UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

ESCOLA DE INFORMÁTICA APLICADA

UMA ABORDAGEM BASEADA EM PROJETOS PARA O ENSINO DE PROGRAMAÇÃO: INVESTIGAÇÕES SOBRE A TECNOLOGIA ANDROID

Cecília de Almeida Soares

Pedro Lamy Zaluar

Orientador

Geiza Maria Hamazaki da Silva

RIO DE JANEIRO, RJ – BRASIL

DEZEMBRO DE 2015

UMA ABORDAGEM BASEADA EM PROJETOS PARA O ENSINO DE PROGRAMAÇÃO: INVESTIGAÇÕES SOBRE A TECNOLOGIA ANDROID

Cecília de Almeida Soares
Pedro Lamy Zaluar

Projeto de Graduação apresentado à Escola de Informática Aplicada da Universidade Federal do Estado do Rio de Janeiro (UNIRIO) para obtenção do título de Bacharel em Sistemas de Informação.

Aprovada por:

_____

Profª. Drª. Geiza Maria Hamazaki da Silva (UNIRIO)

_____

Prof. Me. Pedro Nuno de Souza Moura (UNIRIO)

_____

Prof. Dr. Mariano Pimentel (UNIRIO)

RIO DE JANEIRO, RJ – BRASIL.
DEZEMBRO DE 2015

Agradecimentos

Agradeço primeiramente a Deus, por tudo que já me proporcionou nesta vida, aos meus amigos espirituais, pela dedicação e proteção.

Agradeço a minha família, aos meus pais que sempre fizeram de tudo por mim, agradeço muito a criação que tive. A minha irmã Natália, agradeço por tudo, tudo mesmo, de longe é uma das pessoas que mais tenho a agradecer. Ao meu irmão Mateus, obrigada pela convivência do dia a dia mesmo que às vezes seja difícil. A uma pessoa que não está mais comigo nesta vida, Daisy, mesmo que não possa ler isso, deixo aqui meus agradecimentos a você que teve grande participação na minha vida e na da minha família toda.

Agradeço aos meus amigos que já participam da minha vida há 11 anos, Anna Paula, Felipe Mateus e Taíse, obrigada por toda compreensão, apoio e carinho que sempre me deram até hoje.

Aos amigos que fiz na turma 2010.2 bem como de outras turmas que conheci ao longo do curso, muito obrigada pelos dias que pude conviver com vocês, em especial aos amigos Jefferson, João Felipe e Gabriel, todos de alguma forma contribuíram nesse processo. Aos amigos que possuo longe de mim (e que um dia espero estar mais perto), obrigada pelo apoio e por sempre estarem presentes de alguma forma.

A maravilhosa equipe de fisioterapia da Policlínica de Botafogo, meus agradecimentos pelo apoio, carinho e atenção que sempre tiveram comigo.

A professora Geiza, obrigada por nos auxiliar sempre que precisamos, tanto como tutora de turma e também agora sendo nossa orientadora, você foi fundamental para alcançarmos este resultado.

A banca examinadora, obrigada pela atenção que nos deram mesmo estando em período de recesso. Obrigada a todos os professores e servidores que participaram dos meus dias como aluna da UNIRIO.

Por fim, meu agradecimento a minha dupla Pedro, que me atura arduamente em trabalhos desde o início da faculdade. Obrigada por aceitar fazer mais este trabalho comigo, por dividir o peso e o estresse nessa conclusão de curso, por aguentar meu mau humor em dias ruins, mas principalmente obrigada por ser meu amigo nesses quase 6 anos.

A todos que participaram de tudo isso, muito obrigada!

Cecília de Almeida Soares

Meu agradecimento vai primeiramente para Deus, que todos os dias provê tantas graças à minha vida e a Ele só tenho a agradecer.

Quero agradecer imensamente meu irmão, que foi para mim, principalmente nessa jornada de formação acadêmica, o maior pilar que deu a base emocional e financeira, tornando assim este sonho possível. E também aos meus pais, pela excelente criação (que eu não trocaria por nenhuma outra se pudesse escolher), pelos importantes valores como humildade e senso ético, e, por fim, pelo amor e carinho que eu recebo dos dois, cada um de sua forma.

Queria agradecer também à professora Geiza, pela paciência e dedicação ao nos orientar. À minha dupla, Cecília, pelo companheirismo, trabalho em equipe e amizade, que vão muito além deste projeto. À banca examinadora, que nos atendeu com muita boa vontade mesmo em um período de recesso. E a todos os professores e servidores da UNIRIO que eu tive a honra de conviver e aprender competências que vão além do âmbito acadêmico. Parabéns pela busca de sempre estarem procurando melhorar, isso é admirável e muito animador.

Por fim (mas não menos importante), quero agradecer ao Jefferson, Túlio, João Felipe, Zoucas, Livânio, Gabriel e todos aqueles que participaram e estiveram comigo durante este percurso acadêmico, pelo companheirismo e amizade. E a todos os amigos e familiares, que estando perto ou longe de mim, sei que posso contar, e isso me dá muitas alegrias e muitas forças. Sem vocês nada disso seria possível. Obrigado!

Pedro Lamy Zaluar

RESUMO

As altas taxas de reprovação nas disciplinas podem, muitas vezes, ser devido a um desempenho ruim agravado pela falta de motivação do aluno. Isso ocasiona também na desistência do curso em questão. Na área da computação é uma situação que ocorre habitualmente no inicio do curso, onde a maioria dos ingressantes não sabe lidar com as dificuldades encontradas. A mudança na metodologia ou em como o conteúdo da ementa é apresentada tem mostrado, em experiências, ter tido um efeito positivo. Esse projeto propõe uma abordagem de ensino dos conceitos de programação através da utilização de aplicações Android. Com o desenvolvimento das aplicações apresentadas, espera-se que os alunos se sintam mais interessados e motivados a aprender o conteúdo da disciplina, melhorando assim o desempenho como um todo.

Palavras-chave: Programação, desenvolvimento Android, jogos no processo de ensino-aprendizagem.

ABSTRACT

High disapproval rates on subjects can, many times, be due to a bad performance aggravated by the lack of motivation from the student. This causes departure from the present course too. In computing area that's a situation that occurs usually on the beginning of the course, where majority of the entrants do not know how to deal with the difficulties found. Change on the methodology or in how the content of the curriculum is presented has shown, in experiences, to have been a positive effect. This project proposes an education approach of programming concepts through the use of Android applications. With the presented applications evolution, it's expected that the students will feel more interested and motivated to learn the subject content, making performance better as a whole.

Keywords: Programming, Android development, games in the teaching-learning process.

Índice

Índice de Figuras

1 Introdução

1.1 Motivação

As disciplinas que envolvem programação muitas vezes são as que os alunos têm, no inicio da faculdade, mais dificuldade em aprender. Em vista disso, podem muitas vezes se encontrar desmotivados a prosseguir com o curso. Essa situação foi presenciada pelos autores deste trabalho durante seu período de formação no Bacharelado em Sistemas de Informação. Segundo o autor de [1], o meio tradicional de ensino pode ser desmotivador, pois, entre outras razões, pode não ficar claro aos alunos a importância dos conteúdos apresentados, fazendo com que muitas vezes eles não se interessem por não considerarem que será útil no seu futuro profissional [2] [3]. Além disso, os autores desse trabalho também consideram que a postura que o professor assume em sala de aula, por muitas vezes a de agente ativo no ensino, que repassa a informação e espera que esta seja absorvida pelo aluno, também influencia consideravelmente neste processo de ensino-aprendizagem.

Pensando nisso, para este trabalho foi proposta uma abordagem diferente do que é comumente usado em muitos cursos da área de TI. A abordagem consiste em propor aos alunos que sejam desenvolvidas aplicações Android enquanto são apresentados a eles conceitos de programação orientada a objeto, assim conforme o conteúdo da ementa é transmitido eles iriam praticando e aplicando o conhecimento adquirido. Os autores acreditam que desta forma a aprendizagem possa se tornar mais atraente para os alunos iniciantes do curso da área de computação.

1.2 Objetivos

Este projeto tem como objetivo apresentar uma abordagem diferente para o ensino de programação, a qual consiste em propor que os alunos desenvolvam aplicações na linguagem Java para a plataforma Android, de forma a incentivar o interesse dos mesmos no estudo do conteúdo da disciplina. Para isso foram elaboradas propostas que abordam os conceitos iniciais de orientação a objetos em forma de aplicativos. Entre esses se encontram jogos a serem desenvolvidos, esperando assim que o processo de aprendizagem dos discentes se torne uma experiência mais agradável e prazerosa.

Assim, espera-se que os alunos se sintam mais estimulados a aprender a ementa apresentada, uma vez que estarão utilizando uma linguagem bastante popular no mercado de trabalho, bem como uma tecnologia de desenvolvimento gratuito e amplamente utilizada atualmente, como o sistema operacional Android.

1.3 Metodologia

As etapas da metodologia adotada para este projetos foram as seguintes:

1. Pesquisa bibliográfica sobre metodologias no ensino de programação, dificuldades no ensino de programação, uso de Android no ensino de programação, ambientes lúdicos usados para ensinar programação.
2. Levantamento das informações sobre desenvolvimento Android. Com isso foi possível aprender como fazer uma aplicação Android e também as ferramentas mais indicadas para serem utilizadas.
3. Estruturação do conteúdo a ser abordado na monografia. Nesta etapa foram criados tópicos a serem discutidos e futuramente desenvolvidos e apresentados como capítulos.
4. Definição dos projetos Android a serem apresentados, onde foram levantadas ideias de possíveis aplicações que pudessem tornar interessante dos conceitos. Dentre essas foram escolhidas as que estivessem mais ao alcance de um aluno iniciante da área produzir.
5. Desenvolvimento das aplicações.

1.4 Organização do texto

O presente trabalho está estruturado em capítulos e, além desta introdução, será desenvolvido da seguinte forma:

- Capítulo II: Estado da Arte – Breve apresentação sobre os temas que envolvem a abordagem proposta, primeiramente falando sobre o problema encontrado na aprendizagem de programação, depois mostrando duas metodologias já usadas no processo de ensino e por fim dissertando sobre alternativas já usadas para o ensino de programação.

- Capítulo III: Tecnologias utilizadas – Apresenta as tecnologias utilizadas para o desenvolvimento das aplicações.

- Capítulo IV: Projetos desenvolvidos – Projetos que foram implementados para o estudo dessa abordagem, e as propostas que os envolvem.

- Capítulo V: Conclusões – Considerações finais sobre a proposta feita e também sugestões de trabalhos futuros em relação a ela.

- Apêndice I: Tutorial de como rodar o projeto do Android Studio em seu dispositivo móvel Android - Trata-se de um tutorial contendo informações de como configurar o dispositivo para que seja possível a instalação das aplicações e também duas formas de instalação possíveis.

- Apêndice II: Método para apresentar na tela uma janela com uma mensagem.

- Apêndice III: Código do aplicativo Simon - Contém os códigos do Simon citados no texto em relação à aplicação desenvolvida.

- Apêndice IV: Código do layout do tabuleiro Mancala - Contém o código do arquivo xml (de layout) correspondente a um tabuleiro do jogo Mancala.

2 Estado da Arte

Neste capítulo serão apresentadas algumas metodologias já utilizadas no processo de ensino de programação e também algumas iniciativas que facilitam ou tornam mais interessante a aprendizagem para o aluno.

2.1 Aprendizagem de Programação

Os cursos da área de informática apresentam em suas grades curriculares, disciplinas introdutórias de programação, que de acordo com os autores de [4] e [5] têm altas taxas de reprovação. O desempenho ruim nessas disciplinas pode ter um impacto negativo maior no decorrer da graduação, quando estes forem cursar matérias que as tenham como base.

O insucesso inicial pode estar relacionado ao fato de que, como apresentado em [6], a maioria dos ingressantes na graduação superior são jovens e estão em uma fase de transição de vida neste período. A quantidade de novidades e dificuldades apresentadas nesse ambiente torna a aprendizagem do conteúdo um desafio ainda maior devido à grande instabilidade em que se encontram nessa transição [3]. Os autores deste trabalho consideram a linguagem formal (linguagem de programação) como mais uma dificuldade a ser enfrentada pelo aluno, uma linguagem com uma estrutura diferente do que sempre esteve acostumado a utilizar e que precisará aprender e se adaptar para cursar as disciplinas propostas na grade de seu curso.

Vale também ressaltar que os estudantes por muitas vezes estão acostumados a disciplinas e métodos de estudo baseados em leituras sucessivas e memorização de conteúdo, enquanto que para o aprendizado de programação é necessário prática intensiva além de uma compreensão e reflexão dos assuntos trabalhados.

"O uso exagerado da repetição *ad infinitum* de problemas com enunciados textuais [...] passado numa lista de exercícios impressa ou no quadro torna o processo de ensino e aprendizagem monótono e cansativo" [2].

Entre exercícios aplicados aos alunos também se encontram enunciados que envolvem conceitos matemáticos, como o "clássico" exemplo da *sequência de Fibonacci*, que é visto até em provas de concurso.

Fora isso, pode-se observar também que as metodologias de ensino usualmente aplicadas podem não ser apropriadas no processo ensino-aprendizagem de certas disciplinas. O professor, como agente ativo, tende a repassar a informação e esperar que o aluno assimile e até memorize para que então possa cobrar o conteúdo em forma de uma avaliação [7]. A dificuldade em uma turma também pode ser agravada caso o professor não consiga perceber isso, se acomodando apenas em continuar o seu programa de aulas e não notando que o que já foi dado ainda não foi assimilado pela maioria, acarretando assim num acúmulo de dúvidas que pode normalmente ser percebido pelos insucessos no final do período.

Os autores deste trabalho acreditam ser necessária uma atualização desta postura do professor, mudando o foco do processo para o discente ou também adotando um processo que se adeque melhor ao interesse dos alunos, alguns exemplos de metodologias que possuem este objetivo poderão ser vistos nas seções 2.2, 2.3 e 2.4.

Dentre as metodologias de ensino que são utilizadas, foram delimitadas duas a serem estudadas, consideradas mais relevantes para esse trabalho. Ambas são metodologias ativas, centradas no aluno de forma que ele se torna o protagonista no processo de aprendizagem e serão descritas a seguir. Dentre as metodologias de ensino que são utilizadas, foram delimitadas duas a serem estudadas, consideradas mais relevantes para esse trabalho. Ambas são metodologias ativas, centradas no aluno de forma que ele se torna o protagonista no processo de aprendizagem e serão descritas a seguir.

2.2 Aprendizagem Baseada em Problema (PBL)

É o eixo principal de aprendizagem teórica dentro de alguns cursos de Medicina. Seu estudo é baseado em propor problemas aos alunos com a finalidade de que os mesmos busquem estudar determinados conteúdos para que possam resolvê-los [8].

Nesse método os alunos podem ser divididos em grupos, e então depois de apresentado o caso a ser resolvido os mesmos precisam, em equipe, identificar o problema, investigar sobre esse e concluírem possíveis soluções. Como vantagens do PBL pode-se destacar [9]:

- Estímulo de criatividade;
- Promove do pensamento crítico;
- Incentiva a capacidade de análise e decisão e
- Proporciona trabalho em equipe.

Com esta metodologia o aluno passa de um receptor passivo nesse processo de aprendizagem, e passa a ser um agente ativo e responsável pela formação do seu conhecimento [10]. Isso motiva o despertar de uma autonomia maior no aluno, que pode vir a contribuir na sua carreia profissional e também na sua vida social.

2.3 Aprendizagem Baseada em Projetos

Nessa abordagem o conhecimento também é construído pelo estudante, porém em forma de progresso no desenvolvimento do seu projeto. Esses projetos são baseados em questões ou problemas que proporcionam um desafio ao estudante, devem motivar e conduzir o aluno a uma busca de conhecimento para resolvê-los. [11]

Estes projetos são normalmente atividades a longo ou médio prazo e também muitas vezes interdisciplinares. Com essa abordagem o aluno muitas vezes deve organizar seu tempo e suas prioridades, caso seja feito em equipe, por exemplo, cabe ao grupo ou mesmo a um único aluno a gestão dessa atividade. O professor passa então a ser como um tutor, que auxilia nas dificuldades, mas não tem um papel ativo em repassar conteúdos conceituais [12]. Dessa forma os estudantes desenvolvem mais responsabilidades e também um comportamento mais proativo em relação a trabalhos.

2.4 Abordagens Diferentes

Existem estudos sobre como tornar a aprendizagem das linguagens de programação mais interessante para os alunos. Alguns encontraram no meio lúdico uma possível solução para isso, utilizando, como por exemplo:

2.4.1 Ambientes gráficos

2.4.1.1 *Alice*[1]

O *Alice* é um ambiente de programação que permite uma criação simples de animação em 3D, tanto para desenvolvimento de jogos como também para criação de vídeos. Este software está disponível gratuitamente e tem como foco auxiliar o ensino de programação orientada a objetos. Sua interface interativa (veja Figura 1 e Figura 2) permite que os alunos trabalhem com elementos gráficos facilmente, tornando mais interessante a implementação em linguagens como Java e C++.

