UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

SASHA NÍCOLAS DA ROCHA PINHEIRO

COMICXML: UMA LINGUAGEM PARA AUTORIA DE HQTRÔNICAS PARA A PLATAFORMA SIFTEO CUBES

São Luís
2013

SASHA NÍCOLAS DA ROCHA PINHEIRO

COMICXML: UMA LINGUAGEM PARA AUTORIA DE HQTRÔNICAS PARA A PLATAFORMA SIFTEO CUBES

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Carlos de Salles Soares Neto
Prof. Dr. em Informática pela PUC-Rio

São Luís
2013

Pinheiro, Sasha Nícolas da Rocha

COMICXML: uma linguagem para autoria HQTrônicas para a plataforma Sifteo Cubes / Sasha Nícolas da Rocha Pinheiro. – São Luís, 2013.

57 f.

Orientador: Carlos de Salles Soares Neto.
Monografia (Graduação) – Curso de Ciências da Computação da Universidade Federal do Maranhão, 2013.

1. Linguagem de programação de computador. 2. HQTrônica. 3. Aplicações interativas. I. Título.

CDU 004.438

SASHA NÍCOLAS DA ROCHA PINHEIRO

COMICXML: UMA LINGUAGEM PARA AUTORIA DE HQTRÔNICAS PARA A PLATAFORMA SIFTEO CUBES

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Aprovada em 10 / 12 / 2013

BANCA EXAMINADORA

Carlos de Salles Soares Neto

Prof. Dr. Carlos de Salles Soares Neto (Orientador)
Universidade Federal do Maranhão

Prof. Dr. Anselmo Cardoso de Paiva
Universidade Federal do Maranhão

Allan Kássio Beckman Soares da Cruz

Prof. Bel. Allan Kássio Beckman Soares da Cruz
Universidade Federal do Maranhão

Hedvan Fernandes Pinto

Prof. Bel. Hedvan Fernandes Pinto
Universidade Federal do Maranhão

À minha mãe.

À Morgana (*in memoriam*).

AGRADECIMENTOS

Agradeço à Deus, Jesus Cristo, que me deu a vida e me salvou, e é a razão do meu viver.

À minha família, em especial à minha mãe, Maria Gardileide da Rocha Pinheiro, que dedicou boa parte de sua vida à seus filhos, e ainda se dedica. Por fazer de mim a pessoa que sou hoje. A ela, portanto, um especial obrigado. Ao meu pai, Agnaldo Rayol Soares Pinheiro, sem o qual eu não teria a educação de hoje. À minha irmã, Krishinna Agnes, e ao meu irmão, Agnaldo Júnior, que sempre estiveram do meu lado. À minha tia, Valéria Cristina, que com suas conquistas sempre foi um exemplo a ser seguido. À meus avós, Abilemar Alves e Maria Concita, que sempre me apoiaram incondicionalmente. À tia Sessé, que me ensinou valores que ninguém mais poderia.

Ao meu amigo, professor e orientador, Carlos de Salles, por sua disposição em me orientar e por ser um grande motivador para mim.

À Márcia Ramos dos Santos, que me ajudou muito durante o desenvolvimento desse trabalho.

À minha namorada Brittney Meays, que mesmo de longe me acompanhava enquanto desenvolvia esse trabalho.

Aos meus amigos da Turma 301, que hoje são mais que amigos, e são para a vida toda. Em especial, à Artur e Leonardo, por termos seguido o mesmo caminho, a Ciência da Computação.

Aos amigos do LAWS, em que dividi muitos momentos de alegria.

A todos os amigos do Curso de Computação, que me aturaram durante esses cinco anos, em especial, André Moreira, Eduardo Reis, Maurício Pessoa e Steve Mpinda.

Aos professores do DEINF, sem os quais não poderia ter tido uma excelente formação.

À CAPES, que me deu várias oportunidades e muito apoio.

Aos amigos do CsF, que foram um refúgio nas horas de solidão.

Aos amigos do The Rock, em especial D'Antae, que me acolheram e me fizeram sentir em casa, na terra do tio Sam.

"Anything's possible if you've got enough nerve."

J. K. Rowling

RESUMO

Linguagem para autoria de HQTrônicas para a plataforma Sifteo Cubes. Sifteo Cubes é uma plataforma de jogos em que pequenos cubos podem perceber movimento e sentir toque em uma tela de LCD de 1,5", assim como perceber quando estão próximos uns dos outros. Por isso, essa plataforma é considerada propício ambiente para execução de aplicações interativas. Porém, o desenvolvimento para essa plataforma não é uma tarefa fácil. Exige-se conhecimentos de programação de computadores, excluindo, dessa forma, autores de histórias em quadrinhos eletrônicas, HQTrônicas, que não são programadores. Assim, esse trabalho objetiva desenvolver um para autoria de HQTrônicas para essa plataforma a fim de facilitar esse processo por parte de autores não programadores.

Palavras-chave: linguagem de autoria, HQTrônica, Sifteo Cubes, aplicações interativas.

ABSTRACT

Comics authoring language for Sifteo Cubes platform. Sifteo Cubes is a game platform in which small cubes can perceive movement and feel touch in a 1.5" LCD screen, as well as notice when they are nearby each other. Because of that, this platform is considered a favorable environment to perform interactive applications. However, developing for this platform is not an easy task. It requires that someone knows computer-programming techniques, and it ends up excluding comic authors, who are not programmers. This paper aims to develop a language for comic authoring for this platform in order to ease this process for authors who are not programmers.

Keywords: authoring language, comics, Sifteo Cubes, interactive applications.

LISTA DE FIGURAS

p.

LISTA DE TABELAS

p.

SUMÁRIO

1. INTRODUÇÃO

Novas tecnologias estão sempre em desenvolvimento. Muitas vezes, o advento de novas tecnologias exige do público alvo, os usuários de tais tecnologias, uma curva de aprendizado a fim de adaptar-se a possíveis novas interfaces de interação, ou seja, exige um certo treinamento ou adaptação por parte dos usuários.

Isso, porém, não significa que novas tecnologias sempre oferecem uma interface de interação completamente inovadora, e nem significa que por serem inovadoras não terão espaço no mercado ou importante papel na vida das pessoas. O mouse, por exemplo, constituiu uma interface de interação com o computador completamente inovadora para sua época, quando os usuários estavam acostumados a interagir com o computador apenas com um teclado físico. O iPhone, por sua vez, smartphone considerado uma das grandes revoluções tecnológicas do século XXI, traz uma interface considerada inovadora, em que cria novas interações entre humano e máquina ao mesmo tempo que combina maneiras já existentes de se comunicar com um computador.

O que se percebe com isso é uma certa necessidade de se criar novas tecnologias que exploram novas maneiras de se interagir com um computador. É comum vermos surgir no mercado dispositivos que parecem ter sido inspirados por filmes de ficção científica. São dispositivos capazes de traduzir fala em tempo real, relógios com sistema operacional e conexão à internet, óculos como assistente pessoal, consoles que entendem a linguagem corporal do jogador, enfim, uma miríade de tecnologias que exploram novas formas que um usuário pode utilizar para se comunicar com o computador.

O desenvolvimento de novas tecnologias se faz presente em várias áreas, desde as artes até a saúde, assim como para diferentes tipos de usuários, com diferentes perfis, diferentes idades, como idosos ou crianças. Imperceptivelmente, brinquedos são revolucionados, renovados, reinventados. Um simples conjunto de Lego ganha uma motor, um jogo de tabuleiro ganha animação, sonoridade. E é no escopo da inovação o principal objetivo do Sifteo Cubes.

O Sifteo Cubes é uma plataforma para “jogos inteligentes, para uma sensação real de mãos no jogo” (Sifteo, Inc., 2013). É sistema de jogo composto por pequenos cubos com tela sensível ao toque que sentem quando outros cubos estão

próximos deles. Essa plataforma oferece, portanto, uma nova interface de interação com o usuário. Nela o usuário joga e interage com os cubos a fim de executar comandos, ou seja, suas ações são a interface com o jogo ou uma aplicação. Inova quando oferece uma interface em que o usuário precisa pegar os cubos e arranjá-los, balançá-los, virá-los, pressioná-los. Por isso, a sensação real de mãos no jogo.

Por ser uma plataforma de hardware que possui processador, memória, dispositivos de entrada e saída, mesmo em reduzidas capacidades, representa assim um computador. Com isso, se faz propício ambiente para execução de aplicações interativas que exploram seus recursos e ofereçam aos usuários nova experiência com o computador.

Um tipo de aplicação que consegue explorar seus recursos interativos e oferecer uma nova experiência para os usuários são as narrativas interativas. Nesse grupo destacamos as histórias em quadrinhos interativas, definidas por (Franco, 2013) como HQTrônicas, um neologismo derivado de histórias em quadrinhos (HQ) eletrônicas.

As HQTrônicas têm sua própria linguagem de comunicação (Garone, Bernardi, & Santos). Essa linguagem é composta por elementos que definem o que uma história em quadrinho precisa ter para ser considerada uma HQTrônica. Elementos como interatividade, trilha sonora, narrativa multilinear, diagramação dinâmica, entre outros, fazem com que uma história em quadrinho seja considerada uma HQTrônica e não somente uma simples digitalização de uma história em quadrinho.

HQTrônicas são criadas por autores de histórias em quadrinhos, que muitas vezes não são programadores. Por isso, a tarefa de desenvolver HQTrônicas para a plataforma Sifteo Cubes se faz de forma exclusiva por parte de programadores. Porém isso não deveria ocorrer. Nesse contexto, esse trabalho objetiva permitir que autores de HQTrônicas possam independentemente de programadores criar suas histórias em quadrinhos para a plataforma Sifteo Cubes. Para isso, propõe-se criar um ambiente que facilite a autoria de HQTrônicas na medida em que abstrai processos específicos e desnecessários para essa tarefa que podem ser automatizados e simplificados. Assim, os autores não precisarão se tornar programadores e poderão criar suas HQTrônicas e executá-las na plataforma Sifteo Cubes.

1.1 Objetivos e Metodologia

O objetivo deste trabalho é a criação de uma linguagem que ofereça uma interface simples e fácil para a autoria de HQTrônicas para a plataforma Sifteo Cubes. Essa linguagem juntamente com outros componentes desenvolvidos são referenciados como ambiente. Esse ambiente simplifica e facilita a autoria pois exige do autor a simples definição de uma HQTrônica em uma linguagem declarativa baseada em XML (W3C). A partir dessa definição da HQTrônica, ou seja, um documento de texto, todo o processo que outrora necessitaria de conhecimentos técnicos com Programação Orientação à Objetos, programação em C++ e Lua, entre outros, é feito automaticamente.

Facilitar a autoria de HQTrônicas por parte dos autores não programadores é possível por causa da implementação dos componentes do ambiente. Os componentes desse ambiente consistem basicamente em um compilador responsável por gerar código C++ e Lua, a partir do documento descritivo da HQTrônica e pelo tocador de HQTrônicas, uma aplicação desenvolvida em C++ para a plataforma Sifteo Cubes.

Um estudo sobre definição formatos de textos para representação das histórias em quadrinho é realizado levando em consideração os elementos das HQTrônicas e os recursos de interação oferecidos pela plataforma Sifteo Cubes, resultando assim na definição de uma linguagem declarativa baseada em XML. O exibidor e o compilador foram então desenvolvidos com base na linguagem de definição de HQTrônicas.

1.2 Organização do Trabalho

No Capítulo 2 é feita uma análise dos trabalhos relacionados. Estes são divididos em três seções que abordam diferentes áreas que são exploradas neste trabalho. Os trabalhos relacionados envolvem desenvolvimento de ambiente de autoria, plataformas que apresentam novas interações humano-computador, e trabalhos que envolvem a definição de documentos e linguagem de representação de aplicações interativas.

Em seguida, no Capítulo 3 é feita uma apresentação da plataforma Sifteo Cubes, onde se mostra exemplos de aplicações e formas de interação. Além disso são explicados detalhes técnicos da plataforma como execução de aplicações e gerenciamento de memória.

O desenvolvimento do ambiente é explicado no Capítulo 4. É feita uma coleta de requisitos para o ambiente, definida a linguagem para representação das HQTrônicas, é mostrado como ocorre o processo de compilação de uma HQTrônica e como foi implementado o exibidor dessas histórias.

Dois estudos de caso são realizados e mostrados no Capítulo 5. E em seguida é feita uma conclusão do trabalho discutindo o desenvolvimento do mesmo, destacando as contribuições trazidas e propondo trabalhos futuros.

2. TRABALHOS RELACIONADOS

Os trabalhos relacionados ao proposto nesta monografia são apresentados em três seções diferentes, cada uma representando diferente viés acerca do trabalho desenvolvido. Na primeira seção são abordados trabalhos relacionados à ambientes de autoria que objetivam facilitar ou oferecer uma interface que se interpõe entre o autor não programador e a plataforma alvo de desenvolvimento. Na segunda seção são apresentados trabalhos que exploram plataformas que oferecem interação humano-computador com uma nova abordagem. E por fim, na terceira seção são apresentados trabalhos relacionados à definição de documentos que representem aplicações interativas.

2.1 Ambientes de autoria

Um ambiente de autoria consiste de um software ou um conjunto de softwares com linguagem específica que é utilizado para criação ou desenvolvimento de algum conteúdo. Muitas vezes, ambientes de autoria surgem na tentativa de oferecer uma interface diferente da existente para o desenvolvimento de algo. Como exemplo disso, tem-se o Dreamweaver, um software de autoria de páginas HTML, que oferece uma interface gráfica para desenvolvimento de conteúdo HTML em detrimento de um simples editor de texto.

