

Dissertação de MAC499: jogos para análise de
capacidades executivas em crianças de 4 a 7 anos
de idade, jogo para criança com AME tipo I, e um
Sistema de Autoria de Aplicativos com Interface
Sequencial

Celso Masahiro Shimabukuro

3 de dezembro de 2013

Sumário

1	Parte objetiva	5
1.1	Introdução	5
1.2	Objetivos gerais	5
1.2.1	Jogos para avaliar as crianças	5
1.2.2	Jogo para uma criança portadora de AME tipo I	6
1.2.3	Sistema de autoria de aplicativos com interface sequencial	6
1.3	Conceitos e tecnologias estudadas	6
1.3.1	PsychoPy	6
1.3.2	pygame	6
1.4	Jogos	7
1.4.1	Dots: descrição e objetivos	7
1.4.2	Flanker: descrição e objetivos	8
1.4.3	Ursos	9
1.4.4	Jogo musical	9
1.4.5	Sistema de autoria de aplicativos com interface sequencial	12
1.5	Conclusões	13
2	Parte subjetiva	15
2.1	Desafios e frustrações	15
2.2	Lista das disciplinas mais relevantes para o trabalho	15
2.3	Agradecimentos	15

Capítulo 1

Parte objetiva

1.1 Introdução

Este trabalho foi feito no ano de 2013, e contou com a participação do professor Carlos Hitoshi Morimoto como orientador, da psicóloga Liege Felício do Instituto de Psiquiatria da USP e da fisioterapeuta Graziela Polido, mestranda na USP.

1.2 Objetivos gerais

Há mais de um objetivo para este trabalho de conclusão de curso:

Desenvolver jogos de computador para avaliar crianças participantes do projeto de intervenção de capacidades executivas de L. Felício do Instituto de Psiquiatria da USP.

Desenvolver um jogo que seja adequado para ser utilizado por uma criança portadora de Amiotrofia Muscular Espinhal (Atrofia Muscular Espinhal tipo I). A inspiração para este projeto é o paciente de Graziela Polido, Arthur.

Desenvolver um sistema de autoria de aplicativos com interface sequencial, inspirado na criação do jogo anterior. A ideia é facilitar a criação de novos aplicativos com a mesma estrutura do jogo criado para o Arthur.

1.2.1 Jogos para avaliar as crianças

A ideia do projeto de intervenção de capacidades executivas é desenvolver as chamadas capacidades executivas do grupo participante: memória, inibição (capacidade de se concentrar em uma tarefa e ignorar demais estímulos) e flexibilidade cognitiva (capacidade de se adaptar à mudança de regras). Esses jogos se inserem em um conjunto de testes destinados a verificar a evolução das capacidades cognitivas dessas crianças participantes. Esses jogos já foram entregues à equipe do IPq.

1.2.2 Jogo para uma criança portadora de AME tipo I

Este projeto foi trazido por Graziela Polido destinado a um caso especial: seu paciente, chamado Arthur, portador de Amiotrofia Muscular Espinhal tipo I. A AME tipo I é caracterizada por diminuição dos movimentos musculares antes dos 6 meses de idade, sendo que a criança não chega a andar nem sentar-se por conta própria.

1.2.3 Sistema de autoria de aplicativos com interface sequencial

Nas reuniões a respeito do problema anterior, discutimos a possibilidade de, ao invés de apenas criarmos um jogo seguindo o formato que pensamos, criarmos uma ferramenta para criar aplicativos semelhantes ao jogo criado.

1.3 Conceitos e tecnologias estudadas

A linguagem escolhida para desenvolver os jogos foi python, e com ela o PsychoPy, um software próprio para criação de testes neurológicos e neuropsicológicos.

Para o jogo musical, escolhemos usar python e pygame. Scipy também foi utilizada.

Para o sistema de autoria de aplicativos com interface sequencial, escolhemos python pela conveniência de podermos nos beneficiar com o uso da biblioteca Swing e da biblioteca pygame ao mesmo tempo.

