Bibliotecas e Aplicações em TV Digital

Yan Victor Chaves Pinheiro, Mário Angel Praia Garcia

Faculdade FUCAPI – FACULDADE FUCAPI (INSTITUTO DE ENSINO SUPERIOR FUCAPI)

{y.victor66,marioangelpg}@gmail.com

Abstract. *This article describes the development of digital television applications as well as the approaches of Ginga and its development libraries. It also features the development of a game created in the LUA script called "Monkey and Banana Game" which aims to demonstrate development in language and how simple is its operation and programming for digital television.*

Resumo. *Este artigo descreve o desenvolvimento de aplicações para televisão digital assim como as abordagens do Ginga e suas bibliotecas de desenvolvimento, também consta o desenvolvimento de um jogo criado no script LUA chamado "Jogo do macaco e da banana" que tem como objetivo demonstrar o desenvolvimento na linguagem e quão simples é o seu funcionamento e a programação para televisão digital.*

1. Introdução

Atualmente o mundo evolui tecnologicamente de maneira gradual, ou seja, tudo aquilo que é acessado, compartilhado e distribuído através da internet torna-se possível devido às inovações de *hardware* e *software* existentes, uma vez que os aparelhos de telefonia móvel se tornam de fácil manuseio ao longo de cada geração e os televisores também ao passar do tempo acabam cada vez mais interagindo com os seus usuários.

Os desenvolvedores estão ganhando mais espaço devido ao seu conhecimento e também pela sua criatividade em criar meios para facilitar o cotidiano. No Brasil não seria diferente com chegada da transmissão da TV Digital em 2007 (Dantas, 2018) o país criou mais um nicho na área do desenvolvimento e a partir disso nasceu o Ginga. O modelo estático da televisão de um modo geral começou a ser irrelevante conforme a TV Digital foi ganhando espaço no mundo, interações com os outros aparelhos e aplicativos como a Netflix, Spotify para Android e IOS. Entretanto estes aplicativos na maioria das vezes possuem o código restrito à suas empresas distribuidoras e o desenvolvedor não tem acesso a estes códigos-fonte podendo haver até penalizações judiciais caso isso ocorra.

O ginga é um *middleware* que está padronizado de acordo com a recomendação do setor de normatização das telecomunicações (ITU-T) (Cruz, Moreno, & Soares, 2008). Disposto de um código de software livre é possível criar aplicações e interações para a TV digital de maneira aberta utilizando serviços IPTV (*Internet Protocol Television*) utilizando o sistema Nipo-Brasileiro de TV digital terrestre (ISDB-Tb) [Soares, 2008].Este artigo descreve o desenvolvimento para televisão digital e as abordagens do Ginga, também consta o desenvolvimento de um jogo criado no script LUA chamado "Jogo do macaco e da banana"

que tem como objetivo demonstrar o desenvolvimento na linguagem e quão simples é o seu funcionamento e a programação para televisão digital.

O sistema também permite que os espectadores instalem aplicativos de jogos e redes sociais na TV por meio da internet ou de lojas de aplicativos integradas ao Ginga. A instalação de aplicativos na TV ocorre de maneira similar às TV com conexão à internet, fabricadas por diversas fabricantes. Alguns aplicativos, no entanto, podem demandar o envio de informações para um servidor, o que só pode acontecer em TVs conectadas à internet.

2. Fundamentação Teórica

2.1 Middleware

O middleware é o software que se encontra entre o sistema operacional e os aplicativos nele executados. Funcionando de forma essencial como uma camada oculta de tradução, o middleware permite a comunicação e o gerenciamento de dados para aplicativos distribuídos. Muitas vezes, o middleware é chamado de "encanamento", uma vez que ele conecta dois aplicativos para que os dados e bancos de dados possam ser facilmente transportados através do "cano". O uso do middleware permite que os usuários executem solicitações como enviar formulários em um navegador da Web ou permitir que o servidor Web apresente páginas dinâmicas da web com base no perfil de um usuário.