Há diversas ferramentas e ambientes com tal proposta. É o caso do Composer e do NCL Eclipse. Ambos ambientes propõem diferentes interfaces para a autoria de aplicações hipermídia interativa para o middleware Ginga, especificação padrão do Sistema Brasileiro de TV Digital Terrestre.

O NCL Eclipse é um plug-in para a IDE Eclipse (Azevedo, Neto, Teixeira, Santos, & Gomes, 2011), que propõe que o desenvolvimento de aplicações interativas não deve limitar os recursos da linguagem e ao mesmo tempo oferecer recursos que facilitem a criação das aplicações. O autor, ao criar uma aplicação dispõe de recursos como visualização programática, pré-visualização de imagens, sons, navegação entre links, entre outros, isso em uma interface que não o limita, porém não muda o perfil do desenvolvedor de tais aplicações, oferecendo somente facilidades para o desenvolvedor profissional.

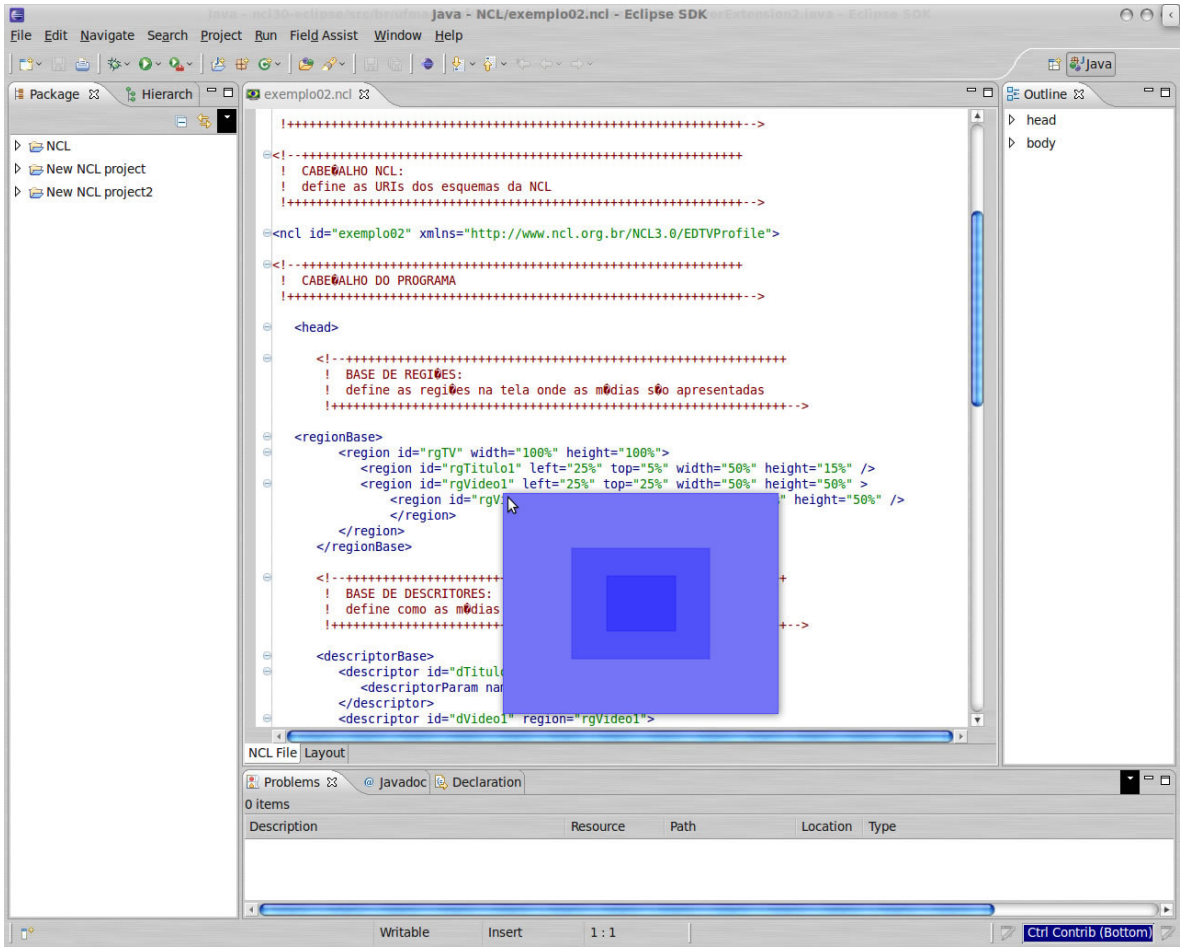

Figura 1 NCL Eclipse Exemplo de Visualização Programática
Fonte: Guimarães, Costa, & Soares, 2007.

O Composer (Guimarães, Costa, & Soares, 2007), por outro lado, oferece o que ele chama de “visões” do documento da aplicação. São outras formas de se ver a aplicação. Antes, a aplicação que era vista somente como um arquivo de texto, passa a poder ser vista como um conjunto de nós, ou como uma linha do tempo, ou como uma apresentação de componentes. Nesse ambiente, a proposta não é esconder o documento de texto representativo da aplicação, mas oferecer outras formas de olhar uma aplicação. Sem dúvida configura um ambiente que facilita mais a autoria de aplicações, na medida que disponibiliza outras formas de se pensar. E com isso, abre oportunidade para outros perfis de desenvolvedores criarem aplicações interativas em NCL.

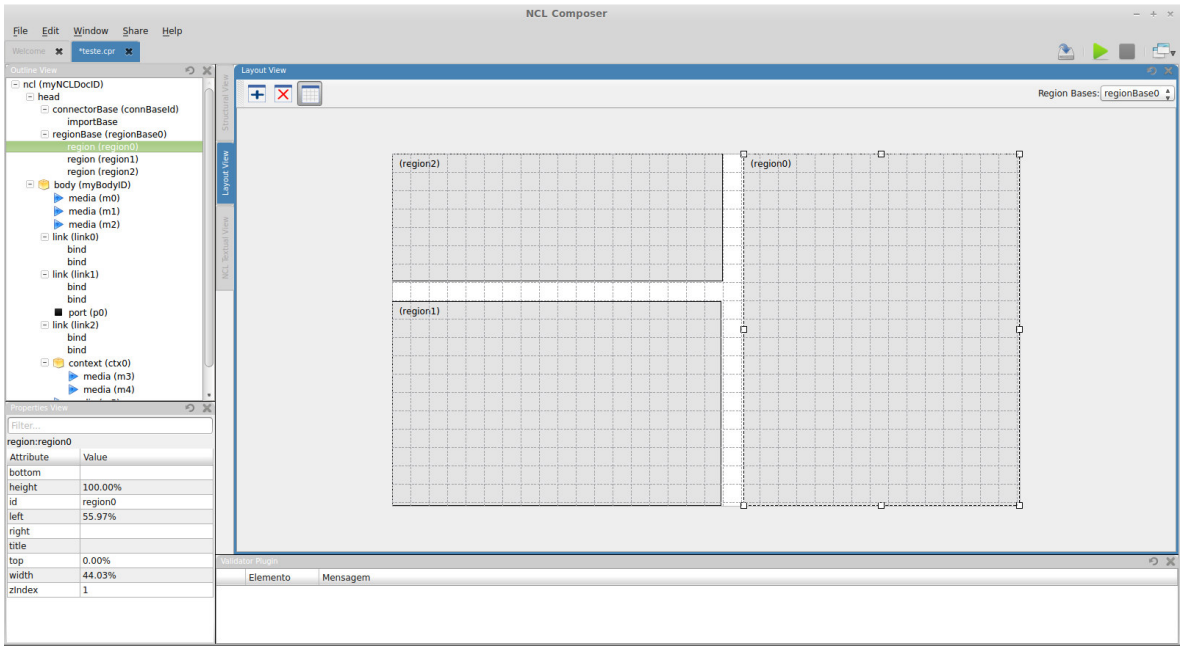

Figura 2 NCL Composer Visão de Layout
Fonte: Guimarães, Costa, & Soares, 2007.

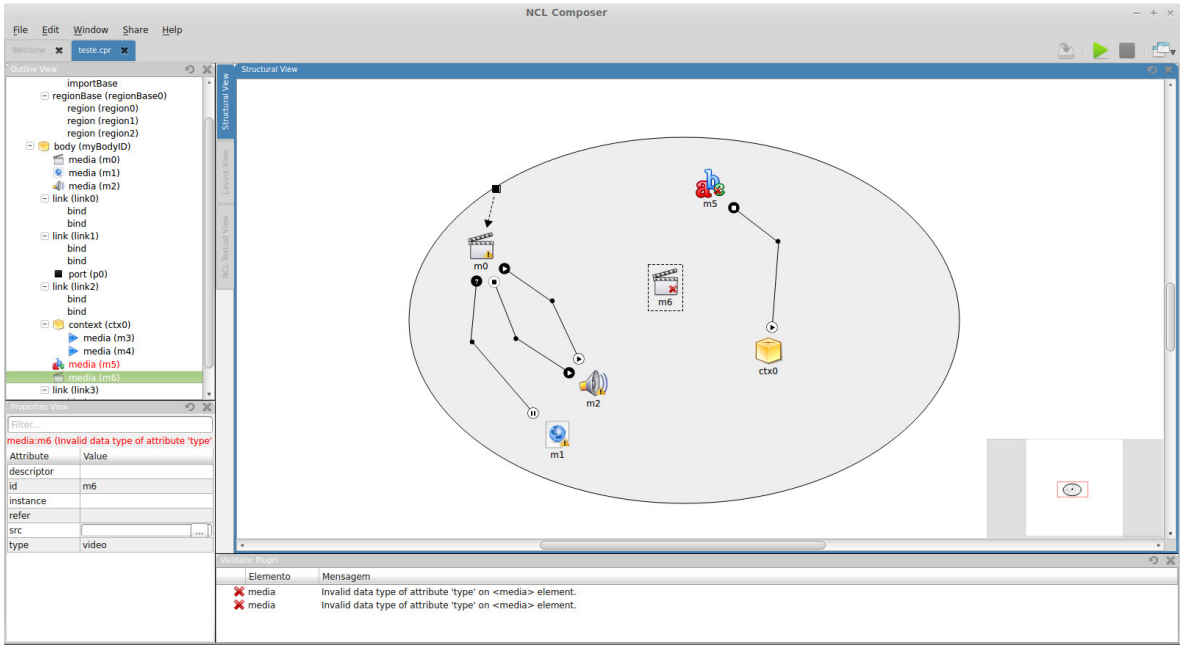

Figura 3 NCL Composer Visão Estrutural
Fonte: Guimarães, Costa, & Soares, 2007.

2.2 Novas abordagens como Interação Humano-computador

Os trabalhos apresentados aqui estão relacionados com a exploração de novas formas de interação do usuário com o computador, não somente na utilização como também na autoria. Em face dos avanços tecnológicos, e de grande difusão de dispositivos com telas sensíveis ao toque e *smartphones*, aqui são apresentados formas de criação e consumo de histórias em quadrinhos diferentes das tradicionais ilustrações em papel.

Em (Lucero, Holopainen, & Jokela, 2012) é apresentado o MobiComics, uma aplicação que permite que um grupo de pessoas criem e edite tirinhas de histórias em quadrinhos usando seus smartphones.

Figura 4 Usuário cria uma tirinha
Fonte: Lucero, Holopainen, & Jokela, 2012.

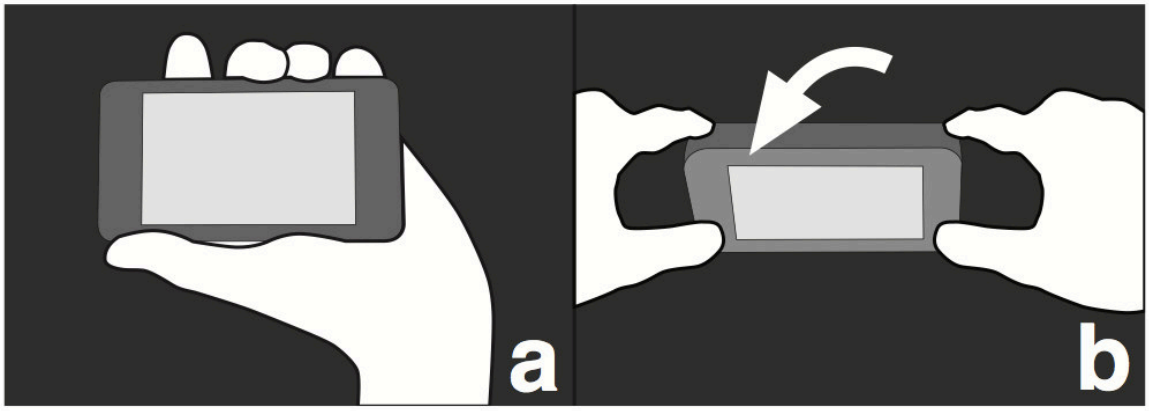

Figura 5 Interações para Compartilhamento de Tirinha
Fonte: Lucero, Holopainen, & Jokela, 2012.

A aplicação permite a criação de tirinhas Figura 4, com uso de balões de fala, e explora interações entre os usuário através de recursos como compartilhamento de conteúdo produzido, com usos de gestos Figura 5.

