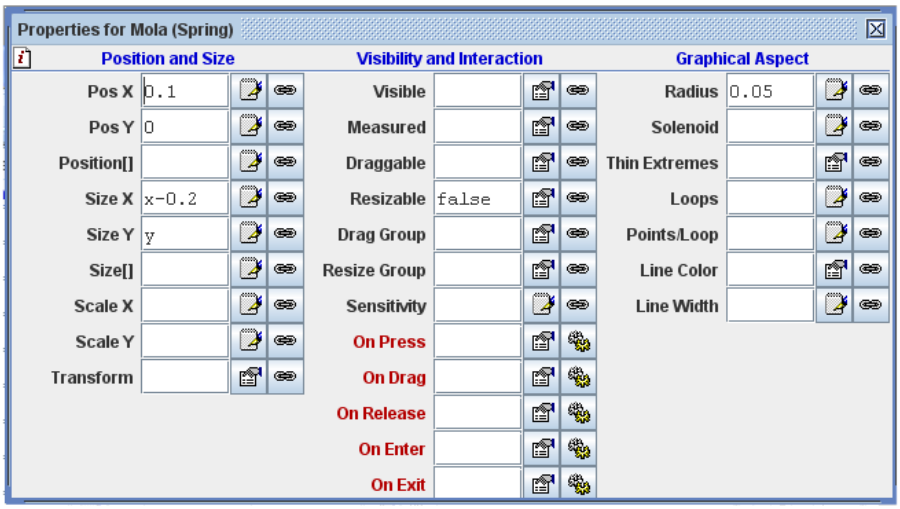


Figura 42 – Propriedades do elemento de desenho Mola.

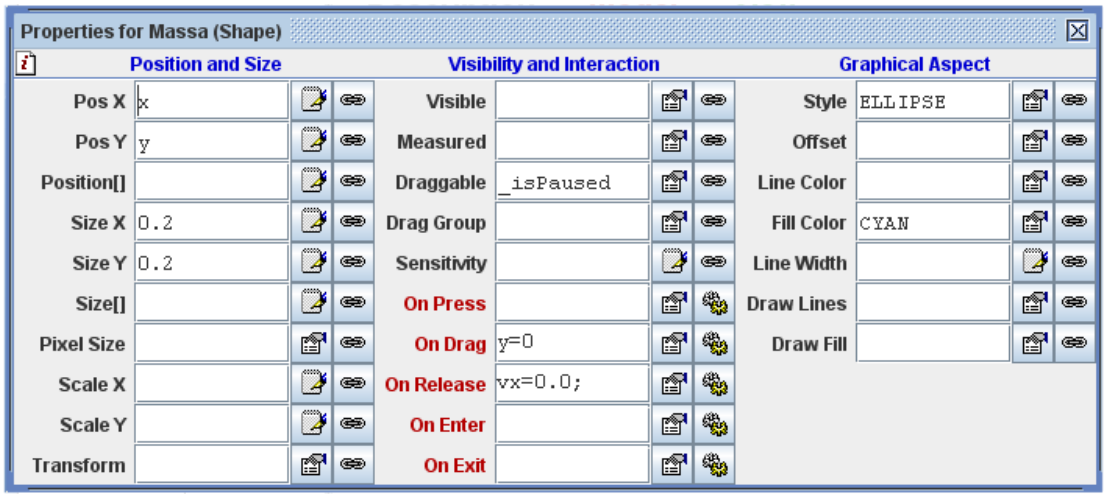


Figura 43 – Propriedades do elemento de desenho Massa.

A terceira etapa da tarefa consiste em criar o contêiner Painel_de_controle. Este contêiner irá guardar outros *containers* onde serão colocados elementos que permitem a interação do usuário com a simulação. O processo de inserção dos elementos é análogo ao descrito nas seções anteriores e, de agora em diante, não se repete mais essa etapa de descrição.

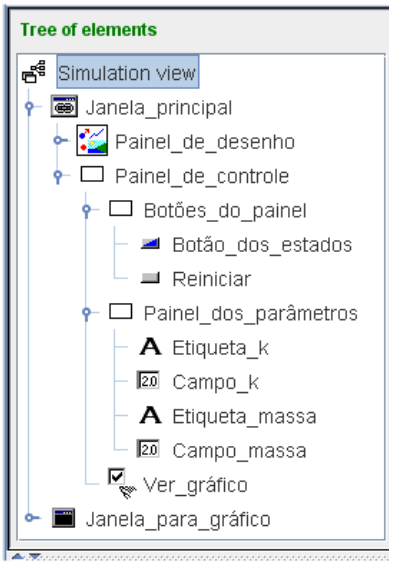


Figura 44 – Destaque dos elementos do Painel_de_controle da Figura 23.

Tomando-se como referência a Figura 23 (p. 107), observa-se a existência de dois outros *containers*: Botões_do_painel e Painel_de_parâmetros. Ao clicar nos círculos que estão do lado esquerdo dos correspondentes ícones, observamos os respectivos "filhos" , mostrado em destaque na Figura 44.

Duplo clique sobre Painel_de_controle para se ter a visualização de suas propriedades (Figura 45). Procedimento semelhante sobre os "filhos" como mostram as Figuras 46 e 47, respectivamente. Atente-se para os valores dos parâmetros.

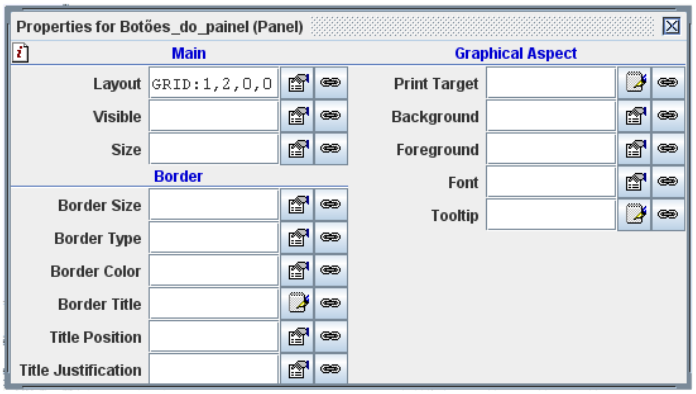


Figura 45 – Propriedades do *container* Botões_do_painel.

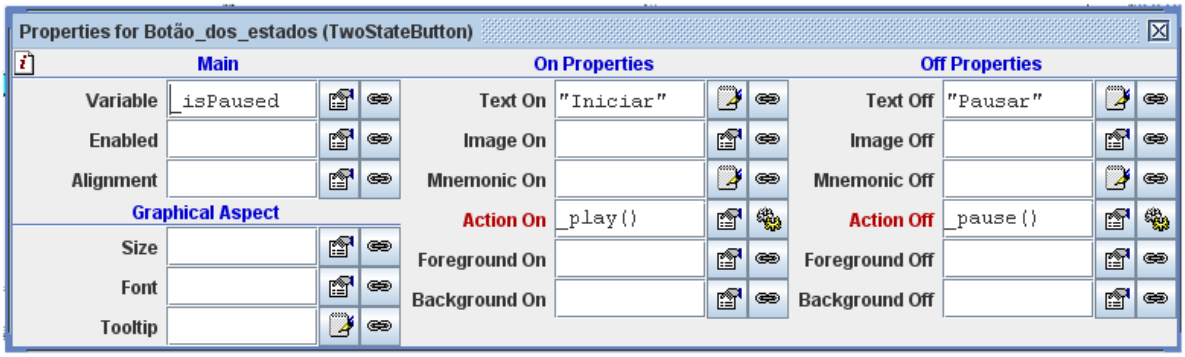


Figura 46 – Propriedades do Botão_dos_estados.

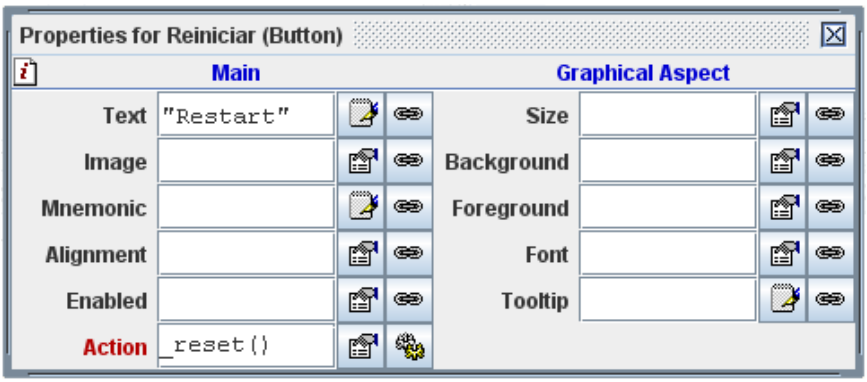


Figura 47 – Propriedades de Reiniciar.

Duplo clique sobre Painel_dos_parâmetros para se ter a visualização de suas propriedades (Figura 48) e para os seus "filhos", Figuras 49, 50, 51 e 52, respectivamente. Novamente, atente-se para os valores dos parâmetros.

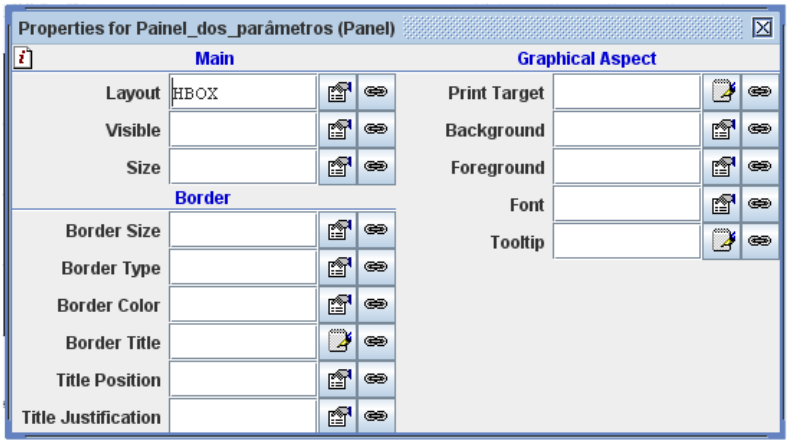


Figura 48 – Propriedades do *container* Painel_dos_parâmetros.

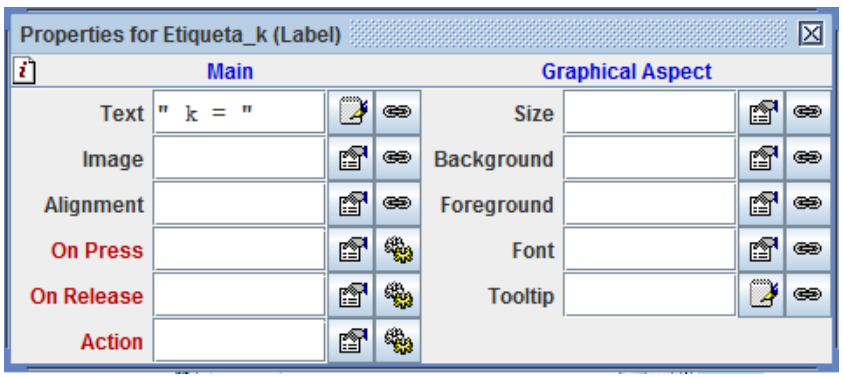


Figura 49 – Propriedades de Etiqueta_k.

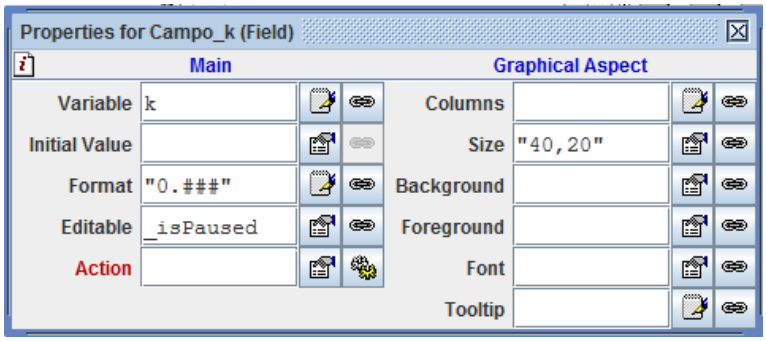


Figura 50 – Propriedades de Campo_k.

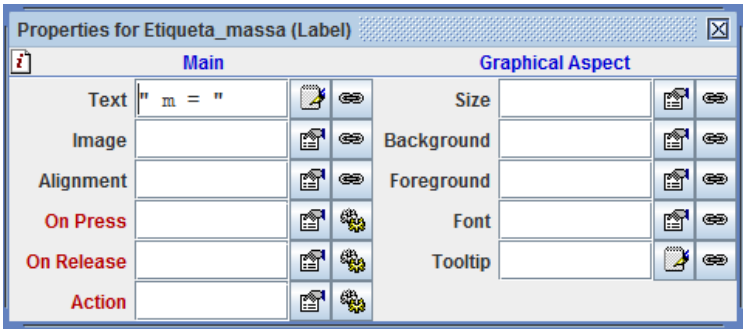


Figura 51 – Propriedades de Etiqueta_massa.

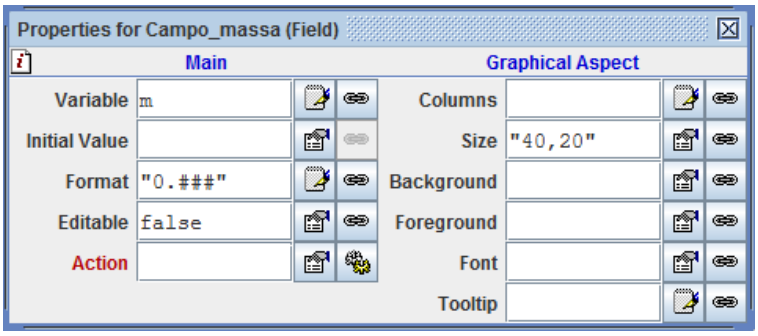


Figura 52 – Propriedades de Campo_massa.

Para completar essa parte, acrescenta-se o Ver_gráfico, que serve para ativar o gráfico que será construído durante a execução da simulação. A Figura 53 mostra o conjunto de propriedades correspondente.

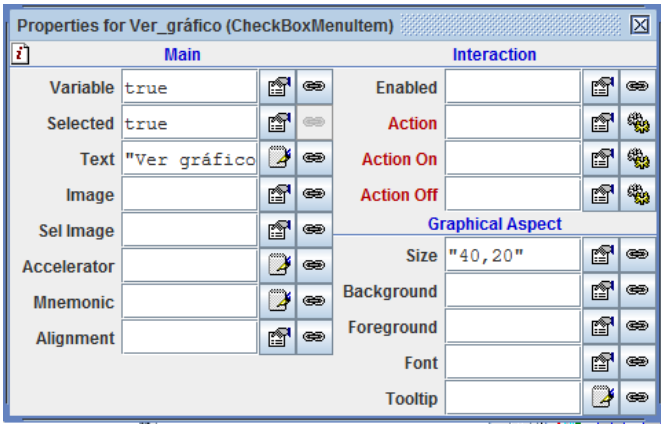


Figura 53 – Propriedades de Ver_gráfico.

O resultado final de toda essa ação é mostrado na Figura 54, isto é, a janela de visualização principal da simulação. O usuário pode acionar a simulação através da tecla "Iniciar", pausar a simulação, recomeçar através da mudança da posição da esfera, mostrando dessa forma a interação entre o usuário e a simulação, alterar os valores de k e de m.

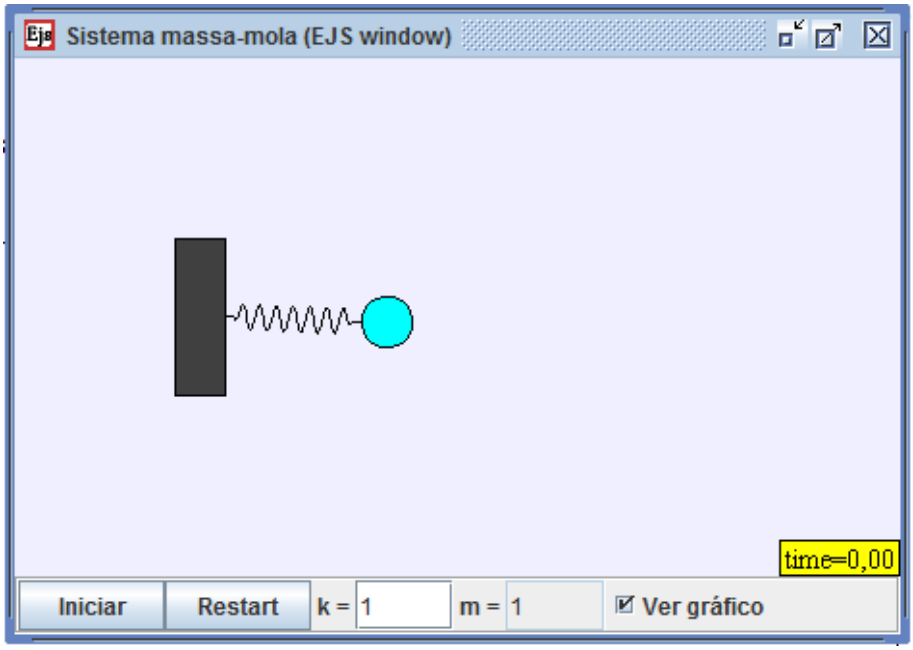


Figura 54 – Janela de visualização principal do EJS para a simulação do Sistema massa-mola.

Mas, deseja-se visualizar, também, a evolução da posição e a variação da velocidade do corpo em função do tempo. Isso é possível através da criação de uma nova janela, cujo efeito é permitir a visualização de gráficos. Dessa forma, cria-se uma janela para essa função específica. Novamente, recorre-se a Figura 23. Na coluna *Tree of Elements*, observa-se o container *View* com o título Janela_para_gráfico. Adiciona-se o container *PlottingPanel*, com o título Painel_com_eixos, que permite a EJS criar um sistema de eixos cartesianos.

A Figura 55 mostra a tabela de propriedades da Janela_para_gráfico e a Figura a 56 as propriedades para Painel_com_eixos. Falta agora "informar" a EJS que deve ser representado no Painel_com_eixos o gráfico da posição da extremidade livre da mola com e tempo e a velocidade do corpo preso a mola em função do tempo. Isso é feito ao selecionar o elemento *Trai*l. Observa-se na Figura 23, novamente, que este elemento aparece duas vezes: um para indicar a curva referente à posição em função do tempo e a outra a velocidade a velocidade em função do tempo. As Figuras 57 e 58 mostram as correspondentes propriedades dos elementos Posição e Velocidade.

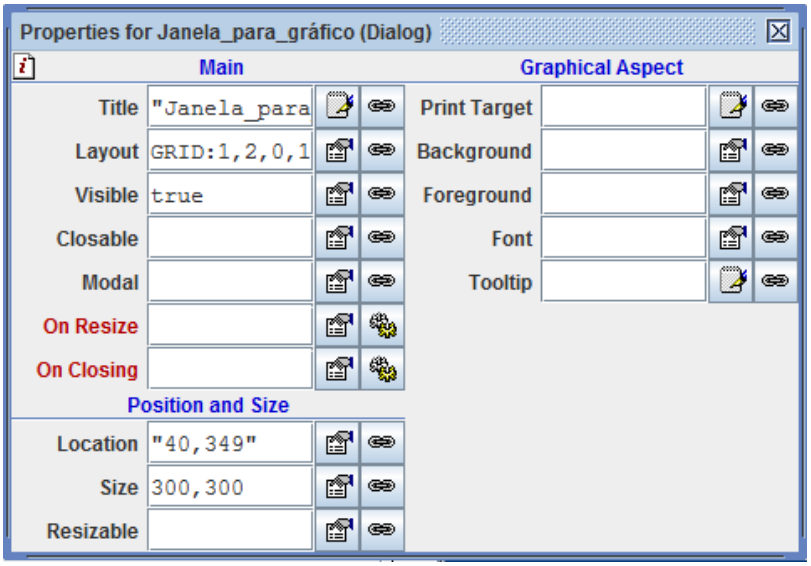


Figura 55 – Propriedades de Janela_para_gráfico.