Exemplos comuns de middleware incluem middleware de banco de dados, middleware de servidor de aplicativos, middleware orientado a mensagens, middleware de web e monitores de processamento de transações.

Os benefícios que o middleware pode trazer:

- **Comunicação com diferentes aplicações**: promover a integração das diferentes aplicações utilizadas por uma empresa. Sendo Assim a TI pode utilizar tecnologia localizadas em outros locais, assim como adotar soluções na nuvem.
- **Processamento de Dados**: O processamento é otimizado e as informações captadas podem ser usadas como insights poderosos.
- **Desenvolvimento de aplicações**: A camada de middleware facilita o desenvolvimento das aplicações complexas utilizadas pelas corporações. Dessa forma, a criação e gestão dos aplicativos se tornam mais fácil e eficiente, permitindo que os recursos e esforços da equipe de TI sejam direcionados para outras áreas da empresa. Isso contribui para melhorar os negócios e as decisões estratégicas de forma pratica e rápida. Desta forma, as organizações conseguem se focar em tarefas mais importantes de seus negócios, aprimorando seus resultados.

2.2 Arquitetura Ginga

A arquitetura do Ginga pode ser dividida em três módulos principais: Ginga-CC, Ginga-NCL e Ginga-J. Os dois últimos módulos compõem a camada de Serviços Específicos do Ginga.

Ginga-J: O ambiente imperativo Ginga-J oferece suporte a aplicações desenvolvidas usando a linguagem Java. Ginga-J é dividido em três módulos: a máquina virtual Java; o núcleo e

suas APIs, também chamadas APIs verde do Ginga-J; e o módulo responsável pelo suporte às APIs específicas do Ginga-J, chamadas de APIs amarela e vermelha do Ginga-J.

Ginga-J seguirá a especificação da Norma ABNT NBR 15606-4. As APIs verde do núcleo são as responsáveis por manter o sistema compatível o máximo possível com os sistemas americano e europeu.

As APIs específicas do Ginga que podem ser exportadas para outros sistemas são chamadas de amarelas. Entre elas estão aquelas que proveem suporte a múltiplos usuários, a múltiplos dispositivos e a múltiplas redes. Estão também aquelas que oferecem suporte às aplicações que podem ser recebidas, armazenadas e executadas em um tempo futuro.

Ginga-NCL: A arquitetura modular do Ginga-NCL para dispositivos portáteis, que é dividida em dois subsistemas lógicos: a Máquina de Apresentação Ginga-NCL e o Núcleo Ginga. No Núcleo Ginga, o módulo Sintonizador é responsável por receber conteúdo de TV Digital para portáteis transmitido por provedores de conteúdo. As aplicações interativas podem chegar ao dispositivo portátil multiplexadas nesse conteúdo, ou por outra interface de rede (por exemplo, canal de dados oferecido pela operadora de celular, conexão Bluetooth ou IEEE 802.11 etc.). No primeiro caso, a aplicação é obtida através de um processamento, realizado pelo módulo Processador de Dados, sobre o conteúdo recebido. No segundo caso, um componente de Transporte foi definido para controlar protocolos e interfaces de rede e atender a demanda da Máquina de Apresentação por conteúdo e aplicações. Para gerenciar o armazenamento das aplicações de TV e o conteúdo que essas aplicações referenciam, o módulo de Persistência foi definido. (Cruz, Moreno, & Soares)

Ginga-CC (Ginga Common Core): Ginga-CC é responsável por oferecer os serviços básicos ligados à plataforma do dispositivo receptor ao Ambiente de Apresentação, às aplicações residentes e a um possível ambiente imperativo adicional. Ginga-NCL é um subsistema lógico capaz de controlar o ciclo de vida das aplicações NCL e suas execuções.

