

Universidade de São Paulo

Instituto de Matemática e Estatística

Trabalho de conclusão de curso

Celso Masahiro Shimabukuro

Nº USP: 3314242

Três abordagens do uso de jogos computacionais: análise de capacidades executivas em crianças de 4 a 7 anos de idade, avaliação de aspectos cognitivos e posicionamento funcional em crianças com Atrofia Muscular Espinhal tipo I e sistema de autoria de aplicativos com interface sequencial

São Paulo
2014

SUMÁRIO

1. PARTE OBJETIVA.....	3
1.1. Introdução.....	3
1.2. Objetivos gerais.....	3
1.3. Objetivos específicos.....	3
1.4. Conceitos e tecnologias estudadas.....	4
1.4.1. PsychoPy.....	4
1.4.2. pygame.....	5
1.5. Jogos para avaliar a capacidade executiva em crianças de 4 a 7 anos...5	
1.5.1. Dots.....	6
1.5.2. Flanker.....	7
1.5.3. Ursos.....	9
1.6. Jogo para avaliar aspectos cognitivos e posicionamento funcional em crianças com Atrofia Muscular Espinhal tipo I.....	10
1.6.1. Detalhes do desenvolvimento do jogo.....	12
1.7. Sistema de autoria de aplicativos com interface sequencial.....	13
1.8. Conclusões.....	14
2. PARTE SUBJETIVA.....	16
2.1 Desafios e frustrações.....	16
2.2 Lista das disciplinas mais relevantes para o trabalho.....	16
2.3 Agradecimentos.....	17
REFERÊNCIAS.....	18

1. PARTE OBJETIVA

1.1. Introdução

O presente trabalho foi desenvolvido ao longo do ano de 2013 e contou com a participação do professor Carlos Hitoshi Morimoto como orientador. Além disso, o trabalho foi realizado em parceria com a psicóloga Liege Felício, do Instituto de Psiquiatria da Universidade de São Paulo (IPq/USP) e da fisioterapeuta Graziela Polido, mestranda em Ciências da Reabilitação pela USP.

1.2. Objetivos gerais

Es trabalho visou enfocar o uso de jogos computacionais como instrumentos de análise e avaliação de aspectos psico-cognitivos em crianças, contribuindo com o trabalho de profissionais de áreas como a psiquiatria e a fisioterapia. A intenção é agregar o aspecto lúdico em formato digital, tão prevalente em nossa realidade atual, aos instrumentos de avaliação das áreas mencionadas, tornando o procedimento de avaliação mais interessante e atrativo ao público a que se destina.

1.3. Objetivos específicos

Para se atingir o estabelecido acima, foram delineados três objetivos específicos, atuando-se, pois, com três abordagens distintas a partir do objetivo geral, a saber:

- a) desenvolver jogos de computador para avaliar a capacidade executiva em crianças participantes do projeto de intervenção de capacidades executivas de Liege Felício do IPq/USP;
- b) desenvolver um jogo que adequado para ser utilizado por uma criança portadora de Amiotrofia Muscular Espinhal (Atrofia Muscular Espinhal tipo I), tendo como inspiração para o paciente de Graziela Polido, Arthur Messias;
- c) desenvolver um sistema de autoria de aplicativos com interface sequencial, inspirado na criação do jogo mencionado no item b acima, para facilitar a criação de novos aplicativos com a mesma estrutura do jogo criado para o Arthur.

1.4. Conceitos e tecnologias estudadas

A linguagem escolhida para desenvolver os jogos de avaliação de capacidades executivas em crianças – projeto associado ao IPq/USP – foi python, e com ela o PsychoPy, um software próprio para criação de testes neurológicos e neuropsicológicos.

Para o jogo musical, destinado à avaliação de aspectos cognitivos e posicionamento funcional em crianças com Atrofia Muscular Espinhal tipo I, escolhemos usar python e pygame. Além disso, Scipy foi utilizada para resolver um problema específico descrito adiante.

Para o sistema de autoria de aplicativos com interface sequencial, escolhemos inicialmente jython pela conveniência de podermos nos beneficiar com o uso da biblioteca Swing e da biblioteca pygame ao mesmo tempo; porém, vimos que não era possível usar pygame em jython.

1.4.1. 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]