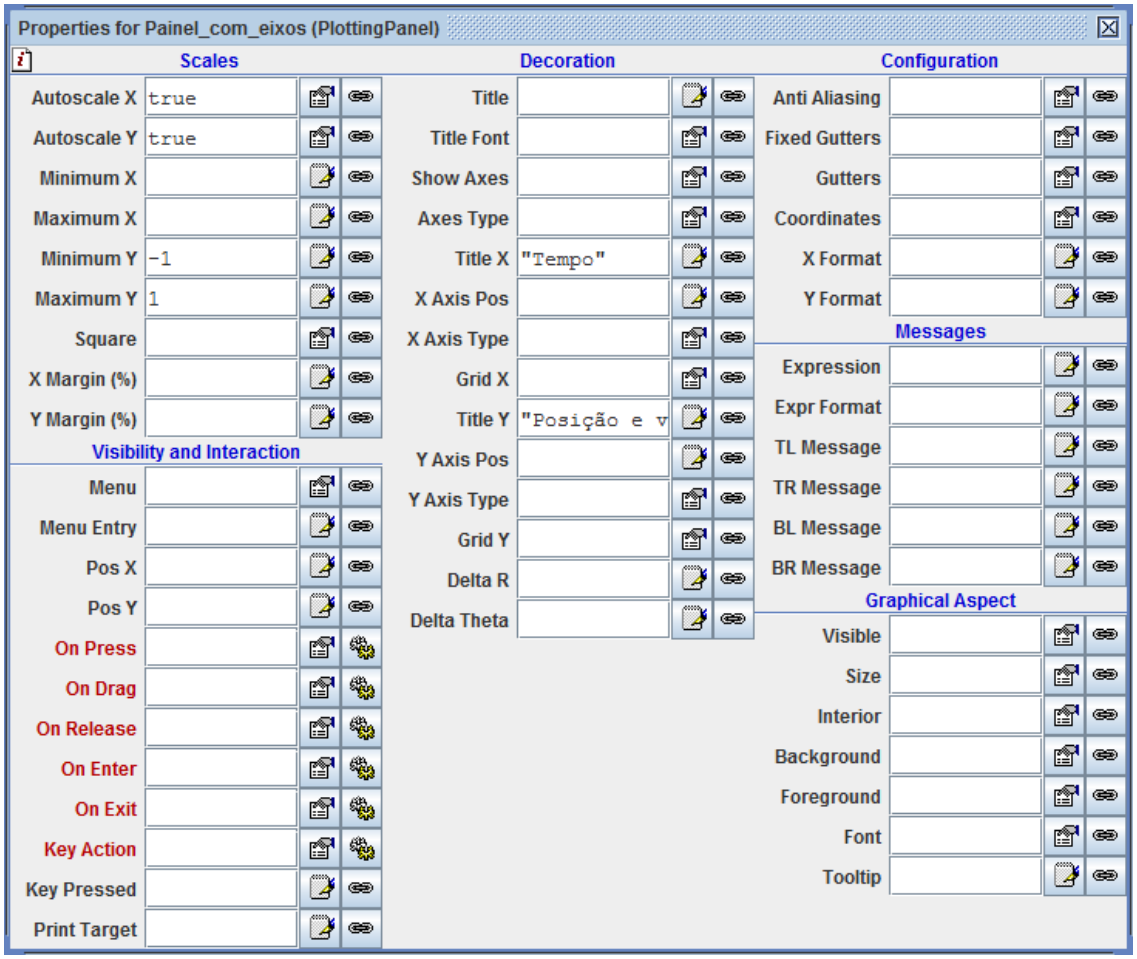


Figura 56 – Propriedades de Painel_com_eixos.

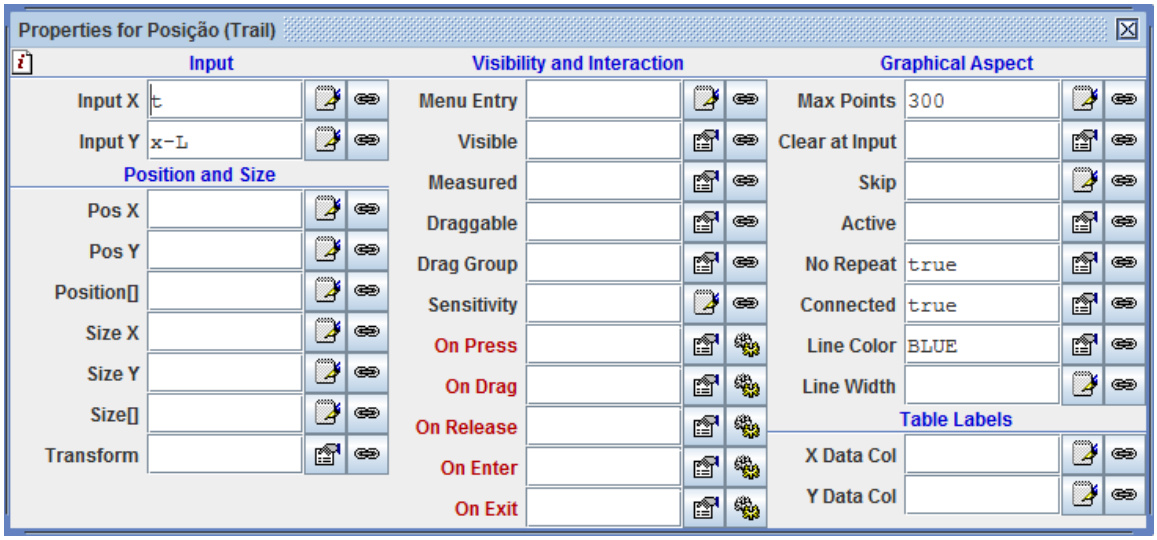


Figura 57 – Propriedades de Posição (*Trail*).

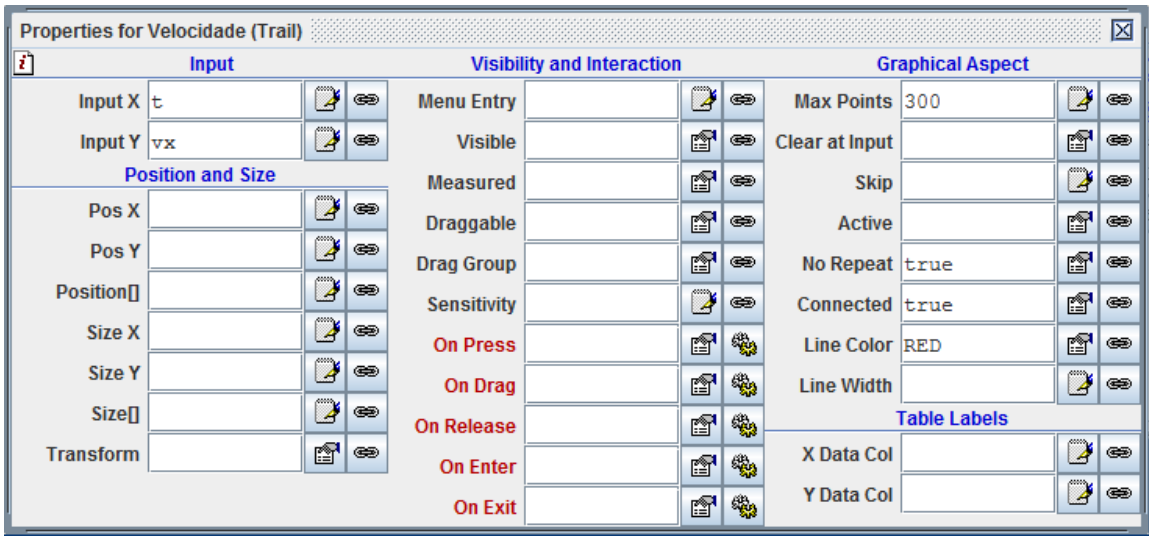


Figura 58 – Propriedades de Velocidade (*Trail*).

Finalizada essa etapa, tem-se a janela de visualização correspondente ao gráfico, Figura 59. A preparação da simulação, de acordo com a visão de quem a elaborou, está pronta.

Para se executar a simulação, clica-se no ícone da barra de tarefas. O EJS começa o processo de compilação e, se tudo estiver correto, após alguns instantes, tem-se a tela correspondente a Figura 59 com uma mensagem dizendo que a compilação foi bem sucedida. Caso contrário, o usuário deverá verificar o que diz a mensagem e tentar resolver o problema. Em geral é uma vírgula que aparece no lugar de um ponto, um tipo de variável, uma alteração de uma determinada propriedade e assim por diante.

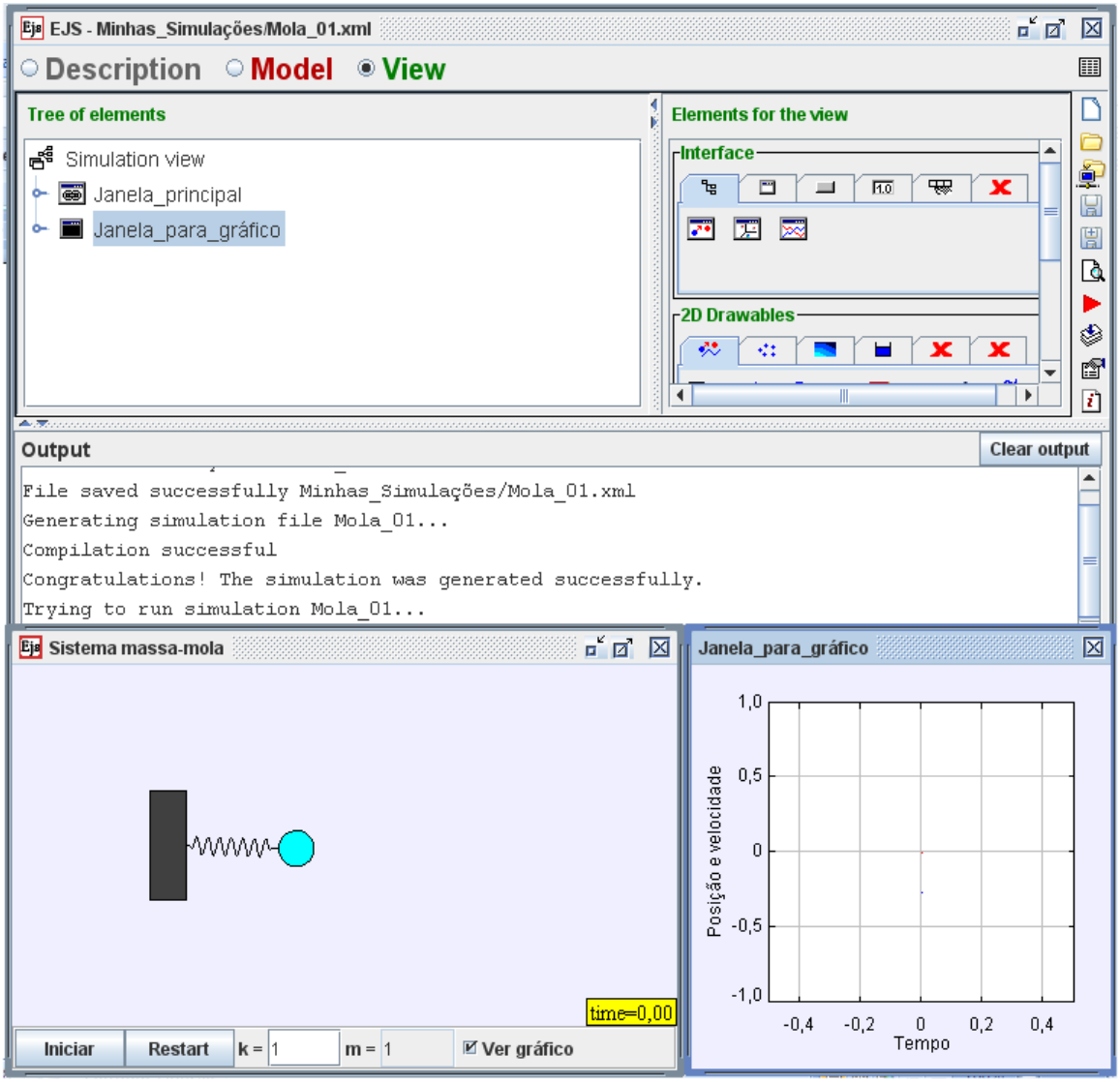


Figura 59 – Estágio inicial da simulação para o sistema massa-mola.

Observe na Figura 59, no campo *Output*, tem-se a mensagem "*Trying to run simulation Mola 01...*" O usuário deve iniciar a simulação clicando diretamente no botão "Iniciar". Tem-se a movimentação da esfera presa na extremidade da mola, ao mesmo tem que se processa o desenrolar do traçado dos gráficos de Posição e velocidade.

É importante salientar que a forma como a simulação ocorre é fruto da ideia de quem a preparou. Por exemplo, é possível obter duas janelas distintas para as representações gráficas. Basta repetir a última etapa do processo que foi descrito duas vezes!

A Figura 60 mostra o estado do sistema massa-mola no instante 12,20 s, obtido através da tecla *Print Screen*, após ter-se iniciado o processo. Chamamos a atenção para as curvas correspondentes para a posição (em azul) e velocidade (em vermelho).

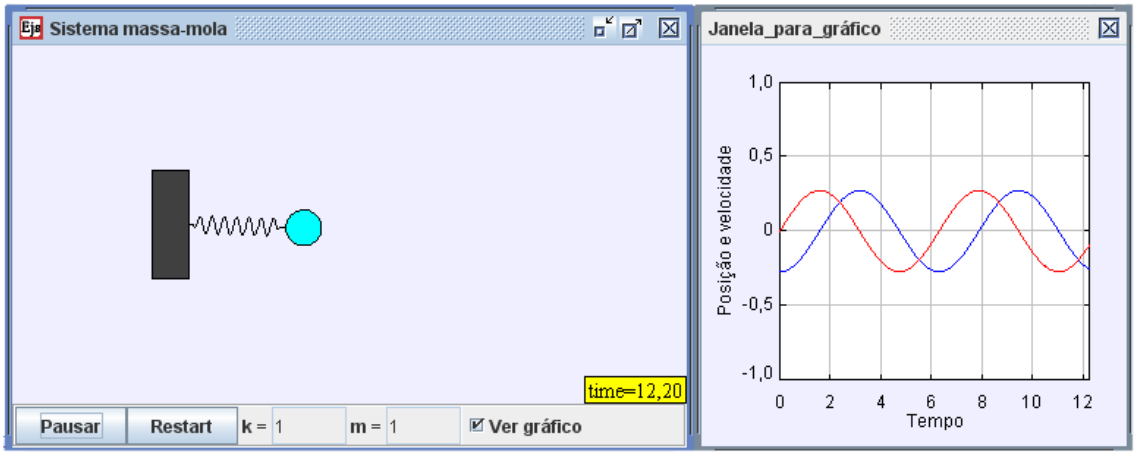


Figura 60 – Estado do sistema massa-mola para o instante 12,20 s.

Nos próximos exemplos, serão apresentados os painéis referentes a algumas simulações simples que desenvolvemos. O objetivo dessas simulações é o de mostrar ao usuário algumas das ferramentas de EJS. Como dissemos anteriormente, esse manual de procedimentos deve ser visto como uma introdução ao EJS e que será complementado com o tempo.

4.2 CONVERSÃO DE TEMPERATURAS

Essa simulação tem o objetivo de mostrar ao usuário como escrever um código em *Java* no painel *Fixed relations*. As grandezas envolvidas nessa simulação não dependem do tempo e, dessa forma, não há necessidade de escrever uma página de códigos no painel *Evolution*.

Alertamos que a escolha do tema, Conversão de temperaturas, tem o propósito de mostrar ao usuário como proceder para obter uma janela de visualização com botões que possam ser selecionados e, assim, obter uma resposta específica. Nessa simulação são escritas as equações de conversão entre os pares de escalas Celsius-Fahrenheit, Celsius-Kelvin e Fahrenheit-Kelvin.

O procedimento de preparação da simulação é basicamente o mesmo daquele descrito da seção anterior. O usuário deve, nessa fase de aprendizagem, ativar cada um dos painéis descritos anteriormente e verificar a lógica utilizada na elaboração dessa simulação. Assim, as próximas figuras mostrarão as etapas seguidas na preparação da simulação.

A Figura 61 mostra o painel *Description* ativado e a mensagem com o objetivo da simulação.

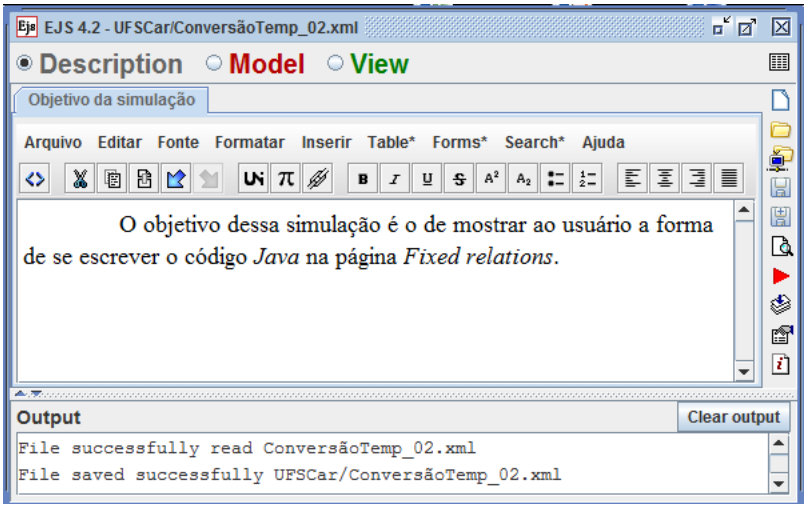


Figura 61 – Objetivo da simulação Conversão_Temp02.xml.

A segunda etapa é declarar as variáveis que intervêm no modelo, atribuir os correspondentes valores iniciais e o tipo de variável. Toda essa etapa é mostrada na Figura 62.

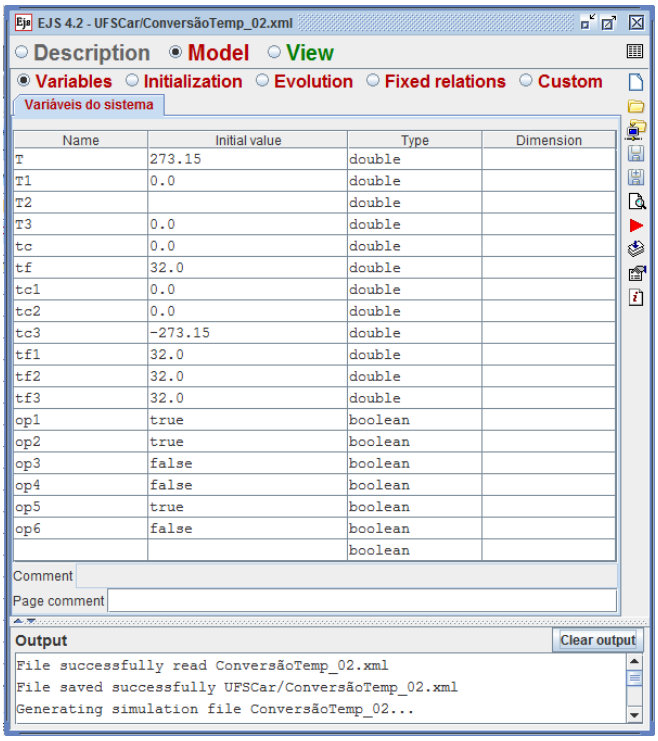


Figura 62 – Declaração das variáveis para a simulação Conversão_Temp02.xml.

A próxima etapa consiste em escrever o código em *Java* com as equações de conversão entre as grandezas e os comandos de seleção (condição *if*), como mostra a Figura 63. A condição *if* permitirá o usuário escolher o tipo de conversão entre as escalas que ele deseja.

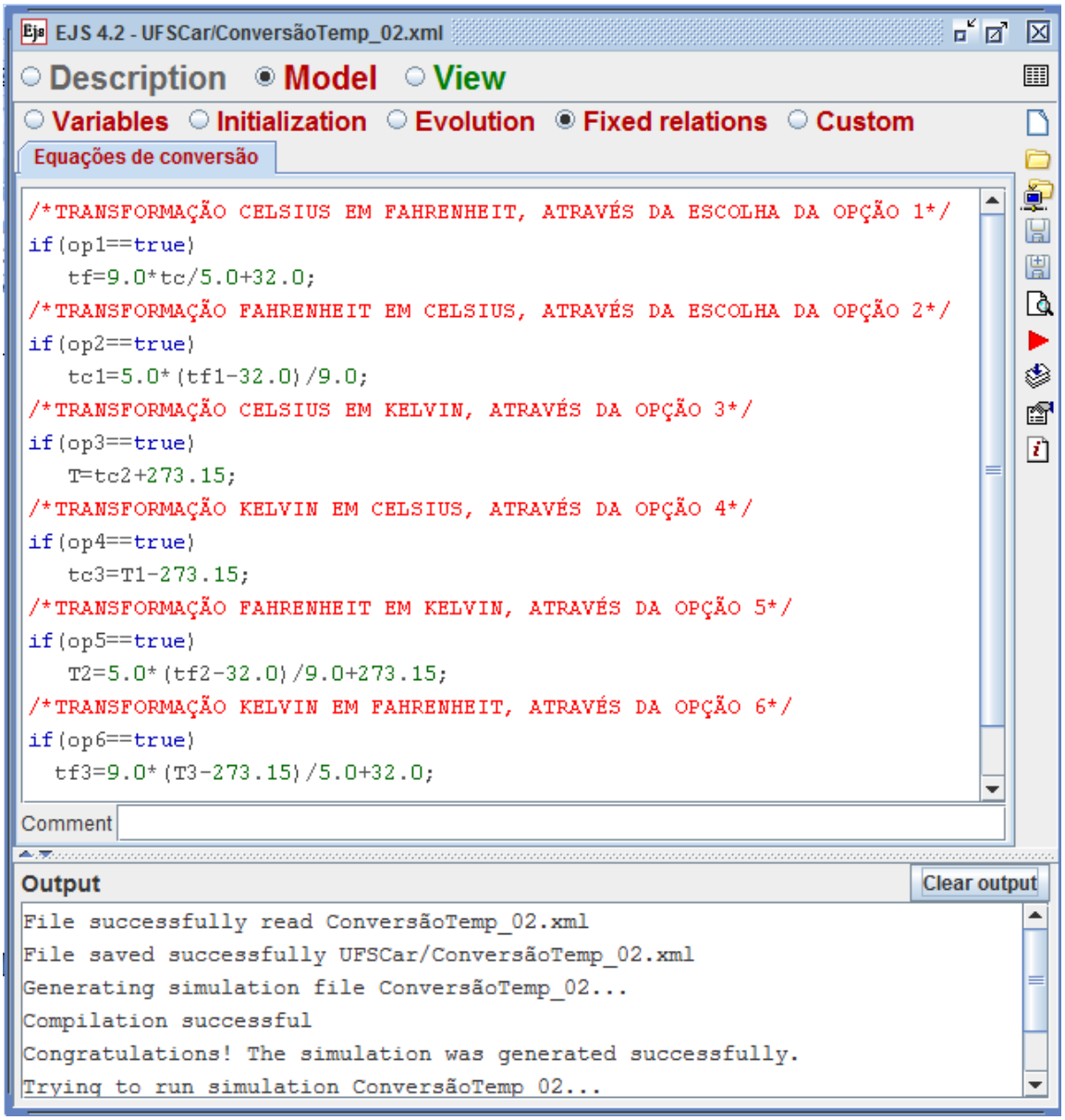


Figura 63 – A página *Fixed relations* com o código em *Java*.