Os módulos que compõem os dois subsistemas lógicos, Ginga-NCL e Ginga-CC, são discutidos nesta seção considerando o desenvolvimento orientado a componentes do middleware. Nessa discussão, os componentes serão classificados em dois tipos: permanentes ou temporários. Os componentes permanentes são aqueles que possuem funcionalidades utilizadas de forma ininterrupta enquanto o dispositivo receptor estiver em operação. Como consequência, eles devem ser mantidos na memória enquanto o dispositivo receptor estiver nesse modo. Quando atualizações em componentes desse tipo forem necessárias, é desejável que elas sejam realizadas quando o dispositivo receptor estiver em modo stand-by. Já os componentes temporários são aqueles em que suas funcionalidades são necessárias apenas durante determinados instantes do tempo em que o dispositivo receptor estiver em operação. Assim, esse tipo de componente pode ser mantido na memória apenas enquanto estiverem em uso pelo middleware. (Mendes, 2010)

Os demais componentes do Ginga-CC são opcionais e dependem da implementação particular de cada receptor. O Gerenciador de Contexto é o encarregado de colher informações do dispositivo receptor, informações sobre o perfil do usuário e sua localização, e torná-las disponíveis ao Ginga-NCL e Ginga-J, para que eles possam efetuar adaptação de conteúdo ou da forma como conteúdos deverão ser apresentados, conforme determinado pelas aplicações.

Ao Gerenciador de Atualizações cabe o controle das atualizações de todo o software residente e do middleware Ginga, durante o ciclo de vida de um dispositivo receptor. Os

componentes CA (Conditional Access) e DRM (Digital Right Management) são os responsáveis por determinar os privilégios de acesso às diversas mídias que compõem uma aplicação (programa) TVD.

3. Metodologia

A metodologia utilizada foi a programação no script LUA utilizando o VisualCode como IDE. As imagens foram criadas usando o Paint, os fluxogramas foram desenvolvidos na ferramenta Lucidchart visando demonstrar as funcionalidades do jogo desenvolvido.

Tabela 1. Versões das Ferramentas

FERRAMENTAS	VERSÕES
Visual Code	1.41.0
Paint	6.1
Lucidchart	--
Virtual Box	6.0.14
Open Ginga	1.6

O **Visual Code** é um editor de código-fonte desenvolvido pela Microsoft para Windows, Linux e MAC. Ele inclui suporte para depuração, controle Git incorporado, realce de sintaxe, complementação inteligente de código, snippets e refatoração de código. Ele também é customizável, fazendo com que os usuários possam mudar o tema do editor, teclas de atalhos e preferências. Ele é um software livre de código aberto, apesar do download oficial estar sob uma licença proprietária. Ele foi utilizado para o desenvolvimento do código do jogo

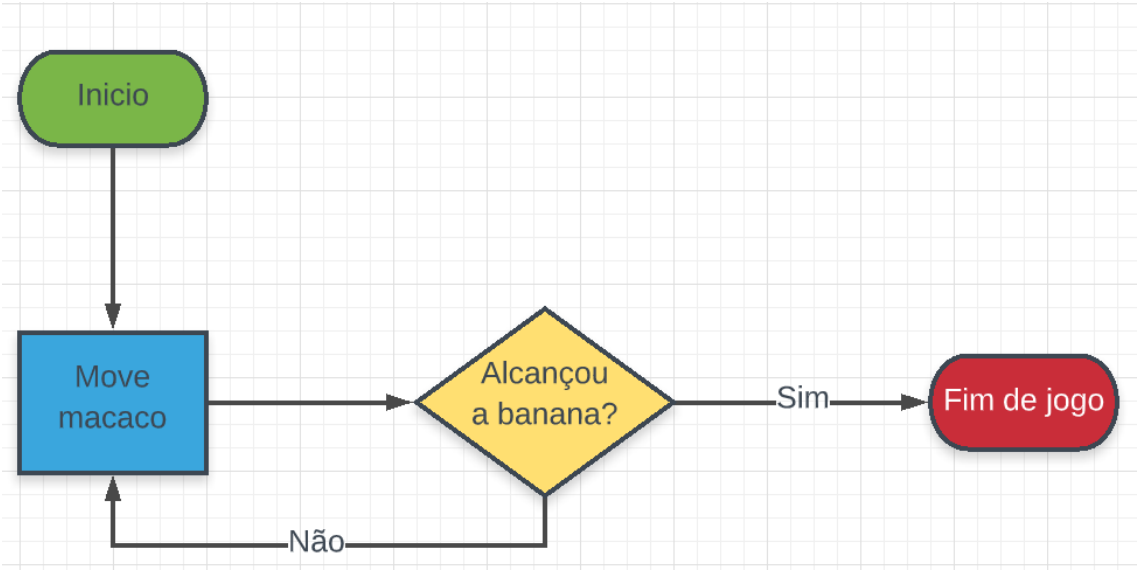
O **Paint** um software utilizado para a criação de desenhos simples e também para a edição de imagens. O programa é incluso, como um acessório, no sistema operacional Windows, da Microsoft. Foi utilizado para editar as imagens dos fluxogramas.

