Alexandre Martins Ferreira de Sousa

Um ambiente em nuvem para criação colaborativa de jogos

Orientador:
Prof. Dr. Flávio Soares Corrêa da Silva

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
UNIVERSIDADE DE SÃO PAULO

São Paulo

1º de Dezembro de 2011

Sumário

1 Introdução

Até meados dos anos 1990, o desenvolvimento de jogos eletrônicos era em grande parte restrito a especialistas. Com a crescente facilidade de uso dos sistemas de computador e com a popularização da Internet, pessoas comuns passaram a ter maior destaque na produção e distribuição de conteúdo - em particular, games. Ferramentas como: The Games Factory, Game Maker, RPG Maker, entre outras, permitiam que usuários sem grandes conhecimentos de computação criassem seus próprios joguinhos, o que encorajava o estudo de: game design, matemática, programação, storytelling, etc.

Sob uma ótica cultural, Lawrence Lessig[1] descreve em seu livro Remix[2] a discrepância entre as leis atuais de copyright, consideradas antiquadas, e a crescente disponibilidade e facilidade de uso das novas tecnologias de informação. Lessig define dois conceitos importantes:

- **Read Only Culture**: onde um pequeno grupo de profissionais produz toda a cultura, sendo esta consumida pelas massas. O público desempenha um papel meramente passivo;
- **Read/Write Culture**: onde o público geral tem a possibilidade de interagir com a cultura, adicionando, modificando e influenciando o conteúdo ali presente. De acordo com Lessig, uma abordagem de leitura e escrita, além de incentivar a criatividade dos indivíduos, torna a cultura mais rica e inclusiva.[3]

Este trabalho de conclusão de curso está imerso na cultura "*Read/Write*". De codinome **Chinchilla**, o sistema aqui desenvolvido se diferencia de ferramentas convencionais de criação de jogos pois, além de eliminar a necessidade de instalação de programas adicionais - tanto para criar quanto para jogar -, adiciona uma faceta colaborativa integrada ao processo de construção dos games. Da forma como foi idealizado, o projeto favorece a criação de protótipos rápidos de jogos.

[1] Professor de direito de Harvard, é uma voz respeitada no que ele considera como sendo a "guerra do copyright".

[2] `http://en.wikipedia.org/wiki/Remix_(book)`. Último acesso, 07/06/2011.

[3] `http://en.wikipedia.org/wiki/Lawrence_Lessig`. Último acesso, 07/06/2011.

Do ponto de vista técnico, o advento de novas tecnologias como o HTML5[4] permite que se crie aplicações cada vez mais interativas e ricas que rodem em navegadores modernos de Internet. O software desenvolvido neste trabalho possibilita que grupos de pessoas se reúnam num site de Internet e criem, colaborativamente, protótipos rápidos de games. Tal software, inspirado no conceito de **computação em nuvem**, é constutído de três partes:

- **Sistema web colaborativo**: interage diretamente com os usuários, permitindo que eles se reúnam em salas virtuais, similares às de bate-papo, e criem seus jogos colaborativamente;
- **Máquina virtual (VM)**: também conhecido como Runtime Engine, roda no navegador de Internet os games criados pelos usuários. Tal processo se dá pela interpretação de um bytecode especial;
- **Compilador**: os usuários usam uma linguagem de programação de alto nível e de fácil aprendizado para desenvolver os jogos. O compilador traduz código desta linguagem para o bytecode a ser interpretado pela máquina virtual.

Sendo este um projeto em nuvem, acreditamos que desenvolvedores de jogos independentes e/ou equipes pequenas espalhadas geograficamente possam se beneficiar de soluções como a que desenvolvemos. Além disso, o sistema também pode ser usado para, de uma forma mais divertida e colaborativa, dar aulas sobre programação.

[4] Quinta versão da linguagem HTML, HyperText Markup Language.

2 *Tecnologias e conceitos estudados*

2.1 Visão geral

Cada uma das três esferas do projeto ataca um problema bem específico. Embora o conjunto de tecnologias seja diverso, objetiva-se sempre a boa integração dos elementos do sistema.

Veremos a seguir o que, tratando-se de novidades, foi necessário aprender para desenvolver cada uma das vertentes da plataforma. Conceitos fundamentais aprendidos no curso e que tenham grande relevância no projeto serão detalhados ao longo da monografia, o que (esperamos) fornecerá uma melhor contextualização das disciplinas.

2.2 Máquina virtual

Em Ciência da Computação, "máquina virtual é o nome dado a uma máquina, implementada através de software, que executa programas como um computador real".[1]

Com a concepção da plataforma, propôs-se que usuários fossem capazes de criar, colaborativamente, seus próprios joguinhos através do (e para o) navegador de Internet. Para que este último objetivo fosse cumprido, foi idealizada uma máquina virtual.

Nossa máquina virtual, também conhecida como VM[2], simula um hardware fictício cuja responsabilidade é executar, no browser, games criados pelos usuários. Desenvolvida na linguagem JavaScript, a VM faz intenso uso de um novo conjunto de tecnologias conhecido como HTML5, que traz inovadoras APIs[3] para o universo das páginas de Internet. Entre os muitos recursos oferecidos pelo HTML5, destacamos aqui: controle de conteúdo multimídia e um mecanismo para desenvolvimento de gráficos em duas dimensões - ferramentas essenciais para

[1] `http://pt.wikipedia.org/wiki/M\'aquina_virtual`. Último acesso, 21/06/2011.

[2] Virtual Machine

[3] Application Programming Interface, "conjunto de rotinas e padrões estabelecidos por um software para a utilização das suas funcionalidades por programas aplicativos que não querem envolver-se em detalhes da implementação do software, mas apenas usar seus serviços". Fonte: `http://pt.wikipedia.org/wiki/API`. Último acesso, 21/06/2011.

jogos. HTML5 é a quinta versão da linguagem HTML, linguagem usada para a construção das páginas da web.

JavaScript é uma linguagem de script cujo uso primário é promover maior interação e controle em páginas HTML, já que estas, sozinhas, apresentam baixo dinamismo. Concebida como uma linguagem orientada a objetos baseada em protótipos, possui tipagem dinâmica e fraca, além de possibilitar o uso de elementos de programação funcional. Para que esta tecnologia pudesse ter sido aprendida até um nível básico de *expertise*, foi utilizado Adams(1), além de tutoriais na Internet: *w3cschools.com*[4], *Dev.Opera*[5], *Mozilla Developer Network*[6], entre outros.

A fonte de aprendizado de HTML5 foi a Internet. Sites como *Dive Into HTML5*[7], além dos diversos tutoriais sobre HTML5 do *Dev.Opera*[8], mostraram-se extremamente úteis. Apesar de esta nova tecnologia ser empolgante no que tange às possibilidades que ela abre, deve-se ter consciência de que ela não é um fim em si, mas apenas um meio, uma ponte com a qual soluções para usuários reais podem ser desenvolvidas.

2.3 Compilador

Para que os usuários criem seus jogos, devem de alguma forma elaborar instruções para o computador. Neste trabalho, decidimos inventar uma linguagem de programação textual. Isto facilita a implementação do sistema web, já que o código-fonte dos games deve ser sincronizado na tela de diversos usuários através da Internet.

Em Ciência da Computação, "um compilador é um programa que recebe como entrada um programa em uma linguagem de programação - a linguagem *fonte* - e o traduz para um programa equivalente em outra linguagem - a linguagem *objeto*" (2, p. 1). Neste trabalho, o compilador traduz o código do jogo do usuário, escrito numa linguagem textual de alto nível, para um bytecode a ser interpretado pela Máquina Virtual.

Disciplinas como: Laboratório de Programação, Linguagens Formais e Autômatos, Conceitos Fundamentais de Linguagens de Programação e Estrutura de Dados fornecem alguns *insights* quanto à construção de compiladores. Infelizmente, por não haver oferecimento, não houve oportunidade de cursar uma disciplina específica sobre compiladores. Mesmo assim, o famoso "livro do dragão", Aho et al.(2), mostrou-se um recurso valiosíssimo nesta empreitada:

[4] `http://www.w3schools.com/js/default.asp`. Último acesso, 21/06/2011
[5] `http://dev.opera.com/articles/javascript/`. Último acesso, 21/06/2011
[6] `https://developer.mozilla.org/en-US/learn/javascript`. Último acesso, 21/06/2011
[7] `http://diveintohtml5.org/`. Último acesso, 21/06/2011
[8] `http://dev.opera.com/`. Último acesso, 21/06/2011

pode-se aprender alguns conceitos elementares sobre compiladores de forma auto-didata.

Diferentemente da VM, o compilador roda num servidor. Para a construção do sistema de compilação foi usada a linguagem C++. Boas fontes de aprendizado e consulta desta linguagem incluem: Hickson(3) e Drozdek(4).

2.4 Sistema web colaborativo

"Software colaborativo (ou groupware) é um software que apoia o trabalho em grupo, coletivamente. Skip Ellis o definiu como um 'sistema baseado em computador que auxilia grupos de pessoas envolvidas em tarefas comuns (ou objetivos) e que provê interface para um ambiente compartilhado'."[9]

Máquina Virtual, Compilador e Sistema web constituem, juntos, um ambiente de criação de jogos em que os usuários colaboram, em tempo real[10], entre si. Enquanto a VM e o Compilador estão intimamente ligados à criação e execução dos jogos, o Sistema web implementa o ambiente compartilhado que possibilita o trabalho em grupo através da Internet.

Inspirado nos jogos online, o software colaborativo aqui desenvolvido se resume à união planejada de diversas tecnologias já existentes. A integração de tecnologias variadas disponíveis na Internet com conhecimentos de: Redes de Computadores, Banco de Dados, Engenharia de Software, Programação Orientada a Objetos, entre outras disciplinas, tornou possível a implementação desta faceta do trabalho de conclusão de curso.

Finalmente, citemos a famigerada sopa de letrinhas: PHP5, SQL, HTML5, CSS3 e jQuery[11] permeiam nosso sistema.

[9] `http://pt.wikipedia.org/wiki/Software_colaborativo`. Último acesso, 17/09/2011
[10] desconsiderando a latência da rede
[11] `http://www.jquery.com`. Último acesso, 17/09/2011

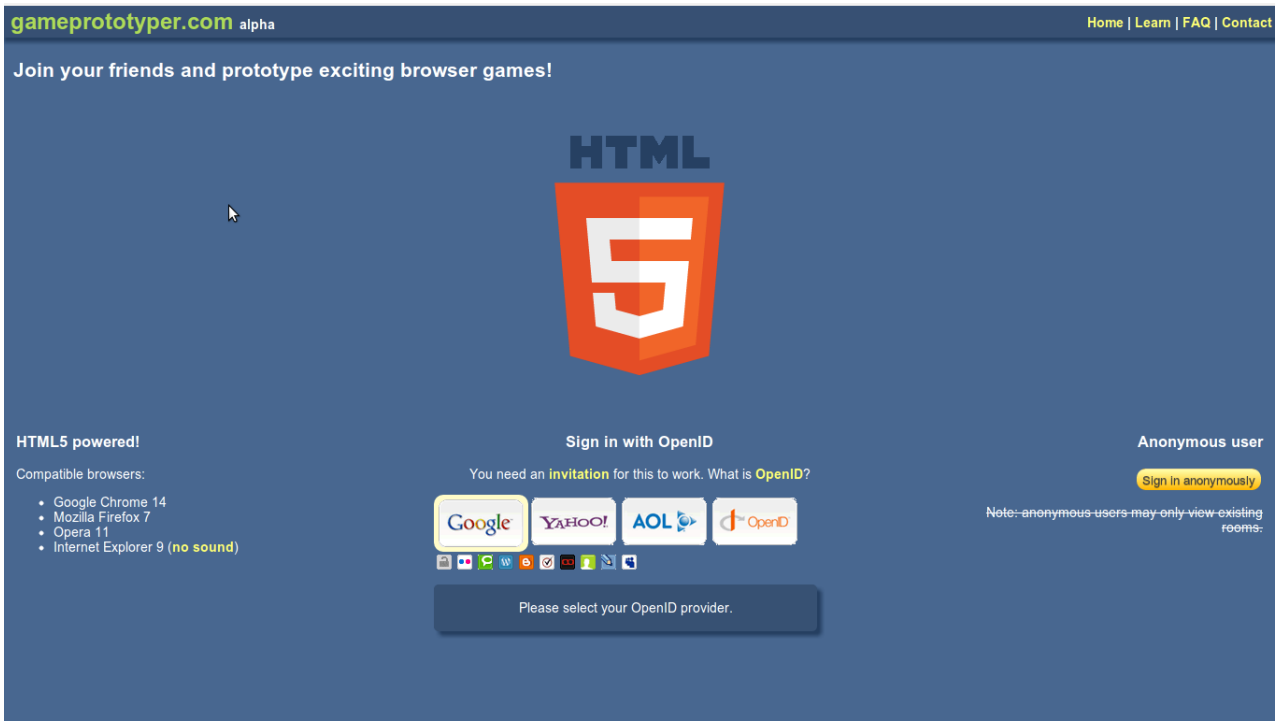


Figura 2.1: Tela de autenticação

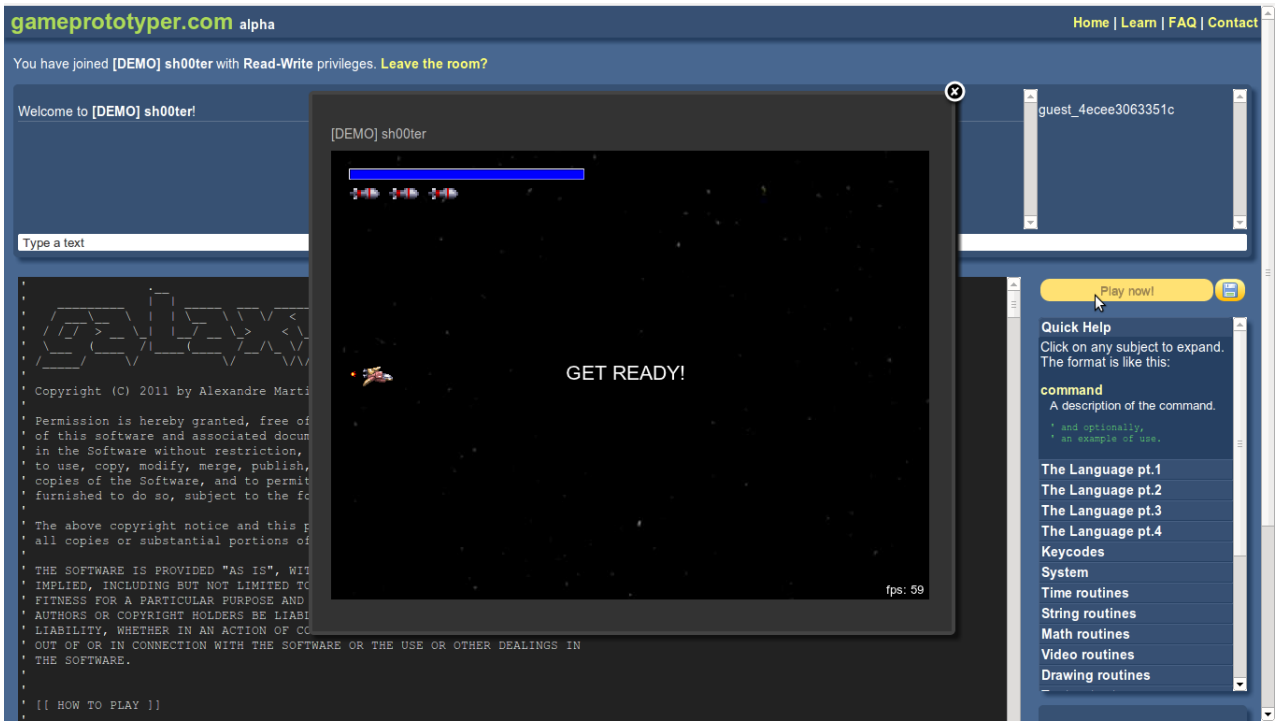


Figura 2.2: A execução de um jogo

3 *Atividades realizadas*

3.1 Visão geral

Neste capítulo, detalharemos cada uma das etapas representadas a seguir:

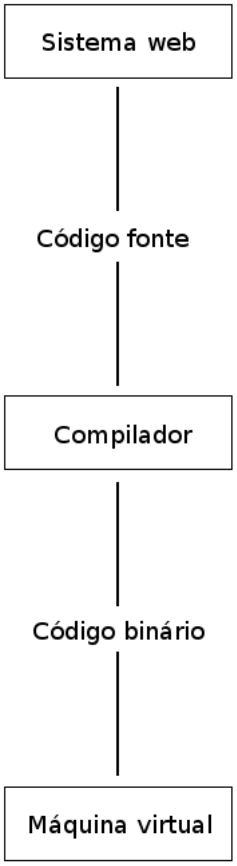


Figura 3.1: Fluxo de dados na plataforma **Chinchilla**

3.2 Máquina virtual

3.2.1 Introdução

A VM é um computador escrito em linguagem JavaScript. Inspirada na *Arquitetura de Harvard*[1], é uma *arquitetura de computador* que pode ser entendida esquematicamente:

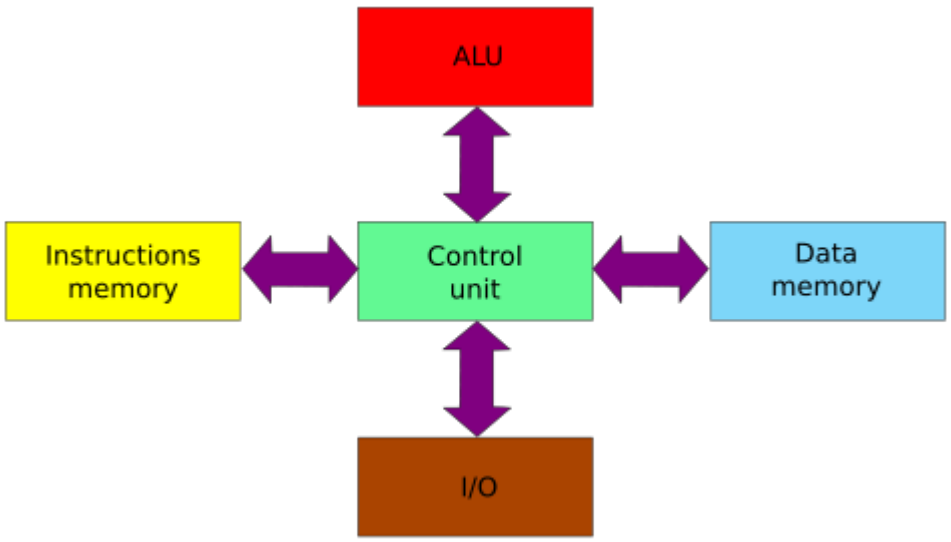


Figura 3.2: Fonte: Wikipedia

Do ponto de vista prático, o termo *arquitetura de computador* se refere à forma como dispositivos de hardware cooperam entre si, de maneira a criar computadores que atinjam objetivos funcionais, de performance e de custo[2].

Na *Arquitetura de Harvard*, uma máquina reúne os seguintes componentes[3]:

- **Unidade de Controle (UC)**: responsável por coordenar o funcionamento das demais esferas da máquina através do ciclo *Fetch-Decode-Execute*, que detalharemos a seguir;
- **Unidade Lógica e Aritmética (ULA)**: uma "grande calculadora" que faz cálculos aritméticos e lógicos a pedido da Unidade de Controle;
- **Dispositivos de Entrada e Saída**: responsáveis por interagir com os usuários finais;

[1] `http://pt.wikipedia.org/wiki/Arquitetura_Harvard`. Último acesso, 16/07/2011
[2] `http://en.wikipedia.org/wiki/Computer_architecture`. Último acesso, 16/07/2011
[3] Omitimos aqui o conjunto de registradores: são memórias rápidas usadas frequentemente nos programas e, assim como a UC e a ULA, fazem parte do processador.

- **Memória de Instruções**: local físico, ligado diretamente ao processador, onde ficam armazenadas as instruções do programa corrente;
- **Memória de Dados**: local físico, também ligado ao processador, onde se armazenam dados utilizados pelos programas, como por exemplo o conteúdo de variáveis.

Disto, vemos que a arquitetura acima é muito similar à de *Von-Neumann*, tendo como principal diferença a separação física entre memória de instruções e de dados. Na plataforma **Chinchilla**, uma vez carregada, a memória de instruções se torna somente-leitura. Ela não pode ser modificada (ou sequer lida) por programas que estejam rodando na Máquina Virtual: apenas a Unidade de Controle do "hardware virtual" é capaz de acessá-la.

Ressaltamos também que a máquina virtual é inspirada na arquitetura RISC. RISC, ou *Reduced Instruction Set Computer*, é caracterizada por ter um conjunto reduzido de operações de máquina[4]. Diferentemente da arquitetura CISC, *Complex Instruction Set Computer*, naquela as instruções são simples e o tempo médio de execução é reduzido, não sendo necessária a existência de microcódigo. Em contrapartida, o número total de instruções dos programas aumenta.

3.2.2 Unidade de Controle

A função de um processador é "executar programas residentes na memória buscando as instruções e executando-as uma após a outra"[5]. A Unidade de Controle, parte do processador, deve interpretar os comandos residentes na memória de instruções. Ela repetidamente executa o que se conhece como ciclo *Fetch-Decode-Execute*:

- Busca a próxima instrução, apontada pelo registrador PC (Program Counter) e a guarda temporariamente;
- Incrementa o PC, de forma que aponte para a próxima instrução;
- Examina a instrução obtida no passo 1: determina seu tipo, seus operandos, etc.;
- Executa a instrução.

[4] `http://www.poliparatodos.poli.usp.br/modulos/17/paginas/html4/Arquitetura\%20CISC\%20e\%20RISC.pdf`. Último acesso, 17/07/2011

[5] `http://www.poliparatodos.poli.usp.br/modulos/17/paginas/html4/PCS405V2.pdf`. Último acesso, 17/07/2011

3.2.3 Unidade Lógica e Aritmética

Em computação, uma Unidade Lógica e Aritmética, também parte do processador, é um "circuito digital que desempenha operações lógicas e aritméticas"[6]. Na plataforma **Chinchilla**, a ULA é um objeto com métodos responsáveis por operações como: soma, subtração, multiplicação, divisão, mudança de sinal, and/or/xor/not binários, comparações entre valores, entre outras.

Como foi criada em software, tomamos a liberdade de implementar também, nesta parte do processador, operações em strings de texto: concatenação, extração de substrings, etc. Isto facilitou a criação das funções de texto em camadas mais altas da plataforma. Outras operações complexas envolvendo números de ponto flutuante, tais como: extração de raiz quadrada, cálculo de logaritmos, entre outras, também fazem parte deste componente e são executadas num único clock da máquina virtual. Isto posto, de um ponto de vista mais estrito, consideramos o nome ULA pouco adequado, porém inteligível. Finalmente, algo que poderia ser extraordinariamente complexo para se implementar num hardware físico, pode ser feito de forma trivial numa máquina implementada em software.

3.2.4 Memória de Instruções

Uma memória é uma sequência de células. Uma célula é definida como a menor unidade endereçável[7] e, diferentemente das máquinas convencionais, na máquina virtual aqui desenvolvida pode-se carregar valores booleanos, números inteiros ou de ponto flutuante, ou ainda strings de texto numa única célula. A forma como esses dados são representados a nível de bits depende da máquina hospedeira, não interferindo nas camadas mais altas da plataforma de criação de jogos.

Em **Chinchilla**, uma instrução é uma tripla:

(**opcode**, **operand1**, **operand2**)

Figura 3.3: Esquema de uma instrução

O campo opcode identifica o tipo da instrução. Em contrapartida, operand1 e operand2 são campos opcionais que podem ser registradores ou constantes. Do ponto de vista de implementação, a memória de instruções é uma sequência composta dessas triplas, sendo carregada uma

[6] `http://en.wikipedia.org/wiki/Arithmetic_logic_unit`. Último acesso, 17/07/2011

[7] `http://www.poliparatodos.poli.usp.br/modulos/17/paginas/html4/PCS405V2.pdf`. Último acesso, 17/07/2011

única vez: no momento em que a máquina virtual é ligada.

Vejamos um exemplo prático. O seguinte código em **Chinchilla Assembly** calcula recursivamente o fatorial de um inteiro:

```
_fat:
push bp
mov bp, sp
mov adr, bp
add adr, 3
load a, adr
mov b, 1
scmp a, b
mov a, true
jge L0
mov a, false
L0:
lcmp a, false
je L1
mov adr, bp
add adr, 3
load a, adr
push a
mov adr, bp
add adr, 3
load a, adr
mov b, 1
sub a, b
push a
call _fat
pop a
mov b, fun
pop a
mul a, b
mov fun, a
jmp L3
L1:
```

```
mov fun, 1
L3:
pop bp
ret
```

Quando usamos tal código como *input* para o **Chinchilla Assembler**, também chamado de montador, recebemos como *output* uma sequência de instruções conhecida como **código de máquina** (ou, num abuso de linguagem, **código binário**):

29, 2, 0, 3, 2, 1, 3, 4, 2, 43, 4, 3, 5, 6, 4, 2, 7, 1, 15, 6, 7, 2, 6, true, 21, 11, 0, 2, 6, false, 10, 6, false, 16, 30, 0, 3, 4, 2, 43, 4, 3, 5, 6, 4, 29, 6, 0, 3, 4, 2, 43, 4, 3, 5, 6, 4, 2, 7, 1, 46, 6, 7, 29, 6, 0, 31, 1, 0, 30, 6, 0, 3, 7, 5, 30, 6, 0, 48, 6, 7, 3, 5, 6, 8, 32, 0, 2, 5, 1, 8, 32, 0, 30, 2, 0, 33, 0, 0

A memória de instruções da máquina virtual é então carregada com este **código binário** e estará pronta para interpretá-lo.

3.2.5 Memória de Dados

A Memória de Dados da VM é um grande vetor de células. É dividida como se segue:

Figura 3.4: Subdivisão da Memória de Dados

Caracterizada como memória de acesso aleatório, qualquer célula pode ser acessada em tempo *O(1)*. Cada uma dessas regiões serve a um propósito específico:

- **Reservado**: porção de memória reservada para uso interno da VM;
- **Dados estáticos**: certos objetos do programa de alto nível do usuário final, tais como variáveis globais, podem ter seus endereços em memória determinados já em tempo de

compilação. Esses dados são colocados numa região de memória conhecida como **estática**;

- **Memória de pilha**: variáveis locais a uma função ou procedimento são alocados numa estrutura *FILO*[8]. Seus endereços físicos não são conhecidos em tempo de compilação. Esta região de memória cresce em tempo de execução, conforme a demanda do programa, e é conhecida como **pilha de execução**;

- **Memória heap**: certos dados podem sobreviver à chamada da função ou do procedimento que os originaram e, portanto, são alocados numa área especial da memória denominada **heap**. De acordo com (2, p. 289), "o heap é uma porção da memória usada para dados que residem indefinidamente, ou até que o programa os exclua explicitamente". Assim como a pilha, o **heap** também cresce, em tempo de execução, conforme a demanda.

A máquina virtual conta com um sistema de alocação dinâmica de memória[9] simples inspirado na sugestão de (2, p. 295). Na técnica conhecida como **Lista de Livres**, conecta-se regiões de memória não alocada utilizando uma lista ligada. A primeira célula de cada região é um ponteiro para a próxima. Nesta abordagem, as operações de alocação e liberação de memória são triviais. Porém, se as requisições por memória pedem um número variável de células, a busca por uma região contígua de memória pode ser cara, já que buscas em listas ligadas consomem tempo linear em seu tamanho.

Devido a restrições de tempo, não foi implementado um coletor automático de lixo.

3.2.6 Dispositivos de Entrada e Saída

"Conteúdo é tudo: ofereça bons gráficos, sons, jogabilidade, ou seja, conteúdo! Tenha certeza de que seu jogo é divertido." (5, p. 36)

Para que um game seja interessante, precisamos de dispositivos de E/S que tornem agradável a experiência do jogador. Assim, para que o usuário da plataforma crie um joguinho que toque uma música bacana enquanto mostra gráficos interessantes, a máquina virtual deve ser capaz de lidar com tais tarefas. No fundo, esses dispositivos são os responsáveis por dar vida à plataforma de jogos.

Chinchilla conta com o seguinte conjunto de dispositivos E/S:

[8] First-In Last-Out

[9] também chamada de *heap-based memory allocation*, é a alocação de memória para uso num programa de computador durante seu tempo de execução.

