UNIVERSIDADE FEDERAL DA PARAÍBA

CAMPUS IV - LITORAL NORTE - RIO TINTO / PB

CENTRO DE CIÊNCIAS APLICADAS E EDUCAÇÃO

DEPARTAMENTO DE CIÊNCIAS EXATAS

LICENCIATURA EM CIÊNCIA DA COMPUTAÇÃO

ANALOGIA COMO APOIO AO ENSINO DE PROGRAMAÇÃO: UMA ANIMAÇÃO DO COMANDO WHILE

DAVID BARBOSA NETO

Orientador: Prof. MSc. Vanessa Dantas

RIO TINTO - PB

2014

DAVID BARBOSA NETO

ANALOGIA COMO APOIO AO ENSINO DE PROGRAMAÇÃO: UMA ANIMAÇÃO DO COMANDO WHILE

Monografia apresentada para obtenção do grau de Licenciado (a) à banca examinadora no Curso de Licenciatura em Ciência da Computação do Centro de Ciências Aplicadas e Educação (CCAE), Campus IV da Universidade Federal da Paraíba.

Orientador: Prof. MSc. Vanessa Dantas

.

RIO TINTO - PB
2014

B238a Barbosa Neto, David.
Analogia como apoio ao ensino de programação: uma animação do comando While. / David Barbosa Neto. – Rio Tinto: [s.n.], 2015.
49 f. : il.

Orientador(a): Prof. Msc. Vanessa Dantas.
Monografia (Graduação) – UFPB/CCAE.

1. Programação - estudo e ensino. 2. Programação - ensino e aprendizagem. 3. Ciência da computação.

UFPB/BS-CCAE CDU: 004.4(043.2)

DAVID BARBOSA NETO

ANALOGIA COMO APOIO AO ENSINO DE PROGRAMAÇÃO: UMA ANIMAÇÃO DO COMANDO WHILE

Trabalho de Conclusão de Curso submetido ao Curso de Licenciatura em Ciência da Computação da Universidade Federal da Paraíba, Campus IV, como parte dos requisitos necessários para obtenção do grau de LICENCIADO EM LICENCIATURA EM CIÊNCIA DA COMPUTAÇÃO.

_____

Assinatura do autor:

APROVADO POR:

_____

Coordenador do Curso
Universidade Federal da Paraíba – Campus IV

_____

Orientador: Prof. Vanessa Dantas
Universidade Federal da Paraíba – Campus IV

_____

Prof. Dr. XXX
Universidade XXX

_____

Prof. Dr. XXX
Universidade XXX

RIO TINTO - PB
2014

“O temor do Senhor é o princípio da sabedoria, e o conhecimento do Santo a prudência.” Provérbios 9:10

AGRADECIMENTOS

A meu Deus imensa gratidão por me proporcionar perseverança para concluir meu curso e entendimento para realização deste trabalho final, a Ele estendo profunda alegria por me permitir ter a grata satisfação de ter pessoas fantásticas ao meu redor que me acompanham e fazem meus dias felizes, dentre estas estão as citadas em seguida.

Agradeço a meus pais e meus irmãos por todo inigualável apoio e amor que estenderam e estendem a mim e por me incentivarem a prosseguir. Por grande estima e amparo, a minha amada esposa, pessoa de único e terno sentimento que está presente sempre, por seu amor e carinho.

As pessoas tão diferentes e guerreiras da turma 2008.2 do curso de Licenciatura em Ciências da Computação, meus estudos foram muito felizes e memoráveis graças a presença de vocês também em minha vida.

A meus amigos André, Aldrins, Igor, Felipe, Ruan e Wesley por todo apoio e incentivo. Aos amigos das empresas VSoft e Thinkbox por toda ajuda, momentos felizes e aprendizado, sou grato por vocês, em especial aos gênios da equipe de pesquisa.

A Vanessa Dantas, orientadora desta monografia, pelo seu exemplo de dinamismo e trabalho, gostaria de agradecê-la por ter acreditado na minha caminhada e estar sempre pronta e disponível para me ajudar em meus deslizes, falhas e dúvidas.

A todos os amigos, funcionários, professores e colegas que fazem parte do DCE da Universidade Federal da Paraíba que de uma forma ou de outra, contribuíram para a conclusão deste trabalho. Principalmente cito os professores Carla, Jorge, Rafael, Pasqueline, Yuri e Laudelino.

RESUMO

O ensino-aprendizagem da disciplina inicial de programação estruturada apresenta frequentes dificuldades de assimilação e abstração por parte dos alunos. Acredita-se que os problemas enfrentados influenciam nos índices de evasão e reprovação corriqueiramente vistos nas instituições de ensino superior do país.

Para melhorar o atual cenário uma abordagem alternativa seria relacionar os conceitos do conteúdo de programação usando analogias de situações cotidianas que facilitem o entendimento das abstrações da disciplina e apresentar esses assuntos de forma dinâmica e interativa**.**

O propósito principal deste trabalho é realizar a construção de uma animação que apoie o entendimento e assimilação de conceitos usados para criar algoritmos e posteriormente, códigos. Para isso, a animação busca contribuir com uma forma mais familiar de apresentar o conteúdo inicial de programação, entendimento difícil para os alunos iniciantes. Pretende-se usar uma analogia a fim de agir como motivador e facilitador para os alunos, usando o recurso de animação como recurso tecnológico mais atraente para o público-alvo.

Palavras chave: Ensino de Programação, Analogia, Animação

ABSTRACT

The teaching and learning of the initial discipline of structured programming features frequent difficulties of assimilation and abstraction by the students. It is believed that the problems faced influence the dropout rates and failure routinely seen in higher education institutions.

One possible alternative would be to relate the concepts of programming content using analogies of everyday situations that facilitate the understanding of the abstractions of discipline and present these issues in a dynamic and interactive way.

The main purpose of this work is the construction of an animation to support the understanding and assimilation of concepts used to create algorithms and later codes. For this, the animation seeks to contribute to a more familiar way of presenting the initial content programming, difficult to understand for beginners. The aim is to use an analogy in order to act as a motivator and facilitator for students using the animation feature as more attractive technological option to the target audience.

Keywords: Programming Education, Analogy, Animation

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Um tema recorrente em estudos científicos consiste nas dificuldades enfrentadas por alunos ingressantes nos cursos de Computação, especificamente nas disciplinas de Algoritmos e Introdução à Programação, mostra-se tema recorrente de estudos científicos. A análise e resolução de problemas, importante elemento e foco do raciocínio lógico, apresentam-se como os primeiros obstáculos para o aprendizado dos alunos de cursos da área de Computação. Esse fato mostra-se preocupante, visto que seu aprendizado constitui-se como pilar sustentador de várias disciplinas presentes no ensino superior de Computação.

Com foco no entendimento e resolução de problemas de forma lógica, os conteúdos lecionados nas disciplinas iniciais desses cursos são base para os estudos posteriores e entendimento de outras atividades "como especificação, projeto, validação, modelagem e estruturação de programas e dados, utilizando as linguagens de programação como ferramentas." (dos Santos, 2006).

Muitos dos recém-chegados nos cursos relacionados a Computação não trazem a abstração, pensamento crítico ou interpretação como habilidades bem desenvolvidas no período do ensino médio. Geralmente, deve-se ao pouco estímulo em sala para o desenvolvimento de conclusões a partir do próprio pensamento e análise, influenciando diretamente em seu desempenho e adaptação aos conteúdos propostos.

Para caracterizar a problemática da disciplina de Introdução à Programação, buscou-se coletar dados no ambiente de ensino dos campi I e IV da Universidade Federal da Paraíba. Os dados adquiridos foram destacados dos cursos de Bacharelado em Sistemas de Informação e Ciência da Computação em suas modalidades de Licenciatura e Bacharelado. As informações apresentam as taxas de aprovação e reprovação, bem como a evasão dos cursos, compreendidas entre os períodos letivos de 2011.1 a 2013.2. Os dados coletados dividem-se em taxa de aprovação, reprovação por falta, reprovação por média e trancamento. Eles estão representados resumidamente no Gráfico 1, no Gráfico 2 e no Gráfico 3.

Como primeira análise dos dados, nota-se que, em todos os três cursos, a taxa de aprovação não chegou a 50%, salvo o período 2011.2 no curso de Licenciatura em Ciência da Computação. Em termos de reprovações, as causadas pelo número

de faltas tendem a ser maiores que as por média. Somente em um caso na coleta de dados, no período 2013.2 do curso de Licenciatura em Ciências da Computação, não houve informações sobre trancamento. A taxa de reprovação aparece predominante em toda a amostra coletada, e tais dados demonstram uma preocupante situação.

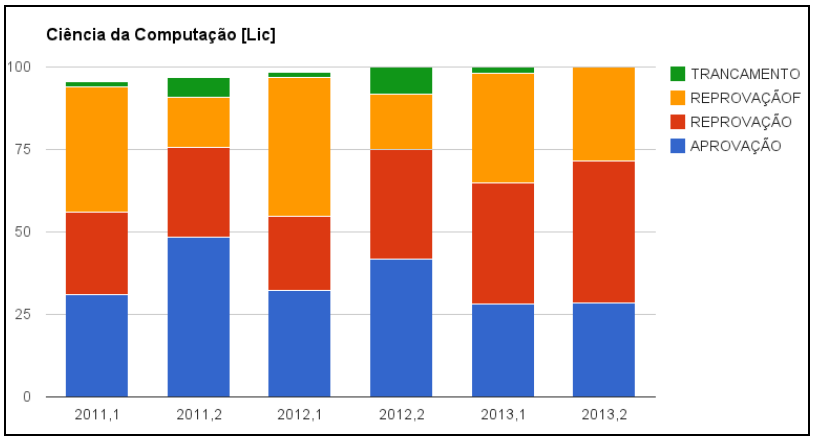


Figura 1 - Índices de Aprovação e Reprovação (Licenciatura em Computação)

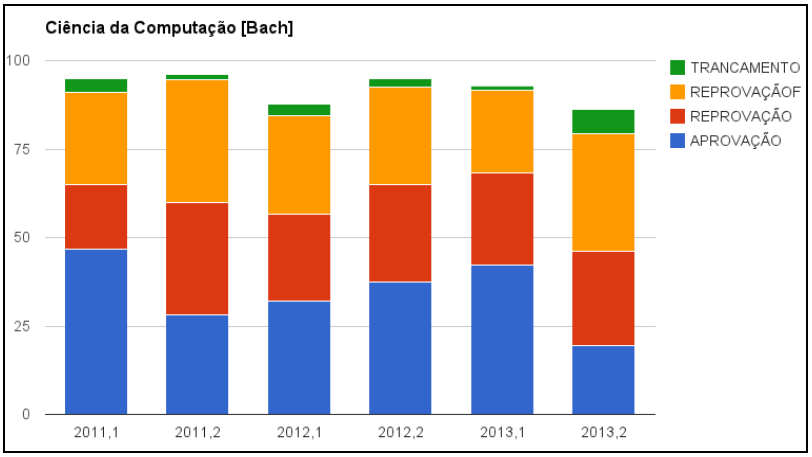


Figura 2- Índices de Aprovação e Reprovação (Bacharelado em Computação)

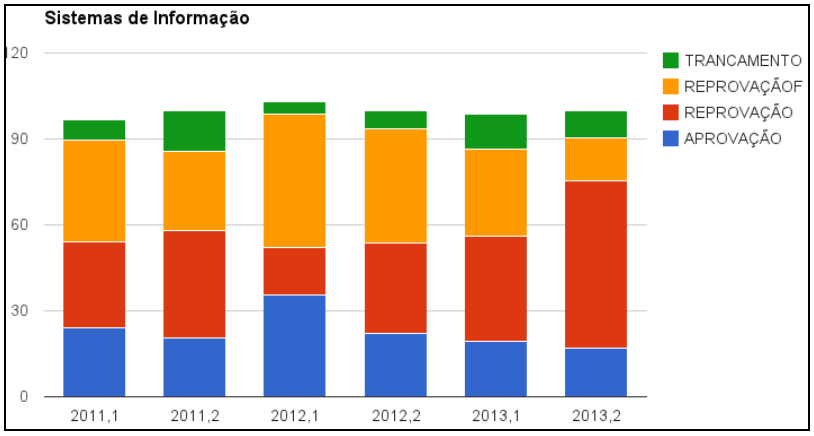


Figura 3 - Índices de Aprovação e Reprovação (Bach. Sistemas de Informação)

As pesquisas realizadas com essa problemática buscam em seus trabalhos científicos entender as dificuldades existentes e trazer melhorias para o ensino de programação. De comum entendimento (Silva, 2009; Aureliano, 2012; Paula, 2009; Rapkiewicz, 2006), nota-se a deficiência no pensamento abstrato, base para a assimilação do raciocínio lógico, que por sua vez guia a compreensão das disciplinas em questão. Tal realidade é um dos fatores que resulta em reprovações e evasão do curso. Silva (2009) explica que as reprovações trazem desmotivação para os alunos, prejudicando sua aprendizagem e a evasão gera seu afastamento, distanciando-os das metas intelectuais ou profissionais quanto ao curso.

As disciplinas iniciais de programação tornam-se, segundo Rapkiewicz (2006) "um dos gargalos existentes nos cursos de graduação, particularmente de Computação, dificultando ou até mesmo impedindo a continuidade dos alunos no curso". Em contato com a adversidade, o estudante apresenta desmotivação quanto a seu rendimento e futuro no curso, já que no primeiro contato com os assuntos ele tem problemas em aprender o conteúdo abordado. Essas disciplinas são base para este estudo, uma vez que apresenta conteúdo que envolve o conceito e a apresentação da lógica computacional, como também entendimento e desenvolvimento da programação de computadores. Buscando-se melhorar o aprendizado dos assuntos vistos, há anos pesquisas se dedicam a desenvolver objetos de aprendizagem que auxiliem o estudante ou maneiras inovadoras de ensino que aprimorem o entendimento dos conteúdos abordados. Segundo Lahtinen, Ala-Mutka e Järvinen (2005), *loops*, variáveis, recursão, e passagem de parâmetros são os principais conceitos para o estudo dos iniciantes.

Consideradas resultado das dificuldades enfrentadas e indicadoras da relevância do problema apresentado por diversos autores, a reprovação e muitas vezes a evasão dos alunos iniciantes apresentam alguns fatores que as justificam. Schuvartz (2006) apresenta como aspectos indicadores a falta de aptidões dos estudantes universitários, as estratégias de ensino utilizadas, as atitudes e expectativas de alunos e professores, bem como a falta de recursos disponíveis, estes corretamente elaborados e empregados.

Deters, et al. (2008) ressalta que o elevado número de problemas de aprendizagem favorece as reprovações e desistências. O autor cita Raabe, et al. (2005), que identificaram três tipos de aspectos que contribuem para a problemática:

a didática empregada, a cognição dos alunos e possíveis problemas afetivos que eles, enquanto novatos no curso, podem apresentar.

Estudos apresentados na revisão feita por Aureliano (2012) vêm frequentemente destacando melhorias na aprendizagem de programação na pesquisa nacional sobre o ensino de Computação. A busca de Aureliano obteve apresenta propostas de ferramentas, linguagens, metodologias de ensino e técnicas de avaliação, algumas delas serão no tópico 2.3 deste trabalho.

Essas abordagens não apresentam destaque no uso de analogias e nos conhecimentos previamente construídos pelos alunos tornando assim maior o esforço para entender novos conceitos, pois não oferecem um tipo de assimilação entre os assuntos de programação com seu entendimento.

Se faz necessário uma solução que funcione como uma ponte entre os conhecimentos de vida adquiridos no dia-dia e os de programação, tornando a compreensão mais clara. Acredita-se que um meio visual que retrate analogias, trazendo saberes comuns baseados na explicação dos conceitos, tornaria o entendimento mais fácil, visto que as relações estabelecidas entre situações cotidianas e suas semelhanças com conhecimentos lógicos da programação tornaria a forma de raciocínio dos alunos mais familiar.

Falando sobre o uso de animações no entendimento de algoritmos, Hansen (2002) afirma que por mais de uma década pesquisadores e educadores investigam o assunto e seu efeito para ajudar o aluno nas dificuldades de entendimento sobre algoritmos. Os trabalhos desses pesquisadores defendem intuitivamente que as animações, ilustrando o comportamento dinâmico dos algoritmos, revelaram-se eficazes no sentido de ajudar os alunos a superar as dificuldades.

O presente trabalho aborda o uso de animações que apresentem analogias como agentes facilitadores no ensino-aprendizagem de conceitos iniciais referentes ao estudo de programação.

As animações neste trabalho têm como propósito apresentar conceitos de programação usando esse recurso como tecnologia atrativa para o público-alvo por serem ilustrações em movimento e também por serem reconhecidos enquanto artifícios bem aceitos no ensino de programação. A proposta é facilitar a compreensão do conteúdo tomando como base situações cotidianas, ações diárias como preparar uma refeição ou se dirigir a determinado local, que possam representar abstrações utilizadas na programação, sendo uteis para assimilação do

conteudo por parte dos alunos.