O **Lucidchart** é uma plataforma proprietária baseada na Web, usada para permitir que os usuários colaborem no desenho, revisão e compartilhamento de gráficos e diagramas. Com isso foi utilizado para a criação dos fluxogramas que representam os códigos do jogo e suas funcionalidades.

O **Virtual Box** é um software de virtualização desenvolvido pela empresa Innotek depois comprado pela Sun Microsystems que posteriormente foi comprada pela Oracle que, como o VMware Workstation, visa criar ambientes para instalação de sistemas distintos. Nele foi feita a simulação do jogo em TV Digital e a gravação do jogo simulado.

O **Open Ginga** uma implementação completa (não emulada) de código aberto de um middleware de TV Digital compatível com o padrão SBTVD e executável em ambiente de PC. De forma a oferecer as APIS definidas nas especificações Ginga-J e GingaNCL através da união da Implementação de Referência Ginga-J sob responsabilidade do Laboratório de Aplicações de Vídeo Digital (LAVID) da Universidade Federal da Paraíba (UFPB) e a Implementação de Referência Ginga-NCL. Este foi responsável para instalar e simular uma televisão dentro do Virtual Box.

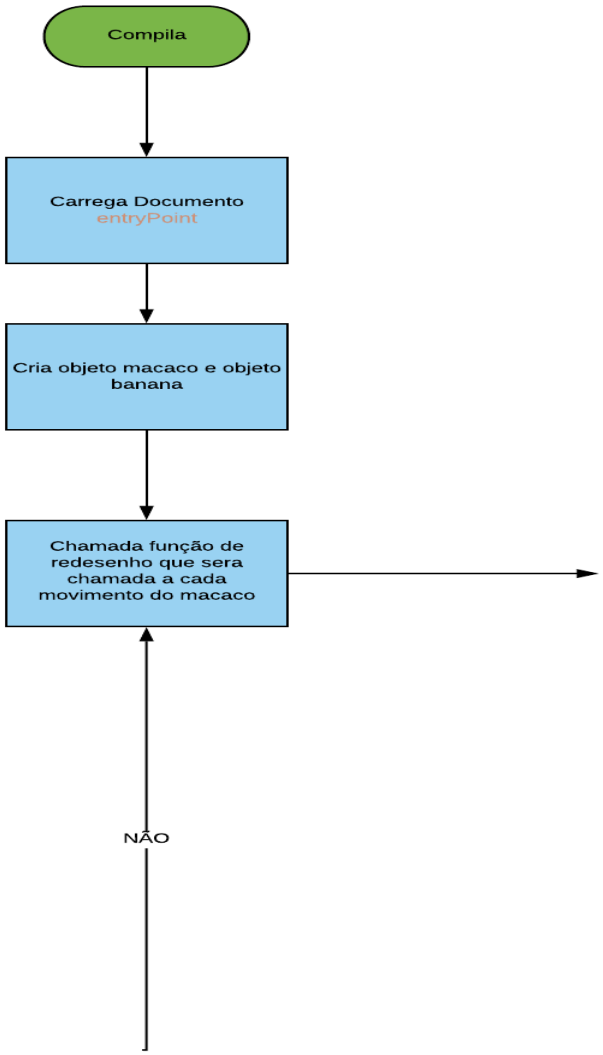
4. Desenvolvimento



Fluxograma 1. Objetivo do Jogo

Fluxograma 1: É um jogo simples onde o usuário deve mover o macaco até a banana. Quando isso ocorre é exibido um botão indicando o fim do jogo. Quando o macaco alcança a banana, o NCLUA sinaliza o início da âncora *fim* e o documento exibe o botão de vitória.