1.3.1 PsychoPy

“O PsychoPy é um pacote open-source para a execução de experimentos em Python (uma alternativa real e livre para o Matlab). PsychoPy combina os pontos fortes de gráficos OpenGL com a sintaxe fácil do Python para dar aos cientistas um sistema gratuito e simples para apresentar estímulos e controlá-los. Ele é usado por muitos laboratórios em todo o mundo para criar experimentos em psicofísica, neurociência cognitiva e psicologia experimental.”[1]

1.3.2 pygame

“O Pygame é um conjunto de módulos Python próprio para fazer jogos. Pygame adiciona funcionalidades ao topo da excelente biblioteca SDL. Ele permite a criação de jogos cheios de recursos e programas multimídia na linguagem python.”[2]

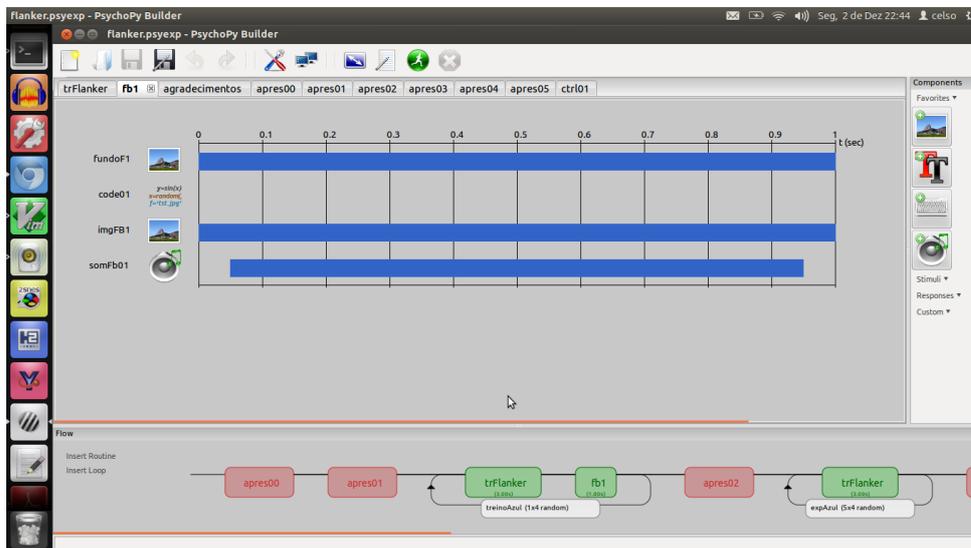


Figura 1.1: PsychoPy modo Builder

1.4 Jogos

Foram desenvolvidos três jogos usando a linguagem python e a biblioteca PsychoPy. Dois deles, chamados Dots e Flanker, são uma reprodução de testes que foram descritos em Matthew C. Davidson et al., 2006 [5]

1.4.1 Dots: descrição e objetivos

O jogo é destinado a avaliar a flexibilidade cognitiva da criança. O jogo é composto de três testes, sendo que os dois primeiros testes possuem uma fase de treinamento. Durante o treino, a criança aprende as regras do jogo e tem a oportunidade de se familiarizar com seu visual e modo de funcionamento. No primeiro teste, é exibida uma figura (um coração) no lado direito ou no lado esquerdo da tela e a criança deve apertar no joystick o mesmo lado em que a figura aparece. Esse teste é chamado de congruent, porque a resposta esperada é congruente ao estímulo apresentado.

No segundo teste (chamado incongruent), a figura exibida é uma flor, e a criança deve aprender a apertar do lado oposto ao da figura, isto é, se a flor aparece do lado esquerdo, a criança deve apertar para a direita, e se a flor aparecer do lado direito, deve apertar para a esquerda.

Por fim, é feito um teste mixed: tanto uma flor quanto um coração podem aparecer, e a criança precisa ser capaz de aplicar a regra adequada a cada contexto; deve clicar no mesmo lado em que aparece um coração ou no lado oposto ao que aparece uma flor.

A bateria de testes é composta de 76 trials, na seguinte ordem: 8 treinos congruent, 20 congruent, 8 treinos incongruent, 20 incongruent e 20 mixed.

Nos treinos, o tempo de exposição de cada imagem é de 8.0s, com descanso de 0.5s entre cada exposição. Nos testes, o tempo de exposição de cada imagem é de 2.5s, com descanso de 0.5s entre cada exposição.

1.4.2 Flanker: descrição e objetivos

Como o Dots, o jogo Flanker é também destinado a avaliar a flexibilidade cognitiva da criança. O jogo é composto de três testes, sendo que os dois primeiros testes possuem uma fase de treinamento. No primeiro teste, é exibida uma figura onde aparecem cinco peixinhos azuis como se vê abaixo:



Figura 1.2: peixinhos azuis

A criança deve observar o peixe do centro e apertar para o lado em que ele olha; na figura acima, ela aperta o botão esquerdo, é assim que ela é instruída pelo aplicador a alimentar o peixinho.

