O USO DE TABLETS PARA ANÁLISE DE CONTROLE MOTOR PELA SOBREPOSIÇÃO DE TRAÇOS

Tiago Pereira Vidigal, Antônio Padilha L. Bo

***Resumo* – O projeto propõe a inserção de tablets em testes de controle motor, amplamente usados em exames psicotécnicos e avaliações de disfunções motoras. O objetivo é aumentar a precisão dos referidos testes. A metodologia usada se dividide nas seguintes etapas: (a) revisão bibliográfica, (b) desenvolvimento de aplicativo, (c) testes, (d) análise dos resultados. O aplicativo resultante possibilita a execução de avaliações de sobreposição de traços pré-definidos, visualização do desenho feito pelo usuário, destaque dos acertos e erros referentes ao traço escolhido, porcentagem de acerto e navegação entre os resultados obtidos na execução. Sua implementação permite o uso de qualquer tipo de linha, além de disponibilizar tabelas globais com todos os dados dos testes. Logo, aplicativos em tablets podem substituir os exames em papel utilizados atualmente. Desta forma, eles são aprimorados em precisão e objetividade, assim como em funcionalidade, devido a possibilidade de tratamentos complementares.**

***Index Terms* – Tablet, controle motor, disfunção motora, linguagem Lua.**

I. INTRODUÇÃO

Em várias situações cotidianas, testes de controle motor são aplicados. Por exemplos: durante os testes necessários para obter a habilitação para conduzir veículos e para aferir a existência de alguma disfunção motora. Com isso, sua utilização demonstra-se ampla e seus objetivos transitam desde auxiliar em uma avaliação psicotécnica até na busca da existência da doença de Alzheimer em pacientes [1].

Em muitos casos, tais testes são realizados de forma subjetiva e, de certa forma, imprecisa. Sua análise é feita por uma pessoa que procura padrões no desenho feito por um usuário. Computadores, que podem processar informações e fazer buscas de padrões de forma mais eficiente e precisa, podem auxiliar em operações como esta. Visando este aprimoramento, tablets mostram ser máquinas muito interessantes na substituição do papel onde os testes são feitos. Estes dispositivos, pela sua naturalidade na escrita manual, têm progressivamente adentrado em várias áreas, inclusive na educação, substituindo cadernos.

Expostos estes pontos, este trabalho propõe o uso de tablets e/ou smartphones na aplicação de testes de controle motor. Um aplicativo foi produzido, afirmando a viabilidade destes dispositivos para verificar o desempenho do usuário. Como exemplo, usou-se testes de sobreposição de traço. O programa demonstrou que o processamento futuro de dados obtidos pelas avaliações também é possível, o que possibilita futuras modificações ou novas aplicações aprimorarem o tratamento de dados após a execução.

II. METODOLOGIA

A metodologia usada é dividida nas seguintes etapas: (a) revisão bibliográfica, (b) desenvolvimento de aplicativo, (c) testes, (d) análise dos resultados. A etapa de desenvolvimento foi separada nas subetapas: (1) formas de traço, (2) fluxo do aplicativo, (3) detecção do desenho, (4) captação e separação de pontos conforme o usuário desenha; (5) tratamento dos dados; (6) disponibilização das informações de forma global. Cada etapa é detalhada a seguir.

(a) *Revisão Bibliográfica*

Para o desenvolvimento de aplicativos para tablets, três linguagens de programação mostraram-se interessantes: Java+XML, Objective-C e Lua. A linguagem Java com XML seria a padrão para sistemas operacionais mobile da Google (Android), enquanto a Objective-C seria a padrão para sistemas da Apple (iOS). O linguagem Lua, desenvolvida na Pontifícia Universidade Católica do Rio de Janeiro (PUC-RJ), foi a escolhida para a implementação.

Inicialmente, o Objective-C foi utilizado. Depois de mais pesquisas, Lua se mostrou mais vantajosa. Lua é uma linguagem procedural baseada em tabelas associativas com semântica extensível, além de ser executada em uma máquina virtual e possuir *garbage collector* (gerenciador de memória que elimina alocações desnecessárias). Como ilustração, Lua pode ser comparada com uma mistura de C com Java. Ela possui uma ótima documentação e já foi usada em projetos grandes, como implementação do middleware Ginga (componente fundamental do sistema de TV digital brasileiro) e jogos como World of Warcraft e Angry Birds. [2][3]

Lua, além de mostrar baixo custo (treinamento simples e capacidade de escrita e adequação alta), também se mostrou robusta. Sua portabilidade também é interessante, uma vez que a linguagem pode ser usada em conjunto com o Corona SDK no desenvolvimento para tablets com SO Android ou iOS, sem contar com sua natural compatibilidade com programas em linguagem C. Desta forma, Lua foi adotada para implementar o aplicativo. Todo seu estudo foi baseado na documentação das APIs disponibilizada pelo Corona [4] e pelo livro *Programming in Lua* [5].

