André Luiz França Batista

GUIA PARA ENSINO DE PROGRAMAÇÃO BASEADO EM CONSTRUÇÃO DE JOGOS

Tese submetida ao Programa de Educação Científica e Tecnológica, da Universidade Federal de Santa Catarina, para a obtenção do Grau de Doutor em Educação Científica e Tecnológica.

Orientador: Prof. Dr. José André Peres Angotti.

Florianópolis
2017

Ficha de identificação da obra elaborada pelo autor através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Batista, André Luiz França
Guia para ensino de programação baseado em construção de jogos / André Luiz França Batista ; orientador, José André Peres Angotti, 2017.
133 p.

Tese (doutorado) - Universidade Federal de Santa Catarina, Centro de Ciências da Educação, Programa de Pós-Graduação em Educação Científica e Tecnológica, Florianópolis, 2017.

Inclui referências.

1. Educação Científica e Tecnológica. 2. Ensino de programação. 3. Jogos digitais. 4. Construcionismo. I. Angotti, José André Peres. II. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Educação Científica e Tecnológica. III. Título.

UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO DE CIÊNCIAS FÍSICAS E MATEMÁTICAS
CENTRO DE CIÊNCIAS DA EDUCAÇÃO
CENTRO DE CIÊNCIAS BIOLÓGICAS
PROGRAMA DE PÓS-GRADUAÇÃO
CURSO DE DOUTORADO EM EDUCAÇÃO CIENTÍFICA E TECNOLÓGICA

"Guia para Ensino de Programação baseado em construção de jogos"

Tese submetida ao Colegiado do Curso de Doutorado em Educação Científica e Tecnológica em cumprimento parcial para a obtenção do título de Doutor em Educação Científica e Tecnológica

APROVADA PELA COMISSÃO EXAMINADORA EM 23 DE NOVEMBRO DE 2017

Dr. José André Peres Angotti (Orientador - PPGECT/UFSC): _____
Dr. David Antônio da Costa (Examinador - PPGECT/UFSC): _____
Dra. Viviane Maria Heberle (Examinadora - PPGI/UFSC): _____
Dr. Esteban Walter Gonzalez Clua (Examinador - IC/UFF): _____
Dra. Reane Franco Goulart (Examinadora suplente - IFTM): _____
Dr. Henrique César da Silva (Examinador suplente - PPGECT): _____

Prof. Dr. José Francisco Custódio Filho
Coordenador do PPGECT

André Luiz França Batista
Florianópolis, Santa Catarina, 2017

"*Porque o Senhor dá a sabedoria, e da sua boca vem o conhecimento e o entendimento. Ele reserva a verdadeira sabedoria para os retos; escudo é para os que caminham na sinceridade, para que guarde as veredas do juízo e conserve o caminho dos seus santos*". Provérbios 2:6-8.

A DEUS, pois a sabedoria é um dom divino, não o simples resultado da capacidade ou esforço humano.

"*Sirva, pois, a tua benignidade para me consolar, segundo a palavra que deste ao teu servo*". Salmos 119:76.

A meu avô Alziro Luiz Ribeiro, que me viu iniciar o doutorado, mas nos deixou antes que eu o concluísse. Durante toda sua vida ele se esforçou para que seus filhos e netos estudassem e se formassem, pois acreditava que a educação é um fator essencial na formação humana.

Dedico.

AGRADECIMENTOS

"O SENHOR guardará a tua entrada e a tua saída, desde agora e para sempre". Salmo 121:8.

Agradeço, SENHOR, por me guardar em todos os momentos do Doutorado, desde o início até agora!

SENHOR, abençoe aqueles que, de uma forma ou de outra, também me ajudaram a concretizar este momento tão especial em minha vida:

João Batista e Suzana, pais sempre presentes, pelo carinho, amor, atenção, educação concedida, zelo, cuidado, orações, e torcida, por mim e pelo Adriano!

Giselle, por ser uma esposa maravilhosa, por me ajudar e apoiar em tudo, por cuidar de todos os detalhes, por seu carinho e amor!

Adriano e Brenda, irmão e cunhada, pela amizade, orações, companheirismo, e pelas partidas de *League of Legends*.

Aos meus primos, tios e tias, avôs (*in memorian*) e avós pelos ensinamentos, e pelas experiências compartilhadas de hoje e de sempre.

Professor Angotti, orientador, pelas excelentes orientações, pelo apoio à realização do Doutorado Sanduíche, pelas conversas, por orientar coisas da vida extra-academia, pela amizade sincera, muito obrigado professor!

Docentes e Técnicos do PPGECT/UFSC, agradeço pelo trabalho sério e compromissado com a educação, pela dedicação e presteza nos atendimentos, por contribuir com a qualidade do nosso programa de pós-graduação.

Professor Thomas M. Connolly, foreign advisor who accepted me so well in Scotland, for his contributions to Games-Based Learning research, for his help with the paper, for our great conversations and academic hints. Thank you so much!

Amigos do IFTM, especialmente Rodrigo Grassi, Ailton Siqueira, Getúlio Pereira, Daniel Pimentel, pela amizade sincera que raramente encontramos no local de trabalho, pelas conversas e ideias, e pelos trabalhos em conjunto. Também a professora Reane Franco Goulart pelas valiosas contribuições na qualificação e na defesa.

Instituto Federal do Triângulo Mineiro (IFTM), agradeço pelos afastamentos concedidos para realização do curso de Doutorado, e estágio de Doutorado Sanduíche (processos 23202.000368/2013-20 e 23202.00188/2015-8), pelo excelente ambiente de trabalho e profissionalismo.

Comissão de Aperfeiçoamento de Pessoal de Ensino Superior (CAPES), agradeço pela concessão da bolsa do Programa de Doutorado Sanduíche no Exterior (PDSE) com número do processo BEX 6369/15-4 pela qual tive a oportunidade de realizar parte da pesquisa na Escócia.

Professores presentes na banca examinadora (defesa e qualificação), agradeço o aceite do convite para participarem, e pelas contribuições para com este texto.

Aos meus alunos (ex, atuais e futuros), principal fonte de motivação para a persistência dos meus esforços e pesquisas!

Colegas da turma de Doutorado de 2014, agradeço a todos os amigos que fiz nesta turma, especialmente aqueles com os quais tive mais proximidade, Taíse e Bruno Simões. Obrigado pelas conversas construtivas ao longo destes quatro anos.

A fluência em novas tecnologias envolve não apenas saber usar ferramentas tecnológicas, mas também saber construir artefatos significativos com essas ferramentas.

Seymour Papert, Mitchel Resnick, 1993.

RESUMO

Os avanços da tecnologia digital tornaram a programação de computadores um elemento essencial na formação de profissionais de Tecnologia da Informação (TI). Os métodos de ensino de programação têm se diversificado ao longo dos últimos anos buscando atrair, reter, e motivar estudantes, uma vez que esta disciplina é uma das principais causas de desistência e evasão nos cursos de TI. Uma alternativa para atingir esse objetivo seria basear a disciplina métodos de ensino que explorem e implementem o aprendizado ativo. Como forma de atrair e engajar os estudantes, diversos educadores têm feito uso de metodologias fundamentadas em jogos digitais, devido suas caraterísticas motivacionais. Nesta tese é proposto um guia que harmoniza métodos de ensino construcionistas com desenvolvimento de jogos digitais para o ensino de programação. O percurso teórico-metodológico percorrido para concepção deste guia teve seu início na elaboração de um protótipo do guia que foi devidamente analisado por um conjunto de especialistas em Educação em TI, que por sua vez colaboraram com críticas e sugestões para a otimização deste instrumento. Tais contribuições fortaleceram a concepção do guia otimizado para ensino de programação baseado na construção de jogos digitais utilizando linguagens de programação convencionais. Este instrumento consiste em quatro fases que abrangem a auditoria inicial, o planejamento, o ensino/aprendizagem, e as avaliações/*feedbacks*. Cada fase deste instrumento conta com instruções e sugestões destinadas a auxiliar o professor no processo de implementação desta metodologia de ensino de programação baseada no desenvolvimento de jogos. Para validação deste guia proposto, foi realizado um estudo de caso no qual, de forma experimental, implementamos este guia na prática em sala de aula, afim de verificar possíveis pontos fracos carentes de melhoria, e averiguar a eficiência prática deste material. Por meio de questionários colhemos as impressões e opiniões dos alunos e professor colaborador sobre essa metodologia, e a análise dessas opiniões nos mostra que o interesse dos alunos entrevistados pelas aulas de programação aumentou. Por fim são apresentadas as considerações finais sobre a processo de concepção do guia, sobre as contribuições que este instrumento pode trazer para o ambiente acadêmico, bem como os possíveis caminhos a serem trilhados como pesquisas futuras frutos deste estudo.

Palavras-chave: Ensino de programação. Jogos digitais. Construcionismo.

ABSTRACT

The breakthroughs in digital technologies have made computer programming a paramount aspect in the training of Information Technology (IT) professionals. The teaching methods for programming language education have changed over the years as an attempt to attract, retain and motivate students, once programming is one of the main causes of high dropout rates in IT training courses. One alternative to change this scenario is to organize a syllabus based on teaching methods that explore and focus on the implementation of active learning practices. As a means to attract and engage students, educators have started using methods based on the use of digital games because of the motivating characteristics of these media. In this doctoral dissertation I provide a guide which intends to align constructivist methods with the development of digital games for teaching programming languages. The theoretical and methodological framework for the organization of this guide started with the development of a prototype, which was carefully analyzed by a group of specialists in IT education who provided suggestions and important remarks regarding the optimization of the material. Such contributions provided important input to develop a textbook material for teaching programming languages based on the development of digital games by using conventional programming languages. This resource consisted of four distinct levels of development, being the collection of data for the textbook material the first one, then planning, teaching and learning, and the final feedback for further analysis. Each part of the content of the textbook material provided instructions and suggestions to help teachers in the process of implementing the teaching methodology based on the development of digital games. To evaluate this textbook material, an experimental case study was conducted as a means to use the material in class to assess the positive and negative characteristics of the resource, and to check for its effectiveness. By means of questionnaires we received the feedback about what students and teachers thought about the material. The further analysis of the responses collected from the questionnaires showed that the interest in the classes increased after the adoption of the textbook material. To conclude, final considerations were presented about the development process of the resource, about the contributions of this instrument for an academic setting, as well as the potential perspectives for the improvement of this research.

Keywords: Programming language teaching; Digital games; Constructivism.

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

ABES: Associação Brasileira de Empresas de Software
ACM: Association for Computing Machinery
BRASSCOM: Associação Brasileira das Empresas de Tecnologia da Informação e Comunicação
CAPES: Coordenação de Aperfeiçoamento de Pessoal de Nível Superior
CH: Carga Horária
CRUD: Create, Read, Update, Delete
DCN: Diretrizes Curriculares Nacionais
IEEE: Institute of Electrical and Electronics Engineers
IFTM: Instituto Federal do Triângulo Mineiro
INEP: Instituto Nacional de Pesquisas Educacionais
MEC: Ministério da Educação e Cultura
MIT: Massachusetts Institute of Technology
OO: Orientado a Objetos
POO: Programação Orientada a Objetos
PUC-RIO: Pontifícia Universidade Católica do Rio de Janeiro
PYPL: PopularitY of Programming Language
RBIE: Revista Brasileira de Informática na Educação
RENOTE: Revista Novas Tecnologias na Educação
RF: Referenciais de Formação em Computação
RSL: Revisão Sistemática de Literatura
SBC: Sociedade Brasileira de Computação
SBGAMES: Simpósio Brasileiro de Games e Entretenimento Digital
TI: Tecnologia da Informação
TIOBE: The Importance Of Being Earnest
UFF: Universidade Federal Fluminense
UFLA: Universidade Federal de Lavras
UFMG: Universidade Federal de Minas Gerais
UFPR: Universidade Federal do Paraná
UFRGS: Universidade Federal do Rio Grande do Sul
UFSC: Universidade Federal de Santa Catarina
UFU: Universidade Federal de Uberlândia
UNICAMP: Universidade Estadual de Campinas
USP: Universidade de São Paulo
WEI: Workshop de Educação em Informática

LISTA DE TABELAS

SUMÁRIO

INTRODUÇÃO

Tecnologias da Informação (TI), especialmente mídias digitais e redes de computadores, estão incorporadas em nosso cotidiano e são partes essenciais do nosso processo de produção do conhecimento, de comunicação e de expressão criativa (ALMEIDA; SILVA, 2011). Os computadores – *laptops*, *tablets*, *smartphones* – tornaram-se acessíveis e extensões pessoais de nós mesmos, e a programação destes dispositivos, antigamente apenas o passatempo erudito de uma minoria, está sendo agora reconhecido por educadores e teóricos como uma habilidade crucial, até mesmo uma nova alfabetização (MORAN, 2007, 2013), para os mais jovens (PAPERT, 2008; KAFAI; BURKE, 2015).

Em uma sociedade digital como a que vivemos, o papel da educação se estende para a formação científica e tecnológica na área de TI, uma vez que as Tecnologias Digitais de Informação e Comunicação (TDIC) permeiam – cada vez mais intensamente – quase todas as atividades humanas, incluindo trabalho, lazer, saúde, educação e comunicação (BATISTA; BAZZO, 2015). Aos profissionais dessa área recai a responsabilidade pelo desenvolvimento de parte ou boa parte de soluções, ferramentas e processos inerentes à TI.

Dessa forma, o ensino de programação tem um papel primordial na formação desses profissionais em TI, pois é a base para o desenvolvimento de *softwares*. A unidade curricular de programação (também nomeada de Algoritmos, Estruturas de Dados ou Lógica de Programação) tem fundamental importância devido a sua contribuição na construção do pensamento lógico e abstrato e na concepção de um sólido alicerce conceitual para as demais disciplinas no curso (FEIJÓ et al., 2009; BAYLISS, 2009; CORRAL et al., 2014; SILVA et al., 2015).

Aprender programação de computadores, ou desenvolver *softwares* aplicativos, é o primeiro – e talvez principal – desafio enfrentado por discentes dos cursos da área de TI. Do ponto de vista do discente, esse tópico demanda o desenvolvimento de planos para resolução de questões de ordem lógico-matemática, que de certa forma são abstratas e alheias ao seu cotidiano. Por outro lado, sob a ótica dos docentes, essa unidade curricular requer uma ampla demanda de interação com o intuito de atender, assistir, mediar e avaliar os alunos. Entretanto, há momentos nos quais não é possível atender essa necessidade de interações, seja pela quantidade de estudantes, ou pela variedade de dificuldades e obstáculos que se apresentam. Considerando que, não raro, tais obstáculos de aprendizagem não são identificados e solucionados em tempo hábil, isto pode ocasionalmente refletir em desmotivação, altas taxas de reprovação e, consequentemente, evasão (RAABE; SILVA,

2005; LUDI, 2011; AURELIANO, TEDESCO, 2012; ANDERSON et al., 2014).