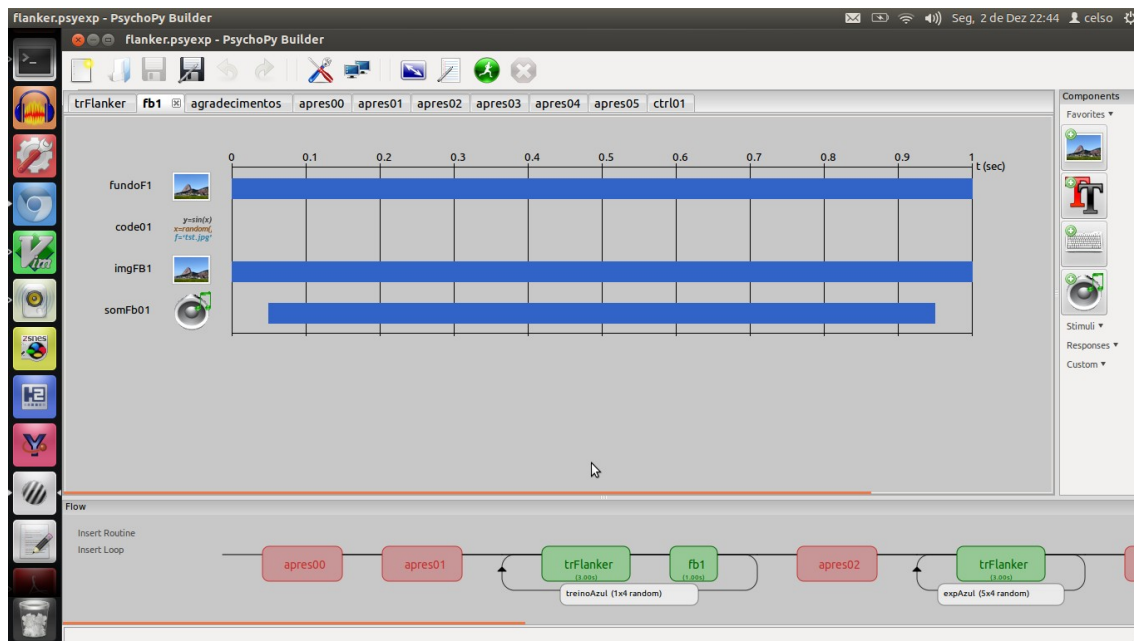


Figura 1.1: PsychoPy modo Builder

1.4.2. 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]

1.5. Jogos para avaliar a capacidade executiva em crianças de 4 a 7 anos

Esses jogos foram desenvolvidos em parceria com uma equipe do IPq, a qual teve como representante a psicóloga Liege Felício.

Inicialmente, a equipe trouxe a necessidade do desenvolvimento de quatro jogos destinados a trabalhar com as chamadas capacidades executivas do grupo participante do projeto de intervenção de capacidades executivas, do IPq/USP, grupo este composto por crianças de 4 a 7 anos de idade. As capacidades executivas a serem trabalhadas consistem em: memória, inibição (capacidade de se concentrar em uma tarefa e ignorar demais estímulos) e flexibilidade cognitiva (capacidade de se adaptar à mudança de regras).

Ao longo das reuniões feitas em conjunto com o professor orientador Carlos Hitoshi e representantes do grupo de estudos do IPq, decidiu-se reduzir o

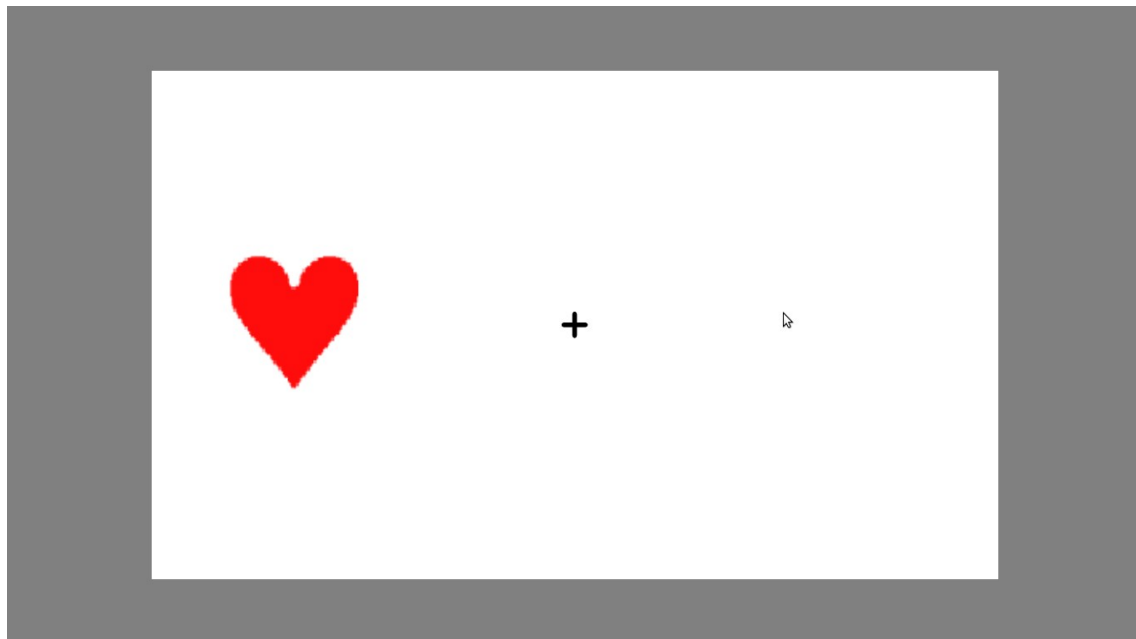
projeto a três jogos, os quais se inserem em um conjunto de testes destinados a verificar a evolução das capacidades cognitivas das crianças participantes.