Em (Andrews, Baber, Efremov, & Komarov, 2012) é apresentado o desenvolvimento de uma superfície multitoque e um software para auxiliar a autoria

e o consumo de histórias em quadrinhos. Aqui há uma tentativa de adicionar interatividade nas tradicionais histórias em quadrinhos, a partir da detecção de toques nos quadrinhos e exibição de outros quadrinhos.

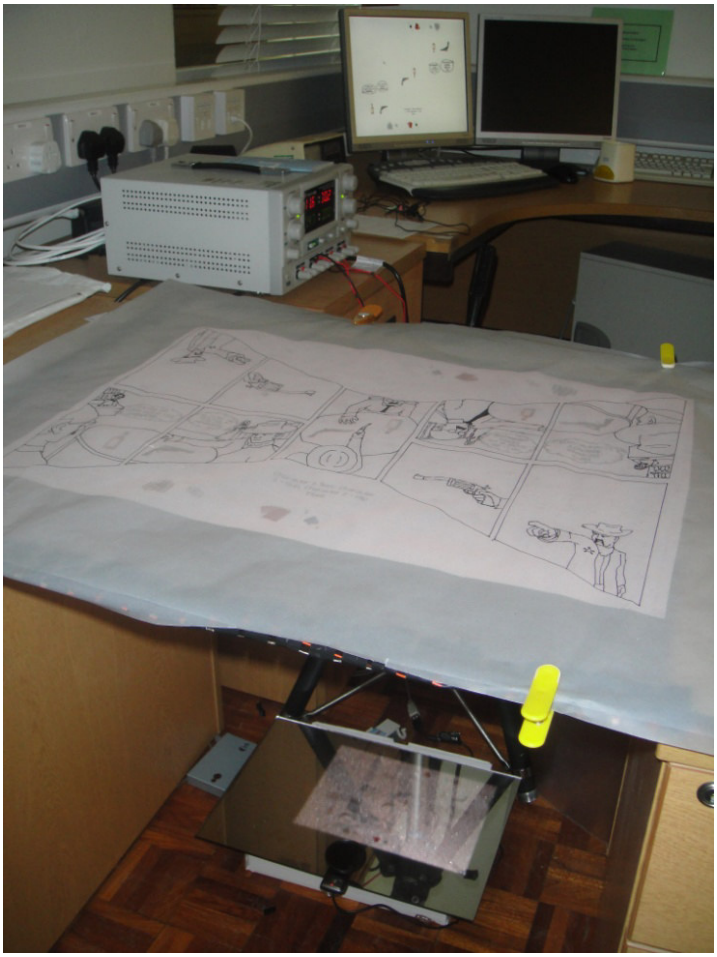

Figura 6 Protótipo da Superfície Multitoque
Fonte: Andrews, Baber, Efremov, & Komarov, 2012.

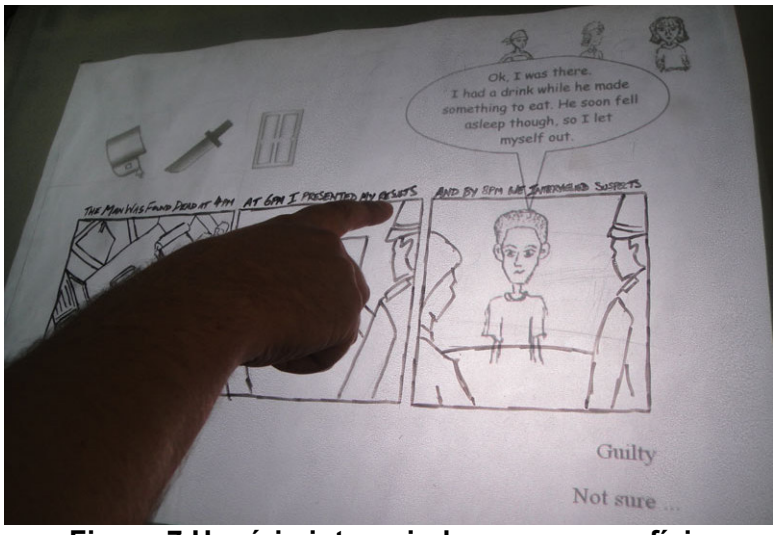

Figura 7 Usuário interagindo com a superfície
Fonte: Andrews, Baber, Efremov, & Komarov, 2012.

2.3 Representação de aplicações interativas

Representar algo através de um documento de texto com um formato predefinido é fundamental para que computadores entendam o que está sendo representado. Exemplo disso é a definição da notação Portable Game Notation um formato de texto que pode ser processado por um computador para o registro e recuperação de disputas de Xadrez (Edwards, 1994).

Transformar uma história em quadrinho em um documento de representação de aplicação interativa se faz necessário quando se pretende executar narrativas interativas em um meio computacional. Isso foi exemplificado com NCL, uma linguagem declarativa para representação de aplicações iterativas para o Ginga.

Em (Gebhard, Kipp, Klesen, & Rist, 2003), é apresentado um processo de autoria, com o uso do kit de ferramentas SceneMaker, em que o usuário pode criar cenas (ou histórias) para execução adaptativa e interativa Figura 8. A criação dessas cenas é feita em dois passos: definição do fluxo da cena Figura 9, e definição do conteúdo da cena.

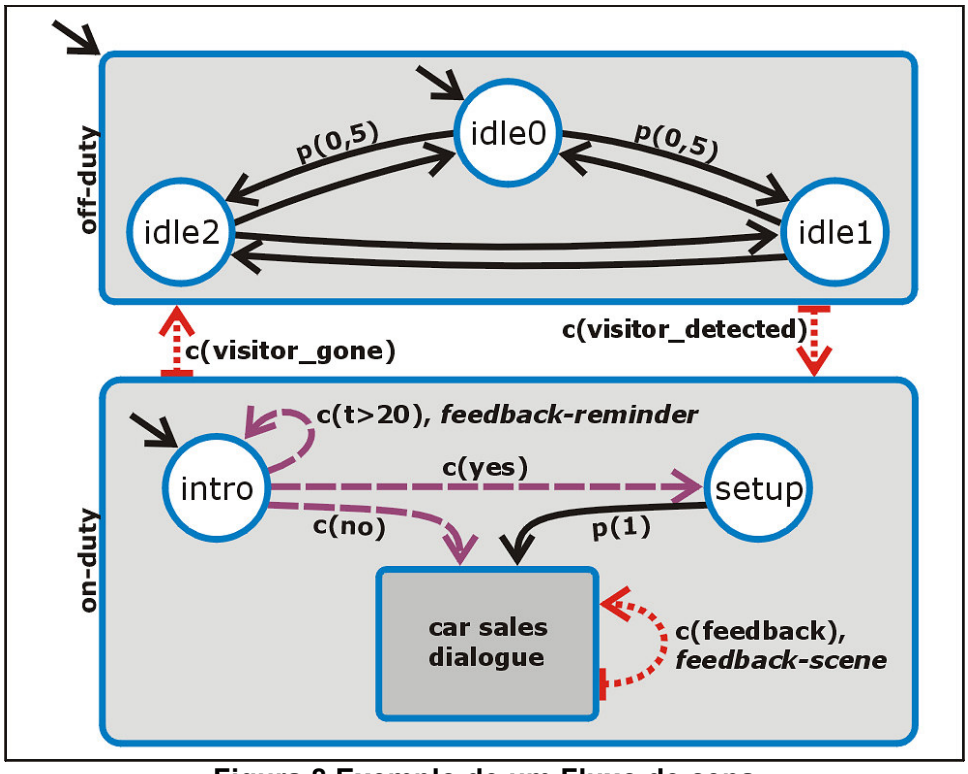

Figura 8 Exemplo de um Fluxo de cena
Fonte: Gebhard, Kipp, Klesen, & Rist, 2003.

```
<scene-flow start="off-duty" name="simpleCrossTalk">
 <super-node name="off-duty" sub-start="node0">
  <i-edge target="on-duty">
   <conditions>
    <event name="visitor_detected" />
   </conditions>
  </i-edge>
  <scene-node name="node0">
   <scene name="idle0" />
    <p-edge target="node1" prob="0.5" />
    <p-edge target="node2" prob="0.5" />
   </scene-node>
   ...
 </super-node>
 <super-node name="on-duty" sub-start="node3">
  <i-edge target="off-duty">
   <conditions>
    <event name="visitor_gone" />
   </conditions>
  </i-edge>
  <scene-node name="node3">
   <scene name="intro" />
   <c-edge target="node4">
    <conditions>
     <event name="yes" />
    </conditions>
   </c-edge>
   <c-edge target="demo">
    <conditions>
     <event name="no" />
    </conditions>
   </c-edge>
   <c-edge target="node3">
    <scene name="feedback-reminder" />
    <conditions>
     <time-greater-than value="20" />
    </conditions>
   </c-edge>
  </scene-node>
  <scene-node name="node4">
   <scene name="setup" />
   <p-edge target="demo" prob="1" />
  </scene-node>
  <super-node name="demo">
   <automatic-scene name="car-sales-dialogue" />
   <i-edge target="demo">
    <scene name="feedback-scene" />
    <conditions>
     <event name="feedback" />
    </conditions>
   </i-edge>
  </super-node>
 </super-node>
</scene-flow>
```

Figura 9 Exemplo de especificação do fluxo de cena
Fonte: Gebhard, Kipp, Klesen, & Rist, 2003.

Os trabalhos relacionados aqui apresentados mostram que há uma necessidade de se criar uma linguagem de representação de HQTrônicas, pois não é de meu conhecimento que exista uma linguagem capaz de permitir a representação de histórias em quadrinhos interativas para o ambiente do Sifteo Cubes.

3. PLATAFORMA DE DESENVOLVIMENTO SIFTEO CUBES

3.1 Apresentação do Sifteo Cubes

Sifteo Cubes é uma plataforma interativa de jogos desenvolvida pela Sifteo, Inc. Atualmente na sua segunda geração, consiste de uma base de controle e de um conjunto de cubos (pequenas caixas ou blocos) que podem perceber movimentos, aproximação e toque em uma tela sensível colorida de 1,5 polegadas.

Com isso, essa plataforma oferece interação entre os cubos e também entre os usuários e os cubos, quando são balançados (*shake*), inclinados (*tilt*), virados (*flip*), tocados (*press*) ou colocados em proximidade de outro cubo (*neighbor*), como mostrado na Figura 10.

Figura 10 Interação com os cubos
Fonte: Sifteo, Inc., 2013.

Esses recursos fazem com que o Sifteo Cubes seja uma ambiente propício para aplicações interativas. Comumente, nas aplicações disponíveis para essa plataforma, o usuário é convidado a aproximar cubos, incliná-los ou a tocá-los a fim de executar alguma ação. É o que ocorre na aplicação Chroma Splash, Figura 11, em que o usuário precisa aproximar cubos de forma que ícones da borda e da mesma cor estejam em contato, a fim de eliminá-los e marcar pontos. E posteriormente inclinar os cubos para que os ícones restantes no interior sejam deslocados para a borda.

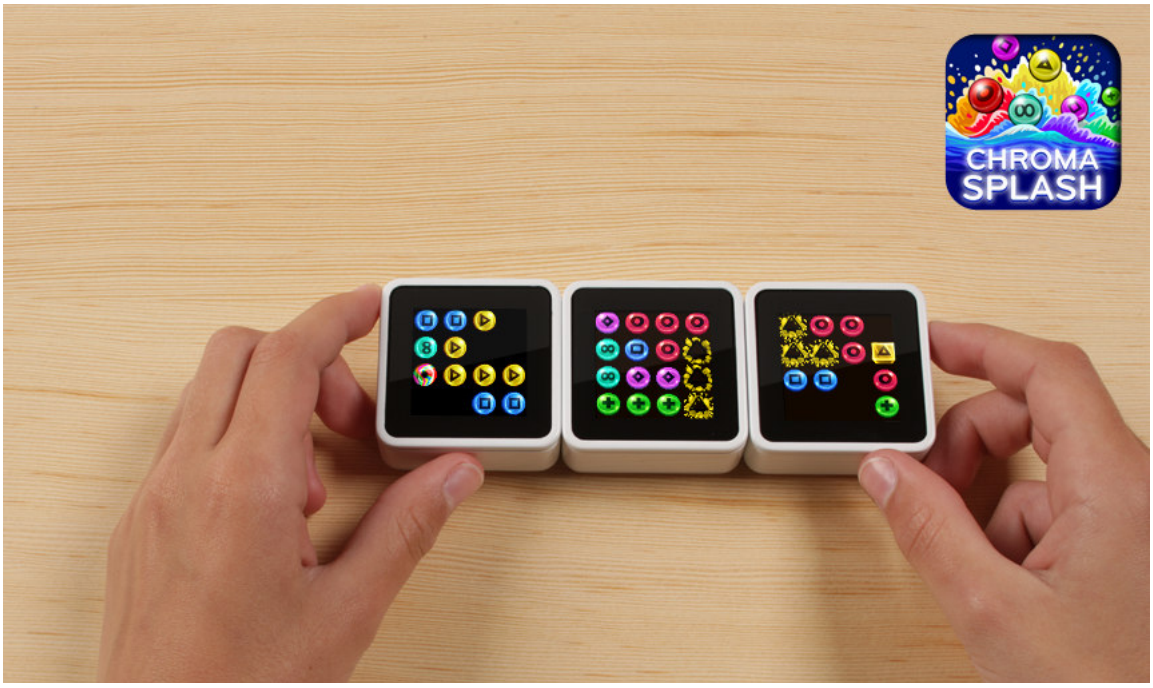

Figura 11 Aplicação Chroma Splash
Fonte: Sifteo, Inc., 2013.