Para esse estudo, pretende-se utilizar especificamente a analogia como recurso e meio para alcançar um novo processo de ensino-aprendizagem na área de conhecimento abordada. Chee, (1993) afirma que a explanação de analogias pode ser vista com propriedades de clareza, riqueza e sistematicidade, e entende-se que o uso de animações pode proporcionar uma explanação mais atrativa e dinâmica para os alunos usando tais propriedades como elementos estimulantes para o aprendizado.

1.1 OBJETIVOS

Desenvolver uma animação interativa que relacione conceitos introdutórios de programação com situações cotidianas a partir de e analogias de modo a apoiar alunos ingressantes em cursos da área de Computação na compreensão de conceitos abstratos. Para isso propõem-se o uso dessas animações como forma e recurso para auxiliarem no entendimento sendo utilizadas como instrumentos de apoio em suas disciplinas.

1.2.1 OBJETIVOS ESPECÍFICOS

- Pesquisar sobre o ensino de programação, seus problemas e possíveis soluções.
- Identificar analogias cotidianas relacionadas a conceitos de programação.
- Pesquisar ferramentas de desenvolvimento de animações.
- Desenvolver uma animação de uma analogia do comando *While*.
- Fazer um estudo de aceitação do instrumento com o público-alvo.

1.2 METODOLOGIA

Para constituir conhecimento sobre o assunto, sendo este trabalho de caráter de pesquisa qualitativa, de exploração da disciplina abordada, os problemas constituídos durante sua execução e as consequências das dificuldades encontradas em trabalhos científicos. Serão identificadas as abordagens que apresentem o

ensino-aprendizagem de programação que usem abstrações de situações do dia-dia por uma pesquisa bibliográfica. Serão coletados dados de aprovação, reprovação e evasão dos alunos na disciplina de Introdução à Programação junto à Pró-Reitoria de Graduação a fim de observar o contexto da Universidade Federal da Paraíba quanto ao problema destacado, motivador desta pesquisa. Será realizada uma busca por possíveis soluções que a comunidade acadêmica propõe para melhorar o ensino de programação, e também serão identificados os conteúdos mais importantes para o entendimento do aluno.

Como base para o entendimento e influência da proposta desse trabalho, será realizada uma pesquisa bibliográfica a fim de identificar a importância e o uso de analogias no ensino de programação e qual o conteúdo eleito para ser abordado na elaboração da analogia na animação.

Será procurado compreender a área de pesquisa sobre o uso de animações e representações gráficas como recursos didáticos para o assunto abordado, será desenvolvida uma analogia sobre um conteúdo de programação de computadores.

Como tarefa final, será produzida uma animação que expresse uma analogia sobre um conceito inicial de programação, investigado e analisado um experimento de aceitação com alunos dos cursos de Sistemas da Informação e Ciência da Computação em Rio Tinto a fim de realizar um experimento de aceitação do público-alvo para captar o *feedback* dos estudantes. O experimento será realizado com alunos da disciplina de Introdução à Programação da UFPB – Campus IV. Eles usarão a animação e logo após responderão um questionário online sobre o conteúdo que será apresentado e as características de aprendizado, atratividade, interação e uso da animação.

1.3 ESTRUTURA DO TRABALHO

Para melhor organização e entendimento, este documento foi dividido em quatro capítulos. No capítulo II serão tratadas as possíveis causas que justificam tal realidade e que se tornam pontos de entendimento das dificuldades e problemas enfrentados em outras instituições de ensino superior em seus cursos de computação. Também será exposto o uso de analogias no ensino-aprendizagem de programação, bem como seu embasamento pedagógico.

Em seguida, no capítulo III será apresentada mais detalhadamente a proposta e idealização desta produção científica como também o seu desenvolvimento, procurando destacar as etapas feitas para a construção do objeto de estudo final proposto e os resultados adquiridos em um experimento avaliativo do instrumento.

Por fim no capítulo IV será apresentada uma conclusão apresentando as considerações finais sobre o assunto tratado neste documento e também o que é esperado em trabalhos futuros relacionados ao tema.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão os tratados os conhecimentos em que os objetivos e a proposta declarados serão teoricamente baseados. Serão apresentados alguns trabalhos relacionados à melhoria do ensino-aprendizagem de programação no país, sobre a reprovação e a evasão presentes nos cursos de Computação, as soluções propostas pela Academia, e recursos que irão ser utilizados no desenvolvimento deste trabalho.

2.1 ANALOGIAS, RECURSO DE ENSINO-APRENDIZAGEM

Segundo Pimentel(2003), as disciplinas iniciais referentes à programação de computadores têm por objetivo o desenvolvimento dos alunos em construir soluções computacionais capazes de resolver problemas do mundo real. Compreende-se que, para a concepção e o desenvolvimento de programas, o processo de entendimento dos conceitos, bem como a simplificação dos detalhes dos problemas são etapas cruciais para a boa e eficaz programação. Falando sobre essas etapas, Kramer(2007) afirma que elas são vistas como aspectos formadores da definição de abstração. Segundo o autor, o pensamento abstrato é essencial para manipular e raciocinar sobre abstrações como análise de modelos ou programas em linguagem de programação.

Forišek (2012) diz que a analogia[1] e a metáfora[2] são apresentadas por muitos autores como ferramentas facilitadoras no desenvolvimento de novos modelos mentais e novas abstrações, em suma, novos conceitos.

Em seu estudo, Silva, *apud* BORGUES e RODRIGUES (2009) apresenta exemplos que tenham relação com o mundo real como uma das possíveis soluções para a desmotivação dos alunos, mostrando

> "uma forma de trabalhar os algoritmos para construção de soluções para problemas próximos do seu cotidiano; diferentemente do que é visto hoje, onde os professores apresentam os algoritmos como problemas distantes (como o do fatorial, por exemplo), dificultando o aprendizado dos alunos que possuem dificuldades com a matemática;"

[1] Uma relação estabelecida entre dois objetos formando um processo cognitivo que transfere ou compara similaridades entre eles.(Forišek, 2012)

[2] Processo cognitivo, uma figura de linguagem, que expressa algo de forma diferente, sendo entendido por ter-se conhecimento sobre o objetivo, significado ou conceito original. (Forišek, 2012)

Na perspectiva pedagógica essa melhor apreensão de conhecimento pode ser justificada apresentando alguns aspectos propostos por Piaget e Vygotsky citados no trabalho de (DIAS, 2009). O primeiro menciona, em sua proposta de processo de aprendizagem, a Adaptação. Como dois pilares desse processo têm-se a assimilação, onde existe uma introdução de novos conteúdos a uma estrutura cognitiva já concebida, e a acomodação, em que as mudanças ocorridas pelos novos saberes exigem uma reorganização da estrutura cognitiva. Esse momento mostra a relação e a importância dos conhecimentos previamente adquiridos em relação aos novos conteúdos. Entende-se que uma abordagem diferenciada pode focar na diminuição da distância entre antigos e novos conceitos, melhorando o entendimento dos novos conhecimentos apresentados no ensino em questão.

O segundo caracteriza o conhecimento cotidiano como aquele que é construído pela observação, manipulação e vivência. “Conceitos científicos se relacionam àqueles eventos não diretamente acessíveis à observação ou ação imediata”, promovendo interação entre os saberes. Aqui, vê-se menção às experiências cotidianas, ou seja, familiares e a existência de relações entre estas e o conteúdo científico.

Carbonell (1983) salienta que a maior parte da resolução de problemas humanos ocorre em espaços de problemas que são ou conhecidos ou variam apenas ligeiramente de situações familiares. Diz ainda que o aspecto mais importante é a memorização de problemas passados e suas soluções que apresentam semelhança com os novos problemas. Esse conhecimento, uma vez adquirido pode ser explorado na resolução de problemas.

2.2 ANIMAÇÃO DE ALGORITMOS

A animação como recurso ilustrativo para a representação de algoritmos mostra-se como uma forma atrativa de apresentar os conceitos de programação. (Byrne, et al. 1999) diz que a animação dá um passo além, enquanto visualizações estáticas mostram como algo se parece ou é constituído, a animação parece mais capaz de explicar um processo dinâmico , em constante evolução.

Segundo Carlson, et al, (1996), Animação de Algoritmos é um tipo de visualização de software de grande importância, caracteriza-se por uma visualização dinâmica das principais abstrações para se entender um programa. Destaca ainda que o valor da animação de um algoritmo está em sua capacidade de retratar a

essência da lógica do programa, evitando possível obscurecimento que vem da visualização detalhada das estruturas e variáveis de dados de um programa.

Esta forma de uso de animação para o ensino de programação que tem propósito semelhante à proposta desse trabalho, porém restringe na maioria das vezes seu objetivo em simular os passos dos algoritmos e códigos, não tendo foco na relação de analogias com o assunto.

2.3 TRABALHOS CORRELATOS

Acadêmicos imbuídos do objetivo de alcançar melhorias no ensino-aprendizagem buscam dar apoio sugerindo ferramentas e metodologias para tratar o problema. Assim, dadas as informações e frequentes estudos evidenciados na revisão feita em Aureliano (2012), o assunto vem ao longo dos anos sendo foco na área de pesquisa nacional sobre o ensino de Computação. Na citada pesquisa, encontra-se uma revisão dos artigos relacionados com o tema de forma que se percebe o interesse e o empenho nessa área de pesquisa. A busca de Aureliano obteve mais de 70 artigos completos apresentando propostas de ferramentas, linguagens, metodologias de ensino e técnicas de avaliação.

Os estudos mostrados na revisão de Aureliano (2012), como outros trabalhos da Academia, detém-se em apresentar como são ou funcionam os algoritmos e conceitos usados em seu entendimento e elaboração. De fato, ao demonstrar os passos e atividades desempenhadas pelo computador usando o conhecimento trabalhado, apresenta-se uma forma de simulação, proporcionando o acompanhamento do aluno e ajudando-o no amadurecimento do entendimento dos assuntos.

Nesta seção, serão apresentadas algumas das soluções e medidas apresentadas nos estudos feitos no Brasil, sobre analogias e sobre o uso de animações em algoritmos.

2.3.1 FERRAMENTAS DE APOIO

Neste tópico são apresentadas algumas ferramentas designadas para apoiarem o ensino de programação encontradas em trabalhos científicos brasileiros. Será mencionado o que elas propõe e algumas de suas características.

Um Verificador de Diferenças Significativas entre Programas(VDSP) é um avaliador automatizado de programas que serve de auxílio aos alunos, percorrendo

o programa submetido pelo estudante e avaliando-o atribuindo um valor de qualidade mediante métricas de programação. Após isso, um relatório é gerado apresentando anomalias e soluções referentes ao programa submetido. O VDSP apresenta métricas para avaliar os programas, verificando o tamanho dos identificadores, percentual dos que são constantes, tamanho dos módulos, quantidade dos módulos, percentual de linhas endentadas e comentadas, além de também as linhas em branco. A ferramenta é proposta em Faria, et al (2005), apresentou como resultados programas mais compreensíveis melhorando a qualidade de escrita dos códigos. Seu uso foi proposto em um ambiente online denominado *Learn on Group*. Essa plataforma online apresenta a idéia de usar um aprendizado colaborativo onde os professores colocam as tarefas a serem desempenhadas pelos alunos, uma vez feitos os programas são sumetidos e avaliados pelo VDSP e os professores dão seu *feedback* na plataforma online.

O Visualg é apresentado em Souza (2009) como ferramenta para codificar, depurar e executar o pseudocódigo, tendo recursos de execução passo a passo e visualização do conteúdo de variáveis, dentre outras funcionalidades. Seu uso propõe um ambiente para escrever algoritmos, sua avaliação é focada na sintaxe do código e na depuração dos resultados, um pouco mais simples que o verificador de diferenças significativas de programas. A imagem abaixo mostra a interface principal do ambiente e um exemplo de pseudocódigo escrito.

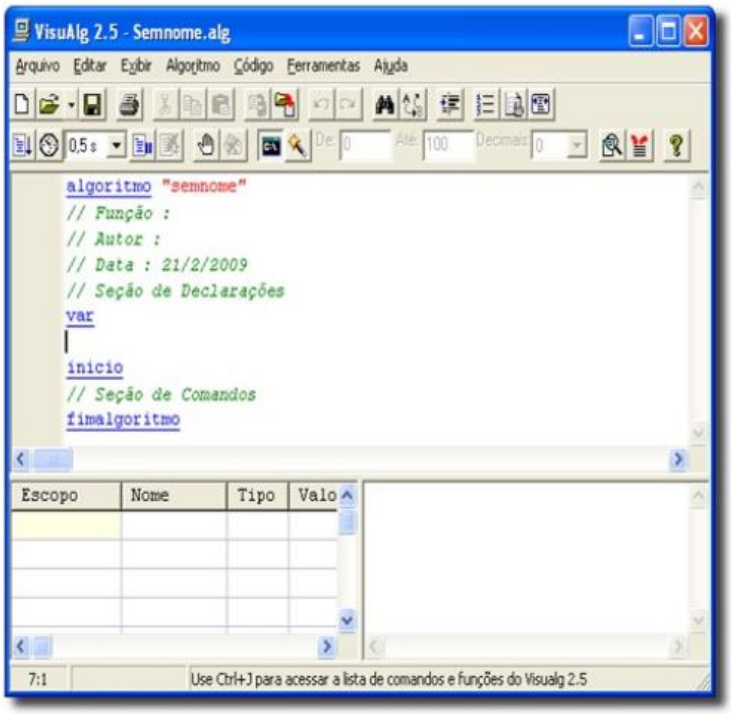


Figura 4 - Tela Principal do Visualg.

Já o JavaTool é uma ferramenta onde o aluno pode editar, compilar e depurar o código como o Visualg, porém utilizando a linguagem Java e podendo também visualizar a animação do código feito.(Mota, et AL, 2008). As partes principais da ferramenta são o local de edição de código, a área de animação, o console e o histórico como ilustrado na figura a seguir.

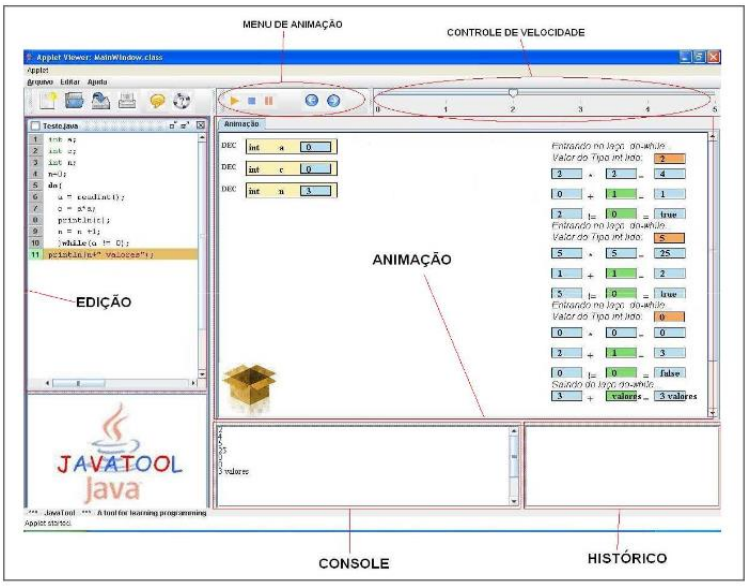


Figura 5 - Interface do Javatool.

O WEB-UNERJOL descrito em Ferrandin, et al (2012), diferente dos outroas ambientes, é focado no ensino *online*, onde o professor pode ensinar programação a distância e acompanhar as submissões de código do aluno usando o interpretador UNERJOL de Portugol *online*. Veja na imagem que a ferramenta se assemelha um pouco com os interpretadores vistos, mas é executada no browser e tem uma interface mais simples.

