Desenvolvimento de Jogos para a plataforma Ginga utilizando NCLua

Ricardo Mendes Costa Segundo Tatiana Aires Tavares

Laboratório de Aplicações de Vídeo Digital (LAVID), Universidade Federal da Paraíba - João Pessoa, PB – Brasil

Resumo

O Ginga, middleware do Sistema Brasileiro de TV Digital, tem ganhado importância internacional, sendo adotado por diversos países da América do Sul e ao se tornar recomendação ITU para IPTV. Com isso, o sinal digital, irá ser recebido por mais televisores, levando vídeo de alta definição e áudio de alta qualidade aos telespectadores, como também a possibilidade de interagir com as aplicações televisivas. Assim este trabalho tem por objetivo apresentar a plataforma de desenvolvimento Ginga, suas características, linguagens, funcionalidades e aplicação no desenvolvimento de jogos, focando especificamente no subsistema GingaNCL..

Keywords: Ginga; TV Digital; GingaNCL; Jogos; Lua

Authors' contact:
{ricardo,tatiana}@lavid.ufpb.br

1. Introdução

A televisão no Brasil assume papéis importantes, sendo fonte de informação e de entretenimento para muitos brasileiros. O advento da Televisão Digital no Brasil traz aos usuários não apenas uma melhor experiência em termos de som e imagem, mas possibilita a interatividade.

A interatividade por si só não é algo novo na TV, programas como Wink Dink and You (Winki Dink, 1953), Você Decide (Você Decide, 1992) e Big Brother (Big Brother, 2000) possuem recursos de interatividade. Porém apenas no primeiro programa citado existe interação direta com a televisão, todos os outros a interação ocorre por meios externos como telefone e Internet. Já o programa Wink Dink and You apesar de interagir através da televisão, o faz de forma analógica. O que a TV Digital traz de novo é a forma de interagir com o telespectador, ele não terá que utilizar outros dispositivos (celular, computador, giz de cera,...) como intermediários para a interação.

Sobre a interatividade, é possível pensar em um tipo de aplicação que leva este termo ao extremo, os jogos. Os jogos possuem a característica de envolver os jogadores de tal forma que fazem com que eles se sintam dentro da aplicação, ou seja, eles não se sentem apenas controlando os personagens, sentem fazer parte deste (Schell, 2008). E este tipo de experiência tem na maioria das vezes o objetivo de entreter, fazendo dos jogos uma forma de entretenimento bastante difundida nos dias de hoje.

2. Programas de TV Interativos

EPGs, Enhanced TV, conteúdo sobre demanda, TV personalizada, Internet na TV, jogos e apostas, publicidade interativa e t-commerce são oito categorias que (Jensen, 2005) classifica como os tipos de aplicações para TV.

EPG é a Interface gráfica que possibilita a navegação pelas várias possibilidades de programação que o usuário encontrará na TV Digital, sendo o equivalente aos guias de horários de televisão publicados nos jornais, com funções e operação análoga a de um portal de internet, contendo informações como horárias de inicio e fim de um programa, recomendação etária e sinopse.

Enhanced TV é, de forma simplificada, conteúdo extra, normalmente na forma de texto e imagens, que são sobrepostos aos vídeos que estão sendo exibidos, passando ao telespectador informações adicionais sobre o programa que esta sendo exibido, sendo este conteúdo alcançado pelo usuário de forma interativa através das aplicações interativas. Exemplos comuns incluem estatísticas em eventos esportivos, informações sobre personagens e artistas de TV. Os conteúdos podem ser independentes da programação como serviços de notícias, email e catálogos de venda. Conteúdo sobre demanda se refere a conteúdos que você pode obter através de serviços disponibilizados, como vídeos sob demanda, música e jogos, pagando taxas para poder obter tais serviços.

TV personalizada pode ser vista como formas de gerir informação. Temos então dois tipos: personalização e customização. Personalização é feito pelo sistema de mídias, onde este envia informações individualizadas ao usuário. Já customização é dirigida pelo usuário, ou seja, ele seleciona opções e obtém conteúdos personalizados a partir daquelas escolhas.

Internet na TV permite aos usuários realizar muitas das atividades normalmente desempenhadas em um computador conectado à Internet através do aparelho de televisão, por exemplo, ler e escrever e-mails e mensagens instantâneas, participar de bate-papos e discussões de grupos, navegarem na web ou fazer pesquisas na internet.

Publicidade interativa é a comunicação de marketing e de mensagens de marcas utilizando a interatividade da transmissão digital, permitindo ao usuário solicitar mais informações sobre produtos apresentados, e também em alguns casos permite que o espectador compre o produto diretamente do anúncio. Neste aspecto a publicidade junta fronteiras com T-commerce.

T-commerce é simplesmente o fenômeno do e-commerce conhecido na Internet e transferido para o meio televisivo. Enquanto a televisão tradicional cria conteúdos criativos, colocando anúncios entre os segmentos, a televisão interativa com T-commerce gera conteúdos também criativos, apoiando as vendas durante o conteúdo. Por outras palavras, T-commerce permite ao telespectador comprar produtos e serviços que ele vê na tela da televisão. Desta forma, os telespectadores se tornam consumidores à medida que eles fazem transações através da televisão, que serão realizadas de diversas maneiras.

Jogos possuem na TV diversas possibilidades, tanto na forma de interação com o conteúdo quanto à forma de distribuição do mesmo. O conteúdo de um jogo pode estar relacionado ou não ao programa de TV que esta sendo exibido, ou até pode não haver programas sendo exibidos, sendo o jogo o foco de entretenimento naquele momento. Na forma de distribuição, os jogos podem ser enviados via broadcasting pela emissora, podem ser baixados pela internet ou ate por serviços Pay-per-play, onde o jogador paga pelos jogos que queira jogar na TV.