Finalizadas as etapas relacionadas à modelagem, o usuário deve preparar a árvore dos elementos. Isso permite obter “aquilo que se quer ver” na tela do computador. Para isso, entretanto, o usuário segue um esquema (esboço), previamente preparado, contendo a disposição final dos elementos (botões, caixas para mensagens, etc.) na janela de visualização. Toda a árvore de elementos para essa simulação é mostrada na Figura 64.

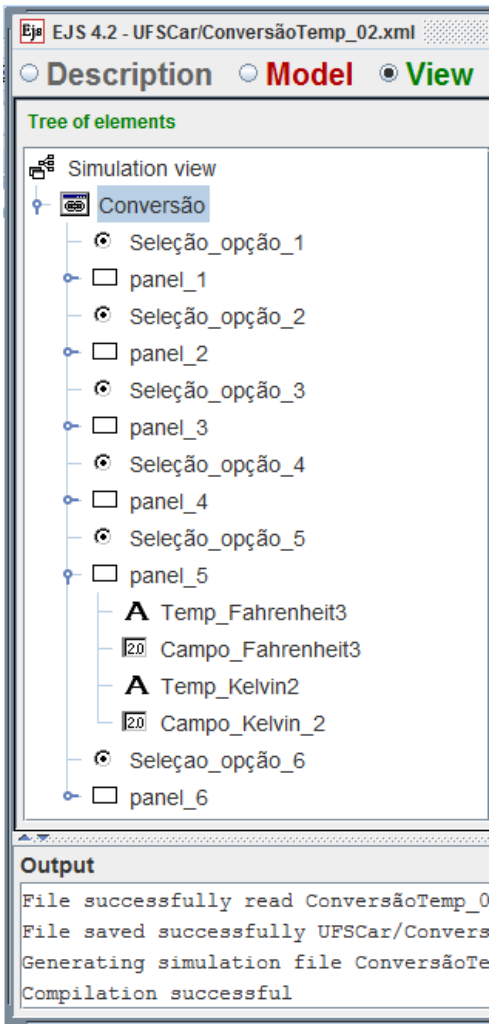


Figura 64 – A estrutura da árvore dos elementos da simulação Conversão_Temp02.xml.

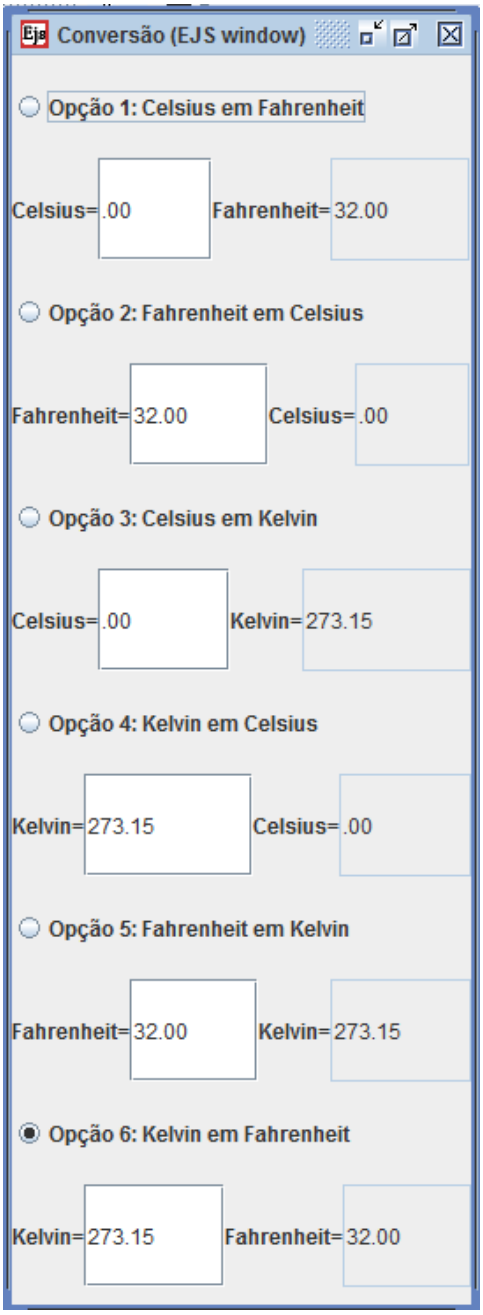


Figura 65 – A janela de visualização final.

A Figura 65 mostra a janela de visualização final da simulação. O usuário deve inicialmente escolher uma opção de transformação entre escalas termométricas. Ao digitar o valor numérico no campo que aparece em branco e apertar a tecla *"enter"*, imediatamente aparece o valor correspondente.

Acreditamos que o valor pedagógico dessa simulação é pobre, mas serve para mostrar a estrutura de confecção em relação à escolha dos elementos que devem figurar na janela de visualização e, também, como relacioná-los.

4.3 QUEDA DE UM CORPO EM MEIO RESISTIVO

O objetivo dessa simulação é demonstrar ao usuário o potencial do EJS na confecção de gráficos. Todas as etapas efetuadas na sua preparação são semelhantes àquelas descritas nas seções anteriores e as próximas figuras representam essas etapas.

Primeira etapa: A página "Objetivo da simulação" escrita no painel *Description*. A EDO apresentada foi escrita com o uso do *MathType* e salva como figura no formato *gif*. A seguir, a figura foi importada para a pasta onde está o arquivo "Resistência_ar_01.xml". A Figura 66 mostra a configuração final da página.

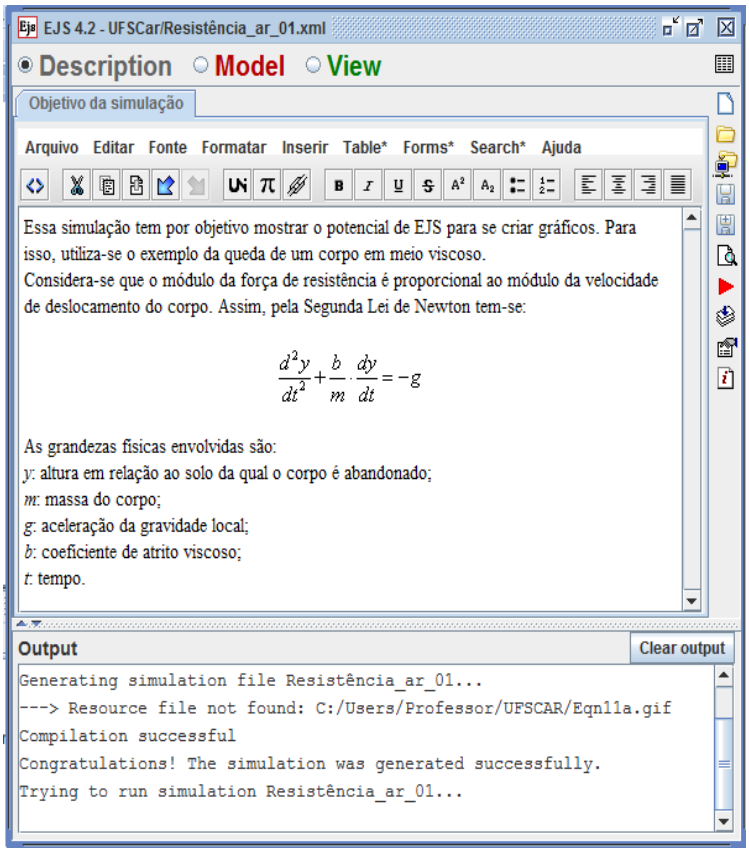


Figura 66 – Objetivo da simulação Resistência_ar_01.xml.

Segunda etapa: A Figura 67 mostra a página no painel *Variables* com as grandezas físicas, os valores iniciais e o tipo. Observa-se que todos os valores iniciais já estão declarados. Dessa forma, não há necessidade de se escrever uma página de códigos *Java* no painel *Initialization*.

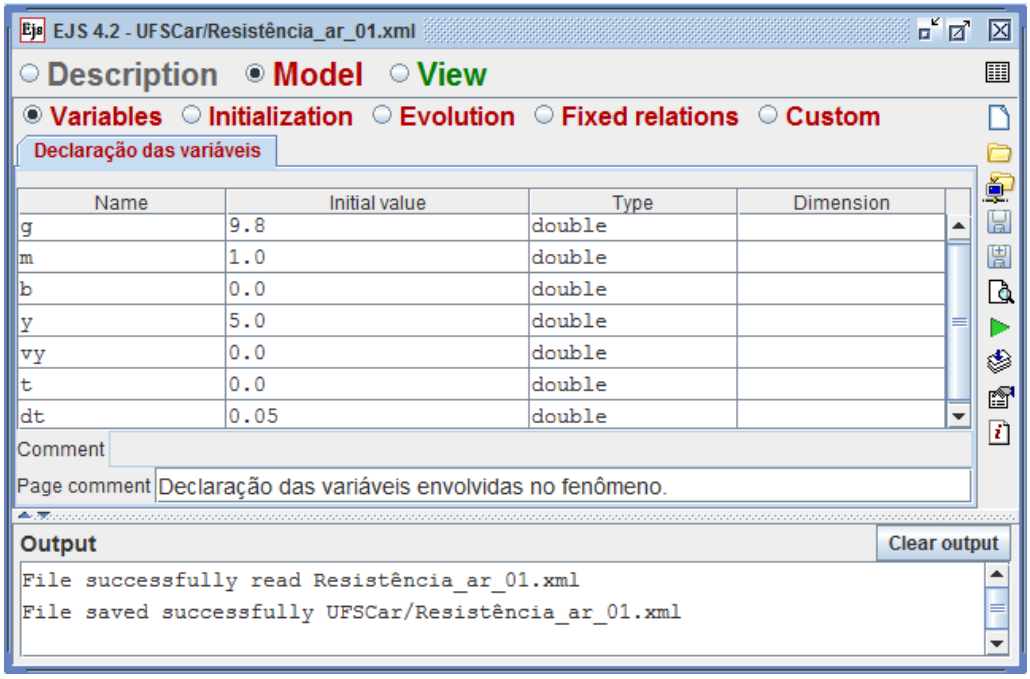


Figura 67 – Declaração das variáveis.

Terceira etapa: Escrever as equações diferenciais numa página do painel *Evolution*, mostrada na Figura 68.

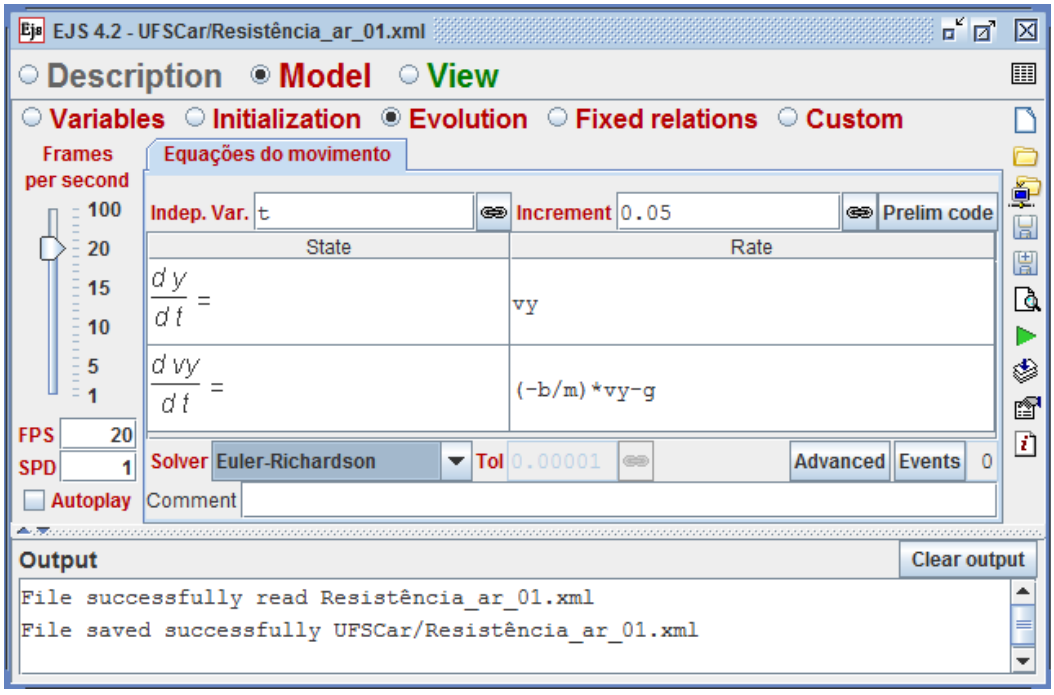


Figura 68 – Equações diferenciais e o método numérico de Euler-Richardson utilizados na modelagem.

Quarta etapa: Uma página “Relações fixas” é escrita no painel *Fixed relations* com o objetivo de fazer parar a evolução da simulação. Observa-se na Figura 69 a linha de comando com a condição *if*.

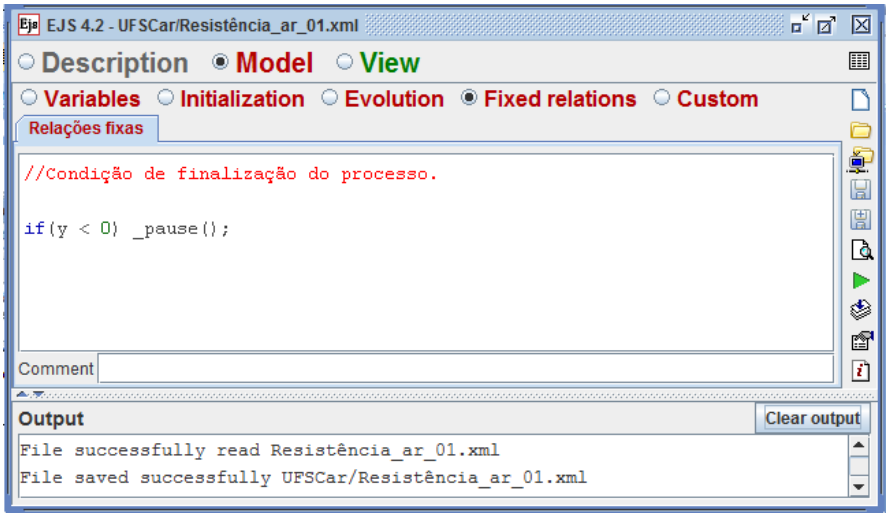


Figura 69 – Janela em que se descreve a condição para a simulação finalizar.

Quinta etapa: A Figura 70 mostra a árvore de elementos.

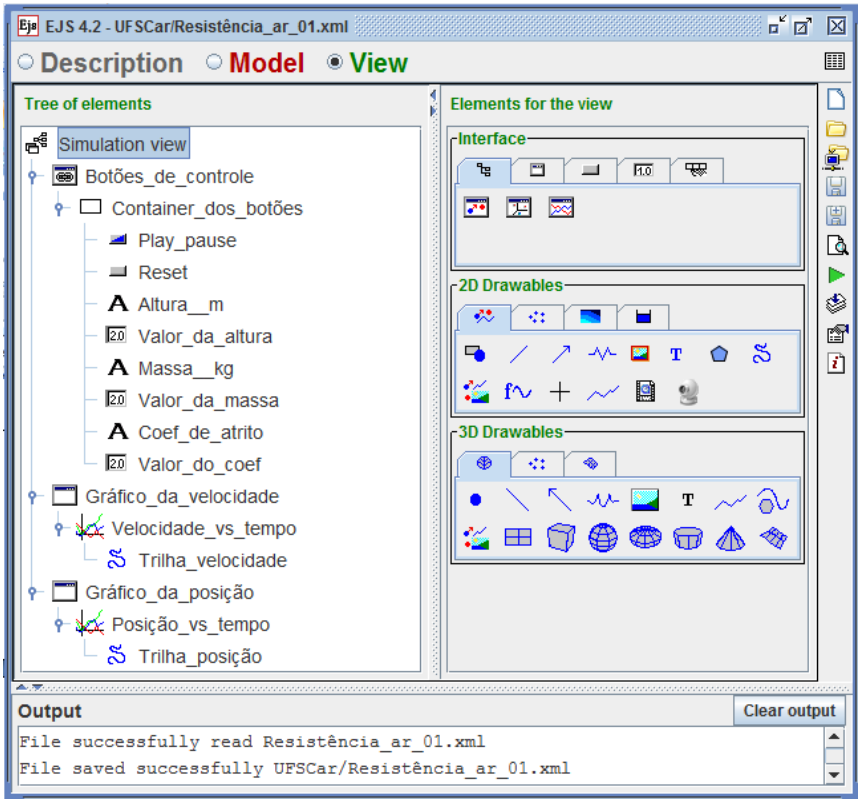


Figura 70 – A estrutura da árvore de elementos da simulação.

A Figura 71 mostra o resultado final e no estado inicial da simulação.

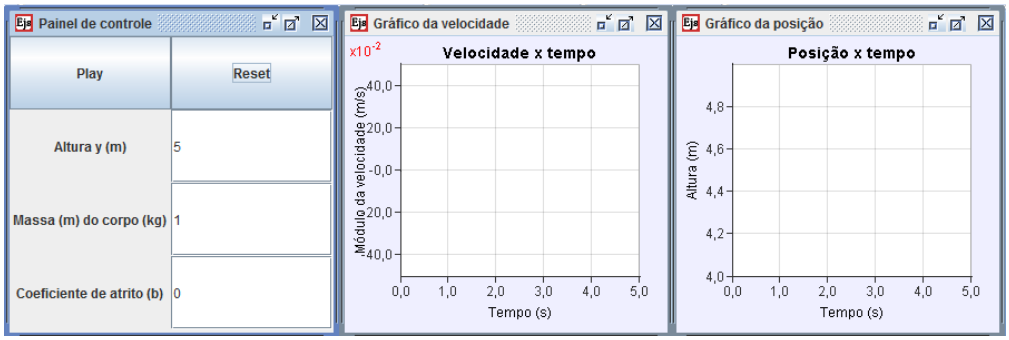


Figura 71 – A janela de visualização mostrando o estado inicial.

A Figura 72 mostra o resultado final da simulação, desprezando-se a resistência do meio.

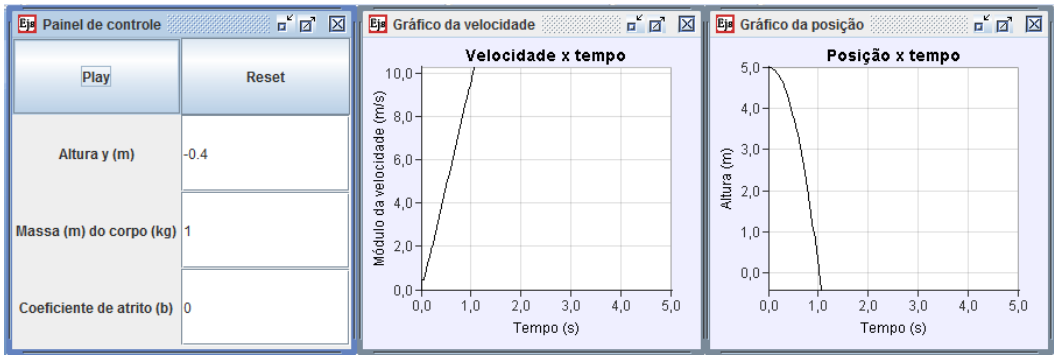


Figura 72 – Estado final da simulação para o corpo em queda livre na ausência de atrito viscoso.

A Figura 73 mostra o resultado final da simulação do corpo em queda no meio que oferece uma resistência ao movimento.

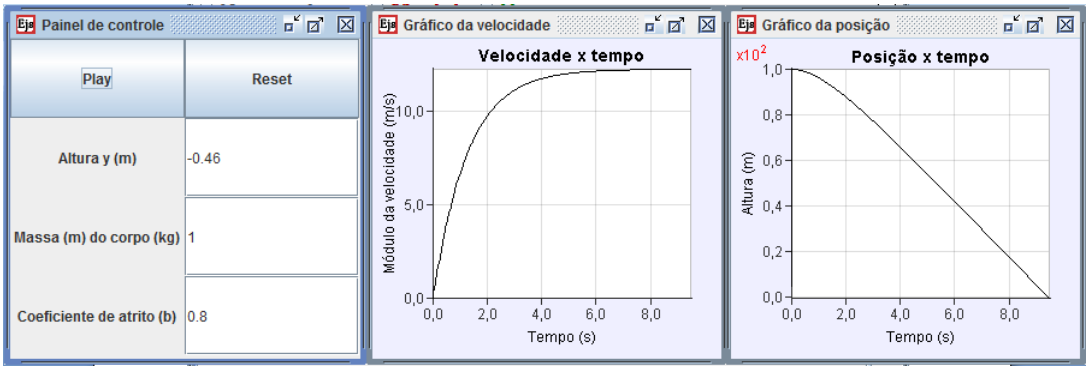


Figura 73 – Simulação para o corpo em queda num meio resistivo (estado final).

4.4 MOVIMENTO EM TRÊS DIMENSÕES

Outra potencialidade do EJS está relacionada à visualização 3D. A próxima simulação mostra como confeccionar gráficos em 3D. Particularmente, partindo-se das equações paramétricas do movimento de uma partícula do espaço, o EJS faz a representação da trajetória dessa partícula em 3D. Todas as etapas descritas anteriormente são, novamente, repetidas. Dessa forma, apenas apresentamos, a seguir, as figuras correspondentes.

A página "Objetivo da simulação" no painel *Description* mostrada na Figura 74.