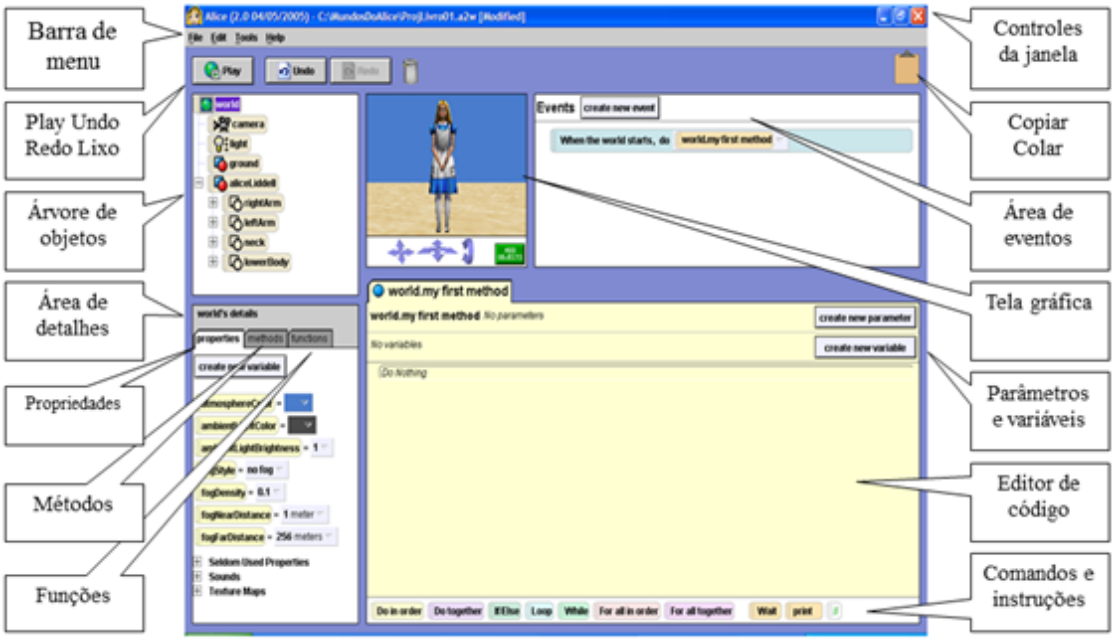

Figura 1 - Interface do Alice com indicações.[2]

[1]http://www.alice.org/index.php (acessado em 13/12/2015)

[2]Fonte: "Alice: Uso do software no processo educacional junto aos cursos de engenharia", http://www.abenge.org.br/CobengeAnteriores/2012/artigos/104311.pdf., Acessado em 20/12/2015

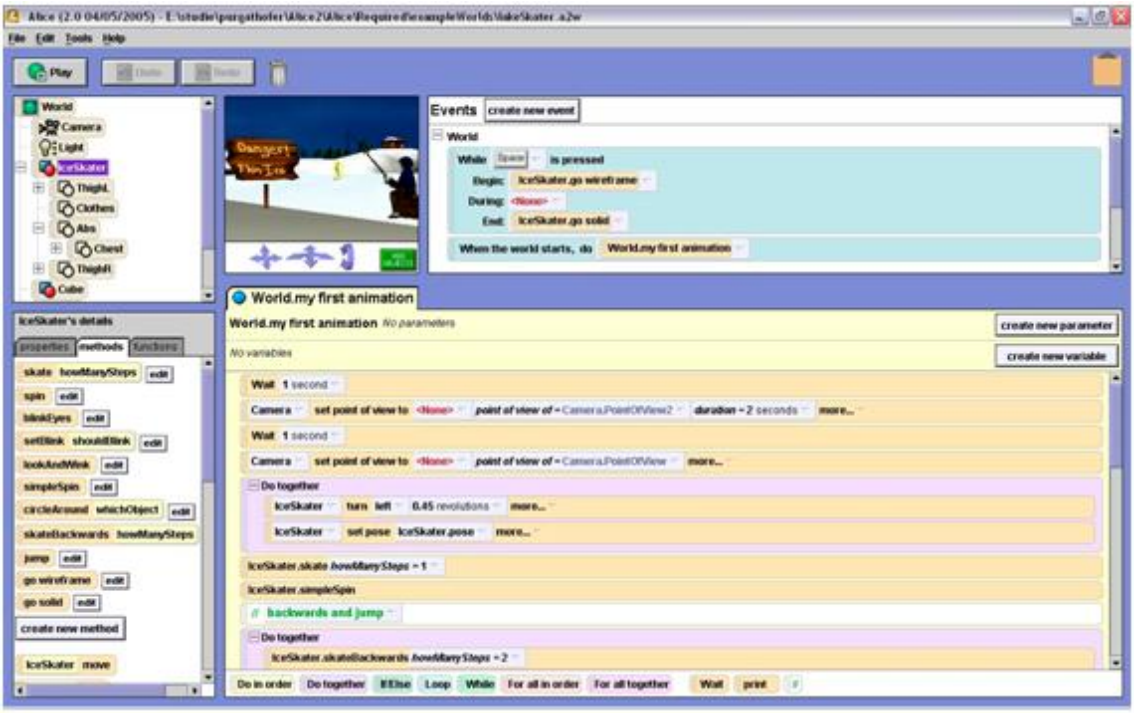

Figura 2 - Interface do Alice.[3]

2.4.1.2 *Scratch*[4]

Este é um ambiente de programação visual desenvolvido pelo *Lifelong Kindergarten Group* (LLK), grupo de pesquisa do *MIT Media Lab*, que tem como objetivo introduzir os conceitos de programação para aqueles que nunca tiveram contato com esse conteúdo. [13]

Este software permite que aplicativos sejam implementados de forma intuitiva (ver Figura 3) através de uma integração com recursos multimídia.

No Scratch, a programação se baseia em blocos, e estes constroem estruturas básicas de programação, que agrupadas formam um script a ser executado. [14]

[3] Fonte: Wikipédia - Alice (software) <https:en.wikipedia.org/wiki/Alice_(software)> Acessado em 14/12/2015

[4] https://scratch.mit.edu/(acessado em 16/12/2015)

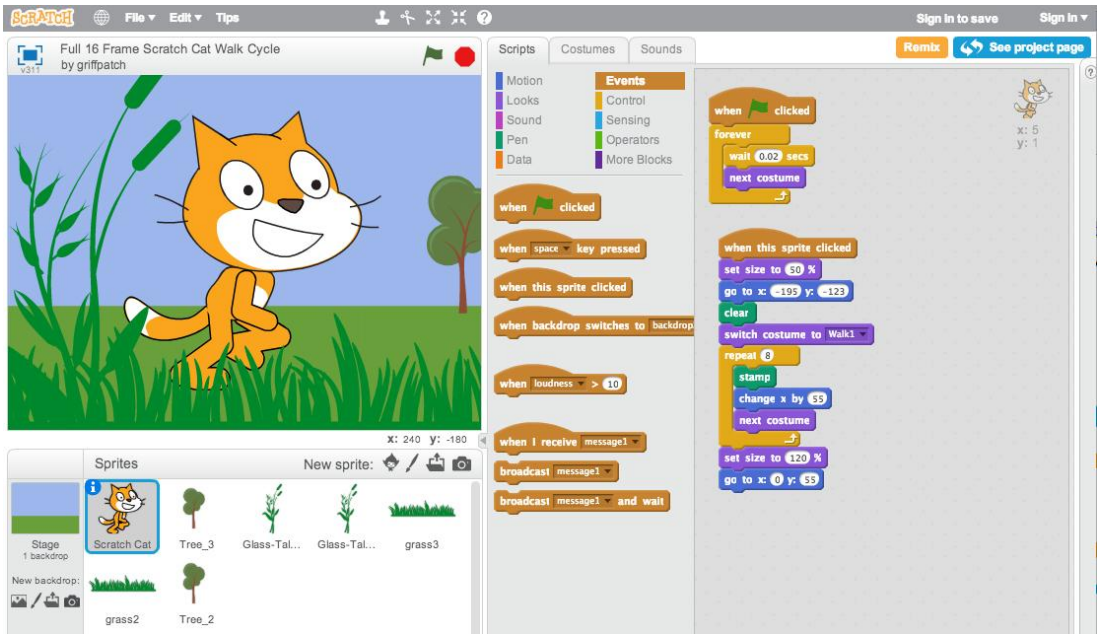

Figura 3 - Interface do Scratch.[5]

2.4.1.3 *Greenfoot*

O Greenfoot é um software baseado em abordagens construtivas de aprendizagem. Com um ambiente interativo, ele motiva a exploração e experimentação da ferramenta e também proporciona ao aluno um acesso mais fácil à manipulação de elementos gráficos [15], como mostrado nas figuras “Figura 4”, “Figura 5” e “Figura 6” a seguir.

[5] Fonte: Mit News <http://news.mit.edu/2013/scratch-two-released-0514> Acessado em 14/12/2015

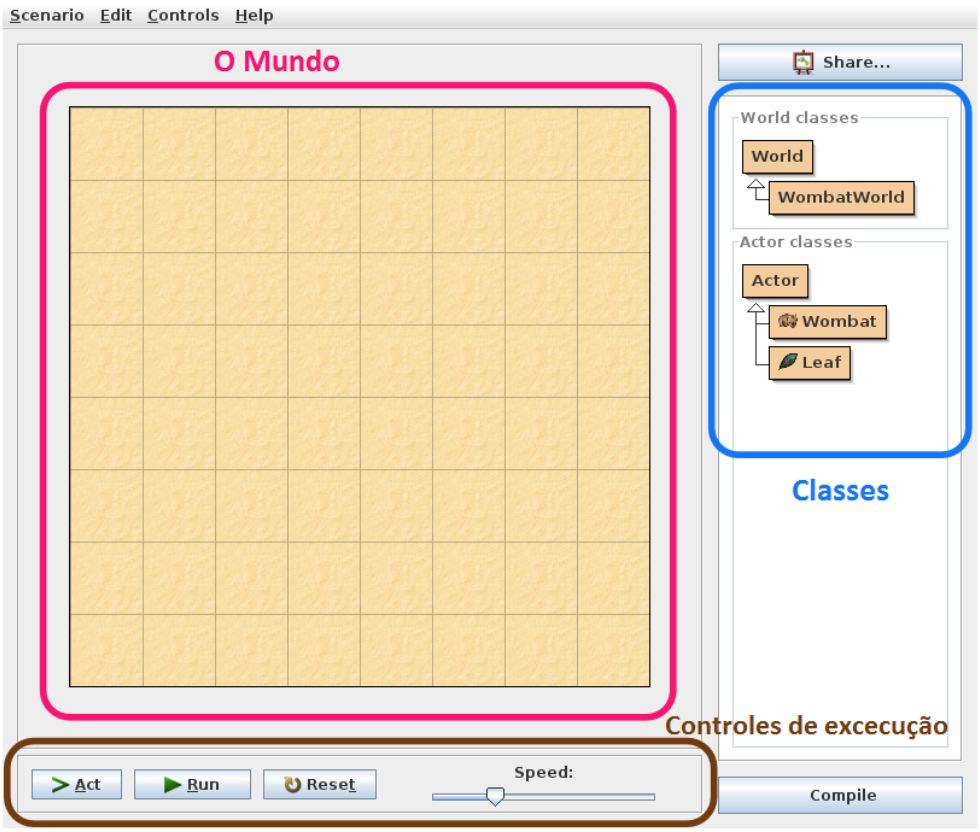

Figura 4 - Interface do Greenfoot com indicações (imagem traduzida).[6]

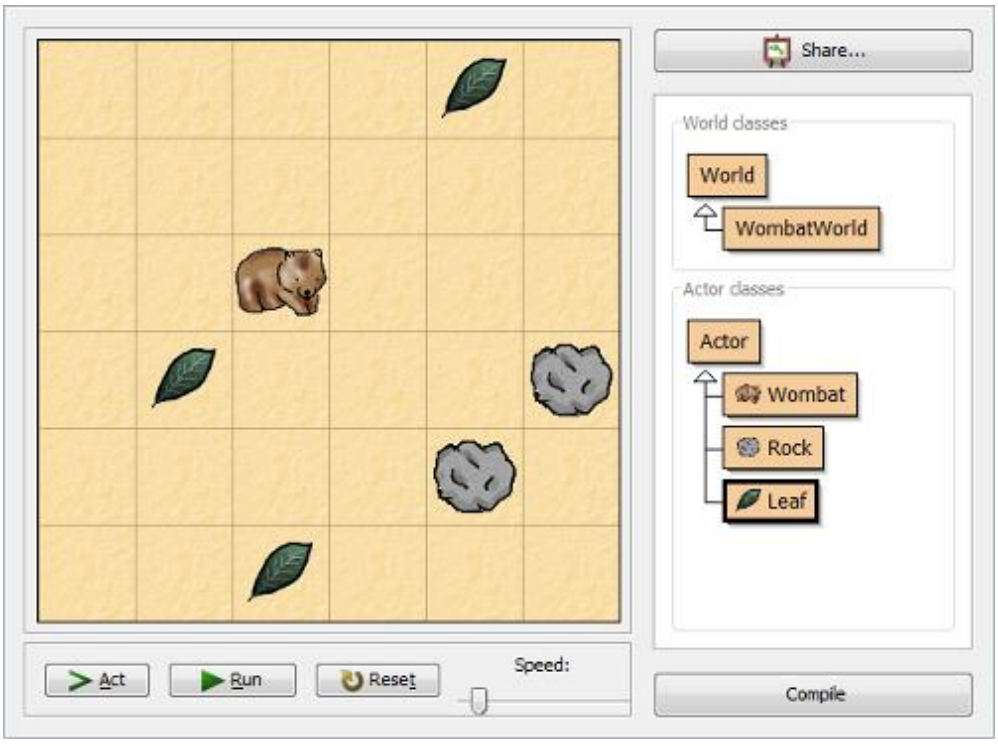

Figura 5 - Interface do Greenfoot.[7]

[6]Fonte: Greenfoot <http://www.greenfoot.org/doc/tut-1> Acessado em 14/12/2015

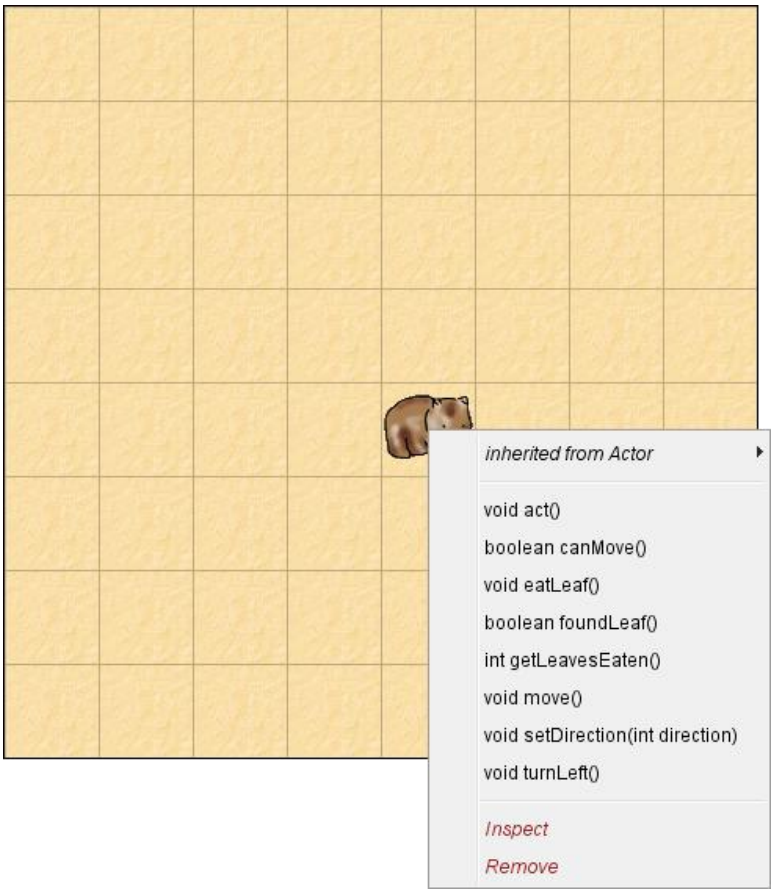

Figura 6 - Um exemplo de ações de um objeto no Greenfoot.[8]

2.4.1.4 *Lego Mindstorms*

O Lego Mindstorms é uma linha de brinquedos LEGO, porém voltada para a área de ensino tecnológico. Trata-se de um kit contendo diversas peças para a montagem de um robô (ver Figura 7) e também o ambiente NXT-G (ver Figura 8), que possui uma linguagem gráfica para programação baseada em blocos (ver Figura 9). [16]

[7]Fonte: Greenfoot <http://www.greenfoot.org/overview> Acessado em 14/12/2015
[8]Fonte: Greenfoot <http://www.greenfoot.org/doc/tut-1> Acessado em 14/12/2015

Figura 7 - Modelo de robôs do Lego Mindstorms[9]

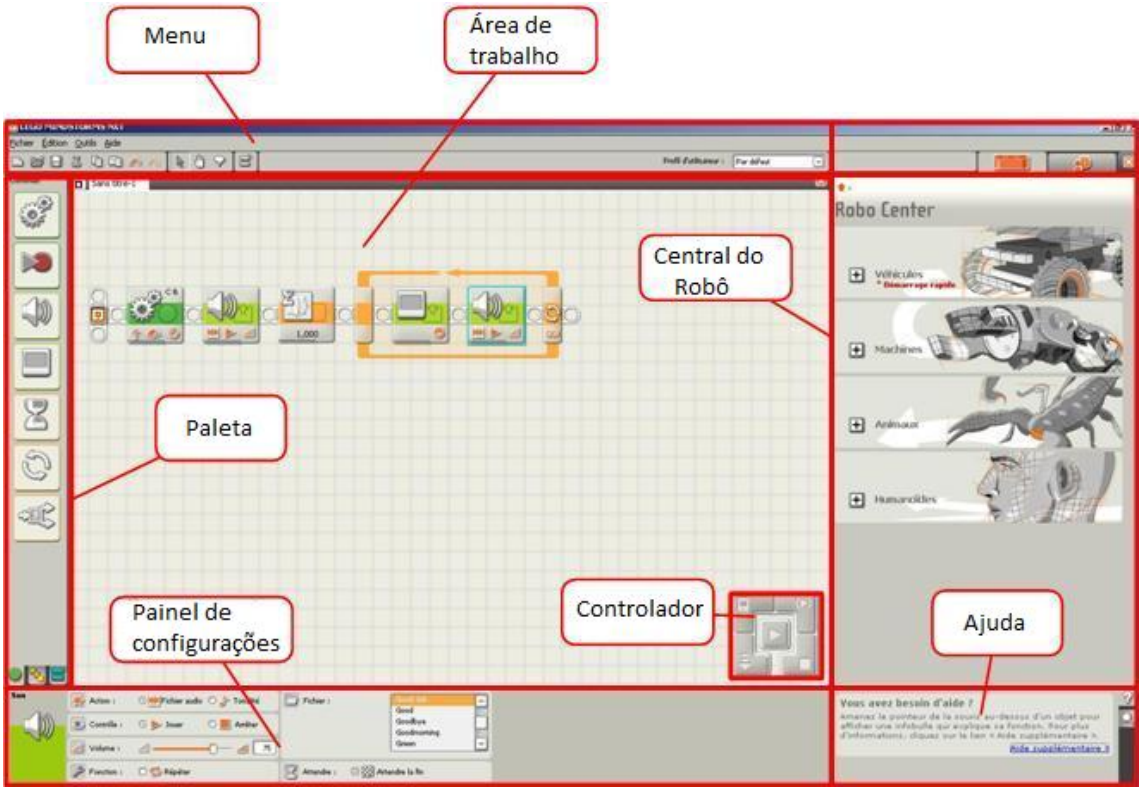

Figura 8 - Interface do Lego Mindstorms com indicações (imagem traduzida).[10]

[9] Fonte: Montagem de imagens retiradas dos sites <http://blog.robotto.com.br/> e <http://www.educatec.ch>

[10] Fonte: <http://www.generationrobots.com/en/content/61-nxt-g-programmation-lego-mindstorms-nxt> Acessado em 14/12/2015

Figura 9 - Programação em blocos no Lego Mindstorms (imagem traduzida)[11]

2.4.2 Uso de jogos no ensino

Como visto na seção 2.1, a motivação em estudar os conteúdos apresentados acerca de programação pode ser considerada um fator importante nos resultados do aluno em relação ao curso. Uma abordagem envolvendo jogos, para alguns autores, foi uma alternativa encontrada para que fosse despertado o interesse no aluno.