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

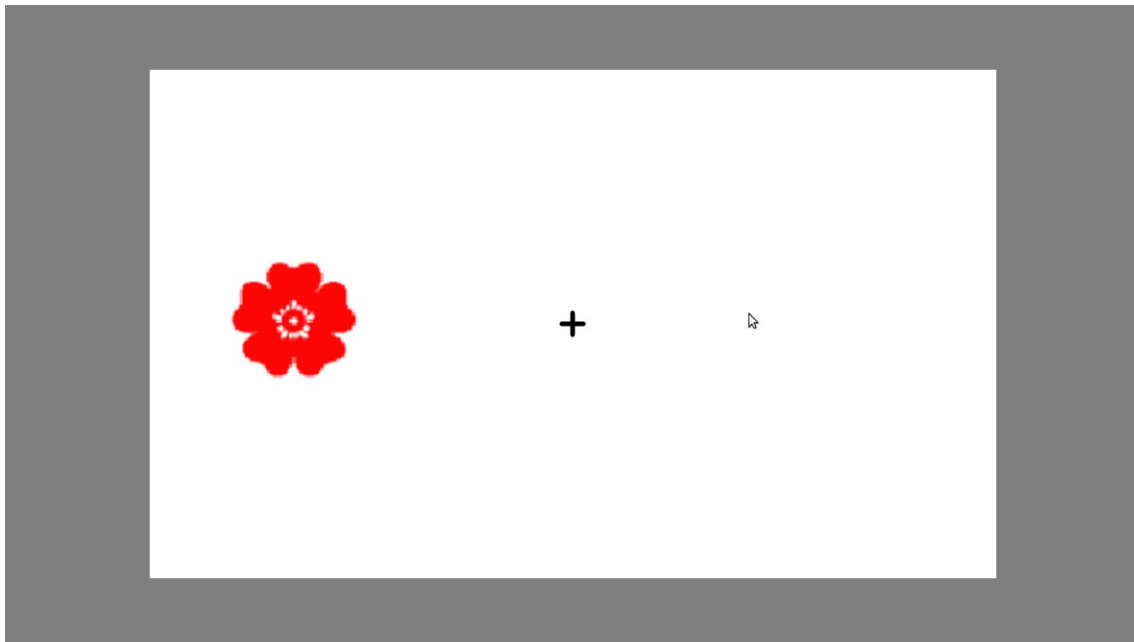
1.5.1. Dots

O jogo é destinado a avaliar a flexibilidade cognitiva da criança. Ele é composto de três testes, sendo que os dois primeiros testes possuem uma fase de treinamento, durante a qual 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 a figura de um coração do lado direito ou do lado esquerdo da tela e a criança deve apertar o botão do 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 o botão do lado oposto ao da figura, isto é, se a flor aparece do lado esquerdo, a criança deve apertar o botão da direita, e se a flor aparecer do lado direito, deve apertar o botão da 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, ou seja, deve clicar no mesmo lado em que aparece o coração ou no lado oposto ao que aparece a flor.

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

Nos treinos, o tempo de exposição de cada imagem é de 8,0 s, com descanso de 0,5 s entre cada exposição. Nos testes, o tempo de exposição de cada imagem é de 2,5 s, com descanso de 0,5 s entre cada exposição.

Durante os testes, são armazenados dados referentes a tempo de reação e ocorrência de acerto ou erro armazenados em um diretório da t a , permitindo à equipe do IPq acessar esses dados e analisá-los conforme sua necessidade.

1.5.2. Flanker

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 em que 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 cinco peixes azuis ou cinco 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 testes do peixe azul, 8 treinos seguidos de 20 testes do peixe rosa e 20 testes *mixed*. Como no jogo anterior, o tempo de exposição de cada imagem nos treinos é de 8,0 s, com descanso de 0,5 s entre cada exposição, e

nos testes, o tempo de exposição de cada imagem é de 2,5 s, com descanso de 0,5 s entre cada exposição.

Durante os testes, são armazenados dados referentes a tempo de reação e ocorrência de acerto ou erro armazenados em um diretório da t a , permitindo à equipe do IPq acessar esses dados e analisá-los conforme sua necessidade.

1.5.3. Ursos

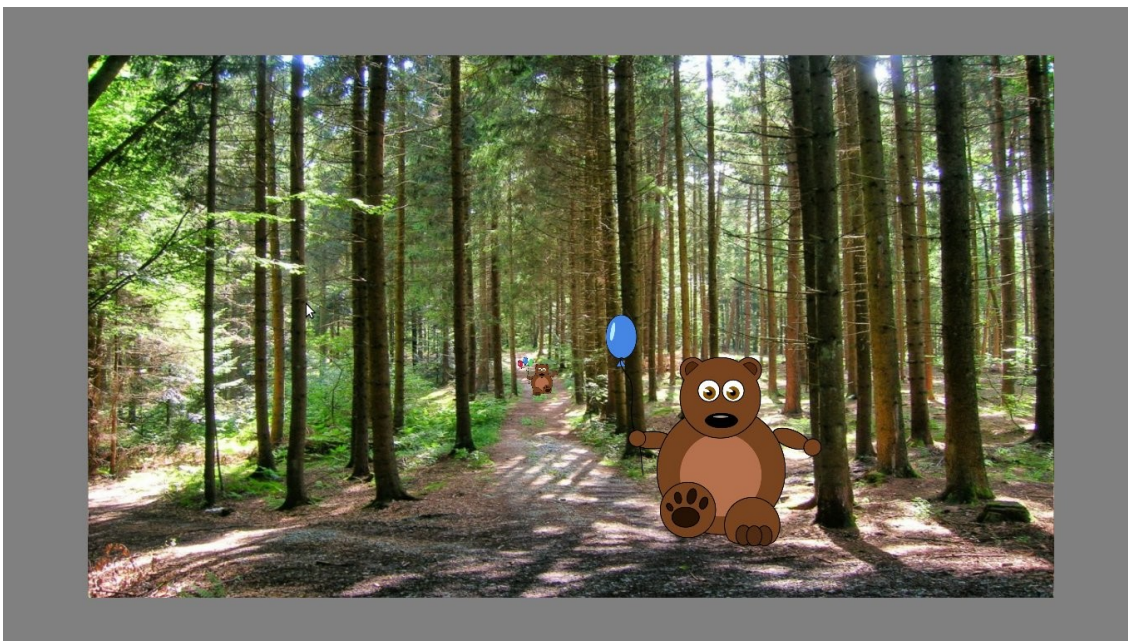


Figura 1.4: jogo ursos

O jogo foi proposto pela psicóloga Liege Felicio 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 quatro segundos até o urso chegar a ela e soltar o balão. Quando este evento acontece, ela deve ganhar um doce do aplicador 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 que ele segura; então ela deve ganhar 3 doces do aplicador do teste como recompensa.