No jogo Code Cracker, Figura 12, esses mesmos recursos são explorados. O usuário precisa aproximar os cubos e tocar nas telas dos cubos para selecionar códigos diferentes.

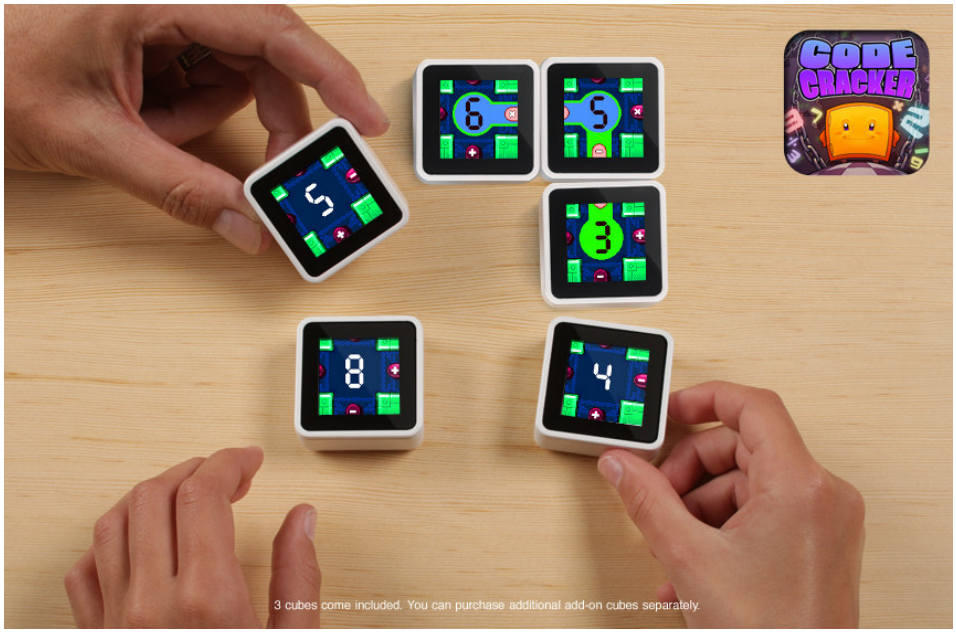

Figura 12 Aplicação Code Cracker
Fonte: Sifteo, Inc., 2013.

No jogo *Word Caravan*, o jogador precisa modificar as posições das letras em cada cubo inclinando-os, e então achar uma ordem que forme palavras. Para isso, ele aproxima os cubos uns dos outros na ordem que achar necessário.

Figura 13 Jogo Word Caravan
Fonte: Sifteo, Inc., 2013.

Um recurso interessante presente no jogo *Sandwich Kingdom*, Figura 14, é a continuação do cenário de uma fase do jogo a partir da aproximação de outros cubos, que revelam novas partes do jogo. Ou seja, o usuário precisa aproximar os cubos em todas as direções e descobrir para onde é possível mover o personagem no jogo.

Figura 14 Jogo Sandwich Kingdom
Fonte: Sifteo, Inc., 2013.

3.2 Desenvolvimento de aplicações Sifteo Cubes

Um desenvolvedor que pretenda criar uma aplicação para o Sifteo Cubes precisa programar em C++ e Lua, conhecer a API disponível para manipular os cubos e se familiarizar com o SDK da plataforma. Além disso, é preciso conhecer técnicas de desenvolvimento de jogos e aprender como o Sifteo Cubes gerencia memória e executa aplicações. A API é disponibilizada no site da empresa juntamente com um kit de programas (SDK) que são necessários para o desenvolvimento das aplicações, como por exemplo o “stir”, que converte imagens em código C++, ou o “swiss”, que faz a comunicação entre o computador e a base de controle via cabo de transferência de dados.

Cada cubo possui a seguinte configuração de hardware:

- CPU ARM de 32 bits;
- 128 x 128 color TFT LCD;
- Acelerômetro de 3 eixos;
- Memória Flash 8MB;
- Comunicação via rádio de 2.4 GHz.

Figura 15 Conexão entre base e cubos
Fonte: Sifteo, Inc., 2013.

Uma aplicação Sifteo Cubes é organizada em um arquivo binário ELF, um formato de arquivo padronizado, e é executada pela base de controle em uma

sandbox [1] chamada de SVM. O código C++ da aplicação é compilado pelo compilador Clang[2] e através da ferramenta “slinky”, disponível pelo Sifteo SDK, é gerado o código e é feita optimização de código.

A base de controle possui recursos bastante limitados. A Figura 16 mostra como a memória ROM (*Main Flash*) e a memoria RAM (*Physical RAM*) são mapeadas para a memória virtual da SVM.

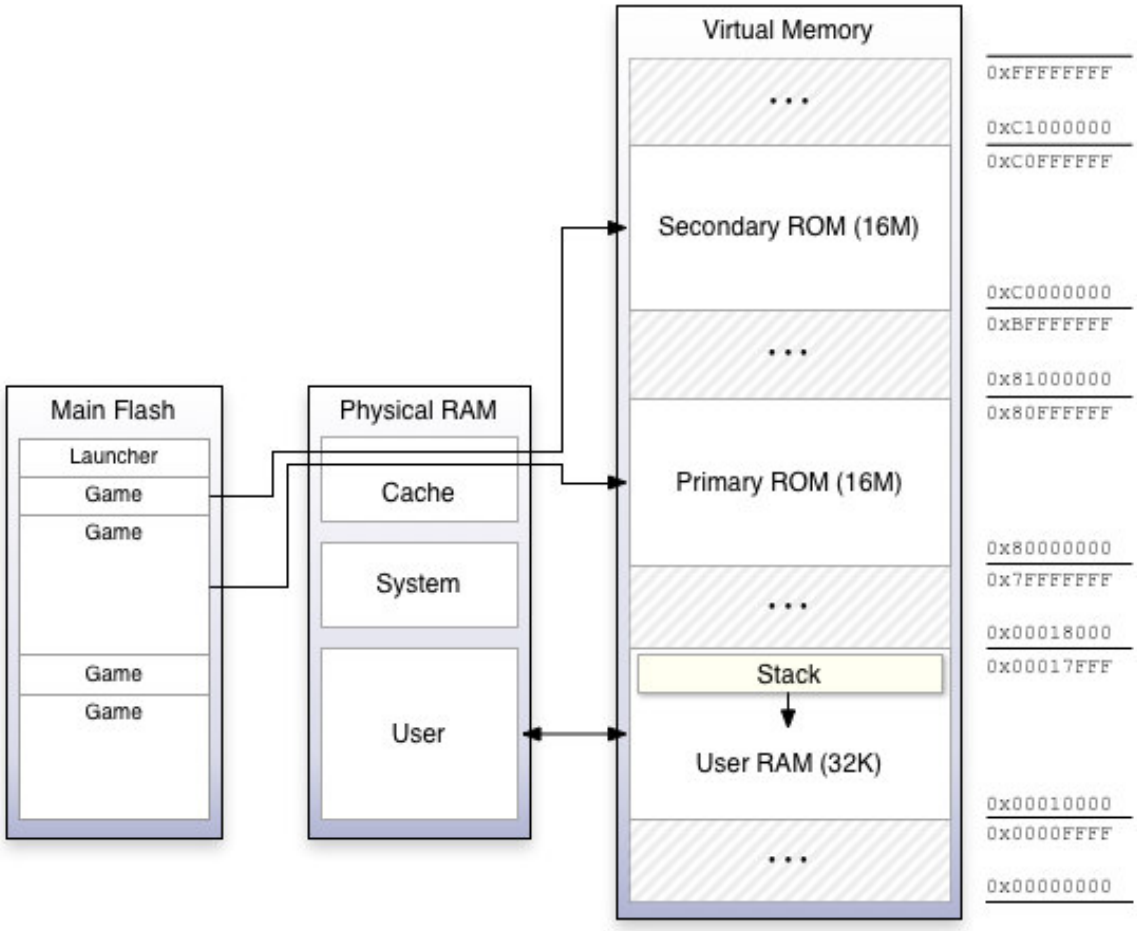

Figura 16 Mapa de memória da SVM
Fonte: Sifteo, Inc., 2013.

A memória de leitura, ROM, disponível na base de controle tem capacidade de 16 MB. Ou seja, se houvesse uma aplicação cujo binário ELF fosse de 16 MB, ela seria a única na base. Por outro lado, a memória RAM disponível para uma aplicação é de 32 KB, e a pilha de execução compartilha essa memória, Figura 16.

[1] Um ambiente de execução controlado usado como mecanismo de segurança para que danos que eventualmente ocorram não afete o sistema por completo.
[2] http://clang.llvm.org/

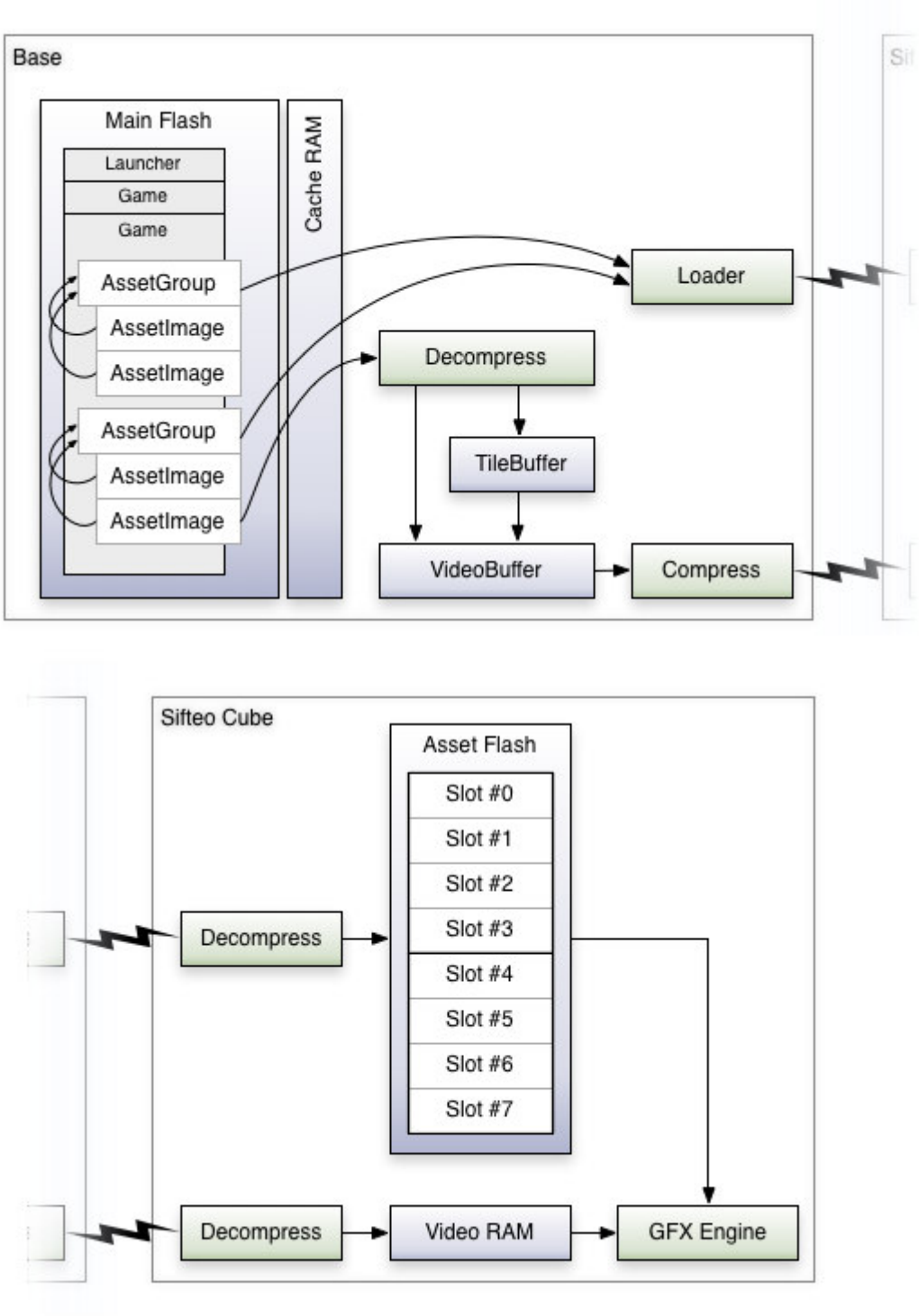

Figura 17 Gerenciamento da memória para Assets
Fonte: Sifteo, Inc., 2013.

Em uma aplicação Sifteo, as imagens, dados e sons são chamados de *Assets*. Cada tipo de arquivo tem seus respectivos métodos de compressão. Como a comunicação entre a base de controle e os cubos é limitada, algumas técnicas para melhorar performance e a experiência do usuário é implementada na plataforma Sifteo Cubes.

No caso de imagens, que serão exibidas no display LCD dos cubos, *tiles*[3] são gerados e agrupados, e então enviados para a memória *Asset Flash* de cada cubo, Figura 17. Ela contem 4 MB, mas somente metade pode ser utilizada por uma aplicação.

É na base de controle que as imagens ficam armazenada e, quando solicitadas para serem exibidas no display de algum cubo, são enviadas através da comunicação via rádio.

Na Figura 18, é possível perceber que o processo de renderização é feito de forma distribuída, onde a base controla o que deve ser mostrado em cada cubo. Como o canal de conexão entre a base e os cubos é limitado, os cubos não armazenam as imagens, somente os tiles.