Em [17] é apresentado um resultado de uma alteração no currículo do curso Técnico em Informática do CEFET/RJ, em 2010, quando foi inserida uma disciplina de Algoritmos no inicio do curso e outra de Desenvolvimento de Sistemas no final do mesmo, e em 2011 essas disciplinas passaram a ter um foco maior em desenvolvimento de jogos computacionais. Os projetos desenvolvidos pelos alunos foram apresentados na feira de tecnologia do CEFET/RJ, entre eles um jogo em Java baseado no *Space Invaders* e outro, na mesma linguagem, chamado de "**Passando em Programação**", onde é ilustrada a relação de alunos em com essa matéria. Após a conclusão da aplicação dessa abordagem, foi feita uma pesquisa com os discentes envolvidos, onde os autores concluíram que 100% dos alunos consideraram que a proposta favoreceu o aprendizado de programação. De acordo com a pesquisa feita, pode-se também concluir que:

- 88% passaram a gostar mais de programação após essa experiência (os outros 12% já gostavam de programação antes disso);

[11]Fonte: <http://www.generationrobots.com/en/content/61-nxt-g-programmation-lego-mindstorms-nxt>Acessado em 14/12/2015

- 59% consideraram que criar jogos tornou a tarefa mais fácil e diminuiu o impacto associado à compreensão da abstração sobre o mundo real;
- 77% consideraram que o déficit de atenção nessas disciplinas diminuiu;
- 35% consideraram que o desenvolvimento de jogos poderia ser integrado com outras disciplinas como, por exemplo, Banco de Dados, o que permitiria uma sofisticação de armazenamento de dados, ou Redes, aonde poderiam jogar com duas ou mais pessoas ao mesmo tempo.

No artigo [18] pode-se encontrar um estudo sobre uma abordagem de ensino proposta através do uso de desenvolvimento de jogos em cursos na área de Ciência da Computação. Os autores afirmam que a motivação dos alunos é um dos principais problemas encontrados, e propõem o uso da metodologia de desenvolvimento de jogos para computadores na tentativa de despertar o interesse deles. A implementação seria aplicada à ementa de disciplinas vistas no decorrer do curso em questão. A abordagem foi aplicada em alunos do primeiro ano do curso de Bacharelado em Sistemas de Informação, em 2004, no *Instituto Superior Tupy*, e como resultado da mesma foi possível notar que o empenho e a motivação dos alunos envolvidos teve uma melhora durante a atividade. Em uma visão geral, os autores do artigo puderam observar que a metodologia teve aceitação positiva por parte dos alunos e também da coordenação do curso.

2.4.2.1 *Castelo dos Enigmas*[12]

O Castelo dos enigmas é um jogo que incentiva a construção do conhecimento através de desafios, que inicialmente exploram os conceitos básicos de sintaxe da estrutura de um algoritmo e vai evoluindo os níveis de dificuldade até chegar, por exemplo, em desafios que façam o aluno escolher corretamente uma estrutura de dados.

No enredo do jogo é contada a historia de um aluno que tem uma prova de programação para fazer, porém só começa seus estudos para ela na noite anterior ao dia da prova. Cansado, ele dorme e então sonha estar em um castelo do qual precisa sair (Figura 10). Em sua busca pela saída o aluno encontra inimigos que o atacam, os quais devem ser derrotados, assim como também enigmas que devem ser resolvidos [19].

[12] Disponível em: https://code.google.com/p/castelo-dos-enigmas/ (acessado em 16/12/2015)

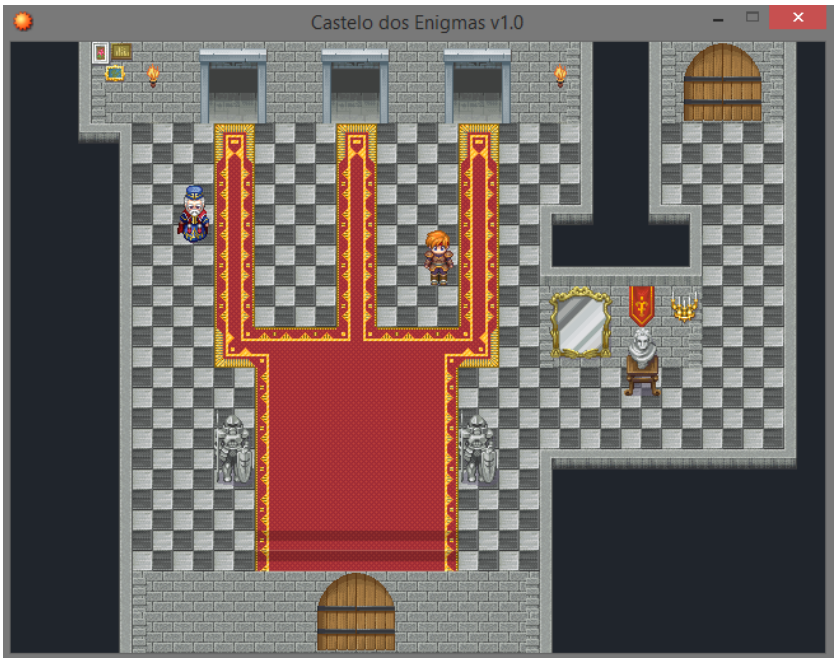

Figura 10 - Hall do castelo.[13]

Na Figura 11 pode-se ver uma cena em que o personagem está lutando contra os inimigos, que no caso são morcegos. A espada que ele utiliza, por exemplo, foi um item encontrado enquanto explorava salas do castelo, assim como a chave para abrir a porta do cômodo em que ele se encontra.

Figura 11 - Personagem atacando morcegos no jogo.[14]

[13]Fonte: *print screen* da aplicação em 01/12/2015

Após derrotar os inimigos, para seguir adiante o personagem precisa abrir um portão através de um enigma (como representado na Figura 12 e Figura 13). O jogador precisa fazer com que a condição do peso da balança seja atendida para que a porta se abra. A indicação do que deve ser feito está em forma de pseudocódigo e cabe ao aluno interpretar o que precisa ser feito.

Figura 12 - Apresentação da tela iniciando o algoritmo a ser interpretado.[15]

[14]Fonte: *print screen* da aplicação em 01/12/2015
[15]Fonte: *print screen* da aplicação em 01/12/2015

Figura 13 - Algoritmo a ser interpretado pelo jogador.[16]

Como o personagem sozinho na balança pesa apenas 5 (Figura 14), o jogador acaba precisando utilizar os sacos de dinheiro obtidos naquela sala após vencer os inimigos. Se observar a Figura 11 novamente, verá mais claramente que na sala há um personagem "velhinho". Ele dá dicas caso o aluno não entenda o que deve ser feito para passar o portão.

Figura 14 - Personagem na balança. [17]

[16]Fonte: *print screen* da aplicação em 01/12/2015

A Figura 15 mostra que assim que o jogador consegue atender a condição (Peso maior que 10) o portão se abre, podendo assim avançar para a próxima fase.

Figura 15 - Personagem conseguiu atender a condição do peso da balança para abrir a porta.[18]

À medida que o jogador vai avançando os níveis, ele está construindo seu conhecimento, e o aluno, personagem do jogo, estudando e aprendendo.

Além desta ferramenta para ensino de programação, vale ressaltar a prática com estudantes do curso de Sistema de Informação da USP (Universidade de São Paulo) durante os anos de 2010 e 2011. A atividade consistia em competições formando equipes entre os alunos, na qual cada uma deveria solucionar um problema proposto. Através de atividades extracurriculares foi possível constatar por meio de questionários respondidos por participantes que a abordagem incentivou o interesse e também contribuiu para o desenvolvimento das habilidades dos mesmos [20].

Na seção seguinte encontra-se um relato de abordagem baseada em jogos feita com alunos do curso de Sistema de Informação da UNIRIO.

[17]Fonte: *print screen* da aplicação em 01/12/2015
[18] Fonte: *print screen* da aplicação em 01/12/2015

2.5 Relato de uma experiência de abordagem com jogos

Na Universidade Federal do Estado do Rio de Janeiro (UNIRIO), o professor Alexandre Luis Corrêa, no segundo semestre de 2014, iniciou na disciplina de *Técnicas de programação 2* um projeto no qual os alunos trabalhavam com desenvolvimento de aplicações do tipo jogos. Este projeto foi continuado no primeiro semestre de 2015, pelo professor Pedro Nuno de Souza Moura, que esclareceu que a abordagem propunha como projeto final da disciplina o desenvolvimento de um jogo em Java (linguagem aprendida pelos alunos), utilizando o *Java 2D*[19], que auxilia, através de uma série de artefatos, a desenvolver um jogo 2D.

Os alunos, segundo o professor, tiveram aproximadamente um mês para desenvolver o jogo e apresentarem em sala de aula. A fim de não permitir que, por exemplo, os alunos escolhessem jogos de complexidades que não se aplicassem à disciplina, foi determinado que estes escolhessem um entre três "tipos" de jogos. A turma então foi dividida em duplas, e para cada dupla foi sorteado o tipo de jogo que esta deveria desenvolver.

As propostas de tipos de jogos eram:

2.5.1 Pacman

Produzido originalmente para *Arcade* (também conhecido como fliperama) tornou-se um dos jogos mais populares da época em que foi lançado e alcançou diversos consoles. O jogo consiste em uma cabeça redonda amarela, que possui uma boca que apenas abre e fecha, e é posicionada em um labirinto com pastilhas nos caminhos onde fantasmas a perseguem. O objetivo é "comer" todas as pastilhas sem que os fantasmas o alcancem. [21]

[19] https://docs.oracle.com/javase/tutorial/2d/overview

Figura 16 - Jogo Pacman.[20]

2.5.2 Enduro

É um jogo de corrida de carro da época do console *Atari*, aonde o jogador controla um carrinho em uma pista, com o objetivo de passar um determinado número de carros oponentes antes de chegar ao fim da corrida, tornando assim possível continuar a corrida no dia seguinte. Alcançar esse objetivo se torna um tipo de *save point* no jogo. [22]

Figura 17 - Tela do jogo Enduro.[21]

[20]Fonte: ScreenMania<http://screenmania.retrogames.com/arcade/03/arcade_0066.html> Acessado em 14/12/2015

2.5.3 Space Shooter

Assim como em *Space Invaders*, neste jogo têm-se uma nave e um ambiente que ilustra o espaço, e nele são encontrados inimigos que devem ser atacados e destruídos.

Figura 18 - Tela de um jogo similar ao Space Shooter sugerido.[22]

O professor pôde perceber uma motivação maior dos alunos nessa atividade, que estavam bastante orgulhosos no dia da apresentação. Alguns grupos conseguiram fazer jogos com mais refinamento, porém a turma no geral conseguiu alcançar um bom nível.

O mesmo relata que os alunos conseguiram evoluir bem, até determinado ponto, precisando dar mais orientações algumas vezes, como por exemplo, em relação à interface gráfica. O professor Pedro descreve que a maior dificuldade foi na aprendizagem da linguagem orientada a objetos, uma vez que era o primeiro contato dos discentes com esse conteúdo. Ele guiou os alunos definindo quais classes deveriam ser implementadas, de modo que eles não tentassem, por exemplo, programar usando um conceito de programação estruturada.

Foi questionado qual a dificuldade encontrada enquanto docente. O mesmo respondeu que nunca havia lidado com o *Java 2D*, foi preciso um estudo para que pudesse ensinar aos alunos. Ele afirmou que a experiência pode ser considerada bem sucedida. E que desenvolver projetos com os alunos é motivador até para o docente.

---

[21]Fonte: Exame.com <http://exame.abril.com.br/blogs/aplicativos/internet/12-apps-para-celebrar-os-40-anos-da-atari> Acessado em 14/12/2015

[22]NoobTuts<http://noobtuts.com/unity/2d-space-shooter-game> Acessado em 14/12/2015

Em [23] é relatado que “o aprendizado precisa se aproximar do entretenimento para conseguir engajar os alunos”. Concordando com esse pensamento, foi decidido direcionar este projeto para o ensino de programação de modo que seja interessante para os alunos, através do desenvolvimento de aplicações para Android como será visto no capítulo 4. Foram propostas aplicações Android que podem ser utilizadas para o ensino dos conceitos básicos de programação orientada a objetos utilizando a linguagem de programação Java. O capítulo a seguir irá discorrer sobre as ferramentas e tecnologias que foram utilizadas nesse projeto para o desenvolvimento dessas aplicações.

3 **Tecnologias utilizadas**

Neste capítulo serão apresentadas informações sobre as tecnologias que foram utilizadas durante a implementação do projeto.

3.1 O sistema Android

O Android é um sistema operacional para dispositivos móveis como *tablets* e *smartphones*. Foi construído com a intenção de que fosse possível, a desenvolvedores inclusive, sempre estar explorando o máximo do potencial do sistema.

Uma aplicação pode, por exemplo, apelar a qualquer uma das funcionalidades do núcleo do dispositivo, de tal forma que permite ao desenvolvedor adaptá-las e evoluí-las no seu sistema [24]. Sendo *open source*[23], o Android permite que novas tecnologias possam ser incorporadas ao sistema operacional de forma a manter a evolução contínua do sistema, uma vez que as comunidades de desenvolvedores trabalham em conjunto para construir aplicações móveis inovadoras.

"A plataforma Android foi desenvolvida com base no sistema operacional Linux e é composta por um conjunto de ferramentas que atua em todas as fases do desenvolvimento do projeto, desde a execução até a criação de softwares específicos." [24]

[23] Código aberto

3.2 A linguagem Java

Um dos motivos da popularização do sistema Android foi a escolha da linguagem Java para o desenvolvimento dos aplicativos. Além de ser uma das linguagens mais utilizadas atualmente na área de programação[24], é gratuita e de código-fonte aberto.

"O grau de abertura da plataforma estimula a rápida inovação. O Android está disponível em dispositivos de dezenas de fabricantes de equipamentos originais" [25].

Java é uma linguagem orientada a objetos e tem bibliotecas de classes que auxiliam a desenvolver aplicativos rapidamente.

Porém, a versão do Java utilizada para desenvolvimento Android não é a "versão completa", e sim um subconjunto de bibliotecas Java que são específicas para Android. Este subconjunto, por exemplo, exclui classes que não são adequadas para os dispositivos móveis. [26]

3.3 Ambiente de desenvolvimento

Para este projeto, foi escolhido como ambiente de desenvolvimento o **Android Studio**[25]. Esta é uma **IDE** (*Integrated Development Environment*) relativamente nova. Ela teve sua primeira versão (*v0.1x*) lançada em maio de 2013, sua primeira versão estável (*v1.0*) lançada em dezembro de 2014 e sua última versão lançada é a versão *v1.5.1* (dezembro de 2015) [27]. Se por um lado isso é mal visto, pois incita sinônimo de instabilidade e *bugs*, por outro ela surpreende com tecnologias mais novas e interface com o usuário mais atual.

Como é uma ferramenta voltada somente a desenvolvimento Android, ela, por ter esse foco, consegue perceber e atender melhor as necessidades desse grupo específico de desenvolvedores. Por exemplo, o *preview* de layouts está mais robusto e completo do que aquele existente na IDE *Eclipse*[26]. A pré-visualização abrange diversos formatos e tamanhos de telas, e de diferentes dispositivos Android, como Tablets, Smartphones, e até relógios e óculos.

Após pesquisas sobre ferramentas, surgiu a dúvida de qual deveria ser utilizada: o Eclipse ou o Android Studio. A Google indica o Android Studio como a ferramenta ideal para desenvolvimento Android, e o Eclipse já estava limitado a desenvolver

[24]http://redmonk.com/sogrady/2015/01/14/language-rankings-1-15/ (acessado em 16/12/2015)
[25]http://developer.android.com/(acessado em 16/12/2015)
[26]https://www.eclipse.org/(acessado em 16/12/2015)

aplicações que alcançavam somente até a versão 4.4.2 do sistema Android, como mostrado na Figura 19. Vale ressaltar também que o Eclipse até o final do ano de 2015 deixará de receber o suporte da Google, já que a mesma empresa lançou o Android Studio em dezembro de 2014 e irá focar apenas nesta IDE [28][29].

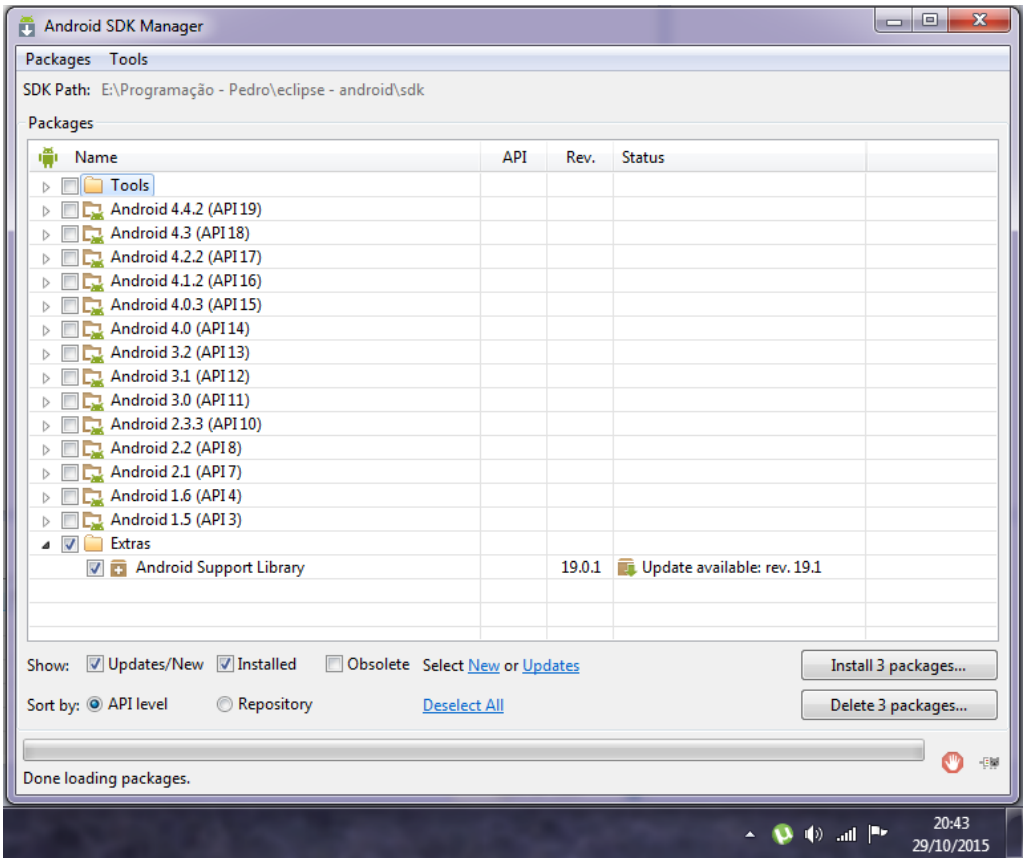

Figura 19 - Tela de atualização do pacote SDK do Eclipse.[27]

3.4 Controle de versão

A finalidade do controle de versão é dar um controle maior sobre tudo que você altera no seu projeto de software [30].

Um **controle de versão centralizado** mantém em um **repositório central** o histórico de todas as versões criadas, as alterações que foram feitas em cada uma dessas versões, o responsável pela alteração e quando ela foi feita. Ele disponibiliza isso a todos os envolvidos no projeto e permite a recuperação de qualquer versão do histórico a qualquer momento.