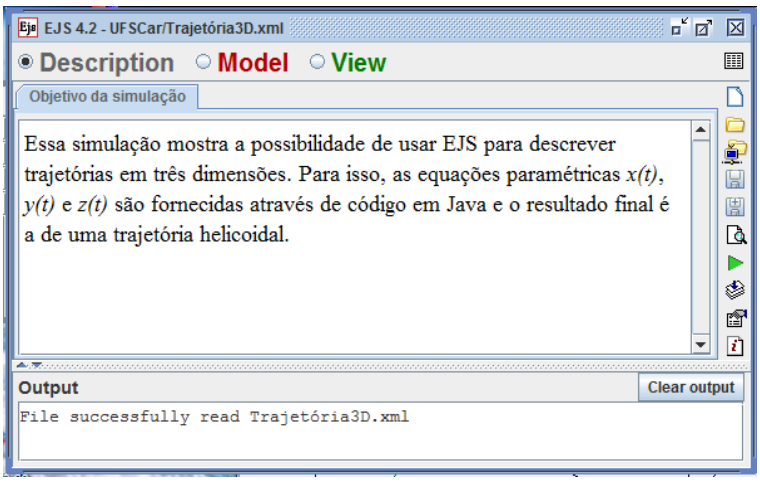


Figura 74 – Página *Description* para a simulação da trajetória helicoidal.

Na Figura 75 tem-se a declaração das variáveis, atribuição dos valores iniciais e o tipo de variável. Observa-se que as variáveis *x*, *y* e *z* são declaradas, mas não "inicializadas".

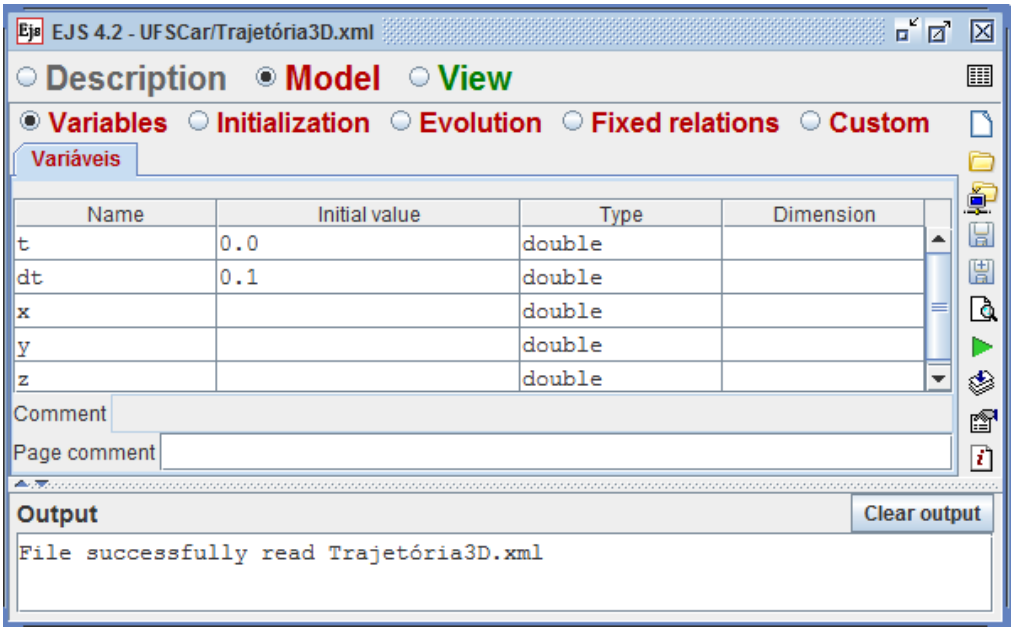


Figura 75 – Declaração das variáveis a modelagem da trajetória 3D.

A Figura 76 mostra o código em *Java* para a atribuição dos valores iniciais, para $t = 0$ *s*, das grandezas *x*, *y* e *z*.

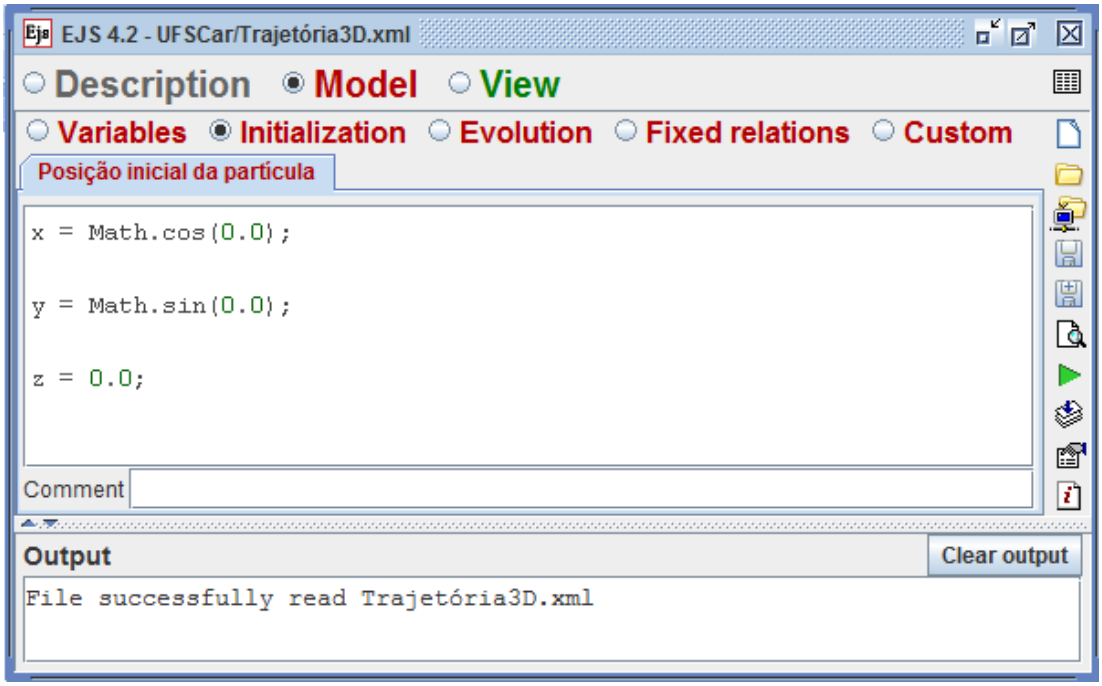


Figura 76 – Inicialização das coordenadas de posição da partícula no instante t = 0 s.

O código em Java para a evolução do tempo é mostrado na Figura 77.

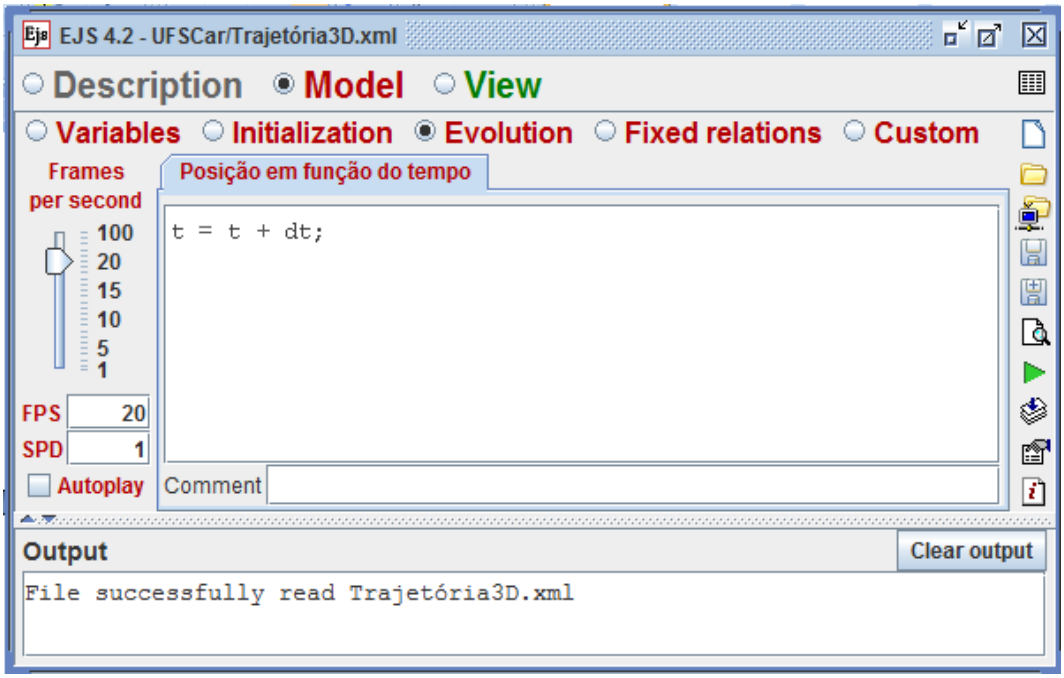


Figura 77 – Código *Java* para representar a evolução da variável tempo.

A Figura 78 mostra o código Java da página “Posição da partícula” no painel *Fixed relations*.

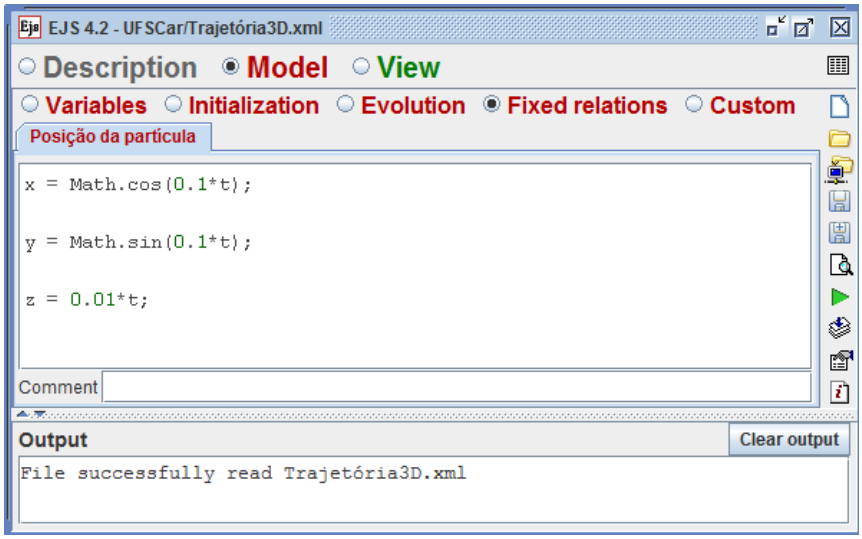


Figura 78 – Código *Java* para demonstrar a evolução da posição em relação ao tempo.

A árvore dos elementos é mostrada na Figura 79. A Figura 80 e a Figura 81 representam, respectivamente, o painel de controle da simulação e a janela de visualização onde será mostrada a trajetória da partícula.

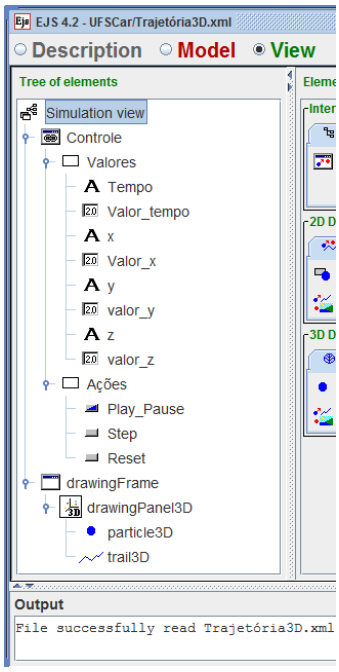


Figura 79 – A estrutura da árvore dos elementos para a simulação Trajetória3D.xml.

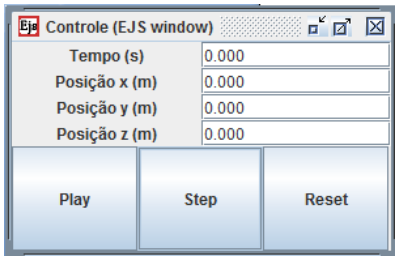


Figura 80 – O painel de controle.

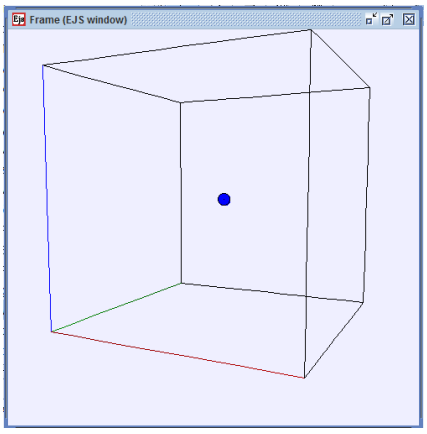


Figura 81 – A janela de visualização 3D.

A Figura 82 mostra o resultado final da simulação, isto é, a trajetória helicoidal descrita pela partícula.

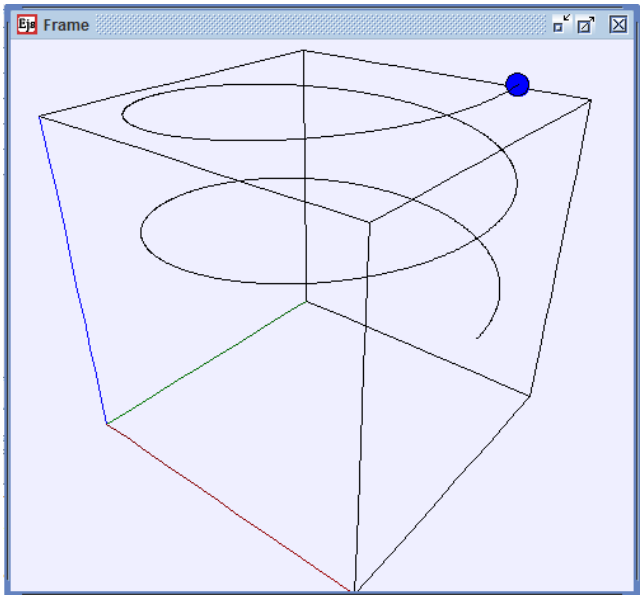


Figura 82 – Trajetória helicoidal descrita por uma partícula.

5 O BÁSICO DA LINGUAGEM *JAVA*

5.1 DECLARAÇÃO E TIPOS DE VARIÁVEIS

Para criar uma variável é necessário fornecer um único nome, especificar o tipo de variável e a sua dimensão, caso a variável represente um vetor ou uma matriz.

Normalmente, ao se modelar um fenômeno físico através de fórmulas matemáticas, as variáveis assumem valores reais (ocasionalmente números complexos). Num programa de computador é necessário distinguir os diferentes tipos de variáveis e, dependendo do tipo, o computador irá reservar diferentes áreas de memória.

A linguagem *Java* utiliza os seguintes tipos de variáveis básicas: *boolean, byte, char, short, int, long, float, double* e *string*. Nesse trabalho, entretanto, utiliza-se as variáveis *int, double, boolean* e *string*. A seguir é apresentada a descrição dessas variáveis usadas nas simulações que foram apresentadas.

a. *boolean*: A variável booleana (*boolean*) pode assumir os valores *true* (verdadeiro) ou *false* (falso).
b. *int*: A variável *int* representa um número inteiro.
c. *double*: Essa variável representa números reais de dupla precisão.
d. *string*: A variável string é usada para representar um texto.

Variáveis do tipo *integer* (inteiras) utilizam menos memória e permitem processamentos mais rápidos, pois muitos sistemas computacionais apresentam implementações otimizadas de rotinas para a aritmética dos números inteiros.

Para economizar espaço na memória e otimizar a velocidade de execução é aconselhável utilizar os tipos padronizados em cada categoria. O EJS utiliza apenas as seguintes variáveis: *boolean, int, double* e *string*, como mencionado acima.

Além dos tipos citados acima, a linguagem *Java* utiliza um tipo denominado de *objeto* (object) para uma família de variáveis avançadas. *Java* é uma linguagem de programação orientada a objeto que permite aos programadores experientes criarem um mundo virtual de novas construções de variáveis que são denominadas de *class* (classes).

O EJS pretende ser uma ferramenta fácil para uma questão específica, permitindo àquele *non-expert* em programação em *Java* criar um modelo focalizado em simulações científicas. Entretanto, EJS introduz a possibilidade de se declarar as variáveis do tipo objeto como uma porta para programadores avançados usarem e criarem novos truques de visualização.

5.2 CRIANDO VARIÁVEIS COM O EJS

A página do painel de variáveis (*Variables*) mantém uma cópia do editor de variáveis que adota a forma de uma tabela. Cada variável é adicionada na tabela através da digitação do nome escolhido apropriadamente, fornecendo o seu valor inicial, selecionando o tipo de variável e, se a nova variável for do tipo *array*, indicando a sua dimensão. Ao se criar uma variável, automaticamente uma nova linha é adicionada à tabela, provendo espaço para mais variáveis. Pode-se, também, inserir uma nova variável entre duas já existentes através da utilização do menu *popup*. Basta clicar com o botão contrário do *mouse*. Este menu pode ser também utilizado para remover variáveis. É conveniente fazer um comentário sobre cada variável criada, utilizando-se para isso, o campo descritivo que aparece logo abaixo do painel e denominado *Comment*. Nele faz-se uma descrição do papel da variável no modelo, indicando para outra pessoa que utiliza o modelo o que a variável representa.

EJS aceita variáveis simples e multidimensionais. Se o campo *Dimension* de uma variável estiver vazio, então a variável criada é do tipo simples. Entretanto, se no referido campo estiver escrito [50], isso significa que a variável declarada é do tipo vetor com 50 coordenadas. Se estiver escrito [10][100], então é criada uma matriz com dez linhas e cem colunas.

O campo abaixo do cabeçalho *Initial value* é opcional. Embora exista o painel dedicado a inicialização (*Initialization*) da simulação, algumas vezes inicializar uma variável reduz-se a fornecer um valor ou o resultado de uma simples expressão. Nesse caso, não é necessário criar uma página de inicialização. Basta deixar o valor da variável indicada neste campo. Se, entretanto, a variável for do tipo *array*, todos os seus elementos apresentarão o mesmo valor inicial. Assim, se o campo *Initial value* estiver vazio, por *default* será atribuído o valor zero para as variáveis numéricas, *false*, para as variáveis booleanas, “ ” (espaço em branco), para *strings* e *null* para as variáveis do tipo objetos.

Ao se utilizar uma simples variável numa expressão em *Java*, basta escrever o seu nome e para as variáveis do tipo *array*, deve-se indicar qual elemento está se referindo. Isso é feito através da escrita do nome da variável pelo índice entre colchetes, lembrando-se que o primeiro elemento tem índice zero e o último, índice igual ao tamanho do arranjo menos um.

5.3 CONVENÇÃO PARA OS NOMES DAS VARIÁVEIS

Os nomes de alguns elementos devem obedecer algumas regras básicas para se evitar conflitos e tornar a simulação fácil para outras pessoas entenderem. A seguir são apresentadas as convenções para se nomear as variáveis.

1. O nome de toda variável, método ou elemento de visualização deve ser único em toda a simulação.
2. Os nomes são formados pela junção de caracteres alfanuméricos de *a* a *z*, de *A* a *Z* e de *0* até *9*, sem limitação de tamanho. A linguagem *Java* diferencia letras maiúsculas de minúsculas.
3. O primeiro caractere deve ser alfabético. Para as variáveis e métodos a primeira letra escolhida deve ser minúscula (de *a* até *z*). Para elementos de visualização, a primeira letra pode ser maiúscula.
4. É recomendado o uso de nomes descritivos. Para isso, várias palavras podem ser colocadas juntas (sem espaços em branco entre elas) para formar o nome. Nesse caso é melhor começar cada nova palavra com uma letra maiúscula.
5. Nomes começando com o hífen caixa baixa (_) são proibidos, pois o EJS utiliza-os para suas próprias variáveis e métodos.

5.4 OPERADORES

Os operadores utilizados na linguagem *Java* são do tipo aritmético, lógico, de atribuição e de comparação, cujas descrições são apresentadas rapidamente a seguir.

a. Operadores aritméticos: Eles são usados para representar as operações aritméticas de soma (+), subtração ($-$), multiplicação (*), divisão (/) e módulo (%).

   Exemplos:

   $3+4$, que indica a soma entre as variáveis inteiras 3 e 4.

   $4.0-5.0$, que indica a subtração entre os números reais 4,0 e 5,0.

   $2*7$, que indica a multiplicação entre os números inteiros 2 e 7.

   $14/2$, que indica a divisão entre os inteiros 14 e 2.

   $20\%7$, que se lê “20 módulo 7” e cujo resultado é o resto da divisão de 20 por 7.

b. Operadores lógicos: Estes são os operadores utilizados para se construir expressões lógicas, concatenando valores lógicos como “verdadeiro” ou “falso”, representados por **e** (&&), **ou** (||) e **não** (!).

c. Operadores de comparação: São usados para comparar duas expressões e os símbolos utilizados são > (maior), >= (maior ou igual), < (menor), <= (menor ou igual), == (igual) e != (diferente).
d. Operadores de atribuição: O símbolo usado para ele é o =. Assim, por exemplo, se aparecer numa linha de comando de um programa em Java a notação $x = 3$, deve-se entender que à variável *x* é atribuído o valor 3.

É importante observar que normalmente a precedência, ou ordem de execução, dos operadores coincide com a da notação matemática. Em caso de dúvida, aconselha-se usar parênteses para indicar o que realmente se deseja.

Outro aspecto que se deve observar é o de não misturar variáveis de tipos diferentes, ou seja, utilizar variáveis diferentes em uma mesma expressão aritmética. Outros tipos de erros ocorrem ao se efetuar divisões entre grandezas de tipos diferentes.

5.5 SENTENÇAS E EXPRESSÕES

Uma expressão é um conjunto de variáveis unidas por operadores e que utilizadas para instruir o computador como ele deverá executar uma operação ou operações. Uma sentença, por sua vez, é uma expressão seguida de um ponto e vírgula (;). Costuma-se escrever uma sentença por linha de modo a facilitar a sua leitura, ainda que seja possível escrever mais de uma sentença, uma após a outra, na mesma linha.