Os modos de ensinar disciplinas de programação são diversificados e têm sido foco de vários estudos e pesquisas (SILVA et al., 2015). No método tradicional expositivo, o docente emite seus conhecimentos aos alunos e, posteriormente, apresenta as suas soluções para os problemas propostos. Tal postura, faz com que o aluno absorva tais informações sem análise crítica e, consequentemente, não construa as suas próprias soluções (ROBINS et al., 2003; VIHAVAINEN et al., 2014). Em contrapartida, alguns educadores e pesquisadores têm tentado se desviar dessa metodologia tradicional para tentar superar as barreiras encontradas pelos iniciantes dos cursos de TI. Tais tentativas são muitas vezes referidas como métodos ou intervenções que vão desde o uso de ambientes virtuais de aprendizagem (JUNIOR et al., 2009; PICCOLO et al., 2010), manipulação de kits de robótica (RIBEIRO et al., 2011; SOARES; BORGES, 2011), utilização de jogos digitais (SILVA; NASCIMENTO, 2012; FERRARI et al., 2014), até o emprego de ferramentas de programação visual e simuladores (RAABE; SILVA, 2005; GONDIM et al., 2009).

O uso de jogos digitais na educação é mencionado na literatura como "Aprendizagem Baseada em Jogos", em inglês *Games Based Learning* – GBL (PRENSKY, 2012; CONNOLLY et al., 2007). Recentemente, os jogos digitais se tornaram componentes integradores do nosso ambiente social e cultural (PAPASTERGIOU, 2009; SAVI, 2011), e, além disso, educadores de diferentes áreas vêm demonstrando, nos últimos anos, interesse por jogos, pois esses aparatos têm a capacidade de ofertar benefícios para os processos de ensino e aprendizagem. (GEE, 2007; PRENSKY, 2012).

No particular caso do ensino de programação, existem evidências de boas taxas de sucesso ao utilizar jogos como método alternativo às metodologias tradicionais de ensino (BAYLISS, 2009; REBOUÇAS et al., 2010; SOUZA; DIAS, 2012). Alguns educadores têm explorado as vantagens dos jogos no ensino de programação, porém com uma abordagem que consiste na criação de jogos pelos próprios alunos (SUNG, 2009; BERMINGHAM et al., 2013; MAJGAARD, 2013; WILSON et al., 2013). Esse tipo de abordagem difere do uso tradicional de jogos na educação em que os alunos se aventuram nas atividades propostas pelos jogos e buscam adquirir o conhecimento durante o processo de jogar o jogo. A adoção de uma metodologia na qual os discentes criam e compartilham seus próprios artefatos (neste caso, os jogos) é apoiada por alguns pesquisadores por engajar e motivar

estudantes no processo de aprendizagem, além de incentivar a construção do conhecimento por meio de atividades práticas (PAPERT; HAREL, 1991; KAFAI, 2006). A construção de jogos em sala de aula pelos alunos é uma metodologia rara e pouco empregada, embora haja registros de que essa ideia vem sendo utilizada por quase três décadas (KAFAI; BURKE, 2015).

Dentro deste contexto de criação de artefatos pelos próprios alunos, alguns autores fazem uso de ambientes digitais educacionais específicos, como *Alice*, *Scratch*, *Greenfoot* (RAABE; SILVA, 2005; GONDIM et al., 2009; UTTING et al., 2010; FINCHER et al., 2010) deixando de lado as linguagens de programação convencionais. Tais ambientes, embora muito úteis para iniciantes, não são próprios para construções de *softwares* com nível de complexidade exigido pelo mercado (MSELLE, KONDO, 2014; SUNG, SNYDER, 2014). Sendo assim, é relevante que métodos alternativos focados no uso de linguagens de programação convencionais sejam explorados pelos docentes (BATISTA et al., 2016).

De forma geral, o objetivo principal deste trabalho é uma contribuição original por meio da concepção de um guia para ensino de programação baseado no desenvolvimento de jogos digitais. Este guia, em linhas gerais, é um conjunto de conceitos, procedimentos e diretrizes esquematizados com o propósito de munir professores interessados com orientações para aplicar essa metodologia em disciplinas iniciais de programação.

As formulações das principais questões de pesquisa, o delineamento dos objetivos principais dessa proposta, e as indicações dos métodos e metodologias utilizados, fazem parte do percurso teórico-metodológico que trilhamos para alcançar nosso objetivo.

PROBLEMA DE PESQUISA

O problema de pesquisa está relacionado a como utilizar uma metodologia de ensino que seja capaz de motivar alunos de programação por meio de uma abordagem que não se desvia das linguagens de programação convencionais. O objetivo principal desta tese é a elaboração de um guia que tenha como base a metodologia de aprendizagem baseada em jogos com aspectos construcionistas para disciplinas iniciais de programação. O intento deste guia é munir professores com recursos esquematizados que orientem a utilização de projetos de construção de jogos digitais em suas disciplinas. Tal esquematização de métodos e técnicas de ensino pode favorecer a motivação e engajamento dos alunos sem distanciamentos dos objetivos de aprendizagem.

Sendo assim, tomaremos a pesquisa qualitativa como fio condutor para nossa pesquisa. De acordo com Lüdke e André (1986), esta abordagem qualitativa tem como característica a obtenção de dados por meio do contato direto do pesquisador com conjunto de questões a serem estudados. Com isso, a ênfase é maior no processo do que no produto, empenhando-se em destacar a perspectiva dos participantes, o que a situa mais próxima do objeto de pesquisa. Paralelamente, recorremos a elementos da pesquisa quantitativa com o propósito de harmonizar métodos quantitativos e qualitativos para melhor abordar a questão de pesquisa, sem adotar inflexivelmente a um ou dois paradigmas. Dessa forma, aderimos a uma combinação de elementos de cada um deles, com atributos e métodos quantitativos e qualitativos (CARMO; FERREIRA, 2008; COUTINHO, 2011).

Ao abordar o problema de pesquisa, a elaboração deste proposto guia se fundamentou em pesquisas qualitativas feitas na literatura. Como esclarecido anteriormente, já existem algumas propostas de guias e diretrizes direcionados para atender algumas demandas da pesquisa em aprendizagem baseada em jogos, como desenvolvimento e avaliação de jogos educacionais; orientações para escolha de jogos educativos; e, jogos como principal atividade de ensino/aprendizagem.

A implementação de uma metodologia de ensino precisa ser um procedimento bem alicerçado e bem fundamentado. Diante do exposto, podemos identificar o seguinte problema de pesquisa: O que considerar (conceitos, componentes e estruturação) na elaboração e implantação de um roteiro para disciplinas iniciais de programação?

OBJETIVOS

O objetivo principal desta tese é a elaboração de um guia para implantação de uma metodologia de ensino baseada na criação de jogos digitais nas disciplinas iniciais de programação. Como passo inicial na concepção dessa proposta, a categorização adequada desses itens favorece a efetiva articulação e organização desses componentes dentro do arranjo global do guia, de forma que as relações entre estes elementos necessitam de fundamentações teóricas e metodológicas.

Na configuração final do guia proposto, os componentes mencionados são peças cruciais dentro do todo, e além disso é necessário que todos estejam bem articulados entre si. Para que essas conexões sejam fortes e estáveis, foram investigadas contribuições na literatura com intuito de analisar como cada um desses princípios interage com os outros baseados na experiência e perspectiva de outros pesquisadores.

O guia proposto pode ser usado como roteiro na implantação de uma metodologia construcionista baseada no desenvolvimento de jogos digitais em disciplinas de programação. A sua aplicação pode ser direcionada também para a avaliação. Se o docente já adota em suas aulas uma metodologia semelhante que porventura não aborda alguns dos elementos deste modelo, este pode auxiliá-lo a avaliar e identificar tais carências, e assim supri-las ou ainda reparar algum ponto fraco.

PERCURSO TEÓRICO-METODOLÓGICO

A revisão e análise de obras na literatura foram essenciais para fundamentar teórica e metodologicamente a concepção deste novo guia, e também analisar elementos imprescindíveis que formam parte de um todo. Cada um dos elementos precisa estar solidamente inserido no contexto global do guia proposto. Para isso, a categorização destes componentes – listados a seguir – facilitará a harmonização dos elementos dentro do conjunto.

1) Particularidades da construção de jogos: são características inerentes a jogos simples, com baixo nível de complexidade, para que o tempo médio de desenvolvimento não extrapole o período da disciplina e, consequentemente, não exceda o calendário acadêmico. Além disso, as funcionalidades dos jogos precisam estar alinhadas com os objetivos da unidade curricular;

2) Abordagem educacional construcionista: são fatores que favorecem a interação do aluno com o objeto de estudo, bem como o seu envolvimento com o processo de aprendizado, além de garantir que a prática esteja aliada à teoria. A contextualização e a perspectiva construcionista são elementos presentes nesse tipo de abordagem;

3) Objetivos de aprendizagem: são conceitos gerais de programação que propiciem aos alunos uma visão crítica e sistemática sobre a resolução de problemas e os capacitem no desenvolvimento de algoritmos computacionais, utilizando uma linguagem de programação estruturada.

No processo de concepção do guia proposto foram feitas algumas revisões de literatura descritas com mais detalhes nos capítulos seguintes. Em uma dessas revisões, procurou-se conhecer quais são os pontos principais a serem considerados em uma abordagem de ensino baseada na construção de jogos. Partindo disso, foram selecionados quatro pontos-chave que funcionaram como pilares para elaboração de um protótipo do guia que propomos nesta tese. Com o intuito de realizar uma primeira validação deste guia, esta primeira versão foi compartilhada com um grupo de profissionais de educação em TI, com experiência acadêmica

em ensino de programação, e também uso de jogos na educação. Este grupo de especialistas teve acesso ao protótipo e mediante um questionário direto fizeram suas contribuições teóricas para uma eventual otimização deste guia. Diante dos *feedbacks* e das contribuições, o guia foi otimizado com alguns conceitos organizados de melhor forma, adicionando-se algumas diretrizes e reformulando-se determinadas ideias que careciam de aprimoramentos.

PANORAMA ESTRUTURAL DA TESE

Esta tese está estruturalmente organizada em três capítulos, além desta Introdução, das Considerações Finais, e das Referências e Apêndices.

O Capítulo 1 apresenta um referencial teórico-metodológico sobre a formação de profissionais de TI no Brasil, discorre sobre os métodos de ensino de programação atualmente utilizados e versa sobre os pontos principais da metodologia de ensino/aprendizagem baseada em jogos, além de ressaltar pontos importantes acerca do ensino baseado na construção de jogos.

No Capítulo 2, encontram-se os caminhos percorridos na elaboração do guia, iniciando com seu protótipo, passando pela avaliação de especialistas que contribuíram para otimização do modelo, até chegar à apresentação do guia otimizado. Neste Capítulo o fruto principal dessa tese – o guia para ensino de programação – está detalhado em suas quatro fases, com aplicações e recomendações para sua implementação.

No Capítulo 3, apresentamos o delineamento do experimento realizado para validação do guia. Esta implementação experimental foi importante para analisarmos e discutirmos aspectos práticos concernentes a utilização deste guia em sala de aula. As opiniões e considerações dos alunos e professor foram obtidas por meio da aplicação de dois questionários aos discentes, e um questionário ao docente. Tais opiniões são analisadas e discutidas neste Capítulo.

No último Capítulo manifestamos as nossas considerações finais acerca do processo de construção deste guia de ensino de programação proposto, e também apontamos algumas sugestões para estudos futuros e pesquisas consequentes desta tese.

1 REFERENCIAL TEÓRICO-METODOLÓGICO

Este capítulo traz um retrato sobre a formação de profissionais de TI no Brasil, mostrando a necessidade da manutenção e melhoria da qualidade na capacitação desse quadro técnico. Além disso, traz uma exposição referente aos métodos de ensino de programação de computadores empregados atualmente. Do mesmo modo, é retratado o estado da arte inerente à aprendizagem baseada em jogos e como essa metodologia tem despertado o interesse de educadores e pesquisadores no Brasil e no exterior. Com mais detalhes, é apresentada uma revisão de literatura referente ao ensino baseado na construção de jogos, com a seleção de pesquisas desenvolvidas nessa área nos últimos anos. Por fim, as linguagens de programação convencionais e os ambientes de programação visual com blocos são colocados lado a lado em um cenário comparativo expondo suas diferenças, similaridades, objetivos e funcionalidades.

1.1 FORMAÇÃO DE PROFISSIONAIS EM TECNOLOGIA DA INFORMAÇÃO NO BRASIL

Os cursos superiores precursores da área de Computação foram instituídos no fim da década de 1960, quando algumas universidades adquiriam computadores, começando assim a formação de recursos humanos e nível superior na área de TI. Pioneiros, os primeiros cursos de graduação plena em computação foram Ciência da Computação, criado em 1969 na Unicamp, e Processamento de Dados, também em 1969 na Universidade Federal da Bahia (CABRAL, 2008).

Conforme relembra Cabral (2008), os cursos superiores de tecnologia, também denominados Cursos Tecnólogos em Processamento de Dados, proliferaram a partir do início da década de 1970, e se assemelhavam aos cursos de Análise de Sistemas existentes hoje. Tais cursos eram regidos por um currículo reduzido e com isso não permitiam o acesso dos seus egressos à pós-graduação *stricto sensu*. Contudo, por representarem a maior parcela dos cursos da área de Computação no país, muitas universidades, contrariando a legislação vigente, aceitavam egressos desses cursos em seus programas de pós-graduação (CABRAL, 2008).

A Sociedade Brasileira de Computação (SBC) disponibiliza atualmente em seu portal algumas estatísticas com dados dos cursos da área de Computação no Brasil. Tais dados são obtidos a partir do censo da educação superior realizado pelo Instituto Nacional de Pesquisas Educacionais do Ministério da Educação e Cultura (INEP/MEC). As

estatísticas mostram o estado quantitativo da Educação Superior em Computação no país desde 2001, vistas a partir do ano 2015.

Os cursos da área de TI no Brasil são categorizados em modalidades diversas. Embora tais modalidades tenham como principal objeto de estudo a Computação e suas aplicações, cada modalidade foca na formação de um perfil profissional diferente. A Tabela 1 mostra a distribuição desses cursos, e a evolução em 2015 quando comparada com o ano de 2014.

Tabela 1: Distribuição dos cursos na área de Computação

Modalidade de Cursos	**2014**	**2015**	**Evolução (%)**	**Panorama 2015 (%)**
Ciência da Computação	366	362	-1,09	15,11
Engenharia de Computação	172	187	8,72	7,80
Sistemas de Informação	604	591	-2,15	24,67
Licenciatura em Computação	91	103	13,19	4,30
Engenharia de Software	17	22	29,41	0,92
Curso Formação Específica	13	9	-30,77	0,38
Cursos de Tecnologia (Todos)	1068	1091	2,15	45,53
Outros Cursos	35	31	-11,43	1,29
Total	**2366**	**2396**	**1,27**	**100,00**

Fonte: Estatísticas, SBC (2016).

A Tabela 1 mostra as diferentes modalidades de cursos na área de Computação no Brasil. Observa-se um pequeno crescimento (1,27%) em relação ao número total de cursos, liderados pelos expressivos aumentos na quantidade de cursos de Engenharia de Software e Licenciaturas em Computação. SBC (2016) alerta que as estatísticas foram construídas a partir de uma base de dados especialmente fornecida pelo INEP, inerente ao censo do ano de 2015.