[27]Fonte: *print screen* do ambiente de desenvolvimento Eclipse em 29/10/2015

As versões guardam os artefatos do projeto em um determinado momento do desenvolvimento (de preferência em que o sistema esteja estável), e devem ser criadas para tratar de uma questão específica e ter uma descrição do seu propósito.

Cada desenvolvedor no projeto deve ter sua "**área de trabalho**" (ou cópia de trabalho), que é uma cópia local dos arquivos do repositório, onde pode-se modificar os artefatos do projeto sem interferir (ou sofrer interferência de) ninguém. Somente após terminar a sua modificação por completo e estando com o sistema estável, deve-se levar as alterações para o repositório central. Na Figura 20 é representado este processo, aonde as cópias de trabalho enviam modificações para o repositório com a operação "*commit*" e obtém a versão atual do repositório com a operação "*update*".

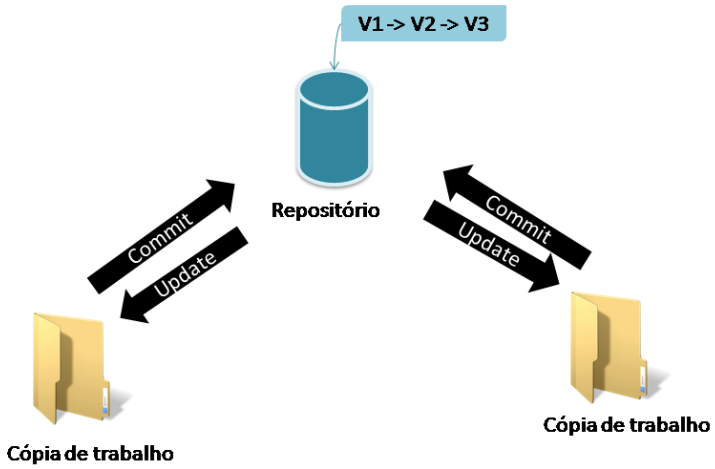

Figura 20 - Controle de versão do trabalho. [28]

> "A utilização de controle de versões apresenta vantagens na solução de problemas para controle e rastreamento de alterações durante a fase de desenvolvimento do software, como a análise de histórico do desenvolvimento, resgate de versões antigas, ramificações do projeto dentre outras atividades." [31]

Para que houvesse um controle melhor de alterações no código, neste projeto, foi adotado um sistema de controle de versão, o ***TortoiseSVN***[29], que é um cliente do **Subversion** (**SVN**) para o Windows e independente de IDE (*Integrated Development Environment*).

[28] Fonte: produzido pelos autores
[29] http://tortoisesvn.net/ (acessado em 16/12/2015)

O sistema de controle de versão do projeto possui um repositório central, que está alocado no ***GitHub***[30], e cada desenvolvedor possui uma cópia do fonte em sua máquina. [31]

Desta forma foi possível manter o controle do histórico do projeto para análises de desenvolvimento, e também graças ao resgate de versões anteriores do projeto conseguiu-se que erros fossem identificados mais rapidamente. Além disso, um controlador de versão minimiza os problemas que existiriam com conflitos em edições, facilitando assim a colaboração em equipe no desenvolvimento da aplicação.

[30]https://github.com/ (acessado em 16/12/2015)

4 Projetos desenvolvidos

Neste capítulo serão apresentadas propostas de aplicativos com diferentes níveis de dificuldade, com o objetivo de utilizá-los como exemplo no processo de ensino-aprendizagem de programação e linguagens orientadas a objetos.

Para que esta atividade seja realizada, é indicado que o docente forneça aos alunos as regras gerais do jogo apresentadas na seção referente ao "Enunciado" de cada projeto, o modelo UML (caso julgar necessário ou relevante) e orientações no que os alunos possam vir a ter maiores dificuldades como, por exemplo, montar interface gráfica.

Para o desenvolvimento do projeto os alunos devem ter noções básicas de programação na linguagem Java, conhecer a ferramenta de trabalho em que desenvolverão o projeto (no caso o Android Studio) e saber definir os botões e as ações quando estes forem clicados.

4.1 Projeto Jogo da Velha

4.1.1 Enunciado

Conhecido como Jogo da Velha, o nome deste jogo tem origem da Inglaterra, onde nos finais de tarde quando as mulheres se reuniam para conversar e bordar, as mais idosas, por não conseguirem mais bordar, jogavam esse jogo para passar o tempo. [32]

As regras são:

1. Jogado por 2 jogadores, cada um com um símbolo diferente. Geralmente os símbolos são um círculo (O) e um xis (X).

2. Joga-se em um tabuleiro que é formado por 9 casas, dispostas em 3 linhas e 3 colunas.
3. A cada vez o jogador deve preencher uma casa do tabuleiro com o seu símbolo e aí passa para a vez do outro jogador.
4. Ganha o jogador que completar primeiro a fileira de 3 casas do tabuleiro com o seu símbolo na horizontal, na vertical ou na diagonal.
5. Se as 9 casas do tabuleiro tiverem sido preenchidas e nenhum jogador conseguiu preencher uma fileira de 3 casas com seu símbolo, diz-se que a partida "**deu velha**", ou seja, ninguém venceu.

A partir das regras do jogo, deve-se implementar uma aplicação para Android desse jogo, onde deve existir uma classe Jogador e uma classe Tabuleiro.

4.1.2 Objetivos

Estruturação de elementos. No caso, foram estruturados os botões da tela alocando estes em uma **matriz** de botões no Java. Desta forma a apresentação ficou mais organizada facilitando a manipulação dos mesmos.

Utilização de **classes** e **construtores** com parâmetros (Jogador e Tabuleiro).

Apresentação do **conceito de acesso a endereço de memória** ("ponteiro") de forma "implícita" ao fazer o atributo "jogadorCorrente" corresponder ao endereço de memória de "jogador1" ou "jogador2" conforme a rodada. E na matriz de botões, onde cada posição da matriz guarda o endereço de memória de um botão do tabuleiro na tela. (Obs.: Há diversas outras utilizações deste conceito, mas os autores citam estas por achar casos mais interessantes, pois em alguns momentos há mais de uma instância referenciando o mesmo objeto).

4.1.3 Modelo UML

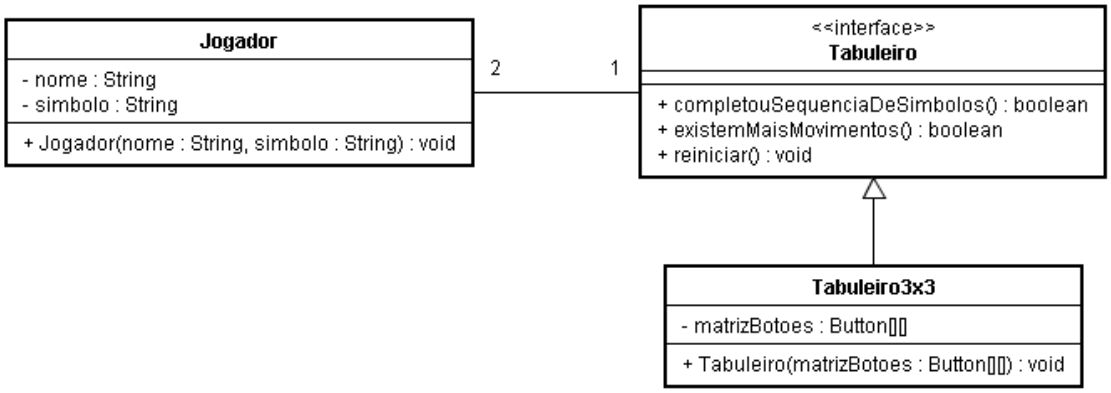

Figura 21 - Modelo UML do aplicativo Jogo da Velha. [31]

4.1.4 Passos a serem executados no processo de ensino aprendizagem

Sugere-se que o professor forneça ao aluno o método implementado "*mostrarCaixaDialogoSimples*",presente no Apêndice II. Este pode ser usado para apresentar ao usuário uma mensagem em uma janela na tela do Android. Na implementação do exemplo para este projeto, foi utilizado esse método para mostrar uma mensagem ao final do jogo, no caso de vitória ou derrota do jogador (ver Figura 22 e Figura 23). Embora a implementação não seja tão complexa, ela é específica para Android. Não é parte do objetivo que o aluno invista seu tempo de estudo procurando uma forma de apresentar uma simples mensagem.

[31]Fonte: produzido pelos autores

Figura 22 – Jogo da Velha: Tela anunciando vencedor. [32]

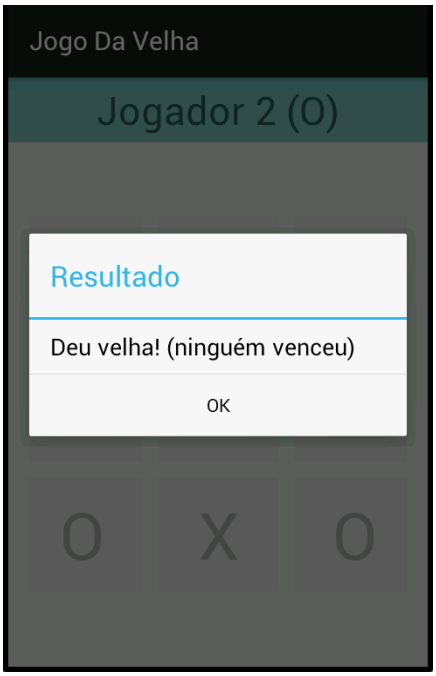

Figura 23 – Jogo da Velha: Tela anunciando empate.[33]

[32]Fonte: *print screen* da aplicação
[33]Fonte: *print screen* da aplicação

4.1.5 Armadilhas e dificuldades encontradas

Para a construção da interface do jogo, a princípio surgiu a idéia de utilizar botões organizados na tela como as casas de um tabuleiro do jogo da velha. Dessa forma, resultaria numa interface simples e fácil de ser criada. Porém na prática, notou-se que conforme o tipo de layout[34] a ser utilizado, esta construção poderia se tornar mais difícil. Devido ao pouco conhecimento que o aluno possa ter sobre interface gráfica no desenvolvimento Android, vale informá-lo quais tipos de layout são mais indicados para montar a tela da aplicação.

No desenvolvimento do jogo da velha foi utilizado o layout *TableLayout*[35] para fazer o tabuleiro do jogo como apresentado na Figura 24 e Figura 25, mas nada impede que o aluno possa utilizar outros tipos disponíveis, como o *Linear Layout*[36] que também facilita o alinhamento das casas do tabuleiro.

Figura 24 – Jogo da Velha: Tela do tabuleiro. [37]

[34]http://developer.android.com/intl/pt-br/guide/topics/ui/declaring-layout.html#CommonLayouts(acessado em 05/12/2015)
[35]http://developer.android.com/intl/pt-br/guide/topics/ui/layout/grid.html(acessado em 05/12/2015)
[36]http://developer.android.com/intl/pt-br/guide/topics/ui/layout/linear.html(acessado em 05/12/2015)
[37]Fonte: *print screen* da aplicação

Figura 25 - Jogo da Velha: Tela do tabuleiro após jogadas. [38]

4.2 Projeto Genius/Simon

4.2.1 Enunciado

Popular na década de 80, o Genius era um brinquedo que através de botões coloridos, apresentava uma sequência a ser seguida pelo usuário. No Brasil o jogo ganhou esse título e foi fabricado pela empresa Estrela. Similar a ele existe o Simon que é da empresa americana Hasbro [33]. O objetivo do jogo é que o usuário memorizasse a ordem apresentada das cores, e depois a repetisse sem errar.

Regras do jogo:

1. O jogo é jogado por um jogador somente.
2. Na tela do jogo são apresentados 4 botões de cores diferentes.
3. Ao iniciar o jogo uma cor deve ser piscada (destaca a cor depois volta ela ao normal) na tela e o jogador deve clicar na cor que piscou. Na próxima rodada uma sequência de duas cores devem piscar na tela, logo após o jogador deve clicar nas cores na mesma sequência em que piscaram. Se o jogador acertar, este passa para a próxima rodada, sempre com uma sequência com uma piscada a mais que a rodada anterior.

[38]Fonte: *print screen* da aplicação

4. Se o jogador erra a sequência de cliques, ele perde e o jogo termina.
5. Pode ser definido um tamanho máximo da sequência para o jogador vencer quando acertar uma sequência deste tamanho, ou deixar em aberto para o jogador ver qual o seu recorde.

Pode ser sugerido também que o jogo seja implementado com mais opções de cores. O jogo funciona da mesma forma, mas a dificuldade para o jogador aumenta. A implementação deve ser alterada principalmente no arquivo xml do layout da tela do jogo, adicionando novos botões, já a lógica no *back-end* será pouco alterada.

4.2.2 Objetivos

Prática do uso de conceitos de **Lista** com o uso de **bibliotecas** como a java.util.List ou até alguma classe similar a esta implementada pelo próprio aluno, como por exemplo a classe Lista que foi utilizada neste projeto, o que tornou mais fácil a manipulação da lista sequencial de botões sorteados se comparado com um array, uma vez que não é preciso saber externamente em que posição do array se deve inserir novas instâncias, por exemplo.

O uso de Lista nesse projeto é uma boa oportunidade para o ensino de ***generics*** em Java (equivalente a **templates**, em C++), visto que conceitualmente uma lista pode ser de qualquer coisa.

Aplicação de conceitos de **Interface**[39] e **Array**.

[39]https://pt.wikipedia.org/wiki/Interface_(programa%C3%A7%C3%A3o_orientada_a_objetos)(acessado em 21/12/2015)

4.2.3 Modelo UML

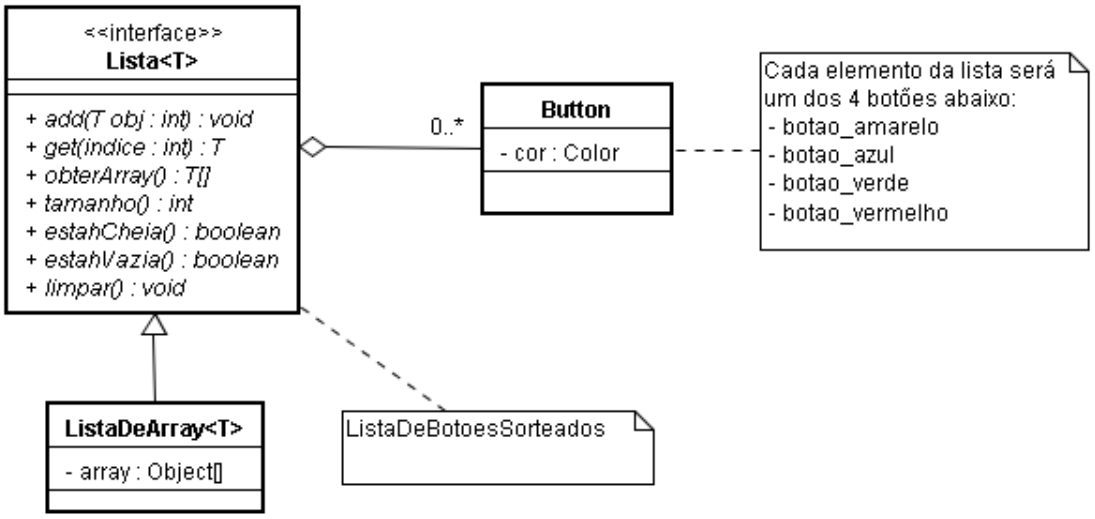

Figura 26 - Modelo UML do aplicativo Simon. [40]

4.2.4 Passos a serem executados no processo de ensino aprendizagem

É sugerido fornecer para os alunos um método para apresentar a sequência de botões que o jogador deverá clicar. Como exemplo temos o "void piscarBotoes(Button[] arrayBotoes, ConfiguracaoPisqueBotao configuracaoPisqueBotao)" da classe *ButtonUtils* do projeto SimonSays, presente no Apêndice III. Este método tem alta complexidade e podem haver grandes dificuldades em resolvê-lo, como será discutido no próximo tópico deste capítulo.

4.2.5 Dificuldades encontradas

Na implementação, inicialmente, foi utilizado para piscar a sequência de botões um método que percorria a lista dos botões, destacando a cor do mesmo, esperando e voltando à cor original. Porém essa abordagem não funcionou, uma vez que o método setColor("") da classe Button faz a cor do botão mudar somente ao final do método chamado pela tela. Logo é necessário o uso de chamadas a métodos assíncronos. Assim chamando o método para destacar a cor do botão e logo após um método assíncrono que vai, depois de um tempo, voltar a cor do botão à cor original, destacar o próximo botão

[40]Fonte: produzido pelos autores

e chamar o mesmo método assíncrono para os botões posteriores (ver Figura 27 e Figura 28).

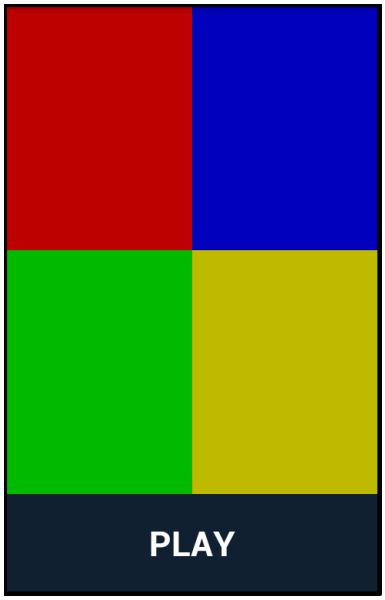

Figura 27 – Simon: Tela inicial do aplicativo. [41]

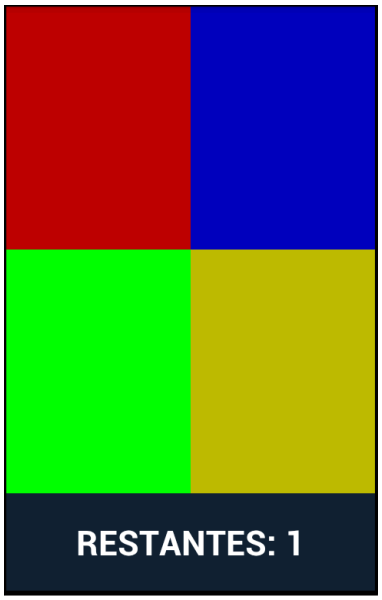

Figura 28 – Simon: Tela do aplicativo demonstrando cor evidente. [42]

Essa dificuldade pode estar além da que deve ser apresentada aos alunos dessa disciplina, então é recomendado que este método seja fornecido aos alunos no pacote para solução do problema.

[41]Fonte: *print screen* da aplicação
[42]Fonte: *print screen* da aplicação

Em relação à implementação pelo aluno, vale ressaltar que estes devem estar atentos à diferença entre a estrutura que armazena os 4 botões da tela (se forem colocados em uma lista ou array) e a estrutura que armazena os botões sorteados que serão destacados em sequência para o jogador.