Ao se escrever um programa de computador é importante fazer comentários, seja para especificar as variáveis envolvidas ou para indicar a função de uma sentença. Na linguagem *Java*, isso é feito através do uso de duas barras (//). Se for necessário utilizar um comentário que utilize mais de uma linha, deve-se escrevê-lo entre os símbolos /* e */.

5.6 O DESVIO CONDICIONAL

O desvio condicional é utilizado para executar somente uma sentença de um grupo de duas ou mais delas.

Um tipo de desvio condicional é dado por *if-else* (se-senão) e sua estrutura é:

if (expressão_booleana) expressão 1;
else expressão 2*;*

O compilador ao encontrar uma estrutura deste tipo, avalia a expressão do tipo booleana e, se ela for verdadeira, então o compilador executa a expressão 1; caso contrário, ele executa a segunda expressão. Se a linha que contém a palavra *else* for suprimida, então o computador irá executar somente a expressão 1.

Caso sejam necessárias mais expressões dentro da estrutura *if-else*, então elas devem ser agrupadas formando um bloco, denominado bloco de código, que é delimitado pelos símbolos {, indicando o começo, e }, indicando o fim. A sua estrutura é representada por:

```
if (expressão_booleana) {
        expressão 1;
         expressão 2;
                        }
else expressão 3;
                }
```

5.7 AS ESTRUTURAS DE REPETIÇÃO

Uma estrutura de repetição é utilizada para se executar uma sentença ou um conjunto de sentenças repetidas vezes. Esse tipo de estrutura apresenta condição lógica de modo que o computador a executa enquanto a condição for verdadeira. Elas podem ser construídas a partir de *while* (enquanto), *do-while* (faça-enquanto) e *for* (para). Os formatos dessas estruturas são as seguintes:

```
while (expressão_booleana) {
expressão 1;
expressão 2;
}
```

```
do {
        expressão 1;
        expressão 2;
}while (expressão_booleana);
```

Na primeira estrutura apresentada acima, a expressão é avaliada antes de se executar o bloco de sentenças. Na segunda estrutura, o bloco é executado e depois se avalia a condição. Isso significa que o bloco é executado pelo menos uma vez.

A estrutura *for* é muito utilizada em programas envolvendo variáveis do tipo matriz. A forma de escrevê-la é:

```
for (expressãoInicialização; expressãoBooleana; expressãoIncremento) {
    expressão 1;
    expressão 2;
    }
```

5.8 INSTRUÇÕES ESPECIAIS E MÉTODOS DE BIBLIOTECA

As instruções especiais utilizadas na linguagem *Java* são as seguintes:

break: Essa instrução pode ser utilizada em um bloco de desvio condicional ou de estrutura de repetição. A sua função é fazer que o programa abandone o bloco sem executar as linhas que se seguem.

continue: É usada somente em estruturas de repetição e produz a interrupção do curso, sem executar as linhas que se seguem. Assim, o programa avaliará a condição booleana da estrutura para decidir se esta deve ser iterada novamente ou não.

return: Essa instrução faz com que o programa abandone completamente o método que está sendo executado. Para as simulações com o EJS, como os métodos se correspondem a páginas de código, a instrução *return* faz com que não se execute o resto da página. Nos métodos próprios (*Custom*) que devolvem um valor, essa instrução de indicar um valor devolvido neste ponto.

Além dos recursos apresentados, a linguagem *Java* dispõe de bibliotecas de rotinas predefinidas, como funções matemáticas, bibliotecas gráficas, etc. e que podem ser usadas através da chamada do seu nome em qualquer lugar do programa. Em *Java*, essas rotinas são chamadas de métodos e as bibliotecas de classes. Existem classes e métodos para, praticamente, tudo que se desejar. Entretanto, dependendo do tipo de simulação que se pretende com o uso do EJS, é necessário conhecer alguns métodos e classes.

O primeiro método está relacionado à chamada de uma rotina da biblioteca *Java*. Os métodos pertencem a entidades denominadas de classes e essas, por sua vez, são grupos de métodos e instruções dedicadas a realizar uma determinada tarefa. Existem classes

para se efetuar cálculos matemáticos, para descrever cores na interface do usuário, para tipos de fontes, para acessar arquivos, estabelecer conexão com a *Internet*, etc.

A linguagem *Java* apresenta dois tipos de métodos, os métodos classe e os métodos de instância. Os métodos de classe são denominados, também, de métodos estáticos e são mais simples de se usar. Esses métodos são os que cumprem com as necessidades básicas de se criar uma simulação com o EJS.

Para se chamar um método estático de qualquer classe, basta unir o nome da classe e o nome do método, separados por um ponto (.), mais dois parênteses que compreendem a lista de parâmetros de chamada do método. Assim, por exemplo, a forma correta de se chamar a função matemática seno é

y = java.lang.Math.sin (x);

em que *x* e *y* são variáveis do tipo *double*. Neste caso, o método *sin* (seno) pertence a classe *Math* que reside no grupo *lang* da família *java*. Entretanto, o caso anterior pode ser simplesmente referenciado como:

y = Math.sin (x);

A forma apresentada acima é uma exceção e só se aplica para classes do grupo *java.lang*. Esta é a classe mais importante, pois contém biblioteca matemática *Java*. Além do mais, essa classe define as constantes Math.PI e Math.E.

A tabela a seguir, mostra os métodos da biblioteca matemática *Java*.

double abs(double x)	Devolve o valor absoluto de x.
double acos(double x)	Devolve o arco coseno de x no intervalo de 0 a π.
double asin(double x)	Devolve o arco seno de x no intervalo de – π/2 a π/2.
double atan(double x)	Devolve o arco tangente de x no intervalo de – π/2 a π/2.
double ceil(double x)	Devolve o menor inteiro maior ou igual a x.
double cos(double x)	Devolve o coseno de x.
double sin(double x)	Devolve o seno de x.
double tan(double x)	Devolve a tangente de x.
double exp(double x)	Devolve a exponencial de x na base *e*, isto é e^x.
double floor (double x)	Devolve o maior inteiro menor ou igual a x.
double log(double x)	Devolve o logaritmo natural (na base *e*) de x
double max(double x, double y)	Devolve o maior dos números x e y.
int max(int x, int y)	Devolve o maior dos números inteiros x e y.
double min(double x, double y)	Devolve o menor dos números x e y.
int min(int x, int y)	Devolve o menor dos números inteiros x e y.
double pow(double x, double y)	Devolve x elevado a y.
double random( )	Devolve um número aleatório entre 0.0 e 1.0, excluindo o 0.0.

double rint(double x)	Devolve um número inteiro na forma *double* próximo de x.
lont round(double x)	Devolve um número inteiro na forma *long* próximo de x.
double sqrt(double x)	Devolve a raiz quadrada de x.
double atan2(double x, double y)	Devolve o argumento do número complexo $x + y.i$.

6 PROPRIEDADES DOS ELEMENTOS DA JANELA DE VISUALIZAÇÃO

Nesse capítulo apresentamos as propriedades de alguns elementos gráficos. Essas propriedades foram obtidas no site http://fem.um.es/Ejs através do *help* no próprio programa EJS. Para isso, o usuário deve apontar a seta do *mouse* sobre o ícone que se deseja conhecer as propriedades e, a seguir, aparecerá uma janela (*on line*) com a descrição de todas as ações relacionadas. Dessa forma, traduzimos as propriedades de alguns dos elementos utilizados nas simulações ao longo desse manual de procedimentos. Alertamos que são muitos e dominar o conhecimento de todas as propriedades, elemento por elemento, exige-se um intervalo de tempo longo. A medida que nosso trabalho se desenvolver, novas propriedades relacionadas aos elementos efetivamente usados nos exemplos de simulações serão acrescentadas.

6.1 ELEMENTOS DA ABA "WINDOWS, CONTAINERES AND DRAWING PANELS" DO GRUPO "INTERFACE"

6.1.1 Elemento *Frame*

Ícone: 

Texto: *A top-level window.*

Descrição: Um *Frame* é um elemento contêiner que funciona como uma janela independente.

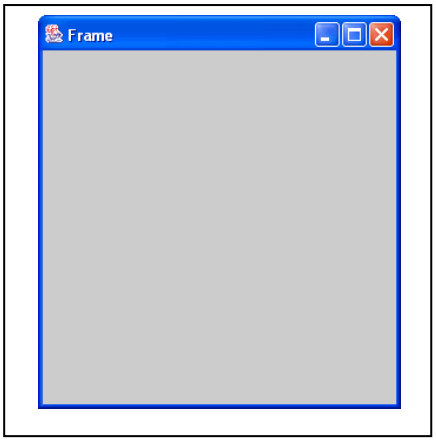


Tabela de Propriedades		
Nome	Descrição	Valores aceitos pelas variáveis
Main (Principal)		
Title	O texto mostrado com título para essa janela.	Alguma constante ou variável do tipo **String**.
Layout	A "política" de distribuição das crianças nesse container.	Uma das seguintes constantes: **border**, **flow**, **grid**, **hbox**, **vbox**. • **flow** requer um parâmetro adicional que indica co alinhamento de suas crianças. Pode ser uma das seguintes constantes: **left**, **center** ou **right**. • **grid** requer dois parâmetros inteiros adicionais, indicando o número de linhas e colunas da grade, respectivamente. Todas as opções aceitam dois parâmetros inteiros adicionais que indicam a separação horizontal e vertical de seus filhos. Por exemplo, **border:10,5**.
Visible	A visibilidade do elemento.	Uma variável **booleana** ou uma das constantes **true** ou **false**.
Location and Size (Local e tamanho)		
Location	A local da janela na tela do computador, em pixels.	Valores inteiros para as posições X e Y, respectivamente, separados por vírgula. O valor especial **center** localiza a janela no centro da tela.
Size	O tamanho do elemento.	Valores inteiros para a largura e a altura, respectivamente, separados por vírgula. O valor especial **pack** ajusta o tamanho para mínimo requerido pela criança do elemento.
Resizable	Se o usuário puder redimensionar a janela.	Uma variável **booleana** ou uma das constantes **true** ou **false**.
Graphical Aspect (Aspecto gráfico)		
Background	A cor usada como fundo no botão.	Uma das cores pré-definidas: **black**, **blue**,

		cyan, **darkGray**, **gray**, **green**, **lightGray**, **magenta**, **orange**, **pink**, **red**, **white**, **yellow**. A cor pode, também, ser especificada pela coordenada inteira RGB (entre 0 e 255), mais uma coordenada de transparência opcional.
Foreground	A cor usada para o texto exibido pelo botão.	Os mesmos valores utilizados em **Background**.
Font	O tipo de fonte exibido pelo botão.	O estilo pode ser um dos tipos: **plain**, **bold**, **italic**, **bold**, **italic**.
Tooltip	O texto exibido quando o cursor do mouse é colocado sobre o botão.	Constante ou variável do tipo **String**.

6.1.2 Elemento *Dialog*

Ícone: 

Texto: *A dialog window.*

Descrição: O elemento *Dialog* é um contêiner que exibe em uma janela independente. As janelas de diálogo podem ser fechadas, mas não minimizadas. Diferentemente dos frames, fechar uma janela de diálogo não produz nenhum efeito.

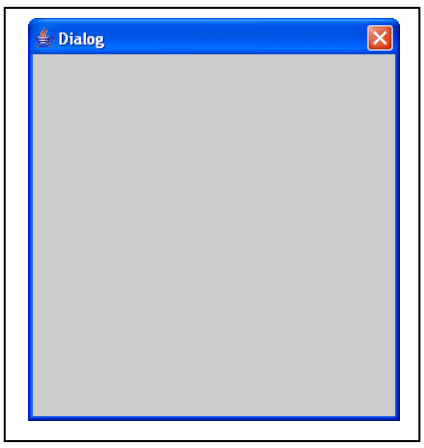
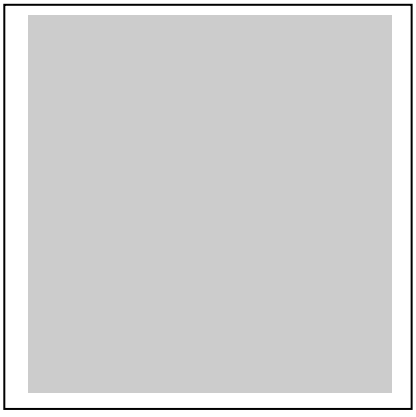


Tabela de Propriedades		
Nome	Descrição	Valores aceitos pelas variáveis
Main (Principal)		
Title	O texto mostrado com título para essa janela.	Alguma constante ou variável do tipo **String**.
Layout	A "política" de distribuição das crianças nesse container.	Uma das seguintes constantes: **border**, **flow**, **grid**, **hbox**, **vbox**. • **flow** requer um parâmetro adicional que indica co alinhamento de suas crianças. Pode ser uma das seguintes constantes: **left**, **center** ou **right**. • **grid** requer dois parâmetros inteiros adicionais, indicando o número de linhas e colunas da grade, respectivamente. Todas as opções aceitam dois parâmetros inteiros adicionais que indicam a separação horizontal e vertical de seus filhos. Por exemplo, **border:10,5**.
Visible	A visibilidade do elemento.	Uma variável **booleana** ou uma das constantes **true** ou **false**.
Location and Size (Local e tamanho)		
Location	O local da janela na tela do computador, em pixels.	Valores inteiros para as posições X e Y, respectivamente, separados por vírgula. O valor especial **center** localiza a janela no centro da tela.
Size	O tamanho do elemento.	Valores inteiros para a largura e a altura, respectivamente, separados por vírgula. O valor especial **pack** ajusta o tamanho para mínimo requerido pela criança do elemento.
Resizable	Se o usuário puder redimensionar a janela.	Uma variável **booleana** ou uma das constantes **true** ou **false**.
Graphical Aspect (Aspecto gráfico)		
Background	A cor usada para o fundo do	Uma das cores pré-definidas: **black**, **blue**,

	elemento e aquelas cores de suas crianças (a menos que elas sejam fixadas explicitamente através das propriedades).	**cyan**, **darkGray**, **gray**, **green**, **lightGray**, **magenta**, **orange**, **pink**, **red**, **white**, **yellow**. A cor pode, também, ser especificada pela coordenada inteira RGB (entre 0 e 255), mais uma coordenada de transparência opcional.
Foreground	A cor usada para o fundo do elemento e aquelas cores de suas crianças (a menos que elas sejam fixadas explicitamente através das propriedades).	Os mesmos valores utilizados em **Background**.
Font	A cor usada para o fundo do elemento e aquelas cores de suas crianças (a menos que elas sejam fixadas explicitamente através das propriedades).	O estilo pode ser um dos tipos: **plain**, **bold**, **italic**, **bold**, **italic**.
Tooltip	O texto exibido quando o cursor do mouse é colocado sobre o botão.	Constante ou variável do tipo **String**.

6.1.3 Elemento *Panel*

Ícone: 



Texto: *A basic container panel.*

Descrição: Um *Panel* é o contêiner mais básico. Ele pode ser usado apenas para hospedar um ou mais filhos, de acordo com sua propriedade de *layout*. O painel pode ser escondido usando a propriedade *Visible*.

Tabela de Propriedades		
Nome	Descrição	Valores aceitos pelas variáveis

Main (Principal)		
Title	O texto mostrado com título para essa janela.	Alguma constante ou variável do tipo **String**.
Layout	A "política" de distribuição das crianças nesse container.	Uma das seguintes constantes: **border**, **flow**, **grid**, **hbox**, **vbox**. • **flow** requer um parâmetro adicional que indica co alinhamento de suas crianças. Pode ser uma das seguintes constantes: **left**, **center** ou **right**. • **grid** requer dois parâmetros inteiros adicionais, indicando o número de linhas e colunas da grade, respectivamente. Todas as opções aceitam dois parâmetros inteiros adicionais que indicam a separação horizontal e vertical de seus filhos. Por exemplo, **border:10,5**.
Visible	A visibilidade do elemento.	Uma variável **booleana** ou uma das constantes **true** ou **false**.
Size	O tamanho do elemento.	Valores inteiros para a largura e a altura, respectivamente, separados por vírgula.
Border	Uma área vazia que circula o elemento.	Quatro números inteiros indicativos das margens superior (top), esquerda (left), inferior (bottom) e direita (right), respectivamente.
Graphical Aspect (Aspecto gráfico)		
Background	A cor usada como fundo no botão.	Uma das cores pré-definidas: **black**, **blue**, **cyan**, **darkGray**, **gray**, **green**, **lightGray**, **magenta**, **orange**, **pink**, **red**, **white**, **yellow**. A cor pode, também, ser especificada pela coordenada inteira RGB (entre 0 e 255), mais uma coordenada de transparência opcional.
Foreground	A cor usada para o texto exibido pelo botão.	Os mesmos valores utilizados em **Background**.

Font	O tipo de fonte exibido pelo botão.	O estilo pode ser um dos tipos: **plain**, **bold**, **italic**, **bold**, **italic**.
Tooltip	O texto exibido quando o cursor do mouse é colocado sobre o botão.	Constante ou variável do tipo **String**.

6.1.4 Elemento *ToolBar*

Ícone: 

Texto: *A toolbar container*.

Descrição: Um *ToolBar* é um elemento *container* utilizado para mostrar controles que são utilizados com frequência. Geralmente é exibida na sua janela principal, mas também podem ser arrastados para uma janela separada.

Tabela de Propriedades		
Nome	Descrição	Valores aceitos pelas variáveis
Main (Principal)		
Floatable	Se o elemento puder ser arrastado para uma posição diferente dentro do próprio container ou para fora de sua própria janela.	Uma variável **booleana** ou uma das constantes **true** ou **false**.
Layout	A "política" de distribuição das crianças nesse container.	Uma das seguintes constantes: **border**, **flow**, **grid**, **hbox**, **vbox**. • **flow** requer um parâmetro adicional que indica co alinhamento de suas crianças. Pode ser uma das seguintes constantes: **left**, **center** ou **right**. • **grid** requer dois parâmetros inteiros adicionais, indicando o

		número de linhas e colunas da grade, respectivamente. Todas as opções aceitam dois parâmetros inteiros adicionais que indicam a separação horizontal e vertical de seus filhos. Por exemplo, **border:10,5**.
Orientation	A orientação do elemento.	Uma das constantes: **horizontal** ou **vertical**.
Visible	A visibilidade do elemento.	Uma variável **booleana** ou uma das constantes **true** ou **false**.
Size	O tamanho do elemento.	Valores inteiros para a largura e a altura, respectivamente, separados por vírgula. O valor especial **pack** ajusta o tamanho para mínimo requerido pela criança do elemento.
Border	Uma área vazia que circula o elemento.	Quatro números inteiros indicativos das margens superior (top), esquerda (left), inferior (bottom) e direita (right), respectivamente.
Graphical Aspect (Aspecto gráfico)		
Border Painted	Se as fronteiras puderem ser pintadas.	Uma variável **booleana** ou uma das constantes **true** ou **false**.
Roll over	Se a fronteira dos botões da barra de ferramentas devem ser estabelecidas apenas quando o ponteiro do mouse estiver sobre eles.	Uma variável **booleana** ou uma das constantes **true** ou **false**.
Background	A cor usada como fundo no botão.	Uma das cores pré-definidas: **black**, **blue**, **cyan**, **darkGray**, **gray**, **green**, **lightGray**, **magenta**, **orange**, **pink**, **red**, **white**, **yellow**. A cor pode, também, ser especificada pela coordenada inteira RGB (entre 0 e 255), mais uma coordenada de transparência opcional.
Foreground	A cor usada para o texto exibido	Os mesmos valores utilizados em

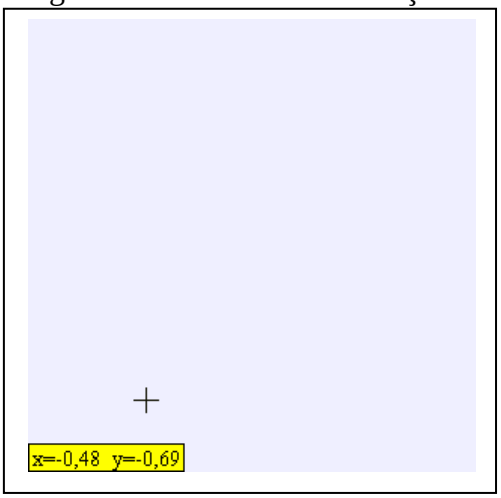
	pelo botão.	**Background**.
Font	O tipo de fonte exibido pelo botão.	O estilo pode ser um dos tipos: **plain**, **bold**, **italic**, **bold**, **italic**.
Tooltip	O texto exibido quando o cursor do mouse é colocado sobre o botão.	Constante ou variável do tipo **String**.

6.1.5 Elemento *DrawingPanel*

Ícone: 

Texto: *A 2D container for Drawables.*

Descrição: O *DrawingPanel* é um contêiner especial que age como anfitrião de "crianças" de elemento *drawable*. Até mesmo quando é considerado um recipiente, não pode ser usado para ser anfitrião de outro tipo de elementos.



Os painéis de desenho representam uma região plana e fornece o seu próprio sistema de coordenadas partido do ponto (X Mínimo, Y Mínimo) para o ponto (X Máximo, Y Máximo), embora as escalas possam ser ajustadas automaticamente.

Os painéis de desenho são interativos e respondem aos diferentes movimentos do *mouse* de acordo com o seguinte procedimento:

"Quando o usuário clicar no painel, a ação associada à propriedade "*On Press*" é chamada. Imediatamente após essa ação, as propriedades "X" e "Y" recebem os valores correspondentes à posição atual do *mouse*. Isto também chama a ação associada à propriedade "*On Drag*".