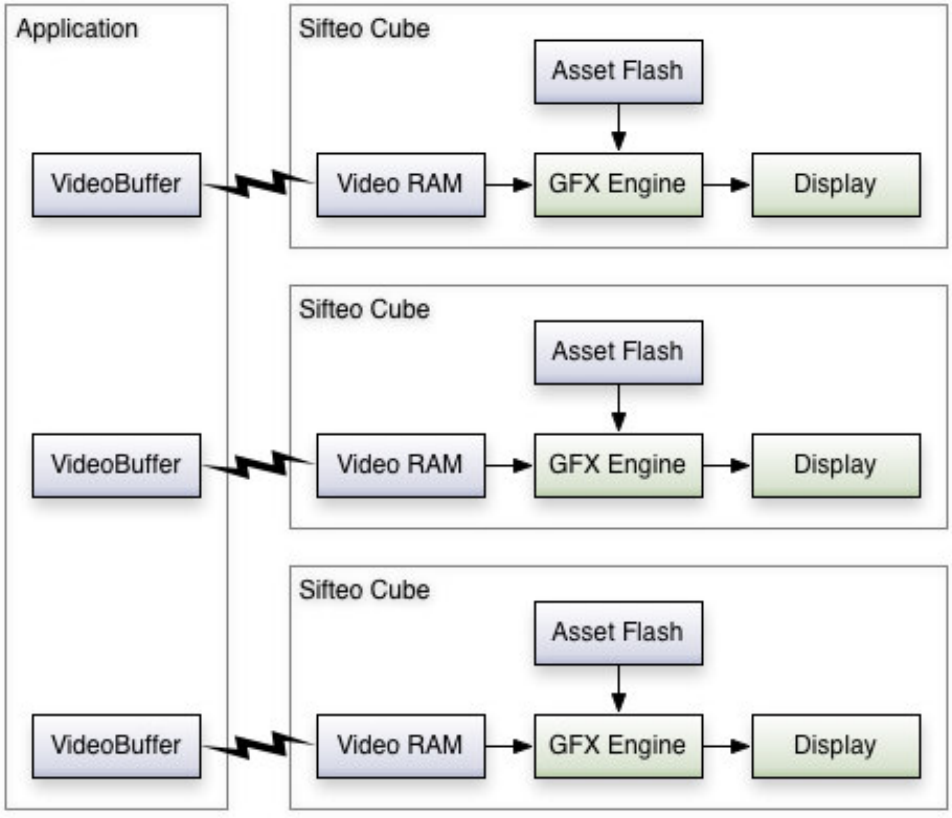

Figura 18 Processo de Renderização Distribuído
Fonte: Sifteo, Inc., 2013.

[3] Conjunto de pixels em um formato de matriz 8x8

4. AUTORIA DE HQTRÔNICAS COM COMICXML

O ambiente para autoria proposto auxilia o processo de autoria de HQTrônicas (histórias em quadrinho digitais) para a plataforma Sifteo Cubes. Tem como objetivo fazer do processo de autoria dessas histórias para essa plataforma um processo mais fácil que o convencional, descrito no Capítulo 3, na medida em que exime os autores da necessidade de saber programar em C++, em Lua, e de conhecer a API do Sifteo Cubes, a fim de criar novas HQTrônicas.

Seu desenvolvimento compreende a criação dos seguintes componentes:

- definição de uma linguagem em XML para representação das histórias em quadrinhos, chamada de *ComicXML*;
- uma aplicação que execute histórias em quadrinhos no Sifteo Cubes, chamada de *Comic Player*;
- uma ferramenta que compile ComicXML para C++ e Lua, e compile automaticamente o exibidor para a plataforma Sifteo Cubes, usando as ferramentas de compilação da plataforma, definida como *Sifteo Comics Tool*.

A implementação dos componentes supracitados é iniciada com a definição dos requisitos necessários para que se tenha histórias em quadrinhos em essência sendo executadas pelo Sifteo Cubes. Portanto, o primeiro passo foi a definição desses requisitos.

4.1 Requisitos

Para se definir os requisitos funcionais do ambiente de autoria proposto, tomou-se como base os principais elementos da linguagem das HQTrônicas. Esses elementos, como explica (Franco, 2013), são: animação, diagramação dinâmica, trilha sonora, efeitos de som, tela infinita, tridimensionalidade, narrativa multilinear e interatividade.

Os elementos de diagramação dinâmica, tela infinita, narrativa multilinear e interatividade são disponibilizados para os autores das histórias em quadrinhos a partir do uso dos próprios recursos de interação com o cubo.

Os elementos trilha sonora e efeitos de som também serão disponibilizados, pois a base de controle possui caixa de som e, com os recursos de interação dos cubos, o autor decide quando os efeitos são tocados.

O elemento de tridimensionalidade depende do autor, pois como os cubos exibem imagens em 2D, a tridimensionalidade é alcançada com o uso de perspectiva nessas imagens, ou seja, de como o autor as desenha.

A partir dessa análise, a implementação do ambiente de autoria é direcionada a fim de implementar esses elementos essenciais das HQTrônicas. Portanto, os requisitos funcionais do ambiente são:

1. Permitir a criação de histórias em quadrinhos interativas, em que o usuário possa utilizar os recursos de interação dos cubos, como movimento, toque e aproximação, para conduzir a narrativa;
2. Permitir o uso de trilha sonora. O autor das histórias em quadrinhos pode definir um arquivo de áudio para tocar durante a execução da HQTrônica;
3. Permitir o uso de efeitos de som. Esses efeitos de som devem ser adicionados pelo autor de acordo com a narrativa escolhida para a sua história;
4. Permitir o uso de tela infinita, ou seja, um cubo pode estar mostrando parte de uma cena, e quando o usuário aproximar outros cubos, novas partes da cena podem ser mostradas. Isso é possível com a detecção de aproximação dos cubos.

Permitir autoria de uma narrativa multilinear. O autor pode desejar oferecer aos usuários uma narrativa que exige pontos de decisão e o usuário deve assim escolher uma opção para seguir.

Esses requisitos funcionais garantem que as histórias em quadrinhos serão consideradas HQTrônicas, pois oferecem ao autor a possibilidade de utilizar recursos elementares para a autoria de histórias em quadrinhos digitais. Por isso, há uma necessidade de representar tais recursos oferecidos. E isso é feito com a definição de uma linguagem declarativa, que formaliza a escrita e representação de HQTrônicas através de documentos de texto. Essa linguagem foi nomeada de *ComicXML*.

4.2 ComicXML: uma linguagem para definição de HQTrônicas

Extensible Markup Language (XML) é uma linguagem padronizada de marcação genérica que pode ser utilizada para gerar linguagens de marcação. Ou seja, com XML é possível criar outras linguagens.

Por existir diversas ferramentas de validação e interpretações de documentos XML e por ela ser declarativa, padrão de programação relativamente mais fácil de leitura e entendimento por humanos, ela foi escolhida para a definição da linguagem para representação de HQTrônicas.

A definição das regras de validação da linguagem ComicXML é dada pelo XML Schema no Apêndice A – XML Schema da ComicXML. A estrutura de um documento ComicXML pode ser generalizada como a seguinte:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<sifteoHQ xmlns:ns2="xml.jaxb.ComicXML.SifteoHQ">
     <begin>
         <action type="start" target="num1" cube="0"/>
     </begin>

     <mediaBase>
         <media id="num1" type="img" src="assets/comeco1.png"/>
     </mediaBase>

     <componentBase>
         <component id="num1" canbemain="true">
           <media target="num1"/>
           <linkBase>
                 <link>
                       <conditionBase>
                             <condition type="touch"/>
                       </conditionBase>
                       <actionBase>
                             <action type="start" target="niceguy"
cube="this"/>
                       </actionBase>
                 </link>
           </linkBase>
      </component>
     </componentBase>
</sifteoHQ>
```

Figura 19 Exemplo de um documento ComicXML

O elemento raiz de um documento ComicXML é o <sifteoHQ>. Este contém três elementos filhos que servem como lista para outros elementos. A seguir é apresentado a definição dos elementos filhos e suas listas.

Tabela 1 Descrição do elemento SifteoHQ

Elemento	Elementos filhos	Descrição
sifteoHQ	begin	Lista de ações a serem executadas no início da HQTrônica
	mediaBase	Lista de mídias utilizadas
	componentBase	Lista de componentes

Tabela 2 Descrição das listas do elemento sifteoHQ

Lista	Itens da Lista	Atributos dos Itens
begin	action	type, target, cube, wait
mediaBase	media	id, type, src
componentBase	component	id, canBeMain

Cada componente tem dois elementos, uma referência para uma mídia e uma lista de links. A seguir é apresentado as definições dos elementos filhos e das listas assim como os atributos necessários para cada elemento.

Tabela 3 Descrição do elemento component

Elemento	Elementos filhos	Descrição
component	media	Elemento único que referencia mídias do elemento mediaBase, através do atributo target.
	linkBase	Lista de links

Tabela 4 Descrição do elemento link

Elemento	Elementos filhos	Descrição
link	conditionBase	Lista de condições
	actionBase	Lista de ações

Tabela 5 Descrição das listas do elemento link

Lista	Itens da Lista	Atributos dos Itens
conditionBase	condition	type
actionBase	action	type, target, cube

Para a definição da ComicXML, as histórias em quadrinhos foram modeladas como um conjunto de componentes que possuem registros de eventos (*links*). Cada link possui um conjunto de condições que devem ser satisfeitas para que ações possam ser executadas. Ou seja, na escrita de uma HQTrônica, o autor deve identificar imagens e associar eventos a essas imagens.

Por exemplo, se em uma dada narrativa, o autor exibe a imagem X e deseja que, caso o usuário aproxime um cubo à direita do cubo que exibe a imagem X, uma imagem Y seja exibida no cubo aproximado, então ele deve imaginar isto como um componente que possui a imagem X e um link que tenha condição de aproximação de vizinho à direita e ação de exibição de outra imagem. Isso é esclarecido no diagrama a seguir.

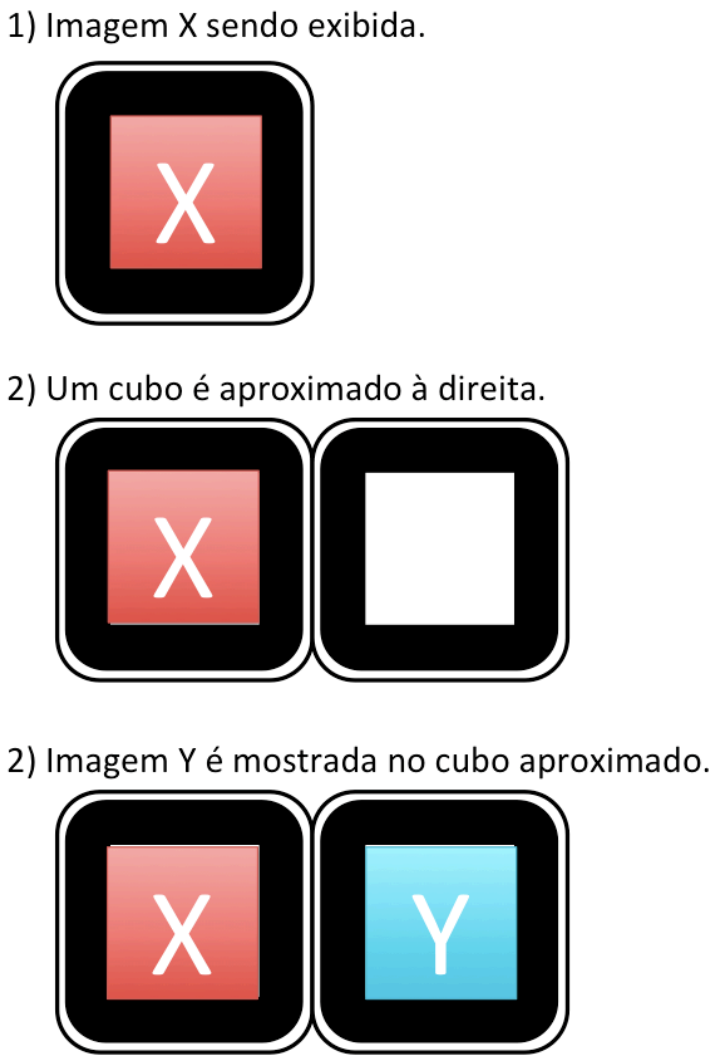

Figura 20 Exemplo de uma possível interação

Com esse exemplo, o autor escreveria um componente no documento de texto ComicXML da seguinte maneira:

```
<component id="comp1" canbemain="true">
      <media target="imagemX"/>
      <linkBase>
            <link>
                  <conditionBase>
                              <condition type="rightNeighbor"/>
                  </conditionBase>
                  <actionBase>
                        <action type="start" target="comp2"
cube="neighbor"/>
                  </actionBase>
            </link>
      </linkBase>
</component>

