UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
COORDENAÇÃO DE ANÁLISE E DESENVOLVIMENTO DE SISTEMAS
CURSO SUPERIOR DE TECNOLOGIA EM SISTEMAS DE INFORMAÇÃO

IGHOR ALEXANDRE MUDREY

USO DE FLASH PARA CRIAÇÃO DE ANIMAÇÕES GRÁFICAS DE PROPÓSITO DIDÁTICO

TRABALHO DE CONCLUSÃO DE CURSO

PONTA GROSSA
2012

IGHOR ALEXANDRE MUDREY

USO DE FLASH PARA CRIAÇÃO DE ANIMAÇÕES GRÁFICAS DE PROPÓSITO DIDÁTICO

Trabalho de conclusão de curso de graduação, apresentado à disciplina Trabalho de Diplomação, do curso Superior de Tecnologia em Sistemas de Informação da Coordenação de Análise e Desenvolvimento de Sistemas da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para a obtenção do título de Tecnólogo.

Orientador: Prof. Dr. André Koscianski

PONTA GROSSA

2012

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Ponta Grossa

Diretoria de Graduação e Educação Profissional

TERMO DE APROVAÇÃO

USO DE FLASH PARA CRIAÇÃO DE ANIMAÇÕES GRÁFICAS DE PROPÓSITO DIDÁTICO

por

IGHOR ALEXANDRE MUDREY

Este Trabalho de Conclusão de Curso (TCC) foi apresentado(a) em 29 de junho de 2012 como requisito parcial para a obtenção do título de Tecnólogo em Sistemas de Informação. O(a) candidato(a) foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
André Koscianski
Prof. Orientador

_____
Simone de Almeida
Membro titular

_____
Lourival Aparecido de Góis
Membro titular

_____
Helyane Bronoski Borges
Responsável pelos Trabalhos
de Conclusão de Curso

_____
Simone de Almeida
Coordenador do Curso
UTFPR - Campus Ponta Grossa

- O Termo de Aprovação assinado encontra-se na Coordenação do Curso -

AGRADECIMENTOS

Agradeço principalmente ao meu bom Deus, que me fortalece e me ampara nos momentos mais difíceis, o Deus que desce e me carrega quando vê que está muito complicado de passar pelo caminho das pedras e me conforta.

Ao meu orientador Dr. André Koscianski, pelas escolhas e definições deste trabalho e sua extrema paciência e dedicação.

Ao apoio e confiança do pai, mãe e madrinha.

Pela motivação e carinho da minha esposa Graceliz, que desde o início desta jornada, esteve presente, sempre acreditando no êxito final e em sucesso.

Pelo carinho e pelo crédito dado pelo professor João Paulo Aires, Psicóloga Cintia pela ajuda nas seções de auxilio e busca de respostas concisas sobre a vida.

Por todas as pessoas que torcem por mim, um muito obrigado e até a próxima fase.

“Agradeço todas as dificuldades que enfrentei; não fosse por elas, eu não teria saído do lugar. As facilidades nos impedem de caminhar. Mesmo as críticas nos auxiliam muito.”

(Allan Kardec)

RESUMO

MUDREY, Ighor Alexandre. **Uso de flash para criação de animações gráficas de propósito didático**, 2012. 48 f. Trabalho de conclusão de curso - Tecnologia de Sistemas de Informação, Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2012.

Este trabalho contém elementos de desenvolvimento de animações com propósito didático, de forma que descomplique o processo acadêmico sem fazer com que o mesmo perca a qualidade. Tem como objetivo desenvolver animações gráficas para melhor explicar o conteúdo dinâmico que em sala de aula é exibido de forma estática. Para a realização deste utilizou-se a pesquisa bibliográfica como base referencial na construção de dados para obtenção de animações gráficas, que solucionem as dificuldades de reprodução do conhecimento, de forma organizada e com aplicabilidade eficiente. A partir da realização deste trabalho foi possível desenvolver animações referentes ao conteúdo de escalonamento da disciplina de Sistemas Operacionais. Para tanto, ao analisar o conteúdo desenvolvido, foi possível concluir que a proposta é válida no que se refere à construção de material de apoio didático diferenciado, capaz de envolver os acadêmicos no processo de ensino aprendizagem de forma resolutiva e eficiente. Contudo, espera-se contribuir com projetos futuros ao fazer este trabalho.

Palavras-chave: Animações. Flash. Sistemas Operacionais. JVM. Tecnologia.

ABSTRACT

MUDREY, Ighor Alexandre. **Using Flash to create graphic animations of didactic purpose** 2012. 48 f. Final Paper – Technology in Information Systems, FederalTechnology University of Paraná, 2012.

This academic work contains elements of animations development with didactic purpose, with easy understanding without losing quality. It's aimed to develop graphic animations to better explain the dynamic content, which in classroom is displayed static. To accomplish this we used the literature as a base reference in the construction of data to obtain graphic animations, addressing the difficulties in the reproduction of knowledge in an organized and efficient applicability. To accomplish this we used the literature as a base reference in the construction of data to obtain graphic animations, addressing the difficulties in the reproduction of knowledge in an organized and efficient applicability. From this study it was possible to develop animations for the contents of the discipline of systems operational. For both developed content at analysis it was concluded that the proposal is valid with respect to the construction of differentiated teaching support material, able to involve students in teaching and learning process with a resulting and efficiency. However, we hope I can do to contribute to future projects.

Keywords: Animations. Flash. Operating Systems. JVM. Technology.

LISTA DE FIGURAS

LISTA DE SIGLAS

ADSL	Assymmetric Digital Subscriber Line
CPU	Unidade Central de Processamento
CS6	Creative Suite 6
CSS	Cascading Style Sheet
FIFO	First In First Out
FPS	Frames Por Segundo
GB	Giga Byte
HTML	Hyper Text Markup Language
I/O	Entrada e Saída
IHC	Interface Humano Computador
JVM	Java Virtual Machine
MB	Mega Byte
O.O	Orientação a Objetos
RAM	Random Access Memory
RIA	Rich Internet Aplication
SJF	Short Job First
SO	Sistema Operacional

SUMÁRIO

1 INTRODUÇÃO

As mudanças advindas da globalização, permitiram avanços no que se refere a utilização de *software* e *hardware*, tanto de cunho tecnológico quanto mercadológico, pois a tecnologia em redes de computadores atrelada à telefonia viabilizou a criação de algo sensacional para a época, a qual se denominou *ADSL* (*Assymmetric Digital Subscriber Line*), resultando na exploração de uma nova geração de oportunidades no mercado.

A exigência do público fez com que algo simples como códigos *HTML* (*Hyper Text Markup Language*), fossem vistos como uma forma pouco atrativa de ofertarem seus produtos, o que com as evoluções tecnológicas citadas anteriormente, possibilitou aos cientistas voltados para o mercado, pensar em algo novo que fosse mais atraente as mentes consumistas, surgindo então à tecnologia de programação *flash*. A evolução desta tecnologia é apresentada a seguir:

> 1996: Flash foi lançado como ferramenta de design;
> 1999: Flash Player atinge 92% dos internautas;
> 2000: Flash 5 com a linguagem de programação ActionScript 1;
> 2002: Macromedia cita pela primeira vez o termo RIA;
> 2004: Flash MX 2004 e Flash Player 8 com ActionScript 2 que possui alguns recursos de orientação a objetos, componentes na versão 2 e Flash Lite, para a disponibilização de aplicações para celulares e dispositivos móveis. Macromedia lança Flex 1 e cita pela primeira vez o temo Flash Platform;
> 2005: Adobe compra a Macromedia;
> 2006: Lançado o Flex 2 e Flash Player 9 com ActionScript 3 que é totalmente orientado a objetos. Foi criada uma nova IDE para o Flex baseado no Eclipse, o Flex Builder; (SALEIRO, apud FRETA; SILVA, 2007, p. 43)

O uso deste tipo de tecnologia ganhou espaço no mercado e foi de suma importância para o surgimento de outras linguagens mais aprimoradas, porém menos importantes, pois todas as demais vieram após o sucesso da linguagem *flash*, a qual é capaz de proporcionar aos consumidores a visualização de objetos de natureza estática em movimento.

Tal ação desperta uma sensação agradável e por consequência se torna o estopim para o crescimento do consumismo neste mercado, o qual foi “engolido” pela presença de imagens dinâmicas, que ao se movimentarem induziam ao consumismo.

Seguindo as evoluções tecnológicas, o projeto visa melhorar a qualidade do ensino, com a proposta de criar soluções interativas através de animações para que

o acadêmico do curso superior de analise e desenvolvimento de sistemas possa ter uma compreensão mais ampla dos assuntos abordados em sala de aula.

Desta forma este estudo visa trabalhar com animações no processo de ensino aprendizagem, a partir da linguagem de programação *flash* buscando nova alternativa, interagindo junto aos acadêmicos e facilitando sua compreensão do conteúdo estudado, desenvolvendo animações gráficas para melhor explicar o conteúdo dinâmico onde na sala de aula é exibido de forma estática em forma de *site*. Esta ação visa enriquecer a aula de Sistemas Operacionais, utilizando recursos visuais de modo que facilite a compreensão dos conteúdos.

1.1 OBJETIVO GERAL

Desenvolver animações gráficas para facilitar a explicação do conteúdo dinâmico ensinado de forma estática no que se refere a disciplina de Sistemas Operacionais.

1.2 OBJETIVOS ESPECÍFICOS

- Criar um site para exibir as animações gráficas;
- Conhecer a ferramenta *Swish Max*;
- Realizar estudo referente aos sistemas similares.