4.3 Projeto Mancala

4.3.1 Enunciado

O Mancala é considerado uns dos jogos mais antigos da história e tem origem na África. Diferente de um jogo de xadrez, por exemplo, em que a movimentação de peça se faz uma a uma por vez do jogador, no Mancala diversas peças podem ser movidas ao mesmo tempo em cada vez do jogador. Normalmente tendo como peças utilizadas sementes ou pedras [34], a forma de jogar o Mancala se dá da seguinte maneira:

1. O jogo deve ser jogado por duas pessoas.
2. Antes de iniciar o jogo, deve-se distribuir as sementes igualmente entre as cavidades de cada jogador. Sendo assim, cada um começará com uma fileira de 6 cavidades, possuindo 4 sementes em cada cavidade. Além da sua fileira de cavidades, cada jogador conta com um repositório para guardar as sementes que capturou do seu lado direito do tabuleiro.
3. A movimentação no jogo tem um sentido de "colheita" e "semeadura". Na vez de cada jogador, este deve escolher uma das suas cavidades que contenha semente(s), pegar todas as sementes que há nela (colheita) e distribuir uma a uma (semeadura), no sentido anti-horário, entre as demais cavidades a partir da cavidade à direita da cavidade que ele escolheu.
4. Na semeadura, após colocar uma semente na sua última cavidade da direita, o próximo lugar a semear é em seu repositório e depois na primeira cavidade da esquerda do adversário e continua semeando no sentido anti-horário conforme houver sementes. Mas quando chegar a vez de semear no repositório do oponente, pule ele e prossiga semeando normalmente.
5. Após a colheita e semeadura, passa para a vez do outro jogador.
6. Condição especial 1: Se a última semente que o jogador semear for semeada em seu próprio repositório, o jogador pode jogar novamente.
7. Condição especial 2: Se a última semente que o jogador semear for semeada em uma cavidade sua que estiver vazia, pegue para o seu repositório todas as

sementes da cavidade do oponente localizada em frente a essa cavidade (captura).

8. O jogo termina quando um dos jogadores não tiver mais sementes em suas cavidades. Com isso as sementes das cavidades de cada jogador vão para o repositório deles, e vence quem tiver o maior número de sementes em seu repositório.

Existem diversas regras diferentes para o jogo do Mancala. A implementação proposta para o jogo seguiram as regras descritas, mas este pode seguir outras regras. Geralmente o tabuleiro é o mesmo, o que muda é a lógica implementada no *back-end*.

4.3.2 Objetivos

Exemplo de uso de **herança** e **composições**, onde cada Cavidade e Repositório do jogo herdam de uma superclasse "LocalDeSementes". Esta por sua vez se comporta como um **Façade**[43] (**design pattern**) incorporando um botão da tela e utilizando somente alguns métodos deste, interessantes para o desenvolvimento no contexto deste jogo.

Incentivo à forte **modularização** do sistema, dividindo a lógica da aplicação em vários métodos e atribuindo mais responsabilidades a outras classes que não sejam a classe central da tela da aplicação (JogoActivity). Pois o jogo tem uma complexidade de implementação maior do que as outras aplicações apresentadas, porém permite bastante **reuso** dos métodos. Com isso, se o aluno utilizar uma programação estruturada e não-modularizada, a lógica do jogo ficará muito complexa, além de haver muito código repetido.

[43]https://pt.wikipedia.org/wiki/Fa%C3%A7ade (acessado em 20/12/2015)

4.3.3 Modelo UML

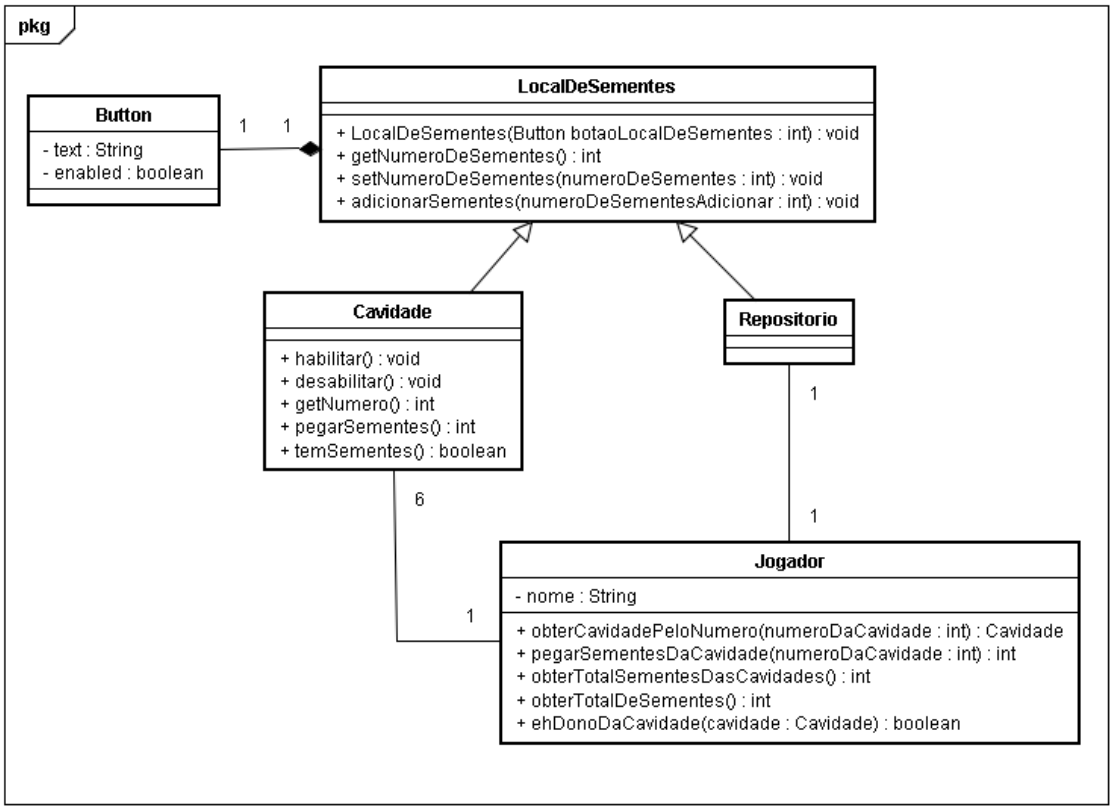

Figura 29 - Modelo UML do aplicativo Mancala.[44]

4.3.4 Passos a serem executados no processo de ensino aprendizagem

Fica à decisão do professor também fornecer o xml do layout do jogo e deixando para os alunos a parte de desenvolver o *back-end* somente, montando a estrutura das classes e programando a lógica do jogo. O xml do layout encontra-se no Apêndice IV

4.3.5 Armadilhas e dificuldades encontradas

Na elaboração da tela do tabuleiro, inicialmente foi feita pegando 20% de cada lateral da tela para serem os repositórios de cada jogador e nos 60% centrais restantes seriam distribuídas as cavidades dos jogadores em cima e em baixo, e no meio teria informações de controle durante o jogo. Acontece que disponibilizando 60% da largura

[44]Fonte: produzido pelos autores

da tela de um dispositivo mobile para apresentar 6 cavidades, cada uma fica muito pequena, e com isso fica difícil para enxergar a quantidade de sementes nela.

Foram analisadas outras interfaces de tabuleiro de Mancala para mobile e, a partir disso, proposto um novo design, que agora aproveita 100% do espaço lateral para disponibilizar as 6 cavidades de cada jogador, e os repositórios não ocupam mais tanto espaço desnecessariamente, aproveitando melhor o espaço total da tela (ver Figura 30).

Figura 30 – Mancala: Tela do tabuleiro [45]

Outro ponto foi que a princípio a idéia era fazer a classe "LocalDeSementes" herdando de "*Button*". Todavia, quando é declarado no xml do layout da tela um elemento "*Button*", o Android instancia um elemento da classe *Button* e o relaciona a este elemento da tela. Dessa forma, não é possível guardá-lo em um atributo do tipo "LocalDeSementes" (pois nem todo *Button* é um LocalDeSementes).

Para fazer isso, teria que ser criado um tipo específico de elemento da tela (*Custom View*[46]) e associado a classe desejada. Porém este artifício aumentaria a complexidade do sistema sem muito ganho, uma vez que nada ensinaria de programação OO e é algo específico do Android. Com isso foi decidido que seria melhor incorporar o botão dentro da classe "LocalDeSementes" e manipulá-lo internamente.

[45]Fonte: *print screen* da aplicação
[46]http://developer.android.com/intl/pt-br/training/custom-views/index.html (acessado em 21/12/2015)

4.4 Projeto Controle Financeiro

4.4.1 Enunciado

Ter um controle das suas finanças é essencial para quem quer poupar, quitar suas dívidas ou evitar se endividar. Muitas pessoas fazem isso tendo pouco controle real, guardando de cabeça o saldo atual de sua conta, o que ainda precisa pagar no mês, etc. Entretanto este controle está suscetível ao esquecimento gerando o descontrole. Outras pessoas, para terem melhor administração de seus custos, anotam no papel, diminuindo o risco de esquecerem alguma informação. Mas isso demanda mais tempo, gera tarefas repetitivas e continua possibilitando erro de cálculos.

Como uma ideia geral do projeto o sistema gerencia as movimentações financeiras, que devem ser agrupadas por mês. Com isso haveria uma listagem das movimentações (do mês) e um total com a soma delas. As movimentações podem representar tanto entrada em caixa (crédito) como saída (débito), e podem ser de 3 tipos:

- Movimentação Variável: É simples, única e independente. Ex.: um lanche, cinema, roupa comprada.
- Movimentação Fixa: Representa uma movimentação que é mensal, ou seja, todo mês o usuário tem aquela mesma movimentação em sua conta. Ex.: Salário, aluguel, conta de internet, etc. Quando é cadastrada em um mês, ela será criada automaticamente ao abrir um mês posterior a este com a mesma descrição, dia de ocorrência e valor. Para parar a criação automática dessa movimentação deve-se entrar no último mês que ela ainda deve ocorrer, marcar o indicativo que este é o último mês de ocorrência.
- Movimentação Parcelada: Representa uma movimentação que sua quitação (em caso de dívida) ou recebimento (em caso de crédito) foi dividido em parcelas. Ex.: Compra de uma televisão dividida em 10 parcelas/prestações. Quando é cadastrada em um mês, se o número da parcela corrente for menor que o total de parcelas informado, ela será criada automaticamente ao abrir um mês posterior a este com a mesma descrição, dia de ocorrência, valor total e total de parcelas, e com a parcela corrente igual a N+1 (sendo N a parcela corrente do mês anterior).

Conforme ilustra a Figura 31, na primeira tela são apresentados os meses que o usuário pode ter movimentações cadastradas, a opção de adicionar ou remover último

mês, assim como de entrar em um mês selecionado para ver os detalhes de suas movimentações nele.

Figura 31 - Controle Financeiro: Tela inicial.[47]

Escolhendo um mês da lista e entrando nele, deverá aparecer uma tela semelhante a da Figura 32, com a lista de movimentações (positivas ou negativas) cadastradas pelo usuário (se houverem), o total do mês, que representa a soma das movimentações listadas, e botões para incluir uma nova movimentação de crédito (positiva) ou de débito (negativa).

[47]Fonte: *print screen* da aplicação

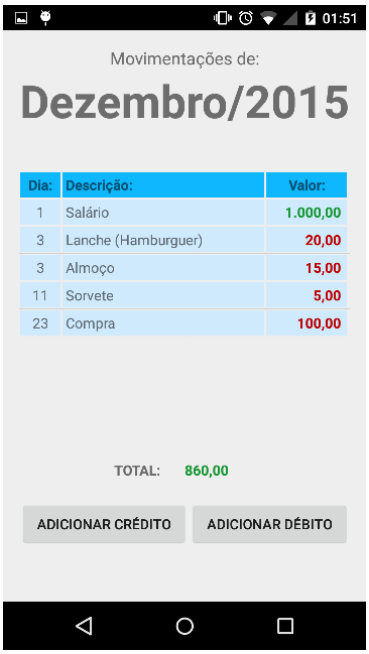

Figura 32 – Controle Financeiro: Tela de detalhes das movimentações de um mês.[48]

Escolhendo a opção de adicionar uma nova movimentação, abrirá uma nova tela com um formulário a ser preenchido pelo usuário, em que ele deverá informar o tipo da movimentação (Variável, Fixa ou Parcelada), uma descrição, o dia do mês em que foi feita, o valor a ser creditado ou debitado, e um indicativo se a movimentação já foi efetuada.

Se a movimentação for do tipo “Variável”, estes campos bastarão para realizar o seu cadastro (Figura 33). Se o tipo for “Fixa”, aparecerá também um campo em que o usuário poderá informar se essa é a última ocorrência dessa sequência de movimentações (Figura 34). Já se for do tipo “Parcelada” o campo com o valor representará o valor total do produto, que deverá ser a soma das parcelas, e aparecerão campos para o usuário informar o total de parcelas e a parcela corrente daquele mês (Figura 35).

[48]Fonte: *print screen* da aplicação

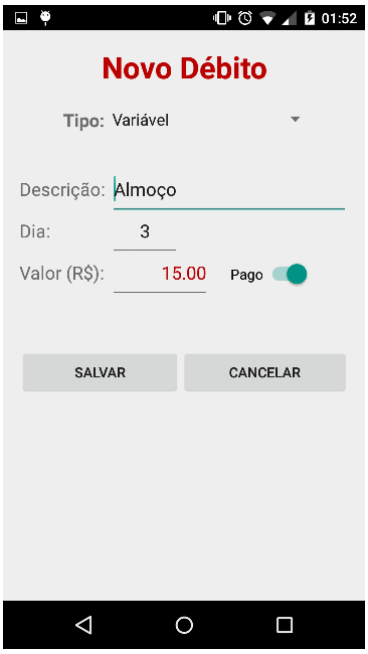

Figura 33 – Controle Financeiro: Tela de cadastro de movimentação do tipo Variável.[49]

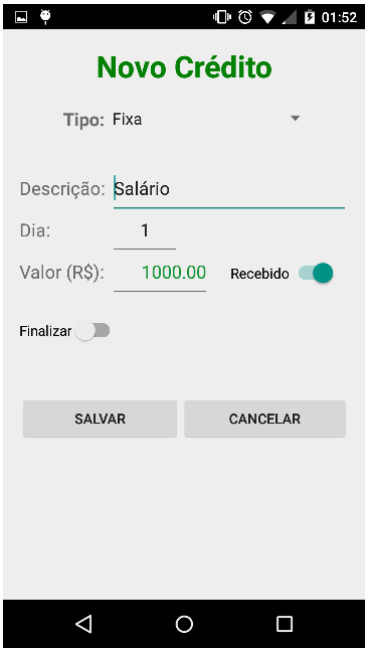

Figura 34 - Controle Financeiro: Tela de cadastro de movimentação do tipo Fixa.[50]

[49] Fonte: *print screen* da aplicação
[50] Fonte: *print screen* da aplicação

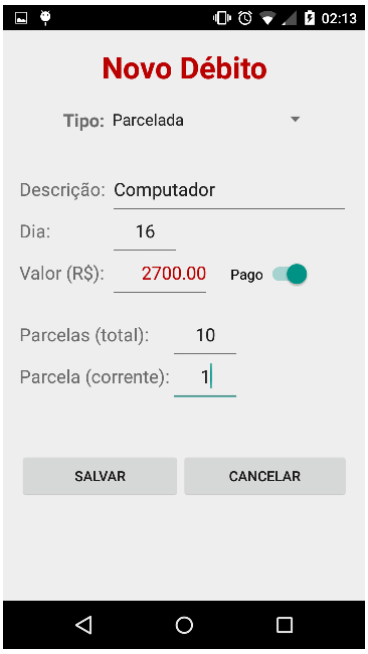

Figura 35 - Controle Financeiro: Tela de cadastro de movimentação do tipo Parcelada.[51]

4.4.2 Objetivos

Este projeto é indicado para o ensino da utilização de **herança**, no exemplo dado é utilizado na classe "Movimentação", e **polimorfismo,** para mostrar a lista dos diferentes tipos de movimentações. É trabalhado também o conceito de **sobrescrita de método**, já que as subclasses da classe Movimentação deverão obrigatoriamente sobrescrever o método "public BigDecimal getValor()" desta, uma vez que este é um método abstrato.

Há também a utilização do conceito de **lista**, para a listagem dos meses que possuem movimentações e também das próprias movimentações do mês. Essa lista pode ser implementada pelo aluno, ou utilizada a "*java.util.List*", da biblioteca padrão do Java.

É feita também a utilização de **interface**, que é um conceito que induz bastante a abstração no código. Isso pode ser utilizado nas classes de persistência (no caso de nosso projeto, as classes do pacote "*br.com.uniriotec.controlefinanceiro.dao*"), onde a interface "*MovimentacaoDao"* tem métodos definidos que serão utilizados, mas que podem ser implementados por uma classe que grave e atualize os dados em memória (que é perdida quando a aplicação é fechada), em arquivo(s) ou em um banco de dados, por exemplo. Esse conceito também pode ser aplicado nas listas, que podem ser

[51]Fonte: *print screen* da aplicação

implementadas de diferentes formas e ter o mesmo funcionamento (de uma visão externa), as implementações mais utilizadas são as **listas utilizando array** (*java.util.ArrayList*) e as **listas encadeadas** por ponteiro (*java.util.LinkedList*).

4.4.3 Modelo UML

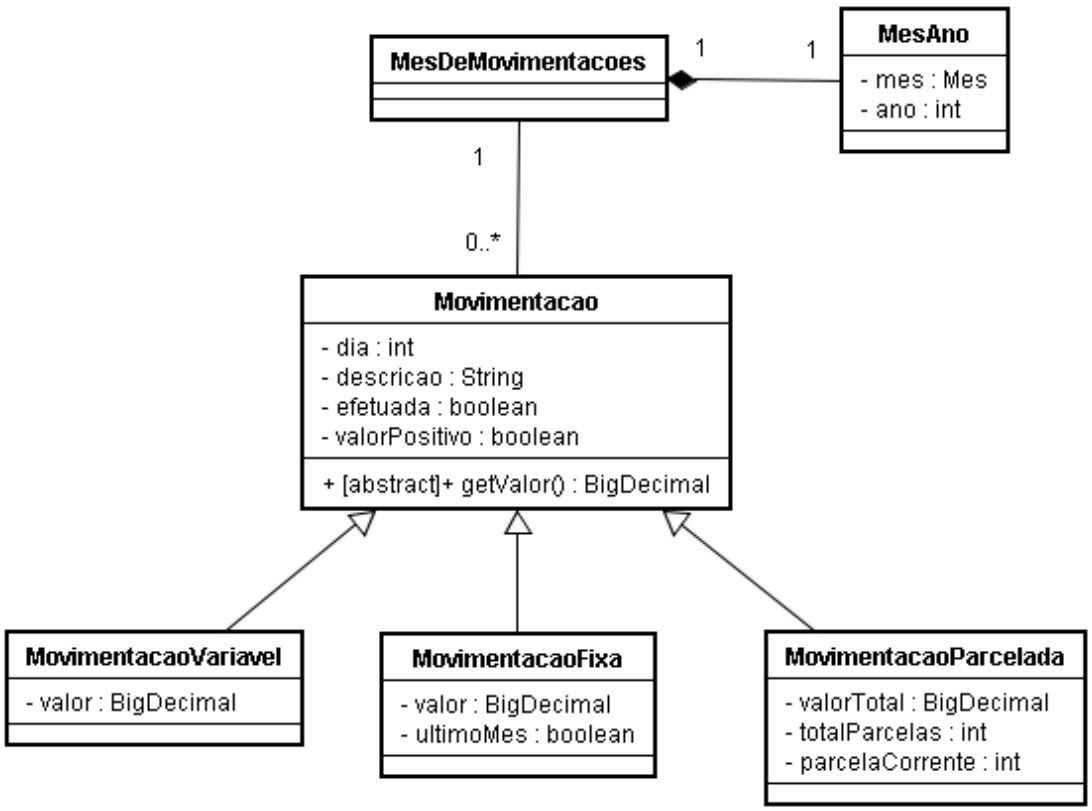

Figura 36 - Modelo UML do aplicativo Controle Financeiro. [52]

4.4.4 Passos a serem executados no processo de ensino aprendizagem

Caso não seja interesse do tutor que o aluno faça uma classe para representar uma lista, é interessante que seja mostrado a ele o uso da classe *List*, da biblioteca *java.util*. Em comparação com um array, a sua utilização é muito facilitada por não ter de se preocupar com redimensionamento de array ao adicionar itens ou contagem de casas preenchidas do array.