2.1 Jogos para TV

A interatividade no mercado nacional ainda esta dando seus primeiros passos, e com isso os jogos para TV aqui ainda estão em fase de estudos

Quanto ao âmbito internacional, existem muitos trabalhos e experiências que servem de base para o desenvolvimento de jogos nacionalmente. Artigos como o de (Nolan , 2002) e (Heliö S. e Järvinen , 2003) mostram classificações dos jogos para TV, desafios para o desenvolvimento nessa plataforma e também propõe soluções imediatas e futuras para tais problemas.

Nolan mostra os perfis das pessoas que jogam na televisão, classificando-os em Geração I, Early Clickers (jovens clicadores) e Daytime Dabblers. A geração I corresponde aos jovens que jogam um jogo sem se ater a jogá-lo novamente e nem em que canal o jogo esta sendo transmitido. Eles simplesmente navegam entre os jogos assim como navegam entre os canais, procurando por algum entretenimento rápido. Early Clickers correspondem às crianças entre três e dez anos de idade que possuem familiaridade com a tecnologia e não possuem acesso ao computador por imposição dos pais, tendo sua atenção transferida para a TV. O ultimo grupo é representado por pessoas que não tem familiaridade com tecnologias, mas que vêem a TV como algo que podem dominar. Essas pessoas jogam por ser uma forma fácil de entretenimento, pois são mais simples que os demais serviços para a TV.

Jogos na TV podem ser utilizados de diferentes formas. Eles podem ser utilizados como produtos únicos de uma emissora, ou seja, um canal especializado em jogos para TV digital. Esses canais transmitem apenas jogos, em horários definidos do dia, como se fossem programas de TV, podendo assim os jogadores praticar todos os dias, e até submeterem seus recordes a partir do canal de retorno. Um exemplo de canal de jogo é o canal PlayinTV, um dos maiores da Europa apresentando em sua lista de jogos para TV mais de 200 títulos, indo de jogos clássicos como xadrez, a criações próprias como o Frogs, cujo objetivo do jogo é levar um sapo de um lado a outro do rio pulando para fugir dos desafios. Como podemos ver na Fig. 1, tais jogos apresentam uma boa qualidade gráfica mesmo com as limitações de jogos para a TV, e a diversidade de jogos encontrados no canal mostra que existem muitas possibilidades de tipos de jogos que podem agradar a todos que jogam pela TV.

Figura 1: Jogos: a) Chess (jogo de tabuleiro); b) Frogs (jogo arcade); c) Johnny Megatone (jogo de ação arcade) d)Tennis (jogo esportivo)

Jogos também podem ser utilizados junto à programação, na forma de conteúdo extra. Eles podem, por exemplo, em um programa de auditório colocar o usuário interagindo com o programa ao responder perguntas que podem ser feitas aos participantes do programa, dando ao telespectador uma imersão maior no programa. Os jogos também podem ser utilizados na TV para atos publicitários substituindo ou estendendo a propaganda com um jogo.

No Brasil, comercialmente as experiências com jogos na TV são escassas, se limitando a jogos pré-instalados em receptores de TVs pagas (vide Fig. 2) e alguns poucos jogos em TV aberta, como o jogo Garganta e Torcicolo (vide Fig. 3), que se tratava de um jogo onde dois participantes ligavam para o programa via telefone, e controlavam os personagens do jogo através das teclas do telefone.

Figura 2: Jogo na TV por assinatura SKY

Figura 3: Jogo Garganta e Torcicolo, cujo objetivo era destruir o maior número de ovelhas

Apesar de poucas, existem tentativas de uso de jogos na televisão brasileira, de forma que com o Ginga estas tentativas podem aumentar, e passarem a ser um nincho de negócio para as emissoras interessadas. Desta forma o desenvolvimento de um framework compatível com o Ginga pode ser capaz de acelerar o desenvolvimento de jogos para TV Digital ajudando na popularização de jogos e programas que se utilizam destes jogos na TV.

2.2 Possibilidades

A programação de aplicações para TV Digital possui diversos desafios relacionados a questões de interface que devem considerar a TV como saída das aplicações e questões de sobreposição do vídeo que é enviado junto à aplicação. Desafios relacionados ao público alvo das aplicações produzidas, pois como a distribuição destas aplicações é feita em broadcast, ou seja, todos os telespectadores sintonizados na emissora de TV irão receber a mesma aplicação, mesmo possuindo perfis muito diferentes. E por fim, desafios de Interação Humano Computador (IHC), tanto no aspecto de usabilidade dos dispositivos de interação (controle-remoto), quanto desafios de aceitabilidade das aplicações.

Tais desafios podem limitar as possibilidades de jogos executáveis na plataforma, porém, podemos levantar algumas possibilidades que deverão ser opções para os desenvolvedores de jogos.

A primeira possibilidade é o uso da plataforma como um console de jogos, onde o jogador pode adquirir o jogo através do canal de retorno conectado a internet e executá-lo em sua TV, ou até mesmo jogar jogos enviados pela emissora que não sejam relacionados à grade de programação.

Outro modelo possível é o surgimento de canais exclusivos de jogos. Nestes canais não há uma programação tradicional, em vez de programas de TV são enviados pela emissora apenas jogos, em horários definidos, como se fossem a programação do canal.