1.3 ESTRUTURA DO TRABALHO

O trabalho apresenta-se composto de seis capítulos, conforme apresentado a seguir:

No capítulo um, é realizado uma breve introdução, a qual é responsável pela descrição dos componentes do trabalho, onde retrata o início das tecnologias voltadas para a *ADSL,* com o intuito de amparar o objetivo geral, que são o desenvolvimento das animações utilizando a ferramenta flash, com o foco voltado para o ensino aprendizagem.

No capítulo dois, é o capítulo do embasamento teórico do trabalho, onde o mesmo estuda questões voltadas para a ferramenta, bem como na sua utilização e questões teóricas envolvendo as animações.

O capítulo três, envolve o estudo de viabilidade do projeto, onde o mesmo procura estudar aspectos de trabalhos similares, além de verificar a possibilidade da utilização de parte dos mesmos na construção das animações.

O capítulo quatro, apresenta a ferramenta onde serão desenvolvidas as animações, o projeto do *site* que tem a função de exibi-las, tratando as questões de IHC utilizadas na construção do *site*.

O quinto capítulo refere-se ao desenvolvimento das animações, onde cada escalonamento abordado contém trechos de códigos com a explicação referente a funcionalidade da linha de código exibida.

O último capítulo destina-se a apresentar as considerações finais da pesquisa, seus resultados e futuros trabalhos relacionados ao tema.

2 REVISÃO DE LITERATURA

Para a realização da pesquisa é necessário embasamento teórico suficiente que qualifique o projeto, para tanto é necessário analisar projetos e sistemas similares. Por meio de pesquisa bibliográfica foi encontrada similaridade deste objeto de estudo com o projeto Sosim.

O referido projeto simula operações realizadas pelo sistema operacional, deixando-as visíveis na tela e possibilitando que o aluno visualize e compreenda o assunto abordado em sala de aula.

Outro sistema analisado é o Marechal, pelo qual se constata similaridade, pois o mesmo tem propósito igual ao projeto deste trabalho de conclusão de curso, porém o sistema marechal foi construído no *Game Maker* através de um jogo de perguntas e respostas, o qual aborda alocação de recursos.

Cabe ainda para realização desta pesquisa, analisar o sistema operacional *Minix* o qual foi desenvolvido por Andrew Tanembaum em 1987 e tem como função auxiliar na prática a sua teoria.

É importante abordar e conhecer conceitos como JAVA, JVM e Máquina Virtual, pois a proposta da construção das animações inerentes a este projeto se dá em uma estrutura deste porte e podem ser adaptadas a qualquer tipo de navegador.

Por se tratar de um projeto que visa auxiliar acadêmicos do curso de Tecnologia em Análise e Desenvolvimento de Sistema, é necessário fazer uma breve descrição referente ao ensino da disciplina de Sistemas Operacionais, a fim de estabelecer as reais necessidades da aplicação destas animações no dia a dia educacional.

2.1 ENSINO DE SISTEMAS OPERACIONAIS

O ensino da disciplina de sistemas operacionais pode ser considerado como uma tarefa árdua, pois além de transmitir o conhecimento teórico para o aluno, o professor precisa fazer com que o mesmo construa um censo crítico de como se comporta um sistema operacional.

Dentre as habilidades necessárias a serem desenvolvidas pelo acadêmico ao término do curso, é conhecer o comportamento do mecanismo que está

transparente a qualquer sistema operacional, pois do contrário o mesmo não terá aproveitado de forma satisfatória impossibilitando a construção de um sistema gerenciador por exemplo.

Um dos elementos contidos nessa tarefa é que um sistema operacional não funciona de forma inerte, e sim de forma ativa, onde o mesmo tem a função de controlar, automatizar processos, gerenciar tarefas. Mas nada disso é visível sendo esta a primeira interrogação que um acadêmico costuma ter, como isso é possível?

2.2 APLICAÇÕES WEB

Aplicações *web* são aplicações cujas funções têm seu funcionamento através da internet, ou seja, o sistema é carregado por meio de um *browser.*

Este tipo de aplicação tornou-se possível graças à evolução da internet que trabalha atualmente em alta velocidade, devido ao uso de fibra óptica.

Ao compararmos este tipo de aplicação com as desenvolvidas para ambiente *desktop* a principal vantagem é que não precisa ser instalada, basta acessar o endereço de onde foi programada e posteriormente localizar o sistema no servidor em que foi desenvolvida, acarretando em economia em espaço em disco, liberação de memória e pelo fato de que o computador não é exposto à sobrecarga.

Com isso, computadores antigos podem voltar a ter serventia, o que torna uma aplicação *web*, um sucesso devido à redução de custos operacionais.

2.2.1 Ria Rich Internet Application

São aplicações *web*, que tem características semelhantes às aplicações feitas para *desktop*. Aplicação RIA, carrega apenas o navegador que tem a função de executar a aplicação que está concentrada em um servidor.

Isto inclui os dados, a base de dados que tem a funcionalidade de armazenar estas informações, bem como o estado da aplicação conforme segue o padrão de transação de um sistema gerenciador de banco de dados relacional.

Um exemplo de aplicação *RIA* é um pacote de aplicação que a empresa *Google* fornece, tal como o *Google Docs* o que possibilita usar um editor de texto, planilha eletrônica ou ver uma apresentação no estilo *PowerPoint*.

A vantagem do uso de aplicações *RIA* é que elas não precisam de instalador, pois são carregadas em um servidor, e acessadas por meio de um *browser* economizando espaço em disco, facilitando o acesso aos dados e principalmente no que se refere à segurança, devido ao funcionamento da aplicação é por *sandbox*.

Aplicações do tipo *RIA* são executadas em uma *sandbox*, ou seja, tudo o que for executado nessa nela, ao reiniciarmos o computador será perdido. Isso ocorre porque é executada em um ambiente virtual, sendo assim é feito um isolamento do sistema operacional, evitando contaminações indesejáveis por vírus.

A Figura 1, mostra como funciona um aplicativo com *sandbox*.

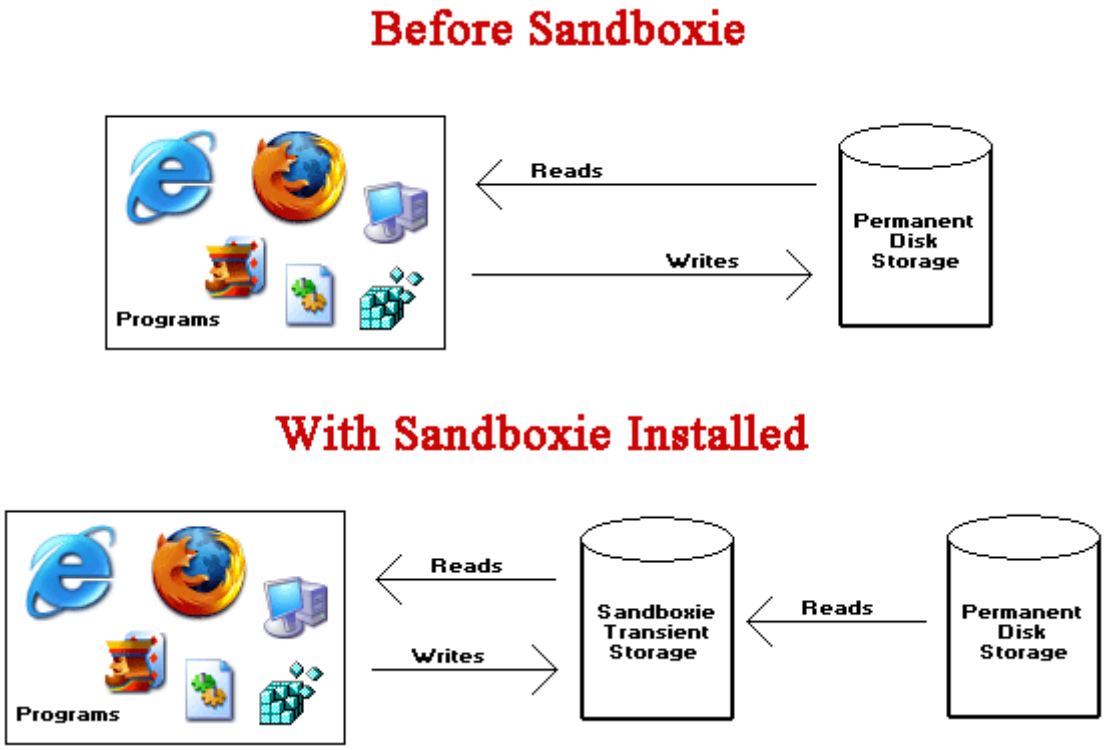

Figura 1 - Funcionamento de uma sanbox
Fonte: Make use of (2012)

Conforme a imagem da Figura 1, uma aplicação executada sem a utilização de uma *sandbox*, tem exposição direta, o que pode acarretar em ônus ao sistema operacional, dados, infecções por vírus.

Já a aplicação que tem auxílio de uma sandbox, é executada com uma espécie de proteção, onde às áreas criticas do sistema são isoladas, evitando perda de dados e conseqüentemente deixando a aplicação segura.

2.2.2 Java

Trata-se de uma linguagem de programação voltada para a orientação a objetos (*O.O*), desenvolvida em meados da década de 1990, por uma equipe de programadores que tinha como chefe James Gosling. É uma linguagem, com características semelhantes à linguagem c++, devido sua sintaxe ser muito parecida.

O fato de ter semelhanças à linguagem C++ faria do Java uma linguagem coirmã da linguagem C++, pois ambos os programadores pensaram no conceito referente à portabilidade.

Desta forma, um sistema desenvolvido na linguagem Java é executado em uma máquina virtual chamada de *JVM* (*Java Virtual Machine*) o que a faz ser uma linguagem portátil, capaz ser executada em vários sistemas operacionais como *Windows, Unix, Linux*, *Mc-Os,* entre outros devido à propriedade da mesma.

A Figura 2, mostra como funciona a linguagem Java.