Diferente dos primórdios, quando os computadores eram privilégios de pouquíssimas universidades, hoje deixaram de ser ferramentas exclusivas de laboratórios universitários. Sua presença é percebida em praticamente todos os setores da sociedade, porém constatamos isso com mais profundidade nos onipresentes dispositivos móveis e telas sensíveis ao toque que executam aplicações digitais desenvolvidas para atenderem às necessidades pessoais e profissionais do indivíduo.

Atualmente, a necessidade do profissional no mercado de trabalho é possuir conhecimento prático para desenvolver suas potencialidades e assim atingir sua excelência. Neste contexto é que se destacam os cursos

de formação de técnicos e tecnólogos, que aliam o conhecimento científico com a aplicação prática necessária para que os profissionais contribuam efetivamente com a crescente inovação tecnológica no âmbito organizacional (MOLL, 2010).

Com função fundamental no direcionamento do ensino de computação no Brasil, a SBC na figura de seus associados foi responsável pelas discussões sobre como os cursos de graduação em TI deveriam ser conduzidos nas últimas décadas (ZORZO et al., 2017). Dentre as ações realizadas pelas SBC destacamos a constituição de comissões para elaboração de Currículos de Referência, também as discussões sobre as formas de avaliação destes cursos junto ao Ministério de Educação. Tomando como base estes currículos e discussões, foram elaboradas as Diretrizes Curriculares Nacionais (DCNs), homologadas em novembro de 2016, por meio da Resolução número 5 de 16/11/2016 (MEC, 2016).

Fruto destas discussões promovidas pela SBC, foi elaborado um conjunto de documentos que auxiliassem os coordenadores de curso de graduação na elaboração de projetos pedagógicos (ZORZO et al., 2017). Chamados de "Referenciais de Formação em Computação" (RF), estes documentos foram elaborados de maneira livre, com o objetivo de conciliar as características particulares de cada curso, buscando atender aos seguintes princípios básicos: (1) Estar alinhado com as DCNs; (2) Seguir um modelo baseado em competências.

Acerca das transformações atuais no mercado do trabalho, algumas questões têm ganhado destaque, a saber: a introdução das inovações tecnológicas no processo produtivo e as mudanças técnicas e organizacionais por elas motivadas; e o aprimoramento da formação profissional de recursos humanos, de modo a atender adequadamente a essas transformações. As questões estão relacionadas entre si, especialmente, em virtude das demandas atuais e futuras de recursos humanos para lidar com essas mudanças motivadas pelas inovações tecnológicas.

O principal agente causador dessa crescente inovação tecnológica é o setor da TI. Esse ramo permeia praticamente todos os outros setores da sociedade. Em sua obra, Carr (2008) aponta que o computador vem transformando o setor empresarial desde a sua criação na segunda metade do século XX. É possível identificar uma intensa aceleração na produtividade em diversos aspectos da sociedade. Nas palavras do autor, "O poder e a presença da TI se expandiram e as empresas passaram a considerá-la como um recurso cada vez mais decisivo para o seu sucesso." (CARR, 2009, p. 3).

Não apenas as organizações foram afetadas, mas a vida e a rotina do ser humano também foi – e continua sendo – fortemente impactada pelos avanços diários da TI. Atualmente, encontramo-nos em uma sociedade que enfrenta intensas transformações proporcionadas pela modernidade, especialmente, pela revolução tecnológica que encurta distâncias e estabelece interação entre as culturas, tornando o mundo em uma aldeia global, que nas palavras de Castells (1996, 1999) é uma sociedade "informacional".

A TI é hoje a força motriz por trás desta sociedade informacional. As mudanças trazidas por essa sociedade e a tecnocultura são combinadas para mobilizar transformações na mídia, na política, na economia e até na própria educação formal (KAHN; KELLNER, 2007, p. 434).

A formação de pessoal tecnicamente competente no campo da TI é essencial para o progresso de uma sociedade bastante conectada como a nossa. Relatórios e estudos mostram que no Brasil existe uma carência de profissionais qualificados com competências e habilidades nessa área. O Índice Brasileiro de Convergência Digital, BRASSCOM (2012), mostra que a educação em TI no Brasil tem alguns desafios a serem enfrentados e vencidos:

> O grande desafio do Brasil nesse sentido consiste em adotar um modelo de educação que dialogue com a realidade da sociedade e que esteja integrado com as vocações de desenvolvimento do País. Trata-se, sobretudo, de avançar nos seguintes aspectos: a) melhorar a infraestrutura e o acesso às tecnologias das escolas; b) formar professores mais qualificados; c) criar incentivos para o exercício da pedagogia (formação, condições de trabalho, incentivos, remuneração adequada, etc.); d) ampliar a formação profissional; e, e) criar boas referências a partir de experiências pedagógicas bem sucedidas (integração de tecnologias à plataforma de ensino) para transformar em modelos de políticas públicas (BRASSCOM, 2012, p. 96).

Assim, temos claro que o atual modelo de educação tecnológica em cursos de TI não atende completamente às necessidades do mercado brasileiro desse setor. É necessário não apenas aumentar a oferta de cursos, como também melhorar e aprimorar o ensino, a construção de saberes e o desenvolvimento de habilidades científicas e tecnológicas dos estudantes de cursos dessa área.

Um estudo feito pela Associação Brasileira de Empresas de *Software*, ABES (2012), mostrou que o Brasil possui uma carência de profissionais de TI. Os dados e resultados revelam que os profissionais desse setor no país encontram um mercado de trabalho em crescimento e com baixa competição qualificada para vagas em aberto. Os pesquisadores ainda alertam para um agravamento na carência por profissionais de tecnologia no país nos próximos anos.

Os resultados indicam que em 2012, havia no Brasil uma carência de aproximadamente 40 mil profissionais qualificados em TI. Esse número pode crescer para até 117 mil vagas abertas, sem que os empregadores consigam contratar pessoas com qualificação necessária para preenchê-las. Outros desafios da educação em TI residem nos aspectos referentes à motivação dos alunos e alto índice de evasão desses cursos. Nos últimos anos, os educadores de cursos ligados à área de TI têm enfrentado dois importantes problemas: (1) aumentar o interesse dos ingressantes e; (2) manter os estudantes desses cursos motivados, conforme citam Lampe et al. (2010), Dorn et al. (2007), Forte e Guzdial (2005), Robins et al. (2003).

A unidade curricular fundamental em um curso de TI é a disciplina de programação – também conhecida como Algoritmos e Estruturas de Dados – na qual são ensinados os conceitos básicos de programação (ROBINS et al., 2003; FEIJÓ et al., 2009; MARTINS et al., 2012). A importância dessa disciplina está no fato de que fomentará o estudante para uma formação de um alicerce sólido para que possa cursar as demais disciplinas de seu curso, e iniciar a construção de um pensamento lógico e abstrato que lhe será útil em sua carreira (DORN et al., 2007; AURELIANO, TEDESCO, 2012; ANDERSON et al., 2014)

De acordo com a análise de Da Silva et al. (2013), uma das principais questões que podem provocar o desinteresse logo no início do curso, e levar à desistência do aluno, é a falta de compreensão e raciocínio nas disciplinas de algoritmos e programação. As reprovações e desistências nas disciplinas iniciais acarretam dificuldades e impedimento por parte dos alunos a da continuidade a seus cursos de TI. O autor ainda complementa dizendo que:

> A evasão em cursos superiores das instituições brasileiras é um problema que vem atingindo grande parte das áreas do conhecimento, principalmente em cursos relacionados à Computação e Informática e já pode ser considerado um dos principais problemas

> acadêmicos do ensino superior no Brasil (DA SILVA et al., 2013, p. 168).

Talvez o índice de evasão e reprovação nessas disciplinas seja tão elevado devido ao alto grau de complexidade e abstração exigidos, fato que aliado aos métodos tradicionais de ensino, resulta nesse fenômeno tal danoso para a educação tecnológica do país.

Conforme o que defendem Freeman et al. (2014), a metodologia tradicional de ensino é a principal causa de desinteresse e, por vezes, da reprovação dos estudantes. A análise do estudo revela que estudantes submetidos a aulas tradicionais, em estilo de palestras, são mais tendentes à reprovação do que alunos em contato com métodos de aprendizado mais ativos e estimulantes.

A pesquisa de Freeman et al. (2014) analisou 225 estudos sobre métodos de ensino. Os resultados mostram que abordagens de ensino que transformam os alunos em participantes ativos, em vez de apenas ouvintes, reduzem taxas de reprovação e impulsionam notas em cerca de 6%. Ainda segundo o autor, as aulas tradicionais não são eficazes em unidades curriculares inerentes a Ciências, Tecnologias, Engenharias e Matemática (FREEMAN et al., 2014, p. 8410). Diante desse relato, autores como Martins et al (2012) e Aureliano e Tedesco (2012) corroboram com a ideia de que é necessário a exploração de métodos alternativos de ensino para motivar e reter os alunos nos cursos de TI.

1.2 MÉTODOS DE ENSINO DE PROGRAMAÇÃO

Com o propósito de esboçar um cenário que retrate quais metodologias têm sido utilizadas no campo do ensino/aprendizagem de programação, Silva et al., (2015) realizaram uma Revisão Sistemática de Literatura (RSL) sobre o assunto. É importante salientar que esta revisão de literatura não teve como objetivo comparar qual método é o mais eficaz para o ensino de programação de computadores.

Silva et al. (2015) utilizaram como fonte de consulta as publicações da Revista Brasileira de Informática na Educação (RBIE), Revista Novas Tecnologias na Educação (RENOTE), Workshop de Informática na Escola (WIE), Workshop de Educação em Computação (WEI), Simpósio Brasileiro de Informática na Educação (SBIE), e Simpósio Brasileiro de Jogos e Entretenimento Digital (SBGAMES). Os autores desta RSL encontraram 2325 trabalhos, e após aplicarem os seus critérios de exclusão, selecionaram 73 obras para continuidade do estudo.

A contribuição acadêmica de Silva et al. (2015) é muito relevante para nossa proposta, pois nos aponta quais metodologias estão sendo

empregadas no ensino de programação no Brasil nos últimos anos. O quadro geral proposto pelos autores apresenta quatro categorias principais: (a) ambientes virtuais de ensino; (b) jogos; (c) robótica; (d) outras estratégias. De acordo com os autores, o uso de ambientes virtuais de ensino compreende a utilização de ferramentas de *software* com propósitos diversificados, tais como: ensino à distância, aprendizagem cooperativa, programação visual em 3D, simulação e visualização de código, ambiente lúdico de programação, apoio à programação e teste de *softwares*.

A utilização de jogos como metodologia no ensino de programação é, segundo Silva et al. (2015), realizada de duas formas. A primeira refere-se à avaliação empírica de jogos já prontos. A segunda refere-se à atividade de jogar um jogo que tem como objetivo ensinar programação de computadores.

O emprego de atividades inerentes à robótica tem alcançado adeptos na área de educação em TI. Silva et al. (2015) mostram que o uso de kits de robótica, e linguagens de programação para microcontroladores, também são bastante utilizadas no ensino introdutório de programação.

As demais estratégias de ensino identificadas por Silva et al. (2015) em sua RSL incluem desde estratégias de aprendizagem motivacional, modelo personalizado de ensino, uso de objetos de aprendizagem, utilização de DOJOs, até o uso de linguagens de programação para estudantes portadores de necessidades especiais (surdos ou cegos).

Os resultados mostram que houve um crescimento no interesse dos educadores em utilizar metodologias que de alguma forma abordam jogos (seja na criação de jogos, ou no ato de joga-los) e robótica (seja na criação de robôs, ou no controle de robôs já prontos). Além disso, segundo os autores, com o uso de jogos e/ou robótica é possível constatar um aumento de interesse por parte dos alunos nas disciplinas de programação. Consequentemente, este fascínio despertado revelou alunos mais comprometidos com a construção de seus próprios conhecimentos em programação de computadores.

1.3 APRENDIZAGEM BASEADA EM JOGOS

A aprendizagem baseada em jogos tem funcionado como um método diferente das aulas em estilo de palestras conforme exalta Prensky (2010, 2012). Em geral, os estudantes de hoje – sobretudo os ingressantes em cursos de TI – são pessoas jovens que, por sua vez, possuem familiaridade com jogos digitais. Podemos considerar as sugestões de autores como Prensky (2012), Connolly et al. (2012), Perrotta et al.

(2013), que defendem a utilização de jogos digitais tanto no campo educacional quanto no campo informal para construção de conhecimentos, desenvolvimento de habilidades, atitudes e comportamentos.

Uma vez que a maioria dos alunos joga videogames em casa, não é uma surpresa o fato de que educadores tenham notado um aumento do envolvimento dos alunos ao usar jogos em ambientes educacionais (GEE, 2007; PRENSKY, 2012). Porventura, os jogos como meios transmissores ou facilitadores da aquisição de conhecimento são merecedores de uma oportunidade dentro do contexto educacional? Considerando as ideias de pesquisadores como Prensky (2012), Gee (2007), Sheldon (2011), McGonigal (2011), Van Staalduinen e Freitas (2011), que defendem o uso de jogos na educação – uma vez que seus relatos mostram que os alunos seriam mais produtivos se sua aprendizagem envolvesse jogos de alguma forma – talvez o uso de jogos no ambiente educacional mereça, sim, uma oportunidade.

Para o educador (professor ou instrutor) que tem em mente o intuito de adicionar jogos em seus métodos de ensino, é importante compreender os riscos e benefícios dos jogos em sala de aula. A apreciação das informações listadas a seguir deve ser capaz de auxiliar o docente em sua decisão. Caso o educador opte por adotar o uso de jogos em suas aulas, as informações sobre os possíveis riscos em sua adoção, pode auxiliar o docente no processo de tentar minimizar tais desvantagens.

De um modo geral, o uso de jogos na educação é uma técnica de ensino que busca inserir elementos lúdicos em torno do currículo e objetivos do curso. A intenção do uso de jogos é promover o envolvimento dos discentes e motivá-los a participarem mais ativamente das atividades propostas. Esse conceito educacional tornou-se muito popular nas últimas duas décadas, incentivando uma grande quantidade de publicações nesta linha de pesquisa.

Quando aplicados corretamente, muitos educadores descobriram que usar jogos como motivadores em suas salas de aula melhoraram os resultados das avaliações do curso, além de identificar estudantes mais atentos e produtivos (SAVI, 2011). Algumas pesquisas e jogos baseados em ciência estão focados em problemas do mundo real, e com isso motivam os alunos/jogadores a aprender e, ao mesmo tempo, a colaborar com a busca de soluções para essas questões do mundo real (KULLENBERG; KASPEROWSKI, 2016).

Entretanto, alguns críticos apontam algumas desvantagens concernentes a esse uso na educação, especialmente no que se refere a um possível desvio de atenção, isolamento social, distanciamento de

objetivos, avaliação de aprendizagem e alto custo de implantação e manutenção. No entanto, é importante salientar que as pesquisas que buscam retratar os aspectos negativos ao uso de jogos na educação são escassas. Encontramos na literatura mais estudos apoiando e apontando os benefícios dos jogos na educação, do que pesquisas que criticam e possuem um olhar mais cético sobre esta metodologia (BOURGONJON et al., 2013). Contudo, são pontos cruciais que não podem ser ignorados e demandam atenção dos educadores interessados. Há, também, aqueles que defendem que o uso de jogos na educação tem sua utilidade somente na educação infantil, porém entendemos que há muitas oportunidades para o uso de jogos nas outras fases do desenvolvimento humano (PRENSKY, 2012). Existem pesquisas suficientes em ambos os lados desse argumento para apoiar a utilização dessas ferramentas tanto nas séries iniciais quanto nas fases posteriores.