Os dois modelos apresentados acima não se utilizam de duas das principais características da TV Digital: som e imagem de alta qualidade. Sendo assim, outra forma de explorar os jogos na TV é correlacionando-os com os programas que estão sendo exibidos. Neste contexto, o maior desafio esta na concepção de como seriam tais jogos, pois nenhuma das plataformas de jogos tradicionais (PC, consoles, celular) possui esta característica, de forma que os GameDesigners que são responsáveis por idealizar os jogos têm que se esforçar para elaborar jogos voltados a essa nova plataforma.

Outra possibilidade é a utilização da TV como terminal para jogos nas nuvens (GameClouding). Neste cenário o motor (engine) do jogo não mais executa no cliente, mas sim na nuvem, onde é realizada a maior parte dos processos mais intensivos do ponto de vista computacional. O cliente do jogo resume-se ao envio de eventos para a nuvem e o recebimento de um stream de vídeo. Os eventos recebidos dos clientes, juntamente com o estado do mundo, permitem a renderização da aparência do ambiente virtual também na nuvem, e as imagens resultantes são enviadas para os clientes via streaming. Desta forma, o jogo possui uma dependência muito menor do dispositivo cliente específico do usuário.

3. GingaNCL

O GingaNCL, inovação completamente brasileira, desenvolvido pela PUC - Rio, é o subsistema lógico responsável pelo processamento de aplicações declarativas NCL (Nested Context Language). NCL (Soares & Barbosa, 2009) é uma linguagem de aplicação XML que separa conteúdo e estrutura, facilitando o suporte a múltiplos dispositivos, a interatividade, sincronismo espaço-temporal, etc. Além desta linguagem também é usada a linguagem de script Lua (Ierusalimschy, 2006), para quando for necessária a geração dinâmica de conteúdo.

Além disso, o GingaNCL é recomendação ITU-T para padrão de televisão digital interativa (ITU-T J.200) e plataformas IPTV (ITU-T H.761).

3.1 Ambiente de desenvolvimento

Para a implementação do tutorial, foi utilizado o ambiente de desenvolvimento Eclipse. O Eclipse é uma IDE (Integrated Development Environment) desenvolvida em Java, com código aberto para a construção de programas de computador. O projeto Eclipse foi iniciado na IBM que desenvolveu a primeira versão do produto e doou-o como software livre para a comunidade. Por ter seu código aberto, muitos plugins para desenvolvimento de aplicações são produzidos para uso no Eclipse, e dentre estes plugins estão alguns utilizados para o desenvolvimento de aplicações para TV Digital, o NCL-Eclipse e o LuaEclipse.

O NCL-Eclipse tem o objetivo de agilizar o desenvolvimento de aplicações para TV digital Interativa em NCL, pois possui funcionalidades como a correção automático de erros sintáticos e semânticos de código, como também permite executar o código da aplicação diretamente no Set-Top-Box virutal, agilizando o desenvolvimento.

O LuaEclipse ajuda no desenvolvimento de aplicações para linguagem Lua, possibilitando ao desenvolvedor um ambiente com syntax highlight, code completion, verificação de erros de compilação, agrupamento de código e comentários e um executor de código para testes.

3.2 Nested Context Language (NCL)

Criada no Laboratório TeleMídia da PUC-Rio, a linguagem NCL - Nested Context Language - é uma linguagem declarativa para autoria de documentos hipermídia baseados no modelo conceitual NCM - Nested Context Model.

O modelo da linguagem NCL visa não apenas o suporte declarativo à interação do usuário, mas ao sincronismo espacial e temporal em sua forma mais geral, tratando a interação do usuário como um caso particular. NCL visa também o suporte declarativo a adaptações de conteúdo e de formas de apresentação de conteúdo; o suporte declarativo a múltiplos dispositivos de exibição, interligados através de redes residenciais (HAN – Home Area Networks) ou mesmo em área mais abrangente; e a edição/produção da aplicação em tempo de exibição, ou seja, ao vivo. Como esses são os focos da maioria das aplicações para TV digital, NCL se torna a opção preferencial no desenvolvimento da maioria de tais aplicações. Para os poucos casos particulares, como por exemplo, quando a geração dinâmica de conteúdo é necessária, NCL provê o suporte de sua linguagem de script Lua.

NCL define como objetos de mídia são estruturados e relacionados, no tempo e espaço. Como uma linguagem de cola, NCL não restringe ou prescreve os tipos de conteúdo dos objetos de mídia. Nesse sentido, podemos ter como objetos de mídia NCL: objetos perceptuais de imagem, de vídeo, de áudio, e de texto; objetos com código imperativo (Lua, entre outros); e objetos com código declarativo (XHTML, entre outros), incluindo objetos com código NCL aninhados. Quais objetos de mídia têm suporte depende apenas dos exibidores de objetos de mídia que estão acoplados ao exibidor (player) NCL. Essa definição depende do Sistema de TV Digital onde o Ginga será utilizado. Por exemplo, no caso do ISDB-TB, objetos com código imperativo podem ser Java (XLet) ou Lua (NCLua); objetos com código declarativo podem ser XHTML (com as funcionalidades mínimas definidas pelas Normas do Sistema) ou outras aplicações NCL embutidas. (nlc.org)

3.3 Lua

Lua é a linguagem de script de NCL. Lua é uma linguagem de programação imperativa eficiente, rápida e leve, projetada para estender aplicações. Lua combina uma sintaxe simples para programação imperativa com construções poderosa para descrição de dados baseadas em tabelas associativas e em semântica extensível. Lua é tipada dinamicamente, é interpretada e tem gerenciamento automático de memória, com coleta de lixo incremental. Essas características fazem de Lua uma linguagem ideal para configuração, automação (scripting) e prototipagem rápida (geração rápida de aplicações). (lua.org).