"Quando o usuário arrasta o *mouse* (com o botão apertado), as propriedades "X" e "Y" são atualizadas e a ação associada à propriedade "On Drag" é chamada.

"Quando o usuário finalmente solta o botão do *mouse* (se dentro do painel), a ação associada à propriedade "*On Release*" é chamada.

Os painéis de desenho também respondem à interação com o teclado. Quando o usuário "teclar" nele, dispara-se a chave *Key Action*. A chave apertada pode ser obtida pela propriedade *Key Pressed*.

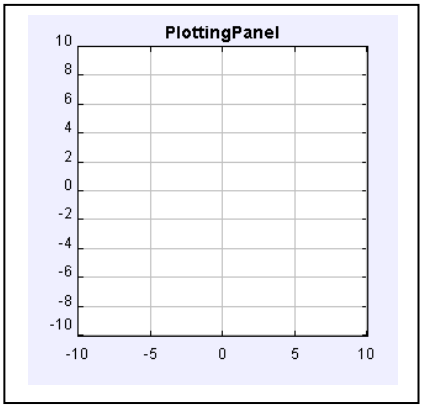
Tabela de Propriedades		
Nome	Descrição	Valores aceitos pelas variáveis
Scales (Escalas)		
Autoscale X	Se a escala do eixo X for computada automaticamente.	Uma variável booleana ou uma das constantes **true** ou **false**.
Autoscale Y	Se a escala do eixo Y for computada automaticamente.	Se a escala do eixo X for computada automaticamente.
Minimum X	O menor valor da coordenada X que pode ser visualizada na janela.	Uma constante ou variável do tipo **double**.
Maximum X	O maior valor da coordenada X que pode ser visualizada na janela.	Uma constante ou variável do tipo **double**.
Minimum Y	O menor valor da coordenada Y que pode ser visualizada na janela.	Uma constante ou variável do tipo **double**.
Maximum Y	O maior valor da coordenada Y que pode ser visualizada na janela.	Uma constante ou variável do tipo **double**.
Interaction (Interação)		
X	A coordenada X da posição do mouse.	Uma constante ou variável do tipo **double**.
Y	A coordenada Y da posição do mouse.	Uma constante ou variável do tipo **double**.
On Press	A ação que invoca quando o mouse deve ser pressionado no painel.	O código Java que invoca a ação.
On Drag	A ação que invoca quando o mouse deve ser arrastado no	O código Java que invoca a ação.

	painel.	
On Release	A ação para invocar quando o mouse é liberado (solto) no painel.	O código Java que invoca a ação.
Mouse Enter	A ação que invoca quando o ponteiro entra no elemento.	O código Java que invoca a ação.
Mouse Exit	A ação que invoca quando o ponteiro sai do elemento.	O código Java que invoca a ação.
Key Action	A ação para invocar quando qualquer tecla é acionada enquanto o painel estiver focalizado.	O código Java que invoca a ação.
Key Pressed	O código inteiro da tecla acionada.	Uma variável do tipo **int**.
Configuration (Configuração)		
Square	Se for mantida a razão 1:1 entre as escalas X e Y.	Uma variável booleana ou uma das constantes **true** ou **false**.
Gutters	Os espaços vazios que devem ser deixados ao redor da área de desenho do painel.	Quatro números inteiros separados por vírgulas, correspondendo às margens top, left, bottom e right, respectivamente.
Coordinates	Se deve aparecer as coordenadas do mouse quando se clicar no painel.	Uma variável booleana ou uma das constantes **true** ou **false**.
X Format	O formato para se visualizar a coordenada X	Algum valor aceito pela classe **java.text.DecimalFormat**. Por default "x=0.000".
Y Format	O formato para se visualizar a coordenada X	Algum valor aceito pela classe **java.text.DecimalFormat**. Por default "x=0.000".
Graphical Aspect (Aspecto gráfico)		
Visible	A visibilidade do elemento.	Uma variável booleana ou uma das constantes **true** ou **false**.
Size	As dimensões do elemento.	Valores inteiros para a largura e a altura, respectivamente, separados por vírgula.
Background	A cor usada para o fundo do elemento e aquelas cores de suas	Uma das cores pré-definidas: **black**, **blue**, **cyan**, **darkGray**, **gray**, **green**, **lightGray**,

	crianças (a menos que elas sejam fixadas explicitamente através das propriedades).	**magenta**, **orange**, **pink**, **red**, **white**, **yellow**. A cor pode, também, ser especificada pela coordenada inteira RGB (entre 0 e 255), mais uma coordenada de transparência opcional.
Foreground	A cor usada para o fundo do elemento e aquelas cores de suas crianças (a menos que elas sejam fixadas explicitamente através das propriedades).	Os mesmos valores utilizados em **Background**.
Font	A cor usada para o fundo do elemento e aquelas cores de suas crianças (a menos que elas sejam fixadas explicitamente através das propriedades).	O estilo pode ser um dos tipos: **plain**, **bold**, **italic**, **bold**, **italic**.
Tooltip	O texto exibido quando o cursor do mouse é colocado sobre o botão.	Constante ou variável do tipo **String**.

6.1.6 Elemento *PlottingPanel*

Ícone: 



Texto: *A 2D drawing panel with a system of axes.*

Descrição: O elemento *PlottingPanel* é um painel de desenho em duas dimensões que fornece, por *default*, um sistema de eixos coordenados. Esses elementos interativos respondem à ação do usuário da mesma forma que os painéis de desenho.

Tabela de Propriedades		
Nome	Descrição	Valores aceitos pelas variáveis
Decoration and Axes (Decoração e eixos)		
Title	O título mostrado no topo do painel.	Uma constante ou variável do tipo **String**.
Tt Font	A fonte para o título.	Valores aceitos pela propriedade **Font**.
Axes Type	O tipo de eixos mostrados. Existem três tipos de eixos cartesianos e dois tipos de eixos para as coordenadas polares.	Uma das constantes: **Cartesian1**, **Cartesian2**, **Cartesian3**, **Polar1**, **Polar2**.
Title X	O título mostrado pelo eixo X.	Uma constante ou variável do tipo **String**.
X Axis Pos	A posição escolhida para o eixo X.	Uma constante ou variável do tipo **String**.
X Axis Type	Linear ou logarítmica (base 10)	Uma das constantes: **LINEAR**, **LOG10**.
Grid X	Se desenhar a grade no eixo X.	Uma variável booleana ou uma das constantes **true** ou **false**.
Title Y	O título mostrado pelo eixo Y.	Uma constante ou variável do tipo **String**.
Y Axis Pos	A posição escolhida para o eixo Y.	Uma constante ou variável do tipo **String**.
Y Axis Type	Linear ou logarítmica (base 10)	Uma das constantes: **LINEAR**, **LOG10**.
Grid Y	Se desenhar a grade no eixo Y.	Uma variável booleana ou uma das constantes **true** ou **false**.
Delta R	O incremento entre as linhas; ro=constante no caso de eixos polares.	Uma constante ou variável do tipo **double**.
Delta Theta	O incremento entre as linhas; theta=constante no caso dos eixos polares.	Uma constante ou variável do tipo **double**.
Scales (Escalas)		
Autoscale X	Se a escala do eixo X for computada automaticamente.	Uma variável booleana ou uma das constantes **true** ou **false**.
Autoscale Y	Se a escala do eixo Y for computada automaticamente.	Uma variável booleana ou uma das constantes **true** ou **false**.
Minimum X	O menor valor da coordenada X que pode ser visualizada na janela.	Uma constante ou variável do tipo **double**.

Maximum X	O maior valor da coordenada X que pode ser visualizada na janela.	Uma constante ou variável do tipo **double**.
Minimum Y	O menor valor da coordenada Y que pode ser visualizada na janela.	Uma constante ou variável do tipo **double**.
Maximum Y	O maior valor da coordenada Y que pode ser visualizada na janela.	Uma constante ou variável do tipo **double**.
Interaction (Interação)		
X	A coordenada X da posição do mouse.	Uma constante ou variável do tipo **double**.
Y	A coordenada Y da posição do mouse.	Uma constante ou variável do tipo **double**.
On Press	A ação que invoca quando o mouse deve ser pressionado no painel.	O código Java que invoca a ação.
On Drag	A ação que invoca quando o mouse deve ser arrastado no painel.	O código Java que invoca a ação.
On Release	A ação para invocar quando o mouse é liberado (solto) no painel.	O código Java que invoca a ação.
Mouse Enter	A ação que invoca quando o ponteiro entra no elemento.	O código Java que invoca a ação.
Mouse Exit	A ação que invoca quando o ponteiro sai do elemento.	O código Java que invoca a ação.
Key Action	A ação para invocar quando qualquer tecla é acionada enquanto o painel estiver focalizado.	O código Java que invoca a ação.
Key Pressed	O código inteiro da tecla acionada.	Uma variável do tipo **int**.
Configuration (Configuração)		
Square	Se for mantida a razão 1:1 entre as escalas X e Y.	Uma variável booleana ou uma das constantes **true** ou **false**.

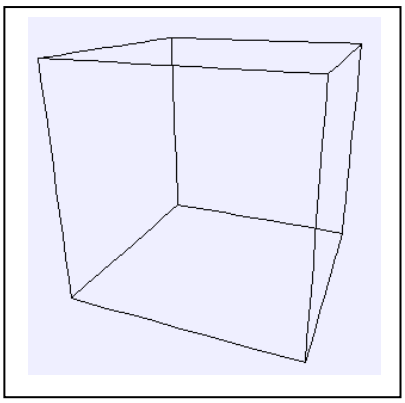
Gutters	Os espaços vazios que devem ser deixados ao redor da área de desenho do painel.	Quatro números inteiros separados por vírgulas, correspondendo às margens top, left, bottom e right, respectivamente.
Coordinates	Se deve aparecer as coordenadas do mouse quando se clicar no painel.	Uma variável booleana ou uma das constantes **true** ou **false**.
X Format	O formato para se visualizar a coordenada X	Algum valor aceito pela classe **java.text.DecimalFormat**. Por default "x=0.000".
Y Format	O formato para se visualizar a coordenada X	Algum valor aceito pela classe **java.text.DecimalFormat**. Por default "x=0.000".
Graphical Aspect (Aspecto gráfico)		
Visible	A visibilidade do elemento.	Uma variável booleana ou uma das constantes **true** ou **false**.
Size	As dimensões do elemento.	Valores inteiros para a largura e a altura, respectivamente, separados por vírgula.
Background	A cor usada para o fundo do elemento e aquelas cores de suas crianças (a menos que elas sejam fixadas explicitamente através das propriedades).	Uma das cores pré-definidas: **black**, **blue**, **cyan**, **darkGray**, **gray**, **green**, **lightGray**, **magenta**, **orange**, **pink**, **red**, **white**, **yellow**. A cor pode, também, ser especificada pela coordenada inteira RGB (entre 0 e 255), mais uma coordenada de transparência opcional.
Foreground	A cor usada para o fundo do elemento e aquelas cores de suas crianças (a menos que elas sejam fixadas explicitamente através das propriedades).	Os mesmos valores utilizados em **Background**.
Font	A cor usada para o fundo do elemento e aquelas cores de suas crianças (a menos que elas sejam fixadas explicitamente através das propriedades).	O estilo pode ser um dos tipos: **plain**, **bold**, **italic**, **bold**, **italic**.
Tooltip	O texto exibido quando o cursor do mouse é colocado sobre o	Constante ou variável do tipo **String**.

	botão.	

6.1.7 Elemento *DrawingPanel3D*

Ícone: 

Texto: *A 3D container for Drawables.*



Descrição: O *DrawingPanel3D* é um recipiente especial cuja função é a de ser o anfitrião das "crianças" dos elementos de desenho. Esses painéis representam uma região do espaço tridimensional e fornecem o seu próprio sistema de coordenada do ponto (X Mínimo, Y Mínimo, Z Mínimo) para o ponto (Máximo X, Máximo Y, Máximo Z), embora as escalas também possam ser ajustadas automaticamente.

O painel de desenho DrawingPanel3D é interativo, mas o seu uso é diferente de painéis de desenho do tipo bidimensional. O seu uso pode ocorrer da seguinte forma:

• Clicando e arrastando o *mouse* sobre uma área vazia do painel, altera-se o ponto de visão da cena. Segurando o botão esquerdo ou o direito do *mouse*, pode alterar o modo como a cena é redesenhada durante a execução da simulação.

• Mantendo a tecla "*Control*" pressionada enquanto o mouse é arrastado pela cena.

• Mantendo a tecla "*Shift*" pressionada, enquanto o *mouse* é arrastado, altera-se o tamanho da cena (positiva ou negativamente, dependendo do movimento do mouse).

• Segurando a tecla "*Alt*", enquanto clicando e arrastando o *mouse*, traz um cursor tridimensional que permite ao usuário selecionar um ponto na cena 3D. O movimento deste cursor segue o movimento bidimensional do mouse afetando apenas duas coordenadas espaciais de cada vez. Para forçar o movimento precisamente sobre uma das coordenadas, é necessário manter a tecla X, Y ou Z pressionada. Quando o cursor muda de posição, as ações associadas às propriedades "*On Press*", "*On Drag*" e "*On Release*" são chamadas exatamente da mesma forma como no caso dos painéis de desenho bidimensional.

Os painéis de desenho 3D painéis também respondem à interação do teclado. Quando o usuário clicar sobre eles, obtém-se o foco de forma que ao acionar uma tecla desencadeia uma ação de chave (*Key Action*). A tecla pressionada pode ser obtida através da propriedade *Key Pressed*.

Tabela de Propriedades		
Nome	Descrição	Valores aceitos pelas variáveis
Scales (Escalas)		
Minimum X	O menor valor da coordenada X que pode ser visualizada na janela.	Uma constante ou variável do tipo **double**.
Maximum X	O maior valor da coordenada X que pode ser visualizada na janela.	Uma constante ou variável do tipo **double**.
Minimum Y	O menor valor da coordenada Y que pode ser visualizada na janela.	Uma constante ou variável do tipo **double**.
Maximum Y	O maior valor da coordenada Y que pode ser visualizada na janela.	Uma constante ou variável do tipo **double**.
Minimum Z	O menor valor da coordenada Z que pode ser visualizada na janela.	Uma constante ou variável do tipo **double**.
Maximum Z	O maior valor da coordenada Z que pode ser visualizada na janela.	Uma constante ou variável do tipo **double**.
Size X	O tamanho do elemento na direção X.	Uma constante ou variável do tipo **double** ou **int**.
Size Y	O tamanho do elemento na direção Y.	Uma constante ou variável do tipo **double** ou **int**.
Size Z	O tamanho do elemento na direção Z.	Uma constante ou variável do tipo **double** ou **int**.
Camera (click to reset)		
Auto adjust	Se a câmara deve ser ajustada automaticamente na mudança.	Uma variável booleana ou uma das constantes **true** ou **false**.
Projection	Como deve ser projetada a cena	Use o editor fornecido para fazer a

	3D no monitor 2D	escolha.
Draggable	Se o elemento pode ser arrastado.	Uma variável booleana ou uma das constantes **true** ou **false**.
Location X	A posição X da câmara.	Uma constante ou variável do tipo **double** ou **int**.
Location Y	A posição Y da câmara.	Uma constante ou variável do tipo **double** ou **int**.
Location Z	A posição Z da câmara.	Uma constante ou variável do tipo **double** ou **int**.
Azimuth	O ângulo azimutal da posição da câmara.	Uma constante ou variável do tipo **double** ou **int**.
Altitude	O ângulo de altitude da posição da câmara.	Uma constante ou variável do tipo **double** ou **int**.
Focus X	A posição X do foco da câmara.	Uma constante ou variável do tipo **double** ou **int**.
Focus Y	A posição Y do foco da câmara.	Uma constante ou variável do tipo **double** ou **int**.
Focus Z	A posição Z do foco da câmara.	Uma constante ou variável do tipo **double** ou **int**.
Rotation	A rotação da câmara.	Uma constante ou variável do tipo **double** ou **int**.
Screen At	A distância da câmara até a tela de projeção.	Uma constante ou variável do tipo **double** ou **int**.
Interaction (Interação)		
Menu Entry	O nome a ser exibido na entrada correspondente a simulação de elementos do menu.	Uma variável do tipo String.
Enabled	Se o elemento responde à ação do usuário.	Uma variável booleana ou uma das constantes **true** ou **false**.
Pos X	A coordenada X da posição do ponteiro do mouse.	Uma constante ou variável do tipo **double**.
Pos Y	A coordenada Y da posição do ponteiro do mouse.	Uma constante ou variável do tipo **double**.
Pos Z	A coordenada Z da posição do mouse.	Uma constante ou variável do tipo **double**.
On Press	A ação que invoca quando o	O código Java que invoca a ação.

	mouse deve ser pressionado no painel.	
On Drag	A ação que invoca quando o mouse deve ser arrastado no painel.	O código Java que invoca a ação.
On Release	A ação para invocar quando o mouse é liberado (solto) no painel.	O código Java que invoca a ação.
Mouse Move	A ação que invoca quando o ponteiro do mouse se movimenta sobre o elemento.	O código Java que invoca a ação.
On Exit	A ação que invoca quando o ponteiro sai do elemento.	O código Java que invoca a ação.
Key Action	A ação para invocar quando qualquer tecla é acionada enquanto o painel estiver focalizado.	O código Java que invoca a ação.
Key Pressed	O código inteiro da tecla acionada.	Uma variável do tipo **int**.
Configuration (Configuração)		
Implementation	A implementação usada para mostra a cena.	Use o editor fornecido para fazer a escolha.
Axes Mapping	O mapeamento entre as coordenadas e os eixos no modo 3D.	Use o editor fornecido para fazer a escolha.
Default Lights	Se é para ter luz por default.	Uma variável booleana ou uma das constantes **true** ou **false**.
Decoration	O tipo de elemento da cena (eixos, caixa) a ser mostrado.	Use o editor fornecido para fazer a escolha.
Axes Labels	O rótulo para os eixos.	Uma variável do tipo String.
Cursor	O tipo de cursor a ser mostrado.	Use o editor fornecido para fazer a escolha.
Hide lines	Se é para remover linhas ocultas.	Uma variável booleana ou uma das constantes **true** ou **false**.
Quick Redraw	Se é para permitir o desenho de fios quando em rotação.	Uma variável booleana ou uma das constantes **true** ou **false**.

Color Depth	Se é para obscurecer o elemento que está se distanciando.	Uma variável booleana ou uma das constantes **true** ou **false**.
Square	Se é para manter um aspecto quadrado.	Uma variável booleana ou uma das constantes **true** ou **false**.
Coordinates	Se é para mostra as coordenadas quando o mouse for pressionado.	Uma variável booleana ou uma das constantes **true** ou **false**.
X Format	O formato da coordenada X.	
Y Format	O formato da coordenada Y.	
Z Format	O formato da coordenada Z.	
Graphical Aspect (Aspecto gráfico)		
Visible	A visibilidade do elemento.	Uma variável booleana ou uma das constantes **true** ou **false**.
Size	As dimensões do elemento.	Valores inteiros para a largura e a altura, respectivamente, separados por vírgula.
Image	O arquivo com a imagem a ser mostrada.	Uma variável do tipo String ou usar o editor que é fornecido para selecionar o arquivo.
Moveable	Se a imagem de fundo é móvel.	Uma variável booleana ou uma das constantes **true** ou **false**.
Background	A cor usada para o fundo do elemento.	Uma das cores pré-definidas: **black**, **blue**, **cyan**, **darkGray**, **gray**, **green**, **lightGray**, **magenta**, **orange**, **pink**, **red**, **white**, **yellow**. A cor pode, também, ser especificada pela coordenada inteira RGB (entre 0 e 255), mais uma coordenada de transparência opcional.
Foreground	A cor usada quando desenhar ou escrever o elemento.	Os mesmos valores utilizados em **Background**.
Font	A fonte usada para mostrar o texto no elemento.	O estilo pode ser um dos tipos: **plain**, **bold**, **italic**, **bold**, **italic**.
Tooltip	O texto exibido quando o cursor do mouse é colocado sobre o elemento.	Constante ou variável do tipo **String**.

6.2 ELEMENTOS DA ABA *"BUTTONS AND DECORATION"* DO GRUPO *"INTERFACE"*

6.2.1 Elemento *Label*

Ícone: 

Texto: *A decorative label.*

Descrição: Um *Label* é um elemento básico usado para exibir um texto ou uma imagem, ou ambos, com propósito informativo ou decorativo.

Tabela de Propriedades		
Nome	Descrição	Valores aceitos pelas variáveis
Main (Principal)		
Variable	O texto exibido pelo elemento.	Alguma constante ou variável do tipo **String**.
Image	A imagem exibida pelo elemento.	Constante ou variável do tipo **String** correspondente a uma imagem com a extensão **GIF** ou a uma imagem animada do tipo **GIF**. A string indica o caminho para o arquivo da imagem correspondente. O caminho pode ser do diretório de trabalho ou por uma URL da internet.
Alignment	Como alinhar o texto do elemento.	Uma das constantes: **left**, **center** ou **right**.
Graphical Aspect (Aspecto gráfico)		
Size	O tamanho do elemento.	Valores inteiros para a largura e a altura, respectivamente, separados por vírgula.
Background	A cor usada como fundo no botão.	Uma das cores pré-definidas: **black**, **blue**, **cyan**, **darkGray**, **gray**, **green**, **lightGray**, **magenta**, **orange**, **pink**, **red**, **white**, **yellow**. A cor pode, também, ser especificada pela coordenada inteira RGB (entre 0 e 255), mais uma coordenada de

		transparência opcional.
Foreground	A cor usada para o texto exibido pelo botão.	Os mesmos valores utilizados em **Background**.
Font	O tipo de fonte exibido pelo botão.	O estilo pode ser um dos tipos: **plain**, **bold**, **italic**, **bold**, **italic**.
Tooltip	O texto exibido quando o cursor do mouse é colocado sobre o botão.	Constante ou variável do tipo **String**.

