

DamasTV: um Jogo de Damas para o SBTVD

Esdras Caleb Oliveira Silva¹ Débora C. Muchaluat Saade²

¹Dep. Engenharia de Telecomunicações, ²Dep. Ciência da Computação
Laboratório de MídiaCom, Universidade Federal Fluminense, Niterói, RJ, Brasil

Resumo

Este artigo apresenta o jogo DamasTV, desenvolvido para o sistema brasileiro de televisão digital (SBTVD). O jogo oferece dois modos de funcionamento, o modo *standalone*, onde o usuário joga sozinho contra a TV (computador) ou com outra pessoa ao seu lado e o modo distribuído, onde jogadores distintos interagem através do canal de retorno. A interface do jogo apresenta o tabuleiro em perspectiva isométrica, tornando-a mais agradável ao usuário. O DamasTV foi desenvolvido com as linguagens NCL e Lua, seguindo as normas do *middleware* declarativo Ginga-NCL.

Palavras Chave: Ginga-NCL, NCL-Lua, TV digital interativa, jogo de damas, DamasTV

Contato do autor:

{caleb,deбора}@midia.com.uff.br

1. Introdução

O Brasil recentemente padronizou o seu sistema de televisão digital, o SBTVD [ABNT 2007]. Sistemas de televisão digital possibilitam imagens em alta definição, melhor qualidade de áudio e programas interativos. Com a adoção nacional do sistema digital, os telespectadores podem usar o sistema de televisão como plataforma de jogos interativos, como já acontece em alguns sistemas pagos. Jogos podem ajudar a diminuir o problema da exclusão digital, proporcionando entretenimento e informação a parcelas da população que não têm acesso a Internet através de computadores tradicionais. Neste contexto, o desenvolvimento de jogos para televisão se torna um tópico de grande interesse. Jogos também ajudam o desenvolvimento do raciocínio lógico de crianças.

O SBTVD usa um *middleware* de padrão aberto chamado Ginga. O *middleware* Ginga é subdividido em Ginga-J, responsável pela execução das aplicações procedurais, e Ginga-NCL, responsável pela execução de aplicações declarativas [ITU 2009]. Comumente se pensa em desenvolver jogos usando o Ginga-J, com o uso da linguagem Java. Este artigo demonstra o potencial do Ginga-NCL para o desenvolvimento de jogos através das linguagens NCL e Lua.

Este artigo apresenta o jogo DamasTV, desenvolvido para o sistema brasileiro de televisão digital. O jogo oferece dois modos de funcionamento, o modo *standalone*, onde o usuário joga contra a TV ou outra pessoa ao seu lado, e o modo distribuído, onde

jogadores distintos interagem através do canal de retorno. A interface do jogo apresenta o tabuleiro em perspectiva isométrica tornando-a mais agradável ao usuário.

A motivação para se criar um jogo de damas para TV digital vem do fato que damas é um jogo bastante popular, principalmente entre crianças e pessoas de terceira idade. Por isso, DamasTV pode motivar o uso e ajudar a divulgar o novo sistema de TV digital.

O artigo está estruturado da seguinte forma. Os trabalhos relacionados são discutidos na Seção 2. A Seção 3 apresenta resumidamente o *middleware* Ginga-NCL e as linguagens usadas para o desenvolvimento do jogo. A Seção 4 apresenta a estrutura do jogo DamasTV em NCL e Lua. Na Seção 5, as considerações finais e trabalhos futuros são comentados.

2. Trabalhos Relacionados

Santos também aborda o tema de jogos para o SBTVD [Santos et al. 2008]. Ele discute um jogo feito para o Ginga-J chamado RummiTV, uma implementação do jogo de tabuleiro *Rummikubi*. O artigo foca no padrão brasileiro de TV digital e nos aspectos técnicos do jogo, tais como o modelo de dados orientado a objeto. O *RummiTV* tem uma implementação distribuída usando o *Java RMI* (*Remote Method Invocation*).

Junior trata do desenvolvimento de jogos para televisão digital [Junior et al. 2008]. O trabalho apresenta dois jogos desenvolvidos para o sistema Ginga-NCL, entretanto, não apresenta a utilização do canal de retorno.

Barboza e Clua [2009] falam sobre o desenvolvimento de um *framework* para jogos de televisão digital. Usando como base o Ginga-J, os autores descrevem o *framework* GingaGame e mostram um jogo projetado com o mesmo. O DamasTV contém partes do código que podem ser reusados em outros jogos ou em um *framework* para jogos de tabuleiro.