- **Vídeo**
- **Áudio**
- **Teclado**
- **Mouse**

Vídeo

Um dos recursos mais empolgantes do HTML5 é o **<canvas>**. Trata-se de uma superfície de desenho na qual, a partir de chamadas JavaScript, pode-se mostrar formas geométicas simples, textos e imagens. Associado a temporizadores, pode-se criar animações e, em particular, jogos.

Versões recentes dos mais populares navegadores de Internet (Internet Explorer, Mozilla Firefox, Google Chrome, Opera, Safari) já implementam o canvas[10] - alguns inclusive contando com aceleração de hardware[11]. A ampla disponibilidade, associada à ótima velocidade deste novo recurso do HTML5, é condição necessária para o bom funcionamento de **Chinchilla**.

A figura abaixo ilustra um jogo simples criado durante o desenvolvimento de **Chinchilla**:

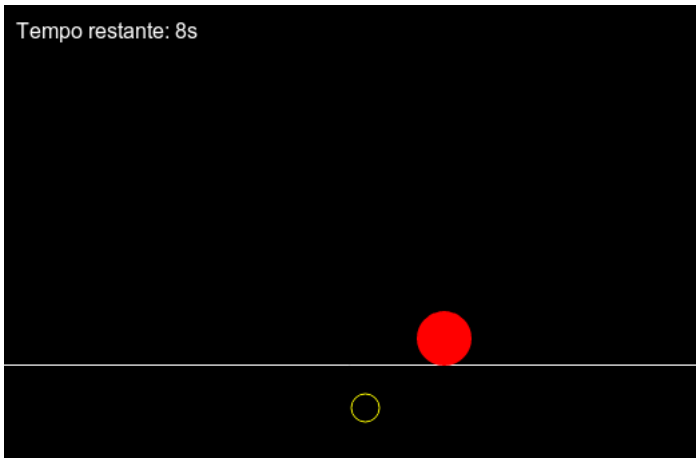


Figura 3.5: Killer Ball, jogo feito com primitivas simples

Do ponto de vista da máquina virtual, o canvas desempenha o papel de "hardware de vídeo". Por questões de eficiência, diferentemente de máquinas antigas, não se trabalha diretamente com os pixels do vídeo. O dispositivo de vídeo provê um conjunto de funcionalidades pré-definidas que podem ser acessadas através do lançamento de **interrupções**.

[10] `http://www.w3schools.com/html5/tag_canvas.asp`. Último acesso, 24/07/2011

[11] `http://blogs.msdn.com/b/ie/archive/2010/07/01/ie9-includes-hardware-accelerated-canvas.aspx`. Último acesso, 24/07/2011

"Em Ciência da Computação, uma interrupção é um sinal de um dispositivo que tipicamente resulta em uma troca de contexto, isto é, o processador pára de fazer o que está fazendo para atender o dispositivo que pediu a interrupção."[12]

Em computadores digitais tradicionais, há mecanismos para se iniciar rotinas em software em resposta a uma interrupção. Tais rotinas, conhecidas como *rotinas de serviço de interrupção*, têm seus endereços armazenados num vetor de interrupções em memória. A motivação do conceito de interrupção vem da necessidade de se evitar desperdício de tempo do processador nos *polling loops*. Neles, o processador era o responsável por checar cada dispositivo de hardware, procurando por eventos relevantes.

Na plataforma **Chinchilla**, o vídeo é acessado através de **interrupções de software**. Suponha que o usuário queira mostrar uma figura na tela. Por trás das chamadas de alto nível da linguagem de programação, há instruções assembly responsáveis por:

- Empilhar parâmetros relevantes para a tarefa, tais como: qual imagem será mostrada, sua posição na tela, etc.;
- Empilhar o número da tarefa "mostrar figura";
- Lançar uma interrupção de software que chame o dispositivo de vídeo.

Os passos acima caracterizam uma **chamada de sistema** (*system call*). No momento em que a interrupção é lançada, o processamento do jogo é suspenso temporariamente e a VM passa o controle da aplicação para o tratador de chamadas de vídeo. Lá, o número da tarefa a ser desempenhada, valor do topo da pilha, é verificado. Procede-se então com a real execução da rotina "mostrar figura", concebida através de chamadas JavaScript destinadas ao canvas. O processamento do jogo é posteriormente resumido.

Exemplo:

```
'' Draws an image.
'' @code
''     drawImage "bunny.png", 10, 10
'' @endcode
'' @param imageName name of the image to be displayed
'' @param x x-coordinate
```

[12] `http://pt.wikipedia.org/wiki/Interrup\%C3\%A7\%C3\%A3o_de_hardware`. Último acesso, 24/07/2011

```
'' @param y y-coordinate
fun drawImage(imageName, x, y)
    __asm "mov adr, bp"
    __asm "add adr, 3"
    __asm "load a, adr" ' a = imageName
    __asm "add adr, 1"
    __asm "load b, adr" ' b = x
    __asm "add adr, 1"
    __asm "load c, adr" ' c = y
    __asm "push c"
    __asm "push b"
    __asm "push a"
    __asm "push 18"
    __asm "int 7"
endfun
```

Áudio e as patentes

O HTML5 introduz a tag **<audio>**. Usada para definir sons, músicas e outras *streams* de áudio, é suportada nos principais navegadores recentes de Internet sem que seja necessária a instalação de programas (codecs, plugins, etc.) adicionais. Infelizmente, ainda em Julho de 2011, este novo recurso enfrenta problemas técnicos e legais.

Do ponto de vista técnico, o áudio do HTML5 ainda deixa a desejar. Diversas fontes[13] relatam problemas envolvendo questões triviais. Browsers com boa avaliação, entre eles o Google Chrome, suportam a tecnologia, mas engasgam em tarefas muito simples. Experimentos simples revelaram que tocar um som repetidas vezes, por exemplo, faz com que ele deixe de funcionar. Reproduzir uma variedade de efeitos sonoros simultaneamente, situação essencial em jogos eletrônicos, faz com que o som comece a "*pipocar*", o que estraga toda a experiência do jogador. Outros navegadores apresentam problemas similares.

Para agravar o quadro, ainda mais preocupantes do que os problemas técnicos são as questões legais. Na presente data, não há sequer um único formato de áudio que funcione em todos os principais navegadores. A nova tag **<video>** enfrenta problemas análogos. A presidente executiva da Vquency Pty Ltd Australia, Dr. Silvia Pfeiffer, analisa a situação:(6)

[13] "Probably, Maybe, No": The State of HTML5 Audio - `http://24ways.org/2010/the-state-of-html5-audio`. Último acesso, 18/07/2011

> "[...] video element support had been implemented in Mozilla Firefox, Apple Safari/Webkit, Google Chrome, and Opera, but each browser vendor had done their own analysis of the situation at hand and different baseline codecs had been chosen. While Mozilla and Opera only supported Ogg Theora/Vorbis, Google decided to support both, Ogg Theora/Vorbis and H.264, and Apple decided to support only H.264"

O quadro a seguir, obtido do site da w3cschools[14], ilustra a situação a que os usuários do recurso **<audio>** estão sujeitos:

Audio Formats

Currently, there are 3 supported formats for the audio element:

Format	IE 9	Firefox 3.5	Opera 10.5	Chrome 3.0	Safari 3.0
Ogg Vorbis	No	Yes	Yes	Yes	No
MP3	Yes	No	No	Yes	Yes
Wav	No	Yes	Yes	No	Yes

Figura 3.6: HTML5 Audio e suas incompatibilidades

Microsoft e Apple implementam em seus navegadores o padrão *MP3*[15]. Apesar de ser amplamente usado, tal padrão é patenteado. Embora os decodificadores sejam implementados nos browsers de Internet, não está claro se desenvolvedores de jogos que façam uso desses *decoders* devem pagar *royalties* aos detentores das patentes. De acordo com a Wikipedia[16],

> "Em setembro de 1998, o Instituto Fraunhofer enviou um comunicado a diversos desenvolvedores de programas MP3, exigindo cobrança de royalties por essa patente. O comunicado informava que o licenciamento era necessário para 'distribuir e/ou vender decodificadores e/ou codificadores', e que os produtos não licenciados infringiam os 'direitos sobre a patente do Instituto Fraunhofer e da Thomson. Para produzir, vender e/ou distribuir produtos que se utilizem do padrão MPEG-1/2 Audio Layer 3 e, portanto, de suas respectivas patentes, é necessário obter uma licença.' "

A licença de uso do *MP3* afirma[17] que, se sua aplicação distribuir arquivos neste formato, deve-se pagar royalties de 2% de toda receita acima de US$100.000,00. Ainda mais preocupante

[14] `http://www.w3cschools.com`. Último acesso, 18/07/2011
[15] MPEG 1 Layer-3
[16] `http://pt.wikipedia.org/wiki/MP3`. Último acesso, 18/07/2011
[17] `http://mp3licensing.com/`. Último acesso, 18/07/2011

é o fato de que, se se distribuir cinco mil cópias de um jogo usando tal tecnologia, deve-se comprar uma licença de US$ 2.500,00. Não é claro se, a rigor da lei, uma aplicação qualquer feita em HTML5, mesmo que utilize um decodificador implementado no browser, deve pagar royalties caso utilize arquivo(s) MP3 e caso receba cinco mil page-views ou mais. Nos padrões atuais, cinco mil page-views é uma quantia irrisória.

Existe um outro padrão de áudio conhecido como *Ogg Vorbis*. Sua especificação é de domínio público[18] e os codecs criados pela Xiph.org são liberados sob uma licença permissiva no estilo BSD. O formato é suportado pelo Mozilla Firefox, Google Chrome e Opera, além de ser adotado pelas principais distribuições do GNU/Linux. Até a presente data, Microsoft e Apple têm se recusado a suportar nativamente tal tecnologia sob a alegação de um "cenário incerto de patentes"[19]. A Wikipedia define o conceito de **patente submarina**[20]:

> "A submarine patent is a patent whose issuance and publication are intentionally delayed by the applicant for a long time, such as several years. This strategy requires a patent system where patent applications are not published. In the United States, patent applications filed before November 2000 were not published and remained secret until they were granted. Analogous to a submarine, therefore, submarine patents could stay 'under water' for long periods until they 'emerged' and surprised the relevant market. Persons or companies making use of submarine patents are sometimes referred to as patent pirates."

Figura 3.7: Mimi & Eunice - Patent. CC-BY-SA 3.0, Nina Paley.

[18] `http://pt.wikipedia.org/wiki/Vorbis#Ogg`. Último acesso, 18/07/2011

[19] `http://www.guardian.co.uk/technology/blog/2010/apr/30/microsoft-ie9-html5-video`. Último acesso, 18/07/2011

[20] `http://en.wikipedia.org/wiki/Submarine_patent`. Último acesso, 18/07/2011

Em outras palavras, *Ogg Vorbis*, apesar de ter sua especificação em domínio público, está supostamente vulnerável a patentes desconhecidas. Curiosamente, o formato *MP3* está tão vulnerável quanto[21]. Vale ressaltar que o suporte à tecnologia *Ogg Vorbis* já foi parte da especificação do HTML5, porém foi posteriormente retirado[22].

Além dos dois formatos supracitados, são opções ainda: o *WAV*, inadequado para a web devido a seu grande tamanho, e o *AAC*, formato tecnicamente superior ao *MP3*[23], porém igualmente proprietário.

Recentemente a indústria viu o aparecimento de um novo formato de áudio e vídeo chamado *WebM*. Patrocinado pela Google, funciona em versões recentes dos navegadores: Mozilla Firefox, Opera e Google Chrome. Internet Explorer e Safari aceitam tal formato somente se um programa de terceiros for instalado[24]. *WebM*, assim como *Ogg Vorbis*, é supostamente um formato livre, muito embora profissionais da indústria expressem dúvidas quanto a essa condição[25].

Ainda em relação à guerra de patentes, o jornal O Estado de S. Paulo publicou(7):

> "O Google perdeu a chance de se proteger contra processos diretos ou indiretos. Um leilão, há duas semanas, de mais de 6 mil patentes da falida empresa de telecomunicações Nortel era a esperança contra novas ameaças. Não deu certo. Apesar de ser favorito, o Google não cobriu o último lance de um grupo de empresas liderado justamente pela Apple [...]
>
> Em um blog do Google, quando anunciou a participação no leião, o vice-presidente sênior Kent Walker desabafou: **'O mundo da tecnologia tem visto uma explosão de processos de violação de patentes. Algumas dessas ações têm sido criadas por pessoas ou empresas que nunca criaram nada; outras são motivadas pelo desejo de bloquear produtos competitivos ou para lucrar com o sucesso de um rival.'** "

[21] `http://en.wikipedia.org/wiki/Use_of_Ogg_formats_in_HTML5`. Último acesso, 18/07/2011

[22] `http://en.wikipedia.org/wiki/Use_of_Ogg_formats_in_HTML5#Recommendation_retracted`. Último acesso, 18/07/2011.

[23] `http://pt.wikipedia.org/wiki/Advanced_Audio_Coding`. Último acesso, 18/07/2011.

[24] `http://en.wikipedia.org/wiki/WebM`. Último acesso, 18/07/2011

[25] `http://blogs.msdn.com/b/ie/archive/2011/02/02/html5-and-web-video-questions-for-the-industry-i aspx`. Último acesso, 18/07/2011

Isto posto, nós, desenvolvedores, estamos no meio de um fogo cruzado entre grandes empresas. No segundo semestre de 2011, a Internet viu o aparecimento da campanha **We Want Ogg**[26], favorável à adoção do formato Ogg Vorbis nos principais navegadores de Internet:

Figura 3.8: Fonte: http://www.wewantogg.com

Além do HTML5 Audio, outras opções a se considerar incluem a utilização de um fallback em Flash ou Silverlight. Testamos a biblioteca SoundManager2[27], que provê funções JavaScript para sons. Para isso, pode-se chavear entre a utilização da tecnologia Flash ou do HTML5 Audio, sendo que este último implica nos problemas supracitados. Embora o fallback em Flash funcione corretamente, suporta apenas os formatos MP3 e MP4/AAC, além de implicar num *delay* considerável entre o momento em que se pede para executar uma demonstração de áudio e o instante em que o som é realmente tocado. Consideramos que esta limitação torna a biblioteca inviável para jogos.

Mozilla implementa ainda uma API própria[28] para a geração de sons em tempo de execução através de chamadas JavaScript. Tal tecnologia poderia ser usada para escrever *decoders*

[26] `http://www.wewantogg.com`. Último acesso, 17/09/2011
[27] `http://www.schillmania.com/projects/soundmanager2/`. Último acesso, 23/07/2011
[28] `https://wiki.mozilla.org/Audio_Data_API`. Último acesso, 23/07/2011

de áudio, o que teoricamente solucionaria os problemas supracitados. Entretanto, além de esta abordagem ser muito complexa para se implementar no tempo reduzido do trabalho de conclusão de curso, não funciona em todos os navegadores.

Enfim, durante a escrita deste texto, **Chinchilla** não tem uma boa solução de áudio. Optamos por continuar utilizando o HTML5 Audio, na expectativa de que os principais browsers de Internet melhorem suas implementações a curto prazo.

Atualização de Novembro de 2011: o suporte a áudio na versão mais recente do Opera parece ótimo. Além disso, a novíssima versão do Google Chrome traz consigo uma nova API chamada Web Audio que, posta à prova, apresentou excelentes resultados (ao contrário do HTML5 Audio tradicional). Conclusão: nesses dois navegadores, o áudio está funcionando bem! Vamos esperar que os outros browsers também melhorem.

Teclado

A biblioteca jQuery[29] fornece mecanismos para se capturar eventos de teclado numa página web. Podemos então associar tais fenômenos de teclado à memória de dados da VM.

Diferentemente do subsistema de vídeo, no de teclado não é necessário trabalhar com interrupções de software. A cada evento disparado pelo teclado, a VM automaticamente muda o valor de uma célula específica da memória. Assim, se por exemplo a tecla A for pressionada (ou solta), uma posição específica da memória recebe tal notificação. O mesmo vale para outras teclas.

Tal abordagem torna muito simples a implementação de funções de alto nível na linguagem **Chinchilla**. Para que o programador tenha acesso ao estado do teclado, chama funções que verificam o conteúdo das tais células na memória.

Mouse

Análogo ao teclado.

3.2.7 Conclusão

A máquina virtual implementada nesta ocasião é um exemplo de projeto interessante que pode ser criado utilizando HTML5. Esperamos que esta discussão tenha deixado clara a importância, para a realização deste trabalho, das disciplinas: Organização de Computadores, Siste-

[29] `http://jquery.com/`. Último acesso, 24/07/2011

mas Operacionais, Álgebra Booleana / Lógica Digital, Laboratório de Programação e Direito e Software livre. Para o trabalho de implementação, também foram relevantes: Programação Funcional e Programação Orientada a Objetos. Computação Gráfica teve um papel menor.

3.3 Compilador

3.3.1 Introdução

De acordo com (2, p. 1),

> "Linguagens de programação são notações para se descrever computações para pessoas e para máquinas. O mundo conforme o conhecemos depende de linguagens de programação, pois todo o software executando em todos os computadores foi escrito em alguma linguagem de programação. Mas, antes que possa rodar, um programa primeiro precisa ser traduzido para um formato que lhe permita ser executado por um computador.
>
> Os sistemas de software que fazem essa tradução são denominados *compiladores*."

Figura 3.9: Um compilador

Usuários da plataforma de jogos aqui desenvolvida criam seus jogos através de uma linguagem textual de alto nível cujo codinome é **Chinchilla**. Com sintaxe inspirada no BASIC[30], possui tipagem fraca e dinâmica, tendo sido pensada para ser de fácil aprendizado e adequada para a criação de protótipos rápidos de games.

Entre as motivações para a existência da linguagem **Chinchilla**, bem como para as linguagens de programação de alto nível de um modo geral, está o fato de se permitir que seus usuários, os programadores, possam expressar seus raciocínios de forma mais inteligível do que fariam caso utilizassem linguagem de máquina. **Chinchilla** é o que se conhece como **linguagem imperativa**: classe de "linguagens em que um programa especifica *como* uma computação deve ser feita [...] Linguagens como C, C++, C# e Java são linguagens imperativas. Nas linguagens imperativas, existe a noção de estado do programa e mudança do estado provocadas pela execução de instruções." (2, p. 9)

[30] Beginner's All-purpose Symbolic Instruction Code, linguagem de programação criada na década de 1960 para fins didáticos.

Apesar da facilidade de uso do BASIC, tal linguagem não está livre de críticas. De acordo com a Wikipedia[31],

> "Muitos anos após seu lançamento, profissionais respeitados da computação, especialmente Edsger W. Dijkstra, expressaram a opinião que o uso da expressão GOTO, que existia em várias linguagens além de BASIC, promovia práticas não desejáveis de programação. [...]
>
> Um dos principais problemas com as versões originais de BASIC era a falta de uma estrutura re-entrante de chamada de sub-rotinas ou funções [...] Isso é uma propriedade similar à dos Fortran originais e um grande entrave à modularização de programas."

Isto posto, diferentemente de sua linguagem mãe, a linguagem de programação desenvolvida neste projeto de formatura não admite o comando GOTO. Ademais, ela suporta funções que podem, inclusive, conter recursão.

Vejamos um exemplo prático. O programa a seguir, escrito em **Chinchilla**, perguntará a idade do usuário e fornecerá uma resposta adequada[32]:

```
idade = input("Qual sua idade?")
if idade >= 18 then
    print "Adulto"
else
    print "Criança"
endif
```

[31] `http://pt.wikipedia.org/wiki/BASIC`. Último acesso, 25/07/2011

[32] legalmente falando...

Executemo-lo:

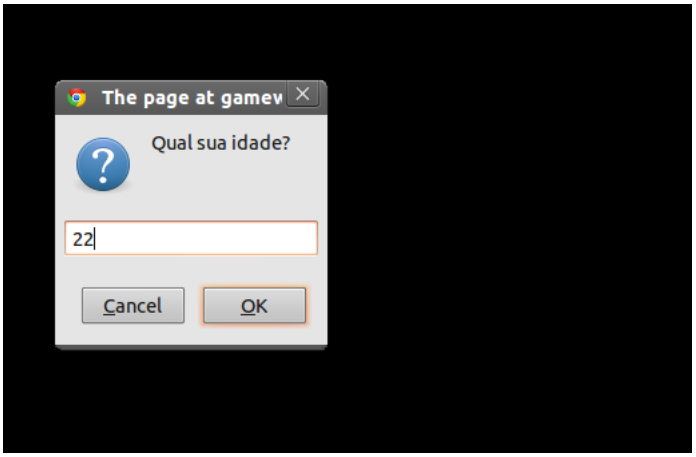


Figura 3.10: Programa em execução

Figura 3.11: Fim do programa

Quando o compilador recebe o código-fonte acima, faz a tradução para o seguinte código em **Chinchilla Assembly**:

```
push "Qual sua idade?"
call _input
pop a
mov a, fun
mov adr, 4096
store a, adr
mov adr, 4096
load a, adr
```

```
mov b, 18
scmp a, b
mov a, true
jge L4
mov a, false
L4:
lcmp a, false
je L6
push "Adulto"
call _print
pop a
jmp L5
L6:
push "Criança"
call _print
pop a
L5:
```

Numa fase seguinte, entrará em cena o **Chinchilla Assembler** (*montador*), que traduzirá este último código para uma **linguagem de máquina** passível de ser interpretada pela **Máquina Virtual**. A tradução do assembly para o código binário não apresenta grande complexidade, e, portanto, não discutiremos esta etapa do processo de compilação.

3.3.2 A estrutura de um compilador

No mapeamento do programa fonte para o programa objeto, consideramos que existem duas fases: análise e síntese.

Na fase de análise, divide-se o programa fonte em partes constituintes e impõe-se uma estrutura gramatical sobre elas. Tal estrutura é usada para criar uma representação intermediária do programa. Ainda nesta fase, coleta-se informações sobre o código-fonte para posteriormente armazená-las no que se conhece como *tabela de símbolos*. Na terminologia da área, considera-se que a fase de análise é o *front-end* do compilador.

Na fase de síntese, usa-se, em conjunto com a *tabela de símbolos*, a representação intermediária criada na etapa anterior para se criar o programa objeto. Esta parte do compilador é o que se conhece como *back-end*.

Alguns compiladores(2, p. 3) incluem uma fase de otimização de código entre o *front-end* e o *back-end*.

A figura abaixo[33] ilustra as etapas de um compilador:

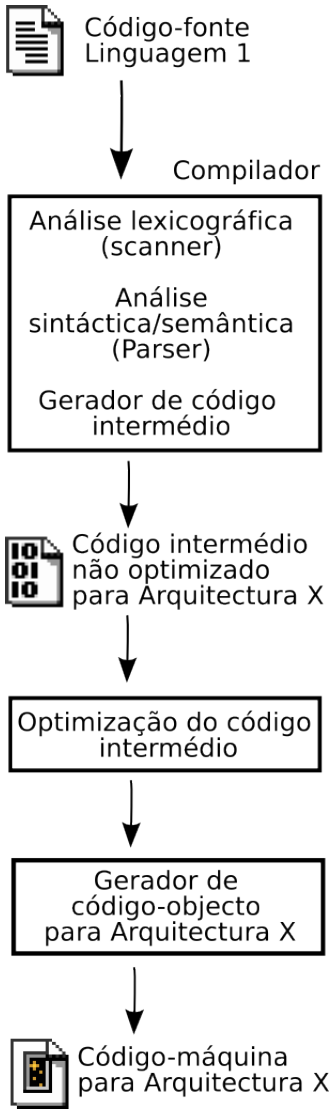


Figura 3.12: Fonte: Wikipedia

[33] Autor: Nuno Tavares; Fonte: `http://pt.wikipedia.org/wiki/Ficheiro:Nt-compilador.png`. Último acesso, 25/07/2011

3.3.3 Análise léxica

A primeira fase de um compilador é conhecida como *análise léxica*. O *analisador léxico*, ou simplesmente *lexer*, lê os caracteres que constituem o código-fonte do programa fornecido como entrada, para então agrupá-los em "sequências significativas" denominadas *lexemas*. Cada lexema é processado pelo *lexer*, que gera um *token* - uma dupla - no formato:

(nome-token, valor-token)

A sequência de tokens do programa fonte será passada à fase seguinte, a de *análise sintática*.

O primeiro componente de um token, seu nome, é um campo abstrato que indica o que, de um modo geral, ele representa (por exemplo: um número). Tal símbolo abstrato é importante para a fase de *análise sintática*. Por outro lado, o segundo componente, seu "valor", apresenta informações complementares sobre o token (exemplo: "7"). Tal campo é relevante para as fases de *análise semântica* e de *geração de código*.

Vejamos um exemplo prático. Suponha que um dado jogo contenha o seguinte comando:

vidas = vidas + 1

O *analisador léxico* lerá este fluxo de caracteres e fornecerá como saída a seguinte sequência de tokens:

(**IDENTIFIER**, **vidas**), (**ASSIGNOP**, **=**), (**IDENTIFIER**, **vidas**), (**BINARYOP**, **+**), (**NUMBER**, **1**), (**NEWLINE**, **$0A**)

Observe que os espaços que separam os lexemas são ignorados pelo *lexer*.

O *analisador léxico* do compilador **Chinchilla** foi feito artesanalmente, de forma *ad-hoc*.

3.3.4 Análise sintática

Base teórica

Segundo (2, p. 122),

> "Em nosso modelo de compilador, o analisador sintático recebe do analisador léxico uma cadeia de tokens repreentando o programa fonte [...] e verifica se essa cadeia de tokens pertence à linguagem gerada pela gramática."

O elemento central de uma linguagem de programação é sua gramática. Antes de continuar com os exemplos práticos, vejamos um pouco de teoria:

Def.: Uma gramática livre de contexto, ou simplesmente gramática, consiste em:

- **Terminais**: conjunto não vazio de símbolos elementares que, juntos, formam cadeias. Dizer "terminal" é o mesmo que dizer "nome do token", ou simplesmente "token" quando não houver risco de confusão;
- **Não-terminais**: representam conjuntos de cadeias. Auxiliam na definição da linguagem[34] da gramática, impondo sobre ela uma hierarquia. Reconhecer tal hierarquia é a chave da análise sintática;
- **Símbolo inicial**: um não-terminal escolhido cuidadosamente. Ele representa o conjunto de cadeias que podem ser geradas pela gramática;
- **Produções**: definem como terminais e não-terminais podem ser combinados de forma a gerar cadeias. Cada produção é da forma *cabeça* → *corpo*, onde:
  - *cabeça* é um não-terminal[35];
  - *corpo* é uma sequência de zero ou mais terminais e não-terminais.

Vejamos outro exemplo prático. Na gramática a seguir, são terminais:

number + - * / ( )

Os não-terminais são:

expr term factor

E o símbolo inicial é *expr*. Com as produções abaixo, a gramática se torna capaz de representar expressões aritméticas simples[36]:

expr → *term* **+** *expr* | *term* **-** *expr* | *term*

term → *factor* ***** *term* | *factor* **/** *term* | *factor*

factor → **(** *expr* **)** | **number**

[34] Conjunto de todas as palavras que podem ser formadas pela gramática através da aplicação de suas produções

[35] Como curiosidade, note que o lado esquerdo de cada produção admite apenas um não-terminal isolado. Assim, eles estão "livres de contexto". ;-)

[36] A notação X → A | B é uma abreviação para as produções X → A e X → B.

Neste exemplo, a produção *expr* $\rightarrow$ *term* **+** *expr* nos diz que, se *expr* representa uma expressão, então *term* **+** *expr* também representa uma expressão. Denotamos uma substituição de *expr* por *term* **+** *expr* por:

$$expr \Rightarrow term + expr$$

Tal substituição é lida como "*expr* **deriva** *term* **+** *expr* em um passo". A aplicação sucessiva de produções gera uma **derivação**. Para mostrar que, por exemplo, a cadeia *1337 + 42 * 7* faz parte da **linguagem livre de contexto** gerada pela gramática acima, podemos mostrar uma **derivação**[37] dela a partir do **símbolo inicial**:

$$expr \Rightarrow$$

$$term + expr \Rightarrow$$

$$factor + expr \Rightarrow$$

$$\textbf{number} + expr \Rightarrow$$

$$\textbf{number} + term \Rightarrow$$

$$\textbf{number} + factor * term \Rightarrow$$

$$\textbf{number + number} * term \Rightarrow$$

$$\textbf{number + number} * factor \Rightarrow$$

$$\textbf{number + number * number}$$

Observe que 1337, 42 e 7 são valores de tokens, obtidos pelo analisador léxico, cujos nomes são **number**. Para o analisador sintático, todos eles são terminais do tipo **number**.