6.2.2 Elemento Button

Ícone: 

Texto:

Descrição: Um *Button* é um elemento básico usado para chamar ações. Ele pode exibir um texto, uma imagem, ou ambos. A ação é chamada quando o botão é clicado (ou seja, pressionado e liberado). Os botões podem ser desativados através do uso da propriedade "*Enabled*". Neste caso, a aparência acinzentada fica clareada.

Tabela de Propriedades		
Nome	Descrição	Valores aceitos pelas variáveis
Main (Principal)		
Text	O texto exibido pelo elemento.	Constante ou variável do tipo **String**.
Image	A imagem exibida pelo elemento.	Constante ou variável do tipo **String** correspondente a uma imagem com a extensão **GIF** ou a uma imagem animada do tipo **GIF**.
Mnemonic	A tecla que (juntamente com Alt) ativa esse menu.	**String** de um só caractere.
Alignment	Como se deve alinhar o texto no botão.	Uma das constantes: **left**, **center** ou **right**.
Enabled	Se o elemento responde à intera-ção do usuário.	Variável do tipo **boolean** ou uma das constantes **true** ou **false**.

Action	A ação que invoca quando o botão é clicado.	Código Java para invocar a ação.
Graphical Aspect (Aspecto gráfico)		
Size	O tamanho do elemento.	Valores inteiros para a largura e a altura, respectivamente, separados por vírgula.
Background	A cor usada como fundo no botão.	Uma das cores pré-definidas: **black**, **blue**, **cyan**, **darkGray**, **gray**, **green**, **lightGray**, **magenta**, **orange**, **pink**, **red**, **white**, **yellow**. A cor pode, também, ser especificada pela coordenada inteira RGB (entre 0 e 255), mais uma coordenada de transparência opcional.
Foreground	A cor usada para o texto exibido pelo botão.	Os mesmos valores utilizados em **Background**.
Font	O tipo de fonte exibido pelo botão.	O estilo pode ser um dos tipos: **plain**, **bold**, **italic**, **bold**, **italic**.
Tooltip	O texto exibido quando o cursor do mouse é colocado sobre o botão.	Constante ou variável do tipo **String**.

6.2.3 Elemento *CheckBox*

Ícone: 

Texto:

Descrição: Um elemento do tipo *CheckBox* é usado para mostrar e modificar um valor do tipo booleano. Ele pode mostrar um texto, uma imagem, ou ambos. Esse elemento invoca a ação quando o valor é modificado. Além do mais, ações individuais podem ser especificadas para o caso em que o valor selecionado deve ser do tipo verdadeiro (*true*) ou falso (*false*).

Os elementos desse tipo podem ser desabilitados usando a propriedade "Enabled". Neste caso, a aparência acinzentada fica clareada.

Tabela de Propriedades		
Nome	Descrição	Valores aceitos pelas variáveis
Variable (Variável)		
Variable	A variável que deve ser editada.	Variável do tipo **boolean**.
Selected	Estado inicial da variável.	Uma variável booleana ou uma das constantes **true** ou **false**.
Decoration (Decoração)		
Text	O texto exibido pelo elemento.	Constante ou variável do tipo **String**.
Image	A imagem exibida pelo elemento.	Constante ou variável do tipo **String** correspondente a uma imagem com a extensão **GIF** ou a uma imagem animada do tipo **GIF**. A string indica o caminho para o arquivo da imagem correspondente. O caminho pode ser do diretório de trabalho ou por uma URL da internet.
Sel. Image	A imagem exibida pelo elemento quando o seu valor é **true**.	Constante ou variável do tipo **String** correspondente a uma imagem com a extensão **GIF** ou a uma imagem animada do tipo **GIF**. A string indica o caminho para o arquivo da imagem correspondente. O caminho pode ser do diretório de trabalho ou por uma URL da internet.
Mnemonic	A tecla que (juntamente com Alt) ativa esse menu.	**String** de um só caractere.
Alignment	Como se deve alinhar o texto no botão.	Uma das constantes: **left**, **center** ou **right**.
Interaction (Interação)		
Enabled	Se o elemento responde à interação do usuário.	Variável do tipo **boolean** ou uma das constantes **true** ou **false**.
Action	A ação que invoca quando o botão é clicado.	Código Java para invocar a ação.
Action On	O tipo de ação invocada quando o valor mudar para **true**.	Código Java para invocar a ação.
Action Off	O tipo de ação invocada quando o valor mudar para **false**.	Código Java para invocar a ação.

Graphical Aspect (Aspecto gráfico)		
Size	O tamanho do elemento.	Valores inteiros para a largura e a altura, respectivamente, separados por vírgula.
Background	A cor usada como fundo no botão.	Uma das cores pré-definidas: **black**, **blue**, **cyan**, **darkGray**, **gray**, **green**, **lightGray**, **magenta**, **orange**, **pink**, **red**, **white**, **yellow**. A cor pode, também, ser especificada pela coordenada inteira RGB (entre 0 e 255), mais uma coordenada de transparência opcional.
Foreground	A cor usada para o texto exibido pelo botão.	Os mesmos valores utilizados em **Background**.
Font	O tipo de fonte exibido pelo botão.	O estilo pode ser um dos tipos: **plain**, **bold**, **italic**, **bold**, **italic**.
Tooltip	O texto exibido quando o cursor do mouse é colocado sobre o botão.	Constante ou variável do tipo **String**.

6.2.4 Elemento *RadioButton*

Ícone: 

Texto:

Descrição: O *RadioButton* é um elemento básico utilizado para exibir e modificar um valor do tipo booleano. Nesse sentido, ele é semelhante ao botão *CheckBox*. A diferença entre eles está relacionada ao aspecto e ao comportamento do grupo. Quando houver mais de um botão no mesmo contêiner, um botão ativado desmarca automaticamente todos os outros.

O *RadioButton* pode exibir uma texto, uma imagem, ou ambos. Eles podem invocar uma ação quando um valor é mudado. Além disso, ações individuais podem ser especificadas para os casos nos quais um valor é selecionado para ser **true** ou **false**.

Esses elementos podem ser desabilitados usando a propriedade "*Enabled*". Neste caso, a aparência acinzentada fica clareada.

Tabela de Propriedades		
Nome	Descrição	Valores aceitos pelas variáveis
Variable (Variável)		
Variable	A variável que deve ser editada.	Variável do tipo **boolean**.
Selected	Estado inicial da variável.	Uma variável booleana ou uma das constantes **true** ou **false**.
Decoration (Decoração)		
Text	O texto exibido pelo elemento.	Constante ou variável do tipo **String**.
Image	A imagem exibida pelo elemento.	Constante ou variável do tipo **String** correspondente a uma imagem com a extensão **GIF** ou a uma imagem animada do tipo **GIF**. A string indica o caminho para o arquivo da imagem correspondente. O caminho pode ser do diretório de trabalho ou por uma URL da internet.
Sel. Image	A imagem exibida pelo elemento quando o seu valor é **true**.	Constante ou variável do tipo **String** correspondente a uma imagem com a extensão **GIF** ou a uma imagem animada do tipo **GIF**. A string indica o caminho para o arquivo da imagem correspondente. O caminho pode ser do diretório de trabalho ou por uma URL da internet.
Mnemonic	A tecla que (juntamente com Alt) ativa esse menu.	**String** de um só caractere.
Alignment	Como se deve alinhar o texto no botão.	Uma das constantes: **left**, **center** ou **right**.
Interaction (Interação)		
Enabled	Se o elemento responde à interação do usuário.	Variável do tipo **boolean** ou uma das constantes **true** ou **false**.
Action	A ação que invoca quando o botão é clicado.	Código Java para invocar a ação.
Action On	O tipo de ação invocada quando o valor mudar para **true**.	Código Java para invocar a ação.
Action Off	O tipo de ação invocada quando o valor mudar para **false**.	Código Java para invocar a ação.
Graphical Aspect (Aspecto gráfico)		

Size	O tamanho do elemento.	Valores inteiros para a largura e a altura, respectivamente, separados por vírgula.
Background	A cor usada como fundo no botão.	Uma das cores pré-definidas: **black**, **blue**, **cyan**, **darkGray**, **gray**, **green**, **lightGray**, **magenta**, **orange**, **pink**, **red**, **white**, **yellow**. A cor pode, também, ser especificada pela coordenada inteira RGB (entre 0 e 255), mais uma coordenada de transparência opcional.
Foreground	A cor usada para o texto exibido pelo botão.	Os mesmos valores utilizados em **Background**.
Font	O tipo de fonte exibido pelo botão.	O estilo pode ser um dos tipos: **plain**, **bold**, **italic**, **bold**, **italic**.
Tooltip	O texto exibido quando o cursor do mouse é colocado sobre o botão.	Constante ou variável do tipo **String**.

6.3 ELEMENTOS DA ABA *"INPUT AND OUTPUT"* DO GRUPO *"INTERFACE"*

6.3.1 Elemento *Bar*

Ícone: 

Texto: *A bar that displays a value.*

Descrição: O *Bar* é um elemento básico usado para exibir um valor numérico. O valor é mostrado de forma gráfica usando uma barra que é preenchida entre dois valores extremos. Ele não pode ser modificado de forma interativa. Se a propriedade "*Format*" não estiver vazia, o valor também é exibido na forma de texto.

Tabela de Propriedades		
Nome	Descrição	Valores aceitos pelas variáveis
Variable (Variável)		
Variable	A variável exibida com esse elemento.	Variável do tipo **double** ou **int**.

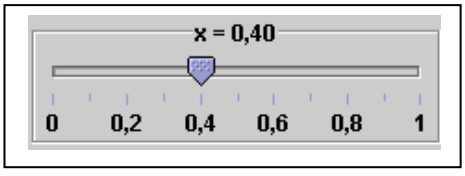
Minimum	O menor valor que pode ser selecionado.	Alguma constante ou variável do tipo **double** ou **int**.
Maximum	O maior valor que pode ser selecionado.	Alguma constante ou variável do tipo **double** ou **int**.
Format	O formato para a visualização do valor na forma texto.	Qualquer valor aceito pela classe **java.text.DecimalFormat**.
Orientation	A direção do elemento.	Uma das duas constantes: **HORIZONTAL** ou **VERTICAL**.
Graphical Aspect (Aspecto gráfico)		
Size	O tamanho do elemento.	Valores inteiros para a largura e a altura, respectivamente, separados por vírgula.
Background	A cor usada como fundo no botão.	Uma das cores pré-definidas: **black**, **blue**, **cyan**, **darkGray**, **gray**, **green**, **lightGray**, **magenta**, **orange**, **pink**, **red**, **white**, **yellow**. A cor pode, também, ser especificada pela coordenada inteira RGB (entre 0 e 255), mais uma coordenada de transparência opcional.
Foreground	A cor usada para o texto exibido pelo botão.	Os mesmos valores utilizados em **Background**.
Font	O tipo de fonte exibido pelo botão.	O estilo pode ser um dos tipos: **plain**, **bold**, **italic**, **bold**, **italic**.
Tooltip	O texto exibido quando o cursor do mouse é colocado sobre o botão.	Constante ou variável do tipo **String**.

6.3.2 Elemento *Slider*

Ícone: 

Texto: *A slider to display and modify a value.*

Descrição: Um *Slider* (controle deslizante) é um elemento que exibe e permite editar um valor numérico. O valor é selecionado através do movimento do cursor



de um extremo a outro. Se a propriedade de formato ("*Format*") não estiver vazia, o valor é também exibido na forma de texto.

Tabela de Propriedades		
Nome	Descrição	Valores aceitos pelas variáveis
Variable (Variável)		
Variable	A variável exibida e modificada.	Variável do tipo **double** ou **int**.
Value	Um valor inicial para a variável.	Alguma constante ou variável do tipo **double** ou **int**.
Minimum	O menor valor que pode ser selecionado.	Alguma constante ou variável do tipo **double** ou **int**.
Maximum	O maior valor que pode ser selecionado.	Alguma constante ou variável do tipo **double** ou **int**.
Format	O formato para a visualização do valor na forma texto.	Qualquer valor aceito pela classe **java.text.DecimalFormat**.
Orientation	A direção do elemento.	Uma das duas constantes: **HORIZONTAL** ou **VERTICAL**.
Ticks (Marcas)		
Ticks	Número de ticks (marcas) que deve ser mostrado no intervalo.	Alguma constante ou variável do tipo **int**.
Ticks Format	O formato dos ticks.	Qualquer valor aceito pela classe **java.text.DecimalFormat**.
Closest	Se o valor selecionado deve ser ajustado ao tick mais próximo.	Uma variável booleana ou uma constante **true** ou **false**.
Interaction (Interação)		
Enabled	Se o elemento exibido pode ser modificado ou não.	Variável do tipo **boolean** ou uma das constantes **true** ou **false**.
On Press	A ação que invoca quando o slider é pressionado.	Código Java para invocar a ação.
On Drag	O ação invocada quando o slider é movido.	Código Java para invocar a ação.
On Release	O tipo de ação invocada quando o cursor for liberado.	Código Java para invocar a ação.
Graphical Aspect (Aspecto gráfico)		
Size	O tamanho do elemento.	Valores inteiros para a largura e a altura,

		respectivamente, separados por vírgula.
Background	A cor usada como fundo no botão.	Uma das cores pré-definidas: **black**, **blue**, **cyan**, **darkGray**, **gray**, **green**, **lightGray**, **magenta**, **orange**, **pink**, **red**, **white**, **yellow**. A cor pode, também, ser especificada pela coordenada inteira RGB (entre 0 e 255), mais uma coordenada de transparência opcional.
Foreground	A cor usada para o texto exibido pelo botão.	Os mesmos valores utilizados em **Background**.
Font	O tipo de fonte exibido pelo botão.	O estilo pode ser um dos tipos: **plain**, **bold**, **italic**, **bold**, **italic**.
Tooltip	O texto exibido quando o cursor do mouse é colocado sobre o botão.	Constante ou variável do tipo **String**.

6.3.3 Elemento *Field*

Ícone: 

Texto: *A text field to display and modify a value.*

Descrição: O elemento *NumberField* exibe e permite editar valores numéricos. O valor numérico é exibido de acordo com o formato (propriedade "*Format*").

Ao editar o valor apresentado, a mudança só é aceita quando a tecla Enter for acionada. Para deixar isso evidente, o fundo do campo mudará de cor enquanto se digita o novo valor e, finalizada a ação, a cor de fundo original reaparece, indicando que o novo valor foi aceito. Se o valor introduzido estiver com o formato incorreto (isto é, não pode ser reconhecido através da indicação do "*Format*"), então o valor é rejeitado e o fundo será exibido em vermelho.

Tabela de Propriedades		
Nome	Descrição	Valores aceitos pelas variáveis
Main (Principal)		
Variable	A variável exibida é modificada usando esse elemento.	Variável do tipo **double** ou **int**.
Value	Um valor inicial para a opção.	Alguma constante ou variável do tipo **double** ou **int**.
Format	O formato exibido pela variável.	Algum valor aceito pela classe **java.text.DecimalFormat**. Por default o formato é **0.000**.
Editable	Se o valor exibido pode ser modificado ou não.	Uma variável booleana ou uma das constantes **true** ou **false**.
Action	A ação invocada quando a opção é selecionada	O código Java para invocar a ação.
Graphical Aspect (Aspecto gráfico)		
Size	O tamanho do elemento.	Valores inteiros para a largura e a altura, respectivamente, separados por vírgula.
Background	A cor usada como fundo no botão.	Uma das cores pré-definidas: **black**, **blue**, **cyan**, **darkGray**, **gray**, **green**, **lightGray**, **magenta**, **orange**, **pink**, **red**, **white**, **yellow**. A cor pode, também, ser especificada pela coordenada inteira RGB (entre 0 e 255), mais uma coordenada de transparência opcional.
Foreground	A cor usada para o texto exibido pelo botão.	Os mesmos valores utilizados em **Background**.
Font	O tipo de fonte exibido pelo botão.	O estilo pode ser um dos tipos: **plain**, **bold**, **italic**, **bold**, **italic**.
Tooltip	O texto exibido quando o cursor do mouse é colocado sobre o botão.	Constante ou variável do tipo **String**.

6.3.4 Elemento *TextField*

Ícone: 

Texto: *A text field to display and modify a string.*

Descrição: Um *TextField* exibe e permite editar um valor do tipo *String*. O valor exibido não é formatado de modo algum e aparece "como está". O elemento chama uma ação quando o valor apresentado é modificado.

Ao editar o valor apresentado, a mudança só é aceita quando a tecla *Enter* for acionada. Para deixar isso evidente, o fundo do campo mudará de cor enquanto se digita o novo valor e, finalizada a ação, a cor de fundo original reaparece, indicando que o novo valor foi aceito.

Tabela de Propriedades		
Nome	Descrição	Valores aceitos pelas variáveis
Main (Principal)		
Variable	A variável é exibida e modificada usando-se esse elemento.	Variável do tipo **String**.
Value	Um valor inicial para a opção.	Alguma constante ou variável do tipo **String**.
Editable	Se o valor exibido pode ser modificado ou não.	Uma variável booleana ou uma das constantes **true** ou **false**.
Action	A ação invocada quando a opção é selecionada	O código Java para invocar a ação.
Graphical Aspect (Aspecto gráfico)		
Size	O tamanho do elemento.	Valores inteiros para a largura e a altura, respectivamente, separados por vírgula.
Background	A cor usada como fundo no botão.	Uma das cores pré-definidas: **black**, **blue**, **cyan**, **darkGray**, **gray**, **green**, **lightGray**, **magenta**, **orange**, **pink**, **red**, **white**, **yellow**. A cor pode, também, ser especificada pela coordenada inteira RGB (entre 0 e 255), mais uma coordenada de transparência opcional.

Foreground	A cor usada para o texto exibido pelo botão.	Os mesmos valores utilizados em **Background**.
Font	O tipo de fonte exibido pelo botão.	O estilo pode ser um dos tipos: **plain**, **bold**, **italic**, **bold**, **italic**.
Tooltip	O texto exibido quando o cursor do mouse é colocado sobre o botão.	Constante ou variável do tipo **String**.

6.4 ELEMENTOS DA ABA *"BASIC 2D DRAWABLES"* DO GRUPO *"2D DRAWABLES"*

6.4.1 Elemento *Particle*

Ícone: 

Texto: *An interactive shape.*

Descrição: O elemento *Particle* é usado para representar uma forma geométrica simples como, por exemplo, um retângulo ou uma elipse, numa dada coordenada especificada no painel de desenho (D*rawingPpanel*). A forma geométrica é desenhada nas coordenadas fornecidas pelo usuário com o tamanho especificado em cada direção (para formas em 3D, tamanhos maiores de X e de Y são usados para as dimensões horizontais). A esse elemento, pode-se, também, aplicar um fator de rotação ou de ampliação (*zoom*).

Esses elementos respondem à interação do usuário através da mudança de sua posição (alteração das coordenadas). A posição precisa da forma geométrica com as respectivas coordenadas pode ser escolhida entre vários valores padronizados.

Tabela de Propriedades		
Nome	Descrição	Valores aceitos pelas variáveis
Position and Size (Posição e Tamanho)		
X	A coordenada X do elemento.	Uma constante ou variável do tipo **double** ou **int**.
Y	A coordenada Y do elemento.	Uma constante ou variável do tipo **double** ou **int**.
Z	A coordenada Z do elemento.	Uma constante ou variável do tipo **double** ou **int**.