3. Ginga-NCL

As aplicações no SBTVD funcionam sobre o *middleware* Ginga, que é fruto do desenvolvimento de projetos de pesquisa coordenados pelos laboratórios Telemídia da PUC-Rio e LAViD da UFPB. Um *middleware* para aplicações de TV digital consiste de

um conjunto de máquinas para execução das linguagens oferecidas e bibliotecas de funções, que permitem o desenvolvimento rápido e fácil de aplicações [Soares et al. 2007].

O Ginga-NCL funciona com base na linguagem declarativa (*Nested Context Language*) [ABNT 2007, ITU 2009]. NCL é uma linguagem declarativa para autoria de documentos de hipermídia baseados no modelo conceitual NCM (*Nested Context Model*) [Soares and Rodrigues 2005] que foi desenvolvida usando uma estrutura modular, seguindo os princípios adotados pelo W3C. A linguagem NCL utiliza os conceitos de nós (de mídia e contexto) e elos. Os nós especificam as mídias em si, tais como vídeos, imagens, legendas e áudio, e os elos representam os relacionamentos entre os nós. Como NCL não é uma linguagem de propósito geral, é muito complexo construir aplicações como jogos usando puramente a linguagem NCL.

A especificação do Ginga-NCL prevê a possibilidade do suporte a *scripts* Lua ou *Ecmascript* para complementar o desenvolvimento de uma aplicação. No caso do DamasTV, optou-se por usar a linguagem Lua, por ser rápida, simples e já amplamente usada em jogos [Celes et al. 2004]. Um script Lua é tratado em NCL como um nó de mídia a ser executado. Ao ser iniciado um nó com o script Lua, o Ginga-NCL envia ao interpretador Lua os eventos do sistema tais como teclas do controle remoto pressionadas, mudanças de estado do nó e informações do canal de retorno. Os eventos são tratados em Lua com a ajuda do módulo *event* que é uma classe parte da biblioteca NCL-Lua [Sant'Anna et al. 2008]. Dessa forma pode se ter um ambiente procedural dentro do ambiente declarativo Ginga-NCL.

Somente dois módulos da biblioteca NCL-Lua serão usados no DamasTV, o módulo *canvas* e o módulo *event*. O módulo *canvas* é usado para desenhar objetos na tela. E ele pode desenhar formas básicas, usar figuras prontas em vários formatos (como png e gif, por exemplo) ou pode também usar fontes em *ttf* para escrever texto. O módulo *event* além de receber os eventos gerados pelo Ginga também pode gerar seus próprios eventos, que podem ativar elos no documento NCL, permitindo uma maior interação entre as linguagens NCL e Lua. Os eventos gerados por este módulo têm classes que definem sua utilidade. A classe *tcp* é usada para comunicação via Internet através do canal de retorno. O módulo *event* ainda permite ver o tempo que a aplicação está em execução e gerar *timers*, o que é útil para depuração e para ajudar a implementar a *loop* principal do jogo.

A linguagem Lua tem as suas próprias bibliotecas, que auxiliam o desenvolvimento de aplicações. A função *random* permite fazer escolhas aleatórias, permitindo assim a criação de uma inteligência artificial mais robusta [Celes et al. 2004]. A biblioteca

socket permite criar uma interface cliente servidor na aplicação. No caso do DamasTV, essa biblioteca foi usada para criar uma conexão direta entre dois jogadores.

O jogo utiliza orientação a objetos para facilitar a organização e reúso do código. Por exemplo, o objeto *pecaTabuleiro* contém um método para mover-se pelo tabuleiro, dentro do método está todo o algoritmo que muda a posição da peça e a coloca na nova posição do tabuleiro. Embora Lua não seja uma linguagem orientada a objeto, é possível implementar alguns conceitos através da estrutura em tabelas [Ellwanger and Balbinot 2003].