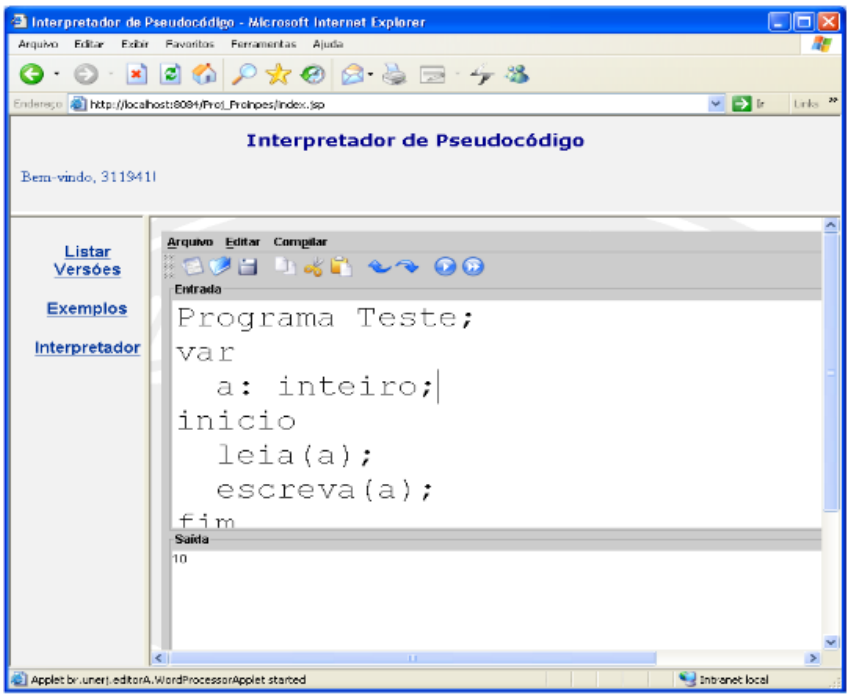


Figura 6 - Interface do Aluno WEB-UNERJOL

O CIFluxProg (de Santiago, et al, 2004) apresenta dois ambientes, um de uso de fluxogramas e outro de Portugol para a resolução de problemas, podendo ser executados, interpretados e compilados, exibindo o resultado final do algoritmo feito.

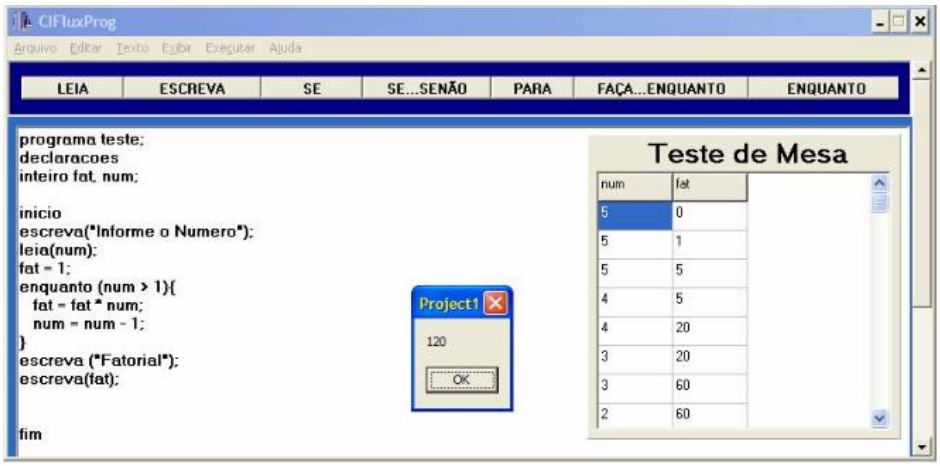


Figura 7- Interface do Portugol no CIFluxProg.

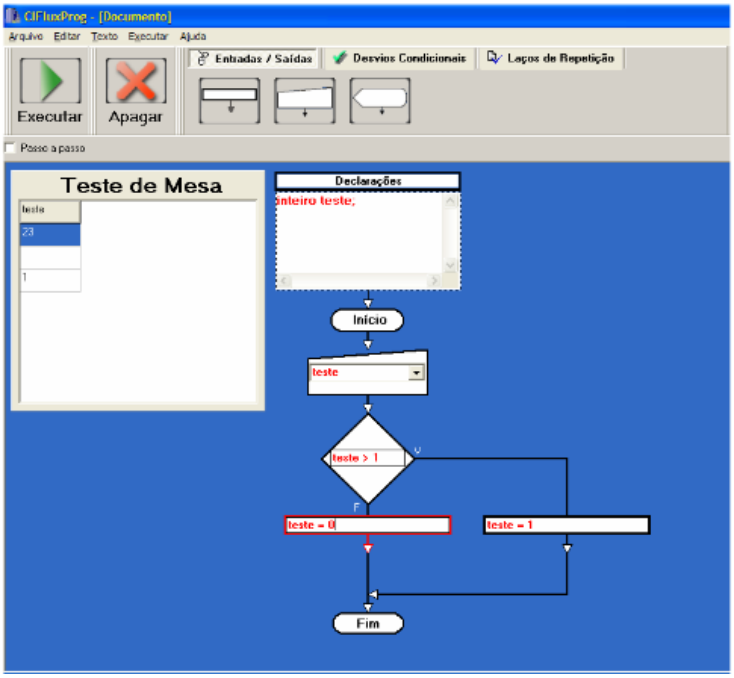


Figura 8 - Interface do Fluxograma no CIFluxProg.

Outro tipo de ferramenta são os STI (Sistemas Tutores Inteligentes) que, segundo Pimentel, et al. (1998) destinam-se a tentar lidar com as carências de conhecimento e pericia dos novos estudantes, usando técnicas de Inteligência Artificial, ambientes construtivistas para exploração e recursos de visualização científica. Eles apresentam medidas cognitivas, como a precisão sintática, checagem de pré-condições e análise de problema.

Ambiente de desenvolvimento para uso do Portugol(Vargas, Martins. 2005), permite o desenvolvimento de algoritmos em uma linguagem de programação estruturada e em português como é mostrado na figura abaixo. Tem por resultado da compilação um código intermediário que pode ser executado passo a passo.

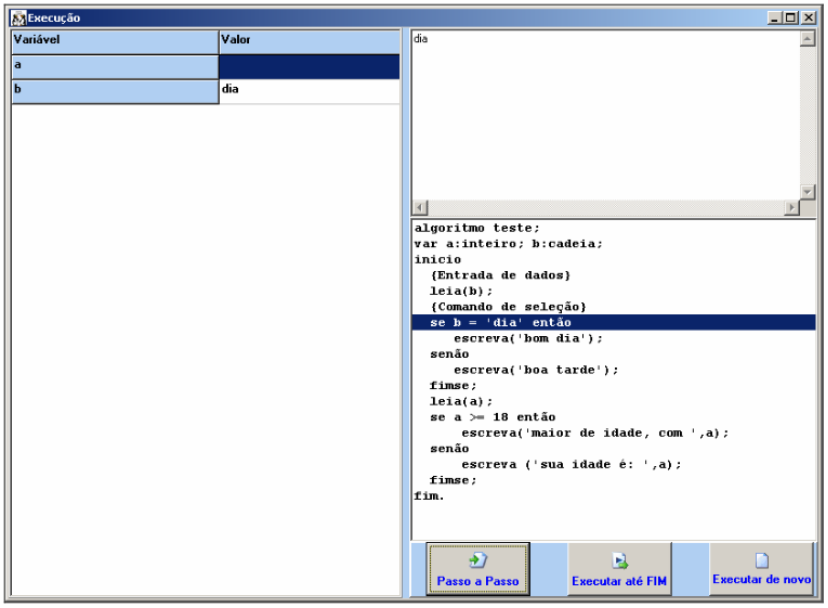


Figura 9 - Execução de Algoritmo no Ambiente de Desenvolvimento em Portugol

A ferramenta Tepequém “é uma Linguagem de Programação imperativa focada no paradigma estruturado, já que a mesma, mesmo podendo ser utilizada em semestres mais avançados, somente tem a finalidade de ser utilizada nos primeiros semestres dos cursos de graduação em computação” (Hinterholz, 2009)

```
algoritmo AtualTec;

variáveis globais
{
    inteiro valor01;
    inteiro valor02;
    inteiro soma;
}

programa principal
{
    imprima("Digite o primeiro valor: ");
    valor01 = leiaInteiro();
    imprima("Digite o segundo valor: ");
    valor02 = leiaInteiro();
    soma = (valor01 + valor02);
    imprimaNL("A soma dos valores é: ", soma);
}
```

Figura 10 - Exemplo da Linguagem Tepequém

Outro tipo de ferramenta encontrada na pesquisa é o jogo educativo. Esse tipo de ferramenta agrega a diversão como elemento motivador no aprendizado.

Como dois exemplos tem-se o jogo Progame, na figura abaixo, apresentado em Sales (2010) e o jogo Castelo dos Enigmas proposto em Scaico (2011) na figura seguinte, desenvolvidos com o intuito de auxiliar o ensino-aprendizagem de programação.

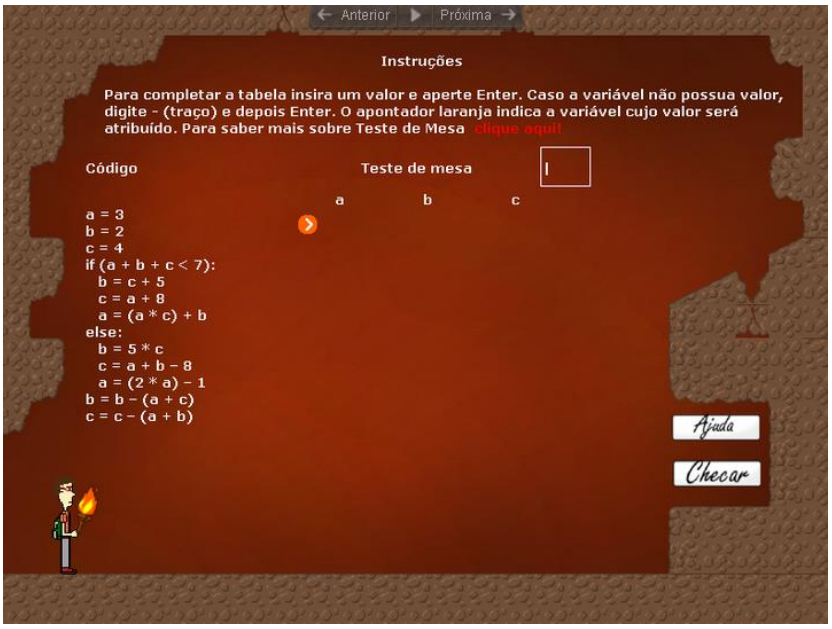


Figura 11 - Tela de um desafio do Progame

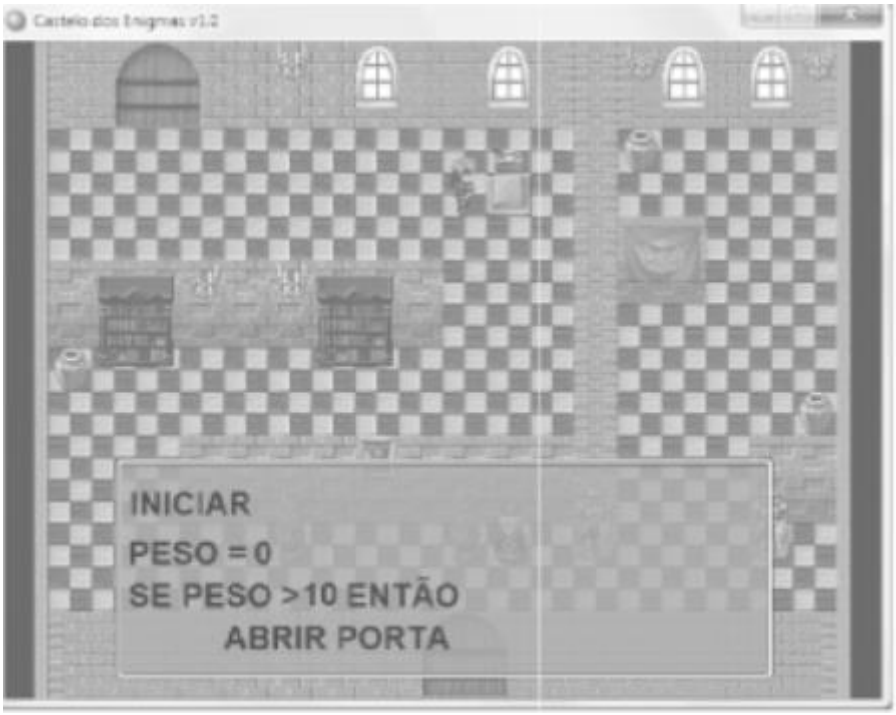


Figura 12 - Problema do Algoritmo da Balança no Castelo dos Enigmas

Para melhor comparar as características das ferramentas de apoio e se os seus propósitos se enquadram na proposta desse trabalho. Foram relacionados na tabela a seguir o tipo de linguagem ou paradigma, o tipo de interação, se existe animações, se tem enfoque no contexto diário ou cotidiano e se o foco da ferramenta são os conceitos iniciais de programação.

A disposição na tabela apresenta algumas características iniciais das ferramentas e se nelas estão presentes os elementos que são abordados nesse trabalho.

	Linguagem/ Paradigma	**Interação**	**Animações**	**Contexto Diário**	**Conceitos Iniciais**
Visualg	Português Estruturado	Codificar, depurar e executar	Não	Não	Não
JavaTool	Java	Codificar, depurar e executar	Sim	Não	Não
WEB-UNERJOL	Portugol	Codificar, verificar versões e executar	Não	Não	Não
CIFluxProg	Fluxograma e Portugol	Codificar e executar	Não	Não	Não
Ambiente Portugol	Portugol e Português Estruturado	Codificar, depurar e executar	Não	Não	Não
Tepequém	Programação Iperativa / Paradígma Estruturado	Codificar, depurar e executar	Não	Não	Não
Progame	Python e Pascal / Paradigma Estruturado	Realizar missões, corrigir códigos	Sim	Não	Sim
Castelo dos Enigmas	Português Estruturado	Realizar missões	Sim	Não	Sim
VDSP		Nenhuma	Não	Não	Não
Learn on Group	Não informado	Colaboração online, submissão e feedback	Não	Não	Não
STI	Não informado	Nenhuma	Não	Não	Não

Tabela 1 - Comparativo das Ferramentas

Essas abordagens, ou em algumas, espécie de simulação sem o uso de analogia não leva em consideração o entendimento prévio do aluno, ou seja, seu

conhecimento anteriormente adquirido. Não levar esse aspecto em consideração pode tornar o processo de assimilação dos novos e distintos saberes da lógica de programação mais lento e dificultoso sem ter algum tipo de analogia que auxilie sua compreensão.

2.3.2 **METODOLOGIAS**

Em busca de metodologias usadas no ensino de programação, foram encontrados alguns citados em Aureliano, et al (2012). Neste trabalho, são listados o uso de jogos, estratégias para melhoria na abstração de resolução de problemas, modelo para avaliação e acompanhamento da aprendizagem, ontologia para auxiliar na revisão de programas, uso de robótica e o uso de blog.

O uso de jogos em Jesus, et al. (2010) é descrito como um experimento em que grupos de alunos foram submetidos ao uso de jogos com desafios de programação e em Marques et al. (2011) o jogo foi tratado como objetivo final da oficina agindo como a motivação para aprender programação, onde na última etapa os alunos usavam os conhecimentos adquiridos para fazer um jogo.

Uma outra abordagem utiliza de estratégias para melhoria na abstração de resolução de problemas. Em Piva (2010), parte-se da leitura do texto para sua representação, depois do desenho para descrição e por fim do problema a sua interpretação.

O modelo para avaliação e acompanhamento da aprendizagem em que Pimentel et al. (2003) integra com o processo de aprendizagem, usou um questionário para levantar informações, aplicação de avaliações, na avaliação uma etapa era somente para ler os enunciados sem responder sendo esta ação realizada na etapa seguinte e por fim uma fase de análise de resultados.

Já Ribeiro et al. (2011), fala da ontologia para auxiliar na revisão de programas em que é determinado um domínio, o reuso de ontologias existentes, enumeração de termos importantes e definição de hierarquias.

O uso da robótica também é mostrado como uma metodologia na pesquisa de Ribeiro et al(2011). É defendido como um experimento orientado a ser feito no final de período para melhor avaliar seu impacto, necessário também avaliar as dificuldades em aplicar os conceitos para a futura programação de um robô, sendo feitas aulas teóricas e aulas práticas para assim realizar as atividades propostas com os robôs.

Como última proposta coletada, encontra-se em Marques et al. (2010) o uso de blogs, onde são apresentados como ambiente de aprendizagem colaborativa, podendo ser baseados em algum problema, uma discussão ou em um projeto, é postado no caso do problema um conteúdo pelo professor onde os alunos serão participantes para soluciona-lo, nos outros, professor e aluno participam para construir o entendimento online.

As metodologias apresentadas tem grande valia no desenvolvimento da aprendizagem dos alunos no que diz respeito ao estudo introdutório de programação de computadores, salvo a abordagem que utiliza de estratégias para melhoria na abstração de resolução de problemas em Piva (2010), todas as outras tem aspectos mais técnicos e podem ser difíceis de aplicar no início das disciplinas de programação.

3 DESENVOLVIMENTO

Neste capítulo serão tratados os itens desenvolvidos de acordo com a pesquisa realizada para promover uma alternativa que apoie o público-alvo em suas dificuldades na disciplina de programação citada.