Benefícios

1. Envolvimento e engajamento

Os estudos de Prensky (2012) mostram que os estudantes se dedicam mais a um jogo com o objetivo educacional quando se oferece um sistema de recompensa. Os emblemas, ou insígnias, e o sistema de pontuação ajudam a visualizar o progresso do aluno e tentam traduzir isso em um benefício tangível. Segundo Gee (2007), ao aumentar o engajamento, também observamos um aumento na retenção de conteúdo, uma vez que a relação entre os alunos e o objeto de estudo torna-se mais fácil por meio da prática, principalmente quando comparada ao ato de apenas ler ou assistir a uma palestra.

2. Motivação e entusiasmo

O uso de jogos no ambiente educacional pode ser útil para despertar entusiasmo em relação ao assunto da aula, especialmente nos tópicos com os quais os estudantes apresentam mais dificuldades. Segundo McGonigal (2011), os jogos que são divertidos para jogar melhoram significativamente o desempenho de aprendizagem, e quando os alunos se divertem, a pressão para aprendizagem se dissipa, permitindo que eles definam e modifiquem livremente suas estratégias de acordo com um objetivo específico. Além disso, a motivação dos jogos é intrínseca, o envolvimento dos alunos com os jogos é um ato de volição.

3. *Feedbacks* instantâneos (respostas rápidas)

A maioria dos sistemas de jogos permitem *feedback* instantâneo, como quadros de classificação e pontuações, que os alunos podem usar para ver em que posição estão entre seus colegas. Prensky (2010) afirma que essas informações podem incentivar um aluno a experimentar um

questionário ou uma atividade novamente para obter uma colocação mais alta e cria motivação para um maior envolvimento nas atividades propostas. Chance (2013) em sua pesquisa sobre aprendizagem e comportamento, mostra que os alunos aprendem mais rápido quando há um intervalo mais curto entre o comportamento e o reforçador. Segundo o autor, é mais encorajador para os alunos quando as atividades realizadas identificam seus erros imediatamente, ao invés de ver uma nota vermelha em avaliações alguns dias depois.

4. Conexões Sociais

Por vezes nos deparamos com alunos que enfrentam dificuldades para criar conexões sociais com colegas em seus cursos. Embora as tendências isolacionistas dos jogadores tenham sido um estereótipo popular (PRENSKY, 2010), muitos dos jogos atuais são desenvolvidos em um paradigma de redes sociais que não somente permitem o trabalho em equipe e colaborativo como o exigem para que os jogadores tenham sucesso no jogo (BERMINGHAM et al., 2013; SANTOS, FREITAS, 2015; DIAS et al., 2016). Essa é uma das principais habilidades necessárias para trabalhar em uma economia global altamente conectada.

5. Letramento digital

O jogo pode promover letramento em diferentes níveis, desde digital e tecnológico até socioemocional (CRUZ; ALBUQUERQUE, 2013). Basicamente, o jogo eletrônico ou digital contribui para o desenvolvimento de habilidades necessárias para operar um computador ou um dispositivo móvel (*tablets*, *smartphones*). Contudo, vai realmente além disso, já que os processos de instalação, manutenção e configuração de redes – necessários para muitos jogos – também contribuem para aprimoramento de habilidades de letramento de alto nível em estudantes.

Riscos

1. Diminuição do intervalo de atenção do estudante

Os críticos acreditam que o uso excessivo, ou exclusivo, de jogos na educação com seu ritmo acelerado e os *feedbacks* imediatos possam criar um problema referente ao alcance da atenção do aluno (PRENSKY, 2012). Os alunos podem começar a esperar o mesmo tipo de respostas de todas as partes da sua educação e não as encontrarão, levando à frustração. Essa é uma crítica recorrente a todos os meios de comunicação modernos, mais recentemente dos *smartphones*. As novas tecnologias exigem novas formas de ver o mundo e a natureza do conhecimento. Os jogos de computador não são diferentes. O ritmo de ação muitas vezes rápido e o *feedback* imediato podem fazer com que as pessoas esperem os mesmos tipos de resposta rápida e instantânea de todas as coisas. Embora isso não

se traduza em todos os contextos, certamente é uma direção na qual nossa sociedade global conectada se encaminha.

2. Custos de implantação e manutenção

Os custos para implementar e/ou manter uma atividade de ensino baseada em jogos são variados com base no tipo de sistema utilizado. Conforme aponta Prensky (2012), estas despesas podem surgir de equipamentos, *softwares* e treinamento para os instrutores. Quando se trata de cursos isolados de curta duração, por vezes, esses custos são transferidos para os alunos por meio de taxas de inscrição, criando uma barreira ainda maior para a adoção em sala de aula. Em outras situações, existem os custos relacionados à manutenção para os sistemas virtuais *online* ou que demandam algum tipo de hospedagem virtual na Internet.

3. Avaliação de aprendizagem do aluno

Ao optar por inserir jogos no currículo escolar, muitas vezes não é claro como os resultados do jogo se encaixam na avaliação de aprendizagem do aluno em relação ao curso (PONTES, 2013). Embora a maioria dos jogos tenha uma maneira intrínseca de rastrear o progresso dos jogadores, o docente precisa encontrar um modo de traduzir o progresso do aluno no jogo em outro meio para visualizar o progresso do discente no processo de aprendizagem (PAPASTERGIOU, 2009; SHUTE; KE, 2012). Encontrar um bom ajuste entre jogos e conteúdo do curso (ou disciplina) não é uma tarefa trivial, e por isso precisa ser desempenhada com dedicação.

4. Logística dedicada ao jogo

A configuração de um jogo para um curso ou disciplina pode exigir demasiado planejamento e logística. Algumas questões demandam a atenção do docente: Os alunos poderão jogar o jogo em casa? Existe um custo adicional se o usarem fora da sala de aula? Existem computadores suficientes para que os alunos possam jogar os jogos em aula? O tempo de aula será utilizado para que os alunos joguem os jogos? Em determinadas situações, é necessário jogar o jogo até o final para entender seus conceitos e objetivos, o que pode requerer mais tempo do que o que se tem disponível.

5. Isolamento social

Uma das maiores e contínuas críticas aos jogos, e à tecnologia em geral, é que promove comportamentos antissociais e isola indivíduos (ABREU et al., 2008). Embora algum desses casos tenha sido verdade antes da explosão das tecnologias *web 2.0*, certamente não é mais. O foco da maioria dos novos jogos é o aspecto social (KAFAI; BURKE, 2015). Embora os jogadores não estejam interagindo cara a cara, eles estão interagindo sim virtualmente. De fato, essas interações tecnologicamente

mediadas refletem grande parte da comunicação do mundo real que impulsiona nossas vidas e negócios pessoais. O processo e as normas sociais ensinadas por essas interações representam habilidades muito reais e úteis que se traduzem perfeitamente fora dos jogos.

Embora o tema "ensino e aprendizagem baseado em jogos" tenha surgido há pouco tempo, a adesão de professores e educadores está em ritmo acelerado tanto na educação básica quanto na superior. Sendo assim, enxergamos que existem ainda muitas oportunidades para novas pesquisas sobre o uso de jogos na educação que nos auxiliarão a compreender melhor os efeitos e impactos que os jogos têm no envolvimento dos alunos em nossas salas de aula presenciais e virtuais.

Os benefícios e riscos elencados aqui são pontos importantes a serem considerados no momento da tomada de decisão ao escolher se, e como, os jogos podem apoiar um ambiente educacional. Se porventura o docente tem certa empatia com jogos, novas tecnologias digitais e suas inerentes características cativantes, o uso de jogos na educação pode ser útil. Entretanto, se o docente enfrenta dificuldades em priorizar e organizar seu tempo para organizar e planejar seu curso ou disciplina, o uso de jogos na educação pode não ser a escolha certa para o momento. Começar pequeno, dar pequenos passos e dedicar tempo para organização e planejamento é crucial para uma caminhada bem-sucedida.

Considerando o nosso caso particular de ensino de programação, o uso de métodos de aprendizagem baseada em jogos digitais é apoiado por autores como Denner et al. (2014), Bermingham et al. (2013), Hwang e Wu (2012), Depradine (2012), Sung et al (2011), Phelps et al (2009), Bayliss (2009), Sung (2009). Embora as pesquisas neste campo educacional sejam recentes, as abordagens utilizadas são diversificadas.

Como discutido em Sung et al. (2008) quando examinamos os recentes esforços para integrar jogos em disciplinas de Computação, podemos observar três categorias gerais:

a) Disciplinas de desenvolvimento de jogos: são disciplinas formatadas especificamente para a criação de jogos digitais como um produto final. Os alunos dessas disciplinas se preocupam com o conjunto de aspectos de produção comercial de um jogo, até mesmo os artísticos. Um conjunto de conceitos de *game design* são considerados, incluindo qualidade audiovisual, simulações físicas, alta performance e taxa de entretenimento.

b) Disciplinas de programação de jogos: nessas disciplinas são abordados os aspectos técnicos necessários à construção de jogos digitais como, por exemplo, ciclos de repetição, inteligência artificial,

representação computacional de terrenos. Em geral, não se exige a produção de um jogo como produto final, e os tópicos computacionais abordados podem ser aplicados em outros domínios do conhecimento.

c) Disciplinas utilizadoras de jogos: essas são disciplinas de Computação que criativamente integram jogos digitais como atividades de programação. Ou seja, os alunos aprendem conceitos de programação por meio do uso, ou criação, de jogos. São chamadas de utilizadoras, ou usuárias, pois se espera que ao fim da disciplina os estudantes entendam os conceitos de programação, mas não os profundos aspectos de desenvolvimento de jogos.

As duas primeiras categorias referem-se a cursos que têm como objetivo o ensino de desenvolvimento de jogos digitais e suas particularidades. Na terceira categoria, enquadram-se os tradicionais cursos de Computação que encontram nos jogos um meio de motivar e engajar os estudantes.

Muitos educadores têm identificado e tirado proveito do interesse e familiaridade dos jovens estudantes com jogos digitais, e assim têm realizado a integração com as disciplinas iniciais de programação. Baseado no tipo de integração, podemos reconhecer três diferentes abordagens:

a) Pouca ou nenhuma programação: nessa abordagem, os estudantes aprendem ao jogar um jogo direcionado para o ensino de programação, mas não constroem seus próprios jogos;

b) Atividades isoladas de desenvolvimento de jogos: os jogos são desenvolvidos como parte de uma atividade avaliativa individual. A cada unidade concluída, os alunos desenvolvem pequenos jogos a partir do assunto que foi ministrado.

c) Atividades integradas de programação de jogos: os professores propõem atividades relacionadas à criação de jogos utilizando ferramentas e recursos específicos para esse domínio.

Sobre a primeira abordagem, consideramos que jogos educacionais são limitados ao escopo do próprio jogo além de posicionar o aluno como jogador ao invés de lhe conceder controle sobre o ambiente lúdico. Na literatura, as recentes pesquisas sugerem que para atender de modo mais eficiente às necessidades de aprendizado dos alunos, desenvolver jogos pode ser mais eficaz do que simplesmente jogá-los. Além disso, é importante destacar que essa prática de desenvolvimento de jogos pode ser implantada por meio de projetos.

1.4 APRENDIZAGEM BASEADA NO CONSTRUCIONISMO

Estas metodologias de ensino que permitem a interação entre o sujeito e objeto encontram lugar também na concepção psicológica de Piaget (1971), denominada construtivismo, na qual o conhecimento é algo a ser construído quando acontece um convívio interativo entre sujeito e objeto.

Na teoria construtivista, o sujeito é visto como um ser em desenvolvimento, ativo e capaz de construir o novo. Os sujeitos são construtores do seu próprio conhecimento. Nesta direção, o conhecimento é construído e a necessidade do aluno de aprender e de desenvolver atividades contribui com a construção do conhecimento. A ideia construtivista de Piaget (1971) defende que a aprendizagem ocorre quando a pessoa estabelece relações com o objeto de conhecimento e procura atender às suas necessidades de aprendizagem. Oriunda dessa vertente piagetiana, surge uma teoria proposta por Seymour Papert (1994), denominada Construcionismo, a qual defende que o aprendiz constrói o seu conhecimento por meio de interações com o computador.

Neste sentido, o construcionismo defende a ideia de que o conhecimento é reconstruído pelo estudante. Considerando um ambiente informatizado, Papert (1994) apoia a ideia de que é preciso permitir ao aluno construir e não apenas responder à programação, mas fazer e programar. É o aluno que domina o computador e não o computador que domina o aluno. Essa abordagem construcionista valoriza as estruturas cognitivas e procura caminhos para que a aprendizagem ocorra por meio de ações, discussões e interações com o objeto de estudo. Em síntese, o aluno que programa o computador atua em um ambiente aberto, inserindo-se por completo na atividade, tornando-a significativa para o seu desenvolvimento.

É importante diferenciar a teoria Construtivista de Piaget da teoria Construcionista de Papert. Indo além do simples trocadilho entre as palavras, entendemos que a distinção é válida e que a integração de ambas as opiniões pode enriquecer a nossa compreensão de como os indivíduos aprendem e crescem. O construtivismo de Piaget dá destaque para os interesses e habilidades das crianças, em diferentes estágios de seu desenvolvimento. Sua teoria descreve como os modos de fazer e pensar das crianças evoluem ao longo do tempo, e em quais circunstâncias as crianças são mais propensas a mudar – ou manter – suas opiniões já formadas. Piaget sugere que as crianças têm boas razões para não abandonar suas visões de mundo apenas porque outra pessoa, mesmo sendo esta pessoa um especialista, diz que elas estão erradas.

Por sua vez, o construcionismo de Papert, em outra direção, concentra-se mais no modo de aprender, ou "aprender a aprender", e no conceito de adquirir conhecimento "criando artefatos". A teoria de Papert ocupa-se em perceber como os alunos se envolvem em uma interação com seus próprios artefatos (ou de outras pessoas), e como essa interatividade pode melhorar a auto-aprendizagem e, consequentemente, facilitar a construção de novos conhecimentos. Além disto, o construcionismo ainda ressalta a importância que as ferramentas, a mídia e o contexto têm no desenvolvimento humano.

O construcionismo de Papert (1994) também encontra lugar na metodologia de aprendizagem por meio de projetos. Segundo Gilakjani et al. (2013) uma proposta construcionista permite que os estudantes façam uso da tecnologia para construir seu próprio entendimento ao incorporar autênticas experiências em situações de aprendizado por meio de projetos. Em um ambiente como esse, os estudantes identificam a necessidade de adquirir novos conhecimentos e continuamente revisar conhecimentos já adquiridos antes de solucionar problemas propostos.

Integrar tecnologia e métodos construcionistas como, por exemplo, aprendizado baseado em projetos torna os estudantes mais responsáveis e ativos no processo de aprendizagem. De acordo com Grant (2002), utilizar tecnologia juntamente com o construcionismo concede flexibilidade aos professores para individualizar o aprendizado para aprimorar os processos cognitivos e metacognitivos. O papel do aluno nessa metodologia de aprendizagem por meio de projetos é de um agente ativo que faz uso dos conhecimentos para produzir algo significativo. Dito de outra forma, o aluno não é um mero receptor de informações sem análise crítica, sem pensamento criativo, dependente exclusivo das informações que o professor oferece, mas toma uma posição ativa e atuante no seu próprio processo de aprendizagem.