4. DamasTV

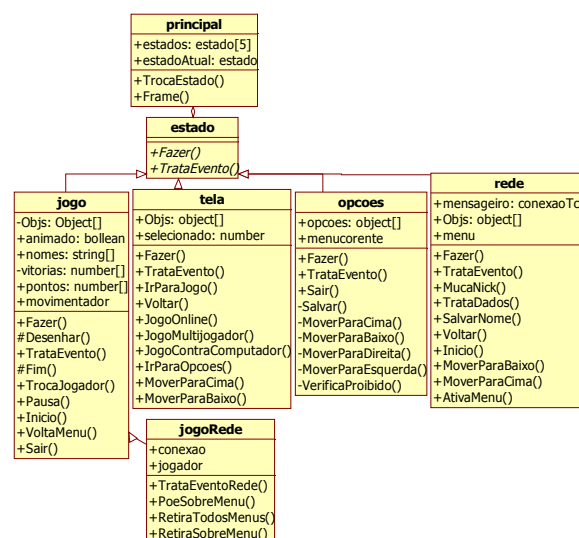


Figura 1: Diagrama de classes

O jogo pode ser dividido em 5 estados: *tela*, *jogo*, *opcoes*, *rede*, *jogoRede*, ilustrados na Figura 1. Cada um destes estados é representado por uma classe filha da classe abstrata *estado*, que define duas funções: *Fazer* e *TrataEvento*. Essas funções são responsáveis por processar e desenhar os objetos na tela e gerenciar os eventos recebidos pelo controle remoto respectivamente. Todo o jogo funciona sob uma classe principal chamada *principal*. Essa classe tem um vetor com os objetos *estado* e um método chamado *TrocaEstado*, que troca o estado atual. A classe *principal* registra a função *TrataEvento* como tradador de eventos NCL através do uso do módulo *event*. Caso o estado possua a opção animado como verdadeira e a variável global ANIMACAO seja verdadeira, um *timer* irá executar a função *Fazer* do estado atual a cada período de 100 milissegundos, gerando as animações na tela.

O estado *tela* permite que o usuário escolha o que ele deseja fazer: jogar, mudar opções, jogar *online*, o que é feito através de um menu com várias opções. A função *TrataEvento* faz a navegação pelas opções, respondendo às teclas do controle remoto. Existe um vetor chamado *objetos*, que contém cada uma das

opções no menu. Cada um desses objetos tem o método (*aoSelecionar*), que diz o que acontece caso o objeto seja selecionado e o atributo *texto*, que contém o nome do objeto para exibição na tela. Quando o objeto é selecionado, ele pode tanto mudar o estado do jogo para outro, como pode trocar as opções disponíveis para o usuário modificando o vetor *objetos*. A função *Fazer* irá varrer o vetor de objetos e escrever o nome deles um abaixo do outro com a ajuda da função de desenhar texto do módulo *canvas*. Se o índice selecionado é o índice que ele está desenhando ele o coloca numa cor diferenciada (vermelho), senão ele o coloca na cor padrão (preto). A tela de opções está ilustrada na Figura 2.

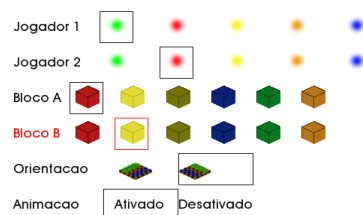


Figura 2: Tela de opções

O estado de opções permite que o jogador altere parâmetros do jogo tais como: a cor das peças, cor do tabuleiro ou orientação em que as peças são postas. A função *Fazer* desenha o fundo da tela que informa ao usuário as opções que ele tem. Em seguida desenha quais estão selecionadas de acordo com os valores das variáveis globais. Um vetor auxiliar chamado *opcoes* contém o valor selecionado pelo usuário no momento e nome da opção correspondente. No *TrataEvento* encontram-se as funções de movimentação que além de selecionar a opção atual, selecionam o valor e verificam as opções inválidas. Ao apertar o botão de selecionar, o *TrataEvento* executa a rotina *Salvar*. Nessa rotina as opções selecionadas são salvas no arquivo *constantes.lua* e, em seguida, ele torna estas escolhas válidas para o jogo retornando ao estado *tela*. A interface do DamasTV está ilustrada na Figura 3.

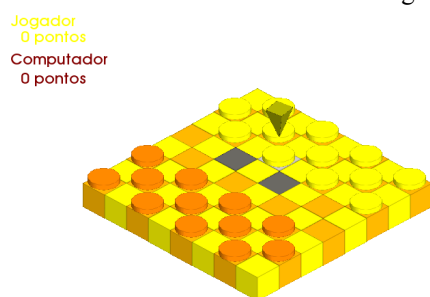


Figura 3: Tela do jogo

O estado *jogo* é o principal estado. É composto de 6 classes que ajudam o jogo a funcionar: *tabuleiro*, *cursorTabuleiro*, *pecaTabuleiro*, *movimentador*, *menuJogo* e *jogo*, que é a classe principal. No *tabuleiro* estão as peças, os blocos, o cursor, o vetor *blocosJogadas*, o método que desenha o tabuleiro e tudo que ele contém (peças, bloco e cursor). Ele tem três métodos auxiliares *TemJogada*, *TemPeca* e *TemCome*. O primeiro checa se em uma dada posição