No segundo teste, é exibida uma figura com cinco peixinhos cor-de-rosa como na figura abaixo:



Figura 1.3: peixes cor-de-rosa

A criança deve observar os peixes ao redor do peixe do centro, que nesse caso são os peixes famintos, e deve alimentá-los apertando para o lado que eles estão olhando.

Por fim, o terceiro teste, mixed, mistura as regras; podem aparecer 5 peixes azuis ou 5 peixes cor-de-rosa. Quando os peixes aparecerem, a criança deve agir de acordo com as regras descritas anteriormente.

A bateria de testes é composta de 76 trials na seguinte ordem: 8 treinos seguidos de 20 trials do peixe azul, 8 treinos seguidos de 20 trials do peixe rosa, 20 trials mixed. Como no jogo anterior, o tempo de exposição de cada imagem nos treinos é de 8.0s, com descanso de 0.5s entre cada exposição, e nos testes o tempo de exposição de cada imagem é de 2.5s, com descanso de 0.5s entre cada exposição.

1.4.3 Ursos



Figura 1.4: jogo ursos

O jogo foi proposto pela psicóloga Liege Felício para medir a capacidade de inibição de impulso da criança. Este jogo foi inspirado pelo chamado “teste do marshmallow” de Stanford [4].

Ele apresenta dois ursos, um ao lado esquerdo e outro ao lado direito da tela, de forma que a criança consiga distinguir qual deles está mais longe. Cada urso carrega uma certa quantidade de balões, e a criança é instruída a escolher um dos ursinhos: o urso que está à direita carrega apenas um balão; se a criança o escolher, ela espera 4 segundos até o urso chegar a ela, isto é, quando ele solta o balão para voar. Quando este evento acontecer, ela deve ganhar um doce do aplicativo do teste. Se ela optar pelo urso da esquerda, ela deve esperar por 12 segundos até o urso chegar perto dela e soltar os três balões; então ela deve ganhar três doces do aplicativo do teste como recompensa.

Este teste armazena a escolha da criança (urso distante, urso próximo) e o tempo que ela leva para a escolha.

1.4.4 Jogo musical

A fisioterapeuta Graziela Polido trouxe ao professor Carlos a ideia de desenvolver um programa que pudesse ser usado por uma criança portadora de uma doença degenerativa chamada amiotrofia muscular espinhal, que causa perda da massa muscular

Pensamos em criar um jogo musical adequado: um jogo que não exigisse agilidade motora do jogador. A ideia então foi desenvolver um jogo em que o jogador, Arthur, compusesse suas músicas: por meio da interface do

programa, ele poderia explorar as possibilidades de ação; sons distintos, identificados com figuras. Cada som que ele escolhe executar fica indicado em uma lista, e quando ele escolhe a ação de tocar (o botão de play), os sons referenciados na lista são tocados.

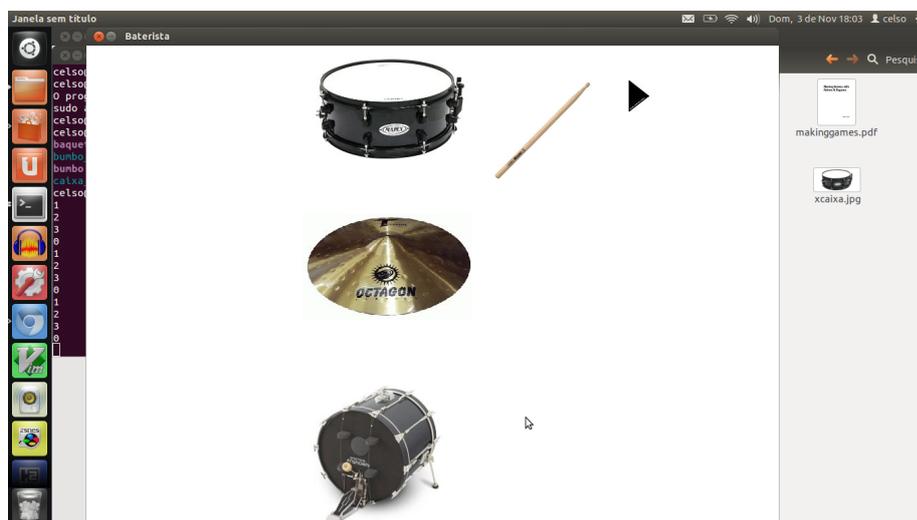


Figura 1.5: jogo baterista

No caso trazido por Graziela, o paciente tem 7 anos de idade e apresenta movimentação nos pés, mas não nas mãos. Pensamos então em formas de permitir que a criança com essa severa restrição de movimentos consiga usar o jogo.