Uma iniciativa tem se destacado nesse campo no qual o estudante assume um papel ativo nesse sistema, é o chamado "Movimento Maker" que possui forte influência de movimentos culturais no hemisfério norte, mas já conta com diversos laboratórios de inovação espalhados pelo Brasil conhecidos como "*fab labs*". Com a temática do "faça-você-mesmo" (do inglês *do-it-yourself*, *DIY*), as pessoas interessadas podem criar, consertar, alterar e construir vários objetos e projetos próprios. A ideia base é a manutenção de uma rede de pessoas que compartilham suas experiências e conhecimentos entre si. Silva e Merkle (2016) declaram que a facilidade de comunicação e o acesso a informações pela internet promovem a estruturação do "Movimento Maker". Esse movimento caracteriza-se, principalmente, por colocar o aprendiz na posição de

fazedor, de construtor ou criador. Alguns autores como Silva e Merkle (2016), Blikstein e Krannich (2013), Halverson et al., (2014) mostram que esse movimento encontrou no campo educacional um espaço interessante, no qual professores e alunos podem interagir com o objeto de estudo por meio das criações, modificações, alterações que o "Movimento Maker" propõe.

1.5 ENSINO BASEADO NA CONSTRUÇÃO DE JOGOS

Segundo Papert e Harel (1991), a teoria do construcionismo considera o aprendizado como um processo no qual os aprendizes constroem seus conhecimentos por meio da interação com o objeto de estudo. A perspectiva construcionista coloca a criação de jogos nas mãos dos alunos para incentivar a busca de conhecimento recorrendo ao desenvolvimento de seus próprios jogos. Nessa abordagem, o objetivo da atividade de construção de jogos é apoiar tal metodologia, proporcionando um ambiente de aprendizagem adequado. Sendo assim, os educadores construcionistas focam seus esforços em proporcionar aos alunos mais oportunidades para desenvolver seus jogos e, ao mesmo tempo, construir novas relações com o conhecimento durante o processo, ao invés de simplesmente incorporar lições diretamente dentro dos jogos (KAFAI, 2006).

Kafai e Burke (2015) também apontam que poucos educadores optaram por tomar um caminho diferente: o de criar jogos ao invés de jogá-los para aprender. Devido a sua simplicidade e facilidade de uso, alguns educadores escolhem ambientes específicos para o ensino de programação como *Alice*, *Scratch*, *Greenfoot* e outras ferramentas de programação visual (UTTING et al., 2010; FINCHER et al., 2010). No entanto, esses ambientes não são adequados para o desenvolvimento de aplicações industriais. Com isso em mente, outros educadores se desviaram de tais ferramentas e empregaram a abordagem construcionista usando linguagens de programação convencionais (ou seja, baseadas em texto) em disciplinas iniciais de programação (MSELLE; KONDO, 2014; SUNG; SNYDER, 2014; WEINTROP; WILENSKY, 2015). É relevante notar que esta tese não tem o objetivo de comparar as diferentes abordagens do ensino de programação. De fato, não questionamos a eficácia de quaisquer métodos de programação visual como ferramentas educacionais, entretanto sugerimos que um paradigma de programação baseado em texto é, contudo, capaz de atingir jovens estudantes com conteúdo atraente e, ainda assim, continuar compatível com o currículo dos cursos de TI.

Do ponto de vista dos cursos de TI, muitos educadores identificam e aproveitam os interesses e a familiaridade da geração mais jovem por jogos digitais, e demais conteúdos relacionados, em suas disciplinas iniciais de programação. A utilização de jogos em disciplinas de programação é feita de diversas formas que vão desde jogar, analisar e desenvolver jogos. O foco desta tese é a abordagem construcionista de Papert, ou seja, uma análise de como a metodologia de criação de jogos pode envolver e motivar jovens estudantes em cursos de TI, utilizando linguagens de programação baseada em texto.

Alguns autores implementaram essa sistemática usando diferentes linguagens de programação baseadas em texto. As linguagens que foram utilizadas estão relacionadas no Quadro 1. Nestas obras, os autores descrevem suas experiências de implementação e *feedback* dos alunos, o que foi útil como suporte para construção de uma estrutura com diretrizes de referência para ajudar outros educadores a implementar essa metodologia em seu próprio ambiente de aprendizagem. É importante lembrar que esses estudos pretendem contribuir no entendimento dos alunos a respeito dos conceitos de programação abstrata, ao invés de conceitos inerentes à construção de jogos.

Com o intuito de identificar prévias pesquisas em educação baseada na criação de jogos, conduziu-se uma revisão de literatura. As bases eletrônicas utilizadas foram *Association for Computing Machinery* (ACM), *Science Direct*, e *Institute of Electrical and Electronics Engineers* (IEEE). Foram utilizados os seguintes termos de busca:

(game OR gaming) (based OR themed) (development OR construction OR design OR making) (teaching OR learning OR course OR education)

Além disso, optou-se por manter o foco nos trabalhos empíricos sobre a construção de jogos utilizando linguagens de programação convencionais dentro de cursos de TI, assim sendo selecionamos 13 obras (ver Quadro 1).

Quadro 1: Educação baseada na criação de jogos utilizando linguagens de programação convencionais

Autores	**Linguagem de programação**	**Estudo/Abordagem/Descrição**
Corral et al. (2014)	C#	Apresenta uma contribuição para o ensino de linguagens orientadas a objetos (OO) através de uma abordagem baseada em jogos e interação com interfaces tangíveis.

Autores	**Linguagem de programação**	**Estudo/Abordagem/Descrição**
Dickson (2010)	Objective-C com OpenGL	Detalha o projeto e implementação de um curso de design de videogames que não apenas permite que os alunos criem jogos, mas também aprendam habilidades que são referências para cursos de TI.
Drake e Sung (2011)	Java	Analisa a inclusão de jogos de tabuleiro, cartas e dados como atividades de programação em cursos iniciais de programação.
Feijó et al. (2009)	Java	Propõe um livro texto para um primeiro curso de introdução à Ciência da Computação, usando a criação de jogos digitais como instrumento de aprendizado e motivação.
Leutenegger e Edgington (2007)	C++	Descreve o uso da abordagem "Game First" para ensinar conceitos introdutórios de programação através da criação de jogos.
Lewis e Massingill (2006)	Java	Descreve um projeto que tem sido usado para ensinar programação no qual os alunos desenvolvem um jogo bidimensional usando uma biblioteca de códigos.
Ludi (2011)	Java	Descreve as experiências de criação de um jogo educativo desenvolvido como um projeto de alunos em uma disciplina de Engenharia de *Software*.
Paralic e Pietrikova (2014)	C	Examina e discute progressos recentes no ensino/aprendizagem de programação baseado em desenvolvimento de jogos.
Schuster (2010)	Java	Relata as experiências de ensino de programação usando uma biblioteca Java para criação de jogos do tipo arcade.

Autores	Linguagem de programação	Estudo/Abordagem/Descrição
Soares, Fonseca e Martin (2015)	Java	Discute a implementação de um ambiente de aprendizagem baseado na resolução de problemas e a condução da disciplina de programação inicial no contexto do *design* de um jogo.
Sung e Snyder (2014)	C# com Microsoft XNA	Demonstra a eficácia geral do paradigma de programação baseado em texto para ensinar princípios básicos da programação de computadores.
Sung et al. (2011)	C# com Microsoft XNA	Concepção, implementação e avaliação de módulos com a temática de jogos para atividades de programação.
Wang (2011)	C# com Microsoft XNA	Relata uma extensa avaliação sobre a inclusão de um projeto em uma disciplina de Arquitetura de *Software* para identificar se a criação de jogos pode ser utilizada com sucesso para ensinar os conceitos desta unidade curricular.

Fonte: Próprio autor.

Ao analisar as abordagens utilizadas pelos educadores, as obras encontradas nesta revisão de literatura referem-se a disciplinas de programação com temáticas de jogos; ou seja, cursos de programação que possuem atividades de criação de jogos. Todas essas unidades curriculares trouxeram os jogos como uma parte de sua experiência de aprendizagem de programação. Em cada caso, os jogos foram construídos de acordo com os conceitos de programação previamente vistos em sala de aula.

Embora a criação de jogos em sala de aula ainda seja um campo emergente, a maioria dos estudos citados relatam um aumento significativo no engajamento e motivação dos alunos nesses cursos (SUNG et al., 2011; LEUTENEGGER; EDGINGTON, 2007). Com base nesses resultados, é possível entender que a integração de jogos em disciplinas iniciais de programação é uma estratégia promissora para despertar o interesse de jovens estudantes.

Considerando os desafios e dificuldades dos cursos iniciais, é importante notar que essa estratégia seja cuidadosamente adotada por

professores e departamentos. Por esta razão, é importante que um plano adequado seja seguido para implementar a abordagem com boas chances de sucesso. Por outro lado, é importante salientar que o docente deve estar atento aos objetivos do curso. Se o curso, ou disciplina, tem como objetivo final o ensino de programação de jogos, fica claro a necessidade da construção do currículo em torno de conteúdos e atividades inerentes aos jogos em todos os seus aspectos, concepções e perspectivas. Todavia, se a disciplina tem como finalidade o ensino de programação de modo geral, e o docente quer adotar uma abordagem de construção de jogos como fator motivador, é crucial que o professor faça ajustes e construa pontes que liguem o desenvolvimento de jogos com a criação de outras aplicações industriais exigidas pelo mercado de trabalho.

1.6 PROGRAMAÇÃO EM TEXTO VERSUS BLOCOS

Há cerca de uma década, o MIT Media Lab[1] introduziu o conceito de programação baseada em blocos (MALONEY et al., 2004; RESNICK et al., 2009). A ideia era desenvolver uma interface que permitisse que os programas de computador fossem construídos simplesmente arrastando e soltando blocos de quebra-cabeça para representar instruções e comandos de programação complexos. Com este novo método de ensino/aprendizagem de programação, nasceu a popular plataforma *Scratch*[2]. Esta abordagem diminuiu a complexidade para experimentações com o pensamento computacional, tornando possível para os alunos a criação de animações interativas e pequenos jogos sem escrever uma única linha de código. Este simples conceito dissolve a necessidade de aprender a sintaxe de uma linguagem de programação formal e tornou o ensino e a aprendizagem dos conceitos básicos da programação acessíveis aos alunos mais jovens e aos professores sem experiências anteriores com programação formal.

Fora da sala de aula, porém, a programação sempre foi, e ainda permanece, um processo de digitação de letras, números e símbolos (SUNG; SNYDER, 2014). Essa programação baseada em texto, usada em linguagens de programação como C, Java e Python, requer codificadores para obedecer e se comportar com uma sintaxe formal. Apesar do dissabor de lidar com falhas de ortografia em nomes de variáveis e erros de sintaxe inevitáveis, tais métodos de codificação desenhados para serem mais "amigáveis" realmente não se popularizaram (PRICE; BARNES, 2015). Algumas ferramentas foram criadas para que executivos pudessem definir

[1] Massachusetts Institute of Technology: MIT Media Lab https://www.media.mit.edu/
[2] Scratch: Imagine, Program, Share – http://scratch.mit.edu

a lógica de negócios por meio de uma interface gráfica de usuário sem escrever linhas de códigos. Ou até mesmo para que desenvolvedores web pudessem adicionar recursos interativos para seus websites sem aprender JavaScript. Mas, tais facilidades não substituem o poder e a flexibilidade da programação baseada em texto (WEINTROP; WILENSKY, 2015). E sendo que esses recursos não têm demonstrado ganhos de uma popularidade significativa, assim intensifica-se a demanda para a habilidade clássica de escrever códigos baseado em texto.

De acordo com Sung e Snyder (2014), programação textual, ou paradigma de programação baseado em texto, refere-se ao método convencional de desenvolvimento de programas de computador (*softwares*) inserindo texto em um editor de texto, por exemplo, utilizando uma linguagem programação como Java ou C. O Quadro 2 mostra alguns exemplos de linguagens de programação baseadas em texto.

Quadro 2: Exemplos de programação baseada em texto

Código em C:

```c
#include<stdio.h>
void main(void){
    printf("Olá Mundo!\n");
}
```

Saída/Resultado:

```
Olá Mundo!
```

Código em Java:

```java
public class HelloWorld {
    public static void main(String[] args)
{
        System.out.println("Olá, Mundo!");
    }
}
```

Saída/Resultado:

```
Olá, Mundo!
```

Código em PHP:

```php
<?php
    echo "Olá Mundo!";
?>
```

Saída/Resultado:

```
Olá Mundo!
```

Código em Python:

```python
print('Olá Mundo!');
```

Saída/Resultado:

```
Olá Mundo!
```

Fonte: Próprio autor.

Para o particular caso do ensino em programação, interessantes contribuições têm emergido com o objetivo de despertar o interesse de jovens alunos a aprenderem a programar. Como exemplo podemos citar *Alice*, *Scratch*, *Greenfoot*, como os mais conhecidos (UTTING et al., 2010; FINCHER et al., 2010). Dentre essas opções, a ferramenta *Scratch* é a mais utilizada nas turmas iniciais devido a sua simplicidade. Entretanto, Mselle e Kondo (2014) argumentam que soluções como estas não podem ser utilizadas para o desenvolvimento de *softwares* comerciais de grande porte, o que resulta na necessidade de explorar métodos alternativos para ensinar programação utilizando as linguagens convencionais como C, C++ e Java (MSELLE; KONDO, 2014, p. 17).

Diferentemente da programação visual, ou paradigma de programação por blocos, no qual são construídos *softwares* por meio da manipulação de ícones predefinidos em ambientes virtuais especializados, por exemplo, programação com *Alice*, *Scratch*, *Greenfoot*, explicam Sung e Snyder (2014). A Figura 1 e Figura 2 mostram dois exemplos de ambientes de programação baseados em blocos.

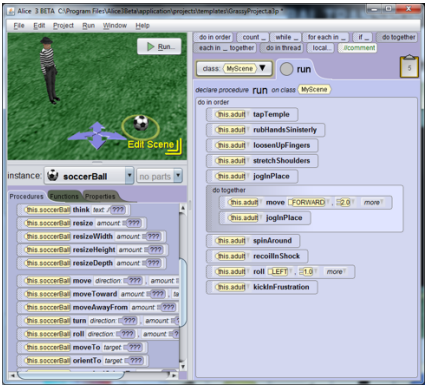

Figura 1: Alice – Ambiente de programação em blocos
Fonte: http://www.alice.org

Figura 2: Scratch – Ambiente de programação em blocos
Fonte: http://scratch.mit.edu

Atualmente, existem dois principais indicadores que analisam a popularidades das linguagens de programação. Um deles é o TIOBE (TIOBE, 2017) que calcula a popularidade das linguagens de programação por meio da quantidade de pesquisas realizadas nos principais buscadores de conteúdo da Internet, como Google, Youtube, Baidu, Wikipedia, Yahoo. O outro índice é o PYPL, do inglês *PopularitY of Programming Language* (PYPL, 2017), que por sua vez usa somente as pesquisas realizadas no Google para compor a sua lista de linguagens de programação mais populares. A seguir, no Quadro 3, vemos a relação com as dez linguagens de programação mais populares em Janeiro de 2017 segundo os dois índices.