Como este projeto tem mais de uma tela, é importante que o aluno saiba como fazer a transição de uma tela para outra, como transportar dados ao chamar a tela e como fechar uma tela. Também será muito útil uma explicação sobre o ciclo de vida de uma

[52]Fonte: produzido pelos autores

activity[53] (veja o esquema na Figura 37), para ter conhecimento dos métodos da classe Activity que o Android chama em cada "momento" da activity. Neste projeto, além do método da activity "*onCreate()*" para carregar as informações quando a activity era invocada pela primeira vez, foi necessário também utilizar o método "*onResume()*", que é chamado sempre que a activity volta para o primeiro plano, para atualizar a lista de movimentações do mês quando volta da tela de cadastro ou alteração de movimentação.

[53]http://developer.android.com/intl/pt-br/guide/components/activities.html (acessado em 20/12/2015)

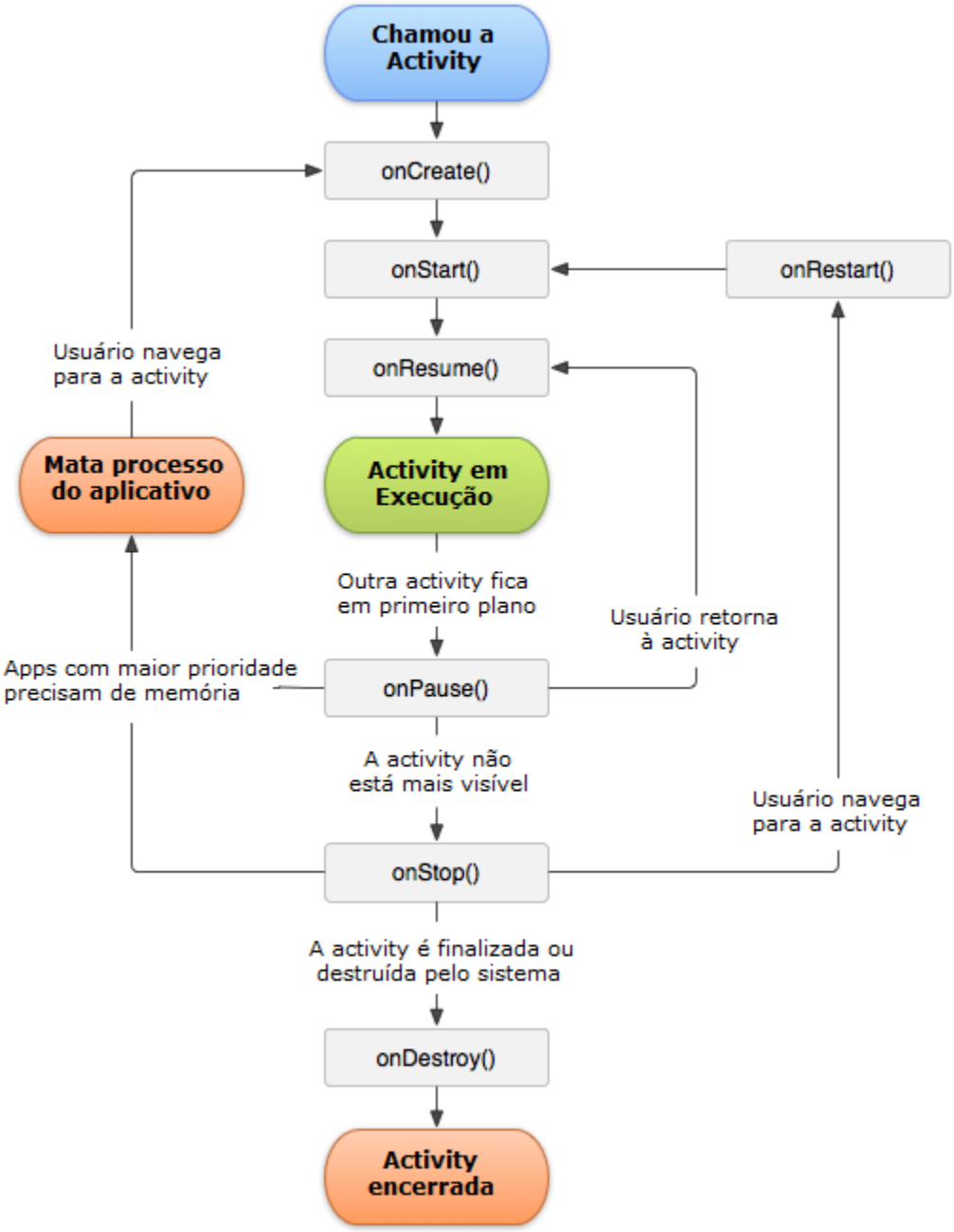

Figura 37 - Ciclo de vida de uma activity (imagem traduzida).[54]

4.4.5 Armadilhas e dificuldades encontradas

Neste projeto foram utilizados vários elementos de tela que ainda, nós autores, não tínhamos visto como era utilizado, como o *Switch*[55] (ToggleButton nas versões anteriores ao Android 4), que serve como um indicativo de valor booleano (Figura 38), o *Spinner*[56], também chamado de combobox, que apresenta uma lista de opções e

[54] Fonte: <http://developer.android.com/intl/pt-br/guide/components/activities.html>Acessado em 20/12/2015).
[55] http://developer.android.com/intl/pt-br/guide/topics/ui/controls/togglebutton.html (acessado 20/12/2015)
[56] http://developer.android.com/intl/pt-br/guide/topics/ui/controls/spinner.html (acessado 20/12/2015)

permite o usuário selecionar uma delas (Figura 39) e o ListView[57], que apresenta na tela uma lista de itens (Figura 40).

Figura 38 - Exemplo de Switch.

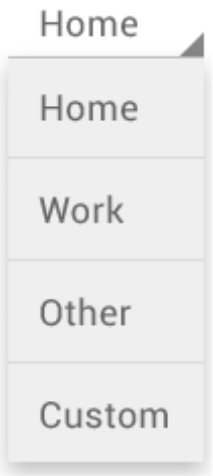

Figura 39 - Exemplo de Spinner.

[57] http://developer.android.com/intl/pt-br/guide/topics/ui/layout/listview.html (acessado 20/12/2015)

Figura 40 - Exemplo de ListView.

Sugerimos que seja explicado o uso destes novos elementos. Principalmente o Spinner e o ListView, que tem uso mais complexo por manusear uma lista de elementos.

No mais, não foram encontradas grandes dificuldades. Apesar de este ser um projeto com mais telas e funcionalidades, a complexidade de suas operações são menores individualmente.

4.5 Extensões

Nesta seção, serão apresentadas algumas alternativas de projetos em relação aos que já foram apresentados anteriormente neste capítulo, tendo assim mais opções a serem escolhidas entre os alunos. A ideia é que o aluno possa escolher o que quer implementar para trabalhar sem fugir aos objetivos a serem alcançados.

4.5.1 Extensões do Jogo da Velha

4.5.1.1 *Five in a Row*

Similar ao Jogo da Velha, este jogo ao invés de possuir 9 casas em uma disposição tabular 3x3 (3 linhas por 3 colunas), conta com, no mínimo, 25 casas em disposição

tabular 5x5 (5 linhas por 5 colunas) e a condição de vitória é completar 5 casas enfileiradas na horizontal, vertical ou diagonal (ver Figura 41).

Do mesmo modo que o jogo da velha tradicional, a implementação deste jogo pode trabalhar conceitos de classes (incluindo construtores, atributos e métodos), matriz e acesso a endereço de memória.

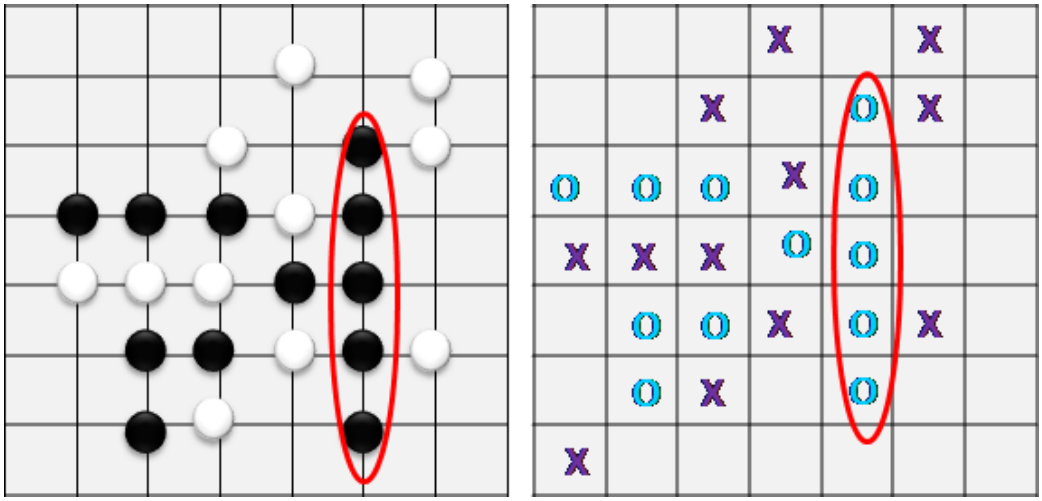

Figura 41 - Do lado esquerdo o jogo Five in a Row, do lado direito um jogo da velha em uma matriz maior. [58]

4.5.1.2 *Sudoku*

O Sudoku conta com um tabuleiro de 81 casas dispostas em uma matriz 9x9. Além disso é dividido em 9 quadrantes de tamanho 3x3, como apresentado na Figura 42. Essas casas podem ser preenchidas com números de 1 a 9. Quando inicia o jogo, algumas casas já vêm preenchidas, não sendo possível mudar seus valores, e as demais devem ser preenchidas pelo jogador. O objetivo é completar todas as casas não deixando nenhum número repetido em nenhuma vertical, horizontal ou quadrante do tabuleiro.

[58] Fonte: produzido pelos autores

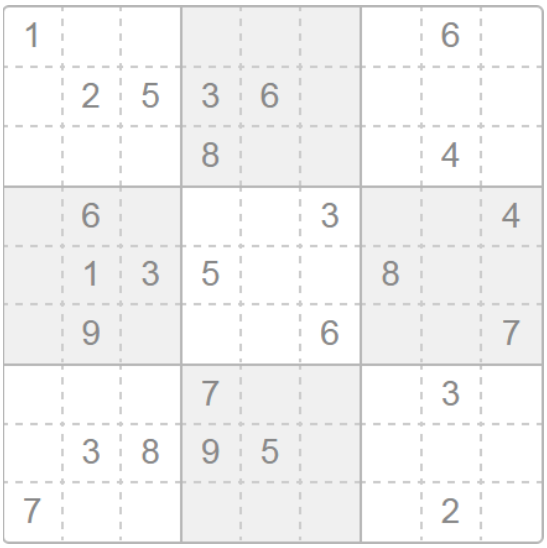

Figura 42 - Tabuleiro Sudoku[59]

No desenvolvimento deste jogo, o aluno deverá trabalhar conceitos como de **matriz** e **acesso a endereço de memória**.

4.5.1.3 ***Resta Um***

O tabuleiro pode ter diversas disposições de representação, a mais tradicional é vista na Figura 43. Na imagem, a bolinha branca ao centro representa uma casa vazia. Já as bolinhas azuis representam casas ocupadas com peças do jogo. A movimentação no jogo se dá de forma que é possível, com uma peça A, "saltar" sobre outra peça B ao lado desta (na horizontal ou vertical), se a próxima casa (em que a peça que "salta" vai "cair") estiver vazia. Com isso a peça B é eliminada. Não é possível "saltar" sobre mais de uma peça por vez.

O objetivo é terminar as possibilidades de mexidas restando apenas 1 (uma) peça no tabuleiro (ou o mínimo possível).

[59] Fonte: <https://www.geniol.com.br/logica/sudoku/> acessado em 19/01/2016

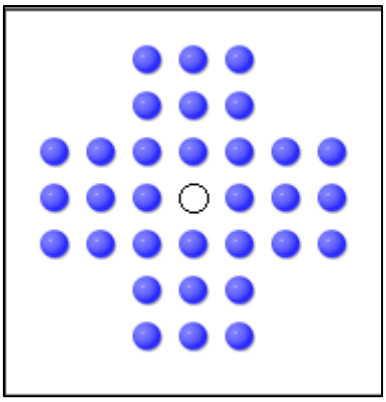

Figura 43 - Tabuleiro Resta Um [60]

Com este jogo é possível trabalhar principalmente os conceitos de **matrizes**.

4.5.2 Extensões do Simon

4.5.2.1 *Snake*

Neste jogo há uma matriz de campos na tela representando uma espécie de "campo", onde uma cobra, representada inicialmente por 3 campos e controlada pelo jogador, pode trafegar. Neste campo aparecem comidas que se deve pegar para aumentar seu tamanho. O objetivo é ficar do maior tamanho possível sem colidir com nenhum dos cantos deste campo e não atravessar a si próprio, "comendo" a própria cauda (ver Figura 44).

[60] Fonte: < http://www.divertudo.com.br/restaum/restaum.html> acessado em 19/01/2016.

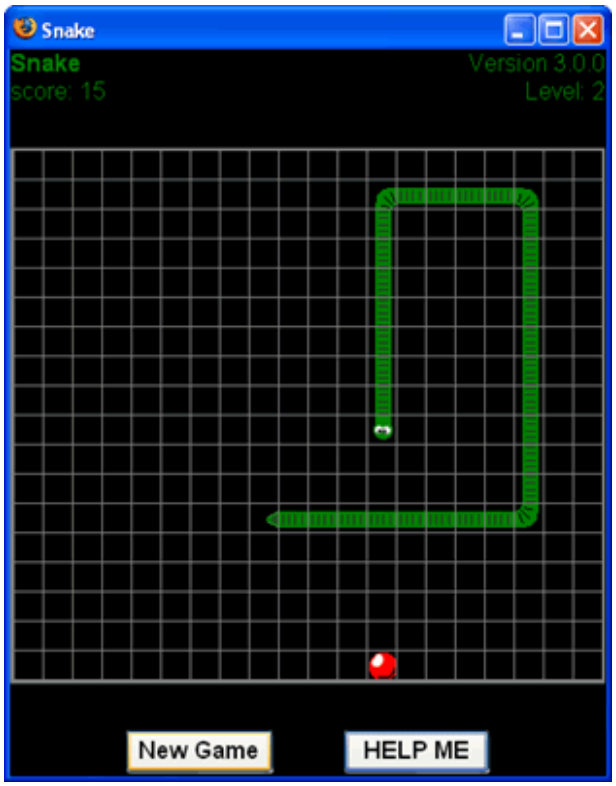

Figura 44 - Tela Jogo Snake[61]

Desenvolvendo este jogo, além de **matriz**, o aluno vai utilizar o **conceito de lista,** crescente (que no caso representa a cobra que cresce a cada alimento comido).

4.5.3 Extensões do Mancala

4.5.3.1 *Jogo da Memória*

Este jogo é baseado na capacidade de memorização do jogador. Na tela são apresentadas diversas cartas com imagens, sendo que cada carta está duplicada, ou seja, tem uma cópia naquele baralho. Depois de um curto tempo as cartas são "viradas para baixo", mostrando somente suas "costas", que são todas iguais (ver Figura 45). Em cada jogada o jogador tem direito a virar 2 cartas, e, se estas forem iguais, retirar elas do tabuleiro. O objetivo é retirar todas as cartas do tabuleiro em menos jogadas ou em menos tempo possível.

[61] Fonte: < http://br.mozdev.org/firefox/vocesabia/jogos-snake.png> acessado em 20/01/2016

Figura 45 - Jogo da Memória[62]

O desenvolvimento da aplicação deve trabalhar o uso de **matriz** e, se for feita implementação com níveis de dificuldade diferentes, aumentando o número de cartas no jogo, pode ser trabalhado também conceito de **herança** ou **interface** também.

4.5.3.2 *Campo Minado*

Um jogo de raciocínio lógico e que conta um pouco com a sorte também. É apresentada uma matriz de campos na tela que a princípio estão ocultos. A jogada se dá clicando nesses campos para revelar o que eles contêm (ver Figura 46). Que pode ser de três tipos:

- Uma bomba: se o campo contiver uma bomba o jogador perde o jogo.
- Um campo com um número: o número informado nesses campos indica a quantidade exata de bombas contidas nos campos adjacentes (horizontal, vertical e diagonalmente).
- Um campo vazio: indica que não há nenhuma bomba perto deste campo.

O objetivo do jogo é revelar todos os campos não-bomba sem explodir (clicar sobre) nenhuma bomba.

[62] Fonte: Aplicação desenvolvida por Jefferson Ferrão em conjunto com os autores deste trabalho

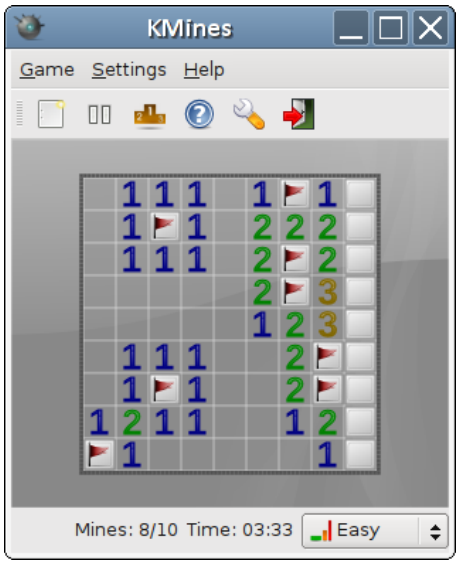

Figura 46 - Tela Campo Minado[63]

Trabalha os conceitos de **matriz**. Como são vários botões e cada um pode ser de três tipos diferentes, pode ser utilizado o *design pattern* **Façade** para facilitar o uso e manutenção destes botões e **herança**, para subdividir os botões em um dos 3 tipos. E assim como o *Jogo da Memória*, ele pode tabuleiros de tamanhos diferentes (que implica em dificuldades diferentes) e com isso usar conceitos de herança ou **interface** também.