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

1.6. Jogo para avaliar aspectos cognitivos e posicionamento funcional em crianças com Atrofia Muscular Espinhal tipo I

A fisioterapeuta Graziela Polido trouxe ao professor Carlos a ideia de desenvolver um programa que pudesse ser usado por uma criança com uma doença degenerativa chamada Amiotrofia Muscular Espinhal (AME), que causa perda da massa muscular, limitando severamente a mobilidade do paciente.

A Atrofia Muscular Espinhal, ou Amiotrofia Espinhal, ou Amiotrofia Espinhal Infantil Progressiva, nomenclaturas que podem ser encontradas na literatura é uma doença neuromuscular de origem genética. O pai e a mãe carregam o gene e têm a chance de 50% de terem uma gravidez em que a criança porta o gene e 25% de terem uma gravidez em que a criança apresenta a doença.

A AME é uma doença progressiva, com perda de aquisições motoras. O tipo I é diagnosticado ao nascer ou até os 6 meses de vida e normalmente evolui para insuficiência respiratória e as crianças fazem uso de traqueostomia com suporte ventilatório contínuo. O tipo II é diagnosticado por volta dos 2 anos de vida, sendo que algumas crianças chegam a adquirir marcha. Existem ainda outros subtipos.

Este projeto em particular foi destinado a atender as características específicas do Arthur, um dos pacientes da Graziela diagnosticado com AME tipo I. Pensamos em criar um jogo musical adequado a ele e a outros portadores da doença, ou seja, 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 e 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, formando uma sequência musical.

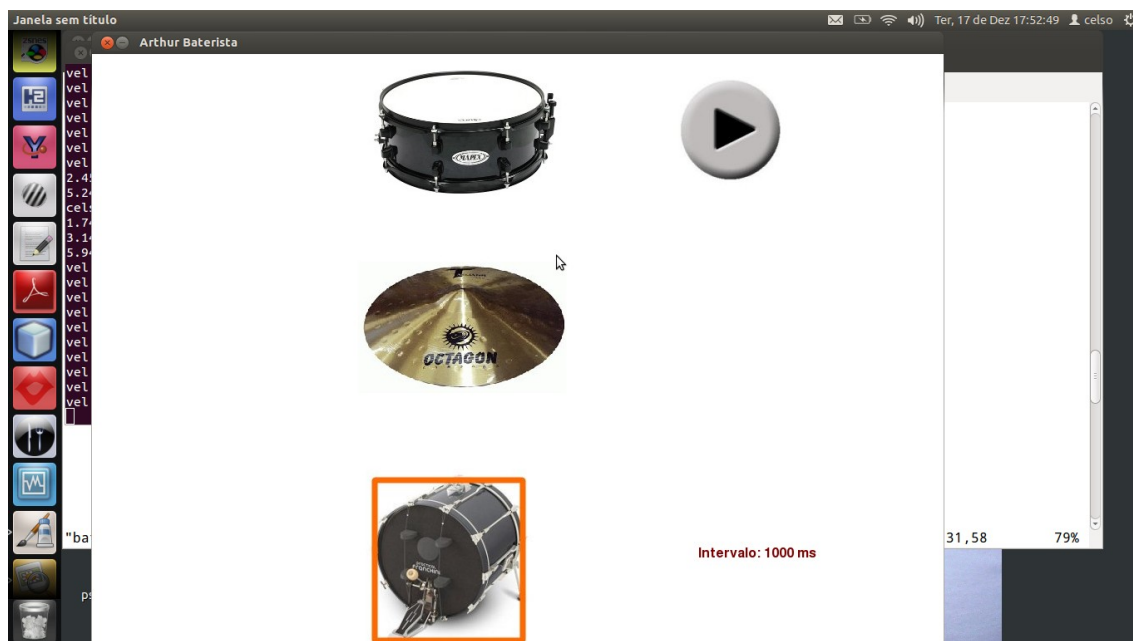


Figura 1.5: jogo baterista

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 para permitir que a criança com essa severa restrição de movimentos conseguisse 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 (os componentes de bateria e 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:

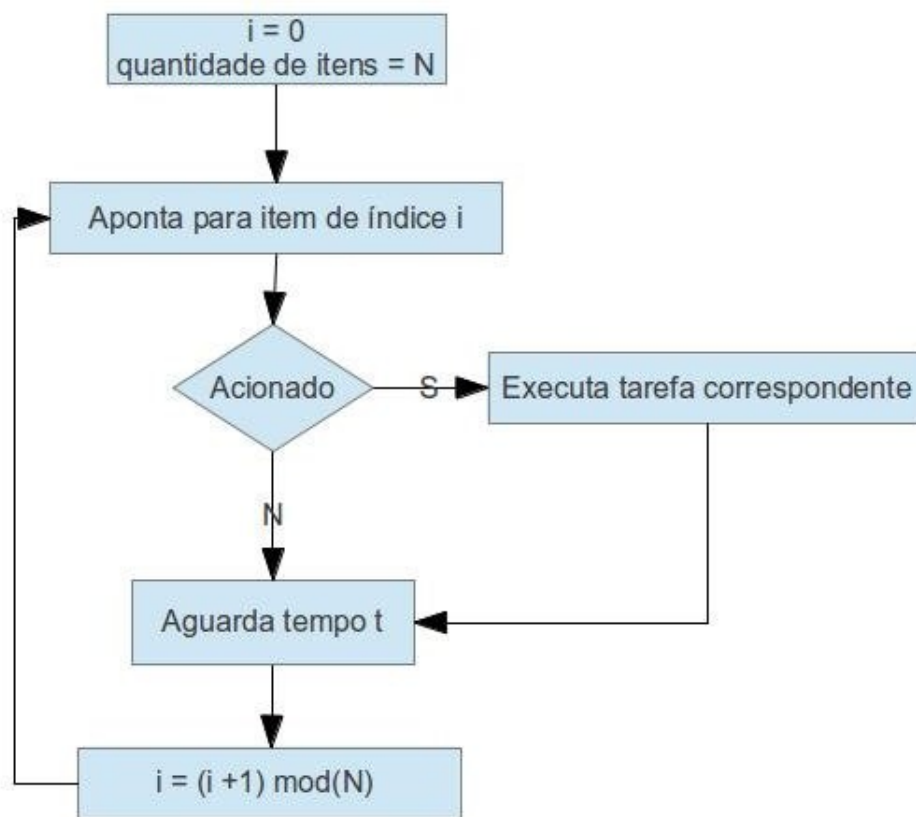


Figura 1.6: interface sequencial

1.6.1. Detalhes do desenvolvimento do jogo

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 *wav_le* 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):