3.4 NCLua

Denominamos NCLua os módulos acessíveis a partir da linguagem Lua específicos da plataforma Ginga. Um fato comum entre estes novos módulos é o de serem extremamente simplistas. Estes módulos não tentam de forma nenhuma definir um modelo de programação ou esgotar em APIs todo possível tipo de uso. O que eles oferecem de maneira geral são acessos a primitivas disponíveis em qualquer sistema computacional. (Sant'Ann)

Algumas vantagens dessa abordagem:

- Não define uma forma amarrada de desenvolvimento de aplicações que pode não fazer sentido com a evolução do padrão e do mercado.
- A definição é pequena, sucinta e de fácil entendimento.
- Atualmente são aproximadamente 30 funções ou tipos.
- A implementação é bastante óbvia, com um mapeamento próximo de 1:1 com o sistema de

implementação do middleware, portanto pequena e com pouca margem a comportamentos não padronizados.
- Fomenta o desenvolvimento de frameworks, game engines, etc, portáveis e com utilidades diferentes sobre a mesma API.
- Por ter uma interface mais baixo-nível permite o desenvolvimento de novos players e até mesmo aplicações nativas portáveis entre set-tops.

3.4.1 Módulo Canvas (ABNT 15606-2)

O módulo canvas oferece uma API para desenhar primitivas gráficas e manipular imagens. Suas principais funcionalidades são:

3.4.1.1 Construtores

Através de qualquer canvas é possível criar novos canvas e combiná-los através de operações de composição. Os construtores disponíveis são:

canvas:new (image_path: string)

Argumentos:
- image_path: Caminho da imagem

Valor de retorno:
- canvas: Canvas representando a imagem

Descrição
- Retorna um novo canvas cujo conteúdo é a imagem recebida como parâmetro. O novo canvas deve obrigatoriamente manter os aspectos de transparência da imagem original.

canvas:new (width, height: number)

Argumentos:
- Width: Largura do canvas
- Height: Altura do canvas

Valores de retorno
- canvas: Novo canvas;

Descrição:
- Retorna um novo canvas com o tamanho recebido. Inicialmente todos os pixels devem ser transparentes, obrigatoriamente.

3.4.1.2 Atributos

Todos os métodos de atributos possuem o prefixo attr e servem tanto para ler quanto para alterar um atributo.

Quando o método é chamado sem parâmetros de entrada o valor corrente do atributo é retornado, em contra-partida, quando é chamado com parâmetros, esses devem ser os novos valores do atributo.

canvas:attrSize ()

Valores de retorno:
- Width: Largura do canvas;
- Height: Altura do canvas;

Descrição:
- Retorna as dimensões do canvas;

canvas:attrColor (R, G, B, A: number)

Argumentos:
- R: Componente vermelha da cor;
- G: Componente verde da cor;
- B: Componente azul da cor;
- A: Componente alpha da cor;

Valores de retorno (caso não sejam passados parâmetros):
- R: Componente vermelha da cor;
- G: Componente verde da cor;
- B: Componente azul da cor;
- A: Componente alpha da cor;

Descrição
- Altera a cor do canvas. As cores são passadas em RGBA, onde A varia de 0 (totalmente transparente) a 255 (totalmente opaco). O valor inicial é ‘0,0,0,255’ (preto).

canvas:attrFont (face: string; size: number; style: string)

Argumentos:
- Face: Nome da fonte;
- Size: Tamanho da fonte;
- Style: Estilo da fonte;

Valores de retorno (caso nenhum argumento seja passado):
- Face: Nome da fonte, ou o caminho e o nome do tipo da fonte;
- Size: Tamanho da fonte;
- Style: Estilo da fonte;

Descrição:
- Altera a fonte do canvas.
- As seguintes fontes devem obrigatoriamente estar disponíveis: 'Tiresias’;
- O tamanho é em pixels e representa a altura máxima de uma linha escrita com a fonte escolhida;
- Os estilos possíveis são: 'bold', 'italic' ou 'bold-italic'. O valor nil assume que nenhum dos estilos será usado;
- Qualquer valor passado não suportado deve obrigatoriamente gerar um erro;
- O valor inicial da fonte é indeterminado;

canvas:attrCrop (x, y, w, h: number)

Argumentos:
- X: Coordenada da região de crop;
- Y: Coordenada da região de crop;
- w: Largura da região de crop;
- h: Altura da região de crop;

Valores de retorno (caso nenhum argumento seja passado):
- X: Coordenada da região de crop;
- Y: Coordenada da região de crop;
- W: Largura da região de crop;
- H: Altura da região de crop;

Descrição:
- Altera a região de crop do canvas;
- Apenas a região configurada passa a ser usada em operações de composição;
- O valor inicial é o canvas inteiro;
- O canvas principal não pode ter seu valor alterado, pois é controlado pelo formatador NCL;

canvas:attrOpacity (opacity: number)

Argumento:
- Opacity: Novo valor de opacidade do canvas

Valor de retorno (caso nenhum argumento seja passado):
- Opacity: Opacidade do canvas;

Descrição:
- Altera a opacidade do canvas;
- O valor da opacidade varia entre 0 (transparente) e 255 (opaco);
- O canvas principal não pode ter seu valor alterado;

canvas:attrRotation (degrees: number)