Size X	O tamanho do elemento na direção X.	Uma constante ou variável do tipo **double** ou **int**.
Size Y	O tamanho do elemento na direção Y.	Uma constante ou variável do tipo **double** ou **int**.
Size Z	O tamanho do elemento na direção Z.	Uma constante ou variável do tipo **double** ou **int**.
Scale X	O fator de ampliação (zoom) do elemento na direção X.	Uma constante ou variável do tipo **double** ou **int**.
Scale Y	O fator de ampliação (zoom) do elemento na direção Y.	Uma constante ou variável do tipo **double** ou **int**.
Scale Z	O fator de ampliação (zoom) do elemento na direção Z.	Uma constante ou variável do tipo **double** ou **int**.
Visibility and Interaction (Visibilidade e Interação)		
Visible	A visibilidade do elemento.	Uma variável booleana ou uma das constantes **true** ou **false**.
Enabled	Se o elemento responde à interação do usuário.	Uma variável booleana ou uma das constantes **true** ou **false**.
Actions (Ações)		
On Press	A ação para invocar quando o elemento é pressionado.	O código Java que invoca a ação.
On Drag	A ação para invocar quando o elemento é movido.	O código Java que invoca a ação.
On Release	A ação para invocar quando o elemento é liberado (solto).	O código Java que invoca a ação.
On Enter	A ação para invocar quando o ponteiro entra no elemento.	O código Java que invoca a ação.
On Exit	A ação para invocar quando o ponteiro deixa o elemento.	O código Java que invoca a ação.
Graphical Aspect (Aspecto gráfico)		
Style	A forma gráfica para mostrar o elemento.	Uma variável das seguintes constantes: **NONE**. Um simples ponto é desenhado. (Equivalente a inteiro 0) **ELLIPSE** (Equivalente a inteiro 1) **RECTANGLE** (Equivalente a inteiro 2) **ROUND_RECTANGLE** (Equivalente a inteiro 3)

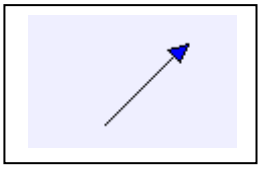
		WHEEL. Uma elipse com seus eixos coordenados. (Equivalente a inteiro 4) Constantes e variáveis do tipo inteira também são aceitas, usando qualquer dos valores indicados acima entre parêntesis. Isso permite a troca da forma durante a execução da tarefa.
Position	A posição da forma do desenho relativa às coordenadas do elemento.	Uma das seguintes constantes: **CENTERED**. A forma é desenhada com o seu ponto central sobre o elemento. **NORTH**. A forma é desenha com o seu ponto médio superior nas coordenadas do elemento. (Equivalente a inteiro 1) **SOUTH**. A forma é desenha com o seu ponto médio inferior nas coordenadas do elemento. (Equivalente a inteiro 2) **EAST**. A forma é desenha com o seu ponto médio direito nas coordenadas do elemento. (Equivalente a inteiro 3) **WEST**. A forma é desenha com o seu ponto médio esquerdo nas coordenadas do elemento. (Equivalente a inteiro 4) **NORTH_EAST**. A forma é desenhada com o seu ponto superior direito nas coordenadas do elemento. (Equivalente a inteiro 5) **NORTH_WEST**. A forma é desenhada com o seu ponto superior esquerdo nas coordenadas do elemento. (Equivale a inteiro 6) **SOUTH_EAST**. A forma é desenhada com o seu ponto inferior direito nas coordenadas do elemento. (Equivale a inteiro 7) **SOUTH_WEST**. A forma é desenhada com o seu ponto inferior esquerdo nas

		coordenadas do elemento. (Equivale a inteiro 8) Constantes e variáveis do tipo inteira também são aceitas, usando qualquer dos valores indicados acima entre parêntesis. Isso permite a troca da posição durante a execução da tarefa.
Rotate	O ângulo de rotação (no sentido anti-horário) aplicado ao elemento.	Uma constante ou variável do tipo **double** para o ângulo em radianos, ou uma constante ou variável do tipo **int** para o ângulo em graus.
Fill Color	A cor usada para preencher o elemento. O valor especial **null** (nulo) desenha um elemento vazio.	Uma variável do tipo **Object** da classe **java.awt.Color** ou uma das cores pré-definidas: **black**, **blue**, **cyan**, **darkGray**, **gray**, **green**, **lightGray**, **magenta**, **orange**, **pink**, **red**, **white**, **yellow**. A cor pode, também, ser especificada pela coordenada inteira RGB (entre 0 e 255), mais uma coordenada de transparência opcional. O valor por default é decidido pelo sistema.
Edge Color	A cor utilizada para a borda do elemento. O valor especial **null** (nulo) desenha o elemento sem borda.	Os mesmos valores de **Fill Color**.
Stroke	A espessura das linhas do elemento.	Qualquer constante ou variável do tipo **double** ou **int**. O valor de default é 1. Variáveis **Object** da classe **java.awt.Stroke** também são aceitas.

6.4.2 Elemento *Arrow*

Ícone: 

Texto: *An interactive vector (or line).*



Descrição: O elemento *Arrow* é usado para mostrar um segmento de reta ou um vetor de tamanho específico em uma posição de coordenada escolhida pelo usuário.

Esse elemento pode ser ampliado através de um fator de ampliação (zoom) e ele responde à interação do usuário ao clicar com o *mouse* na extremidade (*head*). Assim, ao arrastar a seta do *mouse*, o seu tamanho do elemento *Arrow* se altera. Esse procedimento pode ser efetuado na outra extremidade (*tail*) do elemento. Da mesma forma, é possível mover o elemento, isto é, modificar a sua posição.

Tabela de Propriedades		
Nome	Descrição	Valores aceitos pelas variáveis
Position and Size (Posição e Tamanho)		
X	A coordenada X do elemento.	Uma constante ou variável do tipo **double** ou **int**.
Y	A coordenada Y do elemento.	Uma constante ou variável do tipo **double** ou **int**.
Z	A coordenada Z do elemento.	Uma constante ou variável do tipo **double** ou **int**.
Size X	O tamanho do elemento na direção X.	Uma constante ou variável do tipo **double** ou **int**.
Size Y	O tamanho do elemento na direção Y.	Uma constante ou variável do tipo **double** ou **int**.
Size Z	O tamanho do elemento na direção Z.	Uma constante ou variável do tipo **double** ou **int**.
Scale X	O fator de ampliação (zoom) do elemento na direção X.	Uma constante ou variável do tipo **double** ou **int**.
Scale Y	O fator de ampliação (zoom) do elemento na direção Y.	Uma constante ou variável do tipo **double** ou **int**.
Scale Z	O fator de ampliação (zoom) do elemento na direção Z.	Uma constante ou variável do tipo **double** ou **int**.
Visibility and Interaction (Visibilidade e Interação)		
Visible	A visibilidade do elemento.	Uma variável booleana ou uma das constantes **true** ou **false**.
Enabled	Se o elemento responde à	Uma variável booleana ou uma das

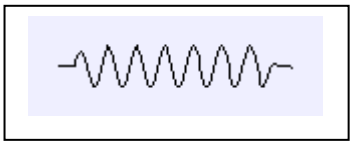
	interação do usuário.	constantes **true** ou **false**.
Movable	Se o elemento responde à interação do usuário sobre a sua origem.	Uma variável booleana ou uma das constantes **true** ou **false**.
Actions (Ações)		
On Press	A ação para invocar quando o elemento é pressionado.	O código Java que invoca a ação.
On Drag	A ação para invocar quando o elemento é movido.	O código Java que invoca a ação.
On Release	A ação para invocar quando o elemento é liberado (solto).	O código Java que invoca a ação.
On Enter	A ação para invocar quando o ponteiro entra no elemento.	O código Java que invoca a ação.
On Exit	A ação para invocar quando o ponteiro deixa o elemento.	O código Java que invoca a ação.
Graphical Aspect (Aspecto gráfico)		
Style	A forma gráfica para mostrar o elemento.	Uma variável das seguintes constantes: **ARROW**. (Equivalente a inteiro 0) **SEGMENT**. (Equivalente a inteiro 1) **BOX**. Um segmento com uma pequena caixa em sua extremidade. (Equivalente a inteiro 2) Constantes e variáveis do tipo inteira também são aceitas, usando qualquer dos valores indicados acima entre parêntesis. Isso permite a troca da forma durante a execução da tarefa.
Line Color	A cor usada para as linhas do elemento. O valor especial **null** (nulo) desenha o elemento sem linha.	Veja os valores da propriedade **Fill Color**.
Fill Color	A cor usada para preencher o elemento. O valor especial **null** (nulo) desenha um elemento vazio.	Uma variável do tipo **Object** da classe **java.awt.Color** ou uma das cores pré-definidas: **black**, **blue**, **cyan**, **darkGray**, **gray**, **green**, **lightGray**, **magenta**, **orange**, **pink**, **red**, **white**, **yellow**. A cor

		pode, também, ser especificada pela coordenada inteira RGB (entre 0 e 255), mais uma coordenada de transparência opcional. O valor por default é decidido pelo sistema.
Stroke	A espessura das linhas do elemento.	Qualquer constante ou variável do tipo **double** ou **int**. O valor de default é 1. Variáveis **Object** da classe **java.awt.Stroke** também são aceitas.
Resolution	Como subdividir o elemento em pequenos pedaços. Esse procedimento é útil, apenas, nos modos de desenho 3D para melhorar o resultado do algoritmo de remoção de linhas escondidas.	Uma constante ou variável do tipo **int** indicando o número de pedaços que se deseja dividir o elemento. O valor de default é 1. Uma constante ou variável do tipo **double** indicando o comprimento máximo de cada pedaço individual.

6.4.3 Elemento *Spring*

Ícone: 

Texto: *An interactive spring.*



Descrição: O elemento *Spring* permite exibir uma mola em uma coordenada especificada no painel de desenho e com um tamanho específico. Ele pode ser a ser ampliado através de um fator de ampliação (*zoom*) e responde à interação do usuário. Para isso, o usuário deve clicar com o *mouse* na extremidade (head) e ao arrastá-lo, o seu tamanho se altera. O mesmo procedimento pode ser efetuado na outra extremidade.

Tabela de Propriedades		
Nome	Descrição	Valores aceitos pelas variáveis
Position and Size (Posição e Tamanho)		
X	A coordenada X do elemento.	Uma constante ou variável do tipo **double**

		ou **int**.
Y	A coordenada Y do elemento.	Uma constante ou variável do tipo **double** ou **int**.
Z	A coordenada Z do elemento.	Uma constante ou variável do tipo **double** ou **int**.
Size X	O tamanho do elemento na direção X.	Uma constante ou variável do tipo **double** ou **int**.
Size Y	O tamanho do elemento na direção Y.	Uma constante ou variável do tipo **double** ou **int**.
Size Z	O tamanho do elemento na direção Z.	Uma constante ou variável do tipo **double** ou **int**.
Scale X	O fator de ampliação (zoom) do elemento na direção X.	Uma constante ou variável do tipo **double** ou **int**.
Scale Y	O fator de ampliação (zoom) do elemento na direção Y.	Uma constante ou variável do tipo **double** ou **int**.
Scale Z	O fator de ampliação (zoom) do elemento na direção Z.	Uma constante ou variável do tipo **double** ou **int**.
Visibility and Interaction (Visibilidade e Interação)		
Visible	A visibilidade do elemento.	Uma variável booleana ou uma das constantes **true** ou **false**.
Enabled	Se o elemento responde à interação do usuário.	Uma variável booleana ou uma das constantes **true** ou **false**.
Movable	Se o elemento responde à interação do usuário sobre a sua origem.	Uma variável booleana ou uma das constantes **true** ou **false**.
Actions (Ações)		
On Press	A ação para invocar quando o elemento é pressionado.	O código Java que invoca a ação.
On Drag	A ação para invocar quando o elemento é movido.	O código Java que invoca a ação.
On Release	A ação para invocar quando o elemento é liberado (solto).	O código Java que invoca a ação.
On Enter	A ação para invocar quando o ponteiro entra no elemento.	O código Java que invoca a ação.
On Exit	A ação para invocar quando o ponteiro deixa o elemento.	O código Java que invoca a ação.

Graphical Aspect (Aspecto gráfico)		
Radius	O raio da mola.	Uma constante ou variável do tipo **double** ou **int**.
Color	A cor usada para desenhar o elemento.	Uma variável do tipo **Object** da classe **java.awt.Color** ou uma das cores pré-definidas: **black**, **blue**, **cyan**, **darkGray**, **gray**, **green**, **lightGray**, **magenta**, **orange**, **pink**, **red**, **white**, **yellow**. A cor pode, também, ser especificada pela coordenada inteira RGB (entre 0 e 255), mais uma coordenada de transparência opcional. O valor por default é decidido pelo sistema.
Stroke	A espessura das linhas do elemento.	Qualquer constante ou variável do tipo **double** ou **int**. O valor de default é 1. Variáveis **Object** da classe **java.awt.Stroke** também são aceitas.
Resolution	Como subdividir o elemento em pequenos pedaços. Esse procedimento é útil, apenas, nos modos de desenho 3D para melhorar o resultado do algoritmo de remoção de linhas escondidas.	Uma constante ou variável do tipo **int** indicando o número de pedaços que se deseja dividir o elemento. O valor de default é 1. Uma constante ou variável do tipo **double** indicando o comprimento máximo de cada pedaço individual.

6.4.4 Elemento *Trace*

Ícone: 

Texto: *A sequence of points.*

Descrição: O elemento *Trace* mostra uma sequencia de pontos em coordenadas especificadas no painel de desenho. Os pontos a serem exibidos são adicionados sequencialmente, um após

o outro, e podem ser visualizados por meio de marcadores ou conectá-los com os segmentos (ou ambos).

O elemento *Trace* pode ser programado para exibir um número máximo de pontos, caso em que, ele irá descartar valores antigos para acomodar os novos. No entanto, se a propriedade "*No Repeat*" for definida como “verdadeiro”, e o novo valor for idêntico ao anterior, então esse novo valor será ignorado. A recepção de dados também pode ser (geralmente temporária) desabilitada. O elemento pode, também, ser instruído a aceitar somente um ponto de um conjunto de pontos recebidos. Isso é útil no caso de o modelo gerar muitos pontos.

Tabela de Propriedades		
Nome	Descrição	Valores aceitos pelas variáveis
Input (Entrada)		
X	A coordenada X do novo ponto a ser adicionado na sequência.	Uma constante ou variável do tipo **double** ou **int**.
Y	A coordenada Y do novo ponto a ser adicionado na sequência.	Uma constante ou variável do tipo **double** ou **int**.
Z	A coordenada Z do novo ponto a ser adicionado na sequência.	Uma constante ou variável do tipo **double** ou **int**.
Position (Posição)		
Position X	A coordenada X do elemento como um todo.	Uma constante ou variável do tipo **double** ou **int**.
Position Y	A coordenada Y do elemento como um todo.	Uma constante ou variável do tipo **double** ou **int**.
Position Z	A coordenada Z do elemento como um todo.	Uma constante ou variável do tipo **double** ou **int**.
Configuration (Configuração)		
Points	Número máximo de pontos a serem mostrados. O valor especial **0** que não existe valor máximo.	Uma constante ou variável do tipo **int**.
Skip	O número de pontos a receber antes de aceitar um ponto.	Uma constante ou variável do tipo **int**.
Active	Se o traço aceita pontos.	Uma variável booleana ou uma das constantes **true** ou **false**.
No Repeat	Se deve ignorar um ponto de	Uma variável booleana ou uma das

	entrada idêntico ao último.	constantes **true** ou **false**.
Connected	Se deve conectar pontos de entrada com um segmento.	Uma variável booleana ou uma das constantes **true** ou **false**.
Visibility and Interaction (Visibilidade e Interação)		
Visible	A visibilidade do elemento.	Uma variável booleana ou uma das constantes **true** ou **false**.
Enabled	Se o elemento responde à interação do usuário.	Uma variável booleana ou uma das constantes **true** ou **false**.
Actions (Ações)		
On Press	A ação para invocar quando o elemento é pressionado.	O código Java que invoca a ação.
On Drag	A ação para invocar quando o elemento é movido.	O código Java que invoca a ação.
On Release	A ação para invocar quando o elemento é liberado (solto).	O código Java que invoca a ação.
On Enter	A ação para invocar quando o ponteiro entra no elemento.	O código Java que invoca a ação.
On Exit	A ação para invocar quando o ponteiro deixa o elemento.	O código Java que invoca a ação.
Graphical Aspect (Aspecto gráfico)		
Line Color	A cor usada para as linhas do elemento.	Uma variável do tipo **Object** da classe **java.awt.Color** ou uma das cores pré-definidas: **black**, **blue**, **cyan**, **darkGray**, **gray**, **green**, **lightGray**, **magenta**, **orange**, **pink**, **red**, **white**, **yellow**. A cor pode, também, ser especificada pela coordenada inteira RGB (entre 0 e 255), mais uma coordenada de transparência opcional. O valor por default é decidido pelo sistema.
Stroke	A espessura das linhas do elemento.	Qualquer constante ou variável do tipo **double** ou **int**. O valor de default é 1. Variáveis **Object** da classe **java.awt.Stroke** também são aceitas.
Maker Shape	A forma gráfica dos marcadores.	Uma variável das seguintes constantes:

		NONE. Um simples ponto é desenhado. (Equivalente a inteiro 0) **ELLIPSE** (Equivalente a inteiro 1) **RECTANGLE** (Equivalente a inteiro 2) **ROUND_RECTANGLE** (Equivalente a inteiro 3) **WHEEL**. Uma elipse com seus eixos coordenados. (Equivalente a inteiro 4) Constantes e variáveis do tipo inteira também são aceitas, usando qualquer dos valores indicados acima entre parêntesis. Isso permite a troca da forma durante a execução da tarefa.
Maker Size	O tamanho dos marcadores em pixels.	Uma constante ou variável do tipo **int**.
Maker Color	A cor dos marcadores.	Veja os valores da propriedade **Line Color**.
Position	A posição de desenho exata dos marcadores em relação aos pontos de entrada.	Uma das seguintes constantes: **CENTERED**. A forma é desenhada com o seu ponto central sobre o elemento. (Equivale a inteiro 0) **NORTH**. A forma é desenha com o seu ponto médio superior nas coordenadas do elemento. (Equivalente a inteiro 1) **SOUTH**. A forma é desenha com o seu ponto médio inferior nas coordenadas do elemento. (Equivalente a inteiro 2) **EAST**. A forma é desenha com o seu ponto médio direito nas coordenadas do elemento. (Equivalente a inteiro 3) **WEST**. A forma é desenha com o seu ponto médio esquerdo nas coordenadas do elemento. (Equivalente a inteiro 4) **NORTH_EAST**. A forma é desenhada com o seu ponto superior direito nas coordenadas do elemento. (Equivalente a

		inteiro 5) **NORTH_WEST**. A forma é desenhada com o seu ponto superior esquerdo nas coordenadas do elemento. (Equivale a inteiro 6) **SOUTH_EAST**. A forma é desenhada com o seu ponto inferior direito nas coordenadas do elemento. (Equivale a inteiro 7) **SOUTH_WEST**. A forma é desenhada com o seu ponto inferior esquerdo nas coordenadas do elemento. (Equivale a inteiro 8) Constantes e variáveis do tipo inteira também são aceitas, usando qualquer dos valores indicados acima entre parêntesis. Isso permite a troca da posição durante a execução da tarefa.
Rotate	O ângulo de rotação (no sentido anti-horário) aplicado aos marcadores.	Uma constante ou variável do tipo **double** para o ângulo em radianos, ou uma constante ou variável do tipo **int** para o ângulo em graus.
Memory	Esse é um inteiro que indica quantos conjuntos de dados o traço deve memorizar.	0: Mostra todos os conjuntos de dados antes de chamar *_resetView( )*. 1: Mostra só o conjunto de dados atuais (default). 2 ou mais: Mostra tantos conjuntos de dados indicados. Subsequentemente, os valores antigos são descartados durante a execução.
Mem Display	Quantos dados antigos são mostrados.	Uma das seguintes constantes: **SHOW_ALL**. Dados antigos são mostrados completamente. (Equivalente a inteiro 0) **AS_ADDED**. Dados antigos são mostrados ao mesmo tempo em que os

		novos são adicionados. (Equivalente a inteiro 1) **X_ORDER**. Todos os pontos antigos com uma coordenada X menor ou igual a coordenada X do último ponto novo é mostrado. (Equivalente a inteiro 2) **Y_ORDER**. Similar a **X_ORDER**. (Equivalente a inteiro 3) **Z_ORDER**. Similar a **X_ORDER**. (Equivalente a inteiro 4) Constantes e variáveis inteiras são aceitas usando um dos valores indicados entre parênteses (acima). Isso permite mudar o modo de exibição durante a execução.
Mem Color	A cor usada para exibir dados antigos.	Veja os valores da propriedade **Line Color**.

7 CONSIDERAÇÕES FINAIS AO PROFESSOR

Dissemos no início da apresentação deste texto que o EJS é uma ferramenta que exige tempo e dedicação daquele que queira criar e desenvolver suas próprias simulações. Assim, o professor do Ensino Médio precisa de uma orientação adequada para que se possa planejar, elaborar ações e executar o tipo de simulação que tem em mente.

Ao participar do ato de criação de uma simulação, o professor estará adquirindo uma melhor compreensão do assunto a ser ensinado, pois ele irá se deparar com conflitos cognitivos. Sem dúvida, conhecendo os próprios conflitos cognitivos, isso irá permitir-lhe planejamento de aulas mais elaboradas e, ao mesmo tempo, poderá propor atividades para os seus alunos possam compreender de forma diferenciada o assunto a ser estudado.

Com a utilização de simulações no Ensino de Física para alunos do Ensino Médio, o professor tem a oportunidade de desenvolver uma atividade mais próxima da realidade da produção científica e tecnológica, rompendo-se com o ensino tradicional. Para isso, o professor deverá mudar suas concepções e ideias pré-concebidas. Perceber que somente com atividades colaborativas, interdisciplinares, ele poderá obter resultados mais valiosos para o Ensino de Física.

Dentro da proposta de se utilizar simulações no ensino, um aspecto crucial ao se desenvolver o laboratório virtual é o de permitir a interatividade. Esse é, ainda, um recurso pouco explorado como atividade pedagógica. É justamente a interatividade que permitirá o estudante controlar, alterar, modificar parâmetros e verificar os efeitos decorrentes de suas ações. O estudante (aluno) terá a oportunidade de obter respostas quantitativas e qualitativas proporcionadas pelo sistema simulado.

Mas, uma interatividade pedagogicamente rica vai além da simples capacidade de alterar valores de parâmetros, rodar algoritmos, analisar e comparar respostas ou, ainda, repetir um procedimento. A interatividade pedagogicamente valiosa deve permitir tudo isso e, também, proporcionar a visualização da evolução do sistema físico sob diferentes aspectos do sistema ou mostrar a resposta, em tempo real, quando o usuário alterar um determinado parâmetro que governa a evolução do sistema que está sendo simulado. É justamente essa resposta imediata que iria ajudar o estudante a adquirir melhor entendimento e visão mais crítica das leis que governam o fenômeno físico simulado.

A combinação entre interatividade, visualização e gráficos fazem do EJS uma ferramenta especial que pode contribuir ao Ensino da Física. Ela merece ser estudada, mas, ao

mesmo tempo, exige dedicação e tempo de estudo para o professor dominar uma parcela de sua potencialidade. Acreditamos que o professor deverá ver nesse estudo e dedicação, um investimento próprio e uma motivação para mobilizar outros professores a participarem de projetos de construção de laboratório virtual. EJS permite a formação de equipes interdisciplinares de professores para a elaboração de tais projetos.

Repetindo-nos, esse texto está longe de estar concluído. Ele mostra a possibilidade de se elaborar atividades diferenciadas e com várias aplicações sob o ponto de vista pedagógico. Prosseguiremos na elaboração e atualização do texto sobre a ferramenta EJS, na sua divulgação e no incentivo à formação de equipes de professores que possam estar engajados no planejamento de atividades de simulação com real valor pedagógico.