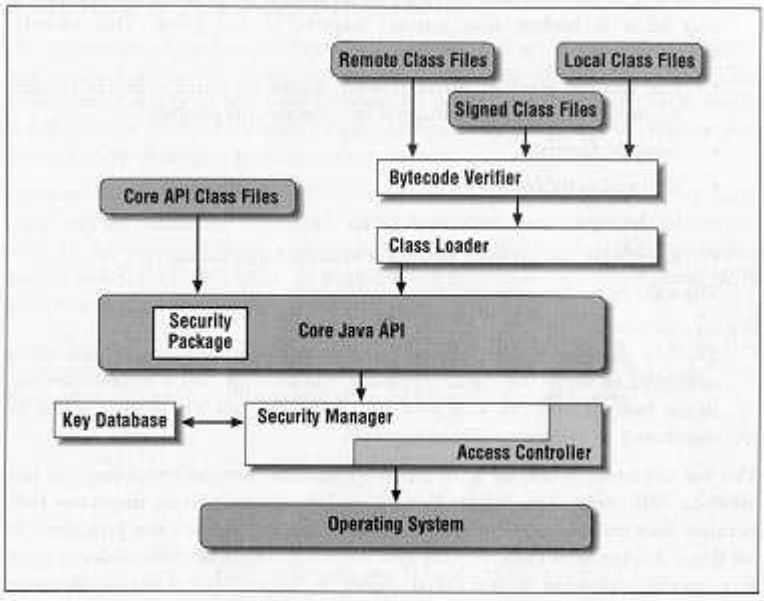

Figura 2 - Funcionamento da Linguagem Java
Fonte: Hermann (2001)

Conforme a Figura 2, os arquivos de classes remotas, arquivos de assinaturas de classes e arquivos de classes locais, são verificadas pelo *Bytecode*, o qual tem a função de carregar as classes e a partir deste momento é feito a entrada no sistema de segurança do núcleo de aplicações Java para que depois de feito os gerenciamentos da segurança cheguem a sua execução final da aplicação.

2.2.2.1 JVM - Java Virtual Machine

A *JVM* é uma Máquina Virtual criada especificamente para executar código Java. *Uma JVM* trata-se de uma poderosa ferramenta de desenvolvimento no âmbito do curso análise e desenvolvimento de sistemas.

Ao contrário de inúmeras outras linguagens de programação, *JAVA* é voltada para o conceito de multi plataforma. Isto significa que um programa Java pode ser executado em vários tipos de *hardwares* e de sistemas operacionais como *Unix, Linux, Windows, Mac OS*.

Ao invés de gerar código de máquina para um processador real, os compiladores Java geram código para a *JVM*. Esse código é comumente chamado de *bytecode* Java.

A Figura 3 mostra um comparativo de compilação e execução da linguagem C++ com a compilação e execução da linguagem *Java*.

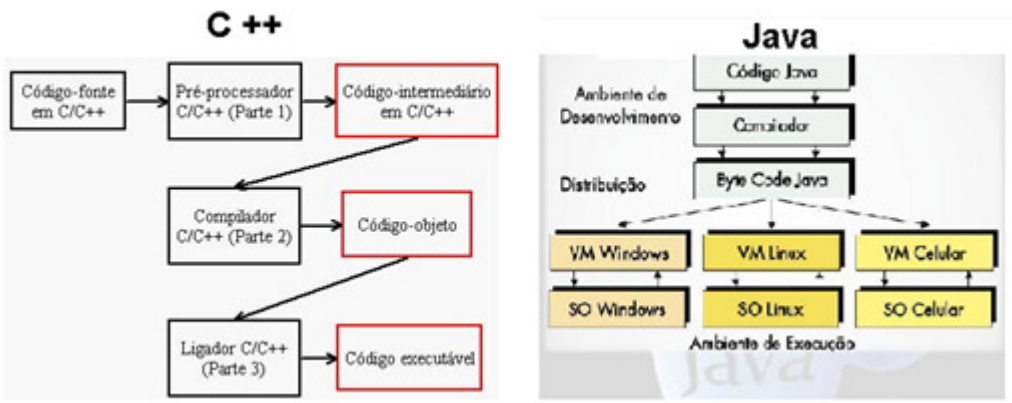

Figura 3 - Comparação da linguagem C++ com Java
Fonte: Caloni (2012)

Para isso acontecer, a partir do código fonte é gerado um bytecode que faz com que a máquina virtual Java “entenda”, que é para mostrar uma operação de saída em vídeo contendo a frase “Meu primeiro programa em **Java**”. Em comparação, quando se trabalha com linguagem C, o código é executado diretamente pela *CPU*, o que faz que seja mais rápido.

Com o programa realizado com sucesso, surge uma dúvida muito importante que se refere à composição de uma *JVM*. Seus principais componentes são:

- Carregador de classe

- Máquina de execução
- Áreas de dados em execução
- Interface de métodos nativos

O carregador de classe tem como função de carregar a classe em que foi escrito o programa, bem como o tipo de *interface* utilizada. Pode ser de um simples *prompt* de comando ou se apresentar na forma de interface. A *interface* tem o nome de *GUI*, do inglês que significa *Graphic User Interface*.

2.2.3 Flash

A origem da linguagem *flash* aconteceu com a ausência de movimento nas imagens que eram trabalhadas na época, devido a um árduo trabalho de Jonathan Gay com o objetivo de fazer com que as imagens que eram estáticas, pudessem apresentar alguma interatividade ou animação.

Tal linguagem pode ser utilizada com o intuito de criar aplicações das mais variadas formas ou ainda criar efeitos visuais inéditos, por meio dos códigos de programação chamado *script*.

Tal produto foi desenvolvido e comercializado pela empresa *Macromedia*, a qual é especializada no desenvolvimento de programas capazes de auxiliar diferentes processos na criação de páginas *web*.

Atualmente a linguagem pertence à empresa Adobe, que está distribuindo a versão *creative* CS6, com o nome de *flash* CS6 ao qual suporta várias extensões de imagens para atender diferentes formas de comercialização.

Possui mercado tanto para *softwares* que utilizam a internet quanto para as que são chamadas de aplicações *desktop*.

2.3 COMPARATIVOS DA LINGUAGEM FLASH COM A LINGUAGEM JAVA

Ambas as linguagens de programação, apresentam em sua constituição a presença de uma máquina virtual. Isso quer dizer que tanto a linguagem *flash* quanto a linguagem Java atendem o principio da portabilidade, pois ambas conseguem ser compatíveis com diversos sistemas operacionais.

Com relação ao desenvolvimento de aplicações *RIA* ambas as linguagens conseguem se adequar a este tipo de aplicação, pois são ferramentas gráficas e podem ser executadas a partir de um servidor.

Porém existe uma diferença entre estas linguagens, onde a linguagem flash apresenta como recursos *plugins* mais eficazes e compatibilidade com a maioria dos navegadores do mercado. A linguagem Java não apresenta mais os mesmos *plugins* devido às reclamações excessivas dos programadores devido às freqüentes falhas apresentadas.

A justificativa da utilização da linguagem de programação *Flash*, foi baseada no resultado de uma pesquisa do site real chat, que tinha o objetivo de verificar a existência de *plugins Flash* e Java.

Para chegar ao resultado, foram colocadas estruturas capazes de coletar a presença de tais *plugins* em 3.000 máquinas de usuários que utilizam a Internet e ao término da coleta de dados apresentou uma aceitação maciça do *plugin* da linguagem de programação de *Flash*. O resultado desta pesquisa informou que quase 96% do grupo tinham o *plugin Flash* instalado e apenas 56,43% possuíam o *plugin* Java instalado, conforme mostra a Figura 4.

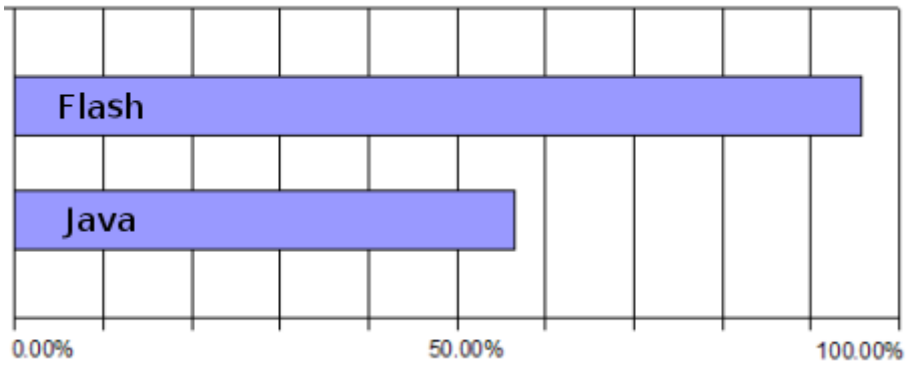

Figura 4 - Comparação de Flash com Java
Fonte: Realchat (2012)

2.4 ANIMAÇÕES GRÁFICAS UTILIZANDO FLASH

Para a construção de animações gráficas utilizando o *Flash*, torna-se necessário conhecer elementos fundamentais que envolvem a linguagem de programação e possibilitam o desenvolvimento das mesmas.

2.4.1Frames

Uma animação criada em *flash* começa pela definição de um cenário. Um cenário é como uma cena em uma peça de teatro, onde temos os objetos que são os atores e as ações o papel de cada objeto.

A animação em *flash* acontece dentro de um laço. A cada iteração do laço, pode acontecer uma modificação do cenário. Por exemplo, um objeto pode mudar de posição.

Cada iteração desse laço é chamada de *frame*, ou quadro de animação.

Com base nestas informações, vemos que a cada *frame* ou quadro, o cenário é modificado.