Argumento:
- Degrees: Rotação do canvas em graus;

Valor de retorno
- Degrees: Rotação do canvas em graus;

Descrição:
- Configura o atributo de rotação do canvas, que deve ser múltiplo de 90 graus.
- O canvas principal não pode ter seu valor alterado;

canvas:attrScale (w, h: number)

Argumentos:
- W: Largura de escalonamento do canvas;
- H: Altura de escalonamento do canvas

Valor de retorno
- W: Largura de escalonamento do canvas
- H: Altura de escalonamento do canvas

Descrição
- Escalona o canvas com nova largura e altura;
- Um dos valores pode ser true, indicando que a proporção do canvas deve ser mantida;
- O atributo de escalonamento é independente do atributo de tamanho, ou seja, o tamanho do canvas é mantido;
- O canvas principal não pode ter seu valor alterado;

3.4.1.3 Primitivas

O modulo canvas permite também o desenho de diversas primitivas como:
- Linhas;
- Retângulos;
- Polígonos;
- Elipses;

3.4.1.4 Outras Funcionalidades

canvas:drawText (x,y: number; text: string)

Argumentos:
- X: Coordenada do texto;
- Y: Coordenada do texto;
- Text: Texto a ser desenhado;

Descrição:
- Desenha o texto passado na posição (x,y) do canvas utilizando a fonte configurada em canvas:attrFont();

canvas:clear ([x, y, w, h: number])

Argumentos:
- X: Coordenada da área de clear;
- Y: Coordenada da área de clear;
- W: Largura da área de clear;
- H: Altura da área de clear;

Descrição
- Limpa o canvas com a cor configurada em attrColor.
- Caso não sejam passados os parâmetros de área, assume-se que o canvas inteiro será limpo;

canvas:flush ()

Descrição:
- Atualiza o canvas após operações de desenho e de composição.
- É suficiente chamá-lo apenas uma vez após uma seqüência de operações.

canvas:compose (x, y: number; src: canvas;)

Argumentos:
- X: Posição da composição;
- Y: Posição da composição;
- Src: Canvas a ser composto;

Descrição;
- Faz sobre o canvas (canvas de destino), em sua posição (x,y), a composição pixel a pixel com src (canvas de origem);

canvas:pixel (x, y, R, G, B, A: number)

Argumentos:
- X: Posição do pixel;
- Y: Posição do pixel;
- R: Componente vermelha da cor;
- G: Componente verde da cor;
- B: Componente azul da cor;
- A: Componente alpha da cor;

Valores de retorno (caso apenas x e y sejam passados):
- R: Componente vermelha da cor;
- G: Componente verde da cor;
- B: Componente azul da cor;
- A: Componente alpha da cor;

Descrição:
- Altera a cor de um pixel do canvas.

3.4.2 Módulo Event (ABNT 15606-2)

O módulo event permite que aplicações NCLua comuniquem-se com o middleware através de eventos (eventos NCL, TCP e de teclas).

Durante a inicialização do NCLua, antes de se tornar orientado a eventos, o script deve registrar pelo menos uma função de tratamento de eventos. A partir de então, qualquer ação tomada pela aplicação é somente em resposta a um evento recebido por essa função tratadora. A função *event.register* efetua o registro de tratadores.

Durante sua execução, um NCLua também pode enviar eventos para se comunicar com o ambiente. Assim é possível que o NCLua envie dados pelo canal de interatividade, sinalize que sua execução terminou, etc. A função event.post efetua o envio de eventos.

Um NCLua também pode usufruir desse mecanismo de eventos internamente, através de eventos da classe user. Tais eventos, portanto, são postados e recebidos pelo próprio NCLua.

Primeiro veremos as funções deste módulo e em seguida as classes de eventos que podem ser tratadas pelo NCLua.

3.4.2.1 Funcionalidades

event.post ([dst: string]; evt: event)

Argumentos:
- dst: Destinatário do evento;
- evt: Evento a ser postado;

Valores de retorno:
- sent: Se o evento for enviado com sucesso;
- err_msg: Mensagem de erro em caso de falha;

Descrição:
- Posta o evento passado;
- O parâmetro "dst" é o destinatário do evento e pode assumir os valores "in" (envio para a própria aplicação) e "out" (envio para o formatador NCL). O valor default é ‘out’.

event.timer (time: number, f: function)

Argumentos:
- time: Tempo em milissegundos;
- f: Função de callback;

Valor: de retorno;
- unreg: Função para cancelar o timer;

Descrição:
- Cria um timer que expira após time (em milissegundos) e então chama a função f;
- A assinatura de f é simples, sem recebimento de parâmetros:
  function f () end
- O valor de 0 milissegundos é válido. Nesse caso, event.timer() deve, obrigatoriamente, retornar imediatamente e f deve ser chamada assim que possível;

event.register (f: function;)

Argumentos:
- f: Função de callback;

Descrição:
- Registra a função passada como um listener de eventos, isto é, sempre que ocorrer um evento, f será chamada;
- A função f é, assim, a função de tratamento de eventos (function “handler”);
- A assinatura de f é:
  function f (evt) end -> handled: boolean
- Onde evt é o evento que, ao ocorrer, ativa a função.

event.unregister (f: function)

Argumentos:
- f: Função de callback;

Descrição:
- Tira do registro a função passada como um listener, isto é, novos eventos não serão mais passados a f;

event.uptime () -> ms: number

Valores de retorno:
- ms: Tempo em milissegundos;

Descrição
- Retorna o número de milissegundos decorridos desde o início da aplicação;

3.4.2.2 Classes

A função event.post() e o "handler" registrado em event.register() recebem eventos como parâmetros. Um evento é descrito por uma tabela Lua normal, onde o campo class é obrigatório e identifica a classe do evento.