3.1 PROPOSTA

A visualização do funcionamento ou comportamento de estruturas, bem como conceitos de programação, é uma forma de apresentar as ações realizadas pelo computador de melhor maneira na tentativa de tornar os conceitos abstratos mais concretos através de apresentações visuais, a fim de esclarecer os novos e incomuns conhecimentos aos recém-chegados à área de Tecnologia da Informação.

Como proposta para melhor sanar as dificuldades e entraves ocorridos no ensino de programação, propõe-se a utilização de representações dos conceitos trabalhados na disciplina utilizando-se de situações cotidianas para ilustrar os assuntos de forma mais próxima e familiar, pretendendo tornar a absorção dos conteúdos mais fácil e clara para então relacionar a apresentação feita ao conceito de programação desejado usando de exemplos na linguagem de programação Python.

3.2 CONTEÚDO

Como forma de diminuir as dificuldades encontradas nas disciplinas iniciais de programação, buscou-se destacar os assuntos mais críticos ou mais importantes. Buscou-se delimitar o escopo dos assuntos a serem abordados e quais os objetivos e procedimentos deveriam ser feitos para a criação da analogia de forma compreensível e efetiva. Como um dos assuntos destacados por Lahtinen et all, (2005), o laço de repetição *While* foi o tema escolhido para desenvolver o instrumento.

O comando em questão é caracterizado por realizar repetições de linhas de código enquanto uma condição, inicialmente verificada, não é satisfeita. Seu conceito apresenta difícil abstração para os estudantes. Não é intuitivo pensar em partes do código que devam se repetir e que, geralmente, cada repetição pode

influenciar na próxima iteração. Uma condição de parada deve ser estabelecida. Saber qual condição e como será estabelecida não são ideias triviais e bem compreendidas nos primeiros contatos com a estrutura de repetição.

3.3 ANALOGIA

Para a concepção da analogia, buscou-se uma situação corriqueira que apresentasse momentos com ações semelhantes às apresentadas na estrutura *While*. O relacionamento é feito usando um momento em que uma pessoa prepara café para outras, cada um tem paladar diferente, portanto, a ação de preparo de café se repete, mas a quantidade de açúcar pode ser distinta de acordo com a verificação de sabor feita pela pessoa ao provar o café feito. Assim, pode-se ilustrar as repetições realizadas no comando *While* como a ação de por açúcar e servir o café, a repetição depende da prova da pessoa que com uma feição de repugnância demonstra que o sabor não está ideal e com uma feição feliz, que o sabor está bom.

Para melhor visualização, a tabela a seguir mostrará um quadro comparativo entre a analogia e o comando de programação.

ANALOGIA	CÓDIGO
Ação de provar o café	Entrada de dados
Prova / Verificação do sabor do café	Teste da condição do *While*
Por açúcar no café	Execução feita dentro do *While*
Feição Negativa da Pessoa	Condição Insatisfeita/ Condição Verdadeira
Feição Positiva da Pessoa	Condição Satisfeita / Condição Falsa

O foco da analogia são as características de verificação da entrada na estrutura, a repetição das ações, sua influência coma próxima iteração e quando termina o comando. Apresentando esse cenário, forma-se um paralelo com o conceito de programação, ilustrando em que tipo de momento é válido usar o *While*.

3.4 DESENVOLVIMENTO DA ANIMAÇÃO

A princípio, foram esboçadas as *storyboards* para determinar a melhor sequência de ações para alcançar o objetivo e o ensinamento proposto para as

animações. Para melhorar a sequência da explicação, decidiu-se primeiro iniciar a apresentação com a analogia permitindo o aluno entender o contexto ou exemplo proposto para então mostrar o paralelo feito com o conceito abordado, as relações da alegoria feita com as características da estrutura de programação, mostrando-a em ação para finalizar a linha de pensamento da animação.

Em seguida serão exibidas algumas telas das *storyboards* usadas para desenvolver a animação.

O esquema é composto por um quadro ilustrativo de cada cena, abaixo é apresentada uma descrição e depois que tipo de animação é feita no momento desse quadro. Todas as *storyboards* usadas na animação estão dispostas no apêndice deste trabalho.

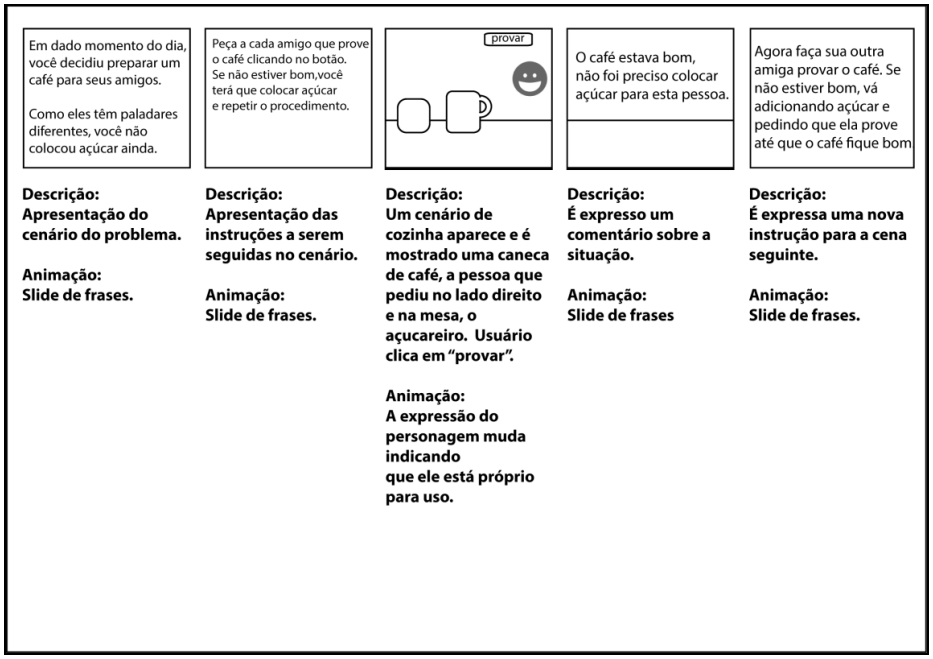


Figura 13 - Imagem do Storyboard 1

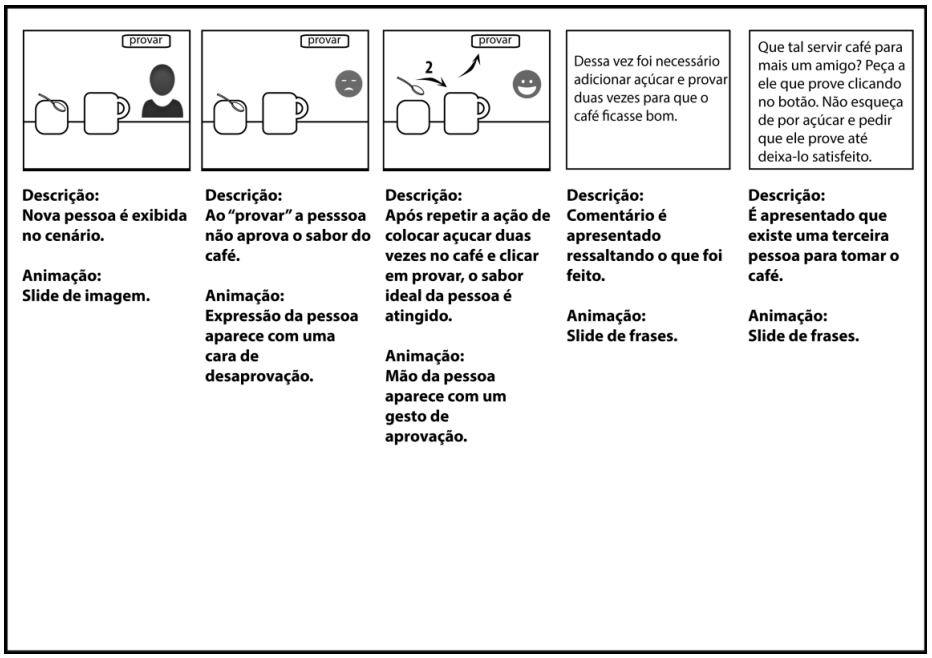


Figura 14 - Imagem do Storyboard 2

3.4.1 **FERRAMENTAS DE DESENVOLVIMENTO**

Como ferramenta de animação, foram encontrados com melhores recursos e resultados o Adobe Flash e o Adobe Illustrator. Mesmo sendo usado por muito tempo para a criação de animações, o Flash está deixando de ser suportado pelas plataformas atuais, preferindo o uso do HTML(*HyperText Markup Language*). Para a animação ser disponível sem problemas de plataforma optou-se pelo uso do Adobe Edge Animate CC, por ser uma ferramenta poderosa na construção de animações e por gerar seus arquivos em HTML na sua última versão. Após criada a animação, o programa exporta o arquivo para HTML5, compatível com os principais browsers, tornando um meio importante para promover a acesso à ferramenta.

O programa conta com uma espécie de palco, onde pode-se compor os elementos usados para animação, um ambiente que apresenta as propriedades, características e opções referentes ao objeto selecionado. Apresenta uma *timeline*(linha do tempo) onde é controlado o tempo, os quadros de animação e como a animação surgirá, também um local onde são elencados os arquivos importados para o projeto.

. Uma vez tendo o escopo formalizado pelas *storyboards*, no Edge Animate cada objeto era dividido e administrado para formar o processo, para criar a interação com o usuário, foi necessário a inserção de comandos em Jquery, linguagem agregada ao programa para eventos de clique do mouse e segurar e

arrastar objetos do cenário.

Na produção, foi utilizado os softwares proprietários Adobe Illustrator CC e o Adobe Photoshop para preparar os elementos gráficos que iriam compor o cenário e as ações realizadas na animação, especificamente ajustes da imagem de fundo, a colher, o bule, a caneca e os personagens.

3.4.2 **ANIMAÇÃO**

A animação caracteriza-se por mostrar primeiramente o cenário da analogia onde o aluno pode interagir com o cenário fazendo parte das ações desempenhadas até a finalização da ilustração. A sequência preza por de início apresentar uma cena conhecida ou familiar para assim fazer o paralelo com o conceito novo. Feito isso mostra-se uma breve revisão do que foi feito, em seguida, apresenta-se a similaridade entre o que foi mostrado com o conceito proposto. Por fim, é apresentado um pequeno código disposto lado a lado com uma simples ilustração do mesmo, enquanto o passo a passo é executado, ao lado uma animação na ilustração é feita. A seguir, estão dispostas imagens da analogia.

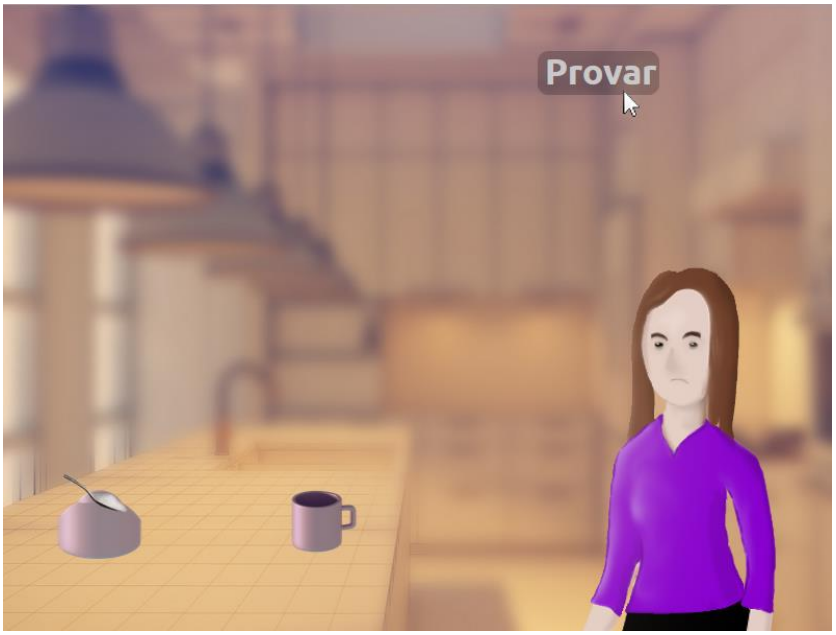


Figura 15 - Animação: Reprovação

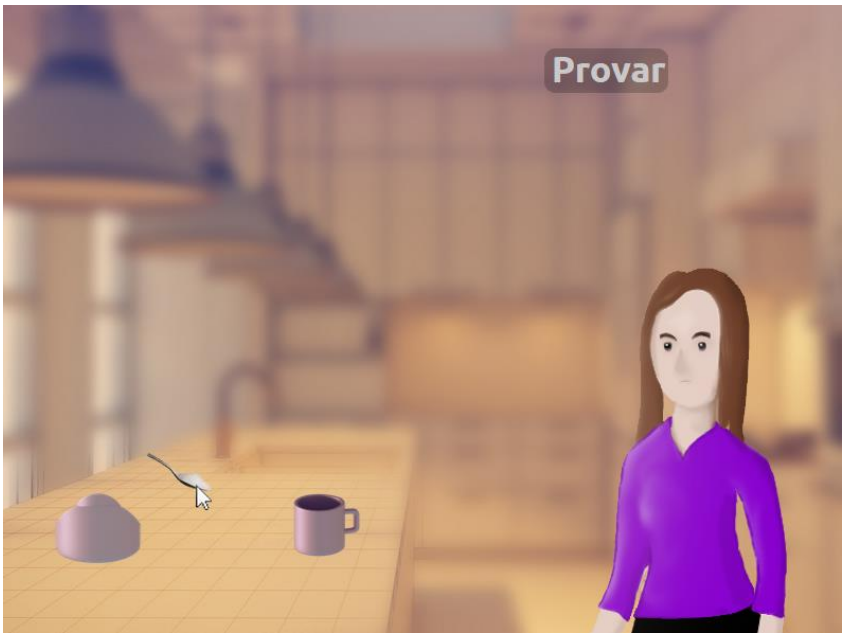


Figura 16 - Animação: Colocando Açúcar

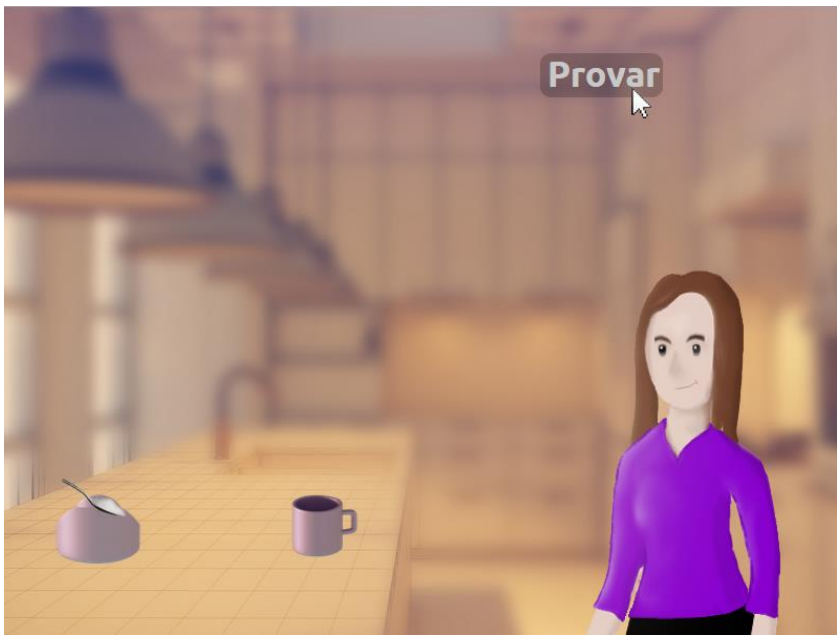


Figura 17 - Animação: Aprovação

Para relacionar a analogia com o conceito proposto mostra-se um algoritmo da analogia apresentada e depois de uma breve simulação de sua execução é explicada a relação com o *While*.