Para denotar que *expr* deriva **number + number * number** em algum número de passos, escrevemos *expr* $\Rightarrow^*$ **number + number * number**.

Em teoria de compiladores, existem ainda os conceitos de árvore de derivação e ambiguidade, porém não os detalharemos aqui. O leitor interessado pode consultar (2, p. 128-129). Essencialmente, uma árvore de derivação é uma forma gráfica de se representar derivações e uma gramática é ambígua quando uma dada sentença da linguagem puder ser gerada por derivações distintas. Para o projetista de linguagens de programação, é recomendável evitar ambiguidades.

[37] Fizemos aqui o que se chama "derivação mais à esquerda": aplicamos sempre uma regra da gramática ao não-terminal mais à esquerda da forma sentencial sendo analisada.

Em **Chinchilla**, foi implementado à mão um analisador sintático preditivo, que é um "analisador de descida recursiva que não precisa de retrocesso". Para saber o que é isso, precisaremos de mais teoria.

Def.: diremos que uma gramática possui *recursão à esquerda* quando tiver um não-terminal A tal que exista uma derivação $A \Rightarrow^* A\alpha$ para alguma cadeia α de símbolos da gramática. A gramática possui uma *recursão à esquerda imediata* se tiver uma produção da forma $A \rightarrow A\alpha$.

Veremos que recursões à esquerda são indesejáveis para nosso *parser*[38]. No caso de recursões à esquerda imediatas, sem alterar as cadeias que podem ser derivadas a partir de A, podemos substituir o par de produções $A \rightarrow A\alpha \mid \beta$ por:

$$A \rightarrow \beta A'$$

$$A' \rightarrow \alpha A' \mid \varepsilon$$

onde A' é um novo não-terminal e ε, a palavra vazia. Para eliminar recursões à esquerda de um modo geral, o leitor interessado pode adotar a estratégia proposta em (2, p. 135-136).

Def.: FIRST(α), sendo α qualquer cadeia de símbolos da gramática, é o conjunto de símbolos terminais que iniciam as cadeias derivadas de α. Por exemplo, na gramática de expressões aritméticas simples fornecida acima, temos que FIRST(*expr*) = { **number**, (}.

Def.: FOLLOW(A), sendo A um não-terminal, é o conjunto de terminais *a* tais que exista uma derivação da forma $S \Rightarrow \alpha A a \beta$, sendo α e β cadeias de símbolos da gramática. Em outras palavras, FOLLOW(A) "é o conjunto de terminais que podem aparecer imediatamente à direita de A em alguma forma sentencial"(2, p.141). Na gramática acima, FOLLOW(*expr*) = {) }.

O cálculo de FIRST e FOLLOW pode ser realizado conforme o algoritmo descrito em (2, p.141-142).

Entre os métodos para se realizar a análise sintática de linguagens de programação reais estão os *descendentes*. Neles, analisamos uma cadeia de terminais "de cima para baixo", ou seja, a partir do símbolo inicial da gramática - raiz da árvore de derivação - até os terminais - folhas da árvore. É possível construir um analisador descendente eficiente (isto é, que consuma tempo linear no tamanho do código fonte fornecido pelo usuário) de forma não muito difícil à mão. Contudo, este método reconhece apenas uma classe restrita de gramáticas livres de contexto conhecida como **LL(1)**.

[38] analisador sintático

Def.: uma gramática livre de contexto é **LL(1)** se, e somente se, sempre que A $\rightarrow \alpha \mid \beta$ forem duas produções distintas, valerem as seguintes condições:

- Tanto α como β não derivam cadeias começando com a, qualquer que seja o terminal a;
- No máximo um dos dois, α ou β, pode derivar ε;
- Se a palavra vazia estiver em FIRST(β), então FIRST(α) e FOLLOW(A) serão disjuntos. Analogamente, se $\alpha \Rightarrow^* \varepsilon$, então β não deriva qualquer cadeia que comece com um terminal pertencente a FOLLOW(A).

Os dois primeiros itens equivalem a dizer que os conjuntos FIRST(α) e FIRST(β) devem ser disjustos. Além disso, conforme (2, p.141),

> "A classe de gramáticas LL(1) é rica o suficiente para reconhecer a maioria das construções presentes nas linguagens de programação, mas escrever uma gramática adequada para a linguagem fonte não é uma tarefa simples. Por exemplo, nenhuma gramática com recursão à esquerda ou ambígua pode ser LL(1)."

Os analisadores sintáticos de descida recursiva que não precisam de retrocesso, também conhecidos como analisadores sintáticos preditivos, podem reconhecer a classe LL(1). O primeiro "L" de LL(1) significa que os caracteres são lidos da entrada a partir da esquerda para a direita (*left-to-right*). O segundo, mostra que são geradas derivações mais à esquerda (*leftmost*). Finalmente, o "1" diz que se usa, da entrada, um token à frente em cada passo. Tal token é conhecido como **símbolo lookahead** e é usado para tomar decisões durante o processo de análise.

Exemplo prático: com alguns ajustes, podemos tornar LL(1) a gramática para expressões aritméticas simples abordada anteriormente. Adicionemos os não terminais *expr'* e *term'* e reescrevamos o conjunto de produções para:

$$expr \rightarrow term\ expr'$$

$$expr' \rightarrow \mathbf{+}\ term\ expr' \mid \mathbf{-}\ term\ expr' \mid \varepsilon$$

$$term \rightarrow factor\ term'$$

$$term' \rightarrow \mathbf{*}\ factor\ term' \mid \mathbf{/}\ factor\ term' \mid \varepsilon$$

$$factor \rightarrow \mathbf{(}\ expr\ \mathbf{)} \mid \mathbf{number}$$

Implementação prática

Do ponto de vista de implementação, um analisador sintático descendente recursivo, que é o implementado no compilador **Chinchilla**, "é um analisador sintático descendente construído a partir de subrotinas mutualmente recursivas (ou qualquer equivalência não recursiva como uma pilha) em que cada subrotina geralmente implementa uma das regras de produção da gramática. Cada subrotina fica associada a um elemento não-terminal da gramática. Consequentemente, a estrutura do programa resultante se assemelha bastante à gramática que ele reconhece."[39]

Isto posto, é fácil ver que uma gramática com recursão à esquerda é inadmissível para o método descendente recursivo de análise, já que o compilador entraria num *loop infinito*.

O analisador sintático é responsável por gerar uma forma intermediária de código conhecida como **árvore sintática abstrata**, ou simplesmente **árvore sintática**. Ela é uma estrutura de dados que representa, de forma hierárquica, o código-fonte do jogo do usuário. Para uma dada expressão, cada nó interior representa um operador da linguagem. Os filhos desses nós são seus operandos. Diferentemente das árvores de derivação, que têm seus nós intermediários representando não-terminais, as sintáticas têm seus nós interiores representando construções de programação. Por exemplo, na gramática mostrada na seção anterior, *expr* seria um nó intermediário de uma árvore de derivação, mas não de uma de sintaxe.

Como exemplo concreto, considere o seguinte trecho de código:

```
' Algoritmo de Euclides
while b <> 0
    if a > b then
        a = a - b
    else
        b = b - a
    endif
wend
return a
```

A figura[40] abaixo mostra a árvore sintática correspondente ao algoritmo acima:

[39] `http://pt.wikipedia.org/wiki/Analisador_sint\%C3\%A1tico_descendente_recursivo`. Último acesso, 27/07/2011.

[40] Autor: Dcoetzee. Fonte: `http://en.wikipedia.org/wiki/File:Abstract_syntax_tree_for_Euclidean_algorithm.svg`. Último acesso, 27/07/2011

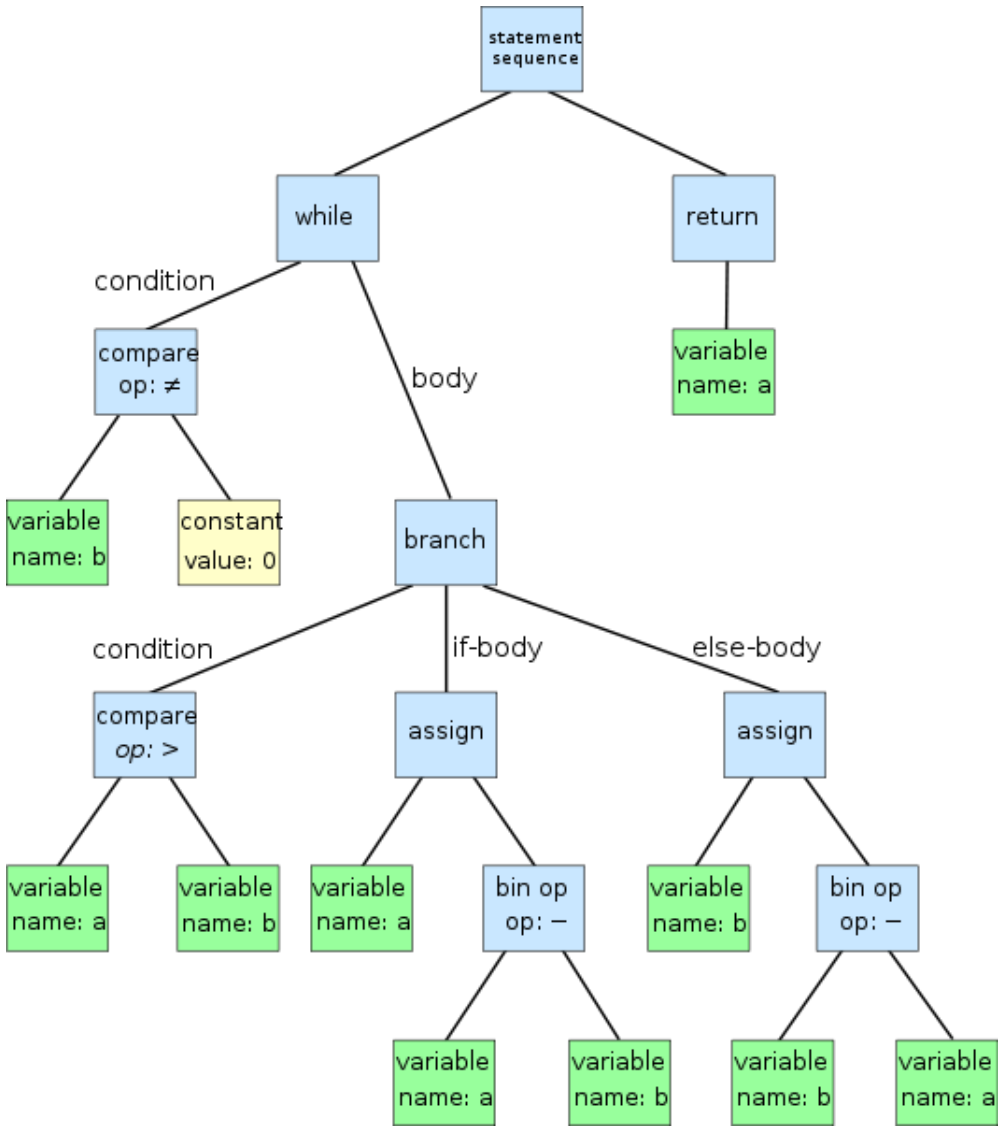


Figura 3.13: Fonte: Wikipedia

Tal árvore será usada na fase de geração de código. Quanto à implementação prática dessas estruturas, são relevantes os *padrões de projeto* **Interpreter**[41] e **Composite**[42], que não entraremos em detalhes neste texto.

3.3.5 Tabelas de símbolos

Conforme (2, p.55),

> "*Tabelas de símbolos* são estruturas de dados usadas pelos compiladores para conter informações sobre as construções do programa fonte. As informações são coletadas

[41] `http://en.wikipedia.org/wiki/Interpreter_pattern`. Último acesso, 27/07/2011.
[42] `http://pt.wikipedia.org/wiki/Composite`. Último acesso, 27/07/2011.

> de modo incremental pelas fases de análise de um compilador e usadas pelas fases de síntese para gerar o código objeto. As entradas na tabela de símbolos contêm informações sobre um identificador, como seu nome ou lexema, [...] seu endereço na memória e qualquer outra informação relevante. As tabelas de símbolos normalmente precisam dar suporte a múltiplas declarações do mesmo identificador dentro de um programa."

Na linguagem de programação desenvolvida neste trabalho, não há a necessidade de se declarar variáveis. Funções não são objetos de primeira classe[43]. As entradas nas tabelas de símbolos são usadas na etapa de geração de código e, além disso, são criadas e utilizadas durante as fases de análise léxica, sintática e semântica. Quais fases da compilação desempenham exatamente quais tarefas, nas tabelas, varia caso a caso.

As tabelas de símbolos de **Chinchilla** armazenam:

- **Funções**: nome, endereço e aridade;
- **Variáveis em memória estática**: nome e endereço real, determinado em tempo de compilação;
- **Variáveis de pilha**: nome e localização, dada por um deslocamento em relação à base da pilha de execução. Note que o endereço real não é conhecido a priori.

Chinchilla não admite aninhamento de funções, ou seja, funções podem ser definidas apenas no escopo[44] global. Por outro lado, **Chinchilla** suporta funções reentrantes. Variáveis em memória estática incluem as globais, que podem ser acessadas a partir de qualquer escopo do programa fonte. Entre as variáveis de pilha estão aquelas locais a uma dada função.

A implementação de escopo é feita mantendo-se um apontador, na tabela de símbolos do escopo que se está processando, para uma tabela "pai". Assim, comandos no escopo global, aquele cuja tabela é "pai" de todas as outras, não podem acessar variáveis locais a funções. Inversamente, comandos dentro de funções conhecem seus identificadores locais e podem também, caso precisem, acessar variáveis globais ou chamar outras funções. Contudo, não podem acessar variáveis locais de outras funções.

Vejamos um exemplo na prática. Em **Chinchilla**, variáveis cujos nomes são prefixados com **g_** são globais.

[43] `http://en.wikipedia.org/wiki/First-class_function`. Último acesso, 28/07/2011.

[44] "[...] o escopo de uma declaração é a parte de um programa à qual a declaração se aplica" (2, p.55)

```
fun f()
    if g_pontuacao >= 1000 then
        print "Vencedor"
        exit
    endif
endfun

fun g()
    if pontuacao >= 1000 then
        print "Vencedor"
        exit
    endif
endfun
```

Notamos que a função f fechará o jogo do usuário após imprimir o texto "Vencedor" caso a global *g_pontuacao* seja não inferior a mil. Em contrapartida, g compara a variável local *pontuacao* a mil. Trata-se de um erro de programação, pois esta última variável, alocada na pilha, não foi inicializada e pode conter lixo de memória.

Finalmente, há apenas dois níveis de escopo: global (fora de qualquer bloco **fun ... endfun**) e local a uma função.

3.3.6 Análise semântica

> "Embora a análise sintática consiga verificar se uma expressão obedece às regras de formação de uma dada gramática, seria muito difícil expressar através de gramáticas algumas regras usuais em linguagem de programação [...] e situações onde o contexto em que ocorre a expressão ou o tipo da variável deve ser verificado."[45]

Gramáticas livres de contexto não conseguem descrever integralmente toda a riqueza de certas linguagens de programação. Sintaticamente falando, certas sentenças no código-fonte do jogo do usuário podem estar corretas. Entretanto, é possível que tais expressões não façam sentido, ou seja, estejam *semanticamente* incorretas.

Na prática, como os modelos das gramáticas formais que abordamos são relativamente simples e como são conhecidas formas eficientes de reconhecê-las via *software*, os projetistas

[45] `http://www.dca.fee.unicamp.br/cursos/EA876/apostila/HTML/node71.html`. Último acesso, 28/07/2011

de linguagens de programação tendem a usá-las. Para expressar construções da linguagem que não se encaixam na gramática, fazem uso de "regras informais". Surge assim o analisador semântico.

No caso de linguagens com tipagem estática, como **não** é o caso de **Chinchilla**, o **verificador de tipos** não permitirá que se façam operações entre tipos incompatíveis. Por exemplo, se de acordo com a tabela de símbolos a variável *pontuacao* for do tipo inteiro, a atribuição *pontuacao = "dez mil"*, embora sintaticamente correta, deve fazer o compilador lançar um erro.

Em relação à linguagem **Chinchilla**, existe um **verificador de fluxo de controle** que pode ser ilustrado pelo exemplo:

```
fun f(x)
    if x >= 7 then break
    return x^2
endfun
```

Ao executar tal código, como era de se esperar, recebemos o erro:

Unexpected "break": this command may only be used in loops.

Outras verificações realizadas na fase de análise semântica do compilador implementado neste trabalho incluem, para uma chamada de função:

- Checar, na tabela de símbolos, se a função realmente existe;
- Ter certeza de que o número de parâmetros passados numa chamada de função é igual ao exigido em sua declaração.

É também importante lançar um erro se se encontrar duas ou mais declarações de funções com o mesmo nome.

3.3.7 Geração de código

Introdução

Nas seções anteriores, estudamos a fase de análise do compilador. Agora estudaremos a etapa de síntese, na qual será gerado o programa objeto.

Vimos que, após a fase de análise, temos em mãos uma representação intermediária e hierárquica do código-fonte do jogo do usuário denominada **árvore sintática** (abstrata). Levando em conta as informações relevantes da tabela de símbolos, podemos agora gerar um código objeto que tenha o mesmo significado que o código fonte.

A rigor, estudaremos aqui a geração de código de uma outra linguagem intermediária, textual e linear, denominada **Chinchilla Assembly**. De acordo com o que já vimos no capítulo sobre a máquina virtual, tal código passará por um novo compilador, o **Chinchilla Assembler** (montador), que o traduzirá para um **código binário** passível de ser interpretado pela VM. Por não apresentar grande complexidade, já que o assembly deste projeto nada mais é que um código binário acrescido de mnemônicos "*human-readable*", o montador não será estudado em detalhes. O leitor pode considerá-lo como uma caixa preta.

Teoria

Em teoria de compiladores, a etapa de geração de código apresenta um campo rico de estudos. De acordo com (2, p. 321),

> "Os requisitos impostos sobre o gerador de código são severos. O código objeto precisa preservar o significado semântico do programa fonte e ser de alta qualidade, ou seja, precisa usar efetivamente os recursos disponíveis da máquina destino. Além do mais, o próprio gerador de código precisa ser executado eficientemente.
>
> [...]
>
> O desafio é que, matematicamente, o problema de gerar um código objeto ótimo para determinado programa fonte é indecidível; muitos dos subproblemas encontrados na geração do código, tais como a alocação de registradores, são computacionalmente intratáveis. Na prática, temos de nos contentar com técnicas heurísticas que geram um código bom, mas não necessariamente ótimo. Felizmente, as heurísticas estão suficientemente maduras para que um gerador de código projetado cuidadosamente possa produzir código várias vezes mais rápido que o código produzido por um gerador ingênuo."

Há na literatura(2, p.321-442) diversas técnicas para a geração de códigos eficientes. Entretanto, devido a restrições de tempo, tais técnicas não foram estudadas - e muito menos implementadas - durante este trabalho de conclusão de curso. Pode-se dizer que nosso gerador de código é, atualmente, "o mais simples que funciona". ;-)

Estudo de caso: if inline

Na linguagem **Chinchilla**, há diversas construções de programação como: condicionais, laços, funções, etc. Estudaremos aqui a geração de código de apenas uma construção em particular: o **if inline**.

O **if inline** é uma linha com a seguinte forma:

if *expressão* **then** *comando*

Como era de se esperar, o *comando* será executado apenas caso a *expressão* seja verdadeira. Na **árvore sintática**, tal construção assume a forma de um nó com dois filhos: *expressão* e *comando*. Assim como o próprio nó do **if inline**, cada filho também é capaz de gerar código assembly.

Quando se pede para o nó *expressão* gerar código, ele o faz de tal forma que seu resultado, ao final de seu processamento, torne-se o valor do topo da pilha de execução. Assim, se *expressão* for uma árvore sintática representando a comparação *g_pontuacao* >= *1000*, tal nó gerará código responsável por consultar o valor da variável *g_pontuacao*, compará-la a mil e, finalmente, empilhar verdadeiro ou falso, dependendo do resultado calculado. O nó do **if inline** poderá então desempilhar esse valor e usá-lo em sua própria lógica. O nó *comando* funciona de forma similar, porém não empilha qualquer valor ao final de sua geração de código particular.

Adicionalmente, o gerador de código pode chamar um método denominado *newlabel()*, responsável por gerar um novo *label* (“etiqueta”) de nome único. Tais etiquetas são importantes para se fazer desvios em tempo de execução.

O algoritmo responsável pela geração de código do **if inline** é aproximadamente o seguinte:

```
// nova etiqueta
String label = newlabel();

// obtendo resultado da expressão
emit( expression.code() );
emit( "pop a" ); // o registrador ‘a’ sabe se <expressão> é verdadeira ou falsa

// verificando resultado da expressão
emit( "lcmp a, true" ); // compare o conteúdo do registrador ‘a’ com ‘true’
emit( "jne " + label ); // pule para <label> se <expressão> for falsa
```

```
// se <expressão> for verdadeira, execute <comando>
emit( command.code() );

// fim do if inline
emit( label + ":" );
```

Outras construções como: **if-else**, **while**, **for** e chamadas de função[46] são mais complexas e não serão abordadas aqui.

Otimização de código

Chinchilla não implementa qualquer estratégia avançada de otimização de código. Entretanto, por fazer uso intenso da pilha, notamos que os assemblies gerados pelo compilador geravam muitas sequências da forma:

```
; [...]
push val
pop reg
; [...]
```

Mesmo sem qualquer teoria, observamos que se $val \neq reg$, podemos simplesmente trocar tal par de linhas por uma única instrução *mov reg, val*. Se $val = reg$, removemos ambas do código.

Surpreendentemente, verificamos que com esta simples "otimização" - que pode ser realizada em tempo linear no número de linhas do assembly - um programa que calcula recursivamente o fatorial de um inteiro presenciou uma queda de cerca de 21% no tamanho de seu código binário. Moral da história: ainda há muito o que melhorar. ;-)

3.3.8 Conclusão

A construção de um compilador é um desafio técnico interessante e, integrado a um ambiente colaborativo de desenvolvimento de jogos, serve a um propósito maior. Destacamos

[46] Numa chamada de função, a pilha é modificada conforme sugestão de (8, p.87-90). Os parâmetros são empilhados pelo chamador. Também são adicionados à pilha: o endereço de retorno (ret) da função e um ponteiro para a base anterior da pilha (registrador bp). Aloca-se, além disso, espaço para as variáveis locais.

aqui a importância, para o desenvolvimento desta parte do trabalho de conclusão de curso, das disciplinas: Estruturas de Dados, Linguagens Formais e Autômatos, Programação Orientada a Objetos, Laboratório de Programação e Conceitos Fundamentais de Linguagens de Programação. Infelizmente, por não haver oferecimento, não foi cursada uma disciplina específica sobre compiladores. Mesmo assim, a bibliografia especializada, Aho et al.(2), foi de imensa ajuda.

3.4 Sistema web colaborativo

3.4.1 Introdução

Para os usuários, o sistema web é a "cara" da plataforma.

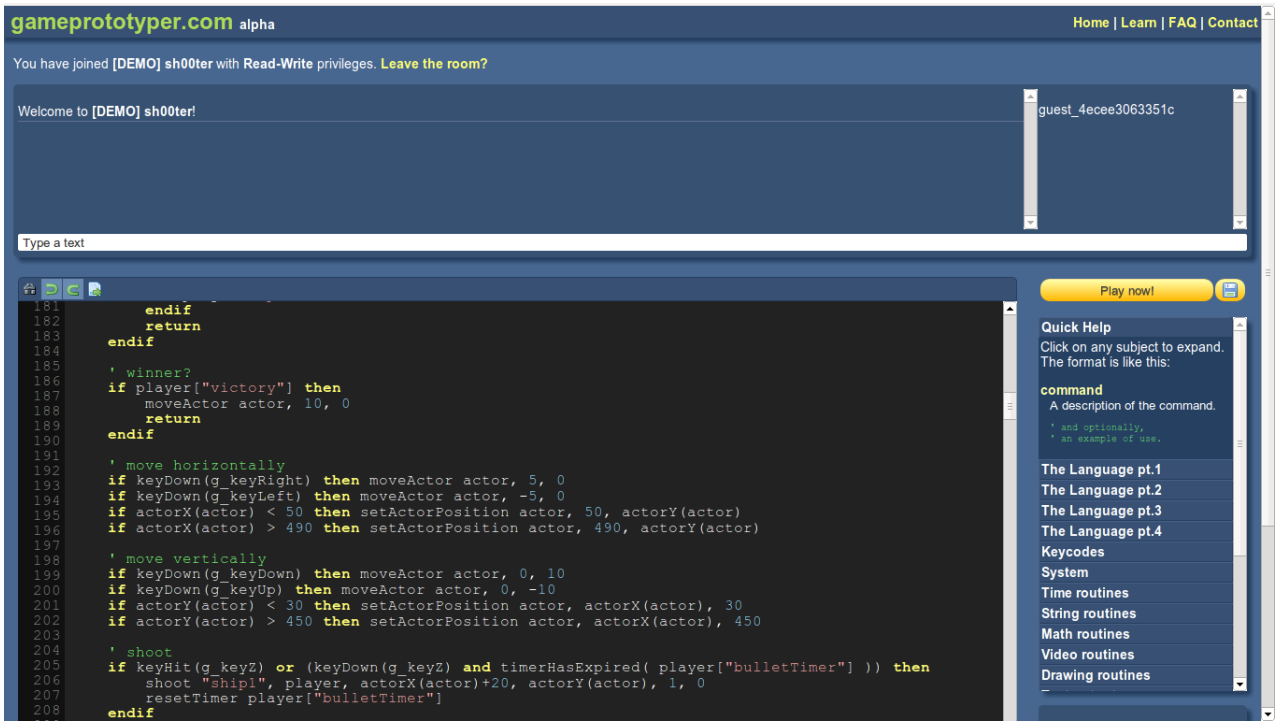


Figura 3.14: A criação de um jogo

Até o momento, discutimos toda a infraestrutura necessária a um ambiente de criação de jogos. No entanto, consideramos que uma importante característica deste trabalho é o fato de a solução desenvolvida residir na nuvem, possibilitando a colaboração entre artistas, compositores, programadores, etc. distribuídos geograficamente.

Segundo a Wikipedia[47], **computação em nuvem** é a entrega de computação como serviço, não como produto. Os dados chegam aos usuários como uma utilidade[48] através de uma rede (tipicamente a Internet). Desta maneira, podemos utilizar recursos computacionais (exemplo: capacidade de armazenamento e/ou de cálculo) fornecidos por máquinas distribuídas e compartilhadas pela rede. Na prática, estamos falando em terceirizar a infraestrutura de TI.

"O armazenamento de dados é feito em serviços que poderão ser acessados de qualquer lugar do mundo, a qualquer hora, não havendo necessidade de instalação de programas x ou

[47] `http://en.wikipedia.org/wiki/Cloud_computing`. Último acesso, 13/11/2011.
[48] como por exemplo a eletricidade que chega em nossas casas.

de armazenar dados. O acesso a programas, serviços e arquivos é remoto, através da Internet - daí a alusão à nuvem. O uso desse modelo (ambiente) é mais viável do que o uso de unidades físicas."[49]

A figura[50] abaixo ilustra o conceito:

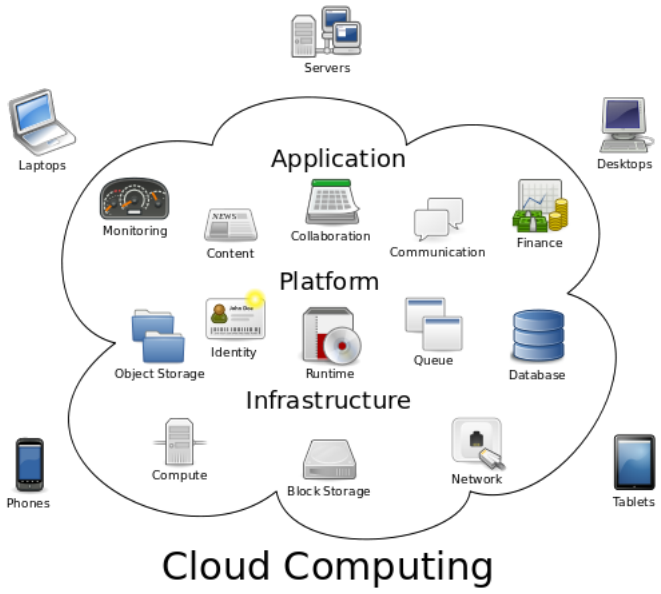


Figura 3.15: Cloud Computing.