As seguintes classes são definidas:

Classe key: esta classe de eventos é utilizada para representar o uso do controle remoto pelo usuário. Para esta classe não faz sentido que o NCLua gere eventos, uma vez que o controle remoto é um dispositivo unicamente de entrada.

Sua estrutura segue o seguinte formato:

evt = { class='key', type: string, key: string}

Onde:
- type pode ser 'press' ou 'release';
- key é o valor da tecla em questão.

Um exemplo de uso é:

evt = { class='key', type='press', key="0"}

Pela classe key, as seguintes teclas podem ser utilizadas:
- Teclas Coloridas: RED, GREEN, BLUE, YELLOW;
- Teclas Numéricas: 0,1,2,3,4,5,6,7,8,9;
- Setas: CURSOR_DOWN, CURSOR_UP, CURSOR_LEFT, CURSOR_RIGHT;
- Teclas de Configuração: MENU, INFO, ENTER;

Classe ncl: as relações entre os nós de mídia NCL são baseadas em eventos. Lua tem acesso a esses eventos através desta classe.

Os eventos podem agir nas duas direções, isto é, o formatador pode enviar eventos de ação para mudar o estado do exibidor Lua, e este, por sua vez, pode disparar eventos de transição para indicar mudanças de estado. Nos eventos, o campo type deve obrigatoriamente assumir um dos três valores: 'presentation', 'selection' ou 'attribution'.

Os eventos podem ser direcionados a âncoras específicas ou ao nó como um todo, isto é identificado no campo label, que assume o nó inteiro, quando ausente.

No caso de um evento gerado pelo formatador, o campo action deve obrigatoriamente ter um dos valores: 'start', 'stop', 'abort', 'pause' ou 'resume'

Classe edit: esta classe permite a edição pelo código lua do documento NCL. Entretanto, há uma diferença entre os comandos de edição provenientes dos eventos do fluxo de vídeo da emissora e os comandos de edição realizados pelos scripts Lua (objetos NCLua). Os primeiros alteram não somente a apresentação de um documento NCL, mas também a especificação de um documento NCL. Ou seja, no final do processo um novo documento NCL é gerado, incorporando todos os resultados da edição. Por outro lado, os comandos de edição provenientes dos objetos de mídia NCLua alteram somente a apresentação do documento NCL. O documento original é preservado durante todo o processo de edição.

Assim como nas outras classes de evento, um comando de edição é representado por uma tabela Lua. Todo evento deve obrigatoriamente carregar o campo command: uma string com o nome do comando de edição.

EXEMPLO

```
evt = {
    command = 'addNode',
    compositeId = 'someId',
    data = '<media>...',
}
```

Através do NCLEdit podemos então adicionar dinamicamente mídias de áudio e executá-las através do código lua.

Classe tcp: o uso do canal de interatividade é realizado por meio desta classe de eventos. Utilizando esta classe é possível:

- Criar uma conexão com uma máquina remota:

evt = { class='tcp', type='connect', host=addr, port=number, [timeout=number] }

- Obter o status da conexão:

evt = { class='tcp', type='connect', host=addr, port=number, connection=identifier, error=<err_msg>}

- Enviar dados:

evt = { class='tcp', type='data', connection=identifier, value=string, [timeout=number] }

- Receber dados:

evt = { class='tcp', type='data', connection=identifier, value=string, error=msg }

Classe si: a classe de eventos 'si' permite acesso a um conjunto de informações multiplexadas em um fluxo de transporte e transmitidas periodicamente por difusão.

O processo de aquisição das informações deve obrigatoriamente ser realizado em dois passos:

1) uma requisição realizada por uma chamada à event.post(), que não bloqueia a execução do script;

2) um evento, posteriormente repassado aos tratadores de eventos registrados do script NCLua, cujo campo data contém um conjunto de subcampos que depende do tipo da informação requisitada, e que é representado por uma tabela lua.

3.4.3 Outros Módulos (ABNT 15606-2)

O módulo settings exporta uma tabela com variáveis definidas pelo autor do documento NCL e variáveis de ambiente reservadas.

Já o módulo persistent exporta uma tabela com variáveis persistentes, que estão disponíveis para manipulação apenas por objetos imperativos.

3.4.4 Sincronização com a Emissora

A emissora de TV pode enviar via o fluxo de vídeo dois tipos de eventos:

- Do it now: assim que o evento chega, ele é executado;
- Schedule: o evento é agendado para executar em um tempo X da execução do vídeo;

Como o NCLEdit nos permite adicionar nós NCL em geral (regiões, medias, links) e atribuir valores de propriedades, podemos obter os valores destas propriedades pelo módulo Settings e classe de eventos SI, podemos utilizar a atualização dos valores das propriedades para sincronizar e mandar valores para nossa aplicação Lua.

Para isso é necessário ficar obtendo informações da tabela e observando se houve mudanças de valor nas propriedades

O problema desta solução, é que hoje, é impossível testá-la, pois os eventos de edição vindos da emissora não estão implementados, nem o módulo Settings e SI, nos ambientes de testes.

4. Técnicas para Jogos 2D

Com as principais funcionalidades do GingaNcl destacadas, podemos utilizá-las na implementação de técnicas para programação de jogos. Aqui falaremos como algumas podem ser desenvolvidas, porém as possibilidades não estão limitadas a elas.