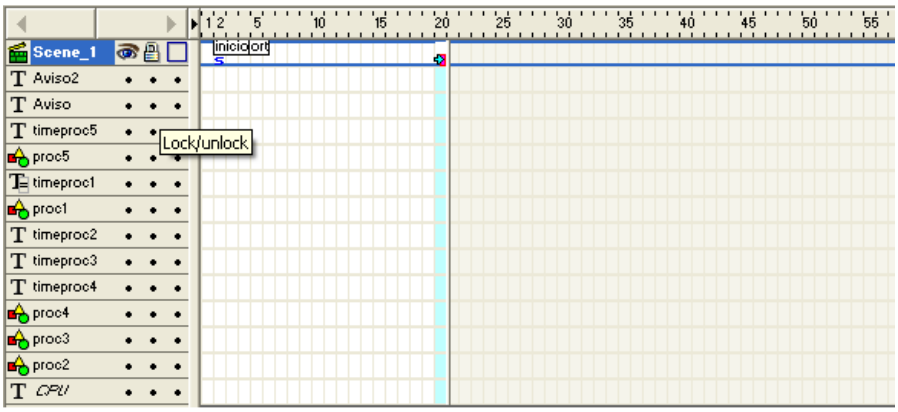

Figura 5 - Funcionamento de um frame
Fonte: Autoria própria

A Figura 5 mostra parte da interface da ferramenta *SwishMax,* na qual foram feitas as animações deste trabalho.

2.4.2 Velocidade de Animação dos Frames

Existem dois tipos de controle de velocidade na execução de uma animação, a velocidade do filme (animação) e a dos objetos que irão compor o cenário.

Velocidade do filme: é realizado um controle clicando com o botão direito no cenário da animação e como queremos mexer nas propriedades do filme, iremos pelo *menu*, até a opção *movie* e depois clicamos em propriedades como mostra a figura 6 abaixo.

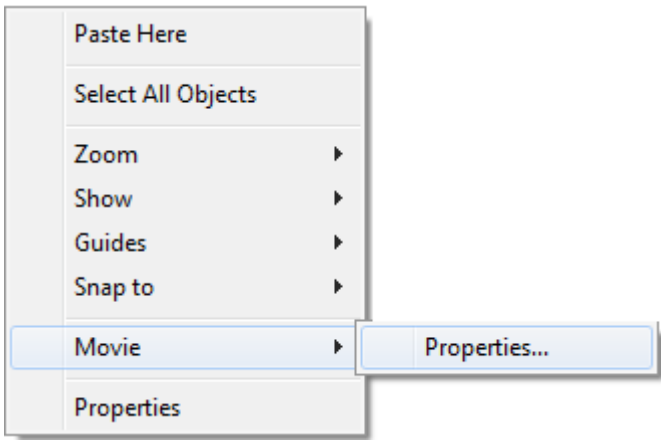

Figura 6 - Velocidade do filme
Fonte: Autoria própria

A seguir é apresentada ao programador a interface que realiza a mudança da velocidade, conforme a Figura 7.

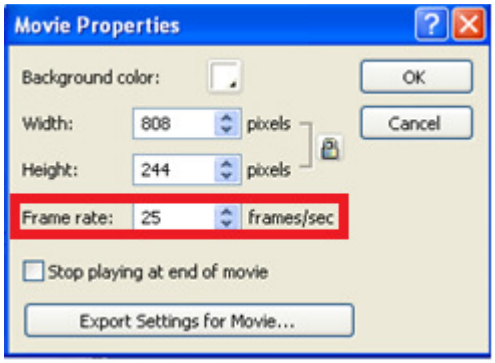

Figura 7 - Mudança da velocidade do frame
Fonte: Autoria própria

O ponto destacado em vermelho é onde temos a velocidade do filme, no caso ilustrado a velocidade é de 25 *frames* por segundos, conforme a definição do programador.

Segundo Arruda (2012) este controle de velocidade do filme é importante porque dependendo da velocidade aplicada na execução do filme, poderão ocorrer distorções que farão da animação um fracasso visualmente percebido por quem está assistindo. O valor mínimo para que isto não ocorra é de 24fps (*frames* por segundo) conforme os padrões de cinema.

Além do filme é necessário controlar a velocidade de movimentos dos objetos do cenário. Isto é feito ao especificar a distância que um objeto vai se mover a cada *frame*.

Por exemplo: digamos que o usuário coloque o objeto no frame um na posição 10 *pixels*, no *frame* dois na posição 20 *pixels* e no *frame* três, na posição 30 *pixels*. Se o filme for configurado para rodar em três *frames* por segundo, o objeto irá se mover na tela com a velocidade de 20 *pixels* por segundo (distância separando a posição do objeto no *frame* um ao *frame* três).

2.4.3 Formas de Programar um Cenário

Existem duas formas clássicas de programar um cenário, uma mais intuitiva podendo ser mais trabalhosa e a outra utilizando programação. A definição de movimentação de objetos à mão, normalmente, é utilizada por usuários leigos, tendo como objetivo fazer com que algo realize movimento dentro de um cenário.

Neste caso, a cada frame o usuário deve determinar à mão a posição de cada objeto, exatamente como se faz na animação pela técnica de *stop motion*. Nesse método são feitas várias fotos de um cenário, onde a cada foto são movidos objetos do cenário. Ao projetar as imagens uma após a outra rapidamente obtém a ilusão de movimento [1].

As animações em flash podem ser definidas usando uma linguagem de programação. No caso da empresa Adobe, existe a linguagem *action script*, já o *SwishMax* usa uma linguagem muito semelhante chamada *SwishScript*.

O usuário trabalha em um editor de textos, onde o texto contém comandos que fazem com que os objetos possam se movimentar durante a sua execução.

No trecho de código abaixo, temos uma linha que demonstra a movimentação de um objeto.

```
_root.proc1._x += 1
```

Este código faz com que o objeto se movimente no cenário para direita, uma vez que estamos modificando a propriedade da posição do objeto, aumentando seu valor. Se fosse feito um decremento o objeto movimentar-se-ia para a esquerda.

```
_root.proc1._x - = 1
```

[1] http://www.producaocultural.org.br/wpcontent/uploads/livroremix/phillipearruda.pdf

2.4.4 Eventos que Ocorrem em cada Frame

Durante a execução de uma animação *flash*, podem acontecer diversos eventos. Dois exemplos de eventos são:

- O usuário clica o *mouse*;
- O usuário pressiona uma tecla.

O programador pode definir funções para cada evento diferente que será chamado pelo flash quando necessário. Além desses eventos gerados pelo usuário do *software*, também existem dois eventos para cada frame de animação: *OnEnter* e *OnExit*. O programador também pode definir funções que serão chamadas quando esses eventos acontecerem.

2.5 TÓPICOS DE SISTEMAS OPERACIONAIS

Sistema operacional é um sistema que gerencia a ligação entre computador e o usuário. Um sistema operacional tem várias atribuições, tais como gerenciar processos, controlar acesso à memória, discos de armazenamento de dados, gerenciar permissões entre usuários, garantir segurança dos dados e trafegar informações.

A disciplina de Sistemas operacionais é parte da ementa obrigatória do curso de Análise e Desenvolvimento de Sistemas, sendo uma das principais e mais difíceis disciplinas a serem compreendidas pelos acadêmicos.

O objetivo deste projeto é fornecer ferramentas para auxiliar no processo de aprendizagem dos acadêmicos, como resultado final apresentar animações em linguagem *flash*, envolvendo um estudo sobre escalonamento.

Escalonamento de processos é uma forma que o sistema operacional tem de atender a todas as solicitações dos usuários por intermédio de uma estrutura chamada escalonador, onde o uso da *CPU* pode ser feito aleatoriamente, por prioridades ou realizando chaveamento entre todos os processos.

Para realizar um escalonamento, existem algumas prioridades, como processos de *I/O* (entrada e saída) e também processos que exijam uma computação mais intensa.

A finalidade de uma *CPU* ao realizar um escalonamento tem as seguintes funções:

- Política da justiça – Todos os processos devem ser tratados e atendidos de forma igual.
- Produtividade - Capacitar à execução de forma que processos sejam executados mais rapidamente reduzindo o tempo de execução.
- Pró Ativismo – Proporcionar respostas rápidas para os usuários de níveis mais interativos.
- Evitar sobrecarga – Serve para que recursos na sejam desperdiçados, embora determinados processos necessitem de maior tempo de execução para garantir uma maior eficiência.
- Regra do "comportamento" – Processos que sejam mais bem adaptados podem receber uma atenção melhor.
- Balanceamento: Garantir que todos os recursos disponíveis sejam aproveitados, de modo que todos os processos sejam atendidos, sem gerar ônus ao sistema, como por exemplo, *Deadlock*.

Os tipos de escalonamentos estão distribuídos da seguinte forma:

- *FIFO* – Primeiro processo a chegar, é o primeiro processo a sair da fila de processos. Forma mais simples de escalonador existente.
- *SJF* -O processo mais curto em relação ao tempo, tem prioridade. Neste tipo de escalonamento foi pioneiro na utilização de mecanismo de preempção.
- Loteria – Processos recebem uma espécie de fichas, que se denominam *token*, onde é sorteado um numero aleatório e o mesmo é atendido pela *CPU*. Ressalta-se que quanto mais tokens um processo tiver, mais chances ele tem de ganhar a *CPU*.
- Garantido – Este tipo garante que o ganho da *CPU* pelos processos seja da forma mais precisa possível.
- *Round Robin* – Este escalonamento, possui um temporizador que tem o nome de *quantum*, onde cada processo executa o tempo que dura o

quantum e quando o mesmo extrapola, ocorre à preempção e o processo seguinte ganha a *CPU*, obedecendo à mesma regra.

Relacionado aos escalonamentos mencionados acima, somente o escalonamento *Round Robin* e Garantido não sofrem com o problema denominado *starvation*, devido a todos os processos serem atendidos pela *CPU* e consequentemente eliminando este problema.