<component id="comp2">
      <media target="imagemY"/>
</component>
```

Figura 21 Componentes com as imagens X e Y em ComicXML

No código acima, é possível perceber que o componente tem uma referência para uma imagem (*media*) e um link. O *link*, por sua vez, contém uma condição simples e uma ação. Esse componente é suficiente para representar o que o autor deseja na Figura 20.

Links podem ter condições simples ou compostas. As condições foram definidas de acordo com os requisitos funcionais, para que possa ser possível o uso dos recursos de interação dos cubos e permitir o desenvolvimento de HQTrônicas com seus elementos de linguagem. As condições disponíveis são:

Tabela 6 Tipos de condições

Tipos de Condição	
Touch	Shake
Tilt	TopNeighbor
LeftNeighbor	BottomNeighbor
RightNeighbor	

As ações foram definidas para manipular os cubos e os componentes e auxiliar o gerenciamento das narrativas no que se refere ao fluxo de leitura do usuário.

Tabela 7 Tipos de ações com respectivos atributos necessários

Tipo de Ação	Atributos necessários
Start	target, cube
Display	target, cube
SetMainComponent	target

4.3 Arquitetura do Processo de Compilação de uma HQTrônica

Para criar uma HQTrônica para a plataforma Sifteo Cubes, o autor deve escrever um documento que represente a história em quadrinho na linguagem de definição de HQTrônicas proposta por este trabalho, ComicXML, gerando assim um arquivo de texto ".cxml", como mostrado na Figura 22, primeira etapa.

O documento em ComicXML é então compilado pelo *Sifteo Comics Tool*, um kit de ferramentas que auxilia o processo de autoria. Nessa segunda etapa, ilustrada pela Figura 22, é gerado códigos em C++ e em Lua a partir do documento ".cxml".

Os arquivos gerados na segunda etapa são então adicionados ao código do tocador de HQTrônicas, o *Comic Player*, uma aplicação para a plataforma Sifteo Cubes previamente desenvolvida que exibe histórias em quadrinhos interativas.

A quarta etapa, Figura 22, compreende a compilação do Comic Player juntamente com a história em quadrinho, através do uso das próprias ferramentas de compilação disponibilizadas no SDK do Sifteo Cubes pela Sifteo, Inc. Esse processo é automatizado pelo *Sifteo Comics Tool*, e deve ocorrer toda vez que o autor modificar a narrativa de sua história.

Por fim, na quinta etapa, o exibidor compilado juntamente com a HQTrônica, compressos em um arquivo ".elf", é enviado para a base de controle, onde poderá ser executado.

Como o objetivo principal do trabalho proposto é facilitar a autoria de histórias em quadrinhos interativos para a plataforma Sifteo Cubes, isentando o autor de conhecer e programar em C++ e Lua, assim como manipular ambientes de linha de comando, todo o processo de compilação ocorre automaticamente nas

etapas 2 até 5, necessitando apenas do autor um comando para execução do processo por completo. Ou seja, o autor edita o documento com a HQTrônica e opta por enviar a história para a base de controle, abstraindo todo o processo de compilação de baixo nível.

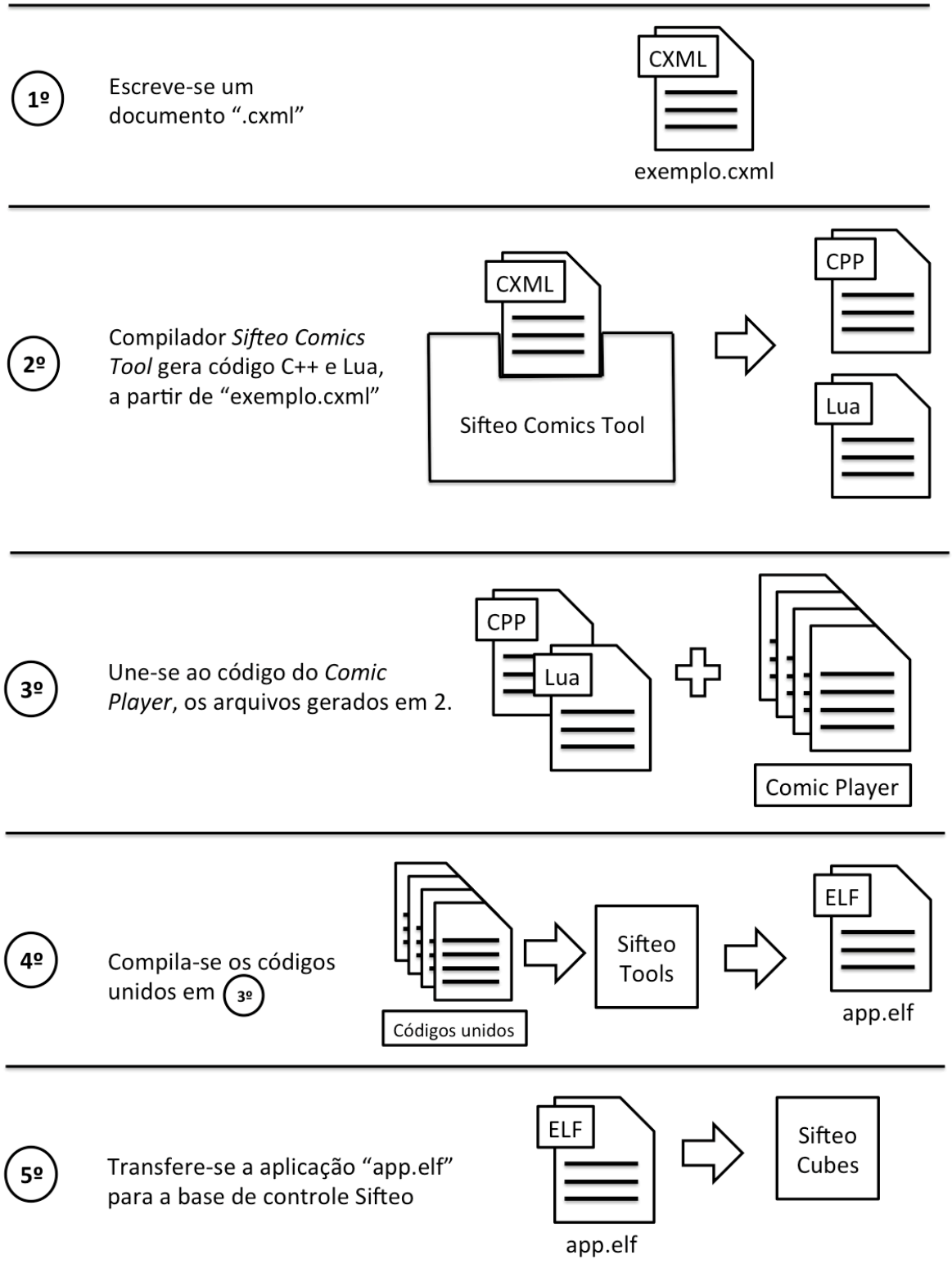

Figura 22 Processo de compilação de uma HQTrônica

4.4 Implementação do Comic Controler

A base de controle do Sifteo Cubes executa códigos em C++ compilados para a arquitetura da plataforma e compressos em um arquivo “.elf”. O Comic Player é, portanto, uma aplicação escrita utilizando a API do Sifteo Cubes que é compilada e enviada para o Sifteo Cubes responsável por exibir a HQTrônica.

Como não se pode enviar arquivos de XML ou em outro formato, o Comic Player precisa ter estruturas que tenha correspondências com as oferecidas pela ComicXML. Isso é necessário para que se garanta a criação de HQTrônicas com base nos requisitos definidos em 4.1 Requisitos.

A implementação do exibidor foi feita com base na programação orientada a objetos. O diagrama de classes do Comic Player é mostrado na Figura 23.

O exibidor gerencia os componentes, os cubos e as mídias. Os eventos de hardware gerados pelas interações entre os cubos, como aproximação e toque, também são gerenciados pela classe Player. Por exemplo, cada vez que um cubo é pressionado, o exibidor se encarrega de verificar em todos os componentes ativos, quais possuem links cuja condição é satisfeita. Assim ocorre com cada evento que ocorre.

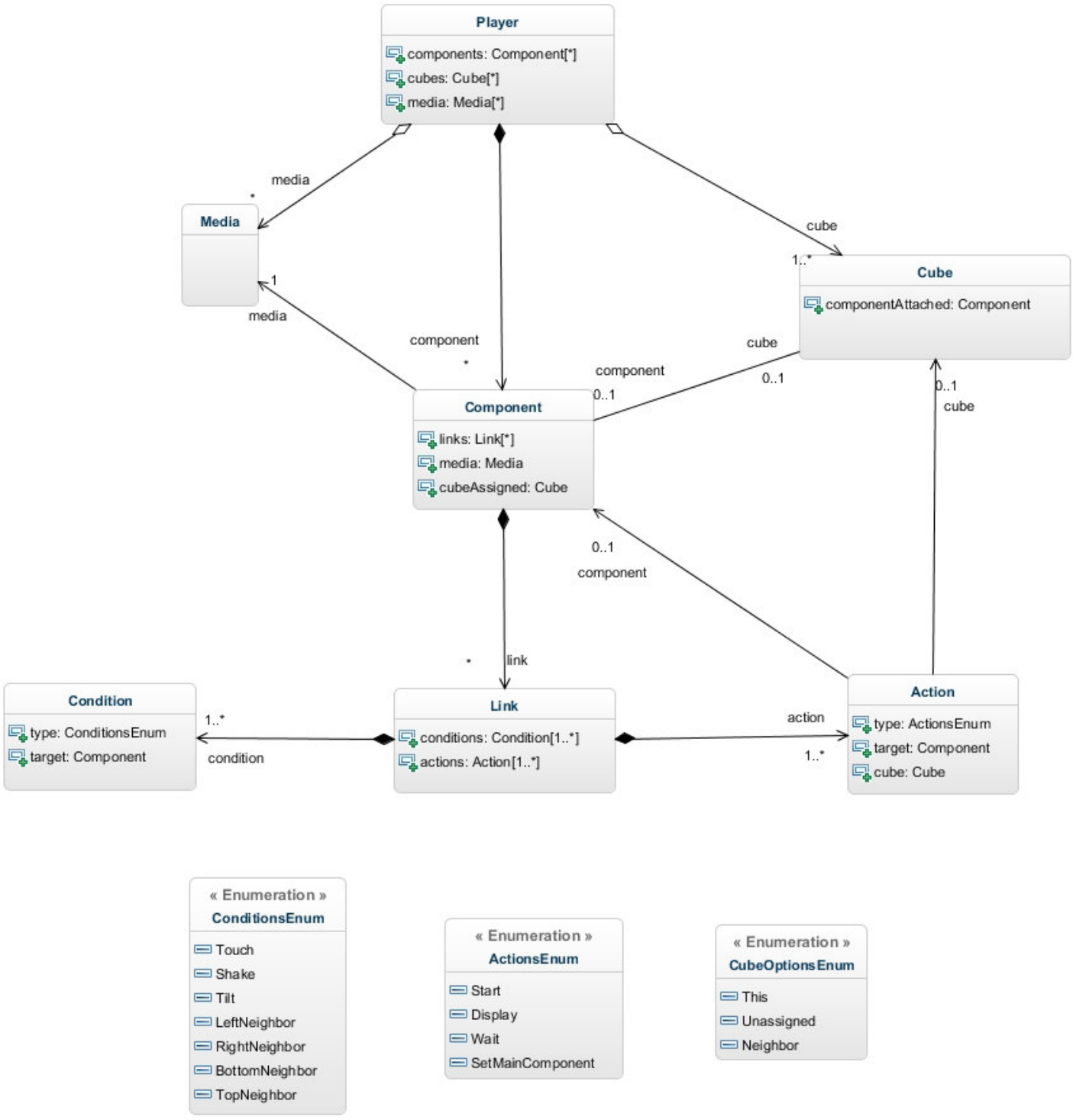

Figura 23 Diagrama de Classe do Comic Player

4.5 Sifteo Comics Tool

A compilação de ComicXML para C++ e Lua é feita pelo Sifteo Comics Tool. Essa ferramenta é responsável por gerir todos os processos das etapas 2 até 5 do diagrama apresentado na Figura 22.

Quando o usuário cria o documento CXML, ele deve executar o Sifteo Comics Tool e selecionar o documento criado numa janela de diálogo para que então possa gerar a aplicação ELF e enviar para o Sifteo Cube ou para o emulador “siftulator”.

Essa ferramenta foi desenvolvida em Java e utiliza o framework JAXP para manipulação de arquivos XML. Dessa forma, é possivel abstrair o processo de compilação de uma HQTrônica, pois o autor precisa somente escrever o documento de representação em ComicXML e utilizar o Sifteo Comics Tool para gerar e enviar a aplicação para o Sifteo Cubes.

Na figura a seguir é mostrado como o Sifteo Comics Tool abstrai o processo de autoria de HQTrônicas para os Sifteo Cubes, fazendo com que a autoria seja mais simples para os autores.

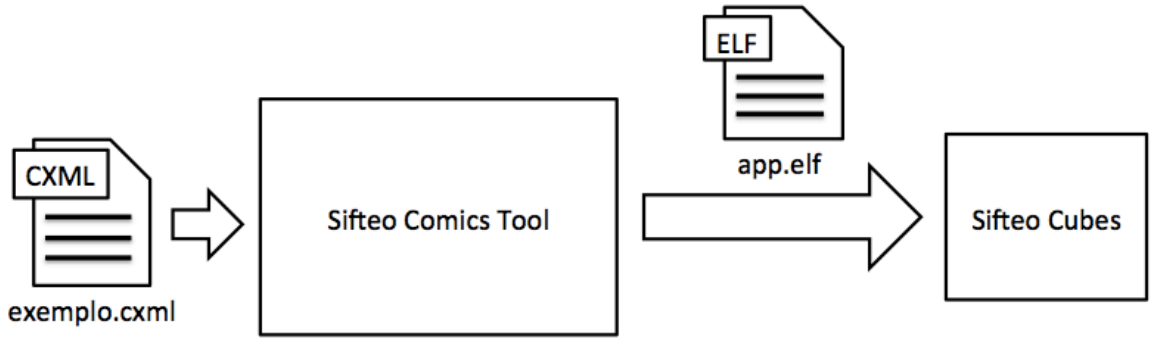

Figura 24 Abstração do Processo de Autoria

5 ESTUDO DE CASO

A fim de verificar o funcionamento do ambiente de autoria proposto, foram feitas duas HQTrônicas e compiladas para o Sifteo Cubes. As HQTrônicas foram idealizadas pela autora de histórias em quadrinhos e mestranda do programa de mestrado do curso de Design da UFMA, Márcia Ramos dos Santos. Este trabalho objetiva portanto contribuir para o trabalho de dissertação da aluna.

A primeira delas, A Formiga Amiga, aborda uma narrativa multilinear em que o usuário toma decisões para qual lago seguir; a segunda, Bully, explora a diagramação dinâmica para criar a sensação de animação. Nos exemplos há uso de recursos de interação da plataforma como aproximação de cubos e toques simples e duplos na tela.

5.1 HQTrônica A Formiga Amiga

Figura 25 Esquema da HQTrônica A Formiga Amiga

Para criar o documento que representa essa HQTrônica, cada imagem foi pensada como um componente que contém ações para iniciar outros componentes. Assim, o componente da primeira imagem foi definido como:

```
<component id="comp1" canbemain="true">
     <media target="imagem1"/>
     <linkBase>
          <link>
               <conditionBase>
                    <condition type="rightNeighbor"/>
               </conditionBase>
               <actionBase>
                    <action type="start" target="comp2"
cube="neighbor"/>
               </actionBase>
          </link>
     </linkBase>
</component>
```

Figura 26 Primeiro componente da HQTrônica em ComicXML

O documento Comic XML dessa HQTrônica encontra-se disponível no apêndice Apêndice B – ComicXML da HQTrônica A Formiga Amiga.

A seguir, uma série de imagens mostram a execução da aplicação do exibidor rodando a HQTrônica A Formiga Amiga:

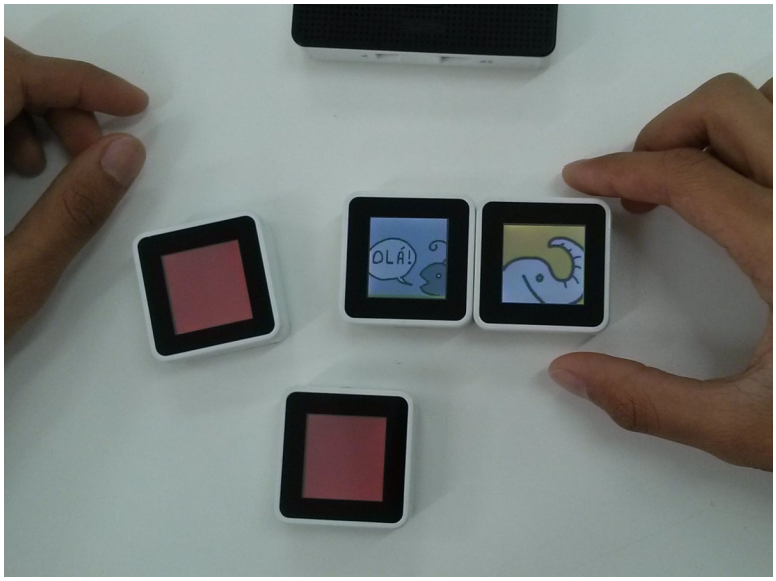

Figura 27 Interação de aproximação entre cubos

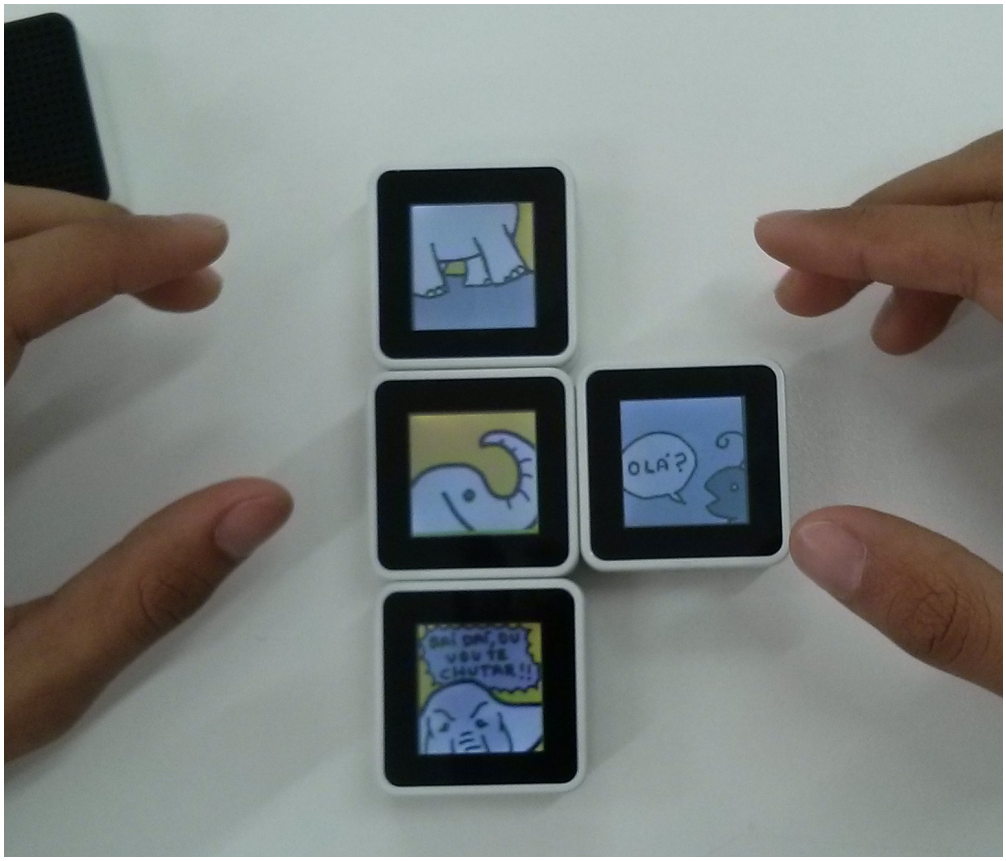

Figura 28 Usuário é convidado a escolher caminho da narrativa

Figura 29 Usuário reproduz uma narrativa

A execução dessa HQtrônica se fez bastante satisfatória. O processo de autoria durou pouco tempo e gerou um arquivo de 175 linhas, como mostrado no Anexo B.

5.2 HQTrônica Bully

Nessa história em quadrinho, o elemento diagramação dinâmica foi o mais explorado pela autora. A intenção é mostrar como a interação de toque em um cubo pode definir ações em outros cubos. Nesse caso, em um dos componentes, quando o cubo de um dos personagens é tocado, as imagens mostradas nos cubos adjacentes são alteradas. O esquema dessa HQTrônica é ilustrado na Figura 31.

A seguir a definição de três dos componentes especificados pela autora da HQTrônica:

```
<component id="niceguy" canbemain="false">
     <media target="niceguy"/>
     <linkBase>
          <link>
               <conditionBase>
                    <condition type="touch"/>
               </conditionBase>
               <actionBase>
                    <action type="start" target="sadguy"
cube="this"/>
                    <action type="start" target="dontpoke"
cube="1"/>
                    <action type="start" target="hihipoke"
cube="2"/>
                    <action type="setMainComponent"
target="dontpoke"/>
               </actionBase>
          </link>
     </linkBase>
</component>

<component id="pokehim" canbemain="true">
     <media target="pokehim"/>
</component>