Chinchilla reside na nuvem e é colaborativo. De acordo com o site do IME-USP[51]:

> "A Web hoje é colaborativa. Sites que possibilitavam apenas interações monousuário passaram incorporar recursos voltados para colaboração direta e indireta entre os usuários. Esta mudança de enfoque vem sendo chamada de Web 2.0. Sites de comércio eletrônico, por exemplo, passaram a oferecer para cada produto diversas possibilidade de interação entre usuários, como por exemplo, suporte a avaliação, resenha, troca de mensagens, wiki, compartilhamento de fotos, filtragem colaborativa, recomendação etc. Os sistemas web ficam melhores na medida em que mais usuários interagem e contribuem. Surge uma 'inteligência coletiva' a partir da análise das interações entre os usuários.
>
> [...]

[49] `http://pt.wikipedia.org/wiki/Computa\%C3\%A7\%C3\%A3o_em_nuvem`. Último acesso, 13/11/2011.

[50] Autor: SamJohnston. Licença: CC-BY 3.0. Fonte: `http://pt.wikipedia.org/wiki/Ficheiro:Cloud_computing.svg`

[51] `http://www.ime.usp.br/dcc/areas/sistemascolaborativos`. Último acesso, 07/10/2011.

> Sistemas colaborativos são amplamente utilizados por organizações, instituições, grupos, comunidades e na sociedade em geral. Correio eletrônico, mensagem instantânea, bate-papo, videoconferência, blog, compartilhamento de arquivos, editores cooperativos, workflow, sites de relacionamentos e comunidades virtuais são alguns dos sistemas colaborativos que se tornaram populares. Estes sistemas são aplicáveis em diversos setores da sociedade, dando suporte por exemplo ao trabalho em empresas e ao ensino-aprendizagem em escolas. As possibilidades de interação através de sistemas colaborativos aumentam com a popularização das novas mídias de interação, providas pela computação móvel, pela TV digital, pelos mundos virtuais e por dispositivos físicos."

Em suma, um **software colaborativo**, também conhecido como **groupware**, provê um ambiente compartilhado que possibilita o trabalho em grupo. Soluções como: wiki, bate-papo, petições online e redes sociais se enquadram nesta categoria.

Apesar de definirmos nosso sistema como colaborativo, não estamos 100% corretos se levarmos em conta a classificação de Rodden[52]. De acordo com o professor, "aplicações no espaço colaborativo"[53] são classificadas em:

- **Sistema de Mensagens**: usuários trabalham sozinhos e há transferência de dados. Exemplo: e-mail;
- **Sistema de Conferência**: permitem a comunicação em tempo real. Exemplos: bate-papo, Skype;
- **Sistema Coordenado**: pessoas no mesmo espaço físico constróem informação, tendo auxílio do computador. Exemplo: lousa interativa;
- **Sistema Colaborativo**: múltiplos autores, visando a geração de conteúdo, cooperam de forma assíncrona através de uma mesma plataforma. Exemplo: wiki.

A figura[54] a seguir ilustra o conceito:

[52] Tom Rodden, professor de Ciência da Computação da Universidade de Nottingham.

[53] `http://pt.wikipedia.org/wiki/Software_colaborativo`. Último acesso, 13/11/2011.

[54] Autor: Vamoss (Wikipedia). Fonte: `http://pt.wikipedia.org/wiki/Ficheiro:Tom-Rodden-Espa\%C3\%A7o-Colaborativo.jpg` (domínio público). Último acesso, 13/11/2011.

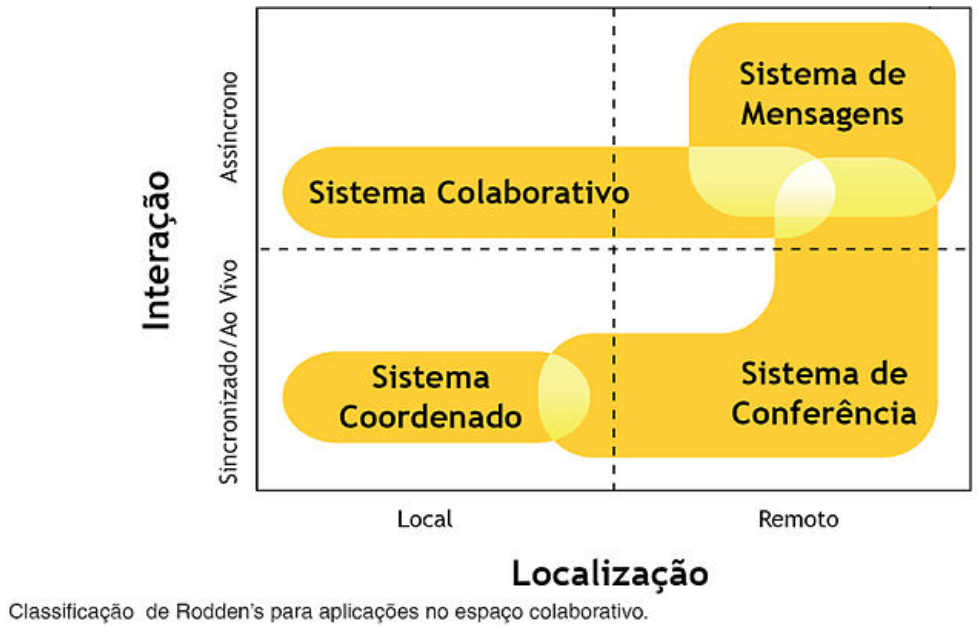


Figura 3.16: Classificação de Rodden.

Neste sentido, **Chinchilla** estaria mais bem encaixado num Sistema de Conferência. Entretanto, acreditamos que tal denominação causa certa confusão com sistemas envolvendo áudio e vídeo e, portanto, decidimos continuar com a designação de Sistema Colaborativo. O leitor pode pensar que as quatro categorias da classificação de Rodden são especializações de Sistemas Colaborativos.

3.4.2 Arcabouço

Para a construção do sistema web utilizamos o CodeIgniter.

CodeIgniter é um arcabouço PHP de código aberto. Ele tem como objetivo facilitar a vida de quem desenvolve sites em PHP, já que fornece funcionalidades frequentemente requisitadas por programadores: acesso a banco de dados, cookies, criptografia, etc. De acordo com o site oficial, o framework é simples, elegante e leve[55]. A documentação também é excelente.

[55] `http://codeigniter.com/`. Último acesso, 24/11/2011.

Figura 3.17: Figura do site oficial do framework

CodeIgniter trabalha com o padrão **Model-View-Controller** (MVC). Neste padrão, visa-se a separação entre lógica de negócio e lógica de apresentação, o que permite que o desenvolvimento e a manutenção de ambos sejam feitos de forma separada. Segundo (9, p. 67),

> "O MVC ou *Model-View-Controller* é um padrão de arquitetura de software que permite a separação entre partes fundamentais de um sistema, com uma arquitetura em três camadas. Ele é um padrão muito usado em fábricas de sofware, pois permite uma separação das tarefas, possibilitando, assim, que um software complexo seja desenvolvido rapidamente e de forma muito objetiva."

A figura[56] a seguir mostra o papel de cada uma das camadas:

[56] `http://en.wikipedia.org/wiki/File:MVC_Diagram_3.jpg`. Autor: BlueSky23 (Wikipedia), CC-BY-SA 3.0. Último acesso, 24/11/2011.

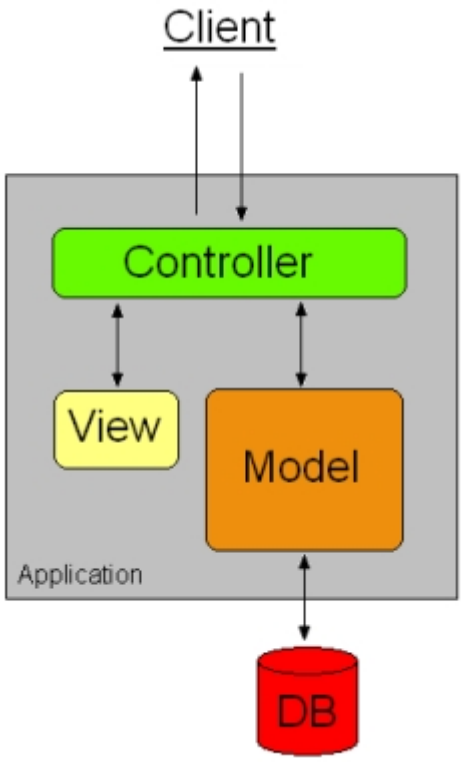


Figura 3.18: Model-View-Controller

Resumidamente,

- **Model**: gerencia os dados e a lógica de domínio. Não é meramente um banco de dados[57];
- **View**: visão do sistema. Apresenta o Model de forma adequada para os usuários (por exemplo: através de uma interface);
- **Controller**: interpreta os comandos do usuário e instrui o Model e a View a reagirem de acordo.

Uma aplicação MVC é, portanto, uma coleção de triplas MVC: cada uma é responsável por alguma parte significativa do sistema.

3.4.3 Autenticação

O sistema permite que os usuários entrem anonimamente para conhecê-lo. Entretanto, para efetivamente usá-lo, é indicada a autenticação via OpenID:

[57] `http://en.wikipedia.org/wiki/Model\%E2\%80\%93view\%E2\%80\%93controller#C.2B.2B`. Último acesso, 24/11/2011.

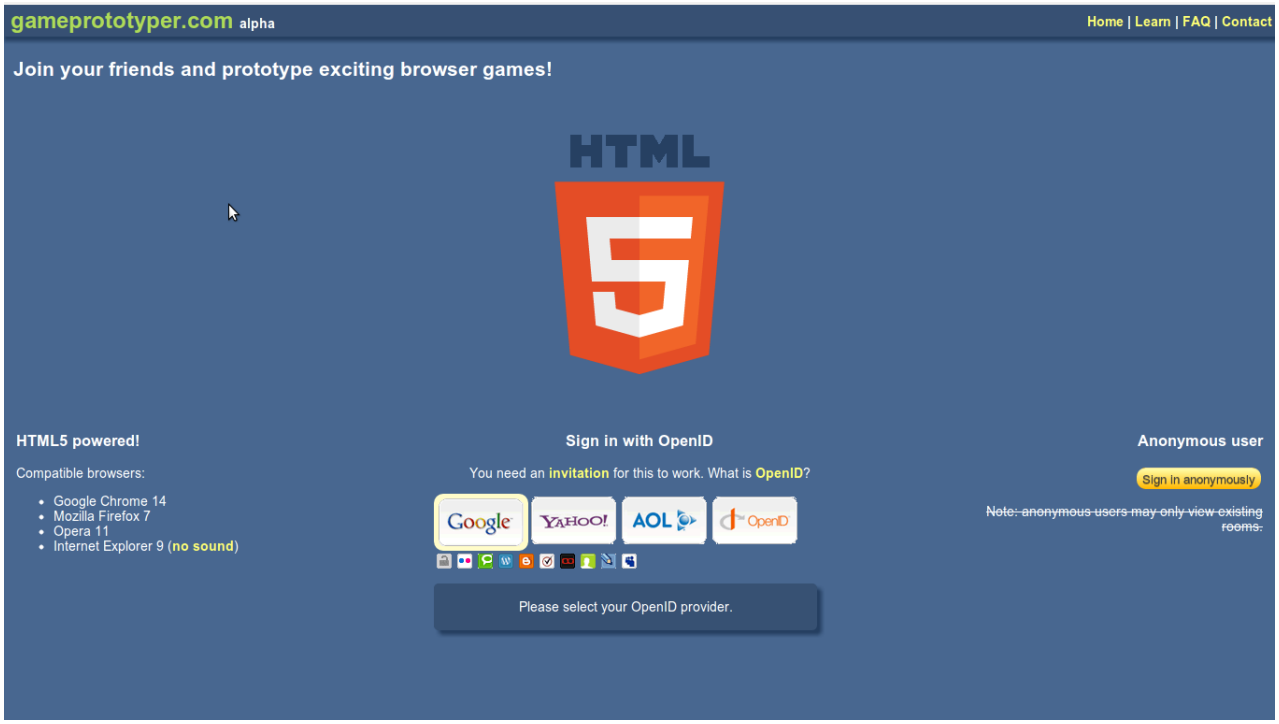


Figura 3.19: Tela de autenticação

OpenID é um padrão aberto de autenticação. Ele descreve como usuários podem, de forma descentralizada, autenticar-se em sistemas, o que evita que desenvolvedores tenham que criar seus próprios mecanismos *ad-hoc* de *login*. Para os usuários, o OpenID pode ser visto como uma identidade digital unificada: não é necessário que eles se registrem em cada novo sistema que encontram. Basta usar o OpenID.

Não existe um banco de dados unificado para identidades OpenID: diversos provedores (Yahoo!, Google, AOL, Wordpress, etc.) implementam o padrão. Usuários que sejam registrados em qualquer um desses serviços já possuem OpenID. Sendo assim, ao se autenticar em um dos provedores, um cliente pode utilizar sistemas de terceiros baseados no padrão. É importante ressaltar que, como a autenticação é feita nos provedores e via HTTPS[58], é garantida a confidencialidade dos dados.

A figura a seguir foi obtida do site OpenID Explained[59].

[58] HyperText Transfer Protocol Secure, uma implementação do HTTP sobre uma camada SSL ou TLS, permite que os dados trafeguem de forma criptografada e com verificação de autenticidade entre cliente e servidor.

[59] `http://openidexplained.com/`. Último acesso, 24/11/2011.

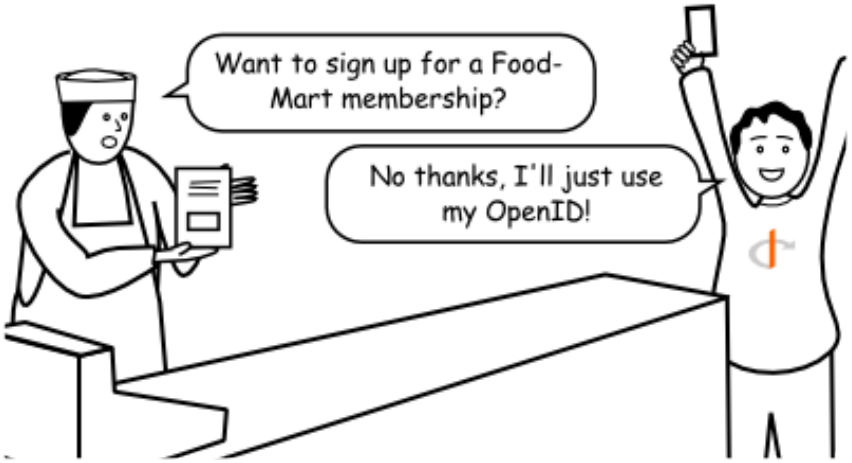


Figura 3.20: OpenID Explained

Para o trabalho de conclusão, foi utilizada a biblioteca livre LightOpenID[60], já que ela abstrai todas as particularidades técnicas do padrão.

3.4.4 Bate-papo

Ao fazer a autenticação, o usuário se depara com uma lista de salas. Cada sala corresponde a um jogo sendo feito individualmente ou em grupo. Pode-se criar novas salas e/ou entrar em projetos existentes. Também é permitido incluir senhas nas salas, caso se queira maior privacidade.

Em cada uma das salas do ambiente de criação de jogos existe um mecanismo de bate-papo.

Figura 3.21: Bate-papo

Inicialmente nossa ideia era integrar algum sistema pronto de bate-papo ao projeto. Porém, os sistemas que encontramos na Internet eram demasiadamente complexos (muito cheios de recursos) e pouco flexíveis. Queríamos algo simples e funcional. Posteriormente, procuramos

[60] `http://gitorious.org/lightopenid`. Último acesso, 24/11/2011.

por “mini-sistemas” prontos em sites sobre PHP. Entretanto, tudo o que encontrávamos eram códigos de baixa qualidade e infestados de vulnerabilidades. Decidimos então criar um bate-papo próprio.

Do ponto de vista de banco de dados, a modelagem do chat é muito simples. Há três entidades: sala, usuário e mensagem. Existe um relacionamento 1:N entre sala e mensagem, um 1:N entre usuário e mensagem e um M:N entre sala e usuário. Criamos chaves estrangeiras, na tabela de mensagens, para indicar quem as enviou e a qual sala elas pertencem. Ademais, existe uma relação adicional para implementar o mapeamento M:N, já que um usuário pode estar em várias salas e uma sala pode ter vários usuários.

O chat implementa alguns serviços[61]:

- **Entrar na sala**
- **Sair da sala**
- **Pedir estado do chat** (listas de mensagens e de usuários participantes)
- **Adicionar mensagem**

Cada um dos serviços é ativado assim que o cliente envia à URL correspondente uma requisição HTTP GET ou POST (varia caso a caso). Tais requisições contêm todos os dados necessários para o processamento da operação[62].

Vejamos, por exemplo, como funciona o mecanismo de pedir o estado do chat:

1. Cliente faz uma requisição assíncrona[63] para http://<endereço>/chat/receive/<id_da_sala>;
2. CodeIgniter dispara o método **receive** do Controller **chat**. O **id da sala** também é capturado;
3. A requisição HTTP traz, em seu cabeçalho, cookies do cliente. Através do Model, cruzamos essas informações com o banco de dados do sistema e verificamos se o usuário está devidamente autenticado;
4. Se tudo correr bem, o Controller dispara uma chamada ao Model, pedindo a lista de usuários e de mensagens da sala[64];

[61] de acordo com a literatura, criamos o que se conhece como Web Services.

[62] dizemos que a operação é *stateless*.

[63] é o que se conhece como AJAX: Asynchronous Javascript and XML. Apesar do que o nome indica, o uso da linguagem de marcação XML não é obrigatório.

[64] para diminuir o tamanho da resposta, são devolvidas apenas as mensagens enviadas a partir de um certo momento.

5. Através de uma query SQL devidamente validada, o Model consulta o banco de dados e devolve ao Controller os dados requisitados;

6. O Controller, agora com os dados em mãos, repassa-os à View;

7. A View, responsável pela apresentação, formata os dados utilizando a notação JSON[65];

8. O JSON é devolvido ao cliente. O papel do servidor acaba aqui;

9. No cliente, fazemos o *parsing* da resposta do servidor;

10. Através de chamadas ao jQuery[66], atualizamos a tela do usuário e terminamos a operação.

O mecanismo de atualização das mensagens é feito através de polling. Periodicamente, os clientes perguntam ao servidor se novos eventos ocorreram e todo o mecanismo descrito acima se repete.

O leitor pode perceber que, infelizmente, usar polling não é uma estratégia muito eficiente: além de causar certo atraso na percepção dos eventos por parte dos usuários, seu uso frequente implica em tráfego intenso na rede. Pior: boa parte dos dados é puro overhead. O estabelecimento frequente de conexões TCP[67] (vindo juntamente como todo seu mecanismo de handshake) e o envio de cabeçalhos HTTP grandes pouco contribuem com o sistema de chat. Infelizmente, na presente data, há pouco o que possamos fazer.

O HTML5 introduz a tecnologia dos WebSockets[68], que permite criar uma conexão full-duplex[69] entre cliente e servidor. Infelizmente, este recurso ainda (2011) está muito imaturo para ser colocado em prática: novas versões da especificação "*pipocam*" a cada dia e o suporte entre os browsers é inconsistente. Além disso, ainda há preocupações por parte dos fabricantes dos navegadores em relação à segurança dos WebSockets[70]. Desta forma, alguns dos browsers recentes vêm com o suporte à tecnologia desativado por padrão.

O uso dos WebSockets tornaria, no contexto das aplicações web, os mecanismos de alta interatividade muito mais: simples, eficientes, robustos e escaláveis. Entretanto, tal tecnologia ainda é inviável. Portanto, por ora, continuamos a fazer polling.

[65] JavaScript Object Notation, um formato leve para troca de dados entre sistemas de computador. É uma alternativa ao XML, que é mais verborrágico.

[66] biblioteca para JavaScript. `http://jquery.com/`. Último acesso, 24/11/2011.

[67] em Redes de Computadores, TCP (Transmission Control Protocol) é um protocolo da camada de transporte.

[68] `http://en.wikipedia.org/wiki/WebSocket`. Último acesso, 24/11/2011.

[69] bidirecional

[70] `http://hacks.mozilla.org/2010/12/websockets-disabled-in-firefox-4/`. Último acesso, 24/11/2011.

3.4.5 Game Assets e Quick Help

Game Assets

> "O desenvolvimento de jogos é um processo ricamente multidisciplinar cuja eficiente construção passa pela compreensão de diversas técnicas e teorias." (5, p.51)

Para que os jogos sejam atrativos, precisamos de gráficos e sons interessantes. Artistas e compositores desempenham um papel crucial nesta questão. Assim, decidimos criar, para cada sala, uma área colaborativa onde são colocados os **Game Assets**: imagens, animações, músicas, efeitos sonoros e outros tipos de arquivos são compartilhados entre o grupo.

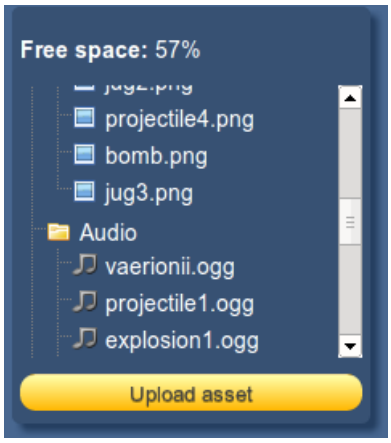


Figura 3.22: Área dos Game Assets

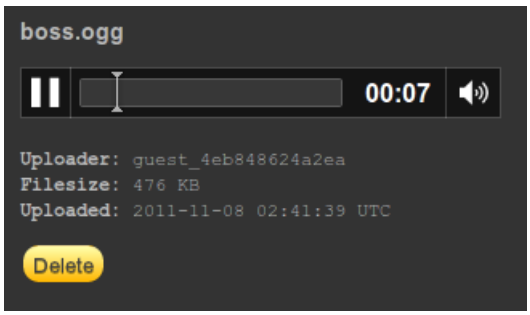


Figura 3.23: Pode-se visualizar/ouvir os assets

Assim que alguém envia um asset novo, ele já pode ser usado no jogo. Mais especificamente, através de uma interface intuitiva é feito o upload dos arquivos. Para que tudo seja colaborativo, assim como no sistema de chat, é feito um polling. Os mecanismos de web services, banco de dados, AJAX, etc. são todos similares aos discutidos na seção anterior.

Quick Help

Ao longo do trabalho, decidimos não criar um manual longo e pesado sobre os comandos da linguagem de programação. Ao invés disso, optamos por uma referência rápida e objetiva, localizada ao lado do campo de código. Tal escolha se justifica pelo fato de o ambiente ser voltado à criação de **protótipos rápidos** de jogos.

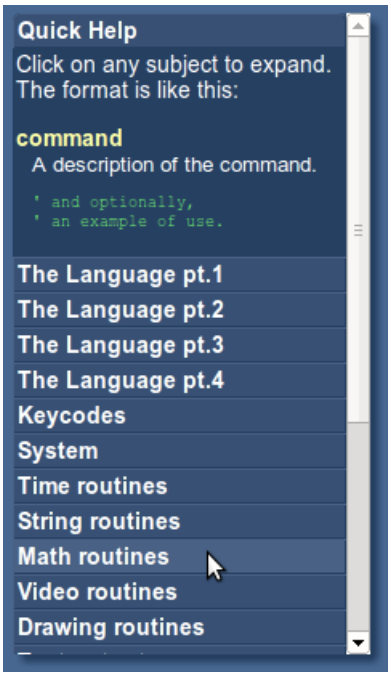


Figura 3.24: Guia rápido dos comandos

Do ponto de vista técnico, existe um arquivo escrito na linguagem de programação do projeto que é a "biblioteca padrão". Nele, há diversos comentários do tipo:

```
'' Draws an image.
'' @code
''     drawImage "bunny.png", 10, 10
'' @endcode
'' @param imageName name of the image to be displayed
'' @param x x-coordinate
'' @param y y-coordinate
```

Fazemos o *parsing* desse arquivo e devolvemos os dados relevantes para o usuário na forma de uma referência rápida. :-)

3.4.6 Campo de código compartilhado

Sendo este um ambiente colaborativo, o campo de código das salas é compartilhado. Precisamos então resolver o problema de múltiplos escritores concorrendo para escrever numa mesma base.

Tal problema não é fácil, mas, para a nossa felicidade, alguém já o resolveu! Neil Fraser[71], desenvolvedor do Google, criou o Mobwrite[72]. A partir da utilização do algoritmo **Differential synchronization**[73], pode-se fazer com que todos os clientes fiquem sincronizados. O esquema a seguir[74] mostra o funcionamento do algoritmo:

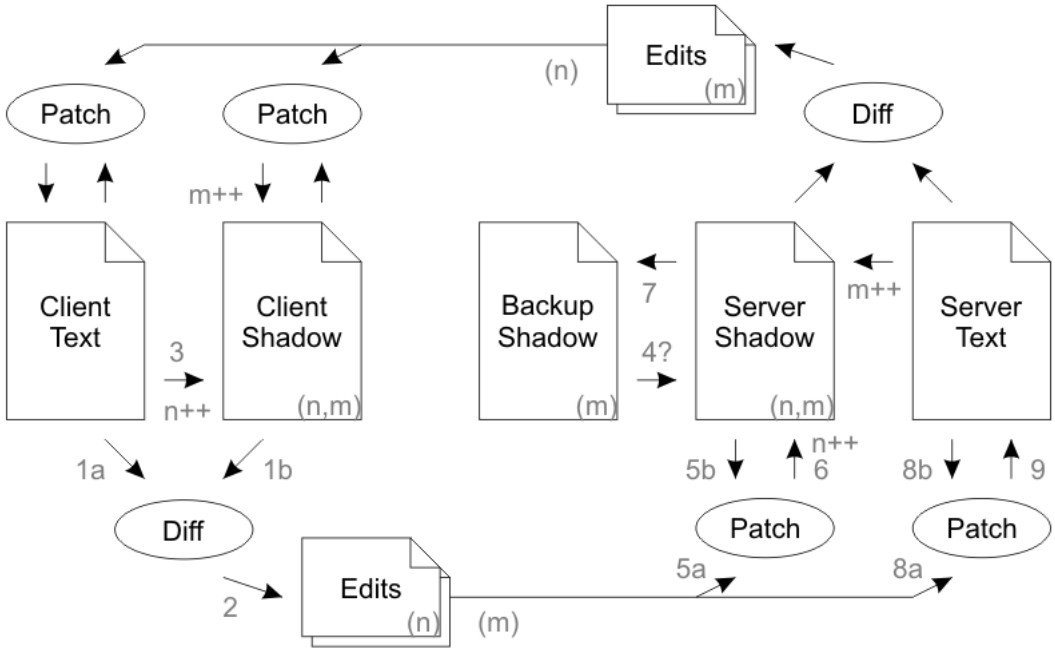


Figura 3.25: Differential synchronization

Perceba que o algoritmo é um tanto complicado. :-) O leitor interessado pode consultar o site do Neil Fraser para maiores informações. Lá, também é disponibilizado um vídeo de cerca de uma hora onde o Neil explica, em detalhes, toda a teoria. Para os nossos propósitos, basta uma explicação simplificada:

- O sistema é baseado no paradigma cliente-servidor;
- Assim como o bate-papo, o Mobwrite também faz polling;

[71] `http://neil.fraser.name/`. Último acesso, 25/11/2011.
[72] `http://code.google.com/p/google-mobwrite/`. Último acesso, 25/11/2011.
[73] `http://neil.fraser.name/writing/sync`. Último acesso, 25/11/2011.
[74] Autor: Neil Fraser

- Além da sincronização, o Mobwrite faz resolução de conflitos;
- Periodicamente, os clientes enviam ao servidor um *diff* do texto que eles tinham para o texto que eles têm. Isto captura eventuais mudanças que o usuário tenha realizado;
- O servidor faz um *patch* "esperto" com base na cópia do texto que ele tem e nos dados que recebe dos clientes. Ele então devolve *diffs* "adequados" a cada cliente;
- Clientes aplicam um *patch*;
- Para que tudo funcione, o Mobwrite combina, de uma forma bem bolada, ideias do protocolo TCP com ideias advindas dos sistemas de controle de versão (CVS, SVN, Git, Mercurial, etc.).