existe uma jogada presente no vetor de jogadas. O segundo checa se há uma peça no espaço correspondente às coordenadas e o terceiro checa o vetor de jogadas e descobre se alguma delas pode comer uma peça. Quem preenche o vetor *blocosJogadas* é o *movimentador*. O *movimentador* é uma classe que serve para controlar, ou seja recebe diretamente os eventos do controle remoto através da função *TrataEvento*, e movimenta o cursor e as peças no tabuleiro. Neste estado, quando uma peça é selecionada, o *movimentador* chama os métodos internos da *peca* para preencher o vetor de jogadas no *tabuleiro*. O tabuleiro muda a cor dos espaços onde podem ser feitas jogadas, da peça selecionada, e onde o cursor está, ajudando a mostrar ao jogador a jogada que ele pode fazer. Quando um espaço onde há uma jogada é selecionado o *movimentador* chama uma função que se encarrega de fazer a jogada e mudar a vez para o outro jogador. O *menuJogo* se encarrega de mostrar a pontuação dos jogadores e criar objetos que controlem o jogo no lugar do *movimentador*, quando o mesmo for pausado ou chegar ao seu fim. Em ambos os casos, um menu será aberto em cima do tabuleiro, dando ao jogador as opções que ele pode escolher. O *menuJogo* também escreve no canto direito da tela as últimas jogadas feitas. A interface do jogo está ilustrada na Figura 3.

A classe *computador* é usada para jogos no modo *standalone*. Ela é programada com uma inteligência artificial simples que pontua as melhores jogadas e decide aleatoriamente qual jogada deve ser feita, entre as jogadas de maior pontuação. Ele se comunica com os objetos *movimentador* e *tabuleiro*, para decidir e realizar suas jogadas.

A classe *rede*, é usada para a conexão com o servidor do DamasTV, acessível via canal de retorno. Para se conectar ao servidor é necessário usar um apelido. O jogador pode escolher um apelido usado anteriormente ou criar um novo. Uma classe chamada *menuDeEscrita* permite ao jogador escrever seu próprio apelido, usando o controle remoto de maneira similar ao teclado do celular (apertando uma mesma tecla ele alterna entre um conjunto de caracteres). Ao escrever um apelido o mesmo é gravado no arquivo *constantesRede.lua* para já trazer o nome com opção para o usuário numa próxima sessão. A conexão com o servidor é feita via *HTTP* por meio de páginas em *PHP*. O usuário acessa as páginas enviando as informações que autenticam sua conexão no servidor. Ao escolher o apelido, o cliente do jogo se conecta ao servidor e entra em modo de procura por um adversário para o usuário. Ao encontrar o adversário, o servidor envia as informações necessárias para iniciar o jogo. Se ambos estiverem na mesma rede local ou estiverem usando endereços IPs válidos na Internet, o servidor envia o IP do adversário ao jogador e o mesmo inicia o *jogoRede* no modo direto, onde os jogadores se conectam e se comunicam diretamente (sem intermediação do servidor). Se um dos usuários estiver

usando NAT o servidor envia apenas as informações relacionadas ao jogo e o *jogoRede* é iniciado no modo indireto, onde os jogadores enviam e recebem as informações do servidor. A comunicação através do canal de retorno é feita pela classe *mensagemiro*, que se conecta com a página pedida e retorna a resposta (que é uma tabela), para função *TrataDados* que trata os dados recebidos. O funcionamento é parecido com o da classe *tela*: várias opções para o usuário selecionar (os apelidos, escrever um novo apelido, voltar) são colocadas em um vetor, que também contém a função correspondente a ser executada quando uma opção é selecionada.

A classe *jogoRede* é uma classe filha da classe *jogo*. Tem todos os métodos e atributos da mesma, e mais alguns outros para a comunicação entre dois usuários remotos, como o *menuJogoRede*. Uma classe filha da classe *menuJogo* permite, além das funcionalidades da classe pai, a troca de mensagens entre os usuários. Como mencionado existem dois modos de conexão com o adversário, o direto e o indireto. Em ambos os casos o primeiro a receber as informações do servidor é responsável por se comunicar com o servidor para dizer quem venceu o jogo e criar um novo jogo, caso queiram jogar de novo. No modo direto, a conexão é feita pela classe *conexaoDireta*, que se utiliza da biblioteca *socket* de *Lua* para fazer a conexão direta entre dois jogadores. No modo indireto, a conexão é feita pela classe *conexaoIndireta*, que usa a classe *mensagemiro* para se comunicar com o servidor, mas tem métodos idênticos ao da *conexaoDireta* para tornar transparente ao usuário o modo de conexão usado. Em ambos os casos haverá um *timer* sendo rodado a cada 100 milissegundos checando se existe alguma nova informação para ser recebida.