2.6 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Neste capítulo estão contidas informações referentes à essência do que se trata este projeto, que retratando a importância da disciplina de Sistemas Operacionais e seus desafios, além de uma breve introdução ao que se propõe realizar com este estudo correlacionado com as animações utilizando a linguagem de programação *flash* na ferramenta *SwishMax*.

Foi realizada pesquisa bibliográfica referente à linguagem *flash* para destacar a sua importância, funcionamento, comportamento e estruturação.

O comparativo entre a linguagem *flash* e *Java* é importante para verificar que mesmo sendo linguagens similares, *flash* pode perfeitamente se adaptar em qualquer sistema operacional ou qualquer tipo de *browser*. Ao contrário da linguagem *Java* cujo *plugin* não é mais distribuído devido a reclamações de programadores.

Tópicos da Disciplina de Sistemas Operacionais foram mencionados, pois os mesmos têm papel importante no resultado final do projeto, referente ao que se pretende alcançar com a construção das animações.

3 SISTEMAS SIMILARES

Para verificar a contribuição deste estudo, torna-se necessário pesquisar trabalhos de natureza similar a fim de enriquecimento do mesmo, aproveitando-se de possíveis elementos que tem relevância neste trabalho.

Foram encontrados três trabalhos semelhantes e por conseguinte estudaram-se suas particularidades, para buscar um enriquecimento bibliográfico.

A síntese de cada sistema, tem sua respectiva descrição abaixo.

3.1 SOSim

O projeto *SOSim* é um simulador com diversas funções que permite aperfeiçoar e fixar os conceitos adquiridos em aula.

Este simulador explica como funcionam várias partes de um Sistema Operacional, retratando como funciona um processo, como é feito o controle de bloqueio dos processos, conceitos referentes à preempção, alocação de memória, escalonamentos, paginação e *swapping*.

Analisando o simulador é possível constatar que o mesmo apresenta algumas limitações para ensino. Em determinados pontos falta clareza de explicação, e a interface às vezes deixa a desejar, como por exemplo, na representação de cinco processos mostrados na Figura 8.

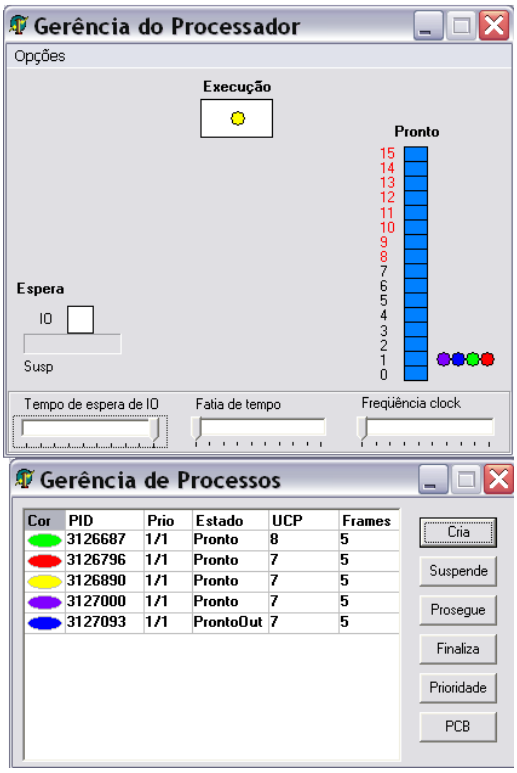

Figura 8 - Representação e Gerência de Processos SOsim
Fonte: Training (2012)

Conforme se observa na figura 8, os componentes que foram utilizados na interface que mostram o sistema não são claros, porque conforme a extração de *screenshot* da tela do simulador SOSim, não é possível ter certeza do que significam os processos estarem relacionados com o valor um (1), pois pode ser prioridade do processo, nível crítico que eles estão representados, entre outros.

Essa tela poderia sofrer modificações de forma a tornar-se mais auto explicativa no que se refere à fila de processos, como os mesmo adquiriram como atributo o valor um (1) e se o que está representado é prioridade, nível crítico, ou outro aspecto que não foi possível realizar a compreensão.

Além desses problemas o *software* poderia ter um guia melhor ou uma ajuda ensinando como utilizar a ferramenta. Deixando as questões críticas referentes ao simulador a proposta é muito válida, pois aborda vários elementos do escopo da disciplina de Sistemas Operacionais.

Portanto, a proposta se readequada nos padrões de *IHC* (Interface Humano Computador), bem como a correção de suas falhas o que poderá contribuir de modo mais conciso com o que foi proposto.

3.2 JOGO MARSHAL

O *software* Marechal que de acordo com o trabalho de Rodrigues (2008), visa auxiliar o aprendizado dos alunos do matriculados no curso de Análise de Sistemas, no que se refere a alocação de recursos que faz parte da ementa da disciplina de Sistemas Operacionais.

Marechal é um jogo que foi construído utilizando a ferramenta *Game Maker*. O jogo tem por objetivo ajudar os alunos a compreender como funciona a alocação de recursos dentro do sistema operacional.

A Figura 9 mostra uma tela do sistema.

Figura 9 - Tela do Software Marechal
Fonte: Rodrigues (2008)

Existem inúmeras vantagens ao transformar um problema real, em uma forma lúdica para ser mostrada como a solução do mesmo. Tais vantagens são: desenvolvimento cognitivo, desenvolvimento de senso crítico, capacidade de tomada de decisões (RODRIGUES, 2009).

Referente à problemática do projeto a mesma apresenta-se intrinsecamente ligada aos objetivos do software Marechal, pois ambos os projetos tem como propósito trabalhar a essência da praticidade e da dinamicidade da forma com que são abordados os conteúdos, e a limitação do professor ao explicar um princípio dinâmico que o sistema operacional realiza (SO) no quadro negro que é estático.

A construção do sistema marechal não apresenta dificuldades em sua essência, desta forma a mesma atende os quesitos de Interface Humano Computador, níveis de dificuldades, implantação, sonoplastia e validar a viabilidade do projeto aplicando o simulador.

Esta ideia de produzir jogos vem do ano de 1962, onde um pequeno computador de baixa capacidade era capaz de processar um jogo de tênis, onde um ponto na tela era a bola e dois bastões eram as raquetes. Esta ação foi uma das precursoras no que se refere aos jogos que se tem hoje, onde cabe destacar telejogo, atari, *master system*, entre outros.

3.3 SISTEMA OPERACIONAL MINIX

Minix foi elaborado como um complemento à obra “Sistemas Operacionais Modernos” (TANEMBAUM, 2003).

O sistema operacional construído por Andrew Tanembaum era relativamente pequeno, comparado com os sistemas encontrados atualmente. O mesmo tinha especificações gerais dos requisitos que incluíam 16*MB* de memória *RAM* e disco rígido inferior a um *GB*.

O *Minix* foi desenvolvido com similaridades com o *UNIX*, versão sete. Com base nisso, o sistema torna-se bastante seguro. A Figura 10 mostra a interface do sistema operacional *Minix*.

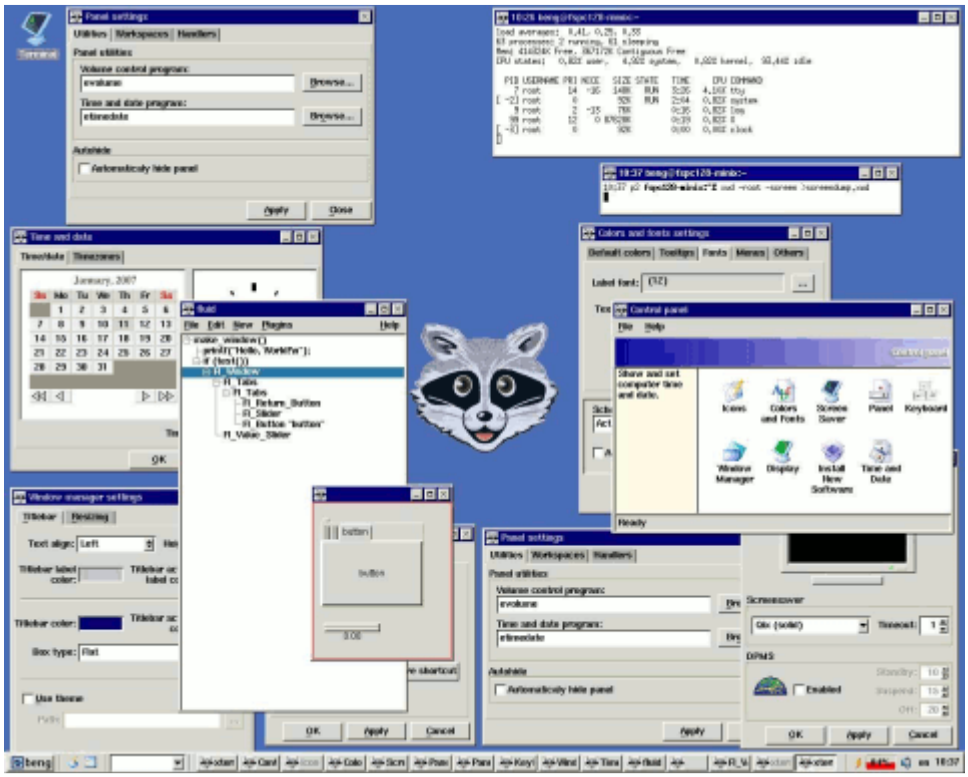

Figura 10 - Tela da Interface do Sistema Operacional Minix
Fonte: Distrowatch (2012)

O *Minix* foi desenvolvido para que os alunos pudessem olhar o código do sistema e pudessem fazer modificações. Para isso é preciso que eles tenham conhecimento de programação.