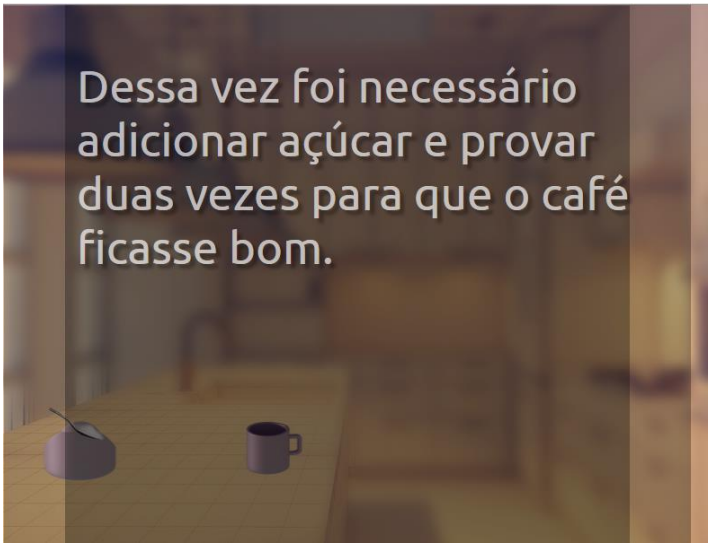


Figura 18 - Animação: Texto Explicativo

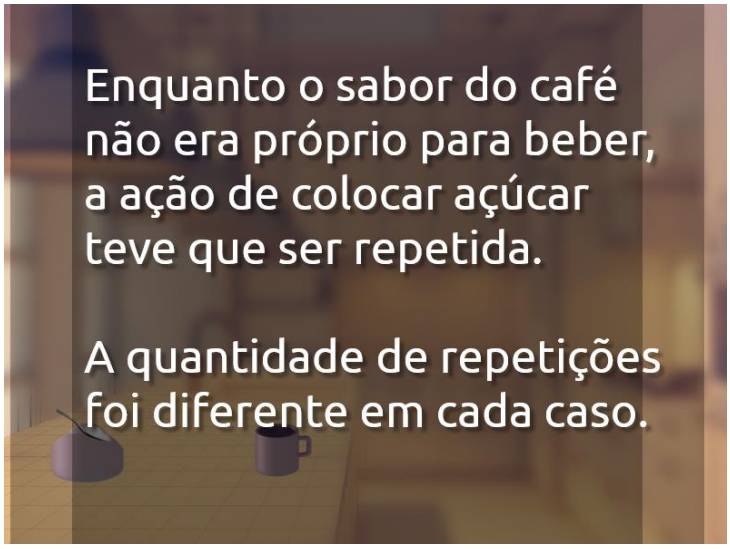


Figura 19 - Animação: Explicação

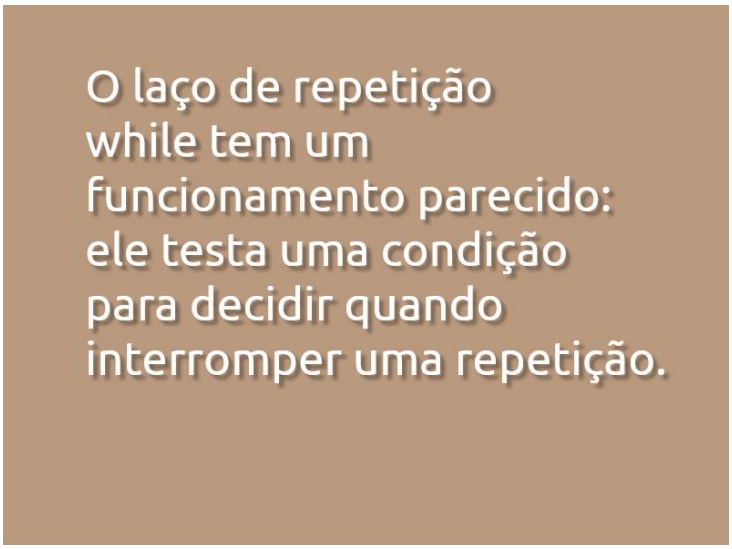


Figura 20 - Animação: Paralelo com o While

Para concluir a animação, é mostrado outro exemplo do comando While, agora com a sintaxe de programação em código Python. É simulada sua execução e explicados seus passos e explicado seu funcionamento para finalizar o raciocínio.

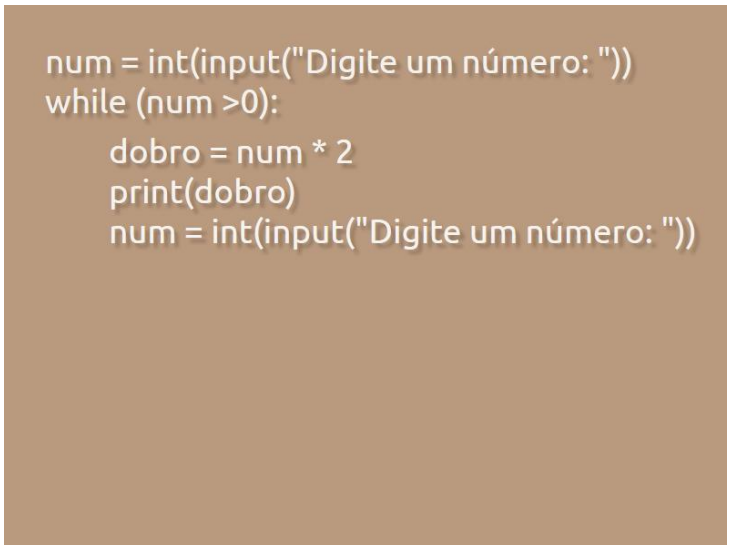


Figura 21 - Animação: Código Exemplo

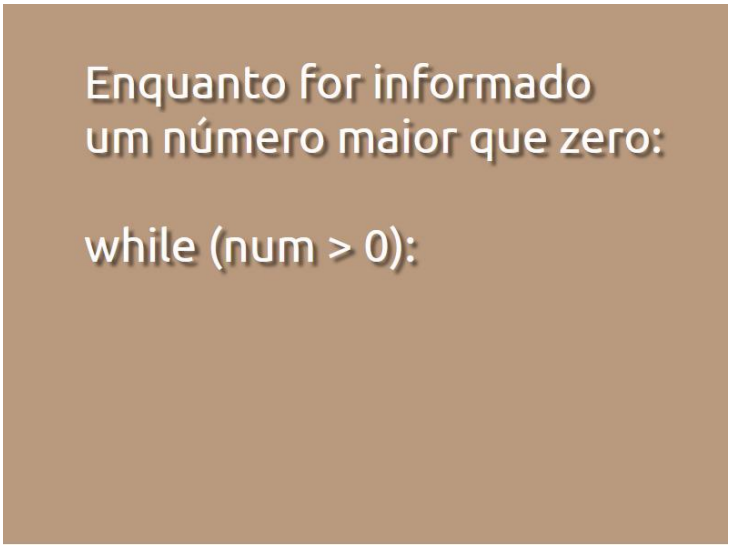


Figura 22 - Animação: Explicação do código

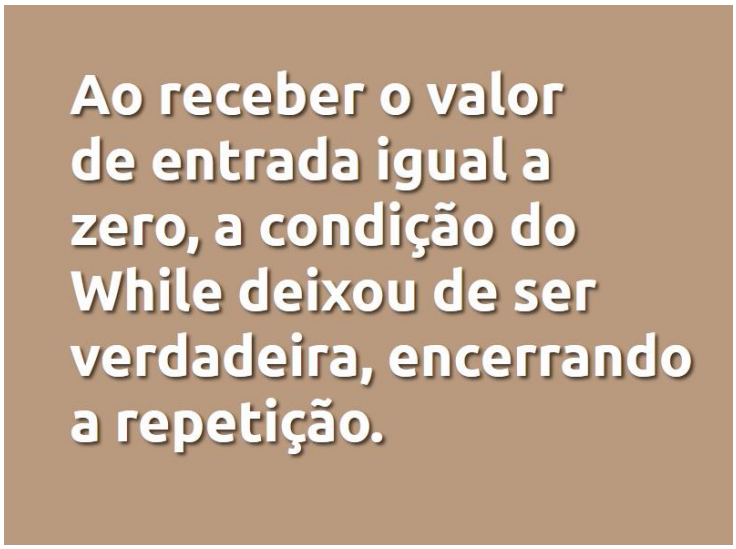


Figura 23 - Animação: Explicação Final

3.5 QUESTIONÁRIO

A fim de avaliar a aceitação do público-alvo, a saber os alunos da disciplina de Introdução à Programação, foi proposto um experimento de uso da animação.

A realização dessa atividade teve como intuito obter *feedback* dos alunos quanto a disponibilidade, aceitabilidade, a similaridade com o assunto destacado e o grau de relevância da animação para seu aprendizado. Pelo prazo em que a animação foi concluída, os alunos estavam no final do período letivo e já haviam visto o comando *While*. Os aspectos importantes a serem destacados com essa experiência são, se o instrumento ajudou no entendimento do conceito de programação abordado, se a animação tornou o aprendizado atrativo e se as situações cotidianas são similares aos conceitos de programação.

Sobre o conteúdo foi perguntado sobre o conhecimento do *While*. Os alunos que participaram da atividade já tiveram aula ministrada sobre o assunto, os gráficos abaixo destacam os resultados encontrados com essa pergunta.

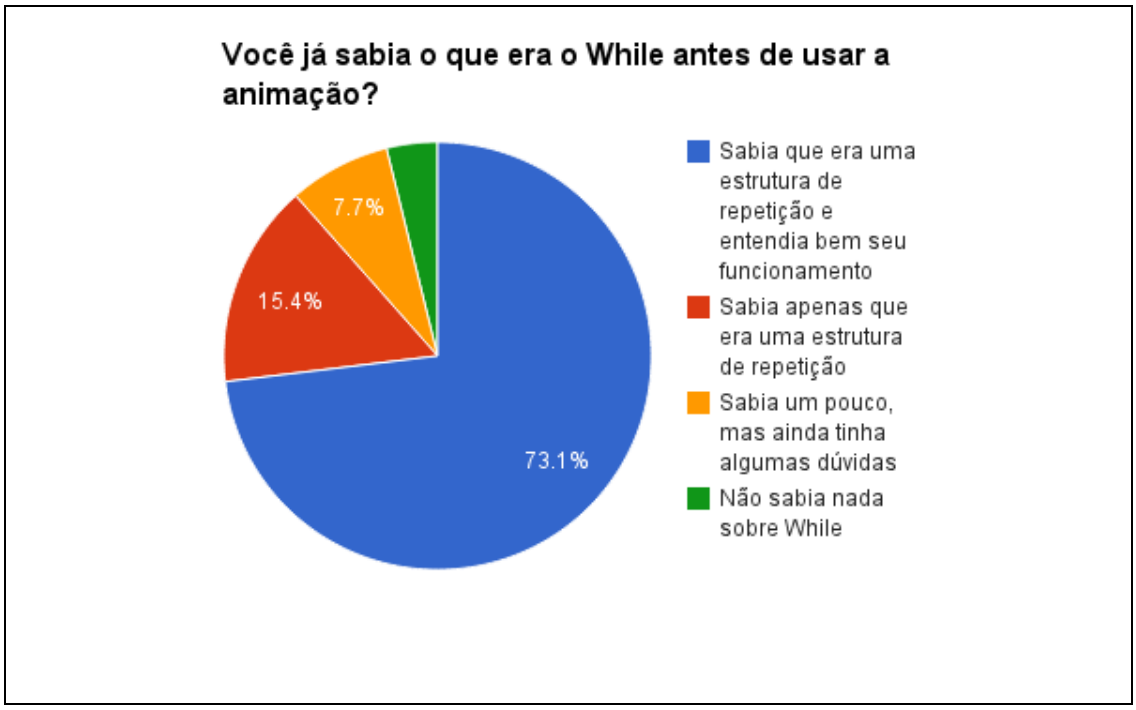


Figura 24 - Gráfico Questão 1

Figura 25 - Gráfico Questão 3

Os alunos, em uma amostra de vinte e seis deles, que participaram de forma voluntária da experiência já tiveram aulas sobre o conceito em questão, assim, a maioria apresentou uma boa base das características do comando. Mas olhando para o primeiro gráfico, um quarto dos alunos não tinham total segurança de seu aprendizado, no terceiro gráfico alguns confundiram o passo executado na estrutura com a condição a ser satisfeita.

Ao apresentar questionamentos sobre a qualidade e aceitação da animação, podendo ser selecionada mais de uma opção no gráfico seguinte, foram gerados os seguintes resultados.

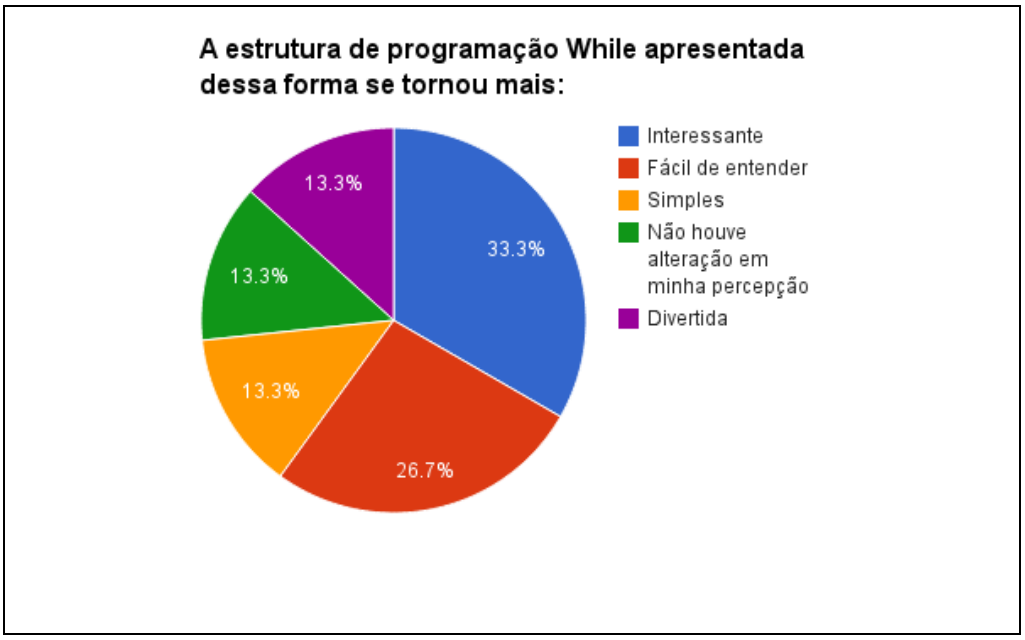


Figura 26 - Gráfico Questão 4

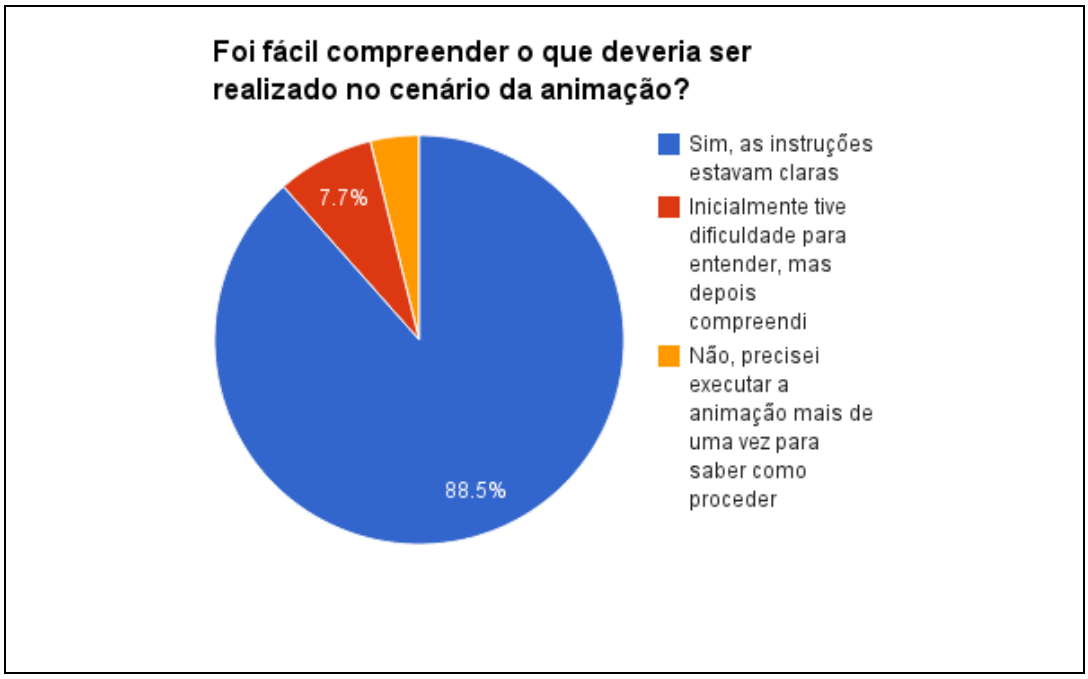


Figura 27 - Gráfico Questão 5

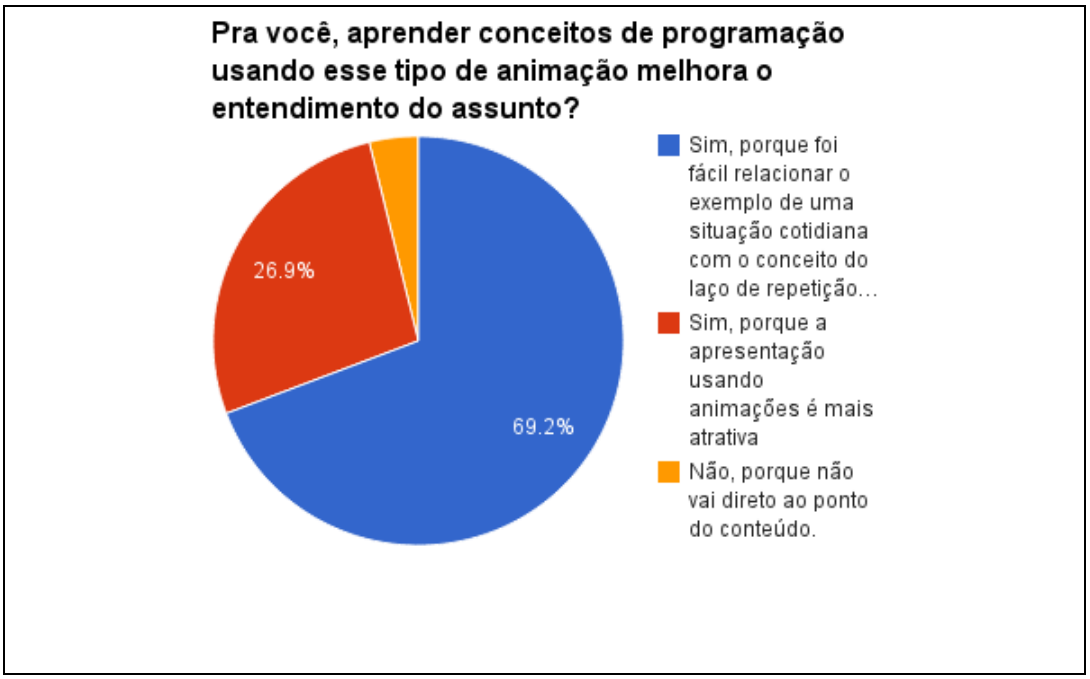


Figura 28 - Gráfico Questão 8

Mais de 85% dos alunos apresentaram boa aceitação à animação, classificando-a como interessante, divertida, simples e fácil de entender. A grande maioria entendeu o que deveria ser feito para interagir com a animação e um pouco menos da metade deles sentiu o ritmo das instruções de texto ideal, os outros divergem entre preferir maior rapidez e mais lentidão.