(b) Desenvolvimeto de aplicativo

(1) Formas de traço

O objetivo do teste é o desenho executado pelo usuário sobrepor o mais precisamente possível a forma apresentada na

tela. Os traços selecionados para realizar esses testes foram a linha vertical, a linha horizontal, uma circunferência e um traço sem forma definida. A escolha foi feita com dois focos: auxiliar os testes de captação dos pontos na tela; trabalho motor realizado pelo usuário ao executar o desenho. Estes pontos são abordados nos parágrafos seguintes, respectivamente.

As formas irão auxiliar nos testes de captação de pontos. As duas primeiras são as mais básicas, escolhidas para trabalhar com os limites nos eixos x e y. A terceira foi escolhida para verificar se os pontos internos a figura são captados (situação de erro). O último demonstrará que qualquer contorno pode ser utilizado seguindo a implementação mostrada a seguir.

O crescente nível de dificuldade e complexidade apresentado dos desenhos pode ser usado para testar a coordenação motora de diversos músculos do braço. Enquanto as linhas retas são mais simples, os traços curvilíneos envolvem o recrutamento de músculos e articulações adicionais. Devido a isso, diferentes níveis de planejamento e coordenação motora para correta execução são requeridos.

(2) Fluxo do aplicativo

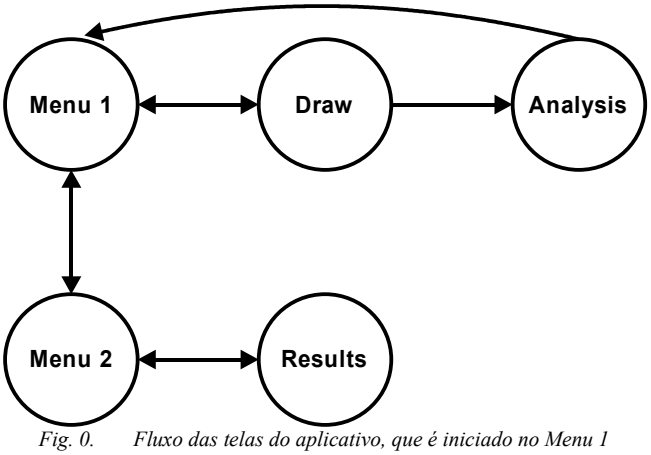

Fig. 0. Fluxo das telas do aplicativo, que é iniciado no Menu 1

O aplicativo que será descrito a seguir implementa o teste motor por sobreposição de traços. Ao selecionar o teste no "Menu 1", o usuário deve tentar repetir a figura mostrada na tela "Draw", onde os pontos serão captados. Em "Analysis", o software processará os dados e, ao final, a taxa de acerto e o desenho do usuários serão exibidos na tela. Após algumas execuções, todos os resultados podem ser verificados em "Results". A Figura 0 mostra o fluxo graficamente e as Figuras 1, 2 e 3 mostram as telas implementadas.

(3) Detecção do desenho

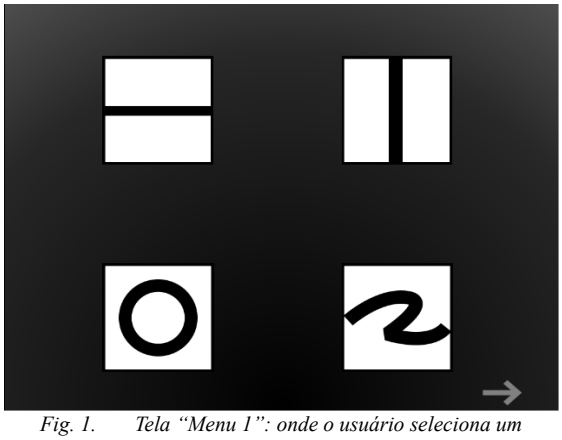

Fig. 1. Tela "Menu 1": onde o usuário seleciona um traço pré-definido a ser usado no teste de sobreposição.