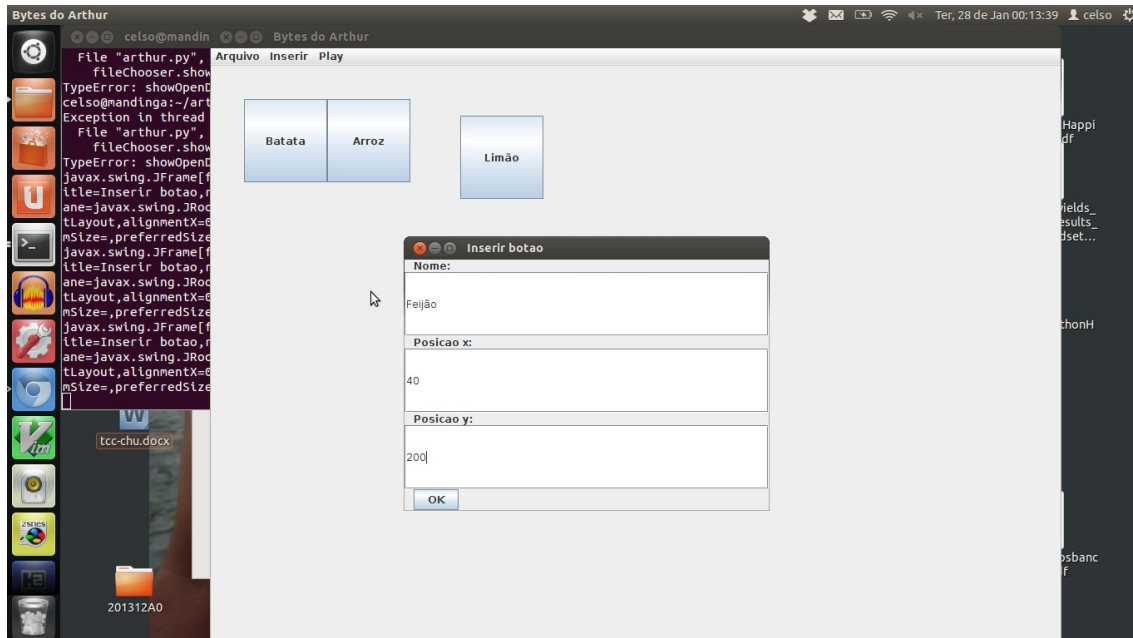
```
composicArray = numpy.concatenate([composicArray, bumbArray])
composicArray = numpy.concatenate([composicArray, bumbArray])
composicArray = numpy.concatenate([composicArray, caixArray])
write ('musica.wav', 44100, composicArray)
somCompos = pygame.mixer.Sound('musica.wav')
somCompos.play()
```

1.7. 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.

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.

O sistema possui interface gráfica simplificada e permitirá ao usuário acrescentar elementos visuais, áudios e definir seu posicionamento na tela, além de outros aspectos como controle de FPS.



ByArthur protótipo

Este sistema foi programado inicialmente em jython, que permite importar classes java em um aplicativo escrevendo com a sintaxe do python.

A partir desse sistema, será possível criar, por exemplo, um jogo com as características do “baterista” exibido anteriormente.

Desenvolvi o programa seguindo os tutoriais de java da Oracle, aprendendo Swing por meio de exemplos. Escrevendo o programa, tive algumas dificuldades: uma delas foi com a forma como o java.swing atualiza seus componentes gráficos;

```
framePrincipal.invalidate()  
framePrincipal.repaint()
```

eu usei o código acima para atualizar o framePrincipal do jogo ao adicionar botões, mas percebi que o frame só era atualizado quando alguma ação acontecia sobre ele, como a passagem ou um clique do mouse (o que obviamente não é aceitável).

Procurei então mais alguma fonte de informação que me ajudasse, e encontrei um livro sobre programação de jogos em Java, o *Killer Java Game Programming^[8] in Java*, de Comecei a reconstruir, em Java e usando códigos apresentados neste livro, as classes que representam uma tela, sendo a classe main chamada ByArthur:

```
public class ByArthur extends JFrame implements WindowListener
```