O Mobwrite disponibiliza uma API que, "como num passe de mágica", faz com que elementos de formulários HTML se tornem colaborativos (!). Até aí tudo bem. Porém, existem dois problemas:

1. O Mobwrite não implementa autenticação. Como as salas podem ser protegidas por senha[75], precisamos de uma boa autenticação! Caso contrário, qualquer pessoa com conhecimentos elementares de redes poderia *hackear*[76] o sistema;
2. Apenas com um uso simples da API, é impossível termos *syntax highlighting* no campo de código compartilhado. Precisamos de um plano!

O Mobwrite implementa um protocolo sob a camada de aplicação do TCP/IP. A imagem abaixo[77] mostra uma conversa entre cliente e servidor:

[75] dois tipos de senha: leitura-escrita, na qual um usuário pode modificar dados da sala e somente-leitura, na qual um usuário pode ver o que se passa, mas não pode alterar nada (útil para, por exemplo, dar aulas)

[76] burlar

[77] `http://code.google.com/p/google-mobwrite/wiki/Protocol`. Último acesso, 25/11/2011.

Client:

```
u:fraser
F:34:abcdef
d:41:=200 -7 +Hello =100
<blank>
```

Server:

```
f:42:abcdef
d:34:=305
<blank>
```

Figura 3.26: Protocolo

No lado do cliente, o campo **u:** (user) especifica o nome do usuário que envia a mensagem (não precisa ser necessariamente o mesmo nome do usuário do ambiente de criação de jogos; basta saber que existe um mapeamento entre este último nome de usuário e o nome de usuário do Mobwrite). Já o campo **d:** (delta) traz dados relacionados ao *diff*. **F:** (file) é o nome da base compartilhada, e esta é a chave para se fazer a autenticação. Um nome de base corresponde a um elemento de formulário HTML compartilhado. A análise do lado do servidor é análoga.

Numa utilização padrão do Mobwrite, o nome da base será o nome do elemento de formulário compartilhado. Problema: qualquer pessoa poderia ver o código fonte do HTML gerado pela aplicação web, descobrir qual é esse nome e comprometer o mecanismo de permissões. Idealmente, os usuários finais não devem ter conhecimento de tal informação.

A solução que bolamos foi introduzir um **proxy** entre cliente e Mobwrite, conforme a ilustração[78]:

[78] Adaptado de `http://en.wikipedia.org/wiki/File:Proxy_concept_en.svg`. Autor: H2g2bob. Domínio Público. Último acesso, 14/11/2011.

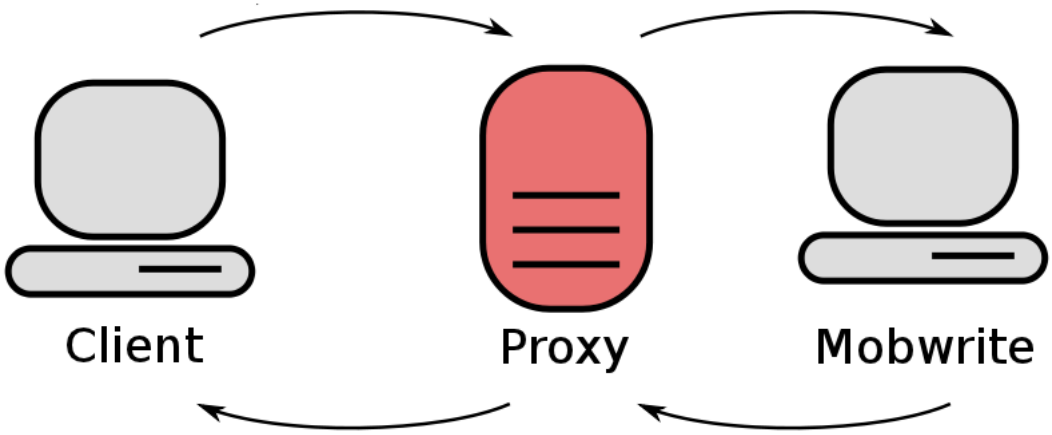


Figura 3.27: Proxy

Cada sala tem, no banco de dados, um nome de base associado. Todos os clientes, independentemente de quais salas participem, enviam um mesmo nome de base para o proxy. Lá, modificamos o campo **F:** (de acordo com a sala) e, via sockets, repassamos a mensagem ao Mobwrite: um *daemon*[79] que roda em algum servidor numa porta alta. A resposta que chega do *daemon* é novamente *hackeada* para que possamos consertar o nome da base que havia sido "estragado". :-)

Client:

```
u:fraser
F:34:abcdef secret_key
d:41:=200 -7 +Hello =100
<blank>
```

Server:

```
f:42:abcdef secret_key
d:34:=305
<blank>
```

Figura 3.28: "Hackeando" as mensagens

Observe que o nome da base nunca é repassado aos clientes.

[79]Disk And Execution MONitor (daemon) é um programa que roda em segundo-plano e que não precisa ser controlado diretamente por usuários.

É importante ressaltar que o uso do proxy pode estar causando problemas de concorrência. No momento da escrita deste texto, existe um *bug* que se manifesta (raramente, mas acontece) quando vários usuários estão escrevendo na base: começam a aparecer "coisas malucas" no texto. Apesar das intensas tentativas, não sabemos exatamente como reproduzir o problema. Também não sabemos a causa, mas desconfiamos que um eventual (raro) atraso provocado pelo uso do proxy inverta a ordem de chegada dos pacotes no Mobwrite, causando problemas. Isto é mera especulação. Não sabemos ao certo. O fato é que, atualmente, o projeto está hospedado numa dessas "hospedagens econômicas" genéricas no Brasil e o Mobwrite, num servidor "grátis" nos Estados Unidos. Nossa expectativa é conseguir, logo, um serviço de cloud mais apropriado. Desta forma, poderemos colocar o proxy e o Mobwrite numa mesma máquina, tornando eventuais atrasos desprezíveis.

Abordada a questão da autenticação, ainda temos o problema do realce de sintaxe (*syntax highlighting*), que nada mais é que apresentar o código-fonte do jogo dos usuários numa formatação específica, dependendo da categoria dos termos (normalmente isto envolve alterações de tipografia e modificações na coloração do texto). Para atingir tal meta, utilizamos a biblioteca CodeMirror[80].

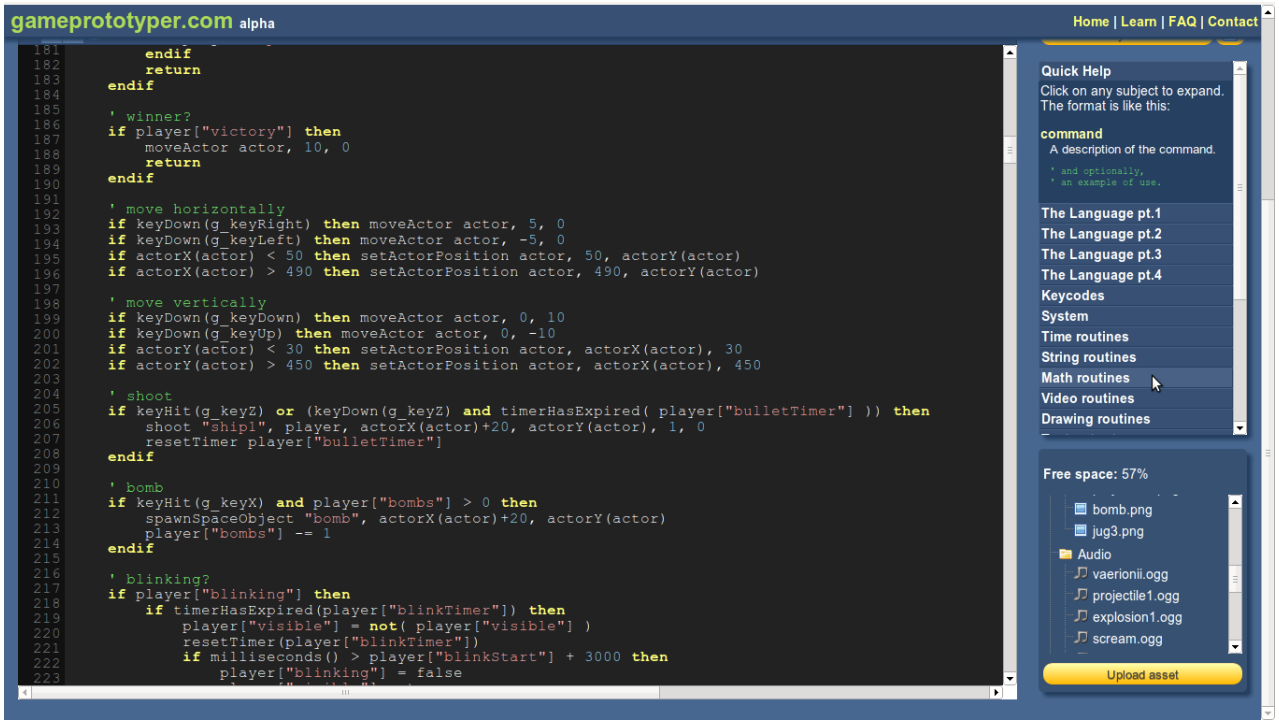


Figura 3.29: Campo de código compartilhado com realce de sintaxe

[80] `http://codemirror.net/`. Último acesso, 25/11/2011.

CodeMirror é uma biblioteca JavaScript que cria interfaces adequadas para editores de código que rodem no browser. Para que o código seja realçado corretamente, é preciso escrever, em JavaScript, um analisador sintático (ou adaptar um já existente).

Integrar o CodeMirror ao Mobwrite foi frustrante. Foram gastos dias em tentativas e erros. As dores de cabeça foram intensas e, para que as coisas funcionassem, tivemos que fazer certa engenharia reversa. O Mobwrite, apesar de ter uma API de fácil aprendizado para usos mais elementares, se mostrou difícil para um uso mais avançado. Adaptamos os mecanismos que aplicam *diff* e *patch* em formulários HTML para o CodeMirror. A documentação do CodeMirror, apesar de parecer boa para usos triviais, também tem lá suas esquisitices quando se deseja fazer qualquer coisa além.

Frustrações à parte, o importante é que conseguimos. :-)

3.4.7 Integração: Sistema web + Compilador + VM

Quando o usuário clica em "Play now!", algo mágico acontece: o jogo é executado! Tudo o que fizemos até agora, desde o início do projeto, culmina neste ponto.

Vejamos em detalhes como as partes do projeto se integram:

1. Usuário clica em "Play now!";
2. Cliente envia ao servidor o código-fonte do jogo;
3. Servidor faz as devidas verificações de segurança;
4. Servidor chama o compilador;
5. Compilador transforma o código-fonte do jogo em linguagem de máquina;
6. Servidor devolve o código compilado ao cliente;
7. Cliente recebe o código de máquina;
8. Cliente carrega o código de máquina na memória da Máquina Virtual;
9. Cliente liga a Máquina Virtual;
10. Máquina Virtual carrega os Game Assets;
11. A mágica acontece!

Eis o resultado:

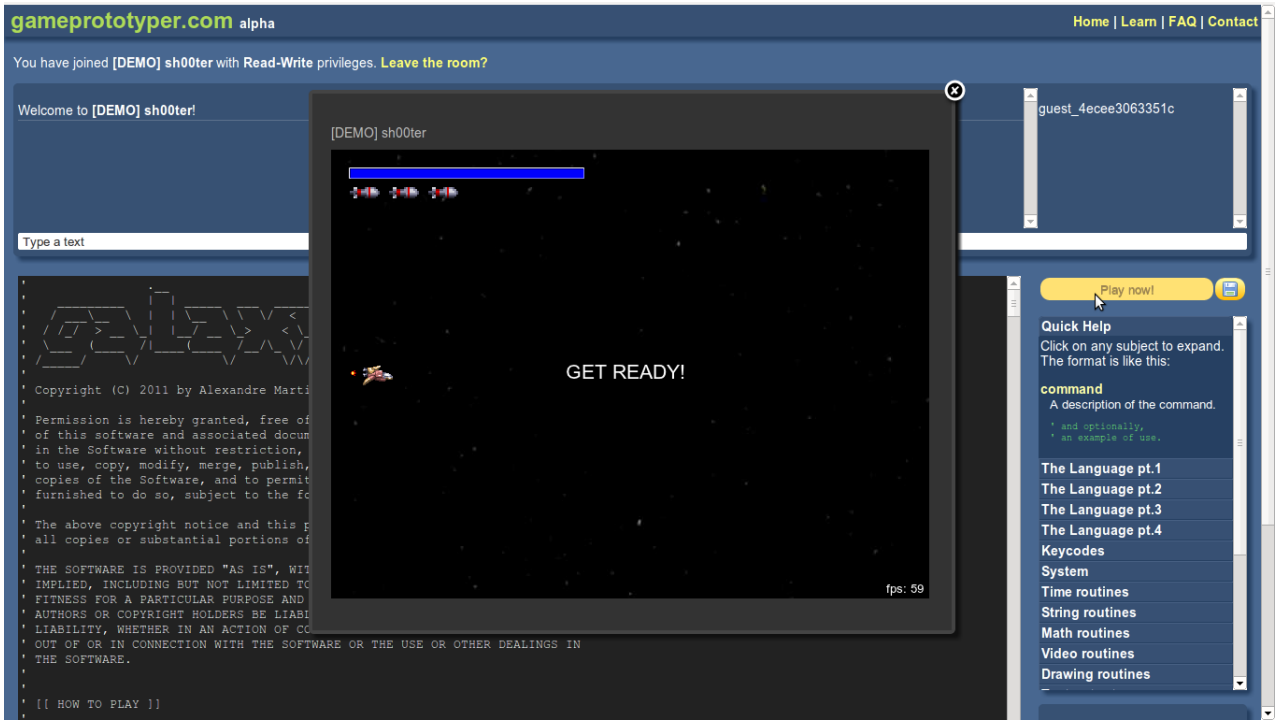


Figura 3.30: GET READY!

FUNCIONOU!!!!!!!!!!!!!!!!!!!!!!!!!!!!

YYYYYYYYEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEESSSSSSSSS!!!!!!!!!!!!!!!!!!!!

3.4.8 Conclusão

Procuramos fazer um trabalho de conclusão de curso bastante interdisciplinar. O Sistema Web é uma das facetas deste projeto. Para a construção desta última parte, esperamos que tenha ficado clara a relevância das disciplinas: Redes de Computadores, Banco de Dados, Engenharia de Software, Programação Concorrente, Programação Orientada a Objetos e Programação Funcional. Acerca deste tema, existem ainda outras disciplinas interessantes no departamento (Desenvolvimento Web, Desenvolvimento de Sistemas Colaborativos...), porém não cheguei a cursá-las.

4 Resultados

4.1 Apresentação do sistema

Ao entrar no sistema, o usuário se depara com a tela de autenticação. Pode-se optar por entrar anonimamente (útil para quem quiser conhecer brevemente a plataforma) ou por fazer a autenticação via OpenID.

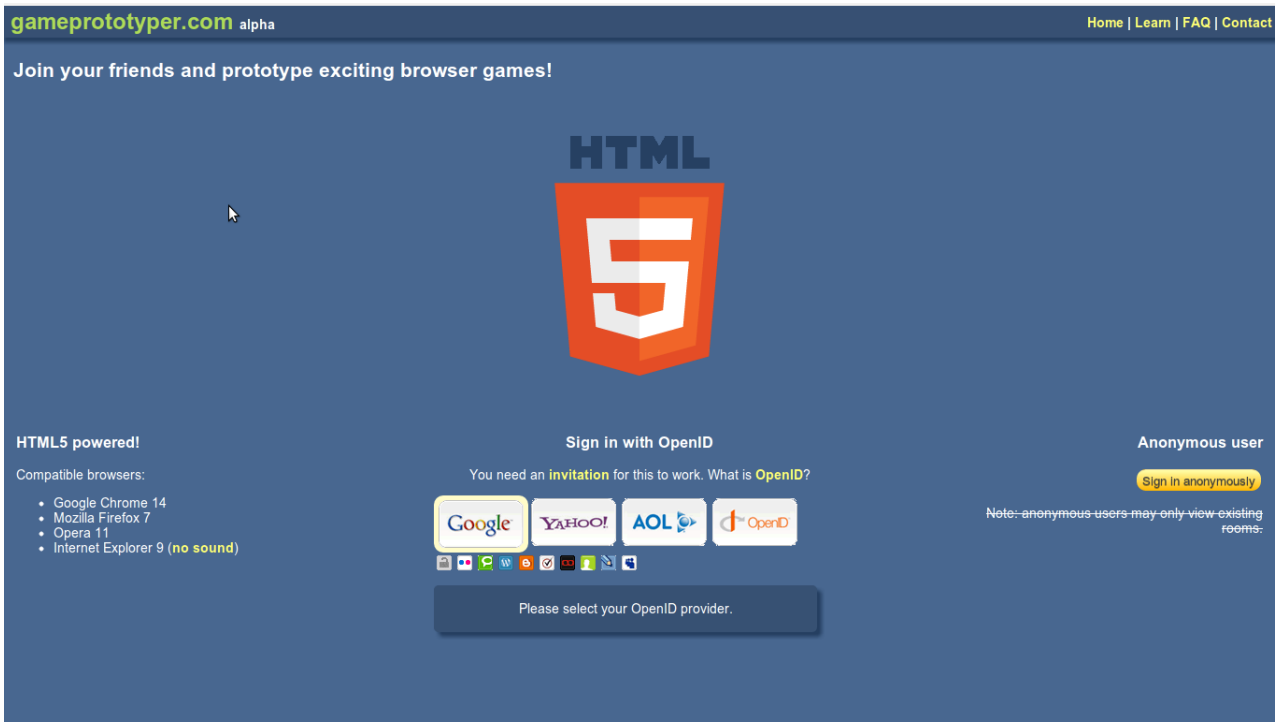


Figura 4.1: Tela de autenticação

Feita a autenticação, há uma lista de salas. Cada sala corresponde a um projeto sendo feito por uma ou mais pessoas. É possível entrar numa sala já existente (e, assim, colaborar com outras pessoas) ou criar uma nova.

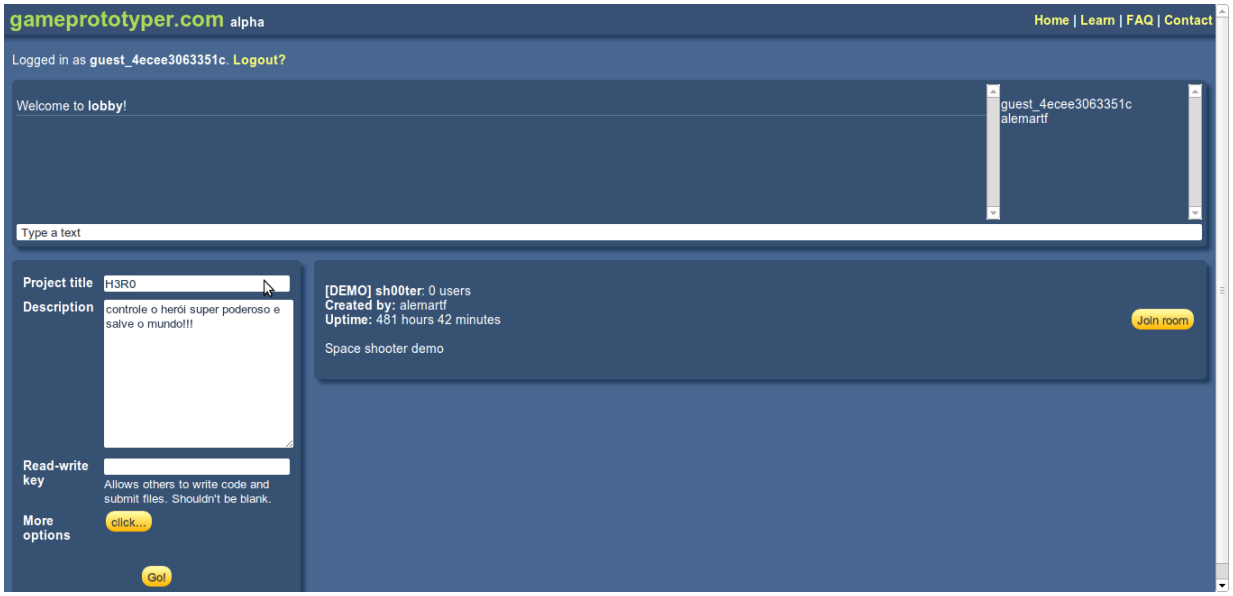


Figura 4.2: Lista de salas

Ao criar uma nova sala, o usuário deve informar o nome e uma pequena descrição do projeto que irá desenvolver. Além disso, ele pode, opcionalmente, progeter a sala com senhas. Senhas podem ser de dois tipos:

- **Read-Write Key**: pessoas que usarem esta senha terão privilégios de leitura e de escrita na sala. Isto significa que elas poderão acompanhar o desenvolvimento do jogo, bem como fazer alterações no mesmo;
- **Read-Only Key**: pessoas que usarem esta senha terão apenas privilégios de leitura. Em outras palavras, elas poderão acompanhar o desenvolvimento do jogo, porém não poderão modificar o que está sendo feito. Isto é útil para quem quiser dar aulas;

É possível que uma sala seja criada sem senha, com apenas uma categoria de senha, ou com ambas.

Após entrar numa sala, o usuário irá se deparar com o ambiente colaborativo de desenvolvimento:

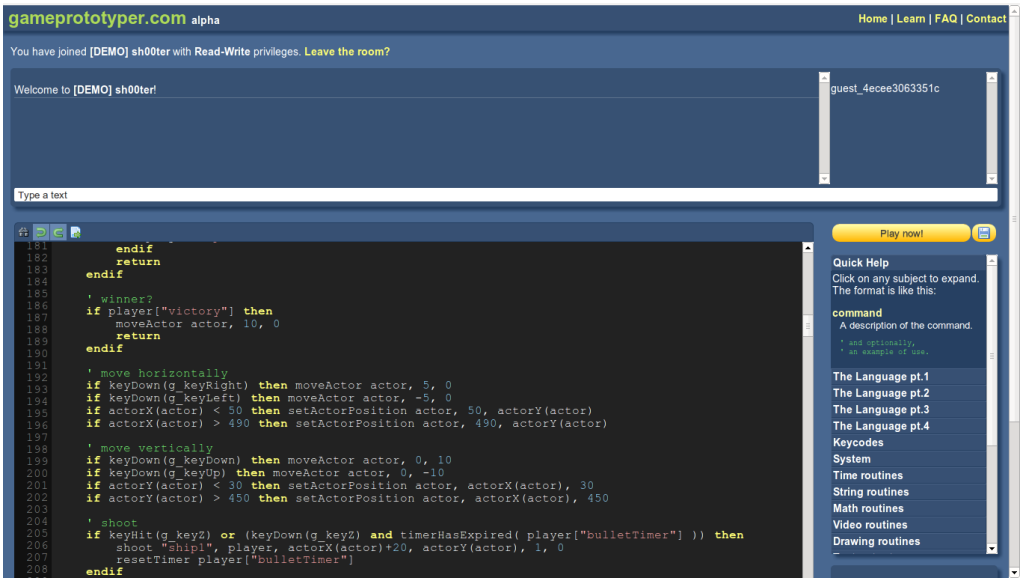


Figura 4.3: Ambiente de desenvolvimento

Há, mais acima, um bate-papo onde todos os participantes do projeto podem trocar ideias. Além disso, pode-se ver na figura um campo de código compartilhado e um pequeno manual de instruções (*Quick Reference*) que, de maneira rápida e prática, mostra os comandos da linguagem de programação.

Abaixo do manual, há uma área - também compartilhada - onde se colocam os *game assets*: usuários podem enviar imagens, músicas, sons, animações, etc. e, desta forma, tais recursos poderão ser utilizados nos jogos.

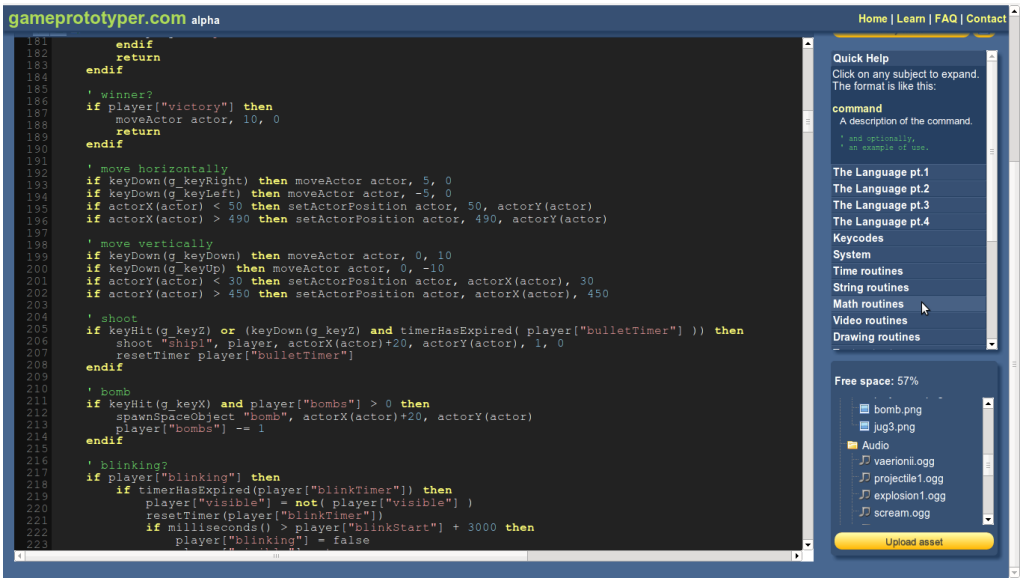


Figura 4.4: Ambiente de desenvolvimento

À medida que o jogo for sendo desenvolvido, os usuários podem clicar em *Play now!* para testá-lo. Isto invocará o compilador e iniciará a execução do game.