Soubemos que o Arthur é capaz de movimentar os pés esquerdo e direito e apertar um botão semelhante a um botão de mouse. Pensamos em usar então dois botões para que ele aprendesse o jogo, mas havia ainda uma restrição: aparentemente o pé direito é mais debilitado que o esquerdo. O professor Carlos sugeriu usarmos então uma scanning interface, que basicamente é um selecionador que navega sobre os alvos (o prato de bateria ou o botão de play) em um tempo constante, de forma que um conjunto grande de ações possa ser oferecido contando-se apenas com um botão e uma certa dose de paciência do usuário. O diagrama abaixo ilustra o funcionamento de uma scanning interface com N itens:

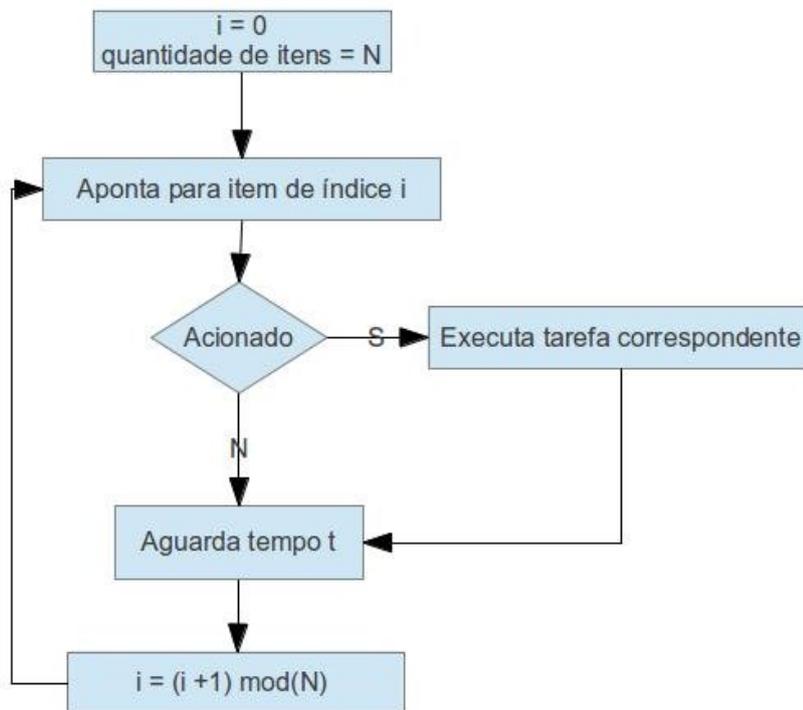


Figura 1.6: interface sequencial

Ilustrando um problema prático: no início tentei usar objetos `mixer.Sound` do `pygame` individualmente para tocar cada som da fila de sons, algo parecido com

```
somBumbo = pygame.mixer.Sound('bumbo_01.wav')
somCaixa = pygame.mixer.Sound('caixa_01.wav')

somBumbo.play()
fpsClock.tick(TEMPO_SEMINIMA)
somCaixa.play()
fpsClock.tick(TEMPO_SEMINIMA)
somBumbo.play()
fpsClock.tick(TEMPO_SEMINIMA)
```

o que se revelou muito custoso de alguma forma, pois a reprodução da composição apresentava muitos lags. Pensei então se seria possível gerar um

wav antes de tocar cada música composta, e isso se mostrou fácil de fazer usando o `scipy`:

```
smplr_t_Bumbo, bumbArray = scipy.io.wavfile.read('bumbo_01.wav')
```

Na linha de código acima aconteceu o seguinte: o método `read` de `wavfile` retorna um número, o `sample rate` do wav lido em `samples/segundo`, e um `numpy array` contendo os dados lidos do wav. Dessa forma, digamos que o usuário tenha tocado o bumbo duas vezes e a caixa uma vez e em seguida mande tocar a composição apertando o `play`; tudo o que o programa tem que fazer é concatenar dois arrays de som de bumbo e um de som de caixa, transformar o resultado em um arquivo wav e tocá-lo. Em código (`composicArray` é o array que armazena os sons já tocados):

```
composicArray = numpy.concatenate([composicArray, bumbArray])
composicArray = numpy.concatenate([composicArray, bumbArray])
composicArray = numpy.concatenate([composicArray, caixArray])
write('musica.wav', 44100, composicArray)
somCompos = pygame.mixer.Sound('musica.wav')
somCompos.play()
```

1.4.5 Sistema de autoria de aplicativos com interface sequencial

Um sistema de autoria de aplicativos é um sistema destinado a possibilitar o desenvolvimento de aplicativos multimídia interativos como o jogo descrito anteriormente. Uma característica dos sistemas de autoria é permitir que o usuário crie aplicativos sem necessidade de conhecimentos de programação.