4.1 Loop de Animação

Utilizando o evento de temporização (event.timer), é possível criar um loop de animação. Isso pode ser feito criando uma função que ao seu final crie um *timer* que passe comoparâmetro a própria função, desta forma a função entrará em *loop* com um intervalo de tempo entre as chamadas de acordo o tempo passado para a função timer.

Este *loop* é bem simplificado, podendo ser melhorado para aceitar interrupções através da chamada da função para anular o a chamada do timer. Além disso, pode ser calculado o erro de tempo que cada iteração esta gerando, de forma que armazenando este erro pode-se analisar se é preciso esperar mais ou menos tempo no próximo *loop* para que seja mantida taxa constante, de frames ou de atualizações. Para o calculo deste erro, basta usar a função *uptime* do módulo *event* antes e depois de cada iteração, e utilizar este tempo para o calculo do tempo para a chamada do próximo *loop.*

4.2 Colisão

Diversas técnicas de detecção podem ser utilizadas com o Ginga. Técnicas de comparação de quadrados e círculos podem ser facilmente utilizadas devido a possibilidade de controle das posições dos objetos .

Porém técnicas mias avançadas taambém podem ser aplicadas, como a de PixelPerfect, que é uma técnica de colisão que consiste em analisar cada pixel de cada imagem para verificar se a parte do desenho da imagem colide com a outra. É uma técnica que em todos os casos a colisão é detectada. Apesar disso é uma das técnicas com menor desempenho e por isso só deve ser utilizada quando realmente houver necessidade de um teste de colisão muito preciso. Ela geralmente é usada em testes com figuras de formas mais complexas. Como temos a possibilidade de pegar os valores de um pixel (através da função getPixel do módulo canvas) da imagem, podemos saber se aquela parte do desenho é o fundo da imagem ou realmente faz parte dela. Assim é possível pegar todos os valores da imagem e verificar se ocorre colisão com outra.

4.3 Sprites

Para a criação de animações utilizando a técnica de sprites, criamos um canvas que recebe na sua criação (canvas:new(path)) a imagem que possui nossa sequência de sprites. Se desenharmos o canvas recém

criado na tela, todos os frames da animação serão desenhados. Desta forma precisamos "cortar" no nosso canvas apenas o pedaço de imagem que corresponde ao frame atual da animação. Para isto utilizamos a função *attrCrop* do módulo *canvas*. Para seus argumentos utilizamos a posição inicial do primeiro frame (x , y) e a largura e altura dos frames (largura, altura), ou seja os quatro parâmetros são x, y, altura e largura. Para passarmos ao próximo frame adicionamos ao valor inicial de x e y o valor da largura e altura respectivamente. Após isto chamamos novamente o *attrCrop* com os novos valores de x e y.

4.4 Tiles

A implementação da técnica de uso de tiles é bem simples com as chamadas NCLua. Inicialmente carregamos um tile em um novo *canvas*. Desta forma o *canvas* representará o nosso tile. Para desenhar o tile em um cenário ,no nosso caso no *canvas* principal, fazemos a composição do tile com o *canvas* principal utilizando a função *compose*. Assim o nosso tile será desenhado no *canvas* do cenário, e podemos repetir esta operação diversas vezes em várias posições de forma a gerar um cenário maior a partir de um tile.

Para aprimorar nossa técnica, podemos adicionar um *tileMapping*, de forma que o nosso *canvas* com o tile não possuiria apenas um tile, mas sim todo um mapa de tiles. De forma semelhante ao que fizemos com a técnica de Sprites (utilizando o *attrCrop*), podemos recortar a parte do mapa de tiles que corresponda apenas ao tile que desejamos utilizar naquele momento, podendo durante o desenho mudar a área de recorte para o tile desejado.

4.5 Double Buffer

Consiste em usar um buffer de memória para resolver os problemas de cintilação (flicker) associados com as operações de pintura múltiplas. Quando o buffer duplo está habilitado, todas as operações de pintura são processadas pela primeira vez para um buffer de memória em vez de a superfície de desenho na tela. Depois de todas as operações de pintura serem concluídas, o buffer de memória é copiado diretamente para a superfície de desenho associado a ele. Como apenas uma operação de gráficos é realizada na tela, o problema da imagem piscando associada com as operações de pintura é eliminado.

Utilizando o GingaNCL, isto pode ser implementado da seguinte maneira: com o *canvas* principal funcionando como o buffer, ou seja, todas as outras imagens são novos *canvas*. A cada ciclo de desenho limpamos o *canvas* principal com a função *clear()*, depois compomos todas as outras imagens criadas no canvas principal e por fim atualizamos o *canvas* principal com o *flush()*. Desta forma a tela só será atualizada quando o comando *flush* for executado, havendo assim a cada ciclo apenas uma atualização total do *display*(TV).

5. Perspectivas

A interatividade hoje não se mostra um caso de sucesso no Brasil. Parte das aplicações não trás a nova experiência em assistir TV interagindo com a programação, mas apenas pequenas aplicações cuja experiência de uso pode ser encontrada em um simples acesso ao computador ou em uma busca no celular. A real experiência interatividade só será atingida quando produtores de conteúdo fizerem as aplicações para TV com verdadeira interatividade com a programação, tornando estas aplicações únicas para a plataforma e consequentemente atraindo a atenção dos telespectadores para utilizar esta inovação.

Além da falta de conteúdo, as plataformas de desenvolvimento comerciais e acadêmicas possuem problemas por não implementar todas as funcionalidades previstas na norma, funcionalidades estas que vão desde funções gráficas (como escalonamento de figuras) à impossibilidade de sincronizar as aplicações criadas com o um fluxo de vídeo broadcast.