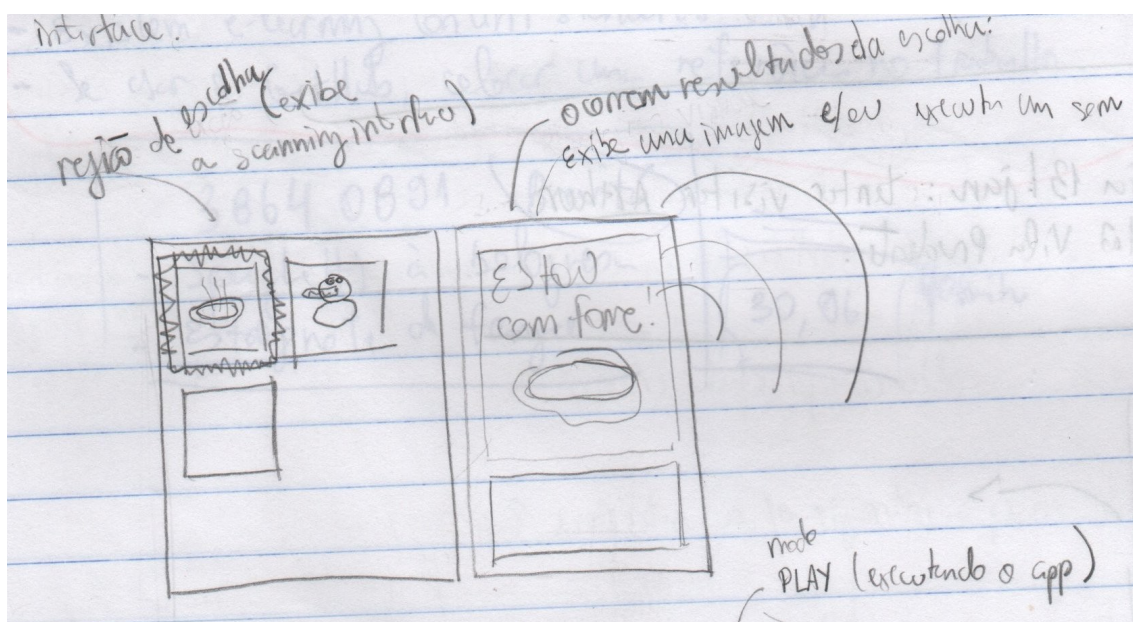
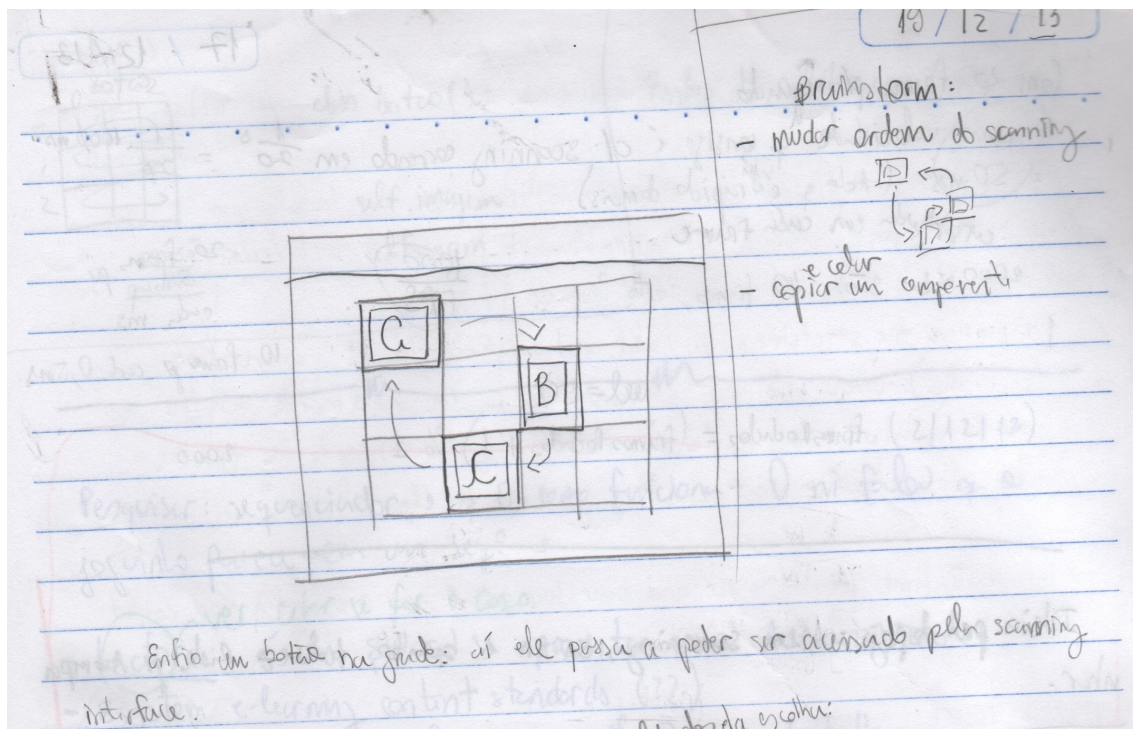
O livro trata de forma detalhada e eficiente a forma de como ter framerate (FPS) confiável, e como relacionar FPS com UPS (update rate, Updates per Second). Porém, revendo o tempo disponível para desenvolver o projeto, acho que eu não deveria ter me preocupado com isso, porque os jogos criados precisam de uma taxa de atualização muito mais baixa do que as técnicas apresentadas no livro ofereciam. Para se ter uma ideia, o jogo Baterista pode ser configurado para ter a velocidade de scanning de um passo por 8000ms no mínimo (um FPS de 1/8 frames por segundo) e 500ms no máximo (FPS == 2 frames por segundo). A técnica descrita no livro procura obter de forma segura máximos de 85 a 90 frames per second.