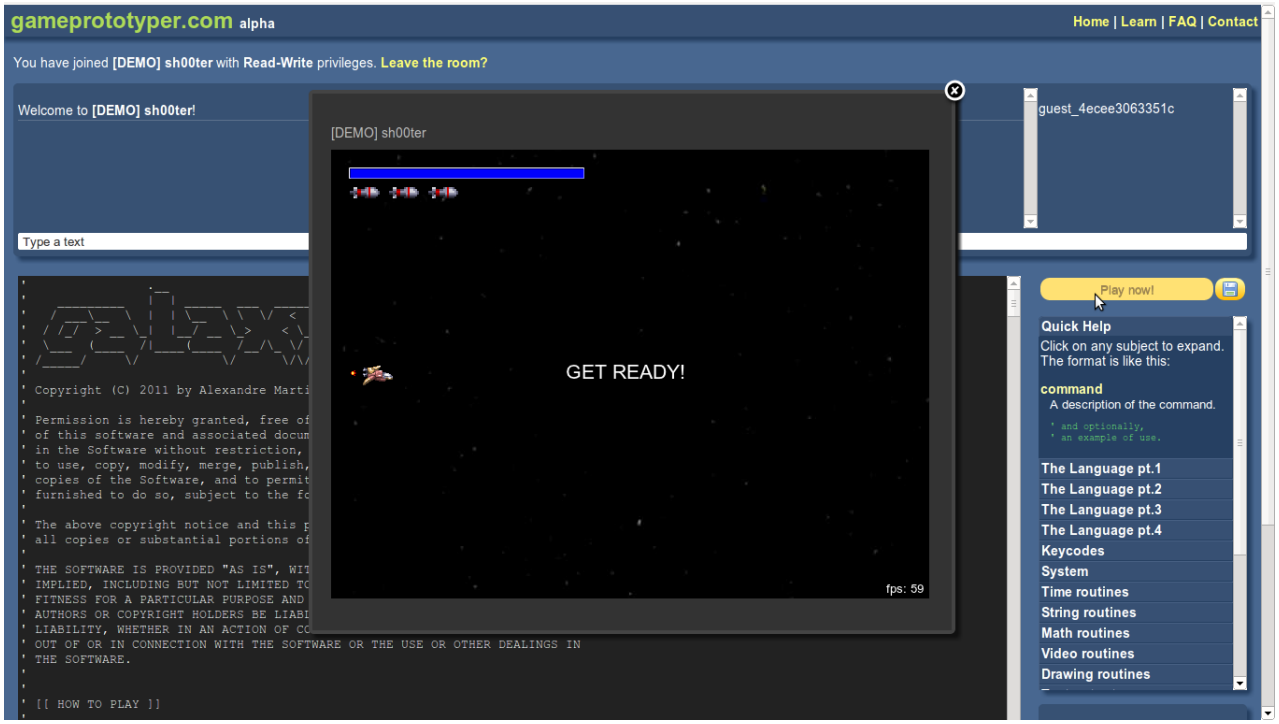
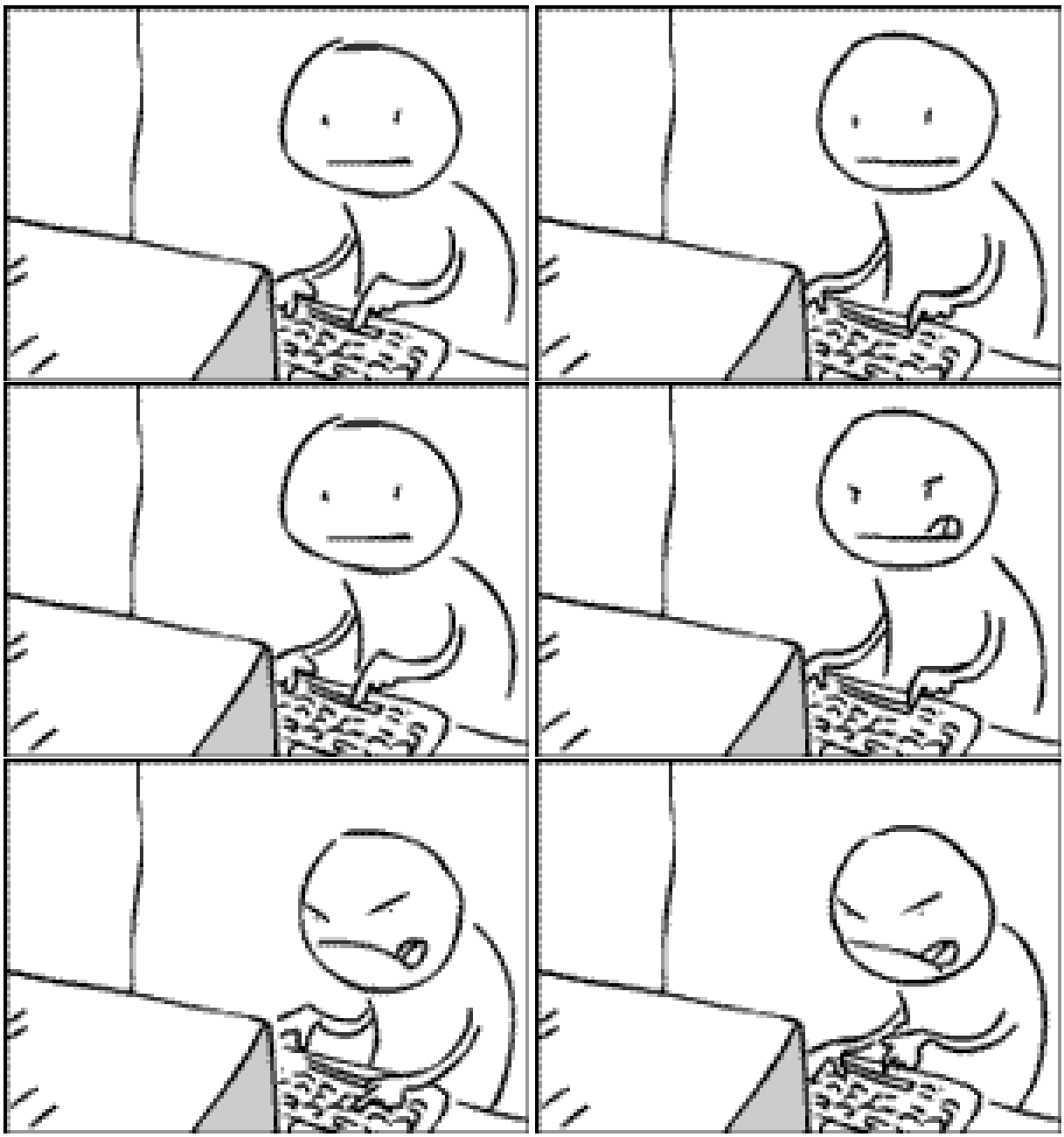


Figura 4.5: A execução de um jogo

4.2 Desafios e frustrações

Desafios e frustrações.[1]



[1]Fonte: `http://img.photobucket.com/albums/v444/ryuuichimitsukai/computer_programmer.gif`. Autor desconhecido.

:-)

4.3 Breve discussão

Em primeiro lugar, sempre tive apoio dos meus pais. Sem isto, não chegaria a lugar algum.

O desenvolvimento do projeto foi muito trabalhoso, porém igualmente empolgante e enriquecedor. Criar soluções úteis para outras pessoas é algo que me interessa, e o curso de Ciência da Computação da Universidade de São Paulo dá as ferramentas para alcançar tal objetivo.

Logo no início do ano, quando fui falar com o Prof. Flávio a respeito da ideia (que até então só era algo doido da imaginação de um aluno que acho que ele nunca tinha visto na vida - tive receio de que ninguém iria comprar a ideia...[2] - ainda bem que ele aceitou minhas maluquices!! :-), considero que foi importante ter uma visão clara do que queria fazer. Embora eu não soubesse, nos mínimos detalhes, todos os detalhes técnicos necessários para a implementação, pelo menos vislumbrava qual o caminho que deveria tomar. Se alguma questão técnica surgia, sabia onde procurar por respostas. Se alguma questão acadêmica aparecia, o professor dava conselhos.

Foi igualmente importante, em conjunto com o professor, definirmos um método de condução do trabalho. De acordo com a sugestão dele, combinamos que, a cada duas semanas, ele receberia algum progresso do projeto.

- ... mas olha, não vá usar esse tempo e depois me dizer que você só estudou tal e tal coisa.

- Ah, então você quer algo visível?

- É, algo visível.

Assim, quarta-feira sim / quarta-feira não, rigorosamente, ele recebia alguma atualização "visível" do projeto: algum avanço na implementação ou na monografia. Isso fez com que um bom ritmo de trabalho fosse mantido durante o ano todo (não fosse isso, o TCC não teria sido concluído). Quando, em setembro, chegou o momento de entregar uma versão parcial do trabalho para o Prof. Carlinhos, não houve correria: já estávamos fazendo entregas parciais desde o início. :-)

Quanto aos problemas técnicos que apareceram, houve algumas dificuldades aqui e ali com o HTML5 Audio, com o campo de código compartilhado e com outras coisinhas a mais. Entretanto, tudo isso pôde ser enfrentado. O trabalho de formatura não foi nem fácil demais, a ponto de ser entediante, e nem difícil demais, a ponto de me fazer desistir. O projeto foi desafiador na medida certa. A diversidade das disciplinas envolvidas, das linguagens pertinentes (desde o baixo até o alto nível) e das tecnologias utilizadas é algo que considero fantástico. Observar

[2] ainda mais porque não tenho o hábito de ir em sala de professor :(

meu amigo *designer*[3] brincando com lógica de programação e aprendendo ao mesmo tempo em que se divertia foi algo que me deixou muito feliz: todos esses meus conhecimentos técnicos estão num contexto maior - eles servem para alguma coisa. Gosto de mexer nos detalhes, mas também gosto da "big picture" das coisas.

Planos futuros dependem da demanda. Como o ambiente reside na nuvem, não basta simplesmente jogá-lo num site de downloads e achar que tudo acabou: existe um custo para continuar o desenvolvimento e para mantê-lo no ar. Se isso não for pago, o projeto morre. Embora eu tenha pensando inicialmente em criar um software voltado a desenvolvedores de jogos independentes e escolas/universidades, a verdade é que eu não sei de absolutamente nada. :-) Num cenário de extrema incerteza, é conveniente perceber algum padrão nas demandas dos interessados. Talvez um ou outro ajuste de direção se faça necessário.

Apesar de tudo, acho factível disponibilizar, pelo menos durante algum tempo, uma versão funcional na Internet para ver o que acontece. Antes disso, entretanto, é preciso:

1. **Melhorar o desempenho da Máquina Virtual**: a velocidade dos jogos ainda não é excelente em todos os browsers (embora seja boa em alguns). Há meios de se melhorar isso e eu sei como fazer. Não melhorei até agora porque o prazo do TCC é curto e, como o projeto como um todo foi consideravelmente trabalhoso, decidi que seria menos arriscado entregar "a coisa mais simples que funcionasse". O fato é que os motores de JavaScript estão cada vez mais "parrudos". O Google V8 JavaScript Engine[4], por exemplo, compila JavaScript em código nativo de máquina[5] - o que é incrível! Creio que posso tirar proveito disso e fazer com que os programas criados na minha linguagem tenham desempenho comparável a programas nativos;

2. **Criar uma área de jogos de demonstração**: embora eu tenha colocado alguns jogos de demonstração na monografia (veja a seguir), seria legal se o sistema web tivesse uma área dedicada a eles. Assim, os usuários poderiam avaliar melhor o que a ferramenta é capaz de fazer;

3. **Criar um Twitter**[6]: é importante que pessoas usem e acompanhem a ferramenta.

A princípio, será só isso. Para fazer mais, preciso de demanda e preciso que meus custos sejam bem pagos. Outras possibilidades, a médio prazo, incluem: disponibilizar meios para que

[3] Veja o jogo "Ancient Aliens" adiante.
[4] O V8 é o motor JavaScript usado no Google Chrome.
[5] `http://code.google.com/apis/v8/design.html#mach_code`. Último acesso, 30/11/2011.
[6] `http://www.twitter.com`. Último acesso, 30/11/2011.

os jogos sejam integrados a redes sociais (o que, acredito, beneficiaria os usuários), criar uma funcionalidade para exportar os games para desktop e para mobile[7], criar uma biblioteca de *game assets* (livres de pagamento de *royalties* por parte dos usuários) para serem prontamente incluídos nos jogos (para criar tal biblioteca, é necessário ajuda de artistas), além de otimizar o compilador. A longo prazo, quem sabe (já estou *viajando na maionese*...), poderia possibilitar a criação de jogos 3D via WebGL[8] e de games multiplayer via WebSockets. Apesar de todas essas ideias malucas, na realidade tudo depende da demanda.

Embora o trabalho não seja o "ápice da perfeição", considero que ele está *bom o suficiente* para o TCC. Estou satisfeito com os resultados obtidos. Espero que vocês também gostem. Agora, realmente, quero minhas férias. :-)

Ao que tudo indica, o HTML5 é o futuro da Web[9]. Neste ponto, Google, Microsoft, Apple, Mozilla, Opera, W3C e até mesmo a Adobe[10] parecem concordar.

Figura 4.6: Fonte: http://www.w3.org/html/logo/

[7] preciso juntar uns $$$ e comprar um smartphone, pois fica difícil dar suporte a mobile se eu não tiver um...

[8] recurso do HTML5 que permite gerar gráficos 3D dentro do browser. Fonte: `http://en.wikipedia.org/wiki/WebGL`. Último acesso, 30/11/2011.

[9] `http://logicpath.com/blog/general/html5-is-the-future-of-the-web`. Último acesso, 30/11/2011.

[10] uma versão online da reportagem impressa(10) está disponível em `http://blogs.estadao.com.br/link/adobe-desiste-de-flash-para-aparelhos-moveis/`. Último acesso, 30/11/2011.

4.4 Jogos de demonstração

4.4.1 Hello world

```
print "Hello, world!"
```

Dispensa explicações.

4.4.2 Guess a number

Neste joguinho, o usuário deve adivinhar o número sorteado pelo computador.

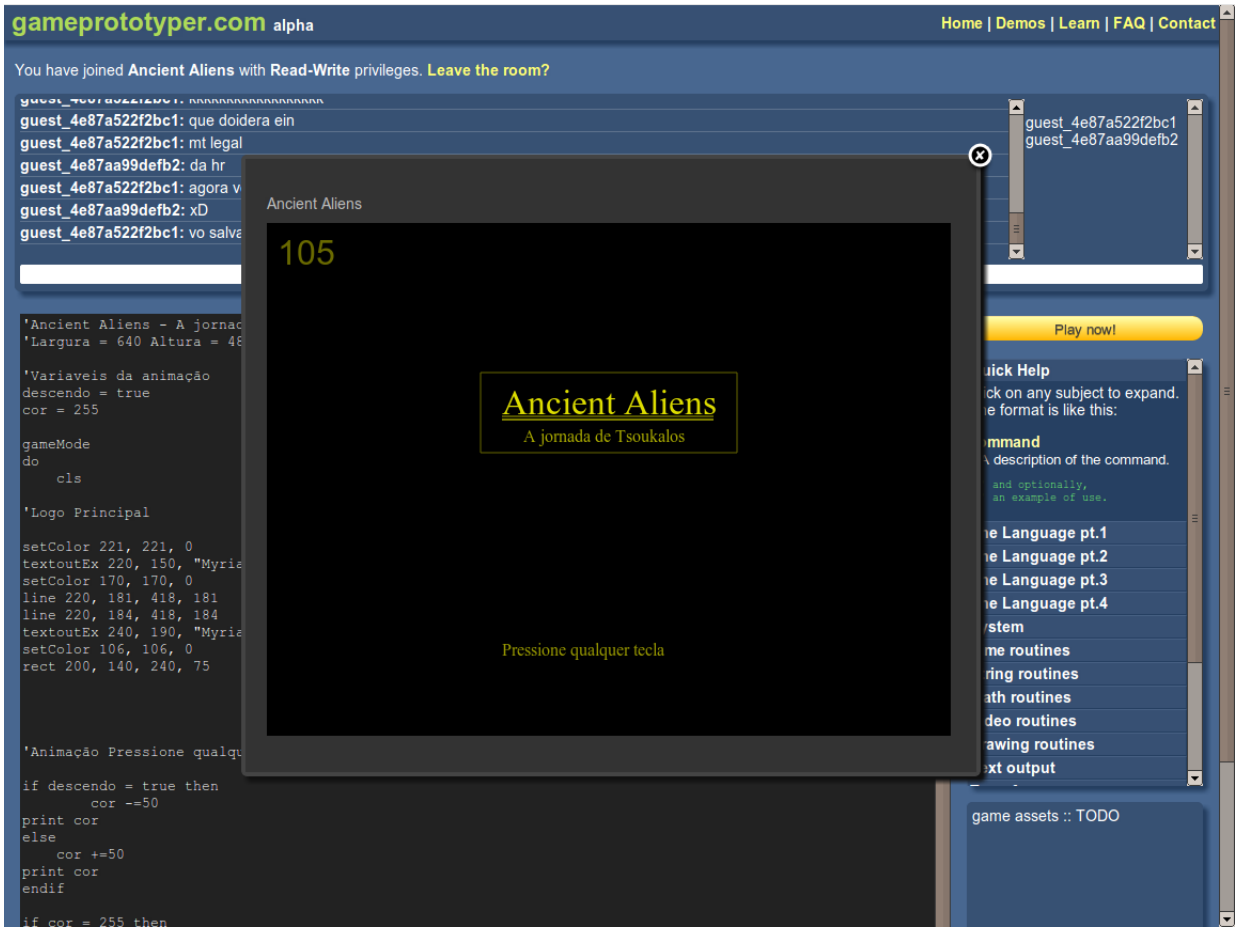
```
do
    cpuNumber = random(1, 10)
    msgbox "Pensei num número de 1 a 10. Vamos ver se você o adivinha!"
    userNumber = input("Digite um número de 1 a 10")

    if userNumber = cpuNumber then
        msgbox "Você acertou!!!"
    else
        msgbox "Você errou. O número era " + cpuNumber
    endif

    playAgain = confirm("Jogar novamente?")
    if not playAgain then exit
loop
```

4.4.3 “Ancient Aliens”

O primeiro usuário do ambiente de criação de jogos foi o Ramon, um amigo de quem vos fala. :-) Ele não é programador, mas conseguiu desenvolver uma animação bacana inspirada numa série de televisão:



Segue abaixo o código 100% escrito pelo Ramon:

```
'Ancient Aliens - A jornada de Tsoukalos
'Largura = 640 Altura = 480

'Variaveis da animação
descendo = true
cor = 255

gameMode
do
    cls

'Logo Principal

setColor 221, 221, 0
```

```
textoutEx 220, 150, "Myriad Pro", 25, "Ancient Aliens"
setColor 170, 170, 0
line 220, 181, 418, 181
line 220, 184, 418, 184
textoutEx 240, 190, "Myriad Pro", 12, "A jornada de Tsoukalos"
setColor 106, 106, 0
rect 200, 140, 240, 75



'Animação Pressione qualquer tecla

if descendo = true then
        cor -=50
print cor
else
    cor +=50
print cor
endif

if cor = 255 then
    descendo = true
endif

if cor <= 0 then
    descendo = false
endif

setColor cor, cor, 0
textoutEx 220, 390, "Myriad Pro", 12, "Pressione qualquer tecla"


    flip
loop
```

A seguir, tomamos a liberdade de introduzir um pouco de subjetividade neste exemplo. Acreditamos que isto ilustrará muito bem o que é um ambiente colaborativo. :-) Colocamos um trecho da conversa entre o Ramon e quem desenvolve o TCC. Tal conversa se deu no próprio mecanismo de bate-papo da plataforma, sendo que o assunto foi como fazer o texto “Pressione qualquer tecla” piscar de forma suave.

```
guest_4e87aa99defb2: ctrl f5
guest_4e87a522f2bc1: pronto
guest_4e87aa99defb2: foi?
guest_4e87a522f2bc1: o que
guest_4e87aa99defb2: melhorou o help?
guest_4e87a522f2bc1: ah ta
guest_4e87a522f2bc1: verdade
guest_4e87a522f2bc1: nossa
guest_4e87a522f2bc1: tava boiando tentando resolver o negocio
guest_4e87a522f2bc1: ehauiehuiaeheaea
guest_4e87aa99defb2: vai tentando, mas nao precisa dessa logica toda complicada
guest_4e87a522f2bc1: eu to fazendo de um jeito mais complicado dq o necessario?
guest_4e87aa99defb2: sim
guest_4e87a522f2bc1: nossa
guest_4e87a522f2bc1: aheuiaehuiaehueai agora fiquei curioso
guest_4e87aa99defb2: n precisa desses loops ai
guest_4e87aa99defb2: lembra q vc ja ta dentro do loop principal (o do ... cls
... flip ... loop)
guest_4e87aa99defb2: nao precisa de +
guest_4e87a522f2bc1: ahh
guest_4e87a522f2bc1: mas
guest_4e87a522f2bc1: é que nao funcionou
guest_4e87a522f2bc1: antes eu fiz sem o Do dentro do if
guest_4e87aa99defb2: lembra q a cor só varia de 0 a 255
guest_4e87a522f2bc1: e nao adicionou nd
guest_4e87aa99defb2: só um cor += 1, -= 1 ja resolve
guest_4e87a522f2bc1: entao, mas eu so tava tentando fazer um tempo normal
guest_4e87a522f2bc1: pra ver ele agindo
guest_4e87a522f2bc1: entao pq nao funcionou antes? tava errado?
guest_4e87aa99defb2: eu nao vi =x tava dando ctrl+f5 pra aumentar o help
```

guest_4e87a522f2bc1: ahuauhauhauha
guest_4e87a522f2bc1: vo deixar do jeito antigo pra vc ver
guest_4e87a522f2bc1: viu?
guest_4e87a522f2bc1: fica parado no 99
guest_4e87aa99defb2: xo compilar
guest_4e87aa99defb2: entao
guest_4e87aa99defb2: qd o codigo vê o "loop", ele volta pra linha do "do", certo?
guest_4e87aa99defb2: aí vc t[a fazendo cor=100 toda hora
guest_4e87a522f2bc1: aham
guest_4e87aa99defb2: entao ele so vai mostrar 99
guest_4e87aa99defb2: pq vc ta voltando pra 100 sempre
guest_4e87aa99defb2: e vc ta fazendo descendo = true sempre
guest_4e87aa99defb2: tendeu?
guest_4e87a522f2bc1: sim
guest_4e87a522f2bc1: mas entao tenho que escrever essas variaveis fora do loop, isso?
guest_4e87aa99defb2: sim
guest_4e87a522f2bc1: ah, legal!!
guest_4e87a522f2bc1: vlw!!
guest_4e87a522f2bc1: HAHAHAHAHhahaa
guest_4e87a522f2bc1: funcionaaaaaaaaaaaaaa
guest_4e87aa99defb2: óóóóóóó
guest_4e87a522f2bc1: funciona
guest_4e87aa99defb2: compilei e ja ta contando!
guest_4e87a522f2bc1: q dahora mano
guest_4e87aa99defb2: ooooooooia
guest_4e87a522f2bc1: pqp genial
guest_4e87a522f2bc1: q dahora
guest_4e87a522f2bc1: sua linguagem é demais
guest_4e87aa99defb2: agora faz mudar a cor
guest_4e87a522f2bc1: ahuahuahuahuahuahuahuaa
guest_4e87a522f2bc1: blz
guest_4e87aa99defb2: bota pra mudar de intensidade + rapido tb
guest_4e87a522f2bc1: ehh
guest_4e87a522f2bc1: verdad

```
guest_4e87a522f2bc1: e
guest_4e87a522f2bc1: AUHAAHuuhauhauhauhuhahuaa
guest_4e87a522f2bc1: aahuauhahuahuahua
guest_4e87a522f2bc1: xD
guest_4e87aa99defb2: xo compilar
guest_4e87a522f2bc1: que viagem mano
guest_4e87aa99defb2: nooooossssssssssa muito SNES mano!
guest_4e87a522f2bc1: ahuauhuhauhauhauhauhauhaa
guest_4e87a522f2bc1: kkkkkkkkkkkkkkkkk
guest_4e87a522f2bc1: que doidera ein
guest_4e87a522f2bc1: mt legal
guest_4e87aa99defb2: da hr
guest_4e87a522f2bc1: agora vou dormir satisfeito!
```

Que bom, um usuário satisfeito! :-)

4.4.4 Jogo da velha

Em Teoria dos Jogos, um jogo estratégico com informação perfeita descreve situações em que os jogadores têm todos os dados necessários para determinar todas as possíveis combinações de movimentos/ações (e, portanto, todos os possíveis resultados das partidas[11]). Existem também os jogos extensivos, que descrevem situações onde a ordem das jogadas é importante[12]. O jogo da velha é um exemplo de jogo extensivo com informação perfeita.

Criamos uma versão do jogo da velha em que o usuário joga contra o computador. Para que este último fizesse boas jogadas, foi utilizado um algoritmo de Inteligência Artificial conhecido como Minimax[13].

> "Jogos-da-velha que utilizam esse algoritmo frequentemente resultam em empate, se o jogador não cometer erros. O algoritmo não cometerá. Em virtude dessa possibilidade, geralmente se coloca nível de dificuldade no jogo. Deste modo, o computador pode cometer algumas falhas aleatórias, para simular certa 'falta' de inteligência."(5, p.155)

[11] `http://en.wikipedia.org/wiki/Perfect_information`. Último acesso, 01/12/2011.
[12] `http://pt.wikipedia.org/wiki/Teoria_dos_jogos#Forma_extensiva`. Último acesso, 01/12/2011.
[13] `http://en.wikipedia.org/wiki/Minimax`. Último acesso, 01/12/2011.