O servidor é programado em *PHP* e usa o *MySQL* para gravar as informações dos usuários, dos jogos e outras informações, como o que cada usuário transmite no caso de conexão indireta.

Toda a lógica e a interface do jogo foram desenvolvidas em *Lua*. A parte do código em *NCL* consiste de quatro nós principais: o jogo, o vídeo introdutório, a música de fundo e o nó de efeitos sonoros. Além de três outros nós que servem para atribuir valores a variáveis de ambiente do *NCL*, como foco inicial, controle dos eventos do controle remoto e a variável *efeito* que diz o efeito sonoro a ser ativado. O nó de efeitos sonoros é um nó com múltiplas mídias alternativas (*switch*) e precisa de uma base de regras no cabeçalho do documento para selecionar a mídia correspondente. As regras analisam o valor da variável *efeito* e dizem qual mídia deve ser executada no nó de efeitos sonoros. Quando o jogo vai iniciar algum efeito sonoro (um som de seleção, o barulho de uma peça sendo comida, etc...) ele dispara um evento *NCL* através do módulo *event*. O evento ativa uma âncora interna no jogo, que ativa um elo no documento *NCL*.

O elo modifica o valor da variável *efeito*, desativa e reativa o nó de efeitos sonoros. Ao ser reativado o nó de efeitos sonoros vai até a base de regras e verifica qual é o efeito sonoro que deverá ser ativado de acordo com o novo valor da variável *efeito*.

5. Conclusão

Este artigo apresentou um jogo de damas interativo para o SBTVD, chamado DamasTV, desenvolvido com as linguagens *NCL* e *Lua*. O jogo DamasTV encontra-se pronto e disponível no site do Clube *NCL* (clube.ncl.org.br).

Como trabalhos futuros pretende-se: implementar diferentes níveis de dificuldade para o jogo no modo contra a TV; a opção de retirar o auxílio sobre jogadas quando uma peça é selecionada para aumentar a dificuldade e a opção do usuário personalizar as regras. Outro trabalho futuro é a separação de componentes reusáveis na implementação, que possam auxiliar na implementação de outros jogos, como xadrez e jogos de RPG.

Referências

- Barbosa, D.C., Clua, E.W.G., 2009, *Ginga Game: A Framework for Game Development for the Interactive Digital Television*, SBGames 2009, Rio de Janeiro.
- Celes W., Figueredo L.H., Ierusalimschy R., 2004, *Linguagem lua e suas aplicações em Jogos*, Workshop de Jogos 2004.
- Ellwanger, F., Balbinot, G., 2003, *A Linguagem de Programação Lua*, III CONSIPA 2003, UNISINOS.
- ITU. ITU International Telecommunication Union 2009. *Nested Context Language (NCL) and Ginga-NCL for IPTV services*. ITU-T Recommendation H.761. 2009.
- Junior, A.N.S., Souza, A.C.S., Santos, L.C.M., Sampaio, R.L., Raimundo, P.O., 2008, *Desenvolvimento de Jogos para o Sistema Brasileiro de TV Digital*, SCGames 2008, RJ.
- Sant'Anna, F., Cerqueira, F.S.R., Soares L.F.G., 2008. *NCLua - Objetos Imperativos Lua na Linguagem Declarativa NCL*. Simpósio Brasileiro de Sistemas Multimídia e Web, Vila Velha, ES.
- Santos, J. ; Ratamero, E. ; Arruda, J. P. S. ; Dantas, M. ; Sanchez, M. L. A. ; Saade, D.C.M. . RummiTV: An Interactive Game for the Brazilian Digital TV System. In: XXVI Simpósio Brasileiro de Telecomunicações, 2008, Rio de Janeiro.
- Soares, L.F.G., Rodrigues, R.F. 2005. *Nested Context Model 3.0 Part 1 - NCM Core* Monografias em Ciência da Computação do Departamento de Informática da PUC-Rio, MCC 18/05, 2005.
- Soares, L.F.G., Rodrigues, R.F. Moreno, M.F., 2007. *Ginga-NCL: the Declarative Environment of the Brazilian Digital TV System*. Journal of the Brazilian Computer Society, volume 4, número 12.