Quadro 3: Linguagens de programação mais populares em Janeiro de 2017

TIOBE	
Posição	Linguagem
1	Java
2	C
3	C++
4	C#
5	Python
6	Visual Basic .NET
7	Javascript
8	Perl
9	Assembly
10	PHP

PYPL	
Posição	Linguagem
1	Java
2	Python
3	PHP
4	C#
5	Javascript
6	C
7	C++
8	Objective-C
9	R
10	Swift

Fonte: TIOBE (2017) e PYPL (2017).

Nota-se que existe um consenso entre os dois índices de que Java é a linguagem de programação mais popular dentre todas pesquisadas. Podemos ver também que, embora em posições diferentes nas duas listas, existe uma consonância no fato de que as linguagens C, C++, C#, Python, Javascript e PHP estão entre as dez linguagens de programação mais populares.

Ambientes virtuais de programação por blocos são ferramentas de *software* que utilizam uma linguagem gráfica para a programação, diferente do modo texto convencional. Tais ambientes virtuais permitem que os seus usuários criem histórias interativas, jogos, animações 2D e 3D. O processo de criação é o mais simplificado possível e é composto por elementos gráficos (blocos) que podem arrastados e encaixados entre si obedecendo regras sintáxicas de programação (PRICE; BARNES, 2015).

Embora não exista um índice para medir a popularidade dos ambientes visuais de programação, há vários trabalhos acadêmicos que se ocuparam em comparar duas ou mais destas ferramentas baseadas em elementos gráficos para o ensino de programação (FINCHER et al., 2010; UTTING et al., 2010; PRICE; BARNES, 2015).

Weintrop e Wilensky (2015) realizaram um estudo para investigar o grau de eficiência do uso de ambientes visuais de programação nas disciplinas iniciais de programação. Os resultados do estudo mostram que embora exista uma facilidade maior de uso e de aprendizado, essas ferramentas visuais se mostram menos poderosas do que as linguagens convencionais. Ou seja, a programação por blocos é mais limitada do que a programação convencional, o que se reflete imediatamente no mercado de trabalho, pois cada vez mais empresas e organizações demandam *softwares* e aplicações com maior robustez e alta complexidade. A construção dessas aplicações só é possível com uso de linguagens de programação convencionais que possuem um leque de recursos e funcionalidades maior do que os ambientes gráficos (WEINTROP; WILENSKY, 2015; MSELLE; KONDO, 2014).

Nessa mesma direção, Valaski e Paraíso (2012) avaliaram o uso do ambiente visual de programação *Alice* em um curso de TI. Os resultados das investigações contradizem pesquisas anteriores, e mostram que o uso da ferramenta *Alice* não contribuiu de forma significativa na ensino e aprendizagem em uma disciplina introdutória de programação. Podemos perceber que esse tipo de ambiente tem aplicações limitadas para alunos dos cursos de TI, pois de acordo com os autores:

> [...] o uso do ambiente Alice pode ser mais indicado para alunos que estão em um estágio bem inicial de aprendizagem e, além disso, estão tendo dificuldades em abstrair os principais conceitos de programação. (VALASKI; PARAISO, 2012, p. 9)

Os motivos pelos quais os autores creditam as limitações dessa ferramenta vão desde erros durante a criação de animações, problemas nas execuções de instruções, interface em língua inglesa, até o propósito da ferramenta que é "distante de uma ferramenta para desenvolvimento de sistemas comerciais" (VALASKI; PARAISO, 2012). Entendemos que tais ambientes visuais de programação são mais úteis para ensinar o raciocínio algorítmico, ou seja, como pensar algoritmicamente do que para ensinar programação de computadores ou dispositivos móveis.

Em virtude do que foi apresentado e discutido, percebemos que o propósito dos ambientes visuais de programação em blocos (*Alice* e *Scratch*, por exemplo) é o de facilitar o ensino e aprendizagem do raciocínio algorítmico, inclusive em crianças, ou seja, alunos que estão nas séries escolares iniciais conforme apontam Kafai (2006), Wilson et al. (2013), Kafai e Burke (2015). Enquanto isso, o propósito das linguagens convencionais de programação é o desenvolvimento de *softwares* e aplicativos. Tendo em vista o objetivo de um curso de TI, que é preparar o aluno para entrar no mercado de trabalho, enxergamos que as linguagens de programação convencionais possam ser utilizadas no ensino de programação nesses cursos desde as primeiras fases.

1.7 SÍNTESE

Em suma, considerando a nossa experiência docente na área de ensino de programação de computadores, refletindo sobre a necessidade da melhoria constante da formação de profissionais em TI no Brasil, observando a diversidade dos métodos de ensino de programação no sistema de ensino técnico/superior brasileiro, analisando a eficácia, atratividade e afetividade que os jogos despertam no público jovem estudantil, valorizando o contato direto do sujeito com o objeto de estudo por meio da teoria construcionismo e, por fim, em virtude do poderio das linguagens de programação convencionais no desenvolvimento de aplicações industriais, vemo-nos diante da oportunidade do desenvolvimento de um guia para ensino de programação de computadores por meio da criação de jogos digitais utilizando linguagens de programação convencionais.

2 GUIA PARA ENSINO DE PROGRAMAÇÃO

Neste capítulo são tratados dois assuntos: o processo de concepção do guia, e a apresentação do guia em si. O processo de construção do guia para ensino de programação proposto nesta tese, inicia-se com seu protótipo, passando pela avaliação de especialistas, até chegar em sua versão otimizada. Este processo consistiu em sua primeira parte de uma revisão de literatura que buscou identificar outras pesquisas referentes a educação fundamentada na criação de jogos, o que forneceu uma base que foi utilizada na idealização de um protótipo do guia. Sendo assim, a versão inicial foi compartilhada com outros pesquisadores, visando obter opiniões para possíveis melhorias no guia antes de sua efetiva implantação. Com as contribuições desses especialistas, aprimoramos o protótipo e desta forma obtivemos o guia otimizado.

O guia otimizado é apresentado e esclarecido com detalhes, fase a fase, a partir da seção 2.4. Contudo, uma versão sintética do guia está disponibilizada no Apêndice D desta tese.

2.1 REVISÕES DE LITERATURA

O primeiro passo na confecção do guia foi realizar uma revisão de literatura para identificar pesquisas relacionadas com planos de implementação de uma metodologia baseada na criação de jogos. O objetivo deste primeiro passo foi analisar as obras selecionadas, identificar aspectos principais que puderam nortear a confecção deste guia. Em outras palavras, foi feita uma busca por pontos-chave a serem considerados na implementação e execução desta metodologia de ensino de programação baseado no desenvolvimento de jogos.

Esta revisão de literatura foi feita em língua inglesa e com a restrição de obras publicadas há até no máximo dez anos. As bases eletrônicas utilizadas foram *Association for Computing Machinery* (ACM), *Science Direct*, Google Acadêmico, Portal de Periódicos CAPES, e *Institute of Electrical and Electronics Engineers* (IEEE). Foram utilizados os seguintes termos de pesquisa em inglês:

(strategies OR guidelines OR framework OR methodology OR approach) (based OR themed) game (development OR construction OR design OR making) (teaching OR learning OR course OR education)

Nesta revisão da literatura foram identificados estudos que discutem a implementação da metodologia de criação de jogos nas primeiras fases escolares (ensino fundamental ou médio) (BERMINGHAM et al., 2013; WILSON; HAINEY; CONNOLLY, 2013) e no ensino superior (DRAKE; SUNG, 2011; SUNG et al., 2011). O

Quadro 4 fornece uma visão geral dos guias/modelos de implementação desta metodologia.

Quadro 4: Criação de jogos como forma de aprendizagem (inglês)

Autores	Guia/Modelo	Propósito
Bermingham et al. (2013)	Modelo de aprendizagem baseada na criação colaborativa de jogos.	Apresenta recomendações para criação colaborativa de jogos em sala de aula.
Drake e Sung (2011)	Guia para o ensino de programação com jogos de tabuleiro populares.	Apresenta questões a serem consideradas na escolha de um jogo de tabuleiro para uma atividade de programação.
Sung et al. (2011)	Módulos de atividades de programação com temática de jogos.	Apresenta módulos de atividades que permitem ao docente explorar e desenvolver seus materiais para o ensino de programação com jogos.
Wilson, Hainey e Connolly, (2013)	Guia para aprendizado baseada na construção de jogos.	Tem como objetivo fornecer aos professores de séries iniciais um ponto de partida para usar a criação de jogos em sala de aula desse nível de ensino.

Fonte: Próprio autor.

Bermingham et al. (2013) apresentam em sua obra um projeto voltado para crianças e adolescente chamado Aprendizagem com Criação Colaborativa de Jogos (*Making Games in Collaboration for Learning, MAGICAL*). Este projeto tem como objetivo explorar o uso da criação colaborativa de jogos como um modelo pedagógico. O principal intuito é verificar se esse tipo de abordagem pode apoiar a colaboração, a resolução de problemas, a criatividade e habilidades de alfabetização digital. De acordo com os autores, "a criação colaborativa de jogos fornece um modelo no qual os alunos podem trabalhar juntos para criar algo que seja significativo para eles, inserindo-os no processo e facilitando o desenvolvimento de uma série de habilidades do século XXI, como alfabetização digital" (BERMINGHAM et al., 2013).

Drake e Sung (2011) discutem a inclusão de jogos de tabuleiro, cartas e dados como tarefas de programação em cursos introdutórios de programação. Os autores relatam que a implementação de tais jogos geralmente requer menos conhecimento prévio do docente. Além disso, levantam algumas questões a serem consideradas no momento da escolha de um jogo para uma atividade dessa natureza, listando 32 jogos específicos que são adequados para ensinar os principais tópicos iniciais de programação. A pesquisa dos autores também analisa a implementação de alguns destes jogos e relata como estes projetos foram bem-sucedidos.

Sung et al. (2011) propõem módulos chamados Atividades com Temáticas de Jogo (*Game-Themed Assignments*, GTA) que são projetadas especificamente para docentes que não possuem muita experiência em jogos. De acordo com os autores, esses módulos são independentes e cada um é uma atividade de programação autônoma que desafia os alunos sobre conceitos relacionados a uma área específica da unidade curricular. A obra também discute a concepção, implementação e avaliação desses módulos, a fim de ajudar os professores a usarem essa abordagem, permitindo-lhes começar a explorar e construir conhecimentos e materiais para ensinar por meio de jogos.

Wilson, Hainey e Connolly (2013) propõem um guia para a introdução de criação de jogos de computador nas séries iniciais de ensino (*primary school*, em inglês), utilizando a ferramenta visual *Scratch*. Os autores apresentam um guia geral baseado em estudos empíricos realizados em três escolas da cidade de Glasgow, na Escócia, nos anos de 2011 e 2012. Segundo os autores, alguns professores enfrentam muitos obstáculos quando tentam implementar essa metodologia na escola, de modo que a finalidade do guia é ajudar esses docentes, dando-lhes um ponto de partida para a criação de jogos em sala de aula.

Para complementar a revisão de literatura feita em língua inglesa, realizou-se também uma outra investigação para identificar pesquisas relacionadas com planos de implementação de uma metodologia baseada na criação de jogos, porém em língua portuguesa, e com a restrição de obras publicadas há até no máximo dez anos. Foram consultadas as mesmas bases eletrônicas utilizadas anteriormente, a saber: *Association for Computing Machinery* (ACM), *Science Direct*, Google Acadêmico, Portal de Periódicos CAPES, e *Institute of Electrical and Electronics Engineers* (IEEE). Os termos de pesquisa utilizados foram:

(estratégia OR guia OR metodologia OR abordagem) jogos (elaboração OR desenvolvimento OR construção OR criação OR design) (ensino OR aprendizagem OR curso OR educação)

Nesta revisão de literatura identificamos pesquisas com foco no ensino superior (KASPERAVICIUS et al., 2008; PONTES, 2013), ensino médio/técnico (SCAICO et al., 2013; DOMINGOS; RECENA, 2010; DIAS; BORGES; PEREIRA, 2016). O Quadro 5 apresenta uma relação das obras selecionadas nesta revisão.

Quadro 5: Criação de jogos como forma de aprendizagem (português)

Autores	**Guia/Modelo**	**Propósito**
Scaico et al. (2013)	Olímpiada Interna de Programação com Scratch (OIPS).	Ensino de programação para alunos do ensino médio, com uma abordagem de ensino que estimula criatividade e poder de exploração.
Kasperavicius et al (2008)	Projeto Primeira Habilitação.	Ensino de criação de jogos digitais, com aplicação de métodos ágeis de desenvolvimento de *software* e seus impactos aprendizagem.
Domingos e Recena (2010)	Jogos didáticos no processo ensino e aprendizagem de química.	Elaboração de jogos didáticos como apoio na construção de conhecimentos científicos de tópicos de química orgânica.
Pontes (2013)	Jogos no processo ensino e aprendizagem em Algoritmos.	Criação de jogos no processo de ensino de disciplinas do currículo de programação pertencentes à área de Ciências da Computação.
Dias, Borges e Pereira (2016)	Gamerama: oficina de criação de jogos analógicos e digitais.	Construção de jogos como dispositivo de ensino e aprendizagem em uma classe do ensino médio integrado à habilitação técnica em saúde.

Fonte: Próprio autor.

Scaico et al. (2013) apresentam o relato de uma experiência que utiliza uma abordagem que concentra esforços não apenas em ensinar uma

linguagem de programação, mas também em estabelecer situações nas quais os alunos possam se interessar pela exploração de novos assuntos, além de reconhecerem o poder de criação oportunizado pelas TDIC. Os autores relatam que apesar das inúmeras deficiências apresentadas pelos estudantes (relacionadas à escrita, à leitura e à fundamentação lógico-matemática), foi possível ensinar e aprender os conceitos básicos de programação "ao passo que se aprendia também sobre o mundo através da própria programação" (SCAICO et al, 2013).

Kasperavicius et al. (2008) discorrem sobre uma experiência com ensino de desenvolvimento de jogos digitais com aplicação de metodologias ágeis de desenvolvimento de *software* e seus impactos significativos no processo de ensino e aprendizagem de alunos de um curso superior em Tecnologia em Jogos Digitais. Os autores se utilizam de uma metodologia institucional chamada "Aprender na Prática" que prevê "a ação educativa na participação ativa e crítica do aluno em sua aquisição de conhecimentos práticos e teóricos" (KASPERAVICIUS et al., 2008). Tal prática didática permitiu que os alunos desenvolvessem, no decorrer da disciplina, jogos com ordem progressiva de complexidade, os quais integram o portfólio profissional inicial dos alunos.

Domingos e Recena (2010) avaliaram a influência do planejamento e elaboração de jogos didáticos, por alunos do ensino médio, na construção de conhecimentos científicos de tópicos de química orgânica, especificamente funções oxigenadas e nitrogenadas. Segundo os autores, durante a execução do projeto os alunos demonstraram interesse em buscar novas informações e de resgatar conceitos prévios criando relacionamentos com conhecimentos já alcançados. Os autores afirmam o seguinte:

> [...] essa foi uma proposta pedagógica de uma aula que envolveu entretenimento, integração e disciplina no momento em que os alunos são dispostos a construir seu próprio material didático favorecendo oportunidades para construção de conhecimentos (DOMINGOS; RECENA, 2010, p. 280).