<component id="wannasee" canbemain="false">
     <media target="wannasee"/>
</component>
```

Figura 30 Três componentes da HQTrônica em ComicXML

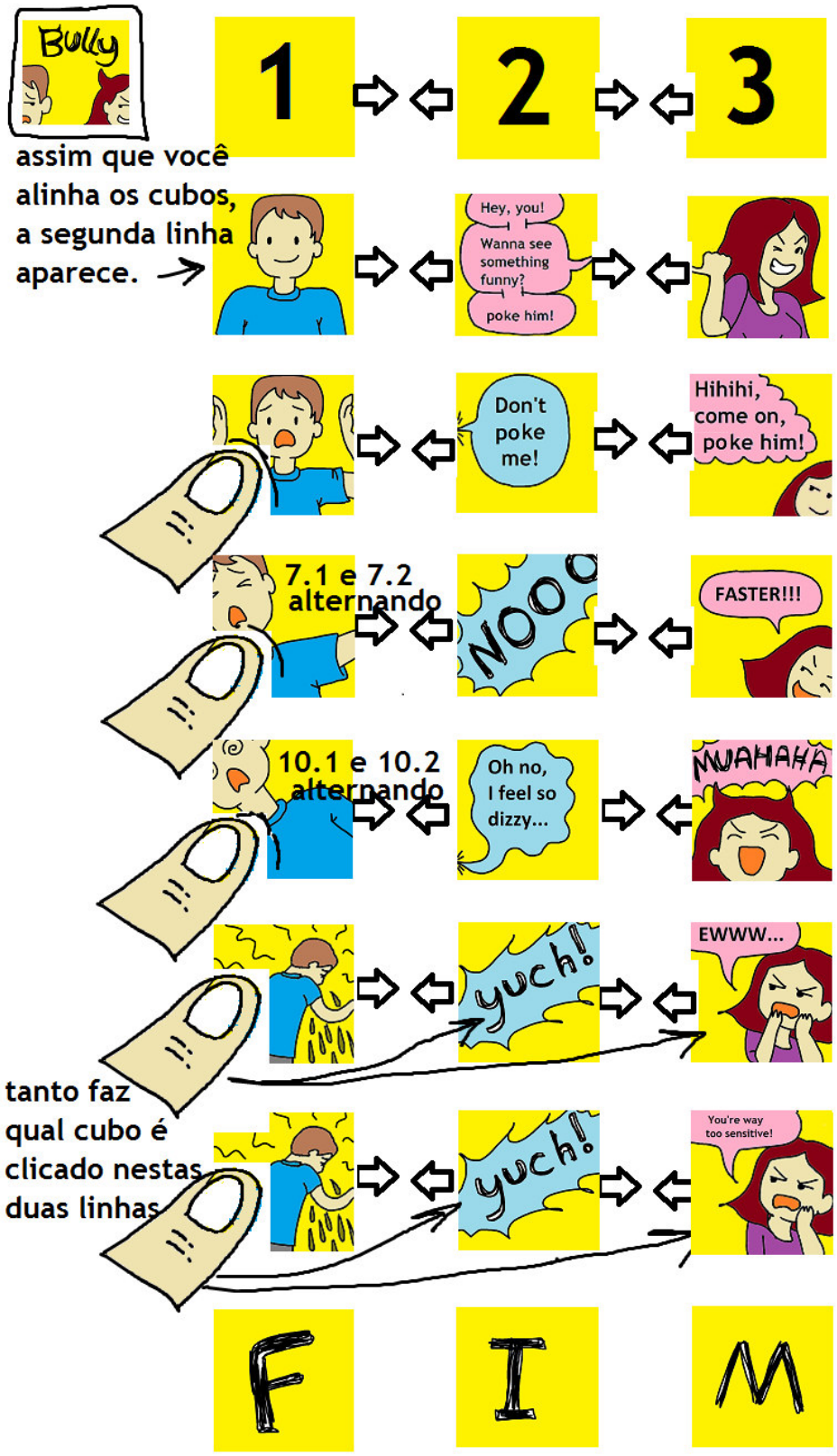

Figura 31 Esquema da HQTrônica Bully

As imagens a seguir mostram a HQTrônica sendo executada no simulador do Sifteo Cubes.

Figura 32 Imagens da HQTrônica sendo executada no simulador

A autoria dessa HQTrônica demorou em torno de uma hora, e consumiu 264 linhas em ComicXML. Mas uma vez, o processo foi bastante simples e muitas partes do código foram copiadas e coladas, com poucas alterações.

6. CONCLUSÃO

O Sifteo Cubes é uma plataforma inovadora que oferece um ambiente propício para aplicações interativas. A pouca oferta de aplicações para a plataforma juntamente com o preço do aparelho fazem com que ele ainda não tenha grande expressão no mercado. A pouca oferta de aplicações e jogos por sua vez pode ser explicada pelo esforço necessário para o desenvolvimento de aplicações para a plataforma, um esforço qualificado, que restringe quem pode desenvolver para a plataforma.

O trabalho proposto aqui vence essa barreira por permitir que mais pessoas desenvolvam aplicações para a plataforma. Isso é possível pois oferece uma outra forma de se desenvolver, uma forma mais fácil e intuitiva, que exige outras qualidades, no caso, conhecimento de XML.

O desenvolvimento do ambiente proposto neste trabalho mostrou que a autoria de HQTrônicas para a plataforma Sifteo Cubes pode ser de fato um processo mais simples, na medida em que oferece uma linguagem mais intuitiva para representar histórias em quadrinhos interativas e gera as aplicações na linguagem da plataforma automaticamente.

A autoria dos estudos de caso, com as HQTrônicas A Formiga Amiga e Bully, mostrou que o processo foi consideravelmente mais rápido. Além disso, o código necessário é menor e mais simples e intuitivo de se escrever. Com esta proposta, o autor não precisa mais ser um programador, mais ainda precisa saber como escrever documentos XML.

Por ser baseado em XML, linguagem declarativa mais intuitiva que linguagens orientadas a objetos, percebe-se que a autoria se tornou mais simples. Porém, autores sem perfil de programador precisariam ainda de treinamento em XML e em ComicXML para o desenvolvimento de suas HQTrônicas.

Com este trabalho, portanto, basta o uso da ComicXML para se criar HQTrônicas. Como trabalho futuro, um estudo sobre como autores sem perfil de programador lidam com essa abordagem e testes de usabilidade do ambiente podem ser realizados.

Talvez, mesmo os programadores que conheçam C++ e Lua escolherão essa abordagem para o desenvolvimento de HQTrônicas porque é necessário mais

que conhecer essas linguagens, é necessário aprender como funciona API da plataforma e conhecer o SDK para desenvolvimento.

Alguns recursos não foram implementados, como as interações de agitação e inclinação dos cubos. Porém isso não invalida o ambiente, já que outros recursos, como toque e aproximação, foram implementados, comprovando assim a eficácia da linguagem proposta.

Um ambiente gráfico seria o ideal para facilitar o processo de autoria de HQTrônicas para o Sifteo Cubes. O trabalho aqui desenvolvido pode no futuro ser base para uma ferramenta gráfica que possibilita a autoria de histórias em quadrinhos interativas. Para isso, a ferramenta deve oferecer ao autor a possibilidade de inserir componentes, definir imagens, criar links, com condições e ações, de forma gráfica, sem a necessidade de manipular diretamente o documento ComicXML.

REFERÊNCIAS

Andrews, D., Baber, C., Efremov, S., & Komarov, M. (2012). Creating and Using Interactive Narratives: Reading and Writing Branching Comics . *ACM SIGCHI Conference on Human Factors in Computing Systems.* Austin, Texas.

Azevedo, R. G., Neto, C. d., Teixeira, M. M., Santos, R. C., & Gomes, T. A. (2011). Textual Authoring of Interactive Digital TV Applications . *EuroITV .* Lisboa, Portugal: ACM.

Edwards, S. J. (1994). *Portable game notation specification and implementation guide.*

Franco, E. (2013). História em quadrinhos e hipermídia: As HQTrônicas chegam a sua terceira geração. In: L. Luiz, *Os Quadrinhos na era digital* (p. 158). Nova Iguaçu, Rio de Janeiro: Marsupial Editora.

Garone, P. M., Bernardi, B. P., & Santos, M. R. Histórias em quadrinhos impressas e digitais: uma análise dos elementos e das possibilidades. *DAMT: Design, Arte, Moda e Tecnologia.* São Paulo.

Gebhard, P., Kipp, M., Klesen, M., & Rist, T. (2003). Authoring Scenes for Adaptive, Interactive Performances . *Autonomous Agents and Multi-Agent Systems.* Melbourne.

Guimarães, R. L., Costa, R. M., & Soares, L. F. (2007). Composer: Ambiente de Autoria de Aplicações Declarativas para TV Digital Interativa . *WebMedia.* Gramado : ACM.

Hartmann, C., Direne, A., Bona, L., Silva, F., Santos, G. d., Castilho, M., et al. (2005). Linguagem e Ferramenta de Autoria para Promover o Desenvolvimento de Perícias em Xadrez. *XVI Simpósio Brasileiro de Informática na Educação .* Curitiba.

Lucero, A., Holopainen, J., & Jokela, T. (2012). MobiComics: Collaborative Use of Mobile Phones and Large Displays for Public Expression . *MobileHCI.* San Francisco: ACM.

Merrill, D., Kalanithi, J., & Maes, P. (2007). Siftables: Towards Sensor Network User Interfaces . *International Conference on Tangible, Embedded and Embodied Interaction.* Baton Rouge: ACM.

Merrill, D., Sun, E., & Kalanithi, J. (2012). Sifteo Cubes . *ACM SIGCHI Conference on Human Factors in Computing Systems.* Austin: ACM.

Santin, R., & Kirner, C. (2008). ARToolKit: Conceitos e ferramenta de autoria colaborativa. *Realidade Virtual e Aumentada: Uma Abordagem Tecnológica, SBC* , pp. 3-13.

Sifteo, Inc. (2013). *Sifteo - Intelligent Play*. Acesso em 2013, disponível em Sifteo: https://www.sifteo.com

Sifteo, Inc. (2013). *Sifteo SDK*. Acesso em 2013, disponível em Sifteo SDK v1.0.0: https://developers.sifteo.com/docs/SifteoSDK/1.0.0/index.html

Simioni, A., & Roesler, V. (2005). *Um framework para o desenvolvimento de aplicações interativas para a Televisão Digital.* Technical Report, Undergraduate Report, Universidade do Vale do Rio dos Sinos, São Leopoldo .

Stirbu, V., & Belimpasakis, P. (2011). Experiences Building a Multi-Display Mobile Application for Exploring Mirror Worlds . *Fifth International Conference on Next Generation Mobile Applications and Services .* Tampere, Finland : IEEE.

Sylla, C. (2013). Designing a Tangible Interface for Collaborative Storytelling to Access 'Embodiment' and Meaning Making . *Interaction Design and Children.* New York: ACM.

W3C. (s.d.). *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. Acesso em 2013, disponível em W3C: http://www.w3.org/TR/REC-xml/

W3Schools. (s.d.). *XML Schema Tutorial*. Acesso em 10 de 2013, disponível em W3Schools: http://www.w3schools.com/schema/

APÊNDICE A – XML SCHEMA DA COMICXML

O documento XML Schema descreve a estrutura de um documento XML. A seguir é apresentado o XML Schema para documentos ComicXML.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="sifteoHQ">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="begin">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="action">
                <xs:complexType>
                  <xs:simpleContent>
                    <xs:extension base="xs:string">
                      <xs:attribute type="xs:string" name="type"/>
                      <xs:attribute type="xs:string" name="target"/>
                      <xs:attribute type="xs:string" name="cube"/>
                    </xs:extension>
                  </xs:simpleContent>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="mediaBase">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="media" maxOccurs="unbounded" minOccurs="0">
                <xs:complexType>
                  <xs:simpleContent>
                    <xs:extension base="xs:string">
                      <xs:attribute type="xs:string" name="id" use="optional"/>
                      <xs:attribute type="xs:string" name="type" use="optional"/>
                      <xs:attribute type="xs:string" name="src" use="optional"/>
                    </xs:extension>
                  </xs:simpleContent>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="componentBase">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="component" maxOccurs="unbounded" minOccurs="0">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="media">
                      <xs:complexType>
                        <xs:simpleContent>
                          <xs:extension base="xs:string">
                            <xs:attribute type="xs:string" name="target" use="optional"/>
                          </xs:extension>
                        </xs:simpleContent>
                      </xs:complexType>
                    </xs:element>
                    <xs:element name="linkBase" minOccurs="0">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="link" maxOccurs="unbounded" minOccurs="0">
                            <xs:complexType>
```

```
                                <xs:sequence>
                                  <xs:element name="conditionBase">
                                    <xs:complexType>
                                      <xs:sequence>
                                        <xs:element name="condition">
                                          <xs:complexType>
                                            <xs:simpleContent>
                                              <xs:extension base="xs:string">
                                                <xs:attribute type="xs:string" name="type"
use="optional"/>
                                              </xs:extension>
                                            </xs:simpleContent>
                                          </xs:complexType>
                                        </xs:element>
                                      </xs:sequence>
                                    </xs:complexType>
                                  </xs:element>
                                  <xs:element name="actionBase">
                                    <xs:complexType>
                                      <xs:sequence>
                                        <xs:element name="action">
                                          <xs:complexType>
                                            <xs:simpleContent>
                                              <xs:extension base="xs:string">
                                                <xs:attribute type="xs:string" name="type"
use="optional"/>
                                                <xs:attribute type="xs:string" name="target"
use="optional"/>
                                                <xs:attribute type="xs:string" name="cube"
use="optional"/>
                                              </xs:extension>
                                            </xs:simpleContent>
                                          </xs:complexType>
                                        </xs:element>
                                      </xs:sequence>
                                    </xs:complexType>
                                  </xs:element>
                                </xs:sequence>
                              </xs:complexType>
                            </xs:element>
                          </xs:sequence>
                        </xs:complexType>
                      </xs:element>
                    </xs:sequence>
                    <xs:attribute type="xs:string" name="id" use="optional"/>
                    <xs:attribute type="xs:string" name="canbemain" use="optional"/>
                  </xs:complexType>
                </xs:element>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:schema>
```

APÊNDICE B – COMICXML DA HQTRÔNICA A FORMIGA AMIGA

A seguir é apresentado o documento ComicXML da HQTrônica A Formiga Amiga.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<sifteoHQ xmlns:ns2="xml.jaxb.ComicXML.SifteoHQ">
      <begin>
       <action type="start" target="comp1" cube="unassigned"/>
      </begin>

      <mediaBase>
       <media id="imagem1" type="img" src="assets/B1.png"/>
       <media id="imagem2" type="img" src="assets/B2.png"/>
       <media id="imagem3" type="img" src="assets/B3.png"/>
       <media id="imagem4" type="img" src="assets/B4.png"/>
       <media id="imagem5" type="img" src="assets/C4.png"/>
       <media id="imagem6" type="img" src="assets/D4.png"/>
       <media id="imagem7" type="img" src="assets/A2.png"/>
       <media id="imagem8" type="img" src="assets/C2.png"/>
       <media id="imagem9" type="img" src="assets/A3.png"/>
       <media id="imagem10" type="img" src="assets/C3.png"/>
       <media id="imagem11" type="img" src="assets/D3.png"/>
      </mediaBase>

      <componentBase>
       <component id="comp1" canbemain="true">
              <media target="imagem1"/>
              <linkBase>
                     <link>
                            <conditionBase>
                                   <condition type="rightNeighbor"/>
                            </conditionBase>
                            <actionBase>
                                   <action type="start" target="comp2"
cube="neighbor"/>
                            </actionBase>
                     </link>
              </linkBase>
       </component>
       <component id="comp2" canbemain="true">
              <media target="imagem2"/>
              <linkBase>
                     <link>
                            <conditionBase>
                                   <condition type="rightNeighbor"/>
                            </conditionBase>
                            <actionBase>
                                   <action type="start" target="comp3"
cube="neighbor"/>
                            </actionBase>
                     </link>
                     <link>
                            <conditionBase>
                                   <condition type="topNeighbor"/>
                            </conditionBase>
                            <actionBase>
                                   <action type="start" target="comp7"
cube="neighbor"/>
                            </actionBase>
                     </link>
                     <link>
                            <conditionBase>
```

```
                                <condition type="bottomNeighbor"/>
                        </conditionBase>
                        <actionBase>
                                <action type="start" target="comp8"
cube="neighbor"/>
                        </actionBase>
                </link>
        </linkBase>
    </component>
    <component id="comp3" canbemain="true">
        <media target="imagem3"/>
        <linkBase>
                <link>
                        <conditionBase>
                                <condition type="rightNeighbor"/>
                        </conditionBase>
                        <actionBase>
                                <action type="start" target="comp4"
cube="neighbor"/>
                        </actionBase>
                </link>
        </linkBase>
    </component>
    <component id="comp4" canbemain="true">
        <media target="imagem4"/>
        <linkBase>
                <link>
                        <conditionBase>
                                <condition type="bottomNeighbor"/>
                        </conditionBase>
                        <actionBase>
                                <action type="start" target="comp5"
cube="neighbor"/>
                        </actionBase>
                </link>
        </linkBase>
    </component>
    <component id="comp5" canbemain="true">
        <media target="imagem5"/>
        <linkBase>
                <link>
                        <conditionBase>
                                <condition type="bottomNeighbor"/>
                        </conditionBase>
                        <actionBase>
                                <action type="start" target="comp6"
cube="neighbor"/>
                        </actionBase>
                </link>
        </linkBase>
    </component>
    <component id="comp6" canbemain="true">
        <media target="imagem6"/>
    </component>
    <component id="comp7" canbemain="true">
        <media target="imagem7"/>
        <linkBase>
                <link>
                        <conditionBase>
                                <condition type="rightNeighbor"/>
                        </conditionBase>
                        <actionBase>
                                <action type="start" target="comp9"
cube="neighbor"/>
                        </actionBase>
                </link>
```

```
            </linkBase>
        </component>
        <component id="comp8" canbemain="true">
            <media target="imagem8"/>
            <linkBase>
                <link>
                    <conditionBase>
                        <condition type="rightNeighbor"/>
                    </conditionBase>
                    <actionBase>
                        <action type="start" target="comp10"
cube="neighbor"/>
                    </actionBase>
                </link>
            </linkBase>
        </component>
        <component id="comp9" canbemain="true">
            <media target="imagem9"/>
            <linkBase>
                <link>
                    <conditionBase>
                        <condition type="bottomNeighbor"/>
                    </conditionBase>
                    <actionBase>
                        <action type="start" target="comp3"
cube="neighbor"/>
                    </actionBase>
                </link>
            </linkBase>
        </component>
        <component id="comp10" canbemain="true">
            <media target="imagem10"/>
            <linkBase>
                <link>
                    <conditionBase>
                        <condition type="bottomNeighbor"/>
                    </conditionBase>
                    <actionBase>
                        <action type="start" target="comp11"
cube="neighbor"/>
                    </actionBase>
                </link>
            </linkBase>
        </component>
        <component id="comp11" canbemain="true">
            <media target="imagem11"/>
            <linkBase>
                <link>
                    <conditionBase>
                        <condition type="rightNeighbor"/>
                    </conditionBase>
                    <actionBase>
                        <action type="start" target="comp6"
cube="neighbor"/>
                    </actionBase>
                </link>
            </linkBase>
        </component>
      </componentBase>

</sifteoHQ>
```