Os traços disponíveis são mostrados no "Menu 1" de forma que o usuário possa decidir qual deles irá desenhar e ser avaliado. O menu pode ser visualizado na Figura 1. Na tela "Draw", imagens PNG foram usadas para mostrar na tela a forma a ser sobreposta, como mostrado na Figura 2. As gravuras possuem fundo transparente e o traço a ser feito em preto.

Fig. 2. Tela "Draw": onde o usuário irá tentar repetir o desenho sobre ele mesmo e os pontos serão capturados.

Um problema ocorreu quando traços diferentes da linha horizontal ou vertical eram empregados. Em Lua, ao usar uma imagem como *displayObject*, qualquer parte do objeto é considerada traço, incluindo a parte transparente. Por tanto, várias formas de traço não poderiam ser usadas, o que limitava tremendamente o uso do aplicativo.

O problema foi resolvido com máscaras. Linhas inclinadas ou curvilíneas exigem, além da imagem com fundo transparente, criar a chamada *mask*. Ela é idêntica à gravura original, porém com fundo preto e o traço em branco. Ela é adotada para filtrar a figura de forma que a parte em preto (fundo) torna transparente a mesma área no display final e não aceita eventos do tipo *touch*. O resto do filtro (que está em branco) deixa visível o display final e aceita eventos de toque na tela. A implementação feita no aplicativo foi:

```
1. elseif draw = = "curva_simples" then
2.         figure = display.newImage("drawCS.png")
3.         figure:translate (242, 114)
4.         mask = graphics.newMask( "maskCS.png" )
```

Com o uso desta técnica, qualquer imagem pode ser utilizada para o teste, apenas gerando-se a máscara correspondente à forma do traço desejado. Isso permite à aplicação ser ampliada para qualquer situação de sobreposição, além de abranger outros objetivos, como treino de caligrafia de alfabetos ocidentais ou orientais. Adicionalmente, o traço branco na *mask* pode ser mais largo que a forma a ser sobreposta. Isso causaria uma margem de erro ajustável, facilitando o acerto.

(4) Captação e separação de pontos

A captação dos pontos mostrou-se relativamente simples com a resolução do problema aludido na seção anterior. Duas tabelas foram separadas para cada desenho: uma contendo todos os pontos gerados e outra contendo apenas os corretos (que sobreporam o formato sendo testado). Com estas tabelas, uma terceira de incorretos não foi necessária, pois esta pode ser gerada a partir das duas já existentes.

Dois métodos ouvintes foram criados, um que atuará apenas no display (o traço) e outro para a tela como um todo. Com o uso da máscara, o primeiro ouvinte será executado apenas quando o ponto desenhado pelo usuário esteja dentro da faixa de acerto definida pelo traço branco na *mask*.

Com o usuário movendo o dedo ou caneta na tela, o método da tela toda captura e armazena as coordenadas X e Y em um índice da tabela de todos os pontos. Caso uma posição obtida coincida com a forma definida, o método do display também é ativado e o mesmo par de coordenadas é guardado na tabela de pontos certos. As funções, adaptadas do programa, são:

```
1.  function getXY( event )
2.          if event.phase = = "moved" then
3.                  table.insert(allPoints, {} )
4.                  allPoints[#allPoints].X = event.x
5.                  allPoints[#allPoints].Y = event.y
6.          end
7.  end
8.
9.  function getGoodXY( event )
10.         if event.phase == "moved" then
11.          table.insert (goodPoints+1, {} )
12.          goodPoints[#goodPoints].X = event.x
13.          goodPoints[#goodPoints].Y = event.y
14.         end
15. end
```

Desta forma, todos os pontos desenhados foram capturados e separados nas tabelas de pontos totais e os corretos. Esta separação permitirá uma manipulação mais efetiva dos pontos gerados pelo teste.

(5) Tratamento sobre os pontos

Os tratamentos desenvolvidos foram a reprodução do desenho na tela e o cálculo da taxa de pontos certos pelo total. A possibilidade de mostrar visualmente as partes em que o usuário acertou e errou também foi criada.