Pontes (2013) descreve em seu trabalho um estudo sobre o desenvolvimento de jogos como parte do processo de ensino de algumas disciplinas do currículo de programação pertencentes à área de TI. O objetivo foi determinar se e como a criação desses jogos poderia estimular o aprendizado dos fundamentos expostos ao aluno em sala de aula, além

de averiguar se esses programas auxiliam na medição dos conhecimentos incorporados pelos alunos. O autor destaca o desenvolvimento de diferentes jogos, conforme os tópicos ministrados em cada uma das disciplinas. Pontes (2013) conclui que os resultados demonstram que o uso da metodologia de desenvolvimento de jogos estimula o aprendizado dos conceitos de programação, além de ampliar a motivação e a confiança do discente no aperfeiçoamento das competências que o definem como profissional de TI.

Dias, Borges e Pereira (2016) apresentam uma experiência de construção de jogos como apoio ao processo de ensino e aprendizagem em uma unidade curricular de um curso técnico em saúde integrado ao Ensino Médio. Com o intuito de estimular a aprendizagem crítica sobre temas específicos na área da saúde, e buscar construção de conhecimento sobre a prática dos serviços de saúde no contexto do Sistema Único de Saúde (SUS), os autores promovem a realização de uma oficina de criação de jogos digitais. Tal oficina, denominada *Gamerama*, funciona como um minicurso de criação e desenvolvimento de jogos analógicos e digitais realizados desde o ano de 2007 em diversas escolas e universidades.

As obras de Kasperavicius et al. (2008), Wilson, Hainey, e Connolly (2013), Bermingham et al. (2013) e Drake e Sung (2011) apontam a importância da realização de um levantamento inicial na infraestrutura institucional para averiguar a disponibilidade de computadores e outros recursos de TI indispensáveis para a ministração da disciplina. Além disso, os autores alertam também para realização de uma inspeção nos documentos institucionais referentes aos objetivos da disciplina de modo a evitar violações inerentes às orientações mencionadas nas ementas curriculares.

Domingos e Recena (2010), Sung et al. (2011), Scaico et al. (2013), Kasperavicius et al. (2008), defendem a ideia da preparação inicial das aulas a fim de evitar improvisos, mudanças repentinas de direção da disciplina e outras vicissitudes desnecessárias que podem acarretar em um prejuízo ao processo de ensino e aprendizagem. Da mesma forma, Dias, Borges e Pereira (2016), Wilson, Hainey e Connolly (2013) e Pontes (2013) ressaltam a importância da elaboração antecipada dos roteiros de atividades avaliativas ou não, de modo que essas estejam coerentes com os tópicos e conteúdos presentes na ementa da unidade curricular.

Praticamente em todos as obras selecionados nessa revisão de literatura podemos verificar a preocupação com a ministração das aulas propriamente ditas. A forma de exposição dos conteúdos programáticos, e a responsabilidade do docente em cumprir as exigências das políticas institucionais são fatores recorrentes em quase todas as obras pesquisadas.

Além disto, identificamos também a preocupação com a aplicação das atividades práticas e avaliações de desempenhos dos alunos. Pontes (2013), Sung et al. (2011) encorajam o uso de exemplos inerentes ao desenvolvimento de jogos cada tópico abordado, de modo a preparar os alunos para as avaliações de desempenho.

Por fim, Pontes (2013), Wilson, Hainey e Connolly (2013), e Dias, Borges e Pereira (2016) destacam o valor das reflexões, análises e *feedbacks*, com coletas de dados, questionários, entrevistas, com o objetivo de identificar pontos positivos e negativos durante a implementação dessa metodologia de criação de jogos. Com tais resultados, será possível analisar e implementar melhorias no processo de aplicação da metodologia proposta neste guia.

Considerando a análise dessas obras selecionadas, identificamos quatro aspectos principais que nortearam a confecção deste guia. São pontos-chave a serem considerados na implementação e execução desta metodologia de ensino de programação de computadores baseado no desenvolvimento de jogos:

(1) Auditoria preliminar;
(2) Planejamento e Projetos;
(3) Ensino;
(4) Avaliação.

Os quatro pontos-chave serviram como base na concepção da versão inicial do guia de ensino, sendo que cada fase possui pontos específicos a serem considerados. A próxima seção versa com mais detalhes sobre a estrutura inicial do guia.

2.2 VERSÃO INICIAL DO GUIA DE ENSINO

Um protótipo inicial do guia para aprendizagem baseada na construção de jogos utilizando linguagens de programação convencionais, foi criado a partir da revisão de literatura mencionada anteriormente. O protótipo é um modelo composto por quatro estágios que tem como objetivo auxiliar o docente que pretende usar a metodologia da criação de jogos em suas disciplinas de programação inicial, mas sem abrir mão da utilização de linguagens de programação convencionais. A Figura 3 representa a estrutura do protótipo desenvolvido.

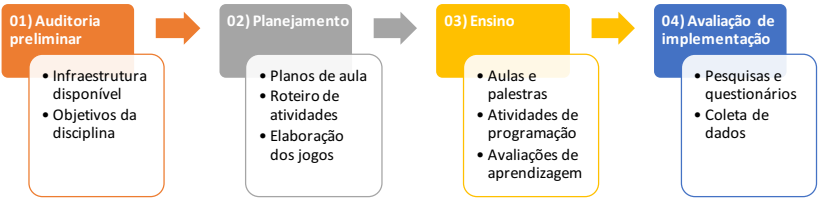

Figura 3: Diagrama estrutural do protótipo do guia
Fonte: Próprio autor.

O protótipo proposto é composto de quatro fases sequenciais de acordo com os pontos-chave identificados nas revisões de literatura mencionados anteriormente. A primeira fase consiste na auditoria inicial, na qual o docente verifica a infraestrutura disponível para condução da disciplina. É importante que a disponibilidade de computadores, ferramentas de programação e acesso à internet sejam averiguadas e consideradas para melhor execução das próximas etapas. A segunda fase refere-se ao planejamento que o professor faz para ministrar a disciplina. É essencial que o docente planeje suas aulas e atividades de modo que os alunos se sintam encorajados a criar os seus próprios jogos digitais. A terceira fase contempla a ministração de aulas, aplicação de atividades e avaliações de aprendizagem (seja por meio de provas ou trabalhos). Na quarta fase, o docente faz uso de questionários e pesquisas para colher as opiniões dos alunos sobre aspectos específicos da disciplina, como aulas, atividades, avaliações e forma de lecionar.

Com o propósito de melhorar essa versão inicial do guia de ensino, procuramos compartilhar esse protótipo com outros educadores da área de TI com experiência no campo da aprendizagem baseada em jogos.

2.3 AVALIAÇÃO DE ESPECIALISTAS

Um passo importante na elaboração deste guia consistiu na avaliação inicial do protótipo do guia. Para isto, optou-se por uma análise especializada, na qual cinco especialistas em aprendizagem baseada em jogos e educação em TI, foram convidados a apreciar o protótipo e fornecer suas opiniões e considerações. Um breve conjunto de informações sobre os especialistas está exposto no Quadro 6.

Quadro 6: Informações situacionais sobre os avaliadores

Avaliador no.	Área de docência	Anos de experiência em docência	Publicações nos últimos 5 anos
Avaliador #1	Computação (Jogos Digitais)	9	11
Avaliador #2	Computação (Educação Tecnológica)	13	12
Avaliador #3	Computação (Linguagens de Programação)	11	14
Avaliador #4	Computação (Linguagens de Programação)	10	6
Avaliador #5	Computação (Jogos Digitais)	8	8

Fonte: Próprio autor.

O objetivo da avaliação era obter pareceres sobre o protótipo e assim detectar possíveis fragilidades antes que o guia fosse utilizado por professores em suas disciplinas. Depois de analisar o modelo inicial, os avaliadores receberam e responderam um questionário com cinco perguntas abertas: uma para cada estágio do protótipo e outra para o protótipo como um todo. Além disso, foram questionados sobre possíveis sugestões de melhoria. Em geral, os pareceres dos avaliadores foram positivos e as seguintes sugestões foram feitas.

- Primeiro estágio: Seria interessante que o professor considerasse as normas de codificação e estilos de programação, ou seja, os padrões de projeto. Um guia para melhores práticas e as regras mais comuns poderiam ser definidos na primeira fase.
- Segundo estágio: (a) Considere o planejamento dos materiais de modo que possam tornar as condições dos alunos as mais iguais possíveis; (b) Ao escolher um jogo para uma atividade, é fundamental que este seja um jogo já existente e bem conhecido (ou o mais semelhante possível); (c) Recomenda-se que as avaliações sejam planejadas antecipadamente, isto é, durante a fase de planejamento e projetos.
- Terceiro estágio: É importante que o professor cuide para que a apresentação dos jogos tome a forma de um processo a ser desenvolvido (ou problema a ser resolvido), não como algo para se

divertir. Dessa forma, a integridade acadêmica do curso será preservada.

- Quarto estágio: De modo a tornar os *feedbacks* mais proveitosos, seria conveniente que os comentários finais fossem de ambos os lados, isto é, de professores para estudantes e de estudantes para professores. Alterar o nome de Avaliação para Reflexões.

Estas sugestões feitas pelos avaliadores foram usadas para otimizar o protótipo implementando melhorias no formato inicial. A estrutura do guia otimizado – revisado por esse painel de avaliadores – é descrita a seguir.

2.4 GUIA OTIMIZADO

Com base na revisão da literatura, juntamente com as análises observacionais dos avaliadores, apresentamos o guia otimizado para integrar a criação de jogos em unidades curriculares de programação inicial. Uma visão geral do guia é apresentada na Figura 4. Este modelo apresenta os quatro estágios de melhores práticas a serem utilizadas pelos professores que desejam implementar tal metodologia. Além disso, ao final do capítulo apresenta-se um resumo de alto nível do guia, fornecendo uma visão mais detalhada da implementação.

O guia tem quatro etapas, ou fases: Auditoria Preliminar; Planejamento e Projetos; Ensino e aprendizagem; Reflexões. Cada fase tem um conjunto de passos específicos com orientações que ajudarão os professores a implantar a metodologia. A execução é conduzida pelos professores e é primordial que em cada etapa eles assumam e demonstrem uma forte liderança a fim de cativar os alunos e progredir com a implementação.

Figura 4: Diagrama de implementação do guia otimizado
Fonte: Próprio autor.

2.4.1 Público alvo

A quem se destina este guia? Antes de nos aprofundarmos no conteúdo especificamente, caracterizaremos qual é o público alvo deste guia de ensino de programação baseado na construção de jogos. Começaremos pelo perfil das unidades curriculares, e seguiremos pelos perfis dos cursos, alunos e, por fim, o do docente.

Quais disciplinas? Este é um guia para o ensino de programação, ou seja, ensino de desenvolvimento de *softwares* e aplicativos. Seu objetivo é fornecer diretrizes para implementação de uma abordagem de ensino baseada na construção de jogos. Em sua essência, destina-se às disciplinas introdutórias de programação, as quais – em geral – configuram-se como o primeiro contato dos alunos com os conceitos de desenvolvimento de *softwares* e aplicações. E, por isso, tal abordagem pode ser mais eficiente nessas disciplinas introdutórias, uma vez que elas são a principal causa de evasão e desistência em cursos de TI. Diante disso, o uso de jogos pode ser usado como fator motivador no processo de ensino e aprendizagem. Porém, não encontramos barreiras que impossibilitem a utilização dessa ferramenta em disciplinas avançadas de programação, como 'Programação para Dispositivos Móveis',

‘Inteligência Artificial’, ‘Programação para Internet’, e ‘Programação Orientada a Objetos’, desde que realizados os devidos ajustes.

Quais cursos? As unidades curriculares de programação não são exclusividade dos cursos – sejam eles de nível técnico ou superior – voltados para análise e desenvolvimento de *softwares* como ‘Técnico em Informática’, ‘Técnico em Desenvolvimento Web’, ‘Ciência da Computação’, ‘Sistemas de Informação’ e ‘Análise e Desenvolvimento de Sistemas’. Os cursos de Engenharia (Elétrica, Mecânica, Produção, Mecatrônica, Controle e Automação, Telecomunicações e outras) também contam com uma ou mais disciplinas de programação. Porém, considerando que um jogo digital é um *software*, esse tipo de abordagem pode despertar mais interesse – e assim ser mais eficaz – em alunos de cursos de TI (técnico ou superior) direcionados para análise e desenvolvimento de *softwares*. Alunos dos cursos de Engenharia podem ter uma motivação maior nas aulas de programação quando munidos de atividades concernentes a sistemas robóticos e/ou geração e transmissão de energia.

Quais alunos? Os videogames, juntamente com os jogos eletrônicos, se tornaram mais populares na década de 1980 e sua popularidade entre jovens e crianças tem aumentado vertiginosamente desde então. As crianças da década de 1980 estão hoje na casa dos 30-40 anos, e embora as últimas duas décadas do século XX tenham sido apenas o início da era dos jogos eletrônicos, é grande a chance de um adulto dessa faixa etária ter vivido a sua infância acompanhado de um videogame, seja próprio ou de amigos. A maioria dos jovens de até 20 anos já nasceram após a virada do século e, consequentemente, cresceram em um mundo altamente digital e tecnológico, cercados de modernos aparatos digitais, videogames de última geração e minicentrais de entretenimento particulares também conhecidas como *smartphones*. Esses jovens, chamados de ‘nativos digitais’ por Prensky (2010), são o público alvo desse guia. São alunos nascidos nas últimas duas décadas, podendo estender até quatro décadas se considerarmos os alunos/jogadores precursores em sua infância/juventude nos anos oitenta do século XX. Sendo assim, o perfil do estudante alvo deste instrumento são aqueles ingressantes no ensino médio (ou técnico) e superior, com idade entre 15 e 20 anos, ambos habilitados com capacidade de abstração, maior potencial de aprendizagem, com intrínseca volição para jogos.

Quais docentes? Considerando o delineamento dos três perfis anteriores (disciplinas, cursos, alunos), poderíamos, de modo rudimentar, definir que o perfil do docente alvo seria o de um professor de uma disciplina de programação, em um curso direcionado para análise e

desenvolvimento de *softwares*, com alunos de até 40 anos de idade. Todavia, isso não é o bastante, é preciso mais. Deve partir do docente a iniciativa de adotar essa abordagem de construção de jogos, ou pelo menos, que este tenha interesse em utilizá-la, uma vez que a imposição, por vezes forçada, de mudanças de paradigmas no trabalho docente não é saudável para o processo de ensino e aprendizagem. Uma vez que o docente tenha o interesse em adotar esse instrumento, é necessário também que este tenha tempo para organização e planejamento da adoção, pois a falta de planejamento pode acarretar problemas na condução da metodologia. É interessante também que o docente interessado manifeste disposição para mudar, para quebrar paradigmas de ensino por vezes enraizado nos moldes tradicionais de instrução. Por fim, não há a necessidade de que o docente seja um especialista em jogos tradicionais, tampouco eletrônicos, e conhecer todas as nuances de tudo que está envolvido no universo dos jogos, entretanto é primordial que ele tenha disposição para aprender um pouco sobre, e se envolver com jogos caso não tenha nenhuma familiaridade com o tema.