Referente à programação do SO *Minix*, devido ao fato dos alunos não possuírem o conhecimento necessário para programá-lo, o que inviabilizou a sua utilização, portanto não seria uma solução para o que foi proposto com o trabalho.

3.4 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Este capítulo preocupou-se com a pesquisa de trabalhos similares com o intuito de validar o estudo possibilitando desenvolver este projeto. Para tanto foram encontrados três trabalhos, um deles apresentado nesta instituição, o que reforça o interesse referente ao auxílio para os acadêmicos de Sistemas Operacionais.

Outro trabalho pesquisado foi o SOsim o qual tem grande contribuição educacional, embora apresentou resultado final insatisfatório, devido à dificuldade em interpretar o que seu funcionamento pretendia abordar.

Quanto ao sistema *Minix* não foi possível ser utilizado neste trabalho, pois se constatou insuficiência de conhecimento do sistema operacional Unix, sendo que sem este tipo de habilidade fica inviável sua utilização.

4 FERRAMENTA DE DESENVOLVIMENTO

Para desenvolver a proposta referente às animações interativas, foi utilizado o software *SwishMax.* Esta ferramenta apresenta a mesma composição dos componentes da *engine* que a linguagem *flash* usa e o *software* é licenciado, pertencendo à empresa privada *Swishzone.com PtyLtd*, que fica localizada em *Sidney* na Austrália.

Esta empresa começou a desenvolver seus trabalhos no início do século XXI e hoje lidera o mercado de desenvolvimento de ferramentas alternativas em *flash*.

Segundo dados da empresa Swishzone (2012) *SwishMax* é a ferramenta que muitos chamam em suas organizações empresariais de "Carro Chefe", onde consta de seis milhões e setecentos mil *downloads*, além de ter seiscentas mil licenças e mais de 327 mil clientes.

A matriz *SwishMax* tem revendedores e distribuidores pelo mundo, sendo adaptados em suas respectivas línguas nativas. A ferramenta tem distribuidoras localizadas na Alemanha, Japão, França, Itália, Polônia, Taiwan e China e tem utilização em mais de 200 países.

4.1 COMPARAÇÃO DE SWISHMAX COM FLASH

Ao realizar uma pesquisa comparativa entre as ferramentas de desenvolvimento de animações gráficas, *Swishmax* e *Flash vejam a* Quadro *1:*

Comparativo da ferramenta Swish Max com a ferramenta Flash				
Ferramenta	**Preço**	**Aprendizado**	**Apoio**	**Potencial**
SwishMax	R$ 150,00	Usuários iniciantes	Menos Divulgado	Feito para usuários iniciantes ou intermediários
Flash	R$1216,26	Usuários experientes	Flash é mais conhecido	Feito para usuários avançados

Quadro 1 - Comparativo das Ferramentas Flash e SwishMax
Fonte: Distrowatch (2012)

Conforme informações apresentadas na Figura 11, o uso da ferramenta *SwishMax* está justificado, porque sua utilização tem custo reduzido, além de ser mais fácil de realizar as animações para pessoas iniciantes, visto que o

conhecimento no início deste projeto, praticamente inexistente. Como a ferramenta *flash* está mais tempo no mercado, é natural que a mesma domine o mercado de usuários que fazem aplicações mais robustas.

4.2 PROJETO DO SITE

O projeto do *site* foi construído de forma que facilite sua navegação, proporcionando maior compreensão e maior amplitude de como se comporta determinada estrutura do escopo da disciplina de Sistemas Operacionais.

4.2.1 IHC – Interface Humano Computador

Para desenvolver o projeto do *site*, é interessante observar algumas regras estabelecidas pela Interface Humano Computador, pois estas ajudam principalmente no aspecto visual.

Itens como padronização do *site*, definição de cores e a escolha correta da fonte, são alguns dos principais quesitos quando se refere à IHC.

A construção do *site* é parte integrante do desenvolvimento da pesquisa, desta forma o mesmo é simples e eficiente.

Para que a interface adquira uma forma mais atraente são aplicados estilos CSS (*Cascading Style Sheet*), contudo o conteúdo (linguagem) é construído apenas utilizando-se de conceitos de *HTML*.

4.3 DIFICULDADES ENCONTRADAS

A disciplina de Sistemas Operacionais por si só apresenta um elevado grau de dificuldade, exigindo maior atenção e principalmente aproveitamento do tempo disponível nas aulas.

O não aproveitamento e o elevado número de ausência nas aulas da disciplina acarretam em dificuldades futuras, por mais que estas sejam provocadas pelo acadêmico, cabe ao professor que leciona a disciplina, transformar suas aulas em um atrativo evitando desta forma a evasão disciplinar.

Além da evasão que se dá por meio de desmotivação de natureza complexa, há a dificuldade em demonstrar para o aluno de forma prática e interativa, o funcionamento de um determinado assunto (matéria).

4.4 OBJETIVOS DA UTILIZAÇÃO DAS ANIMAÇÕES

O objetivo principal é resolver principalmente o impasse da dificuldade da explanação dos conteúdos que se apresentam de forma estática. Além disso, proporcionar aos alunos um processo de ensino aprendizagem dinâmico, que atinja a maioria dos acadêmicos no que se refere à diminuição da evasão escolar, ou em particularmente em determinadas matérias.

Esta proposta possibilita a busca de conhecimento autônomo, haja vista que à medida, em que ocorrem várias leituras surgem novas interpretações e inquietudes no âmbito do “querer saber mais” e da incansável pesquisa, trazendo o aluno de volta ao ambiente de sala de aula.

4.5 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Neste capítulo, foram abordadas questões referentes à ferramenta SwishMax, pois animações foram desenvolvidas a partir desta ferramenta. Foi desmistificado a partir dos comparativos, que apesar da ferramenta SwishMax não ser do tipo Open Source, ela é mais viável de se trabalhar em relação com a ferramenta da Adobe que se encontra hoje na versão CS6.

Detalhou-se o trabalho com a ferramenta SwishMax e como esta pode auxiliar os acadêmicos na questão de conteúdo estático e dinâmico bem como a definição dos objetivos que deverão ser alcançados com o uso das animações.

5 IMPLEMENTAÇÃO

Neste capítulo, será explicado como foram feitas as animações desenvolvidas na ferramenta *SwishMax* .

Foram desenvolvidas três animações, que explicam como funcionam na prática os escalonamentos: FIFO, SJF, e *Round Robin*.

5.1 FIFO

Foram definidos dois pontos básicos da animação que seriam o ponto inicial de onde surgem os processos e o ponto final, onde é demonstrada a execução do processo pela *CPU*.

Então a animação começa no *frame* um, portanto neste frame um irão nascer os processos e finalizando a animação no *frame* 20, onde o processo é atendido pela CPU e é executado conforme apresentado na Figura 12.

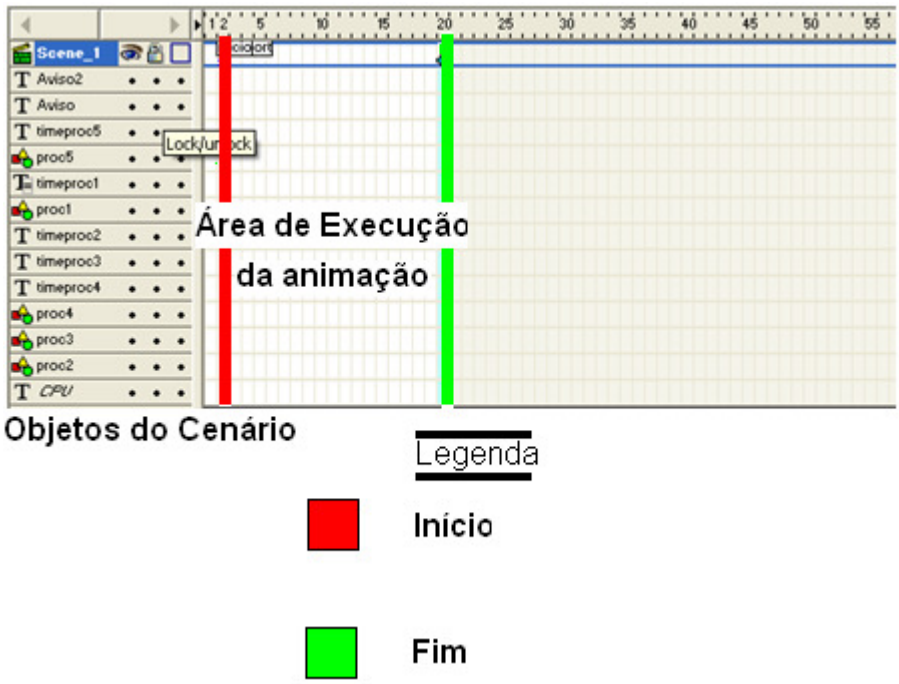

Figura 11 - Animação FIFO
Fonte: Autoria própria

5.1.1 Animação FIFO

Esta seção, encontra-se a parte da programação do algoritmo em flash. Isso torna-se necessário devido ao fato das animações serem programas utilizando uma linguagem de programação, torna-se necessário sua descrição.

```
onSelfEvent (enter Frame) {

_root.Aviso._Visible =    (_root.proc1._x < 100)
                           or (_root.proc2._x < 100)
                           or (_root.proc3._x < 100)
                           or (_root.proc4._x < 100)
                           or (_root.proc5._x < 100);
```

Figura 12 - Início de um novo processo
Fonte: Autoria própria

A mensagem aparecerá quando qualquer um dos processos estiver no extremo esquerdo do vídeo, com coordenada x inferior a 100.