Na janela de análise, os pontos traçados durante o teste são mostrados na tela. Para isto, se percorre a tabela que contém todos os pares de coordenadas X e Y e os imprime na tela. A representação da figura foi gerada com círculos de centro (X,Y) e raio 2 de cor preta. As formas criadas foram armazenadas em um grupo chamado *noDetails*. Este objectGroup representa o desenho sem detalhes.

```
1.  while allPoints[count] ~= nil do
2.          pointX = allPoints[count].X
3.          pointY = allPoints[count].Y
4.
5.          point = display.newCircle(pointX, pointY, 2)
6.          point:setFillColor(0,0,0)
7.          noDetails:insert( point )
8.          (… )
```

Durante o passo supracitado, os pontos considerados corretos recebem um segundo tratamento. O procedimento adicional consiste em criar um novo círculo, idêntico ao anterior, porém de cor diferente. Sendo bom, a cor utilizada é verde. Caso contrário, a cor é vermelha. Os displays gerados também são alocados em grupos (*goodDetails* e *badDetails*). Este procedimento é mostrado a seguir (com edições). O trecho é complementar do código mostrado antes deste parágrafo:

```
8.  -- If it is good, save a green point
9.  if goodPoints[countAux] ~= nil
10.   and pointX == goodPoints[countAux].X
11.   and pointY == goodPoints[countAux].Y then
12.         point = display.newCircle(pointX, pointY, 2)
13.         point:setFillColor(0,255,0)
14.         goodDetails:insert( point )
15.
16. -- Else, save a red point
17. else
18.         point = display.newCircle(pointX, pointY, 2)
19.         point:setFillColor(255,0,0)
20.         badDetails:insert( point )
21. end -- End of the if
22. end -- End of the while
```

A taxa de acerto é calculada a partir das tabelas que contêm os pontos. O calculo é feito dividindo-se o número de elementos da tabela *goodPoints* pelo da tabela *allPoints*. Uma frase declarando o valor do *rate* é exibido na tela.

```
1.  rateXY = (goodPoints/allPoints)*100
2.  textRateXY = display.newText
                    ("The rate was: "..rateXY.."%", 100, 700)
3.  textRateXY.size = 25
4.  textRateXY:setTextColor(0,0,0)
```

(6) Disponibilização global das informações

Os dados extraídos dos desenhos do usuário são facilmente acessados para aplicações extras se acoplarem. Desta maneira, é possível abranger mais a utilidade dos aplicativo, permitindo diferentes processamentos a partir do desenhos do usuário. No programa construído, todos os resultados dos testes feitos pelo usuário podem ser obtidos de forma global. Não existe persistência, portanto são disponibilizadas apenas informações em *runtime*.

O desenvolvimento da tela de resultados (que possibilita verificar o rendimento do usuário) se mostrou bem parecido com o da tela de análise (mostrada após o teste de um traço ser feito). No entanto, os pontos são acessados por uma tabela global disponibilizando todas as coordenadas de todos os desenhos feitos em tempo de execução. Também são disponibilizados os pontos certos e a taxa de acerto de cada teste.

III. TESTES

A execução de testes de controle motor em tablets é possível, como a implementação do aplicativo descrito ao longo da metodologia demonstra. Formas de traço diferentes podem ser utilizadas com a técnica apresentada. Tanto imagens com áreas internas vazias (como um círculo) ou curvas sem uma forma definida podem ser analisadas, sem apresentar erros de captação. Para testar a captação e tratamento de pontos, fizemos dois testes mostrados nas Figuras 3, 4 e 5.

A análise seguida do teste da linha horizontal mostra na tela "Analysis" o desenho feito pelo usuário. Como a Figura 3 expõe, os pontos foram guardados e imprimidos na tela.

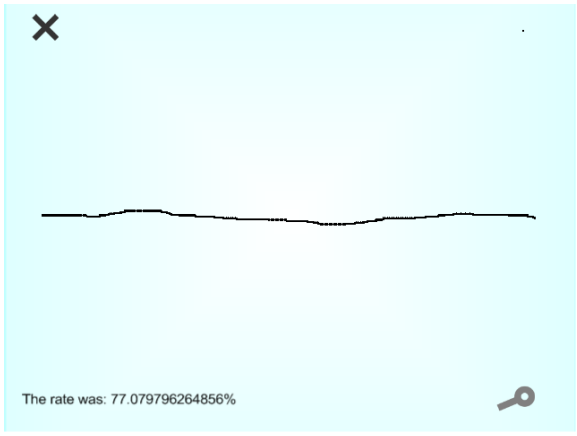

Fig. 3. Tela "Analysis" mostrando o desenho gerado com base nos pontos capturados a partir do desenho feito pelo usuário.