Quando a disponibilidade, somente uma pessoa necessitou trocar de plataforma, no caso o browser, para que a animação executasse corretamente.

Os alunos demonstraram que a animação ilustrou bem a situação real e esse paralelo foi fator importante para eles apresentarem que o instrumento melhorou seu entendimento e que conceitos de programação apresentados dessa forma trariam benefícios, outro fator abordado por eles é que a animação tornou a experiência mas atrativa. Sobre a similaridade, responderam que a analogia proposta representou bem a funcionalidade do *While* e todos afirmaram que essa semelhança ajudou a entender e perceber as características do comando. Ao ser proposto um grau de semelhança onde um representava que nada na situação proposta era parecida com o conceito de programação e cinco a similaridade máxima, 57,7% dos alunos quantificaram como grau máximo e nenhum apresentou grau um ou dois. Os números no gráfico a seguir representam um grau estabelecido em que o valor 1 representa que não existe semelhança entre as situações e o valor 5 apresenta total relação.

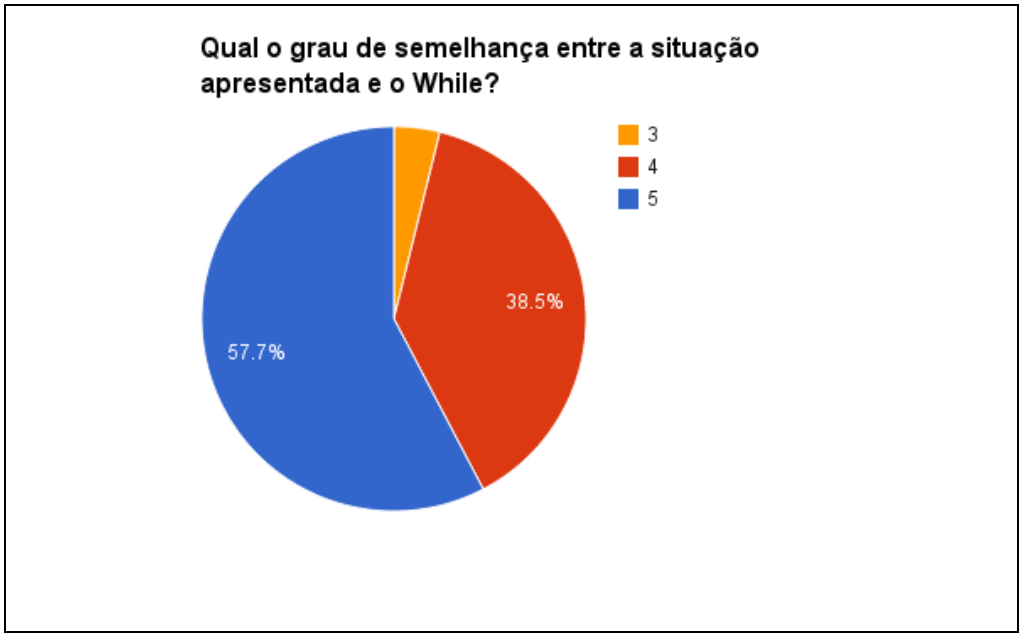


Figura 29 - Gráfico Questão 11

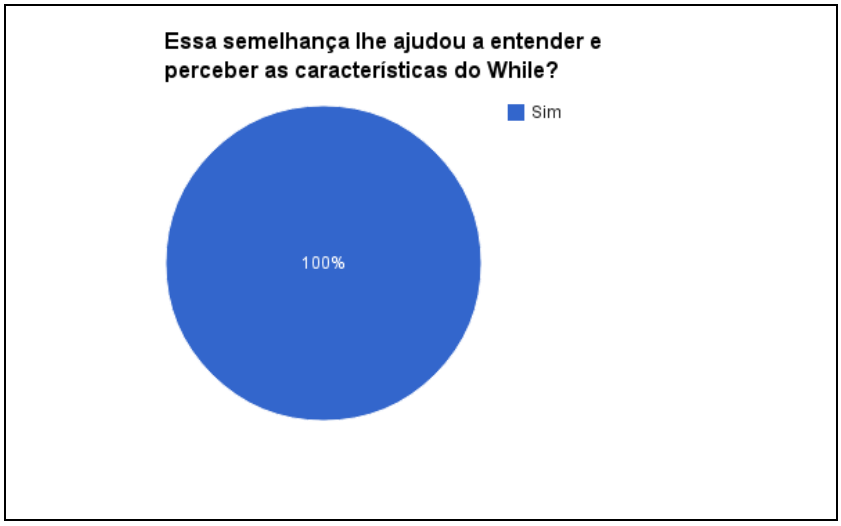


Figura 30 - Gráfico Questão 12

Os resultados obtidos apresentaram boa receptividade dos alunos ao uso das animações como forma de melhor apresentação do conteúdo de programação. Destaca-se que o paralelo entre uma situação de seu dia-dia e as características do conceito apresentado foi bem estabelecida e tornou melhor o entendimento e o instrumento mais aceito para o aprendizado dos alunos.

Em um espaço para comentar sobre a animação, alguns colocaram a animação como útil, apresentando boa percepção do assunto, bem adequada e atrativa. Um deles afirmou que se essa forma de apresentação de conteúdo fosse usada nas aulas, todos aprenderiam com mais facilidade apesar de tomar um pouco

de tempo. As maiores críticas apresentadas foram a respeito do tempo de transição de texto, opiniões bem heterogêneas, mas muitos desejaram que o avançar dos textos fossem controladas por eles e não com um tempo fixo como a animação foi construída, esse ponto, segundo eles traria mais interação e maior controle da apresentação.

4 CONCLUSÃO

4.1 CONCLUSÃO

Este trabalho científico buscou um meio de amenizar as dificuldades encontradas pelos alunos iniciantes nos cursos de Computação na disciplina de Introdução à Programação, frente a índices de reprovações e trancamentos coletados dos cursos da Universidade Federal da Paraíba.

Assim, essa pesquisa teve como propósito apresentar a importância do uso de analogias de circunstâncias vivenciadas no cotidiano utilizando animações como representações dinâmicas como forma de aproximar o raciocínio. Pois mesmo existindo várias ferramentas para melhorar o ensino-aprendizagem desse público, estes não apresentavam características que promovessem um vínculo ou relacionamento entre o conhecimento previamente adquirido do aluno e os novos conceitos apresentados na disciplina.

A concepção de uma analogia, bem como a construção de uma animação que ilustrasse dinamicamente e iterativamente o paralelo entre uma situação cotidiana com características análogas as apresentadas no conceito escolhido apresentou boa aceitação dos estudantes. A experiência promoveu melhor entendimento e assimilação do conteúdo por parte dos alunos e concluiu que a animação tornou o entendimento mais claro, simples e atrativo. O assunto foi bem recebido e gerou interesse dos participantes do uso da animação para que outros assuntos de programação fossem apresentados da mesma forma, alegando que tornaria o entendimento mais fácil.

Este estudo apresentou uma proposta diferente de apresentação dos conteúdos de programação que pode agregar mais valor ao processo de ensino-aprendizagem nas instituições de ensino. O tema mostra-se importante para o estudo e pesquisa sobre o uso de animações e analogias para melhorar a compreensão dos alunos.

4.2 SUGESTÕES DE TRABALHOS FUTUROS

Como expectativas para futuras pesquisas e trabalhos acadêmicos, espera-se a melhoria da animação produzida, agregar mais elementos de interatividade na animação e melhorar o ritmo dos textos explicativos. É importante a realização de outras animações embasadas em novas analogias que apresentem mais conceitos vistos na disciplina de programação. Também é necessária a validação do instrumento com alunos que estejam no início do período antes de terem visto a estrutura de repetição, a fim de verificar a real contribuição da animação para o aprendizado dos estudantes que não tiveram contato com o conceito abordado.

É interessante um melhor estudo da ferramenta de criação das animações e explorar mais recursos multimídia, como o áudio, para serem agregados à animação e também testar a execução da animação nos diversos browsers em versões *desktop* e *mobile* para verificar seu correto funcionamento bem como aplicar a proposta a outras disciplinas de computação.

REFERÊNCIAS BIBLIOGRÁFICAS

Aureliano, Viviane Cristina Oliveira, and Patrícia Cabral de Azevedo Restelli Tedesco. "Ensino-aprendizagem de Programação para Iniciantes: uma Revisão Sistemática da Literatura focada no SBIE e WIE." *Anais do Simpósio Brasileiro de Informática na Educação*. Vol. 23. No. 1. 2012.

Aureliano, Viviane Cristina Oliveira, and Patrícia Cabral de Azevedo Restelli Tedesco. "Ensino-aprendizagem de Programação para Iniciantes: uma Revisão Sistemática da Literatura focada no SBIE e WIE." *Anais do Simpósio Brasileiro de Informática na Educação*. Vol. 23. No. 1. 2012.

Byrne, Michael D., Richard Catrambone, and John T. Stasko. "Evaluating animations as student aids in learning computer algorithms." *Computers & education* 33.4 (1999): 253-278.

Carbonell, Jaime G. *Learning by analogy: Formulating and generalizing plans from past experience*. Springer Berlin Heidelberg, 1983.

Carlson, Paul, Margaret Burnett, and Jonathan Cadiz. "A seamless integration of algorithm animation into a visual programming language."*Proceedings of the workshop on Advanced visual interfaces*. ACM, 1996. Computação, 13, 2005. Anais. São Leopoldo: UNISINOS, 2005. 1 CD-ROM.

Deters, Janice Inês, et al. "O Desafio de Trabalhar com Alunos Repetentes na Disciplina de Algoritmos e Programação." *Simpósio Brasileiro de Informática na Educação* (2008).

Dias, Adelaide Alves. Fundamentos Psicológicos da Educação. In: SILVA, Antônio de A, et al. Licenciatura em Matemática a Distância. João Pessoa: UFPB, 2009

Faria, Eustáquio São José de, Jamil Miranda Vilela, and Juan Manoel Adán Coello. "Um sistema de aprendizado colaborativo de programaçao baseado em agentes chamado learn in group." *XIII Workshop de Educação em Computação (WEI'2005). São Leopoldo, RS, Brasil*. 2005.

Ferrandin, Mauri, and Simone Lilian Stephani. "Ferramenta para o ensino de programação via Internet." *Anais SULCOMP* 1.1 (2012).

Forišek, Michal, and Monika Steinová. "Metaphors and analogies for teaching algorithms." *Proceedings of the 43rd ACM technical symposium on Computer Science Education*. ACM, 2012.

Hansen, Steven, N. Narayanan, and Mary Hegarty. "Designing educationally effective algorithm visualizations." *Journal of Visual Languages & Computing*13.3 (2002): 291-317.

Hinterholz, O. "Tepequém: uma nova Ferramenta para o Ensino de Algoritmos nos Cursos Superiores em Computação." *XVII-Anais do Workshop sobre Educação em Informática*. 2009.
Homepage: <http://www.cinted.ufrgs.br/renote/dez2006/artigosrenote/25157.pdf/>

Jesus, E. A. de; Raabe, A. L. A. Avaliação Empírica da Utilização de um Jogo para Auxiliar a Aprendizagem de Programação. 21° Simpósio Brasileiro de Informática na Educação, João Pessoa, Brasil. 2010.

Kramer, Jeff. "Is abstraction the key to computing?." *Communications of the ACM* 50.4 (2007): 36-42.

Lahtinen, Essi, Kirsti Ala-Mutka, and Hannu-Matti Järvinen. "A study of the difficulties of novice programmers." *ACM SIGCSE Bulletin*. Vol. 37. No. 3. ACM, 2005
.

Marques, A. de M.; Pimentel, M.; Siqueira, S. (2010) Dinâmicas Educacionais com o Uso de Blogs: Requisitos a partir de Experiências. In: 16º Workshop de Informática na Escola, Belo Horizonte, Brasil.

Marques, D. L.; Costa, L. F. S.; Silva, M. A. de A.; Rebouças, A. D. D. S. (2011) Atraindo Alunos do Ensino Médio para a Computação: Uma Experiência Prática de Introdução a Programação utilizando Jogos e Python. In: 17° Workshop de Informática na Escola, Aracaju, Brasil.

Mota, Marcelle Pereira, Lis W. Kanashiro Pereira, and Eloi Luiz Favero. "Javatool: uma ferramenta para ensino de programação." *Congresso da Sociedade Brasileira de Computação. Belém. XXVIII Congresso da Sociedade Brasileira de Computação*. 2008.

Neves, M. de F.; Coello, J. M. A..(2006) OntoRevPro - Uma Ontologia sobre Revisão de Programas para o Aprendizado Colaborativo de Programação em Java. In: 17º Simpósio Brasileiro de Informática na Educação, Brasília, Brasil.

Paula, L., Jr, D., Freitas, R. "A Importância da Leitura e da Abstração do Problema no processo de formação do raciocínio lógico-abstrato em alunos de Computação"CSBC(2009)

Pimentel, Andrey Ricardo, and Alexandre Ibrahim Direne. "Medidas cognitivas no ensino de Programaçao de Computadores com Sistemas Tutores Inteligentes." *Revista Brasileira de Informática na Educaçao (IE)* 3 (1998): 17-24.

Pimentel, Edson P., et al. "Avaliação contínua da aprendizagem, das competências e habilidades em programação de computadores." *Anais do Workshop de Informática na Escola*. Vol. 1. No. 1. 2003.

Piva Jr., D.; Freitas, R. L. (2010) Estratégias para melhorar os processos de abstração na disciplina de Algoritmos. In: 21º Simpósio Brasileiro de Informática na Educação, João Pessoa, Brasil.

Raabe, A. L. A.; Silva, J. M. C. (2005) “Um Ambiente para Atendimento as Dificuldades de Aprendizagem de Algoritmos”. In: Workshop de Educação em

Rapkiewicz, C. et al. Estratégias pedagógicas no ensino de algoritmos e programação associadas ao uso de jogos educacionais. CINTED-UFRGS: Novas Tecnologias na Educação. 2006.

Ribeiro, P. C.; Martins, C. B.; Bernardini, F. C. A Robótica como Ferramenta de Apoio ao Ensino de Disciplinas de Programação em Cursos de Computação e Engenharia. In: 17º Workshop de Informática na Escola, Aracaju, Brasil. 2011.

Sales, Chrystian Gesteira, and Vanessa Farias Dantas. "ProGame: um jogo para o ensino de algoritmos e programação." *Anais do Simpósio Brasileiro de Informática na Educação*. Vol. 1. No. 1. 2010.

Santiago, Rafael, and Rudimar Luís Scaranto Dazzi. "Ferramenta de apoio ao ensino de algoritmos." (2004).