4.5.3.3 *Tchuka*

Este jogo é uma variação do jogo Mancala, porém é jogado por um jogador somente. O tabuleiro deve ter 4 cavidades e um repositório (chamado de "*Ruma*") como ilustrado na Figura 47. Ao iniciar o jogo coloca-se 2 sementes em cada cavidade. A movimentação do jogo funciona como colheita e semeadura, ou seja, escolhe-se uma cavidade com sementes, pega-se as sementes dessa cavidade e distribui uma a uma nos locais a direita desta. Se ainda existirem sementes a serem semeadas, estas voltam a ser distribuídas a partir da primeira cavidade da esquerda. [34]

A partir disso existem três situações:

- A última semente é semeada na Ruma: O jogador joga novamente.
- A última semente é semeada em uma cavidade vazia: O jogador perdeu.
- A última semente é semeada em uma cavidade ocupada: O jogador deve colher as sementes dessa cavidade e reiniciar a semeadura.

[63] Fonte: <https://pt.wikipedia.org/wiki/Campo_minado#/media/File:KMines.png> acessado em 20/01/2016

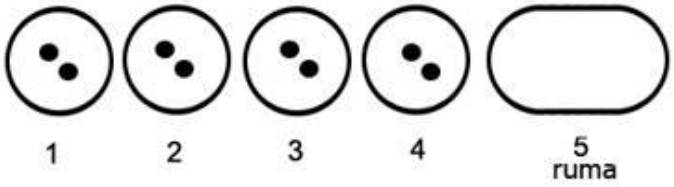

Figura 47 - Tabuleiro do jogo Tchuka[64]

Assim como o Mancala, este jogo também pode usar uma classe para incorporar cada botão, para facilitar o seu uso, fazendo assim uso do design pattern "**Façade**". E, além disso, o uso de **herança** para criar uma classe base às classes de Cavidade e Repositório (Ruma), que contenha o comportamento que é comum às duas.

4.5.4 Extensões do Controle Financeiro

O projeto Controle Financeiro apresentado na seção anterior é bem genérico, podendo ser proposto ao aluno até mesmo um sistema com um "tema" diferente. Com algum outro cadastro qualquer, como de séries acompanhadas, filmes que serão lançados, agenda de compromissos, entre outros assuntos que o aluno considere interessante de trabalhar.

Com isso o aluno já deve trabalhar o conceito de **lista** para guardar os itens cadastrados e de **interface**, para deixar genérica a implementação da persistência de dados e da própria lista de itens cadastrados. É interessante que os itens possam ser de tipos diferentes e tenham algum comportamento especial no sistema, para o aluno poder trabalhar também conceitos de **herança**, **polimorfismo** e **sobrescrita de métodos**.

Além disso, pode-se trabalhar:

- Muitas ou poucas funcionalidades;
- Persistência dos dados em memória, arquivo ou banco de dados;
- Preocupação com interface do usuário; e
- Maior organização, modularização ou desempenho.

---

64 Fonte: < http://www.ilhadotabuleiro.com.br//imagem_thumbs.php?src=/upload/jogo/imagem/normal/14147.jpg> acessado em 20/01/2016

5 Conclusão

5.1 Considerações finais

Após os estudos apresentados neste trabalho acerca da dificuldade de aprendizado e também das metodologias comumente usadas no ensino, percebeu-se que há uma necessidade de mudança na forma como alguns conceitos são tratados nas disciplinas.

A proposta de uma abordagem baseada em desenvolvimento para a plataforma Android, neste contexto, se torna algo relevante a ser utilizado e experimentado com alunos, porém as aplicações apresentadas neste trabalho ainda não foram aplicadas. Os aplicativos desenvolvidos encontram-se, em sua versão final, disponíveis em <https://github.com/ProjetoTCC2015/aplicativos_desenvolvidos> e seguindo as instruções do Apêndice I é possível executá-las em seu *smartphone* Android. É interessante que com essa proposta o aluno possa chegar neste nível de finalização, onde seja possível utilizar seu próprio aplicativo e compartilhá-lo caso queira, assim proporcionando uma satisfação maior com o que ele próprio produziu.

Por fim, vale a pena ressaltar que convergindo a atenção do estudante para algo mais "agradável" como o desenvolvimento de jogos e utilizando uma tecnologia atual como o Android, ele se sentirá mais motivado nesse processo de ensino-aprendizagem.

A implementação de aplicações para plataforma móvel proporciona ao discente uma oportunidade de capacitar-se mais para o mercado de trabalho, onde pode-se ver que a área mobile anda ganhando cada vez mais espaço. Nós autores acreditamos que

cativando os alunos desta forma é possível também reter um pouco a evasão que ocorre em cursos da área de computação, o que seria um ganho significativo.

5.2 Trabalhos futuros

A aplicação desta abordagem em alguma turma é um trabalho futuro a ser considerado, assim será possível testar a eficácia da proposta de acordo com o desempenho dos alunos. Espera-se que com isso possa influenciar positivamente a taxa de desistência de disciplinas que possuam o conteúdo apresentado neste projeto, através do estímulo de interesse ao aluno aprender e, quem sabe, gostar da área de desenvolvimento.

Também seria interessante motivar a expansão desta abordagem a outras áreas de disciplinas como Banco de Dados ou Redes. Os alunos poderiam assim utilizar até mesmo projetos já criados por eles em disciplinas anteriores, e aprimorarem em outras áreas de acordo com a proposta dos professores. No mercado de trabalho algumas vezes têm-se que lidar com conhecimentos de diferentes áreas de TI para a resolução das atividades. Proporcionar essa experiência tornando os projetos propostos interdisciplinares seria uma ideia relevante a se por em prática.

Referências Bibliográficas

[1] Borges, Marcos AF. (2000) Avaliação de uma metodologia alternativa para a aprendizagem de programação. In: VIII Workshop de Educação em Computação–WEI.

[2] Rapkiewicz, Clevi Elena et al. (2006) "Estratégias pedagógicas no ensino de algoritmos e programação associadas ao uso de jogos educacionais", http://hdl.handle.net/10183/22862, Acessado em 20/12/2015.

[3] Gomes, A.; Henriques, J., Mendes, A.J. (2008) "Uma proposta para ajudar alunos com dificuldades na aprendizagem inicial de programação de computadores", http://eft.educom.pt/index.php/eft/article/view/23, Acessado em 20/12/2015.

[4] Bergin, S.; Reilly, R. (2005) The influence of motivation and comfort-level on learning to program. In Proceedings of 17th Workshop of the Psychology of Programming Interest Group (PPIG 2005), p. 293-304. Brighton, UK.

[5] Mahmoud, Q. H.; Dobosiewicz, W.; Swayne, D. (2004) Redesigning introductory computer programming with HTML, JavaScript, and Java. In Proceedings of the 35th Technical Symposium on Computer Science Education (SIGCSE 2004), p. 120-124.Norfolk, Virginia, USA.

[6] JENKINS, T. (2002). On the difficulty of learning to program.In Proceedings of 3rd Annual LTSN_ICS Conference (Loughborough University, United Kingdom, August 27-29, 2002).The Higher Education Academy, p.53-58.

[7] Mendes de Oliveira, Wilandia, "Uma Abordagem sobre o papel do professor no processo ensino/aprendizagem" <https://www.inesul.edu.br/revista/arquivos/arq-idvol_28_1391209402.pdf> Acessado em 20/01/2016.

[8] Berbel, N. A. N. (1998). A problematização e a aprendizagem baseada em problemas. Interface Comun Saúde Educ, 2(2), 139-154, http://www.scielo.br/pdf/icse/v2n2/08, Acessado em 20/12/2015.

[9] Wikipédia - Aprendizagem Baseada em Problemas <https://pt.wikipedia.org/wiki/Aprendizagem_baseada_em_problemas> Acessado em 14/12/2015.

[10] UNIFESP <http://www2.unifesp.br/centros/cedess/pbl/> Acessado em 14/12/2015.

[11] Nobre, J. C. S., Loubach, D. S., Cunha, A. M., & Dias, L. A. V. (2006). Aprendizagem Baseada em Projeto (Project-Based Learning–PBL) aplicada a software embarcado e de tempo real. In SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO (SBIE) UNB/UCB, 17, 258.

[12] Aprendizagem Baseada em Projetos <https://en.wikipedia.org/wiki/Project-based_learning> Acessado em 14/12/2015.

[13] Aureliano, V.C.O.; Tedesco, PC de AR. (2012) Avaliando o uso do Scratch como abordagem alternativa para o processo de ensino-aprendizagem de programação. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO.

[14] MÉLO, FÉN de et al. (2011) Do Scratch ao Arduino: Uma Proposta para o EnsinoIntrodutório de Programação para Cursos Superiores de Tecnologia. In: XXXIX Congresso Brasileiro de Educação em Engenharia, Blumenau, Brasil.

[15] Wikipédia - Greenfoot <https://en.wikipedia.org/wiki/Greenfoot> Acessado em 14/12/2015.

[16] Suzuki, A.P. et al. (2010) "Introdução à Programação com Robôs Lego", http://www.obr.org.br/wp-content/uploads/2014/03/apostilaprogramaorobs1-111023145650-phpapp02.pdf, Acessado em 20/12/2015.

[17] de Toledo Quadros, J. R., Ogasawara, E., dos Santos Amorim, M. C. M., & Ribeiro, R. C. (2012) Estudos sobre o Uso de Jogos para Apoiar o Aprendizado de Programação em um Curso Técnico de Informática. In Simpósio de excelência em gestão e tecnologia. http://www.aedb.br/seget/arquivos/artigos12/24816249.pdf, Acessado em 20/12/2015.

[18] da Silva, R. E.,& Martins, S. W. (2007). "Ensino de Ciência da Computação através do Desenvolvimento de Jogos." http://www.ufrgs.br/niee/eventos/RIBIE/2004/posters/poster1286-1295.pdf. Acessado em 20/12/2015.

[19] Scaico, Pasqueline Dantas et al. "Combinando Diversão e Educação: Castelo dos Enigmas, um Jogo Sério para o Ensino de Algoritmos", http://www.br-ie.org/sbie-wie2011/workshops/applets/10_95016_1.pdf, Acessado em 20/12/2015.

[20] Digiampietri, L. A., Peres, S. M., Nakano, F., Roman, N. T., Wagner, P. K., Silva, B. B. C., ... & Pereira, G. R. (2014). Complementando o Aprendizado em Programação: Revisitando Experiências no Curso de Sistemas de Informação da USP.In iSys-Revista Brasileira de Sistemas de Informação, 6, 5-29. http://www.seer.unirio.br/index.php/isys/article/view/2178, Acessado em 20/12/2015.

[21] Wikipédia - Pac-Man <https://pt.wikipedia.org/wiki/Pac-Man> Acessado em 14/12/2015.

[22] Wikipédia - Enduro <https://pt.wikipedia.org/wiki/Enduro_(jogo_eletr%C3%B4nico)> Acessado em 14/12/2015.

[23] Koster, R. (2004) Theory of fun for game design. Scottsdale: Paraglyph, p. 80-99.

[24] Lucio Camilo Oliva Pereira; Michel Lourenço da Silva (2009), Android para Desenvolvedores, Brasport.

[25] Harvey M. Deitel, Abbey Deitel, Michael Morgano (2013), Android para programadores: Uma Abordagem Baseada em Aplicativos, Bookman.

[26] Burton, Michael; Felker, Donn (2014), Desenvolvimento de Aplicativos Android para Leigos, Atlas Books, 2ª edição.

[27] Android Studio Release Notes<http://developer.android.com/intl/pt-br/tools/revisions/studio.html>Acessado em 21/12/2015.

[28] Loucos por Android <http://loucoporandroid.com/google-descontinua-suporte-ao-eclipse-para-o-desenvolvimento-android/> Acessado em 14/12/2015.

[29] Android Developers Blog <http://android-developers.blogspot.mx/2015/06/an-update-on-eclipse-android-developer.html> Acessado em 14/12/2015.

[30] Wikipédia - Gerência de configuração de software<https://pt.wikipedia.org/wiki/Gerência_de_configuração_de_software> Acessado em 21/12/2015.

[31] de Queiroz, F. B.; Ferreira, F. S.; da Silva, L.S. "Análise de usabilidade dos Sistemas de Controle de Versão Subversion e Git", http://estatistica.googlecode.com/svn-history/r88/trunk/docs/especificacao/especificacao.pdf., Acessado em 20/12/2015.

[32] Wikipédia - Jogo da Velha <https://pt.wikipedia.org/wiki/Jogo_da_velha> Acessado em 14/12/2015.

[33] Wikipédia - Genius <https://pt.wikipedia.org/wiki/Genius_(jogo)> Acessado em 14/12/2015.

[34] Jogos Antigos <http://www.jogos.antigos.nom.br/mancala.asp> Acessado em 14/12/2015.

Apêndice I - Tutorial de como rodar o projeto do Android Studio em seu dispositivo móvel Android

Passo 1. Ativando o modo desenvolvedor em seu dispositivo móvel

Para ativar o menu "Programador" nas configurações, vá em “Configurações”, depois clique em “Sobre o telefone”, como ilustrado na Figura 48.

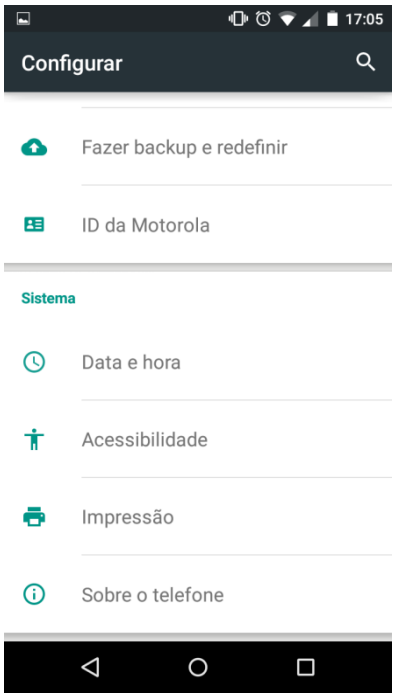

Figura 48 - Tela de configurações

Clique 7 vezes onde é indicado o "Número da versão". Com isso aparecerá uma mensagem na tela: "Você agora é um desenvolvedor" conforme mostra a Figura 49.

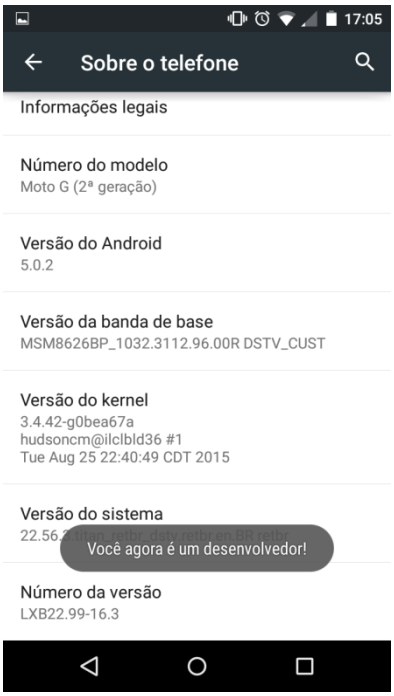

Figura 49 - Mensagem de confirmação

Agora, no menu "Configurações", aparecerá uma nova opção chamada "Programador", onde você pode ativar/desativar o modo programador (além de poder alterar diversas configurações avançadas do dispositivo que podem ser úteis para o desenvolvimento de aplicações) como mostrado a seguir na Figura 50.

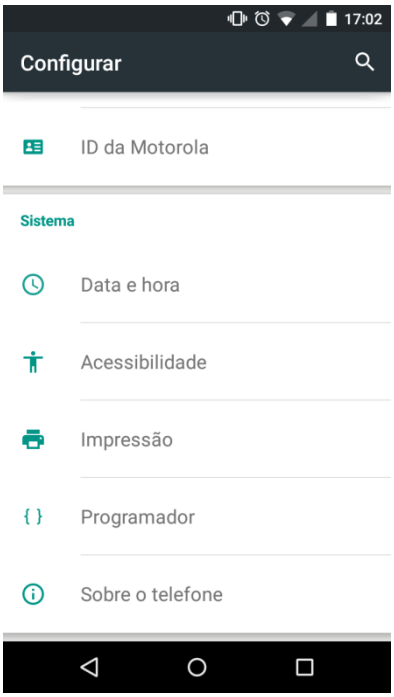

Figura 50 - Tela de configurações após executar passo 1.2

Passo 2. Ativando o modo de depuração via USB no seu dispositivo

No menu “Configurações”, na opção “Programador”, habilite o modo programador e marque a caixa "Depuração USB" conforme mostra as Figuras “Figura 51” e “Figura 52”.

Figura 51 - Tela do modo programador inicialmente

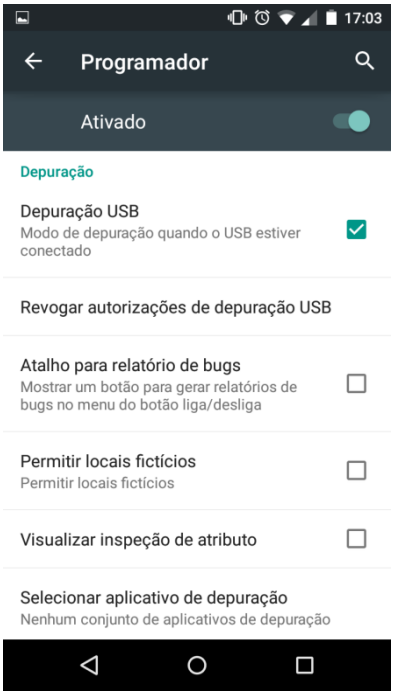

Figura 52 - Tela com modo programador ativado e depuração USB ativada

Passo 3. Instale o apk no seu aparelho

3.1 Instalando apk pelo Android Studio

Plugue o dispositivo móvel no computador e abra o Android Studio. Lembre-se de desbloquear a tela do dispositivo caso a mesma esteja bloqueada, assim permite que o computador possa reconhecer o aparelho. Confirme então a caixa de diálogo que ira aparecer: "Permitir a depuração USB?", para que o Android Studio possa usar o seu dispositivo para a depuração do projeto conforme indicado na Figura 53.

Figura 53 - Mensagem de permissão para depuração USB

Veja a Figura 54, no canto inferior da ela vá na aba "Android" em seguida na aba "Devices | logcat", verifique na área "Devices" que deverá mostrar na combobox o seu dispositivo móvel como conectado e habilitado.