Este sistema está em fase inicial de desenvolvimento.

1.5 Conclusões

Neste ano, trabalhando em conjunto com orientador e outros pesquisadores, tive uma ideia de como é desenvolver software com o peso da responsabilidade pelos, digamos, clientes da aplicação. Eu me via diversas vezes checando e recheando os tempos de respostas que o jogo estava gerando (Dots e Flanker).

Tive o conhecimento de ferramentas novas, aprendi um pouco delas e acho que isso é sempre bom; PsychoPy, pygame, jython. Escutei bastante durante as reuniões, ideias a respeito dos experimentos, sugestões, dei as sugestões que pude, o que de certa forma foi uma visão prática de projeto de software a que fui exposto.

Com o problema da construção de jogos para o paciente de AME fiquei satisfeito por poder participar um pouco da parte criativa, já que se trata de uma produção original, diferentemente da replicação de jogos que fiz anteriormente que teve que seguir protocolos de experimento que não permitiam, por exemplo, adicionar diversos elementos gráficos, ou áudio de resposta ou música de fundo.

Capítulo 2

Parte subjetiva

2.1 Desafios e frustrações

Quando o Hitoshi trouxe a primeira proposta, a de criar um jogo para testar crianças junto ao IPq, fiquei ao mesmo tempo feliz e atormentado; feliz porque gosto de jogos eletrônicos (o que não é muito raro entre os alunos do BCC), e atormentado pela responsabilidade de programar um aplicativo que, embora simples, não poderia falhar ao ser usado para uma tarefa tão importante. Não posso evitar de me lembrar de uma frase do professor Siang, quando eu cursava MAC 412, Organização de computadores: “não precisa se preocupar, o máximo que pode acontecer é o programa travar, não é como um médico que não dá pra reiniciar o paciente(...)”, alguma coisa assim. É, gostaria de ter me lembrado disso mais cedo.

Continuando, fiquei feliz em participar de um projeto que considero bonito por ter a iniciativa de interferir, no bom sentido, na educação de algumas pessoas.

2.2 Lista das disciplinas mais relevantes para o trabalho

MAC0110 - Introdução à computação, MAC0122 - Princípios de desenvolvimento de algoritmos, MAC0323 - Estruturas de dados, acredito que foram matérias essenciais como a base para todo o curso;

MAC0211 - Laboratório de programação I, MAC0242 - Laboratório de programação II, fortaleceram, digamos, a alegria de programar.

2.3 Agradecimentos

Agradeço à minha esposa Talitha pelo amor, pela paciência, pelo apoio, por desenhar ursinhos para o meu jogo, por me incentivar nos momentos certos.

Agradeço a meus pais, pelo amor, pela paciência e por tudo mais.

Agradeço a meus novos amigos do BCC, Suzana Siqueira Santos e Samuel Praça de Paula, que me apoiaram e me acompanharam nesses últimos dois anos.

Agradeço a Liege Felício e Graziela Polido por trazer desafios relevantes, bonitos, além de uma porção de ideias.

Agradeço a todos os professores do BCC, e em especial, o professor Hitoshi, que me ajudou com ideias, sugestões, paciência, correções e bons propósitos para o trabalho.

Referências Bibliográficas

- [1] <http://www.psychopy.org/>
- [2] <http://www.pygame.org/wiki/about>
- [3] Peirce JW (2009) “Generating stimuli for neuroscience using PsychoPy”. *Front. Neuroinform.* 2:10. doi:10.3389/neuro.11.010.2008
- [4] Mischel, Walter; Ebbesen, Ebbe B. “Attention in delay of gratification”. *Journal of Personality and Social Psychology*, Vol 16(2), Oct 1970, 329-337. doi: 10.1037/h0029815
- [5] Davidson, Matthew C.; Amso, Dima; Anderson, Loren Cruess; Diamond, Adele “Development of cognitive control and executive functions from 4 to 13 years: Evidence from manipulations of memory, inhibition, and task switching” *Neuropsychologia*. 2006 ; 44(11): 2037–2078
- [6] Sweigart, Al; “Making games with Python and Pygame”
- [7] Oetiker, Tobias; “The not so short introduction to L^AT_EX2e”