Pode-se verificar que o valor de *Visible* é definido por uma expressão condicional. Essa expressão será verdadeira quando a posição x for menos que 100. Isso controla a visibilidade de uma mensagem no vídeo que aparece no canto superior esquerdo da *interface* da animação:

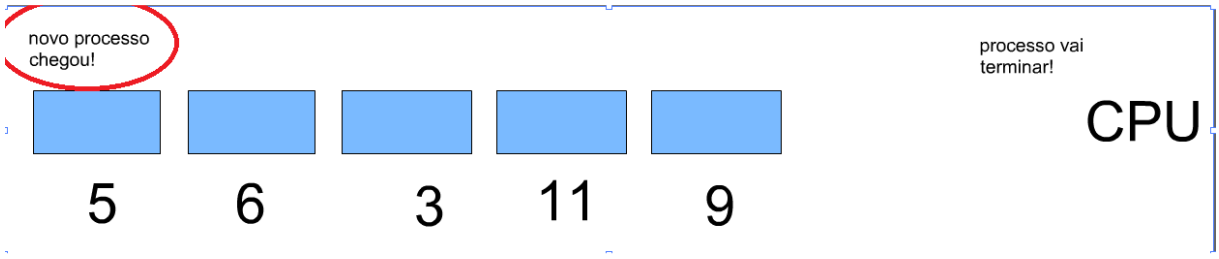

Figura 13 - Especificação da Mensagem de Aviso
Fonte: Autoria própria

O próximo trecho faz a movimentação dos objetos:

```
_root.proc1._x += 1;
_root.proc2._x += 1;
_root.proc3._x += 1;
_root.proc4._x += 1;
_root.proc5._x += 1;
```

Figura 14 - Movimentação dos Retângulos na Interface
Fonte: Autoria própria

Neste caso, o código realiza um movimento dos objetos para a direita de um em um *pixel*.

```
if (_root.proc1._x >= 660) {
        _root.proc1._x = 60
        _root.timeproc1.text = '  '
        _root.timeproc1.text = Math.randomInt(20)+1
{}
if (_root.proc2._x >= 660) {// mesma lógica para processo2, processo3, ... até processo5
```

Figura 15 - Verifica a posição do Objeto na interface
Fonte: Autoria própria

Este trecho verifica a posição, na tela, do processo de número 1. Se a posição x for igual a 660, a estrutura condicional será executada porque o valor da premissa é verdadeiro. Isso faz com que se executem três (03) instruções.

Um objeto que ao passar na posição 660, o objeto tem sua posição automaticamente alterada e volta a assumir valor 60.

Objetos além de ser parte de um cenário eles têm em seu interior propriedades. Neste caso a propriedade texto do objeto é zerada. Ou seja, o valor de tempo respectivo ao processo um (*timeproc1*) é zerado.

Posteriormente, com o conteúdo da propriedade texto zerado, ele receberá um valor randômico ou aleatório compreendido no intervalo de 1 a 20. Isso é possível, porque a propriedade texto de *timeproc1* recebe um valor aleatório quando executado a instrução randômica.

Uma biblioteca, nada mais é que uma coleção de arranjos que já vem pré-definido para ser utilizado, basta o programador invocá-las*. Math.random();*. Isto quer dizer que o programador quer gerar números randômicos ou aleatórios. A função *random* está dentro da biblioteca *math*, por isso é chamada de função *random* desta forma (http://www.tato.ind.br/files/Workshop.pdf).

Exemplos de funções que existem na biblioteca Math:

- ***Constrain* (x,a,b).**Restringe um número x ao intervalo de valores [a, b] para todos os tipos de dados.
- ***pow*(base, exponent).** Calcula a potência da base.
- ***sin*(angle) *cos*(angle) *tag*(angle).**Funções trigonométricas. Seno, cosseno, tangente.
- **Math.**Random(). Sorteia valores de forma automática.

```
_root.timeproc1._x = _root.proc1._x;
   _root.timeproc2._x = _root.proc2._x;
```

Figura 16- Atribuição de Variáveis.
Fonte: Autoria própria

O trecho da figura verifica que a posição do tempo do processo 1*(timeproc1*) recebe a posição do processo 1.

Este trecho verifica que a posição do tempo do processo 2(*timeproc2*) recebe a posição do processo 2.

5.2 SJF

5.2.1 Animação SJF

```
_root.Aviso2._Visible =    (_root.proc1._x > 620) //avisa que o processo terminou quando chega na
ordenada 620
```

Figura 17 - Término da execução de um processo.
Fonte: Autoria própria

Neste trecho de código, o campo *Visible* será verdadeiro quando qualquer um dos processos estiver no extremo esquerdo do vídeo, com coordenada x superior a 620.

Pode-se ver que o valor de *Visible* é definido por uma expressão condicional. Essa expressão será verdadeira quando a posição x for maior que 620.

Isso controla a visibilidade de uma mensagem no vídeo que aparece na extremidade direita da *interface* da animação conforme mostra a Figura 19.

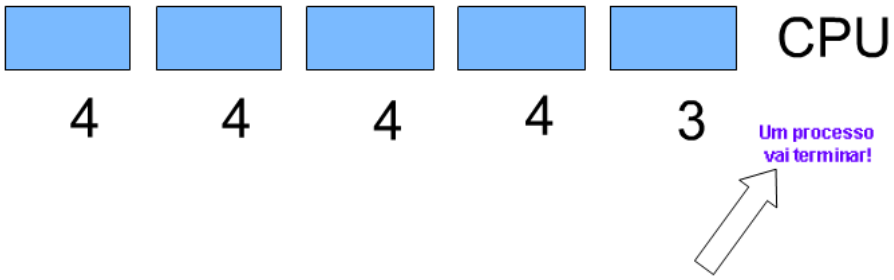

Figura 18 - Mensagem de aviso de término de processo.
Fonte: Autoria própria

```
if (_root.proc1._x >= 660){
        _root.proc1._x = 60;
        _root.timeproc1.text = Math.randomInt(5)+1;
        _root.fila_anda = false;
```

Flag

Figura 19 - Demonstração de onde a fila para de andar
Fonte: Autoria própria

Na primeira linha do trecho de código da figura 20, é verificado se o processo está na ordenada 660. Caso esteja, a posição X do objeto é alterada para 60, fazendo o objeto voltar à posição extrema esquerda na tela.

Em seguida, o valor do tempo de processo recebe valor randômico entre 1 a 5 porque ao passar na posição 660, o objeto executou a instrução e o mesmo retorna a posição 60 como se fosse um processo novo que chegou à fila de processos e consequentemente terá que receber um tempo de execução.

Na última linha, o *flag* indica que a fila tem que parar de andar. Esta fila de processos é paralisada para dar tempo ao aluno de ver que um novo processo chegou. Após isso, aparece um botão, onde tem a instrução de que seja realizado um clique no mesmo. Após realizar o clique a fila de processos é reordenada, com processos de menor duração sendo executados primeiro porque se trata de um escalonamento SJF, ou seja, processos mais curtos são executados primeiro.

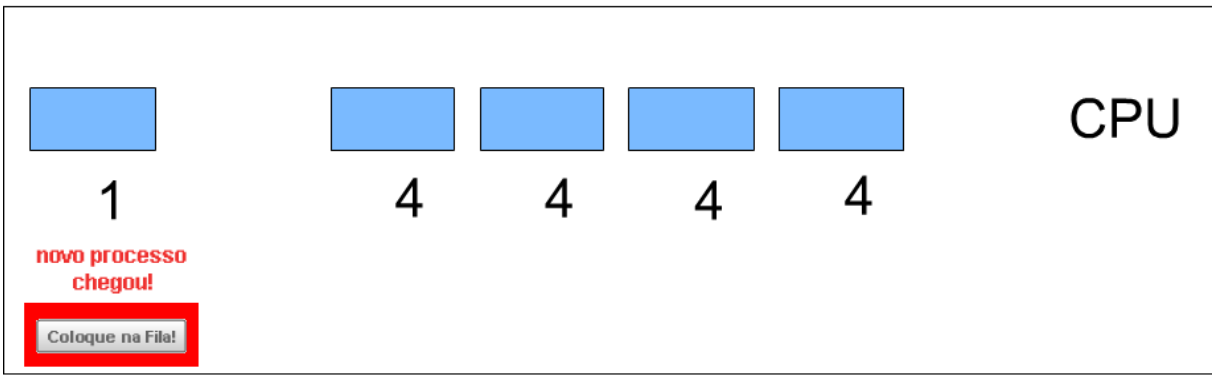

Figura 20 - Botão que reordena processos
Fonte: Autoria própria

Conforme mostrado na figura 21, o número com a duração do processo também é um objeto dentro da animação *flash*. O trecho de código anterior faz com que esse número acompanhe a posição do retângulo que representa o processo.

O clique no botão é um evento externo à animação. Abaixo tem o código para que o botão tenha funcionalidade.

```
onSelfEvent (release) {

    if (_root.fila_anda) {
       return;
    }
    _root.numbers[0] = parseInt (_root.timeproc1.text);
    _root.numbers[1] = parseInt (_root.timeproc2.text);
    _root.numbers[2] = parseInt (_root.timeproc3.text);
    _root.numbers[3] = parseInt (_root.timeproc4.text);
    _root.numbers[4] = parseInt (_root.timeproc5.text);
    _root.shellSort(5);

    _root.proc1._x = 60;
    _root.proc2._x = 160;
    _root.proc3._x = 260;
    _root.proc4._x = 360;
    _root.proc5._x = 460;

    _root.timeproc1.text = _root.numbers[0];
    _root.timeproc2.text = _root.numbers[1];
    _root.timeproc3.text = _root.numbers[2];
    _root.timeproc4.text = _root.numbers[3];
    _root.timeproc5.text = _root.numbers[4];


    _root.fila_anda = true;
}
```

Figura 21 - Mostra o trecho do código que dá funcionalidade ao botão
Fonte: Autoria própria

Evento onde ocorre clique no botão.