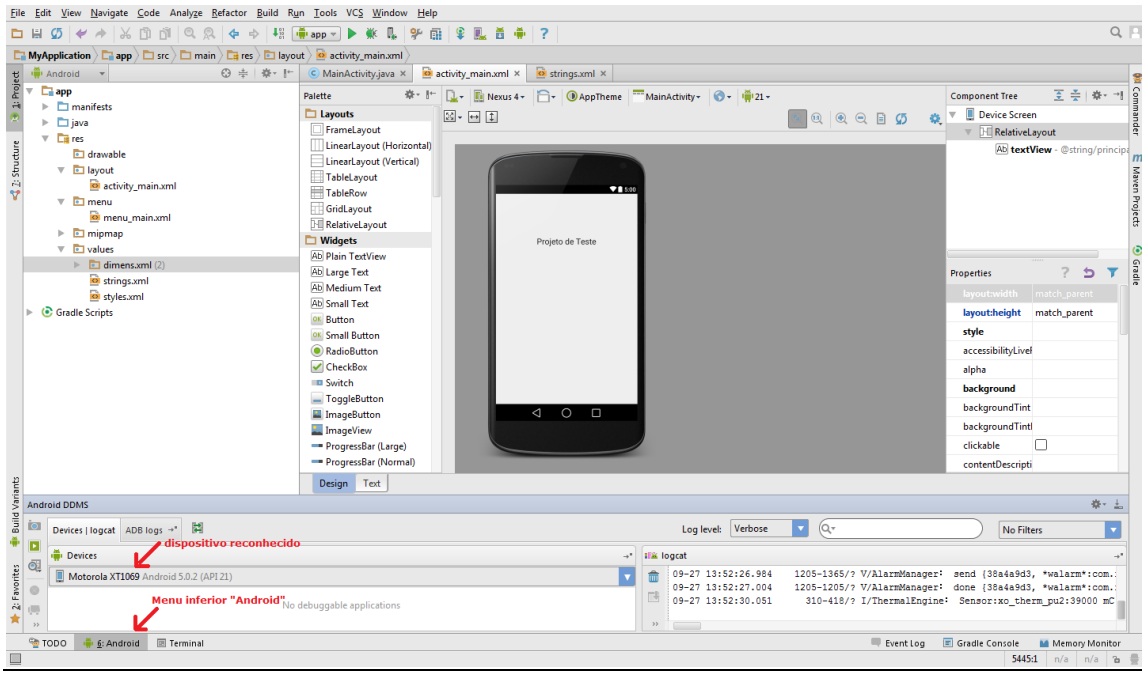

Figura 54 - Tela do Android Studio

Obs: Se o dispositivo não aparecer na combobox, o computador pode não ter conseguido baixar e instalar automaticamente o "USB driver" específico do seu dispositivo.Então você terá que baixá-lo (no site da fabricante do dispositivo) e instalá-lo manualmente (o processo muda conforme a marca/modelo do dispositivo).

Para executar o projeto no seu dispositivo execute o mesmo no Android Studio (vá em Run ou Debug). Deverá aparecer uma janela para a escolha do dispositivo ou emulador como ilustrado na Figura 55.

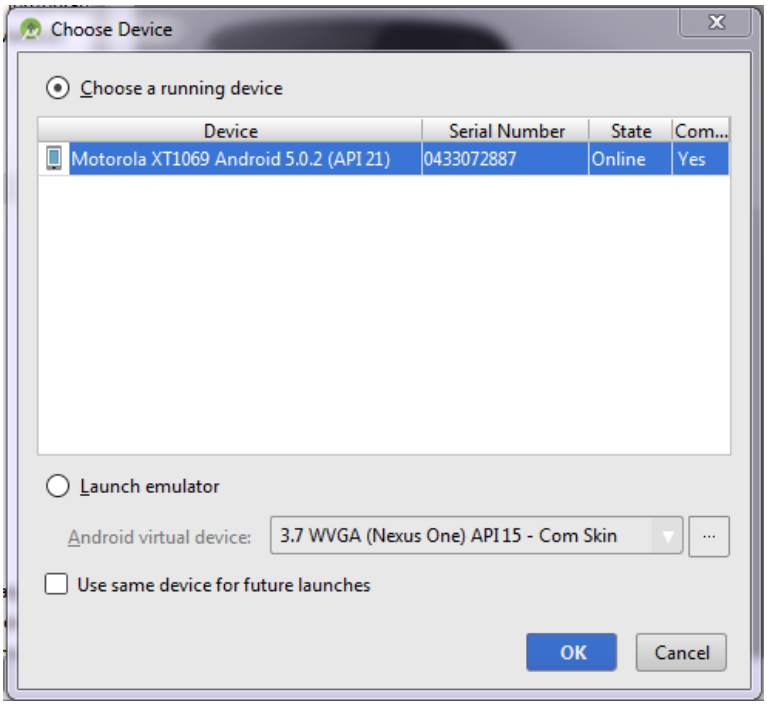

Figura 55 - Tela com dispositivos

Selecione o seu dispositivo e depois confirme clicando no "Ok". Agora o projeto será instalado no seu dispositivo e iniciado.

Obs[1]: Após terminar todo o processo, de desconectar o celular do computador o projeto não é desinstalado automaticamente do seu dispositivo. Se quiser desinstalá-lo, faça como com qualquer outro aplicativo: em seu aparelho vá em Configurações, depois em Aplicativos, selecione o aplicativo que deseja e em seguida selecione Desinstalar.
Obs[2]: Para desativar o modo programador no dispositivo móvel: em seu dispositivo móvel, vá em Configurações, em seguida Programador e marque-o como "desativado".

3.2 Gerando APK no Android Studio

A cada vez que se compila o aplicativo para teste (debug) é gerado um arquivo APK do projeto, que pode ser encontrado em *Nome_Projeto\app\build\outputs\apk* e o nome normalmente é algo como "app-debug.apk". Para testes no seu próprio dispositivo pode-se utilizar esse arquivo, porém é indicado só para testes.

Para a versão final do aplicativo, na barra superior do Android Studio deve-se ir em "Build", depois em "Generate Signed APK" e após isso será apresentada uma tela com informações a serem preenchidas, as quais serão utilizadas na Google Store para o

envio desta aplicação, esta será uma assinatura digital no seu apk. Quando tiver concluído esta etapa o Android Studio compila o apk assinado e abre uma pasta mostrando aonde o arquivo se encontra. Este é o arquivo que deve ser utilizado pelos usuários.

Para instalar este apk em seu dispositivo faça como explicado no Passo 1 e em seguida conecte seu aparelho no computador. Salve o apk em alguma pasta dentro do dispositivo, em seguida desconecte-o do computador. Acesse a pasta que contem o aplicativo e clique nele. Se o aparelho não estiver com permissão para instalação de aplicações de fontes que não seja do Google Play aparecerá uma mensagem avisando.

Vá em "Segurança" e na parte de "Administração do dispositivo" selecione "Fontes desconhecidas" para que o aparelho permita a instalação.

Em seguida acesse o aplicativo de novo e tente instalar novamente. Após concluir a instalação será perguntado se quer iniciá-lo ou não.

Apêndice II - Método para apresentar na tela uma janela com uma mensagem

```
/**
 * Abre um diálogo simples na tela e um botão "OK", que quando pressionado faz o diálogo sumir
 * OBS: Método assíncrono. Ou seja, após abrir o diálogo continua executando as próximas linhas de código.
 * @param context - Activity que chamou o diálogo (ex: MinhaActivity.this)
 * @param titulo - Titulo do diálogo
 * @param mensagem - Mensagem do diálogo
 */
public static void mostrarCaixaDialogoSimples(Context context, String titulo, String mensagem) {

    AlertDialog.Builder builder = new AlertDialog.Builder(context);
    builder.setTitle(titulo);
    builder.setMessage(mensagem);
    builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int id) {
            dialog.dismiss();
        }
    });

    AlertDialog alertDialog = builder.create();
    alertDialog.show();
}
```

Código 1 - Classe *MensagemUtils*

Apêndice III - Código do aplicativo Simon

```
importandroid.os.Handler;
import android.widget.Button;

/**
 * Classe de utilidades para botões no Android
 */
public class BotaoUtils {

	private static final int MILLISEGUNDOS_PISCAR_BOTAO = 1000;
	private static final int MILLISEGUNDOS_ESPERAR_PISCAR_BOTAO = 500;

	/**
	* Pisca os botões na sequência que eles estão na lista 'listaBotoes'
	* Atenção: Método Assíncrono
	* @param arrayBotoes
	* @param configuracaoPisque
	*/
	public static void piscarBotoes(Button[] arrayBotoes, ConfiguracaoPisqueBotao
configuracaoPisque) {
		piscarBotoes(arrayBotoes, 0, configuracaoPisque);
	}
```

Código 2 - Classe ButtonUtils (parte 1 de 3)

```
/**
 * Pisca os botoes a partir do indice "indiceBotaoInicioPiscar" na lista "listaBotoes"
 * O 'piscar' do botão é executado conforme a configuração 'configuracaoPisque' passada
 * @param arrayBotoes
* @param indiceBotaoInicioPiscar
* @param configuracaoPisque
*/
private static void piscarBotoes(final Button[] arrayBotoes,
                                 final int indiceBotaoInicioPiscar,
                                 finalConfiguracaoPisqueBotao configuracaoPisque) {

    // marca o botão para acender ao final do método
    configuracaoPisque.acender(arrayBotoes[indiceBotaoInicioPiscar]);

    // chama método assíncrono para executar daqui a X milissegundos para apagar o
botão e acender os próximos botões
    new Handler().postDelayed(new Runnable() {
        @Override
        public void run() {
            // marca o botão para "apagar" ao final do método
            configuracaoPisque.apagar(arrayBotoes[indiceBotaoInicioPiscar]);

            // Continua piscando os próximos botões ou, se terminou, habilita para o
jogador fazer a jogada
            int proximoIndiceBotaoPiscar = indiceBotaoInicioPiscar + 1;
            boolean temMaisBotoesParaPiscar = proximoIndiceBotaoPiscar
<arrayBotoes.length;

            if (temMaisBotoesParaPiscar) {
                esperarParaPiscar(arrayBotoes, proximoIndiceBotaoPiscar,
configuracaoPisque);
            } else {
                configuracaoPisque.executarAposPiscarBotoes();
            }
        }
    }, MILLISEGUNDOS_PISCAR_BOTAO);
}
```

Código 3 - Classe ButtonUtils (parte 2 de 3)

```
    /**
     * Espera um tempo para piscar o próximo botão da lista
     * @param arrayBotoes
     * @param indiceBotaoInicioPiscar
     * @param configuracaoPisque
     */
    private static void esperarParaPiscar(final Button[] arrayBotoes,
                                          final int indiceBotaoInicioPiscar,
                                          finalConfiguracaoPisqueBotao configuracaoPisque) {
        new Handler().postDelayed(new Runnable() {
            public void run() {
                piscarBotoes(arrayBotoes, indiceBotaoInicioPiscar, configuracaoPisque);
            }
        }, MILLISEGUNDOS_ESPERAR_PISCAR_BOTAO);
    }
}
```

Código 4 - Classe ButtonUtils (parte 3 de 3)

```
importandroid.widget.Button;

/**
 * Permite a definição personalizada do 'pisque' de um botão
 */
public interface ConfiguracaoPisqueBotao {

    /**
     * Define como será o realce do botão quando ele for piscar
     * @param button - botão a ser piscado
     */
    public void acender(Button button);

    /**
     * Define como o botão voltará ao normal quando terminar de piscar
     * @param button - botão a ser apagado
     */
    public void apagar(Button button);

    /**
     * Opcional: Define uma execução para quando terminar de piscar todos os botões.
     */
    public void executarAposPiscarBotoes();
}
```

Código 5 - Interface ConfiguracaoPisqueBotao

Apêndice IV - Código do layout do tabuleiro Mancala

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:id="@+id/tabuleiro"
android:layout_width="match_parent" android:layout_height="match_parent"
android:orientation="vertical"
android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
android:paddingBottom="@dimen/activity_vertical_margin"
tools:context=".activity.JogoActivity">

    <!-- BOTÕES DO JOGADOR 2 -->
    <TableRow android:layout_width="match_parent" android:layout_weight="30">

        <Button android:tag="jogador2-cavidade6" android:layout_weight="1"
android:layout_height="match_parent" android:layout_width="0dp"
android:background="@drawable/estilo_botao_jogador_2" android:onClick="onClickButtonCavidade"
android:layout_marginLeft="0dp" android:layout_marginRight="8dp"
android:textSize="@dimen/tamanho_fonte_cavidade" android:text="0"/>
        <Button android:tag="jogador2-cavidade5" android:layout_weight="1"
android:layout_height="match_parent" android:layout_width="0dp"
android:background="@drawable/estilo_botao_jogador_2" android:onClick="onClickButtonCavidade"
android:layout_marginLeft="8dp" android:layout_marginRight="8dp"
android:textSize="@dimen/tamanho_fonte_cavidade" android:text="0"/>
        <Button android:tag="jogador2-cavidade4" android:layout_weight="1"
android:layout_height="match_parent" android:layout_width="0dp"
android:background="@drawable/estilo_botao_jogador_2" android:onClick="onClickButtonCavidade"
android:layout_marginLeft="8dp" android:layout_marginRight="8dp"
android:textSize="@dimen/tamanho_fonte_cavidade" android:text="0"/>
        <Button android:tag="jogador2-cavidade3" android:layout_weight="1"
android:layout_height="match_parent" android:layout_width="0dp"
android:background="@drawable/estilo_botao_jogador_2" android:onClick="onClickButtonCavidade"
android:layout_marginLeft="8dp" android:layout_marginRight="8dp"
android:textSize="@dimen/tamanho_fonte_cavidade" android:text="0"/>
        <Button android:tag="jogador2-cavidade2" android:layout_weight="1"
android:layout_height="match_parent" android:layout_width="0dp"
android:background="@drawable/estilo_botao_jogador_2" android:onClick="onClickButtonCavidade"
android:layout_marginLeft="8dp" android:layout_marginRight="8dp"
android:textSize="@dimen/tamanho_fonte_cavidade" android:text="0"/>
        <Button android:tag="jogador2-cavidade1" android:layout_weight="1"
android:layout_height="match_parent" android:layout_width="0dp"
android:background="@drawable/estilo_botao_jogador_2" android:onClick="onClickButtonCavidade"
android:layout_marginLeft="8dp" android:layout_marginRight="0dp"
android:textSize="@dimen/tamanho_fonte_cavidade" android:text="0"/>

    </TableRow>
```

Código 6 - Layout *activity_jogo*.xml (parte 1 de 3)

```
<!-- CENTRO DA TELA: REPOSITÓRIOS, BOTÃO INICIAR E INFORMAÇÕES -->
<TableRow android:layout_width="match_parent" android:layout_weight="40"
android:layout_marginTop="10dp"  android:layout_marginBottom="10dp" >

    <Button android:id="@+id/repositorioJogador2" android:layout_weight="1"
android:layout_height="match_parent" android:layout_width="0dp"
android:background="@drawable/estilo_botao_jogador_2"
android:layout_marginRight="10dp" android:enabled="false"
android:textSize="@dimen/tamanho_fonte_repositorio" android:text="0" />

    <LinearLayout android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:layout_margin="12dp"
        android:gravity="center">

    <TextView
        android:id="@+id/txtInfoJogadorCorrente"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/txt_info_jogador_corrente"
        android:textSize="@dimen/tamanho_fonte_jogador_corrente"
        android:visibility="gone"/>
    <TextView
        android:id="@+id/txtNomeJogadorCorrente"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="@dimen/tamanho_fonte_jogador_corrente"
        android:visibility="gone"/>
    <Button
        android:id="@+id/botao_iniciar"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="@string/txt_btn_inicio"
        android:textSize="@dimen/tamanho_fonte_botao_iniciar"
        android:onClick="onClickButtonIniciar"/>
    </LinearLayout>

    <Button android:id="@+id/repositorioJogador1" android:layout_weight="1"
android:layout_height="match_parent" android:layout_width="0dp"
android:background="@drawable/estilo_botao_jogador_1"
android:layout_marginLeft="10dp" android:enabled="false"
android:textSize="@dimen/tamanho_fonte_repositorio" android:text="0" />

</TableRow>
```

Código 7 - Layout *activity_jogo*.xml (parte 2 de 3)

```
    <!-- BOTÕES DO JOGADOR 1 -->
    <TableRow android:layout_width="match_parent" android:layout_weight="30">

    <Button android:tag="jogador1-cavidade1" android:layout_weight="1"
android:layout_height="match_parent" android:layout_width="0dp"
android:background="@drawable/estilo_botao_jogador_1"
android:onClick="onClickButtonCavidade" android:layout_marginLeft="0dp"
android:layout_marginRight="8dp" android:textSize="@dimen/tamanho_fonte_cavidade"
android:text="0"/>
    <Button android:tag="jogador1-cavidade2" android:layout_weight="1"
android:layout_height="match_parent" android:layout_width="0dp"
android:background="@drawable/estilo_botao_jogador_1"
android:onClick="onClickButtonCavidade" android:layout_marginLeft="8dp"
android:layout_marginRight="8dp" android:textSize="@dimen/tamanho_fonte_cavidade"
android:text="0"/>
    <Button android:tag="jogador1-cavidade3" android:layout_weight="1"
android:layout_height="match_parent" android:layout_width="0dp"
android:background="@drawable/estilo_botao_jogador_1"
android:onClick="onClickButtonCavidade" android:layout_marginLeft="8dp"
android:layout_marginRight="8dp" android:textSize="@dimen/tamanho_fonte_cavidade"
android:text="0"/>
    <Button android:tag="jogador1-cavidade4" android:layout_weight="1"
android:layout_height="match_parent" android:layout_width="0dp"
android:background="@drawable/estilo_botao_jogador_1"
android:onClick="onClickButtonCavidade" android:layout_marginLeft="8dp"
android:layout_marginRight="8dp" android:textSize="@dimen/tamanho_fonte_cavidade"
android:text="0"/>
    <Button android:tag="jogador1-cavidade5" android:layout_weight="1"
android:layout_height="match_parent" android:layout_width="0dp"
android:background="@drawable/estilo_botao_jogador_1"
android:onClick="onClickButtonCavidade" android:layout_marginLeft="8dp"
android:layout_marginRight="8dp" android:textSize="@dimen/tamanho_fonte_cavidade"
android:text="0"/>
    <Button android:tag="jogador1-cavidade6" android:layout_weight="1"
android:layout_height="match_parent" android:layout_width="0dp"
android:background="@drawable/estilo_botao_jogador_1"
android:onClick="onClickButtonCavidade" android:layout_marginLeft="8dp"
android:layout_marginRight="0dp" android:textSize="@dimen/tamanho_fonte_cavidade"
android:text="0"/>

    </TableRow>
</TableLayout>
```

Código 6 - Layout *activity_jogo*.xml (parte 3 de 3)