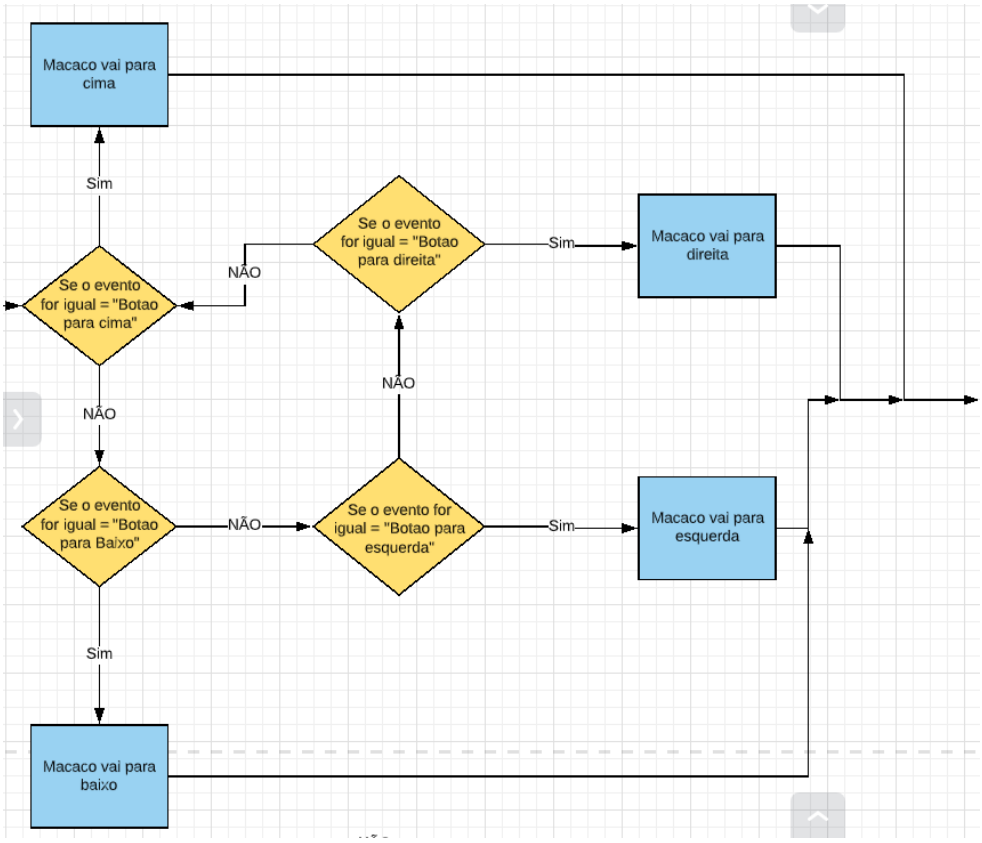
Os fluxogramas a seguir representam as funcionalidades do jogo:



Fluxograma 2. Parte 1 do Código

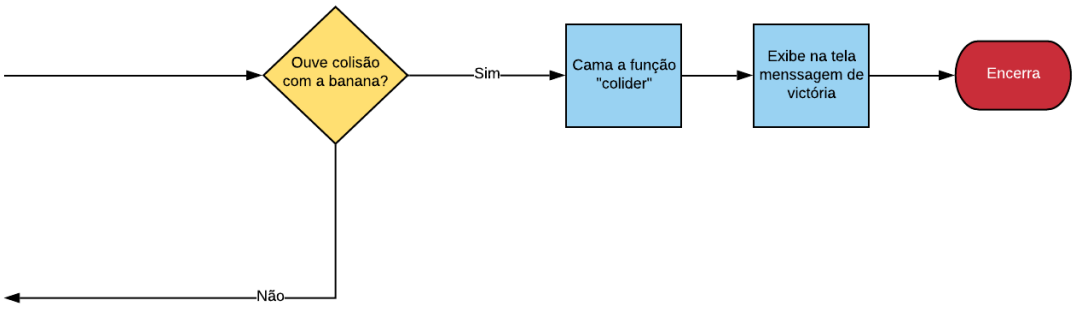
Fluxograma 2: O jogo é iniciado assim que o documento é carregado(“entryPoint”). As primeiras linhas do código NCLUA criam objetos para representar o macaco e a banana. Já na primeira linha de código há uma referência para o pacote [canvas](../referencia/canvas.html), com o qual todas as operações gráficas são efetuadas. A chamada a [canvas](../referencia/canvas.html#function_new) do trecho acima carrega a imagem `monkey.png` que é guardada na variável `img`. A última linha cria uma tabela que guarda as propriedades do macaco: sua imagem, posição (em `x`,`y`) e tamanho (em `dx`,`dy`). Esse uso de tabelas é similar ao de objetos em linguagens orientadas a objetos e é bastante recorrente em LUA.

A variável ‘canvas’ referencia a região NCL destinada ao NCLUA. Isso significa que a variável ‘canvas’terá valor igual a `nul` em um NCLUA cujo documento NCL correspondente não definiu uma região para ele. Em seguida é definida a função de redesenho que será chamada a cada movimento feito pelo macaco, seguindo para o Fluxograma 3.



Fluxograma 3. Parte 2 do Código

Fluxograma 3: Definida a função de tratamento de eventos onde toda a ação do jogo acontece. Estamos interessados em tratar eventos de pressionamento de tecla, sendo assim, a função deve filtrá-los: O tratador de teclas, então, testa o valor da tecla e altera a posição do macaco de acordo. Um evento de tecla carrega a informação `type` que indica se o botão está sendo pressionado ou soltado; e `key` que indica o valor da tecla em questão. Após o movimento do macaco, a função se houve colisão com a banana, seguindo para o Fluxograma 4.



Fluxograma 4. Parte 3 do Código

Fluxograma 4: Nessa função o método [canvas].drawRect, é desenhado dentro de um retângulo que pinta a tela toda de preto (cor configurada anteriormente). As duas linhas seguintes utilizam o método [canvas].compose para desenhar a banana e o macaco sobre o canvas. Por fim o canvas é atualizado com uma chamada à [canvas].flush, a função `colider`, definida.

Essa função recebe dois objetos tais como o macaco e a banana e retorna se ambos estão se superpondo. Utilizando a mesma estrutura em tabela, novos personagens podem ser incluídos no jogo e usados pela mesma função de colisão.

Repare que até aqui apenas foram definidos os objetos (macaco e banana) e funções (`redraw` e `collide`), mais uma vez, o *script* é apenas um inicializador e ações são tomadas somente em resposta a eventos.

Caso haja a colisão, o NCLUA sinaliza ao formatador que sua âncora *fim* teve início e marca um *flag* (`IGNORE=true`) para que eventos futuros sejam ignorados.