Apesar destes obstáculos, existe muito a ser percorrido, diversos outros pontos positivos nos estimulam a seguir este caminho, como declarado por (Soares, 2011) em mensagem a comunidade Ginga:

- "Foram criadas, só no Brasil, 13 empresas para o desenvolvimento do Ginga e ferramentas. Algumas dessas empresas já são hoje de médio porte. Elas empregam, no total, mais de 1.000 profissionais de nível superior";
- "Na Comunidade Ginga do Software Público (Software Público, 2011) existem mais de 54 prestadoras de serviço cadastradas";
- "Na Argentina o Ginga é uma realidade. Mais de 1.200.000 set-top boxes com Ginga-NCL e licitação em marcha para mais 3.000.000, inclusive para notebooks, etc";
- "A Comunidade Ginga no Brasil conta com mais de 10.500 brasileiros desenvolvedores e 1.000 estrangeiros (mais de 30 países). Além dela, temos as comunidades Ginga na Argentina, Chile, Peru, Equador e Bolívia";
- "Ginga-NCL é padrão mundial ITU-T. Nunca o Brasil teve um padrão na área de TICs na sua íntegra";

Muito ainda deve ser feito para que a interatividade na TV se torne um atrativo para os telespectadores, mas com as possibilidades internacionais, recomendações ITU-T para TV Digital e IPTV podemos criar diversas expectativas para o Ginga. Assim, investir em seu estudo, descobrir novas possibilidades e dominar esta plataforma devem ser consideras opções para todos.

Referências

WINK DINK AND YOU. DISPONÍVEL EM < HTTP://WWW.TVPARTY.COM/REQUESTED2.HTML >. ACESSADO EM: MAIO, 2011.

VOCÊ DECIDE. DISPONÍVEL EM < HTTP://BIT.LY/MB420L >. ACESSADO EM: MAIO, 2011.

BIG BROTHER. DISPONÍVEL EM < HTTP://GLO.BO/LFMJKZ >. ACESSADO EM: MAIO, 2011.

SCHELL J.. THE ART OF GAME DESIGN. 2008.

JENS F. JENSEN. 2005.INTERACTIVE TELEVISION: NEW GENRES, NEW FORMAT, NEW CONTENT. PROCEEDINGS OF THE SECOND AUSTRALASIAN CONFERENCE ON INTERACTIVE ENTERTAINMENT.

HELIÖ S. E JÄRVINEN A.. FITV: GAMES AND GAMING FOR INTERACTIVE DIGITAL TELEVISION. NOVEMBER 2003. UNIVERSITY OF TAMPERE HYPERMEDIA LABORATORY.

SOARES, L. F. G. ; BARBOSA, SIMONE DINIZ JUNQUEIRA . PROGRAMANDO EM NCL. 1. ED. RIO DE JANEIRO: ELSEVIER - CAMPUS, 2009. 483 P.

IERUSALIMSCHY, R. . PROGRAMMING IN LUA, SECOND EDITION. 2. ED. RIO DE JANEIRO: LUA.ORG, 2006. 326 P.

Luiz Fernando Gomes Soares. TV DIGITAL INTERATIVA NO BRASIL NÃO É UM SONHO. MENSAGEM RECEBIDA POR < TVINTERATIVA@GOOGLEGROUPS.COM> EM 09 DE SETEMBRO DE 2011.

SOFTWARE PÚBLICO. DISPONÍVEL EM < WWW.SOFTWAREPUBLICO.GOV.BR/GINGA-A-TV-DIGITAL-NO-SOFTWARE-PUBLICO>. ACESSADO EM 13 DE SETEMBRO DE 2011

ITU-T J.200: WORLDWIDE COMMON CORE - APPLICATION ENVIRONMENT FOR DIGITAL INTERACTIVE TELEVISION SERVICES. DISPONÍVEL EM <HTTP://ENGINEERS.IHS.COM/DOCUMENT/ABSTRACT/SEMFGBAAAAAAAAAA>. ACESSADO EM 10 DE SETEMBRO DE 2011.

ITU-T H.761: NESTED CONTEXT LANGUAGE (NCL) AND GINGA-NCL FOR IPTV. DISPONIVEL EM <HTTP://WWW.ITU.INT/ITU-T/AAP/AAPRECDETAILS.ASPX?AAPSEQNO=1894>. ACESSADO EM 10 DE SETEMBRO DE 2011.

NCL.ORG. NESTED CONTEXT LANGUAGE. DISPONÍVEL EM <HTTP://WWW.NCL.ORG.BR/PT-BR/INICIO>. ACESSADO EM 9 DE SETEMBRO DE 2011.

LUA.ORG. O QUE É LUA?. DISPONÍVEL EM <HTTP://WWW.LUA.ORG/PORTUGUES.HTML>. ACESSADO EM 9 DE SETEMBRO DE 2011.

SANT'ANN F. . MANUAL DE REFERÊNCIA DE NCLUA. DISPONÍVEL EM <HTTP://WWW.LUA.INF.PUC-RIO.BR/~FRANCISCO/NCLUA/REFERENCIA/INDEX.HTML>. ACESSADO EM 9 DE SETEMBRO DE 2011.

ABNT 15606-2 (NORMA BRASILEIRA ABNT NBR 15606-2:2011). DISPONÍVEL EM <HTTP://WWW.DTV.ORG.BR/DOWNLOAD/PT-BR/ABNTNBR15606-2_2011ED2.PDF>.