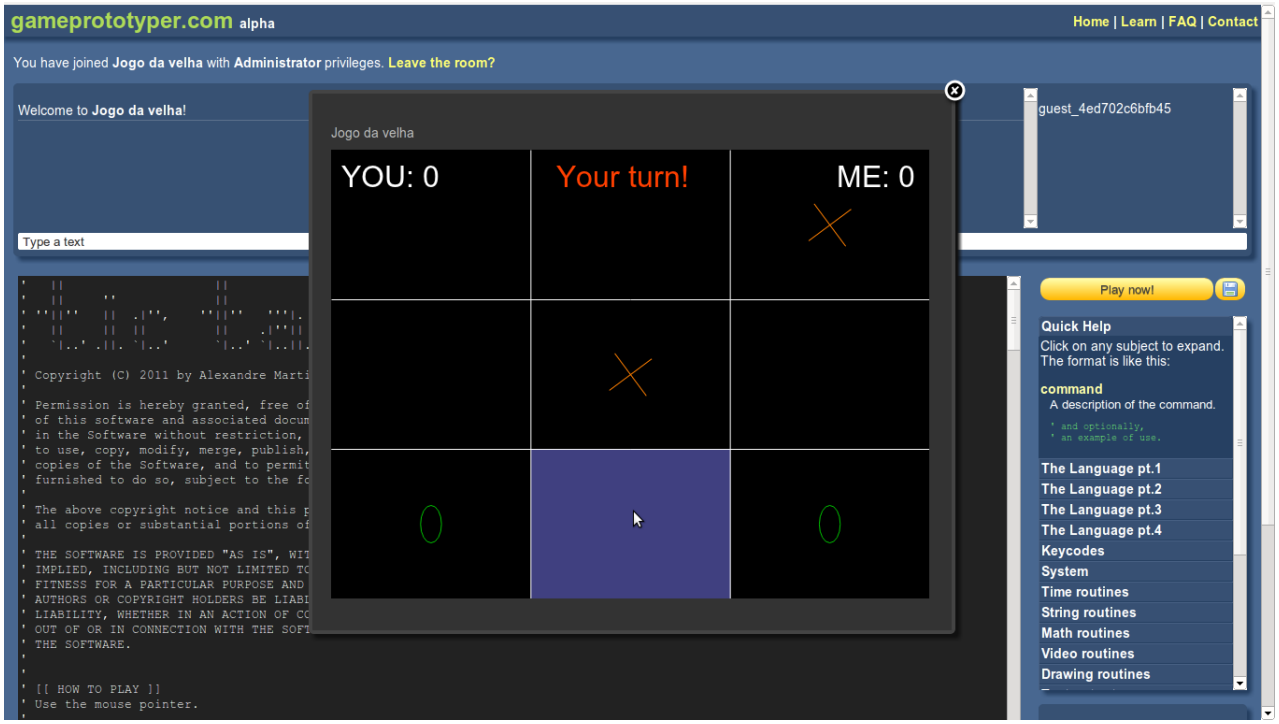


Figura 4.7: Jogo da velha

Segue o código do jogo:

```
'   ||                    ||                    ||
'   ||     ''             ||                    ||
' ''||''   ||  .|'',    ''||''   '''|.  .|'',    ''||''  .|''|, .|''|,
'   ||     ||  ||         ||    .|''||  ||         ||    ||  || ||..||
'   `|..' .||. `|..'      `|..' `|..||. `|..'      `|..' `|..|' `|...
'
' Copyright (C) 2011 by Alexandre Martins <alemartf at gmail.com>
'
' Permission is hereby granted, free of charge, to any person obtaining a copy
' of this software and associated documentation files (the "Software"), to deal
' in the Software without restriction, including without limitation the rights
' to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
' copies of the Software, and to permit persons to whom the Software is
' furnished to do so, subject to the following conditions:
'
' The above copyright notice and this permission notice shall be included in
' all copies or substantial portions of the Software.
```

```
'
' THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
' IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
' FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
' AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
' LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
' OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
' THE SOFTWARE.
'
'
' [[ HOW TO PLAY ]]
' Use the mouse pointer.
'



' ----------------------------------
' TIC-TAC-TOE BOARD
' ----------------------------------

' creates a new board
fun createBoard()
    table = createTable()

    ' some generic data...
    table["playerScore"] = 0
    table["cpuScore"] = 0
    table["draws"] = 0
    table["turn"] = 0

    ' select a level
    table["level"] = askLevel()

    ' the cells of the 3x3 tic-tac-toe matrix
    for i=0 to 8
        table[i] = " "
```

```
    next

    ' done
    return table
endfun

' destroys an existing board
fun destroyBoard(board)
    destroyTable board
endfun

' resets an existing board
fun resetBoard(board, winner)
    ' reseting data
    board["turn"] = random(0,1)
    board["level"] = askLevel()
    for i=0 to 8
        board[i] = " "
    next

    ' processing the winner
    if winner = "x" then board["playerScore"] += 1
    if winner = "o" then board["cpuScore"] += 1
    if winner = "d" then board["draws"] += 1
endfun

' defines a new value ("o" or "x") for a cell of the
' board. 0 <= row,col <= 2
fun setBoardCell(board, row, col, value)
    board[row*3 + col] = value
    board["turn"] = (board["turn"] + 1) mod 2
endfun

' gets the value of a board cell (" ", "o" or "x")
fun getBoardCell(board, row, col)
```

```
    return board[row*3 + col]
endfun

' draws a board to the screen
fun drawBoard(board)

    ' initializing
    w = screenWidth()
    h = screenHeight()

    ' cell values
    ox = getOriginX()
    oy = getOriginY()
    for row=0 to 2
        for col=0 to 2
            value = getBoardCell(board, row, col)
            setOrigin((col+0.5)*w/3, (row+0.5)*h/3)
            if value = "x" then
                setcolor 255,128,0
                r = getRotation()
                setRotation 15 * cos(0.21 * milliseconds())
                line -20, -20, 20, 20
                line 20, -20, -20, 20
                setRotation r
            elseif value = "o" then
                setcolor 0,255,0
                s = getScaleX()
                setScale abs(cos(0.21 * milliseconds())), getScaleY()
                circle 0, 0, 20
                setScale s, getScaleY()
            elseif board["turn"] = 0 then
                if mouseX() > col*w/3 and mouseX() < (col+1)*w/3 then
                    if mouseY() > row*h/3 and mouseY() < (row+1)*h/3 then
                        setcolor 64,64,128
                        rectfill -w/6, -h/6, w/3, h/3
```

```
                    endif
                endif
            endif
        next
    next
    setOrigin ox, oy

    ' the board
    setcolor 255, 255, 255
    line w/3, 0, w/3, h
    line 2*w/3, 0, 2*w/3, h
    line 0, h/3, w, h/3
    line 0, 2*h/3, w, 2*h/3

endfun

' is there a winner?
' returns: "x" (player), "o" (cpu), "d" (draw) or " " (if there's no winner yet)
fun gotWinner(board)

    ' check rows and cols
    for a=0 to 2
        if board[3*a] <> " " and board[3*a] = board[3*a+1] and
board[3*a] = board[3*a+2] then return board[3*a]
        if board[a] <> " " and board[a] = board[a+3] and
board[a] = board[a+6] then return board[a]
    next

    ' check diagonals
    if board[4] <> " " then
        if board[4] = board[0] and board[4] = board[8] then return board[4]
        if board[4] = board[2] and board[4] = board[6] then return board[4]
    endif

    ' draw?
```

```
    draw = true
    for a=0 to 8
        if board[a] = " " then draw = false
    next
    if draw then return "d"

    ' no winners yet.
    return " "

endfun

' ask for a game level (easy, medium or hard)
fun askLevel()
    lv = "?"
    while lv <> "" and lv <> "easy" and lv <> "medium" and lv <> "hard"
        lv = input("Please type: 'easy', 'medium' or 'hard' ?")
        lv = lowercase(lv)
    wend
    if lv = "" then lv = "hard"
    msgbox uppercase(lv) + " mode!"
    return lv
endfun



' ---------------------------------------
' LOGIC
' ---------------------------------------

' player's turn!
fun playerTurn(board)
    w = screenWidth()
    h = screenHeight()

    for row=0 to 2
```

```
        for col=0 to 2
            ' got a free cell?
            if getBoardCell(board, row, col) = " " then
                if mouseX() > col*w/3 and mouseX() < (col+1)*w/3 then
                    if mouseY() > row*h/3 and mouseY() < (row+1)*h/3 then
                        if mouseHit(1) then setBoardCell(board, row, col, "x")
                    endif
                endif
            endif
        next
    next
endfun

' CPU's turn!
fun cpuTurn(board)
    if (board["level"] = "easy" or (board["level"] = "medium" and
random(0,7) < 1)) then
        ' easy mode = pick a cell at random
        do
            row = random(0,2)
            col = random(0,2)
            if getBoardCell(board, row, col) = " " then
                setBoardCell(board, row, col, "o")
                break
            endif
        loop
    else
        ' hard mode = minimax algorithm
        minimax board, 2, 1
        col = board["best"] mod 3
        row = (board["best"] - col) / 3
        setBoardCell(board, row, col, "o")
    endif
endfun
```

```
' someone should play now!
fun processTurn(board)
    if board["turn"] = 0 then
        playerTurn board
    else
        cpuTurn board
    endif
endfun

' ----------------------------------------
' HUD
' ----------------------------------------

' draws the hud
fun drawHud(board)
    a = getAlpha()

    setAlpha 0.5
    setcolor 0, 32, 128
    'rectfill 0, 0, screenWidth(), 50

    setAlpha 1.0
    setcolor 255, 255, 255
    textout 10, 10, 24, "YOU: " + board["playerScore"]
    textout screenWidth()-100, 10, 24, "ME: " + board["cpuScore"]
    if board["turn"] = 0 then
        setcolor 255,64,0
        msg = "Your turn!"
    else
        setcolor 0,255,0
        msg = "My turn..."
    endif
    textout 240, 10, 24, msg

    setAlpha a
```

```
endfun



' ----------------------------------------
' A.I.
' ----------------------------------------

' heuristic: from http://www.codeproject.com/KB/game/TicTacToeByMinMax.aspx
' For each row, if there are both X and O, then the score for the row is 0.
' If the whole row is empty, then the score is 1.
' If there is only one X, then the score is 10.
' If there are two Xs, then the score is 100.
' If there are 3 Xs, then the score is 1000, and the winner is Player X.
' For Player O, the score is negative.
' Player X tries to maximize the score.
' Player O tries to minimize the score.
' If the current turn is for Player X, then the score of Player X has
more advantage. I gave the advantage rate as 3.
fun heuristicValueOf(board, whoami)

    ' initializing...
    h = 0
    if whoami = 0 then
        me = "x"
        other = "o"
    else
        me = "o"
        other = "x"
    endif

    ' nice rows, cols or diagonals for me.
    for i=0 to 2
        col = board[3*i] + board[3*i+1] + board[3*i+2]
        if col = "   " then h += 1
```

```
        if col = "  "+me or col = " "+me+" " or col="  "+me then h += 10
        if col = " "+me+me or col = me+" "+me or col = me+me+" " then h += 100
        if col = me+me+me then h += 1000

        row = board[i] + board[i+3] + board[i+6]
        if row = "   " then h += 1
        if row = "  "+me or row = " "+me+" " or row="  "+me then h += 10
        if row = " "+me+me or row = me+" "+me or row = me+me+" " then h += 100
        if row = me+me+me then h += 1000
    next

    diag1 = board[0] + board[4] + board[8]
    if diag1 = "   " then h += 1
    if diag1 = "  "+me or diag1 = " "+me+" " or diag1="  "+me then h += 10
    if diag1 = " "+me+me or diag1 = me+" "+me or diag1 = me+me+" " then h += 100
    if diag1 = me+me+me then h += 1000

    diag2 = board[2] + board[4] + board[6]
    if diag2 = "   " then h += 1
    if diag2 = "  "+me or diag2 = " "+me+" " or diag2="  "+me then h += 10
    if diag2 = " "+me+me or diag2 = me+" "+me or diag2 = me+me+" " then h += 100
    if diag2 = me+me+me then h += 1000

    if board["turn"] = whoami then h *= 3

    ' nice rows, cols or diagonals for my opponent.
    for i=0 to 2
        col = board[3*i] + board[3*i+1] + board[3*i+2]
        if col = "   " then h -= 1
        if col = "  "+other or col = " "+other+" " or col="  "+other then h -= 10
        if col = " "+other+other or col = other+" "+other or col = other+other+" "
then h -= 100
        if col = other+other+other then h -= 1000

        row = board[i] + board[i+3] + board[i+6]
```

```
        if row = "   " then h -= 1
        if row = "  "+other or row = " "+other+" " or row="  "+other then h -= 10
        if row = " "+other+other or row = other+" "+other or row = other+other+" "
then h -= 100
        if row = other+other+other then h -= 1000
    next

    diag1 = board[0] + board[4] + board[8]
    if diag1 = "   " then h -= 1
    if diag1 = "  "+other or diag1 = " "+other+" " or diag1="  "+other then h -= 10
    if diag1 = " "+other+other or diag1 = other+" "+other or
diag1 = other+other+" " then h -= 100
    if diag1 = other+other+other then h -= 1000

    diag2 = board[2] + board[4] + board[6]
    if diag2 = "   " then h -= 1
    if diag2 = "  "+other or diag2 = " "+other+" " or diag2="  "+other then h -= 10
    if diag2 = " "+other+other or diag2 = other+" "+other or
diag2 = other+other+" " then h -= 100
    if diag2 = other+other+other then h -= 1000

    ' done
    return h

endfun


' minimax algorithm
fun minimax(board, depth, whoami)

    if depth > 0 then
        if whoami = 0 then
            ' human player: minimize his/her reward
            value = 99999
            for i=0 to 8
```

```
            if board[i] = " " then
                board[i] = "x"
                m = minimax(board, depth-1, 1-whoami)
                board[i] = " "

                if m < value then
                    value = m
                    best = i
                endif
            endif
        next
    else
        ' CPU: maximize my reward
        value = -99999
        for i=0 to 8
            if board[i] = " " then
                board[i] = "o"
                m = minimax(board, depth-1, 1-whoami)
                board[i] = " "

                if m > value then
                    value = m
                    best = i
                endif
            endif
        next
    endif

    ' full board?
    if abs(value) = 99999 then return heuristicValueOf(board, whoami)

    board["best"] = best
    return value
else
    return heuristicValueOf(board, whoami)
```

```
    endif

endfun

' ----------------------------------------
' GAME
' ----------------------------------------
b = createBoard()
gamemode

do
    cls

    ' logic
    if gotWinner(b) <> " " then
        winner = gotWinner(b)
        if winner = "x" then
            if confirm("You win! Play again?") then
                resetBoard b, winner
            else
                exit
            endif
        elseif winner = "o" then
            if confirm("You lose. Play again?") then
                resetBoard b, winner
            else
                exit
            endif
        elseif winner = "d" then
            if confirm("A draw. Play again?") then
                resetBoard b, winner
            else
                exit
            endif
        endif
```

```
    endif
    processTurn b

    ' render
    drawboard b
    drawhud b

    flip
loop
```

4.4.5 Galaxy Wars

Neste jogo de ação, o herói super poderoso deve enfrentar as naves alienígenas, os robôs assassinos e os terríveis monstros do espaço!

Figura 4.8: é muita emoção!

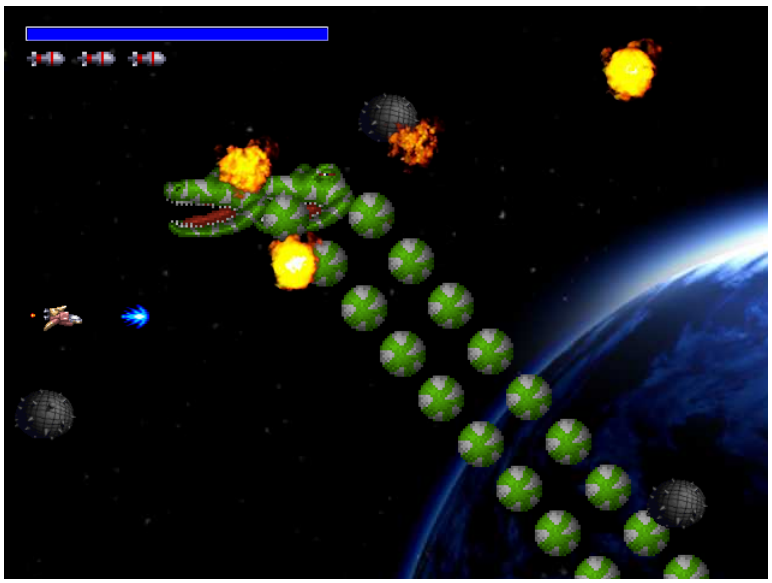


Figura 4.9: as poderosas cobras malignas!

Figura 4.10: o terrível chefão!

Confira o código-fonte:[14]