Santos, R., Costa, H., "Análise de Metodologias e Ambientes de Ensino para Algoritmos, Estruturas de Dados e Programação aos iniciantes em Computação e Informática". Infocoomp, Journal of Computer Sciense, v. 5, n. 1, 2006

Scaico, Pasqueline Dantas, et al. "Combinando Diversão e Educação: Castelo dos Enigmas, um Jogo Sério para o Ensino de Algoritmos." *XXII SBIE-XVII WIE* (2011).

Schuvartz, Aguinaldo Antonio. "Ferramenta Computacional de Apoio ao Processo De Ensino-Aprendizagem dos Fundamentos de Programação de Computadores." *Repositório de TCCs-Sistemas de Informação* 2 (2014).

Silva, Italo Fernandes Amorim, Ivanda Maria Martins Silva, and Marizete Silva Santos. "Análise de problemas e soluções aplicadas ao ensino de disciplinas introdutórias de programação." *Universidade Federal Rural de Pernambuco, Recife–PE*.

Souza, Cláudio Morgado. "Visualg-ferramenta de apoio ao ensino de programação." *Revista TECEN* 2 (2009): 1-9.

Vargas, Karly Schubert, and Joyce Martins. "Ferramenta para apoio ao ensino de Introdução à Programação." *XIV Seminário de Computação, Universidade Regional de Blumenau, Santa Catarina* (2005).

ANEXO

UNIVERSIDADE FEDERAL DA PARAÍBA
PRÓ-REITORIA DE GRADUAÇÃO
COORDENAÇÃO DE ESCOLARIDADE

Desempenho dos alunos na disciplina Introdução a Programação:

CIENCIAS DA COMPUTACAO JP -

CODIGO	DISCIPLINA	MATRIC	APROV	%	R_MED	%	R_FT	%	TRANC	%	Periodo
1107136	INTRODUCAO A PROGRAMACAO	85	36	42,35	22	25,88	20	23,53	1	1,18	2013.1
1107136	INTRODUCAO A PROGRAMACAO	87	17	19,54	23	26,44	29	33,33	6	6,9	2013.2
1107136	INTRODUCAO A PROGRAMACAO	97	31	31,96	24	24,74	27	27,84	3	3,09	2012.1
1107136	INTRODUCAO A PROGRAMACAO	80	30	37,5	22	27,5	22	27,5	2	2,5	2012.2
1107136	INTRODUCAO A PROGRAMACAO	77	36	46,75	14	18,18	20	25,97	3	3,9	2011.1
1107136	INTRODUCAO A PROGRAMACAO	75	21	28	24	32	26	34,67	1	1,33	2011.2

CIENCIAS DA COMPUTACAO (LIC.) - LN -

CODIGO	DISCIPLINA	MATRIC	APROV	%	R_MED	%	R_FT	%	TRANC	%	Periodo
8103104	INTRODUCAO A PROGRAMACAO	87	21	24,14	26	29,89	31	35,63	6	6,9	2011.1
8103104	INTRODUCAO A PROGRAMACAO	83	17	20,48	31	37,35	23	27,71	12	14,46	2011.2

UNIVERSIDADE FEDERAL DA PARAÍBA
UFPB
CODESC

8103104	INTRODUCAO A PROGRAMACAO	71	23	32,39	12	16,9	33	46,48	3	4,23	2012.1
8103104	INTRODUCAO A PROGRAMACAO	91	20	21,98	29	31,87	36	39,56	6	6,59	2012.2
8103104	INTRODUCAO A PROGRAMACAO	82	16	19,51	30	36,59	25	30,49	10	12,2	2013.1
8103104	INTRODUCAO A PROGRAMACAO	53	9	16,98	31	58,49	8	15,09	5	9,43	2013.2

SISTEMAS DE INFORMACAO - LN -

CODIGO	DISCIPLINA	MATRIC	APROV	%	R_MED	%	R_FT	%	TRANC	%	Período
8103104	INTRODUCAO A PROGRAMACAO	68	21	30,88	17	25	26	38,24	1	1,47	2011.1
8103104	INTRODUCAO A PROGRAMACAO	33	16	48,48	9	27,27	5	15,15	2	6,06	2011.2
8103104	INTRODUCAO A PROGRAMACAO	62	20	32,26	14	22,58	26	41,94	1	1,61	2012.1
8103104	INTRODUCAO A PROGRAMACAO	24	10	41,67	8	33,33	4	16,67	2	8,33	2012.2
8103104	INTRODUCAO A PROGRAMACAO	57	16	28,07	21	36,84	19	33,33	1	1,75	2013.1
8103104	INTRODUCAO A PROGRAMACAO	7	2	28,57	3	42,86	2	28,57	0	0	2013.2

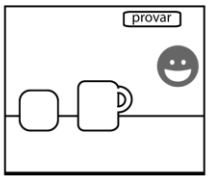
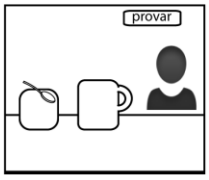
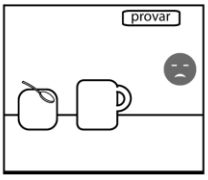
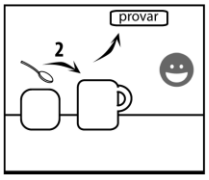
João Pessoa, 23 de outubro de 2014

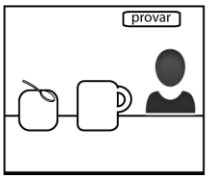
Igor Araújo Alves
Subcoordenador de Programação Acadêmica

Igor Araújo Alves
Sub-Coordenador
Programação Acadêmica
Mat. 1073884

UNIVERSIDADE FEDERAL DA PARAÍBA
UFPB
CODESC

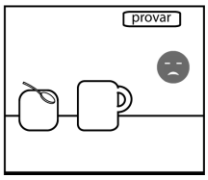
APÊNDICE 1 – Storyboard: While

Em dado momento do dia, você decidiu preparar um café para seus amigos. Como eles têm paladares diferentes, você não colocou açúcar ainda.	Peça a cada amigo que prove o café clicando no botão. Se não estiver bom,você terá que colocar açúcar e repetir o procedimento.		O café estava bom, não foi preciso colocar açúcar para esta pessoa.	Agora faça sua outra amiga provar o café. Se não estiver bom, vá adicionando açúcar e pedindo que ela prove até que o café fique bom.
Descrição: **Apresentação do cenário do problema.** **Animação:** **Slide de frases.**	**Descrição:** **Apresentação das instruções a serem seguidas no cenário.** **Animação:** **Slide de frases.**	**Descrição:** **Um cenário de cozinha aparece e é mostrado uma caneca de café, a pessoa que pediu no lado direito e na mesa, o açucareiro. Usuário clica em "provar".** **Animação:** **A expressão do personagem muda indicando que ele está próprio para uso.**	**Descrição:** **É expresso um comentário sobre a situação.** **Animação:** **Slide de frases**	**Descrição:** **É expressa uma nova instrução para a cena seguinte.** **Animação:** **Slide de frases.**
			Dessa vez foi necessário adicionar açúcar e provar duas vezes para que o café ficasse bom.	Que tal servir café para mais um amigo? Peça a ele que prove clicando no botão. Não esqueça de por açúcar e pedir que ele prove até deixa-lo satisfeito.
Descrição: **Nova pessoa é exibida no cenário.** **Animação:** **Slide de imagem.**	**Descrição:** **Ao "provar" a pesssoa não aprova o sabor do café.** **Animação:** **Expressão da pessoa aparece com uma cara de desaprovação.**	**Descrição:** **Após repetir a ação de colocar açucar duas vezes no café e clicar em provar, o sabor ideal da pessoa é atingido.** **Animação:** **Mão da pessoa aparece com um gesto de aprovação.**	**Descrição:** **Comentário é apresentado ressaltando o que foi feito.** **Animação:** **Slide de frases.**	**Descrição:** **É apresentado que existe uma terceira pessoa para tomar o café.** **Animação:** **Slide de frases.**



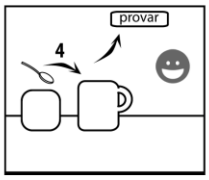
Descrição:
Exibição de uma nova pessoa no cenário.

Animação:
Slide de imagem.



Descrição:
Ao "provar" a pesssoa não aprova o sabor do café.

Animação:
Expressão da pessoa aparece com uma cara de desaprovação.



Descrição:
Após repetir a ação de colocar açucar quatro vezes no café e clicar em provar, o sabor ideal da pessoa é atingido.

Animação:
Expressão da pessoa aparece com uma cara de aprovação.

Dessa vez foram necessárias quatro colheres de açúcar.

Você notou que cada um prefere seu café de um jeito diferente?

Descrição:
Comentário é apresentado.

Animação:
Slide de frases.

Enquanto o sabor do café não era próprio para beber, a ação de colocar açúcar teve que ser repetida.

A quantidade de repetições foi diferente em cada caso.

Descrição:
Explicação sobre a analogia realizada.

Animação:
Slide de frases.

Não havia como prever a quantidade de repetições, pois ela dependia de uma condição: o café não estar mais amargo!
Foi necessário provar várias vezes para saber se estava bom.

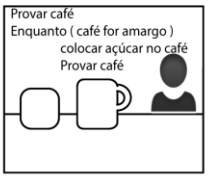
Descrição:
Explicação da analogia.

Animação:
Slide de frases.

Vejamos agora o algoritmo seguido para tomar café.

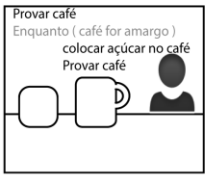
Descrição:
Apresentação do algoritmo da analogia.

Animação:
Slide de frases.



Descrição:
Exibição do algoritmo no cenário.

Animação:
Slide de frases.



Descrição:
Cada linha de código tem sua cor de fonte modificada sequencialmente ao ser executada.

Animação:
Cada estrutura de código
é realçada no momento de sua execução, ao mesmo tempo a ilustração faz movimento correspondente.

Ao tomar o café é verificado se ele está amargo ou não:

Provar café
Enquanto (café for amargo)

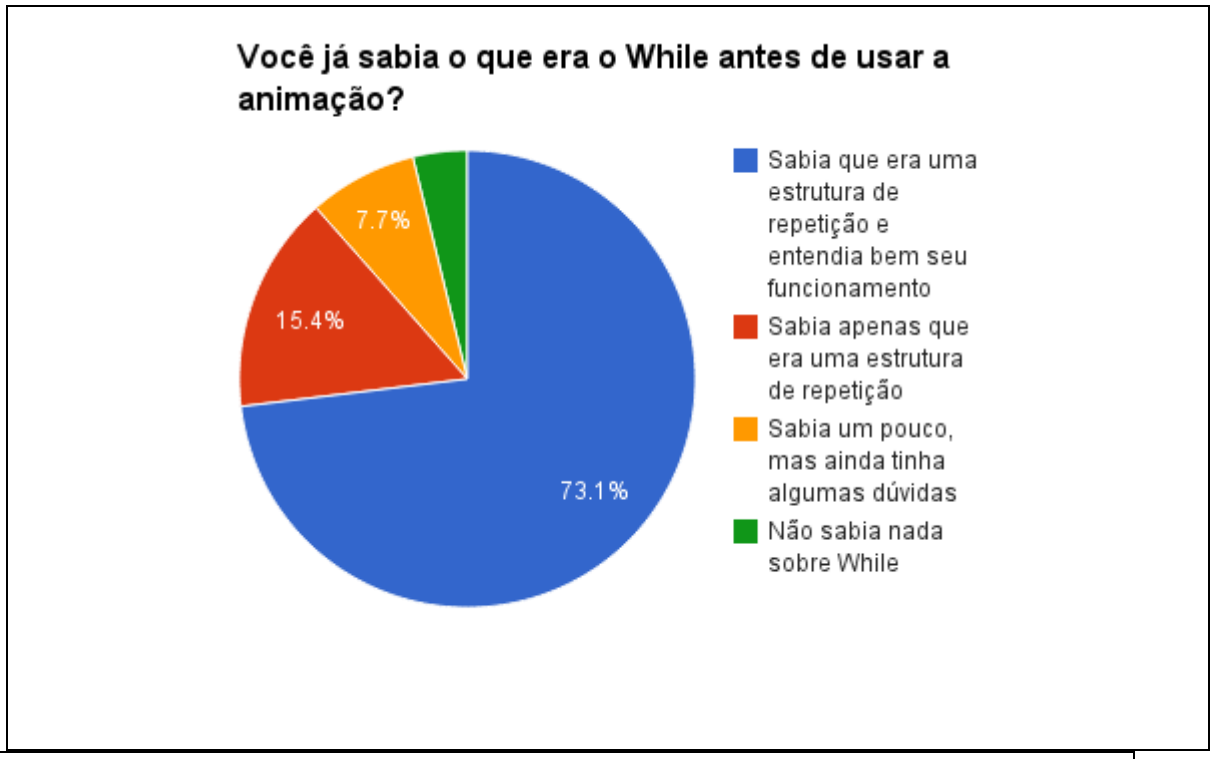
Descrição:
Explicação do algoritmo usada.

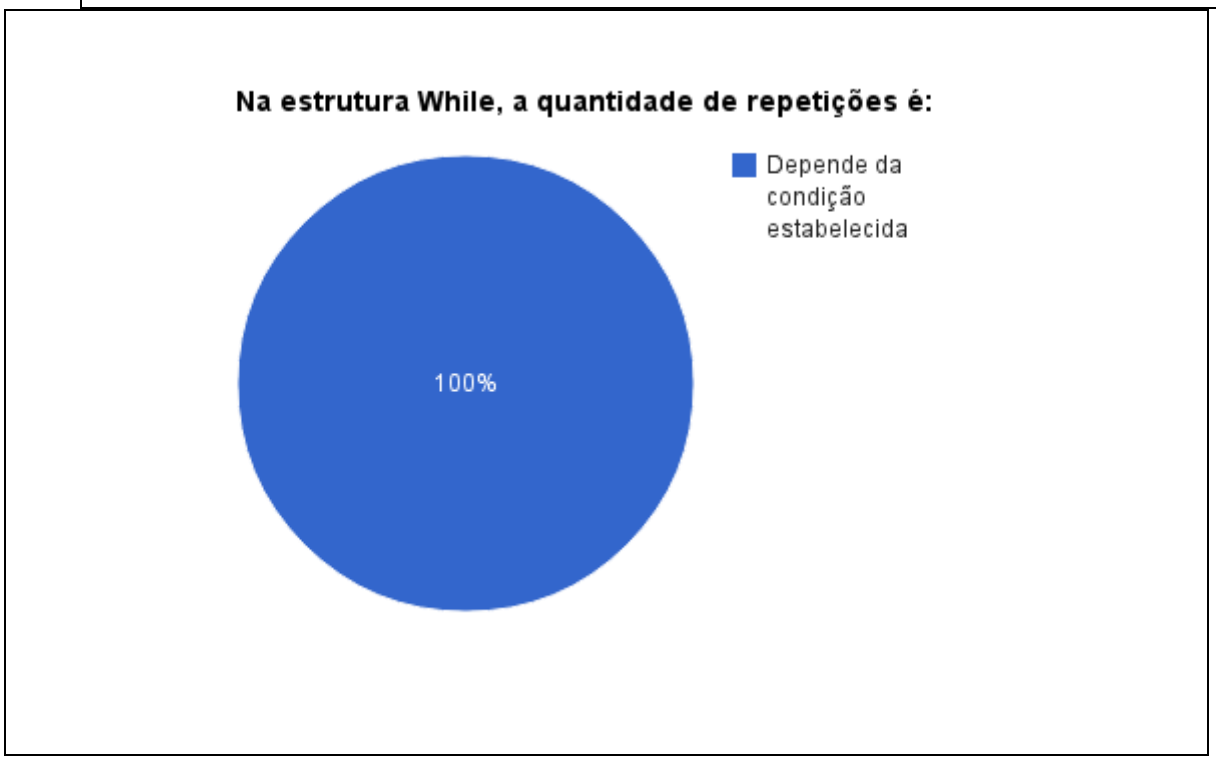
Animação:
Slide de frases.