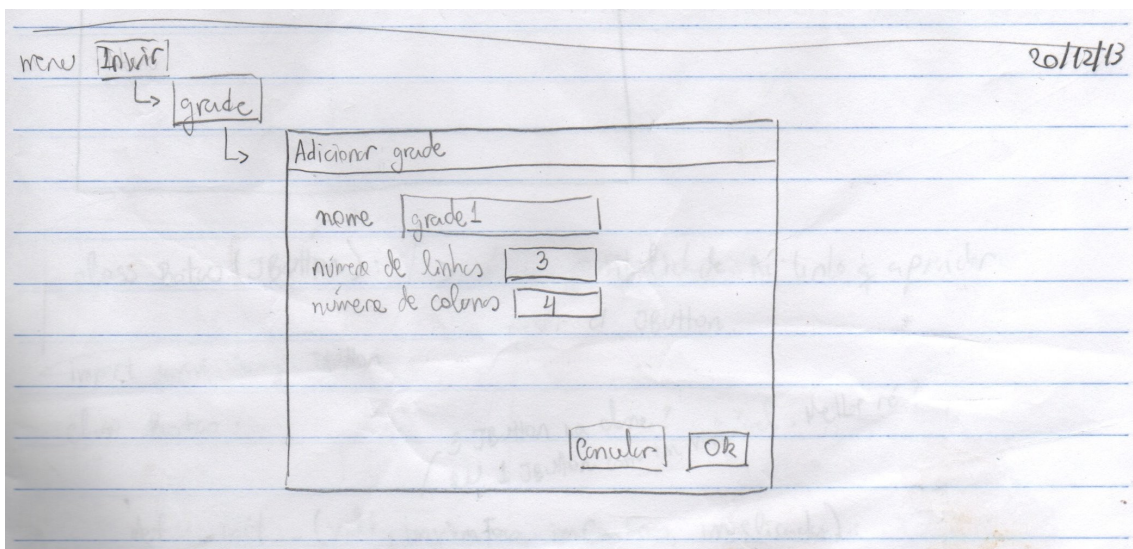
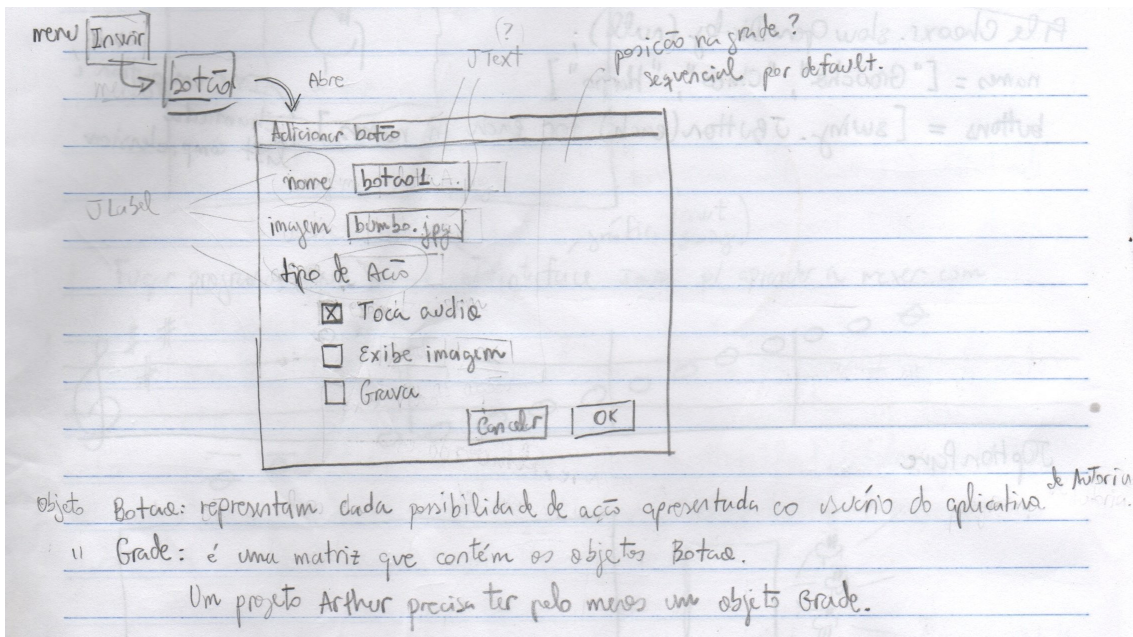
Outra dificuldade que tive foi, também, descobrir que não poderia combinar, pelo menos de forma simples como imaginei, jython e pygame: o pygame não é compatível até o presente momento com o jython, de forma que tive que escolher entre

- continuar a usar jython e java e abandonar as funcionalidades para jogos do pygame;
- voltar atrás e fazer o sistema em python (cpython), fazendo a interface com o usuário sem aproveitar as vantagens do java.awt, java.swing e tudo mais.

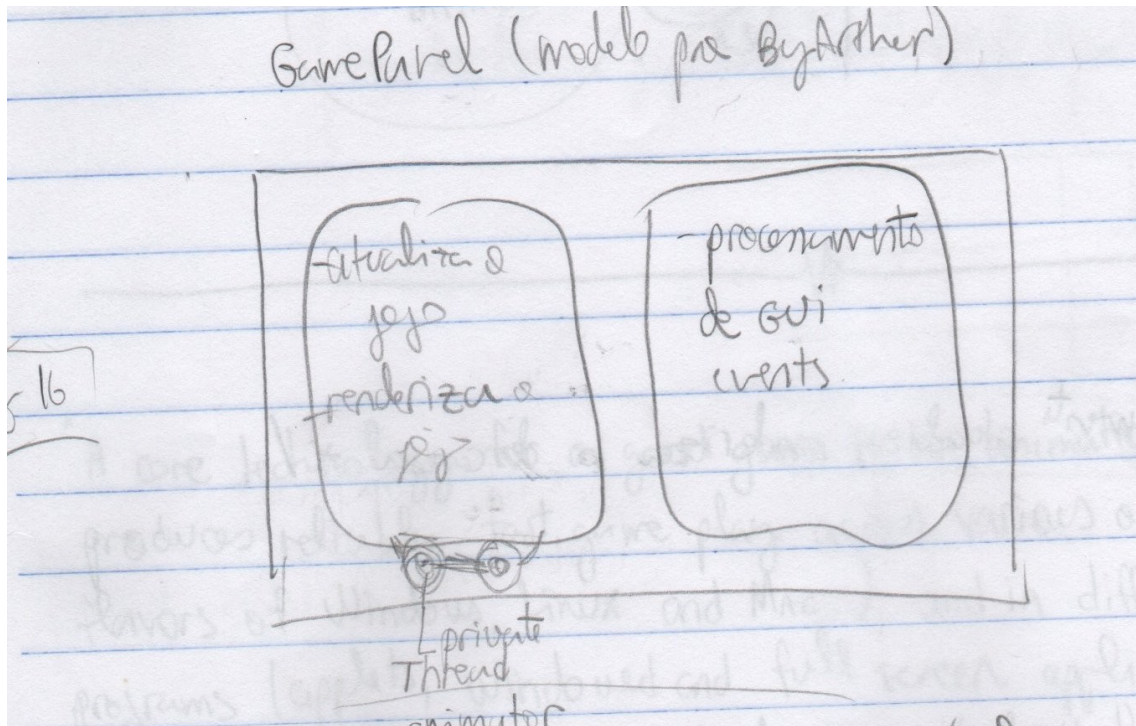
Escolhi refazer o sistema em Java, até porque meus estudos até então estavam concentrados em aprender Swing e AWT (Abstract Window Toolkit).