5. Resultado Final

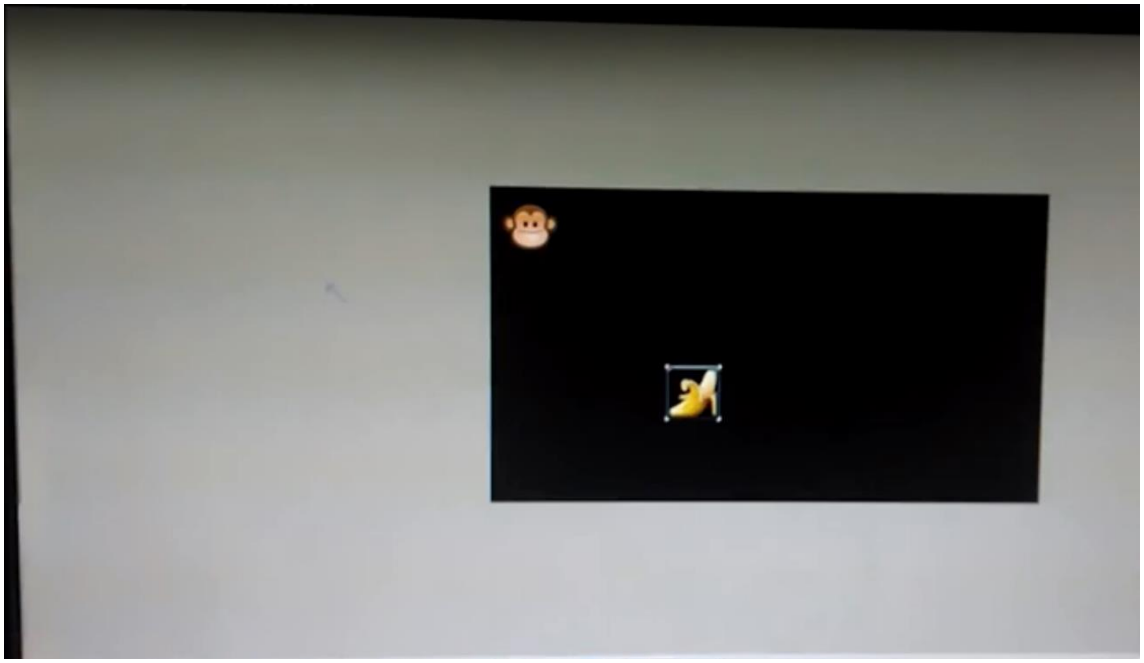


Figura 1. Parte inicial do jogo

Figura 2. Movimentação do macaco

Figura 3. Termino do jogo.

6. Conclusão

O sistema brasileiro encontra-se atualmente entre um dos mais avançados sistemas de TV digital, não apenas por usar as tecnologias mais avançadas, mas principalmente por dispor de tecnologias inovadoras como é o caso de seu middleware Ginga (Soares, 2008).

LUA tem um sólido manual de referência e existem vários livros sobre a linguagem. Dessa forma, a implementação de referência em Ginga NCL escrito em LUA e utilizada como base para o desenvolvimento do jogo desenvolvido encontra-se disponível no site de referência da linguagem LUA (NCL-LUA, 2010). Ferramentas de autoria para o desenvolvimento de aplicações NCL, também em código aberto, estão disponíveis ainda no mesmo site, bem como tutoriais, livros e exemplos de várias aplicações NCL e LUA.

Na Aplicação do Jogo do macaco podemos ver como é aplicado o sistema do NCL-LUA e como é executado o código em uma televisão digital.

O que levou ao desenvolvimento deste artigo foi o intuito de demonstrar o desenvolvimento em televisão digital e as abordagens do Ginga, também consta o desenvolvimento de um jogo e quão simples é a programação em televisão digital e o seu funcionamento

Foi escolhido o LUA pois a linguagem de script é muito simples de se programar e também muito rápida no processamento na hora da compilação. Caso fosse desenvolvido em outra linguagem o período de compilação seria maior e a consumo de memória também seria alta. LUA é usada em muitas aplicações industriais (e.g., Adobe's Photoshop Lightroom), com ênfase em sistemas embutidos (e.g., o middleware Ginga para TV digital) e jogos (e.g., World of Warcraft e Angry Birds). LUA é atualmente a linguagem de script mais usada em jogos (Mendes, 2010).

7. Referências

Cruz, V. M., Moreno, M. F., & Soares, L. F. (s.d.). Ginga-NCL: Implementação de Referência para Dispositivos Portateis.

LUA. (03 de 09 de 2018). *LUA*. Fonte: LUA: http://www.lua.org/about.html

Mendes, R. (29 de Setembro de 2010). *Clube NCL*. Fonte: Clube NCL: http://clube.ncl.org.br/node/108

NCL-LUA. (27 de Setembro de 2010). *CLUBE NCL*. Fonte: CLUBE NCL: http://clube.ncl.org.br/node/101

Soares, L. F. (2008). As múltiplas possibilidades do Middleware Ginga. *PRODUÇÃO PROFISSIONAL*, 76-82.