Enquanto estiver amargo serão feitas as ações: colocar açúcar no café Provar café	O laço de repetição while tem um funcionamento parecido: ele testa uma condição para decidir quando interromper uma repetição.	A cada execução,a quantidade de repetições pode mudar.	Veja um exemplo onde enquanto o usuário digitar um valor maior do que zero, será calculado e mostrado o dobro desse número.	num = int(input("Digite um número: ")) while (num >0): dobro = num * 2 print(dobro) num = int(input("Digite um número: "))
Descrição: **Explicação do algoritmo usado.** **Animação:** **Slide de frases.**	**Descrição:** **Apresentação do While e o começo de sua explicação.** **Animação:** **Slide de frases.**	**Descrição:** **Continuação da explicação.** **Animação:** **Slide de frases.**	**Descrição:** **Apresentação de um exemplo proposto.** **Animação:** **Slide de frases.**	**Descrição:** **Exibição do código While do exemplo.** **Animação:** **Slide de código.**

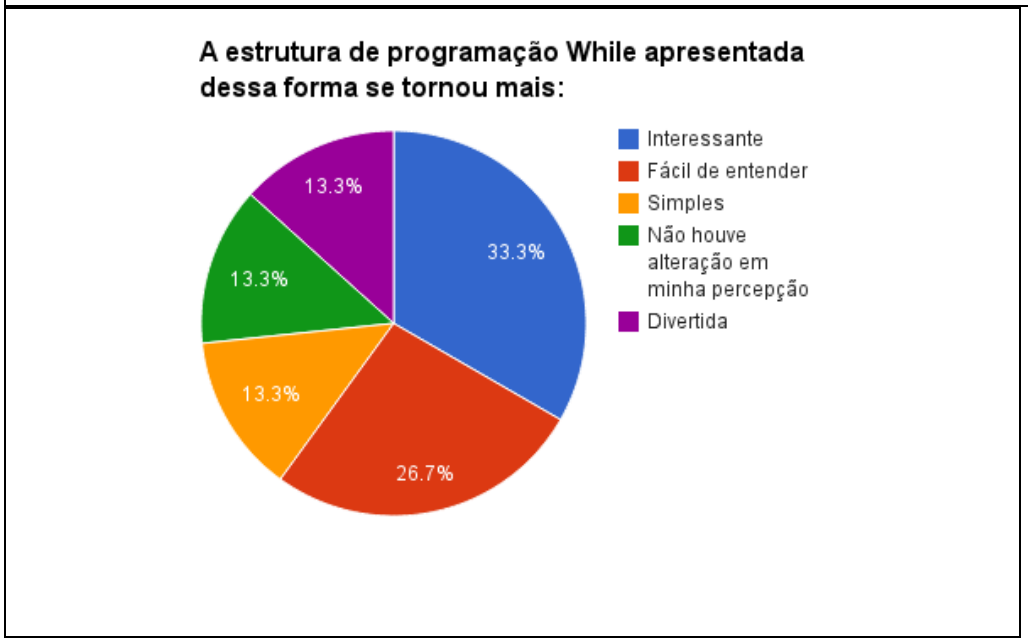
num = int(input("Digite um número: ")) while (num >0): dobro = num * 2 print(dobro) num = int(input("Digite um número: ")) **2**	Enquanto for informado um número maior que zero: while (num > 0):	Os passos seguintes serão executados. dobro = num * 2 print(dobro) num = int(input("Digite um número: "))	Ao receber o valor de entrada igual a zero, a condição do While deixou de ser verdadeira, encerrando a repetição.
Descrição: **Simulação de uma execução do código, abaixo, mostra a entrada e saida de dados abaixo.** **Animação:** **Slide de linhas do código e surgimento dos números de entrada e os de saída.**	**Descrição:** **Explicação do código.** **Animação:** **Slide de frases.**	**Descrição:** **Continuação da explicação.** **Animação:** **Slide de frases.**	**Descrição:** **Explicação final.** **Animação:** **Slide de frases.**

APÊNDICE 2 – Gráficos do Questionário

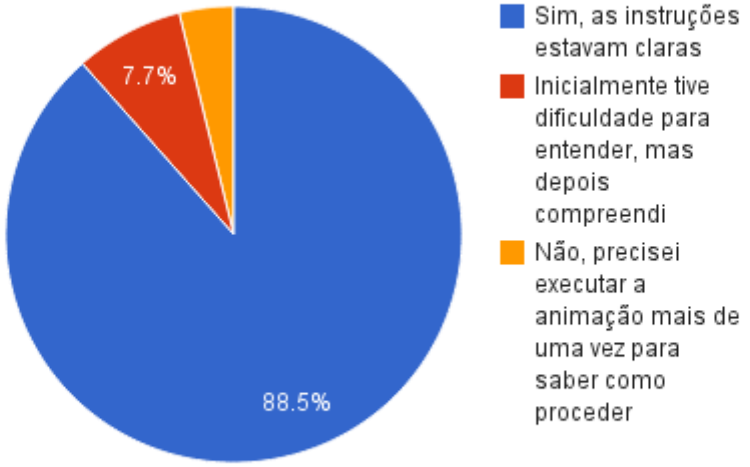




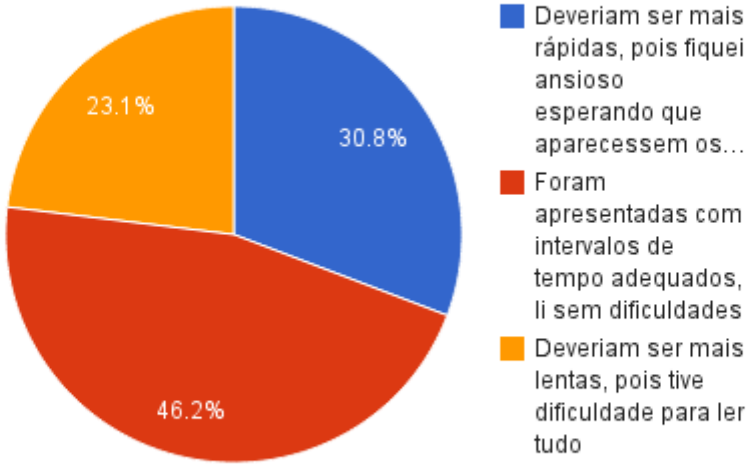

Você entendeu qual condição deveria ser satisfeita na animação?
15.4%
84.6%
Sim, a condição era a quantidade de colheres de açúcar
Sim, a condição era o café não estar amargo.


A estrutura de programação While apresentada dessa forma se tornou mais:
33.3%
26.7%
13.3%
13.3%
13.3%
Interessante
Fácil de entender
Simples
Não houve alteração em minha percepção
Divertida

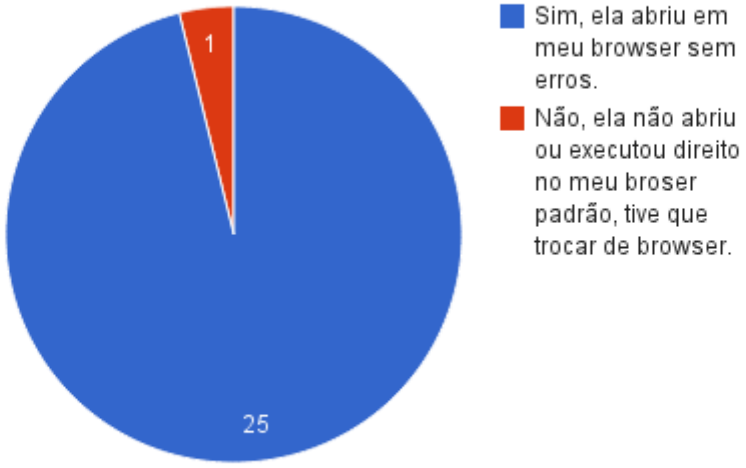
Foi fácil compreender o que deveria ser realizado no cenário da animação?



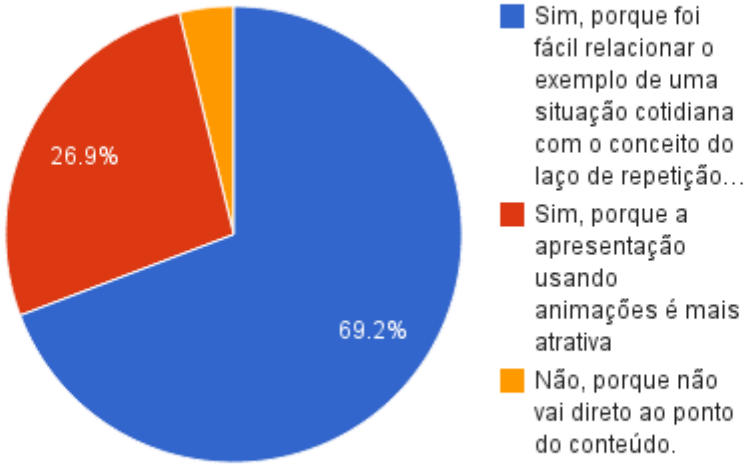
As transações de texto presentes na animação:

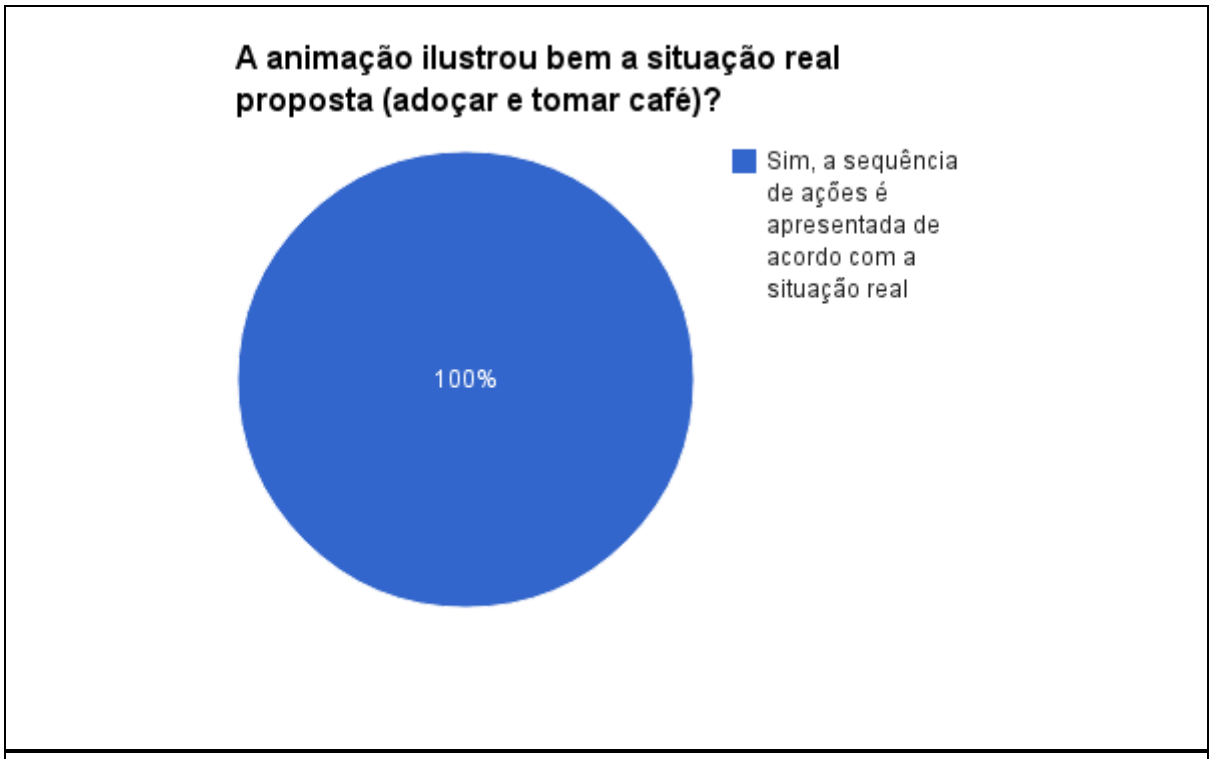


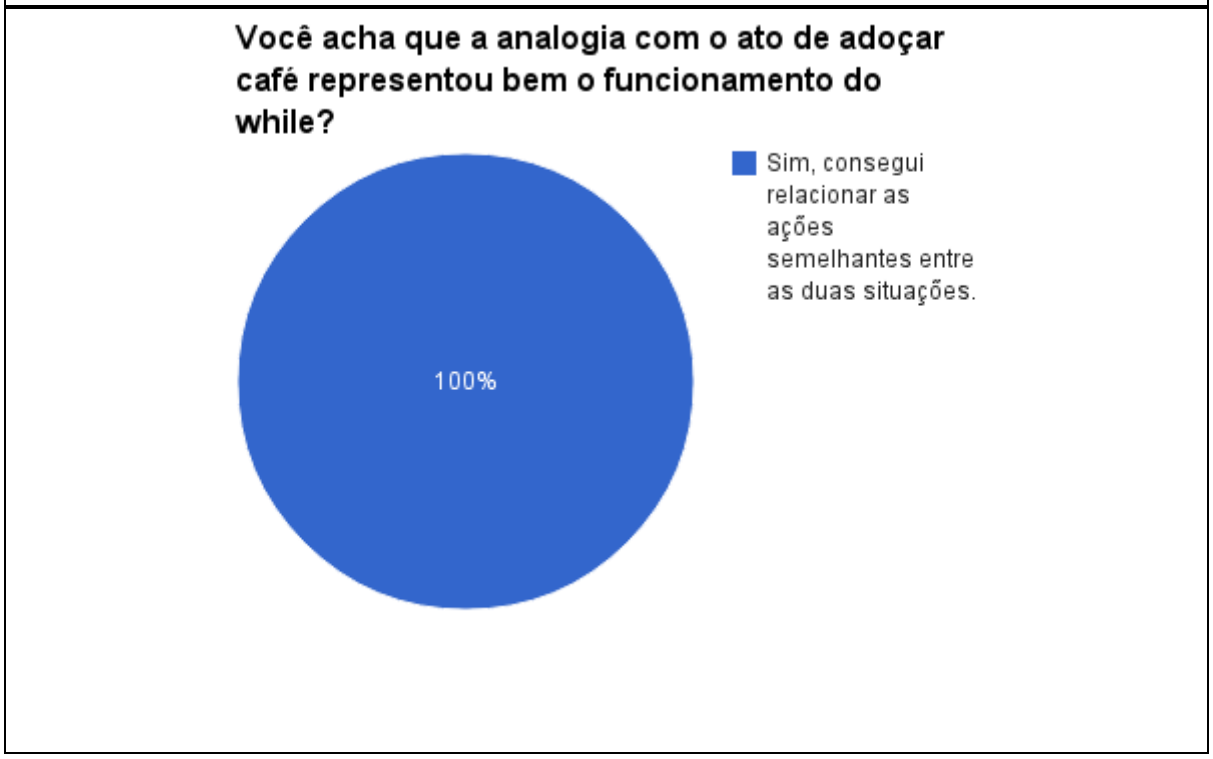
A animação executou corretamente em seu computador?

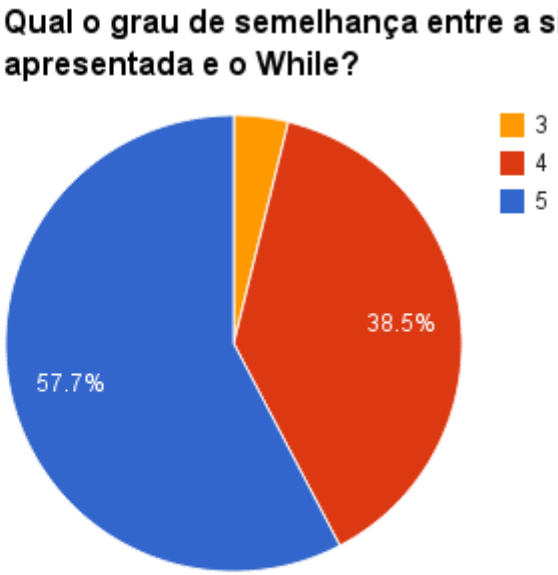


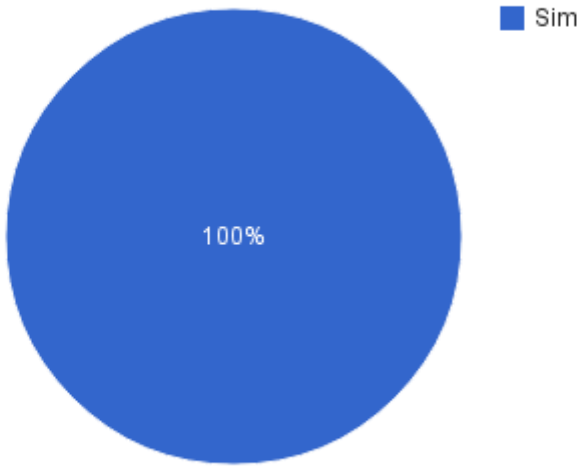
Pra você, aprender conceitos de programação usando esse tipo de animação melhora o entendimento do assunto?




A animação ilustrou bem a situação real proposta (adoçar e tomar café)?
100%
Sim, a sequência de ações é apresentada de acordo com a situação real


Você acha que a analogia com o ato de adoçar café representou bem o funcionamento do while?
100%
Sim, consegui relacionar as ações semelhantes entre as duas situações.


Qual o grau de semelhança entre a situação apresentada e o While?
3
4
5
38.5%
57.7%


Essa semelhança lhe ajudou a entender e perceber as características do While?
Sim
100%