[14]Levei cerca de quatro dias para criar o game.

```
'                .__
'    _________   |  | _____  ___  ______.__. __  _  _______  _______ ______
'   / ___\__  \  |  | \__  \ \  \/  <   |  | \ \/ \/ /\__  \ \_  __ |  ___/
'  / /_/  > __ \_|  |__/ __ \_>    < \___  |  \     /  / __ \_|  | \|___ \
'  \___  (____  /|____(____  /__/\_ \/ ____|   \/\_/  (____  /|__| /____  >
' /_____/     \/           \/      \/\/                    \/           \/
'
' Copyright (C) 2011 by Alexandre Martins <alemartf at gmail.com>
'
' Permission is hereby granted, free of charge, to any person obtaining a copy
' of this software and associated documentation files (the "Software"), to deal
' in the Software without restriction, including without limitation the rights
' to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
' copies of the Software, and to permit persons to whom the Software is
' furnished to do so, subject to the following conditions:
'
' The above copyright notice and this permission notice shall be included in
' all copies or substantial portions of the Software.
'
' THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
' IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
' FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
' AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
' LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
' OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
' THE SOFTWARE.
'
'
' [[ HOW TO PLAY ]]
'
' 1. Z:      shoot;
' 2. X:      bomb;
' 3. Arrows: move.
'
'
```

```
' [[ GAME ASSETS ]]
'
' References:
' 1. opengameart.org
' 2. lostgarden.com
' 3. widgetworx.com (SpriteLib)
'
' basic_enemy.png, brain.png by Daniel Cook from Lostgarden.com (cc-by-sa 3.0)
' player.png, projectile*.png, fire.png, bomb.png by surt (cc-by-sa 3.0)
' explosion.png by Charlie (public domain)
' asteroids.png, dino.png by Ari Feldman (cpl 1.0)
' vaerionii.ogg, danger.ogg, boss*.ogg by Gobusto (cc-by-sa 3.0)
' galaxy background by hc (cc-by-sa 3.0)
' projectile*.ogg, explosion*.ogg, tesla_tower.ogg, gravity_bomb.ogg,
' wolfman.ogg, volcano_eruption.ogg by Stephen M. Cameron (cc-by-sa 3.0)
' bold.png, jug*.png by Clint Bellanger (public domain)
' bg.jpg by Luke.RUSTLTD (public domain)
' yeah.ogg by Blender Foundation (cc-by 3.0)
' charge.ogg by Alexandre (cc-by-sa 3.0)
'



' ---------- // CODE // --------------



' enables game mode: double buffering, 60 fps
gameMode

' -- entities --
g_spaceobjects = createTable() ' space objects
g_projectiles = createTable() ' projectiles
g_enemies = createTable() ' enemies
g_player = createPlayer() ' the player
```

```
g_hud = createHud() ' the hud (heads-up display)

' -- music --
g_music = playMusicEx("vaerionii.ogg", -1)

' -- main loop --
do
    cls
    updateLogic
    renderGame
    flip
loop

fun updateLogic()
    updateBackground
    updatePlayer g_player
    updateEnemies
    updateProjectiles
    updateSpaceObjects
    updateHud g_hud
endfun

fun renderGame()
    renderBackground
    renderPlayer g_player
    renderEnemies
    renderProjectiles
    renderSpaceObjects
    renderHud g_hud
endfun

' -- bye! --
destroyHud g_hud
destroyPlayer g_player
destroyTable g_projectiles
```

```
destroyTable g_enemies
destroyTable g_spaceObjects




' ---------- // BACKGROUND // --------------
fun updateBackground()
    g_backgroundX -= 0.15
    if g_backgroundX <= -imageWidth("bg.jpg") then g_backgroundX = 0
endfun

fun renderBackground()
    drawImage "bg.jpg", g_backgroundX, -200
    drawImage "bg.jpg", g_backgroundX + imageWidth("bg.jpg"), -200
endfun







' ---------- // PLAYER ENTITY // --------------

fun createPlayer()
    player = createTable()

    player["hp"] = 100
    player["actor"] = createActor()
    player["bulletTimer"] = createTimer(200)
    player["blinkStart"] = 0
    player["blinkTimer"] = createTimer(100)
    player["visible"] = true
    player["blinking"] = false
```

```
    player["explodeTimer"] = createTimer(125)
    player["bombs"] = 3
    player["victory"] = false

    addActorAnimationEx player["actor"], "stopped", "player.png", 9, 1, 4, 1, 8, tr
    setActorAnimation player["actor"], "stopped"
    setActorPosition player["actor"], 30, 240
    setActorAnchor player["actor"], 24, 22

    spawnSpaceObject "fire", 0, 0

    return player
endfun

fun destroyPlayer(player)
    destroyTimer player["explodeTimer"]
    destroyTimer player["blinkTimer"]
    destroyTimer player["bulletTimer"]
    destroyActor player["actor"]
    destroyTable player
    return 0
endfun

fun updatePlayer(player)
    actor = player["actor"]
    animateActor actor

    ' dead player?
    if player["hp"] <= 0 then
        player["hp"] = 0
        player["blinking"] = false
        if timerHasExpired(player["explodeTimer"]) then
            resetTimer player["explodeTimer"]
            explode actorX(actor) - actorXAnchor(actor) + random(0, actorWidth(acto
        endif
```

```
    moveActor actor, 0, 5
    if actorY(actor) >= 500 then
        blinkPlayer player
        setActorPosition actor, 30, 240
        player["hp"] = 100
    endif
    return
endif

' winner?
if player["victory"] then
    moveActor actor, 10, 0
    return
endif

' move horizontally
if keyDown(g_keyRight) then moveActor actor, 5, 0
if keyDown(g_keyLeft) then moveActor actor, -5, 0
if actorX(actor) < 50 then setActorPosition actor, 50, actorY(actor)
if actorX(actor) > 490 then setActorPosition actor, 490, actorY(actor)

' move vertically
if keyDown(g_keyDown) then moveActor actor, 0, 10
if keyDown(g_keyUp) then moveActor actor, 0, -10
if actorY(actor) < 30 then setActorPosition actor, actorX(actor), 30
if actorY(actor) > 450 then setActorPosition actor, actorX(actor), 450

' shoot
if keyHit(g_keyZ) or (keyDown(g_keyZ) and timerHasExpired( player["bulletTimer"
    shoot "ship1", player, actorX(actor)+20, actorY(actor), 1, 0
    resetTimer player["bulletTimer"]
endif

' bomb
if keyHit(g_keyX) and player["bombs"] > 0 then
```

```
        spawnSpaceObject "bomb", actorX(actor)+20, actorY(actor)
        player["bombs"] -= 1
    endif

    ' blinking?
    if player["blinking"] then
        if timerHasExpired(player["blinkTimer"]) then
            player["visible"] = not( player["visible"] )
            resetTimer(player["blinkTimer"])
            if milliseconds() > player["blinkStart"] + 3000 then
                player["blinking"] = false
                player["visible"] = true
            endif
        endif
    endif
endfun

fun renderPlayer(player)
    if player["visible"] then drawActor player["actor"]
endfun

fun playerActor(player)
    return player["actor"]
endfun

fun damagePlayer(player, damage)
    if not player["blinking"] and player["hp"] > 0 then
        player["hp"] -= damage
        if player["hp"] < 0 then player["hp"] = 0
        blinkPlayer player
        return true
    endif

    return false
endfun
```

```
fun blinkPlayer(player)
    player["blinkStart"] = milliseconds()
    player["blinking"] = true
    resetTimer player["blinkTimer"]
endfun

fun playerEnableVictoryMode(player)
    player["victory"] = true
endfun




' ---------- // PROJECTILES MANAGEMENT // --------------

' shoots a new projectile
' projectile types: "default", "missile"
' owner is the ship who created the projectile
' (xpos, ypos) is the initial position
' (direction_x, direction_y) is a non-zero vector
fun shoot(type, owner, xpos, ypos, direction_x, direction_y)
    ' adds a new projectile to the g_projectiles table
    c = g_projectileCounter
    g_projectileCounter += 1
    g_projectiles[c] = createProjectile(type, owner, xpos, ypos, direction_x, direc
endfun

' updates the projectiles list
fun updateProjectiles()
    for i in g_projectiles
        p = g_projectiles[i]
```

```
        updateProjectile p
        if p["state"] = "dead" then
            destroyProjectile p
            removeTableEntry g_projectiles, i
        endif
    next
endfun

' renders the projectiles
fun renderProjectiles()
    for i in g_projectiles
        renderProjectile g_projectiles[i]
    next
endfun




' ---------- // PROJECTILE ENTITY // --------------

fun createProjectile(type, owner, xpos, ypos, direction_x, direction_y)
    proj = createTable()
    snd = ""

    proj["type"] = type
    proj["state"] = "alive"
    proj["dirx"] = direction_x / sqrt( direction_x^2 + direction_y^2 )
    proj["diry"] = direction_y / sqrt( direction_x^2 + direction_y^2 )
    proj["owner"] = owner
    proj["actor"] = createActor()
    setActorPosition proj["actor"], xpos, ypos

    if type = "default" then
        snd = "projectile1.ogg"
        proj["speed"] = 20
```

```
        proj["damage"] = 2
        addActorAnimation proj["actor"], "stopped", "projectile1.png", 3, 1, 0.4, f
        setActorAnimation proj["actor"], "stopped"
        setActorAnchor proj["actor"], actorWidth(proj["actor"])/2, actorHeight(proj
    elseif type = "default2" then
        snd = "projectile1.ogg"
        proj["speed"] = 20
        proj["damage"] = 5
        addActorAnimation proj["actor"], "stopped", "projectile2.png", 3, 1, 0.4, f
        setActorAnimation proj["actor"], "stopped"
        setActorAnchor proj["actor"], actorWidth(proj["actor"])/2, actorHeight(proj
    elseif type = "default3" then
        snd = "projectile2.ogg"
        proj["speed"] = 20
        proj["damage"] = 10
        addActorAnimation proj["actor"], "stopped", "projectile4.png", 3, 1, 0.4, f
        setActorAnimation proj["actor"], "stopped"
        setActorAnchor proj["actor"], actorWidth(proj["actor"])/2, actorHeight(proj
    elseif type = "ship1" then
        snd = "projectile1.ogg"
        proj["speed"] = 20
        proj["damage"] = 5
        addActorAnimation proj["actor"], "stopped", "projectile3.png", 1, 2, 0.4, t
        setActorAnimation proj["actor"], "stopped"
        setActorAnchor proj["actor"], actorWidth(proj["actor"])/2, actorHeight(proj
    else
        fatalError "Unknown projectile: " + type
    endif

    playSample snd

    return proj
endfun

fun destroyProjectile(projectile)
```

```
    destroyActor projectile["actor"]
    destroyTable projectile
    return 0
endfun

fun updateProjectile(projectile)
    ' move & animate
    actor = projectile["actor"]
    dx = projectile["speed"] * projectile["dirx"]
    dy = projectile["speed"] * projectile["diry"]
    moveActor actor, dx, dy
    animateActor actor

    ' collided to an enemy?
    owner = projectile["owner"]
    if owner = g_player then
        ' enemy = enemy of the player
        for i in g_enemies
            e = g_enemies[i]
            if actorCollision(actor, e["actor"]) then
                explode actorX(actor), actorY(actor)
                e["hp"] -= projectile["damage"]
                projectile["state"] = "dead"
            endif
        next
    else
        ' enemy = player
        if actorCollision(actor, playerActor(g_player)) then
            if damagePlayer(g_player, projectile["damage"]) then
                explode actorX(actor), actorY(actor)
                projectile["state"] = "dead"
            endif
        endif
    endif
```

```
    ' left the screen?
    if actorX(actor) > screenWidth() + 100 then projectile["state"] = "dead"
    if actorX(actor) < -100 then projectile["state"] = "dead"
endfun

fun renderProjectile(projectile)
    ang = atan2( projectile["diry"], projectile["dirx"] ) ' the correct angle
    drawActorEx projectile["actor"], 1, 1, ang ' draw it!
endfun









' ------------- // ENEMIES MANAGEMENT // ----------------

fun numberOfEnemies()
    return tableSize(g_enemies)
endfun

fun spawnEnemy(type, xpos, ypos)
    return spawnEnemyEx(type, xpos, ypos, 0)
endfun

fun spawnEnemyEx(type, xpos, ypos, options)
    c = g_enemyCounter
    g_enemyCounter += 1
    g_enemies[c] = createEnemy(type, xpos, ypos, options)
    return g_enemies[c]
```

```
endfun

fun updateEnemies()
    updateEnemySpawner
    for i in g_enemies
        p = g_enemies[i]
        updateEnemy p
        if p["state"] = "dead" then
            destroyEnemy p
            removeTableEntry g_enemies, i
        endif
    next
endfun

fun renderEnemies()
    for i in g_enemies
        renderEnemy g_enemies[i]
    next
endfun

' this will spawn enemies the way we want
g_ess = 0 ' enemy spawner step
g_est = createTimer(1) ' enemy spawner timer
fun updateEnemySpawner()

    if g_ess >= 0 and g_ess < 1 then

        ' STEP 1: get ready!
        if g_ess = 0 then
            setTimerInterval g_est, 4000
            displayBlinkingText "GET READY!"
            g_ess += 0.1
        elseif timerHasExpired(g_est) then
            g_ess = 1 ' <--- first step!
            setTimerInterval g_est, 500
```

```
    endif

elseif g_ess >= 1 and g_ess < 2 then

    ' STEP 2: spawn basic enemies
    if timerHasExpired(g_est) then
        spawnEnemyEx "basic", 680, 40 + (g_ess - 1) * 400, 0.015
        g_ess += 0.1
        resetTimer g_est
        if g_ess >= 2 then setTimerInterval g_est, 500
    endif

elseif g_ess >= 2 and g_ess < 3 then

    ' STEP 3: wait
    if timerHasExpired(g_est) and numberOfEnemies() = 0 then
        g_ess = 3
        setTimerInterval g_est, 500
    endif

elseif g_ess >= 3 and g_ess < 4 then

    ' STEP 4: spawn faster basic enemies
    if timerHasExpired(g_est) then
        spawnEnemyEx "basic", 680, 400 * (4 - g_ess), 0.03
        g_ess += 0.1
        resetTimer g_est
        if g_ess >= 4 then
            setTimerInterval g_est, 5000
            g_ess = 4
        endif
    endif

elseif g_ess >= 4 and g_ess < 5 then
```

```
    ' STEP 5: spikey things
    if ((g_ess > 4) or (g_ess = 4 and numberOfEnemies() = 0)) and timerHasExpir
        for i=0 to floor(30 * (g_ess - 3.9))
            spawnEnemy "bold", 680+random(0,50), random(0, screenHeight())
        next
        resetTimer g_est
        g_ess += 0.34

        if g_ess >= 5 then
            g_ess = 5
            setTimerInterval g_est, 1500
        endif
    endif

elseif g_ess >= 5 and g_ess < 6 then

    ' STEP 6: robots
    if ((g_ess = 5 and numberOfEnemies() = 0) or (g_ess > 5)) and timerHasExpir
        spawnEnemyEx "jug", 600 - (g_ess - 5.5)^4 * 200, 480, 40 + (g_ess - 5)
        g_ess += 0.15
        resetTimer g_est
    endif

elseif g_ess >= 6 and g_ess < 7 then

    ' STEP 7: wait a bit
    if numberOfEnemies() = 0 then
        setTimerInterval g_est, 5000
        g_ess = 7
    endif

elseif g_ess >= 7 and g_ess < 8 then

    ' STEP 8: watch out!
    if g_ess = 7 then
```

```
        if timerHasExpired(g_est) then
            displayBlinkingText "WATCH OUT!"
            setTimerInterval g_est, 2000
            g_ess = 7.5
        endif
    else
        if timerHasExpired(g_est) then
            playSample "scream.ogg"
            setTimerInterval g_est, 1500
            g_ess = 8
        endif
    endif

elseif g_ess >= 8 and g_ess < 9 then

    ' STEP 9: make the green monsters appear!
    if timerHasExpired(g_est) then
        if g_ess = 8 or g_ess = 8.5 then
            x = 480 - (g_ess - 8) * 150
            g_dino = spawnEnemy("dino", x, 520)
            for i=0 to 10
                explode 420+16*i, 450
            next
            for i=0 to 12
                spawnEnemyEx "asteroid", x, 480, (1+i) * 12
            next
            setTimerInterval g_est, 10000
            g_ess += 0.25
        elseif g_ess = 8.25 then
            playSample "scream.ogg"
            setTimerInterval g_est, 1500
            g_ess += 0.25
        else
            g_ess = 9
            setTimerInterval g_est, 1000
```

```
        endif
    endif
elseif g_ess >= 9 and g_ess < 10 then

    ' STEP 10: kill the dinos
    isDinoAlive? = isValidTable(g_dino)
    if timerHasExpired(g_est) and isDinoAlive? then
        resetTimer(g_est)
        spawnEnemy "bold", 680+random(0,50), random(0, screenHeight())
    elseif not(isDinoAlive?) and numberOfEnemies() = 0 then
        g_ess = 10
        setTimerInterval g_est, 2000
    endif

elseif g_ess >= 10 and g_ess < 11 then

    ' STEP 11: wait...
    if timerHasExpired(g_est) then
        g_ess = 11
        setTimerInterval g_est, 1000
    endif

elseif g_ess >= 11 and g_ess < 12 then

    ' STEP 12: a LOT of stronger robots!
    if timerHasExpired(g_est) then
        spawnEnemy "jug2", 680, random(0, screenHeight())
        g_ess += 0.1
        resetTimer g_est
        if g_ess >= 12 then setTimerInterval g_est, 7000
    endif

elseif g_ess >= 12 and g_ess < 13 then

    ' STEP 13: wait...
```

```
    if timerHasExpired(g_est) then
        if numberOfEnemies() = 0 then
            g_ess = 13
            playSample "danger.ogg"
            displayBlinkingText "DANGER!"
            setTimerInterval g_est, 5000
        else
            resetTimer g_est
        endif
    endif

elseif g_ess >= 13 and g_ess < 14 then

    ' STEP 14: spawn the boss
    if timerHasExpired(g_est) then
        g_brain = spawnEnemy("brain", 770, 240)
        setTimerInterval g_est, 1000
        g_ess = 14
    endif

elseif g_ess >= 14 and g_ess < 15 then

    ' STEP 15: defeat the boss
    if not isValidTable(g_brain) then
        setTimerInterval g_est, 5000
        g_ess = 15
        for k in g_enemies
            enemy = g_enemies[k]
            if enemy["type"] = "asteroid" then
                explode actorX(enemy["actor"]), actorY(enemy["actor"])
                enemy["hp"] = 0
            endif
        next
    elseif timerHasExpired(g_est) then
        spawnEnemyEx "asteroid", 680, random(0, screenHeight()), random(135, 22
```

```
            resetTimer g_est
        endif

    elseif g_ess >= 15 and g_ess < 16 then

        ' STEP 16: victory!!!
        if timerHasExpired(g_est) then
            g_ess = 16
        endif

    elseif g_ess >= 16 and g_ess < 17 then

        ' STEP 17: end
        setTimerInterval g_est, 3000
        playerEnableVictoryMode(g_player)
        g_ess = 17

    elseif g_ess >= 17 and g_ess < 18 then

        ' STEP 18: level cleared
        if timerHasExpired(g_est) then
            spawnSpaceObjectEx "levelcleared", 0, 0, g_music
            g_ess = 18
        endif

    elseif g_ess >= 18 then

        ' done!

    endif

endfun
```

```
' ----------------------- // ENEMY ENTITY // ---------------------

fun createEnemy(type, xpos, ypos, options)
    enemy = createTable()
    enemy["type"] = type
    enemy["state"] = "alive"
    enemy["actor"] = createActor()
    enemy["collisionDamage"] = 10
    enemy["dieOnCollision"] = true
    enemy["hflip"] = false
    enemy["bulletTimer"] = createTimer(500)
    enemy["spawnXPos"] = xpos
    enemy["spawnYPos"] = ypos
    setActorPosition enemy["actor"], xpos, ypos

    if type = "basic" then
        ' creating a basic enemy
        setTimerInterval enemy["bulletTimer"], 500 + random(0,1500)
        enemy["hp"] = 1
        enemy["collisionDamage"] = 10
        enemy["speedMultiplier"] = max(0.015, options)
        addActorAnimation enemy["actor"], "stopped", "basic_enemy.png", 1, 5, 0.4,
        setActorAnimation enemy["actor"], "stopped"
        setActorAnchor enemy["actor"], actorWidth(enemy["actor"])/2, actorHeight(en
    elseif type = "bold" then
        ' creating that bold, spikey, floating thing
        enemy["hp"] = 5
        enemy["collisionDamage"] = 10
        enemy["speed"] = 2+random(0,3)
        enemy["dirx"] = (-10-random(0,10))
        enemy["diry"] = -5+random(0,10)
        norm = sqrt(enemy["dirx"]^2 + enemy["diry"]^2)
        enemy["dirx"] /= norm
        enemy["diry"] /= norm
        addActorAnimationEx enemy["actor"], "stopped", "bold.png", 3, 2, 0, 4, 0.1,
```

```
    setActorAnimation enemy["actor"], "stopped"
    setActorAnchor enemy["actor"], actorWidth(enemy["actor"])/2, actorHeight(en
elseif type = "jug" then
    ' robots
    setTimerInterval enemy["bulletTimer"], 250 + random(0,500)
    enemy["hp"] = 25
    enemy["collisionDamage"] = 20
    enemy["hflip"] = true
    enemy["dest_y"] = options
    addActorAnimationEx enemy["actor"], "appearing", "jug.png", 6, 4, 21, 1, 1,
    addActorAnimationEx enemy["actor"], "shooting", "jug.png", 6, 4, 5, 3, 0.1,
    setActorAnimation enemy["actor"], "appearing"
    setActorAnchor enemy["actor"], actorWidth(enemy["actor"])/2, actorHeight(en
elseif type = "jug2" then
    ' stronger robots
    setTimerInterval enemy["bulletTimer"], 250 + random(0,500)
    enemy["hp"] = 50
    enemy["collisionDamage"] = 20
    enemy["hflip"] = true
    g_jug2Id += 1
    image = "jug" + toString(2 + g_jug2Id mod 2) + ".png"
    addActorAnimationEx enemy["actor"], "idle", image, 6, 4, 20, 2, 0.1, true
    addActorAnimationEx enemy["actor"], "shooting", image, 6, 4, 5, 3, 0.1, fal
    setActorAnimation enemy["actor"], "idle"
    setActorAnchor enemy["actor"], actorWidth(enemy["actor"])/2, actorHeight(en
elseif type = "asteroid" then
    ' asteroid
    enemy["hp"] = 30
    enemy["collisionDamage"] = 30
    enemy["angle"] = options
    enemy["speed"] = 1+random(0,2)
    if random(0, 1) = 0 then
        addActorAnimationEx enemy["actor"], "idle", "asteroids.png", 8, 2, 0, 8
    else
        addActorAnimationEx enemy["actor"], "idle", "asteroids.png", 8, 2, 8, 8
```

```
    endif
    setActorAnimation enemy["actor"], "idle"
    setActorAnchor enemy["actor"], actorWidth(enemy["actor"])/2, actorHeight(en
elseif type = "dino" then
    ' dino
    setTimerInterval enemy["bulletTimer"], 2000
    enemy["hp"] = 200
    enemy["collisionDamage"] = 50
    enemy["dieOnCollision"] = false
    addActorAnimationEx enemy["actor"], "idle", "dino.png", 4, 1, 0, 2, 0.05, t
    addActorAnimationEx enemy["actor"], "bite", "dino.png", 4, 1, 2, 1, 0.25, t
    setActorAnimation enemy["actor"], "idle"
    setActorAnchor enemy["actor"], 84, 46
    for i=1 to 10
        spawnEnemyEx "dinoBody", xpos, ypos, enemy
    next
elseif type = "dinoBody" then
    ' the body of the dino
    enemy["hp"] = 9999999
    enemy["collisionDamage"] = 10
    enemy["dieOnCollision"] = false
    enemy["parent"] = options ' the head of the dino
    enemy["id"] = g_dinoBodyId mod 10 ' my unique identification number
    g_dinoBodyId += 1
    addActorAnimationEx enemy["actor"], "idle", "dino.png", 4, 1, 3, 1, 1, true
    setActorAnimation enemy["actor"], "idle"
    setActorAnchor enemy["actor"], actorWidth(enemy["actor"])/2, actorHeight(en
elseif type = "brain" then
    ' boss
    g_music = playMusicEx("boss.ogg", -1)
    setTimerInterval enemy["bulletTimer"], 3000
    enemy["hp"] = 700
    enemy["collisionDamage"] = 30
    enemy["dieOnCollision"] = false
    addActorAnimationEx enemy["actor"], "idle", "brain.png", 2, 1, 0, 1, 0.1, t
```

```
        addActorAnimationEx enemy["actor"], "charging", "brain.png", 2, 1, 0, 2, 0.
        setActorAnimation enemy["actor"], "idle"
        setActorAnchor enemy["actor"], actorWidth(enemy["actor"])/2, actorHeight(en
    elseif type = "defeatedBrain" then
        ' defeated boss
        setTimerInterval enemy["bulletTimer"], 125
        enemy["hp"] = 99999
        enemy["collisionDamage"] = 30
        enemy["dieOnCollision"] = false
        addActorAnimationEx enemy["actor"], "idle", "brain.png", 2, 1, 0, 1, 0.1, t
        setActorAnimation enemy["actor"], "idle"
        setActorAnchor enemy["actor"], actorWidth(enemy["actor"])/2, actorHeight(en
    endif

    return enemy
endfun

fun destroyEnemy(enemy)
    destroyTimer enemy["bulletTimer"]
    destroyActor enemy["actor"]
    destroyTable enemy
    return 0
endfun

fun updateEnemy(enemy)
    ' stuff
    type = enemy["type"]
    actor = enemy["actor"]
    x = actorX(actor)
    y = actorY(actor)

    ' animate the actor
    animateActor actor

    ' each enemy type has a behavior
```

```
if type = "basic" then

    ' >> BASIC ENEMY <<
    lim = 320 + 0.003 * (y - 240)^2
    if x > 30 + lim then x -= enemy["speedMultiplier"] * (x - lim)
    if timerHasExpired(enemy["bulletTimer"]) then
        shoot "default", enemy, x-10, y, -1, 0
        resetTimer enemy["bulletTimer"]
    endif

elseif type = "bold" then

    ' >> BOLD, SPIKEY ENEMY <<
    x += enemy["dirx"] * enemy["speed"]
    y += enemy["diry"] * enemy["speed"]

elseif type = "jug" then

    ' >> ROBOT <<
    if y > enemy["dest_y"] then y -= (y - enemy["dest_y"]) * 0.02
    if timerHasExpired(enemy["bulletTimer"]) then
        setActorAnimation actor, "appearing"
        setActorAnimation actor, "shooting"
        shoot "default2", enemy, x-10, y-2, -1, 0
        shoot "default2", enemy, x-10, y-2, -2, -1
        shoot "default2", enemy, x-10, y-2, -2, 1
        resetTimer enemy["bulletTimer"]
    endif

elseif type = "asteroid" then

    ' >> ASTEROID <<
    x += cos(enemy["angle"]) * enemy["speed"]
    y += -sin(enemy["angle"]) * enemy["speed"]
```

```
elseif type = "dino" then

    ' >> DINO <<
    if enemy["state"] = "alive" then

        setActorAnimation actor, "bite"
        dest_x = enemy["spawnXPos"]
        dest_y = 50
        speed = 4
        if timerHasExpired(enemy["bulletTimer"]) then
            setTimerInterval enemy["bulletTimer"], 5000
            enemy["state"] = "waiting"
        endif

    elseif enemy["state"] = "waiting" then

        setActorAnimation actor, "idle"
        dest_x = enemy["spawnXPos"]
        dest_y = actorY(playerActor(g_player)) + 50 * cos(0.180 * milliseconds(
        speed = 2
        if timerHasExpired(enemy["bulletTimer"]) then
            enemy["state"] = "hunting"
            setTimerInterval enemy["bulletTimer"], 1000
            playSample "wolfman.ogg"
        endif

    elseif enemy["state"] = "hunting" then

        setActorAnimation actor, "bite"
        dest_x = actorX(playerActor(g_player))
        dest_y = actorY(playerActor(g_player))
        speed = 6
        if timerHasExpired(enemy["bulletTimer"]) then
            setTimerInterval enemy["bulletTimer"], 5000
            enemy["state"] = "waiting"
```

```
        endif

    endif

    dx = dest_x - x
    dy = dest_y - y
    norm = sqrt(dx^2 + dy^2)
    if norm > 0 then
        dx /= norm
        dy /= norm
        x += dx * speed
        y += dy * speed
    endif

elseif type = "dinoBody" then

    ' >> DINO BODY <<
    if enemy["state"] = "alive" then
        dino = enemy["parent"]
        if isValidTable(dino) then ' the dino hasn't been killed yet
            sx = dino["spawnXPos"] + 180 + 20 * cos(0.180 * milliseconds())
            sy = dino["spawnYPos"] + 20 * cos(0.180 * milliseconds())
            dx = actorX(dino["actor"]) + 65
            dy = actorY(dino["actor"]) - 10
            alpha = enemy["id"] / 10
            x = alpha * sx + (1 - alpha) * dx
            y = alpha * sy + (1 - alpha) * dy
        else
            enemy["state"] = "dying"
        endif
    elseif enemy["state"] = "dying" then
        y += y * 0.03
    endif

elseif type = "jug2" then
```

```
' >> STRONGER ROBOT <<
if enemy["state"] = "alive" then
    enemy["state"] = "choose_dest"
elseif enemy["state"] = "choose_dest" then
    setActorAnimation actor, "idle"
    dx = enemy["dest_x"]
    dy = enemy["dest_y"]
    while (enemy["dest_x"] - dx)^2 + (enemy["dest_y"] - dy)^2 < 100^2
        enemy["dest_x"] = random(actorX(playerActor(g_player)) + 150, scree
        enemy["dest_y"] = random(50, screenHeight() - 50)
    wend
    enemy["state"] = "follow_dest"
    enemy["bullets"] = 5
    setTimerInterval enemy["bulletTimer"], random(1500, 3000)
elseif enemy["state"] = "follow_dest" then
    dx = enemy["dest_x"] - x
    dy = enemy["dest_y"] - y
    norm = sqrt(dx^2 + dy^2)
    dx /= norm
    dy /= norm
    speed = 5

    if norm > 5 then
        x += speed*dx
        y += speed*dy
    endif

    if timerHasExpired(enemy["bulletTimer"]) then
        setTimerInterval enemy["bulletTimer"], 200
        enemy["state"] = "shooting"
    endif
elseif enemy["state"] = "shooting" then
    if timerHasExpired(enemy["bulletTimer"]) then
        setActorAnimation actor, "idle"
```

```
            setActorAnimation actor, "shooting"
            shoot "default3", enemy, x-10, y-2, actorX(playerActor(g_player)) -
            resetTimer enemy["bulletTimer"]

            enemy["bullets"] -= 1
            if enemy["bullets"] <= 0 then enemy["state"] = "choose_dest"
        endif
    endif

    ' look to the player
    enemy["hflip"] = (x > actorX(playerActor(g_player)))

elseif type = "brain" then

    ' >> BRAIN <<
    if x >= screenWidth() * 0.8 then x -= 2.25
    if y < actorY(playerActor(g_player)) - 10 then y += 1
    if y > actorY(playerActor(g_player)) + 10 then y -= 1

    if enemy["state"] = "alive" then
        setHudBoss g_hud, enemy
        enemy["state"] = "waiting"
    elseif enemy["state"] = "waiting" then
        setActorAnimation actor, "idle"
        if timerHasExpired(enemy["bulletTimer"]) then
            setTimerInterval enemy["bulletTimer"], 2000
            enemy["state"] = "charging"
            playSample "charge.ogg"
        endif
    elseif enemy["state"] = "charging" then
        setActorAnimation actor, "charging"
        if timerHasExpired(enemy["bulletTimer"]) then
            spawnSpaceObjectEx "brainLaser", actorX(actor), actorY(actor), enem
            playSample "tesla_tower.ogg"
            setTimerInterval enemy["bulletTimer"], 5000 - (700 - enemy["hp"])*9
```

```
                enemy["state"] = "waiting"
            endif
        endif

        if enemy["hp"] <= 100 then
            enemy["hp"] = 0
            spawnEnemy "defeatedBrain", x, y
        endif

    elseif type = "defeatedBrain" then

        y += 1
        if timerHasExpired(enemy["bulletTimer"]) then
            explode x - actorXAnchor(actor) + random(0, actorWidth(actor)), y - act
            resetTimer enemy["bulletTimer"]
        endif

    endif

    ' move the actor
    setActorPosition actor, x, y

    ' collided to the player?
    if actorCollision(actor, playerActor(g_player)) then
        if damagePlayer(g_player, enemy["collisionDamage"]) then
            explode actorX(playerActor(g_player)), actorY(playerActor(g_player))
            if enemy["dieOnCollision"] then enemy["state"] = "dead"
        endif
    endif

    ' left the screen or took too much damage?
    if (enemy["hp"] <= 0 or x > 850 or x < -100 or y > 580 or y < -100) then enemy[
endfun

fun renderEnemy(enemy)
```

```
    if enemy["hflip"] then
        drawActorEx enemy["actor"], -1, 1, 0
    else
        drawActor enemy["actor"]
    endif
endfun









' ---------- // SPACE OBJECTS MANAGEMENT // --------------
fun spawnSpaceObject(type, x, y)
    return spawnSpaceObjectEx(type, x, y, 0)
endfun

fun spawnSpaceObjectEx(type, x, y, options)
    c = g_soId
    g_soId += 1
    g_spaceObjects[c] = createSpaceObjectEx(type, x, y, options)
    return g_spaceObjects[c]
endfun

fun updateSpaceObjects()
    for i in g_spaceObjects
        p = g_spaceObjects[i]
        updateSpaceObject p
        if p["state"] = "dead" then
            destroySpaceObject p
            removeTableEntry g_spaceObjects, i
```

```
        endif
    next
endfun

fun renderSpaceObjects()
    for i in g_spaceObjects
        renderSpaceObject g_spaceObjects[i]
    next
endfun

' creates an explosion at the given position
fun explode(x, y)
    return spawnSpaceObject("explosion", x, y)
endfun

' a super flashbang!
fun flashbang()
    return spawnSpaceObject("flashbang", 0, 0)
endfun

' display a blinking text
fun displayBlinkingText(text)
    return spawnSpaceObjectEx("text", 0, 0, text)
endfun



' ------------- // SPACE OBJECTS // ---------------

fun createSpaceObject(type, xpos, ypos)
    return createSpaceObjectEx(type, xpos, ypos, 0)
endfun

fun createSpaceObjectEx(type, xpos, ypos, options)
    so = createTable()
```

```
so["type"] = type
so["timer"] = createTimer(1000)
so["actor"] = createActor()
so["state"] = "alive"
so["spawnXPos"] = xpos
so["spawnYPos"] = ypos
so["visible"] = true

if type = "explosion" then
    ' an explosion
    playSample "explosion1.ogg"
    act = so["actor"]
    setActorPosition act, xpos, ypos
    addActorAnimation act, "idle", "explosion.png", 4, 4, 0.3, false
    setActorAnchor act, 16, 16
    setActorAnimation act, "idle"
elseif type = "fire" then
    ' a small fire
    act = so["actor"]
    setActorPosition act, xpos, ypos
    addActorAnimation act, "idle", "fire.png", 3, 1, 0.3, true
    setActorAnchor act, 30, 6
    setActorAnimation act, "idle"
elseif type = "bomb" then
    ' super bomb
    setTimerInterval so["timer"], 500
    act = so["actor"]
    setActorPosition act, xpos, ypos
    addActorAnimation act, "idle", "bomb.png", 1, 1, 0.3, true
    setActorAnchor act, 16, 6
    setActorAnimation act, "idle"
    playSample "gravity_bomb.ogg" ' cool stuff!!!
elseif type = "flashbang" then
    ' flashbang
    setTimerInterval so["timer"], 250
```

```
        playSample "volcano_eruption.ogg"
        so["alpha"] = 1.0
    elseif type = "brainLaser" then
        ' brain laser
        act = so["actor"]
        setActorPosition act, xpos, ypos
        so["alpha"] = 1.0
        so["height"] = 50
        so["parent"] = options
    elseif type = "text" then
        ' text
        setTimerInterval so["timer"], 200
        so["text"] = options
        so["loops"] = 15
    elseif type = "levelcleared" then
        ' level clared
        so["alpha"] = 0.0
        so["music"] = options
        so["score"] = max(120*60*1000 - milliseconds(), 100) ' 5 minutes max.
        hideHud g_hud
    endif

    return so
endfun

fun destroySpaceObject(so)
    destroyTimer so["timer"]
    destroyActor so["actor"]
    destroyTable so
    return 0
endfun

fun updateSpaceObject(so)
    type = so["type"]
    act = so["actor"]
```

```
tmr = so["timer"]
x = actorX(act)
y = actorY(act)

if type = "explosion" then
    ' explosion
    animateActor act
    if actorAnimationFinished(act) then so["state"] = "dead"
elseif type = "fire" then
    ' small fire
    animateActor act
    px = actorX(playerActor(g_player))
    py = actorY(playerActor(g_player))
    setActorPosition act, px, py-1
    so["visible"] = g_player["visible"] and not keyDown(g_keyLeft)
elseif type = "bomb" then
    ' super bomb
    animateActor act
    moveActor act, 3, 0
    if x < screenWidth() * 0.75 then
        resetTimer tmr
    elseif timerHasExpired(tmr) then
        flashbang
        so["state"] = "dead"
    endif
elseif type = "flashbang" then
    ' flashbang
    if so["alpha"] = 1.0 then
        for key in g_enemies
            enemy = g_enemies[key]
            enemy["hp"] -= 50
        next
    endif

    so["alpha"] = max(0, so["alpha"] - 0.005)
```

```
    if so["alpha"] <= 0.0 then so["state"] = "dead"
elseif type = "brainLaser" then
    ' brain laser
    if so["alpha"] = 1.0 then
        ' did the laser hit an enemy (different than the brain)?
        for key in g_enemies
            enemy = g_enemies[key]
            eact = enemy["actor"]
            if enemy <> so["parent"] then
                if actorX(eact) <= x and actorY(eact) + actorYAnchor(eact) >= y
                    explode actorX(eact), actorY(eact)
                    enemy["hp"] -= 100
                endif
            endif
        next

        ' did the laser hit the player?
        player = playerActor(g_player)
        if actorY(player) + actorYAnchor(player) >= y - so["height"]/2 and acto
    endif

    so["height"] = max(1, so["height"] - 0.5)
    so["alpha"] = max(0, so["alpha"] - 0.05)
    if so["alpha"] <= 0.0 then so["state"] = "dead"
elseif type = "text" then
    ' blinking text
    if timerHasExpired(tmr) then
        resetTimer tmr
        so["visible"] = not so["visible"]
        so["loops"] -= 1
        if so["loops"] <= 0 then so["state"] = "dead"
    endif
elseif type = "levelcleared" then
    ' level cleared
    ' fadeout
```

```
        if so["alpha"] >= 1.0 then
            so["alpha"] = 1.0
        else
            so["alpha"] += 0.01
            so["ms"] = milliseconds()
        endif

        ' fade out music
        setVoiceVolume so["music"], max(0, voiceVolume(so["music"]) - 0.01)
    endif

    ' left the screen?
    if (x > 740 or x < -100 or y > 580 or y < -100) then so["state"] = "dead"
endfun

fun renderSpaceObject(so)
    if so["visible"] then
        if so["type"] = "flashbang" then
            ' flashbang
            a = getAlpha()
            setAlpha so["alpha"]
            setColor 255, 255, 255
            rectfill 0, 0, screenWidth(), screenHeight()
            setAlpha a
        elseif so["type"] = "brainLaser" then
            ' brain laser
            a = getAlpha()
            setAlpha so["alpha"]
            setColor 255, 0, 0
            setBlendingMode "additive"
            rectfill 0, actorY(so["actor"]) - so["height"]/2, actorX(so["actor"]),
            setBlendingMode "normal"
            setAlpha a
        elseif so["type"] = "text" then
            ' blinking text
```

```
        txt = so["text"]
        x = 320-len(txt)*7
        y = 225
        setcolor 0, 0, 0
        textout x+1, y+1, 16, txt
        setcolor 255, 255, 255
        textout x, y, 16, txt
    elseif so["type"] = "levelcleared" then
        ' level cleared
        a = getAlpha()
        setAlpha so["alpha"]
        setColor 0, 0, 0
        rectfill 0, 0, screenWidth(), screenHeight()
        setAlpha a

        if so["alpha"] >= 1.0 then
            seconds = floor(so["ms"] / 1000) mod 60
            minutes = floor(so["ms"] / 60000)

            c = max(0, (so["ms"] + 2000 - milliseconds()) * 0.01)
            setscale 1.0+c, 1.0+c

            setcolor 255, 255, 255
            line 55, 40, 585, 40
            textoutEx 180, 70, "serif", 48, "WINNER!"
            line 55, 170, 585, 170
            textoutEx 45, 210, "serif", 18, "Time:"
            textoutEx 45, 250, "serif", 18, "Score:"
            textoutEx 210, 425, "serif", 18, "Thanks for playing!"
            setcolor 255, 255, 0
            textoutEx 205, 210, "serif", 18, minutes + "min " + seconds + "s"
            textoutEx 205, 250, "serif", 18, so["score"]

            setscale 1, 1
```

```
                if c = 0 and not so["playedSample"] then
                    so["playedSample"] = true
                    playSample "yeah.ogg"
                endif
            endif
        elseif type = "explosion" then
            ' explosion
            setBlendingMode "additive"
            drawActor so["actor"]
            setBlendingMode "normal"
        else
            ' other object
            drawActor so["actor"]
        endif
    endif
endfun



' ---------- // HUD // --------------

fun createHud()
    hud = createTable()
    hud["player"] = g_player
    hud["boss"] = 0
    hud["visible"] = true
    hud["fpsTimer"] = createTimer(1000)
    hud["fps"] = 0
    hud["fpsAccum"] = 0
    return hud
endfun

fun destroyHud(hud)
    destroyTimer hud["fpsTimer"]
    return destroyTable(hud)
```

```
endfun

fun updateHud(hud)
    ' fps counter
    hud["fpsAccum"] += 1
    if timerHasExpired(hud["fpsTimer"]) then
        hud["fps"] = hud["fpsAccum"]
        hud["fpsAccum"] = 0
        resetTimer hud["fpsTimer"]
    endif
endfun

fun renderHud(hud)
    if not hud["visible"] then return

    x = 20
    y = 20
    w = 250
    h = 10
    sw = screenWidth()
    sh = screenHeight()
    player = hud["player"]
    boss = hud["boss"]

    ' player hp
    setcolor 255, 255, 255
    rect x-1, y-1, w+1, h+1
    setcolor 0,0, 255
    rectfill x, y, w * (player["hp"]/100), h

    ' player bombs
    for i=1 to player["bombs"]
        drawImage "bomb.png", x + (i-1)*(imageWidth("bomb.png")+10), y+h+10
    next
```

```
    ' boss hp
    if isValidTable(boss) then
        setcolor 0, 0, 0
        textout -x+sw-49, y+h+11, 12, "BOSS"
        setcolor 255, 255, 255
        textout -x+sw-50, y+h+10, 12, "BOSS"
        rect -x-1+sw-w, y-1, w+1, h+1
        setcolor 255, 0, 0
        rectfill -x+sw-w, y, w * ((boss["hp"]-100)/600), h
    endif

    ' fps counter
    setcolor 0, 0, 0
    textout sw-46, sh-19, 10, "fps: " + hud["fps"]
    setcolor 255, 255, 255
    textout sw-47, sh-20, 10, "fps: " + hud["fps"]
endfun

fun setHudBoss(hud, boss)
    hud["boss"] = boss
endfun

fun hideHud(hud)
    hud["visible"] = false
endfun
```

Referências Bibliográficas

1 ADAMS, K. Y. C. *Só JavaScript*. [S.l.]: Bookman® Companhia Editora, 2009. ISBN 978-85-7780-542-6.

2 AHO, A. V. et al. *Compiladores - Princípios, técnicas e ferramentas (2ª edição)*. [S.l.]: Pearson Education do Brasil, 2008. ISBN 978-85-88639-24-9.

3 HICKSON, R. *C++ Técnicas Avançadas*. [S.l.]: Editora Campus Ltda., 2003. ISBN 85-352-1275-2.

4 DROZDEK, A. *Estruturas de Dados e Algoritmos em C++*. [S.l.]: Pioneira Thomson Learning Ltda., 2002. ISBN 85-221-0295-3.

5 PERUCIA, A. S. et al. *Desenvolvimento de Jogos Eletrônicos - Teoria e Prática*. [S.l.]: Novatec Editora Ltda., 2005. ISBN 85-7522-068-3.

6 PFEIFFER, S. Parents and their effect on standards: Open video codecs for html5. International Free and Open Source Software Law Review, v. 1, 2010.

7 SERRANO, F. No Tapetão. O Estado de S. Paulo, São Paulo, 18 jul. 2011, Caderno Link, p. L4.

8 ERICKSON, J. *Hacking*. [S.l.]: Digerati Books, 2009. ISBN 978-85-60480-30-2.

9 LOBO, E. J. R. *Guia Prático de Engenharia de Software*. [S.l.]: Digerati Books, 2009. ISBN 978-85-7873-036-9.

10 DIAS, T. de M. Em celulares, venceu o HTML5. O Estado de S. Paulo, São Paulo, 14 nov. 2011, Caderno Link, p. L5.