O destaque dado para os erros e acertos na Figura 4 demonstra que os dados coletados podem ser manipulados (neste caso, para gerar pontos). Complementarmente, as informações geraram uma outra saída (no caso, o rate), demonstrando as possibilidades de processamento.

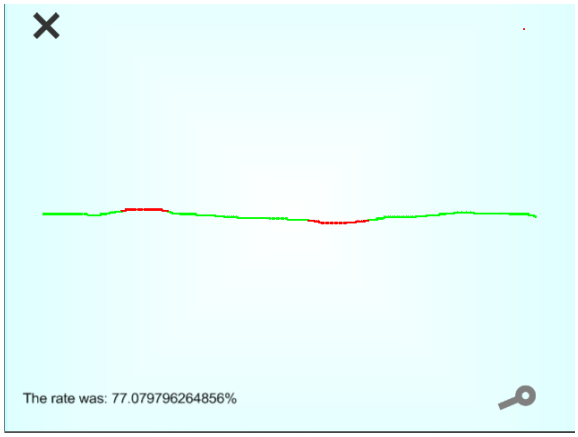

Fig. 4. Tela "Analysis" com pontos corretos e incorretos da Figura 3 sendo mostrados. Note a taxa de acerto.

Os pontos (certos e errados) evidenciam que o desenho foi captado corretamente, sem erros. Note na Figura 5 que os cantos transparentes e a parte interna da forma não foram detectados.

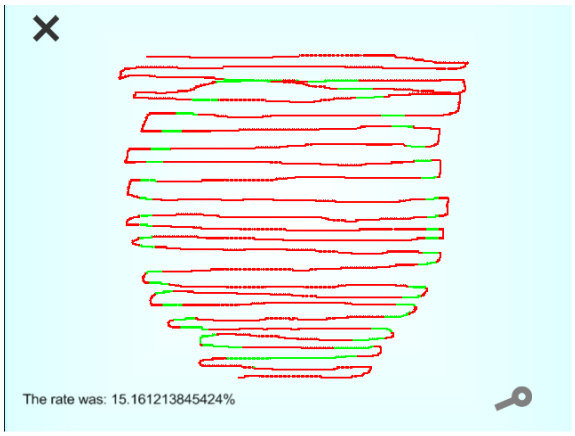

Fig. 5. Imagens possuem formas retangulares, independente da forma desenhada. Utilizando máscara, é demonstrado nesta figura que a parte interna e as quinas não são consideradas corretos pontos

Os dados coletados não ficaram retidos ao programa. Eles foram disponibilizados de forma global, como foi demonstrado no caso da tela "Results". Com essa atitude, qualquer outra aplicação pode ser acoplada a este aplicativo, possibilitando diversos processamentos sobre as informações do desenho.

IV. ANÁLISE DE RESULTADOS

A proposta de aprimoramento de testes de controle motor se provou possível e simples de ser implementada e expandida. O programa desenvolvido aborda um dos tipos de teste, o teste de sobreposição de traço, porém as ideias e métodos utilizados podem ser aproveitados para outros tipos de avaliação. A altíssima precisão e a possibilidade de tratamento de dados são possibilidades fornecidas pelo uso de tablets. As informações coletadas podem ser acessadas globalmente, possibilitando gravar as informações em disco, por exemplo. A busca de padrões de erro a partir das informações do desenho também pode ser explorada.

O uso dos tablets é viável e deve ser implementado. A subjetividade do teste, um defeito apresentado nas avaliações atuais, é eliminada. A forma de se avaliar estes testes de coordenação é aprimorada de forma satisfatória com o uso desta tecnologia. Em suma, a frágil forma de detecção de disfunções motoras usada atualmente deve e pode ser melhorada utilizando aplicações como a apresentada neste artigo.

REFERÊNCIAS

[1] AlzheimerMed, "Avaliação Cognitiva" at http://www.alzheimermed.com.br/diagnostico/avaliacao-cognitiva

[2] Roberto Ierusalimschy, "Uma Introdução à Programação em Lua" in *JAI 2009,* Jul, 2009

[3] Roberto Ierusalimschy, Luiz Henrique de Figueiredo, Waldemar Celes Filho, "Lua - an extensible extension language" in *Software: Practice & Experience* 26 #6, 1996 pp. 635–652

[4] Corona Docs, "SDK API Reference" at http://docs.coronalabs.com/

[5] Roberto Ierusalimschy, *Programming in Lua* ASCII Media Works 1st edition, Aug, 2009