Abaixo, seguem algumas ideias que tive para o sistema:





Abaixo, algumas das coisas que estudei no livro *Killer Java Game Programming*:



animador

17) Uso de volatile ^{proibido que a variável seja copiada a memória local do thread.}

JMM permite q cada thread tenha sua própria memória local. Variáveis q precisam ser vistas por mais de 1 thread são candidatas a serem definidas como volatile, p/ evitar inconsistência entre diferentes versões da mesma variável em threads diferentes.

why sleep?

Thread.sleep(20); // thread deixa de executar por 20ms.

(// coleta de lixo (JVM)
der tempo p o repaint() em seguida
redes as chances de event coalescence (??)

double buffering é a técnica de aplicar as complicadas operações de desenho em uma imagem secundária (no caso um Graphics) e não diretamente na tela.

A grande vantagem de usar double buffering é reduzir as piscadas (flicker) da tela.

public void paintComponent(Graphics g) {
 super.paintComponent(g); // pode ser chamada pela JVM
 // independente da thread de animação
 if (dbImage != null) {
 g.drawImage(dbImage, 0, 0, null);
 }
}

Essa linha deve ser simples. NÃO ENVIAR o componente do jogo aqui.

1.8. Conclusões

Neste ano, trabalhando em conjunto com o orientador e outros pesquisadores, foi possível perceber como é desenvolver um software com o peso da responsabilidade perante os clientes da aplicação. Eu me via diversas vezes checando e recheando os tempos de respostas que o jogo estava gerando (especialmente no caso do Dots e do Flanker).

Tive o conhecimento de ferramentas novas (PsychoPy, pygame, jython), aprendi um pouco delas e acho que isso é sempre bom. Durante as reuniões semanais, escutei bastantes ideias e sugestões a respeito dos experimentos e 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, para o projeto do IPq, que teve que seguir protocolos de experimento que não permitiam, por exemplo, adicionar diversos elementos gráficos, ou áudio de resposta ou música de fundo.

Boa parte dos objetivos a que nos propusemos neste trabalho foram alcançados satisfatoriamente e, embora não tenha sido possível receber o feedback da aplicação dos jogos criados.

Este trabalho demonstrou , ainda, a importância da interação entre as diversas áreas do conhecimento humano (neste caso, a computação com a psiquiatria e a fisioterapia), trabalhando em conjunto para es atingirem os mais diversos objetivos, complementando-se e enriquecendo-se mutuamente.

2. PARTE SUBJETIVA

2.1 Desafios e frustrações

Quando o professor Hitoshi trouxe a primeira proposta, a de criar um jogo para testar crianças junto ao IPq, fiquei bastante feliz porque gosto de jogos eletrônicos (o que não é muito raro entre os alunos do BCC), e me senti desafiado pela responsabilidade de programar um aplicativo que, embora relativamente 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.

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. Fiquei frustrado por não ter desenvolvido a contento o sistema de autoria que visualizamos.

2.2 Lista das disciplinas mais relevantes para o trabalho

MAC0110 Introdução à computação: o curso de introdução à computação me marcou bastante, trazendo desafios em forma de EP's (exercícios programa) e aulas inesquecíveis, como a aula na qual “implementamos” o computador a papel (com a professora Nami Kobayashi), forma lúdica de aprendermos o funcionamento de um computador.

MAC0122 Princípios de desenvolvimento de algoritmos: treinando, na prática, a observar os comportamentos de diferentes algoritmos, vimos como a diferença entre os modelos óbvios (como um Bubble Sort) e um modelo elegante (como um MergeSort) leva a resultados impressionantes.

MAC 0211 Laboratório de programação I e MAC 0242 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 Felicio 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

1. *Psychopy* [Internet]. Disponível em: <<http://www.psychopy.org/>>
2. *Pygame* [Internet]. Disponível em: <<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 W, Ebbesen EB (1970). Attention in delay of gratification. *Journal of Personality and Social Psychology*, 16(2):329-337. doi: 10.1037/h0029815
5. Davidson MC, Amso D, Anderson LC, Diamond A (2006). Development of cognitive control and executive functions from 4 to 13 years: Evidence from manipulations of memory, inhibition, and task switching. *Neuropsychologia*. 44(11): 2037-2078.
6. Sweigart A. *Making games with Python and Pygame*
7. Oetiker T. *The not so short introduction to LATEX2e*
8. Davison A (2005). *Killer Java Game Programming*. CA: O'Reilly