Duração de cada processo guardada no array "numbers"

```
    _root.numbers[0] = parseInt (_root.timeproc1.text);
    _root.numbers[1] = parseInt (_root.timeproc2.text);
    _root.numbers[2] = parseInt (_root.timeproc3.text);
    _root.numbers[3] = parseInt (_root.timeproc4.text);
    _root.numbers[4] = parseInt (_root.timeproc5.text);
```

Organiza um vetor com 5 posições de 0 a 4 através da função shellsort.

Essa função vai organizar o array "numbers"

```
   _root.shellSort(5);
```

A fila de processos é rearranjada colocando processos nos seus lugares.
Os processos são colocados de novo no ponto de partida da animação

```
    _root.proc1._x = 60;
    _root.proc2._x = 160;
    _root.proc3._x = 260;
    _root.proc4._x = 360;
    _root.proc5._x = 460;
```

ordena que o tempo do proc 1 deva aparecer em baixo do proc respectivo.

```
    _root.timeproc1.text = _root.numbers[0];
    _root.timeproc2.text = _root.numbers[1];
    _root.timeproc3.text = _root.numbers[2];
    _root.timeproc4.text = _root.numbers[3];
    _root.timeproc5.text = _root.numbers[4];
```

muda para verdadeiro e a fila anda.

```
_root.fila_anda = true; a flag
}
```

Figura 22 - Mostra o que acontece ao clicar no botão
Fonte: Autoria própria

Quando o processo chega ao final da linha (à direita), um *flag* muda e paralisa a fila. Depois, quando usuário clica no botão, o código chama *shellsort*, muda posição de todos na fila, muda o *flag* e a fila começa a andar novamente.

5.3 ROUND ROBIN

Neste trecho de código, a variável *flag* tem a função de informar o tipo de *status* da fila, conforme já explicado, neste caso a fila anda porque a *flag* está configurada como *true*, ou seja, verdadeira.

```
_root.numbers = new Array;
```

Esta instrução cria um vetor. Quando se tem a instrução *new array*, isso quer dizer que o vetor é instanciado na memória e o mesmo está pronto para ser utilizado.

```
for (i = 0; i < 5; i++)
```

Laço para percorrer 5 posições, ou seja, ele é executado cinco vezes para que a condição seja falsa e não seja mais executada.

```
_root.numbers[i] = 1+Math.randomInt(6);
```

Cria valores aleatórios no vetor.

```
shellSort(5);
```

Executa a função do algoritmo *shelsort*.

```
_root.timeproc1.text = _root.numbers[0];
_root.timeproc2.text = _root.numbers[1];
```

São configurados os números dos processos na ordem crescente.

```
_root.raio_círculo = 120
```

Valor do raio da circunferência = 120.

```
_root.alpha = 0;
```

Ângulo alfa é a posição inicial do processo antes de girar.

```
  _root.beta = 360 / 5;
```

Ângulo beta é fracionado para andar de 72,5 em 72,5 graus porque são cinco processos.

_root.proc1._x = 350 + _root.raio_círculo * Math.cos (_root.alpha * 2 * 3.1415 / 360.);

Processo na posição x recebe 350 mais o valor do raio do círculo vezes o cosseno que é obtido pela multiplicação do ângulo alfa vezes o número de pi ao quadrado dividido por 360 graus que é a volta completa do círculo.

_root.proc1._y = 150 + _root.raio_círculo * Math.sin (_root.alpha * 2 * 3.1415 / 360.);

Processo na posição y recebe 350 mais o valor do raio do círculo vezes o cosseno que é obtido pela multiplicação do ângulo alfa vezes o número de pi ao quadrado dividido por 360 graus que é a volta completa do círculo.

_root.alpha = _root.alpha + _root.beta;

O ângulo alpha recebe o valor anterior do ângulo alpha somado com o valor do ângulo beta.

5.4 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Neste capítulo foi abordada a questão prática deste projeto, ou seja, a programação das animações. O mesmo contém trechos do código fonte das animações e a sua explicação do que significa determinada instrução.

Foi definido que o código fonte fosse tratado como imagem, por uma questão de organização e por facilitar a compreensão.

6 RESULTADOS

Como resultado final deste trabalho foi produzido animações e a construção do site. Em relação ao desenvolvimento das animações foi possível identificar a realidade de como o conteúdo de sala de aula é ministrado e posteriormente qual é a contribuição destas animações com vistas a um processo de ensino aprendizagem com resultados mais satisfatórios.

As animações poderão contribuir principalmente no aspecto da análise do aluno em relação ao comportamento de estruturas, de modo que o aluno compreenda como a mesma se comporta visualmente.

A construção do *site* possibilitará uma nova forma de aprendizado, pois com poucos cliques, é possível que se torne mais abrangente e precisa a assimilação do aluno em relação ao conteúdo ministrado. Sendo esta a principal contribuição, pois construir *sites* não é o foco principal desta pesquisa, mas sim melhorar o processo de ensino aprendizagem.

6.1 TRABALHOS FUTUROS

Esta é a parte de sugestões referentes ao que pode ser realizado para a continuação deste trabalho. Tem papel de alta relevância, pois caracteriza no aprimoramento da ideia inicia, sendo esta transformada em novos estudos de mesma essência.

Com relação ao aspecto global deste trabalho de conclusão de curso, foi constatado que o mesmo pode ser ampliado nos seguintes aspectos:

- Incluir interatividade nas animações, onde o aluno pode experimentalmente modificar o tempo de um processo e constatar visualmente o que acontece no escalonador com o tempo alterado;
- Criar outras animações de mecanismos dinâmicos de SO, tais como:
    - Memória virtual;
    - Descarte de páginas (FIFO, menos usada, etc.);
    - Sincronia de processos: problemas como o jantar dos filósofos.

Para tanto o trabalho de conclusão de curso possibilitou uma nova gama de conhecimentos, onde o mesmo foi importante na construção do pensamento crítico, mostrando que medidas simples podem ser de grande valia.

Após as intervenções nas animações e no *site* como produto final deste trabalho, observaram-se pontos fortes e pontos fracos resultando de maneira positiva a realização deste trabalho.

Os pontos positivos são percebidos pelo aprendizado proporcionado por meio da revisão de conceitos adquiridos no longo do curso, aperfeiçoando-os e possibilitando uma nova visão do assunto.

Novas práticas de abordagem de conteúdo proporcionam crescimento profissional e acadêmico, visando o aperfeiçoamento do professor, o que acaba refletindo no comportamento do aluno e consequentemente melhorando a relação teoria e prática para ambos.

Os pontos negativos existem e são das mais diversas naturezas, seja em relação ao nível dos alunos, ou por contrastes regionais de ensino, ou ainda pela falta de conhecimento da linguagem *flash*. Portanto em relação ao (des) conhecimento no que se refere a operar a ferramenta SwishMax, podem ser transpostos ao longo da experimentação de novas formas de abordagem dos conteúdos programáticos.

REFERÊNCIAS

ARRUDA, P. Produtor executivo da Animaking. Disponível em: <http://www.producaocultural.org.br/wpcontent/uploads/livroremix/phillipearruda.pdf> Acesso em: 28 de Abr. 2012.

DEHAAN, J. **Flash MX 2004: guia autorizado macromedia**. Rio de Janeiro: Elsevier, 2004. 461 p.

DEITEL, H. M.; DEITEL, P. J.; **Java Como Programar; tradução Carlos Arthur Lang Lisbôa.** - 4ª Edição – Porto Alegre: Bookman, 2003 – Reimpressão 2004.

FERREIRA, S. **Adobe flash CS3: crie e desenvolva animações profissionais e sites interativos**. São Paulo: Digeratti Books, 2008. 125 p.

FIGUEIREDO E., LOBATO, C., DIAS, K., LEITE, J. e LUCENA, C. **Um Jogo para o Ensino de Engenharia de Software Centrado na Perspectiva de Evolução.** Anais Eletrônicos do XXVII Congresso da SBC, Rio de Janeiro, 2007.

GUEDES, J. R., GUEDES, C. L. **Hipermídia para auxilio ao Ensino de Sistemas Operacionais.** In: II Congresso Sul Catarinense de Computação, 2006, Criciúma. Anais do II Congresso Sul Catarinense de Computação, 2006.

MAIA, L. P. (2001). **SOsim:Simulador para o Ensino de Sistemas Operacionais.** Tese de Mestrado, Núcleo de Computação Eletrônico da Universidade Federal do Rio de Janeiro (NCE/UFRJ), Marco.

OLIVEIRA NETTO, A. A. de. **IHC - Interação Humano Computador: modelagem e gerência de interfaces com o usuário**. Florianópolis: Visual Books, 2004. 120 p. ISBN 85-7502-138-9.

PIOLOGO, R.; PIOLOGO, R. **Flash animado com os irmãos Piologo.** São Paulo: Novatec, 2009. 269 p.

RODRIGUES, T. S. **Marechal: jogo pedagógico para a disciplina de sistemas operacionais**. Ponta Grossa, PR, 2008. 40 f. Trabalho de Conclusão de Curso

(Graduação) - UTFPR. Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas. Ponta Grossa, 2008.

SILBERSCHATZ, A., GAGNE G., GALVIN P. B. **Fundamentos de Sistemas Operacionais**. 6ª. ed. Rio de Janeiro: LTC, 2004.

SILVA, A. A.; FRETTA, R. **Arqmax 2008: Software Gerenciador de documentos.** Disponível em: <http://www.mxstudio.com.br/tecnologia/geral/a-evolucao-da-internet/>. Acesso em: 05 jul. 2011.

SWISHZONE. Disponível em: <http://www.swishzone.com/index.php?area=aboutus&tab=aboutus>. Acesso em: 13 de Fev. 2012.

TANENBAUM, A. S. **Sistemas Operacionais Modernos**. 2ª. Ed. Rio de Janeiro: Pearson Brasil, 2003.