Definidos os perfis para quem esta abordagem se dirige, nos aprofundaremos – fase por fase – nas ideias e conceitos deste guia de ensino de programação baseado na construção de jogos.

2.5 FASE 1 – AUDITORIA PRELIMINAR

A primeira fase refere-se à etapa de Auditoria Preliminar. Nessa primeira parte, o professor foca seus esforços para analisar quatro componentes: (a) políticas e normas institucionais; (b) objetivos da disciplina; (c) infraestrutura disponível; (d) padrões de codificação e estilos. Com isso, o docente terá uma visão geral do ambiente acadêmico que ele terá a disposição para lecionar a disciplina, para assim poder tomar melhores decisões.

2.5.1 Políticas e normas institucionais

Para evitar a violação de qualquer regra institucional, é fundamental que o comitê departamental, ou a coordenação pedagógica do curso, estejam cientes e concordem com mudanças significativas na condução da disciplina.

Onde pode ser encontrado? As políticas e normas institucionais são conjuntos de regras e diretrizes que norteiam o funcionamento de instituições. Esses regulamentos podem ser encontrados em documentos institucionais, tais como: Regimento Interno da instituição, Projeto Político Pedagógico, Ementas ou Plano de Curso das disciplinas.

O que verificar? Um dos aspectos a serem averiguados diz respeito à flexibilidade quanto à temática das atividades de ensino. Por exemplo, é importante checar se é possível utilizar a temática de construção de jogos. Outra informação a ser examinada refere-se à flexibilidade quanto ao uso de linguagens de programação em modo texto. Por exemplo, é relevante conferir se é possível a utilização de qualquer linguagem de programação. Nesse caso, é importante também que o docente averigue a obrigatoriedade da utilização de uma linguagem, ou ambiente, de programação específica.

2.5.2 Objetivos da disciplina

É indispensável que o docente cuidadosamente revise os objetivos da disciplina para boa orientação no momento de preparo de aulas e atividades na segunda etapa.

Onde pode ser encontrado? Os objetivos da disciplina são as expectativas de aprendizado, ou seja, é o que se espera que os alunos tenham aprendido ao encerramento da disciplina. Esses objetivos podem ser encontrados em documentos institucionais, tais como: Projeto Político Pedagógico, Ementários de disciplinas, Planos de curso.

O que verificar? O primeiro aspecto a ser examinado é o conteúdo programático da disciplina. Nesse ponto, o docente tenta responder questões como: Quais assuntos serão abordados ao longo da disciplina? Quais são os conceitos que o aluno aprenderá até o fim da disciplina? O que se espera que o aluno tenha aprendido ao encerrar a disciplina?

Outros pontos importantes a serem considerados são a verificação da exigência de uma metodologia de ensino específica a ser empregada, e também analisar qual é o sistema de avaliação e atribuição de pontos ou conceitos. É importante que o docente tenha ciência dessas informações para estar municiado no momento do planejamento e preparo das atividades (avaliativas ou não-avaliativas).

2.5.3 Infraestrutura disponível

No sentido de garantir o suporte adequado à infraestrutura, é recomendado que o professor verifique se as ferramentas e recursos estão disponíveis gratuitamente e se as demais demandas relacionadas à infraestrutura estão acessíveis.

Onde pode ser encontrado? O setor ou departamento de Tecnologia da Informação da instituição pode fornecer informações detalhadas sobre a disponibilidade de recursos para condução da disciplina. Caso não exista um setor responsável pela TI na instituição, o

próprio docente pode averiguar *in loco* as condições dos laboratórios de informática.

O que verificar? É essencial que o docente saiba qual é a quantidade de computadores disponíveis para ministrar a disciplina. Além de conhecer quantidade de dispositivos, é importante inteirar-se dos recursos computacionais disponíveis em cada computador. Sugerimos a verificação da disponibilidade de plataformas de programação, ambientes de desenvolvimento integrado (IDE) e bibliotecas de programação que atendam aos objetivos da disciplina e também não contrariem nenhuma norma ou política institucional.

2.5.4 Padrões de codificação e estilos

Padrões de codificação são um conjunto de diretrizes para uma linguagem de programação específica que recomendam o estilo de programação, práticas e métodos para cada aspecto de um programa escrito nessa linguagem. Essas convenções geralmente cobrem organização de arquivos, indentação de códigos fonte, comentários, declarações de variáveis, espaços em branco, convenções de nomenclatura, práticas e princípios de programação, além de melhores práticas estruturais.

Os padrões de codificação atendem às seguintes finalidades:

1. Criar uma aparência consistente para o código, para que os leitores possam se concentrar no conteúdo e não no *layout*;
2. Permitir que os leitores entendam o código com mais rapidez, fazendo suposições com base na experiência prévia;
3. Facilitar a cópia, a alteração e a manutenção do código.

Embora seja um aspecto pouco presente nas disciplinas de introdução à programação, o docente pode considerar a recomendação de padrões de codificação e estilos. O objetivo principal é preparar os alunos para o mercado de trabalho, uma vez que este é um conceito muito presente em empresas desenvolvedoras de *softwares*.

Em geral, tais padrões de codificação não são encontrados nos documentos institucionais, como as ementas das disciplinas. Sendo assim, a recomendação, ou não, da aplicação dessas diretrizes é uma decisão particular do docente. O conjunto de diretrizes de programação pode ser definido pelo próprio professor. Caso o docente não se sinta apto a fazê-lo, sugerimos analisar padrões de codificação já existentes e disponíveis na Internet. Tome como sugestão inicial os seguintes termos de busca:

- padrões de codificação <*nome da linguagem*>

    - Por exemplo: padrões de codificação C
- estilo de codificação <*nome da linguagem*>
    - Por exemplo: estilo de codificação Java

É importante ressaltar que, em geral, as empresas têm suas próprias políticas de codificação que aguardam os alunos no mercado de trabalho, por isso consideramos importante o uso de padrões de codificação e estilos.

2.6 FASE 2 – PLANEJAMENTO E PROJETOS

A segunda fase é a etapa de planejamento e projetos (ou delineamento de ações). Uma vez que haja um quadro claro das circunstâncias em termos de políticas institucionais, objetivos da disciplina, suporte de infraestrutura e padrões de codificação e estilos, é o momento de considerar os aspectos pedagógicos da implementação. Nesta fase, o docente planeja e elabora as aulas, palestras e atividades que serão aplicadas aos alunos.

2.6.1 Planos de aula detalhados

Planos de aulas detalhados consideram o número de computadores disponíveis, o número de alunos na classe, bem como os objetivos que o professor deseja alcançar durante o processo de aprendizagem. Durante o planejamento das aulas, é relevante considerar o contexto e pedagogia adequados para preservar a integridade acadêmica da disciplina e do curso em si.

Em geral, o conteúdo programático de uma disciplina de programação já está definido em sua ementa. Não é o nosso objetivo aqui propor ou sugerir ementas, ou conteúdos programáticos, para as disciplinas de programação. Cada departamento, e colegiado de curso, tem essa responsabilidade inerente em suas atribuições. Os tópicos abordados nas disciplinas de programação introdutória não variam muito entre instituições, sendo assim os tópicos principais são comuns entre várias instituições.

Com uma sucinta investigação em ementas disponíveis na internet, podemos traçar um panorama geral dos tópicos que são comumente abordados nessas unidades curriculares em questão. Com o objetivo de investigar quais os assuntos mais comuns nessas disciplinas, e assim traçar um panorama geral, selecionamos 10 ementas de disciplinas de programação introdutória. A seguir relacionamos os nomes das disciplinas com as suas respectivas cargas horárias (CH) em horas por semestre, as linguagens de programação utilizadas, os cursos nos quais elas são ofertadas e as universidades.

- Programação I
    - 60 horas/semestre.
    - Linguagem de programação C (indicada na bibliografia, mas não indicada no conteúdo programático).
    - Ciência da Computação; Engenharia da Computação; Engenharia de Controle e Automação.
    - Pontifícia Universidade Católica – Rio de Janeiro (PUC-RIO).
- Programação I
    - 80 horas/semestre.
    - Linguagem de programação não indicada.
    - Ciência da Computação.
    - Universidade Federal Fluminense (UFF)
- Introdução aos Algoritmos
    - 102 horas/semestre.
    - Linguagens de programação C/C++ (indicadas na bibliografia, mas não indicadas no conteúdo programático).
    - Ciência da Computação; Sistemas de Informação.
    - Universidade Federal de Lavras (UFLA)
- Algoritmos e Estruturas de Dados I
    - 60 horas/semestre.
    - Linguagens de programação C/C++ (indicadas na bibliografia e também no conteúdo programático).
    - Ciência da Computação.
    - Universidade Federal de Minas Gerais (UFMG)
- Algoritmos e Estruturas de Dados I
    - 60 horas/semestre.
    - Linguagem de programação Pascal (indicada na bibliografia e também no conteúdo programático).
    - Ciência da Computação; Informática Biomédica.
    - Universidade Federal do Paraná (UFPR)
- Algoritmos e Programação
    - 60 horas.
    - Linguagem de programação C (indicada na bibliografia, mas não indicada no conteúdo programático).
    - Ciência da Computação; Engenharia da Computação.
    - Universidade Federal do Rio Grande do Sul (UFRGS)
- Introdução à Ciência da Computação
    - 54 horas/semestre.

    - Linguagem de programação Pascal (indicada na bibliografia e também no conteúdo programático).
    - Ciência da Computação.
    - Universidade Federal de Santa Catarina (UFSC)
- Programação Procedimental
    - 90 horas.
    - Linguagem de programação C (não indicada na bibliografia, mas indicada no conteúdo programático).
    - Ciência da Computação.
    - Universidade Federal de Uberlândia (UFU)
- Algoritmos e Programação de Computadores
    - 90 horas/semestre.
    - Linguagem de programação C (indicada na bibliografia, mas não indicada no conteúdo programático).
    - Ciência da Computação.
    - Universidade Estadual de Campinas (UNICAMP)
- Introdução à Computação
    - 60 horas/semestre.
    - Linguagem de programação C (indicada na bibliografia, mas não indicada no conteúdo programático).
    - Ciência da Computação; Matemática Aplicada e Computação Científica; Sistemas de Informação.
    - Universidade de São Paulo (USP)

A essência de todas as ementas selecionadas é a mesma, ou seja, introduzir os conceitos principais para o aprendizado de programação. Contudo, os nomes das disciplinas são diferentes entre as universidades bem como a sua carga horária semestral. Os conteúdos programáticos expostos em cada ementa são também diversificados entre si. Porém, podemos perceber a existência de assuntos que são comuns em todas elas. Após a coleta das ementas foi realizado um comparativo para examinar quais são os tópicos mais populares entre as ementas selecionadas. O Quadro 7 mostra uma relação com os assuntos mais comuns entre as ementas selecionadas e indica a presença (S), ou ausência (-), de cada um deles em cada ementa analisada.

Quadro 7: Tópicos comuns entre as ementas

Tópicos	Universidades									
	PUC-RIO	**UFF**	**UFLA**	**UFMG**	**UFPR**	**UFRGS**	**UFSC**	**UFU**	**UNICAMP**	**USP**
Entrada e saída de dados	S	S	S	S	S	S	S	S	S	S
Variáveis	S	S	S	S	S	S	S	S	S	S
Comandos condicionais	S	S	S	S	S	S	S	S	S	S
Comandos de repetição	S	S	S	S	S	S	S	S	S	S
Vetores e matrizes	S	S	S	S	S	S	S	S	S	S
Manipulação de arquivos	-	S	S	S	-	-	S	S	S	-
Funções e procedimentos	S	S	S	S	S	S	S	S	S	S
Recursividade	-	-	S	-	-	-	-	-	S	-
Ponteiros	-	-	S	-	-	-	-	-	-	-
Busca e ordenação	-	-	S	-	S	-	-	-	S	-

Fonte: Próprio autor com dados contidos nas ementas selecionadas.

Quando um determinado tópico é abordado em uma ementa e não abordado em outra, possivelmente esse assunto é tratado em outra unidade curricular do mesmo curso. Com o objetivo de posteriormente exemplificar o processo de elaboração de atividades de programação com a temática de construção de jogos, separamos um conjunto composto de seis conceitos essenciais na disciplina de programação, descrevendo o tópico e apresentando possíveis ações.

Conceito
Entrada e saída de dados
Descrição
Comandos de entrada são utilizados para receber informações do jogador/usuário via teclados, *mouses*, *joysticks* (botões), microfones e inseri-las no jogo. Comandos de saída são utilizados para retornar informações do jogo para o jogador via telas, monitores, alto-falantes, *joysticks* (vibração).
Ações
Permitem a troca de informações e dados entre jogador e jogo, entre usuário e *software*. O jogador insere as informações via comandos de entrada; o jogo processa tais informações de acordo com as regras estabelecidas pelo desenvolvedor; o jogo então retorna as informações processadas via comandos de saída.

Conceito
Variáveis
Descrição
São utilizadas para armazenar e alterar valores (números, textos, cores, sons e imagens) dentro do *software*.
Ações
Diferentes ações para diferentes tipos de variáveis. Números podem ser somados e subtraídos. Imagens podem ser recoloridas ou redimensionadas. Textos podem ser concatenados ou fragmentados.

Conceito
Comandos condicionais
Descrição
São consideradas estruturas de controle (aliadas aos comandos de repetição) utilizadas para especificar o fluxo de execução de um programa. Tais comandos seletivamente executam ou ignoram um conjunto de instruções sob diferentes condições.
Ações
Representam decisões e são compostos de uma condição booleana que especifica ações a serem executada conforme a avaliação da condição, que pode ser verdadeira ou falsa. A lógica booleana e seus operadores (por exemplo: E, OU, NÃO) podem ser utilizados para especificar o conjunto de instruções a serem executados sob determinadas condições.

Conceito
Comandos de repetição
Descrição
São consideradas estruturas de controle (aliadas aos comandos condicionais) utilizadas para especificar o fluxo de execução de um programa. As estruturas de repetição controlam a quantidade de execuções de um grupo de instruções, de acordo com condições estabelecidas pelo programador.
Ações
Representam decisões e são compostos de uma condição que especifica a quantidade de execuções de um conjunto de instruções conforme a avaliação da condição. Um determinado grupo de instruções pode ser executada uma ou infinitas vezes, ou pode até mesmo ser ignorado.

Conceito
Vetores e matrizes
Descrição
Também são variáveis, contudo podem armazenar e alterar grupos de valores do mesmo tipo (conjuntos de números, grupos de textos). Vetores contêm uma dimensão, matrizes são multidimensionais.
Ações
Funcionam como as demais variáveis. São úteis na implementação de estruturas de dados, tais como: tabelas, listas, pilhas e filas.

Conceito
Funções e procedimentos
Descrição
Os programas usam procedimentos para organizar o código, ocultar detalhes de implementação e tornar o código mais fácil de reutilizar. Os procedimentos podem ser reutilizados em novos programas. A definição de parâmetros para procedimentos pode generalizar o comportamento e aumentar a reutilização.
Ações
As tarefas complexas podem ser divididas em instruções mais simples, algumas das quais podem ser quebradas ainda mais. Da mesma forma, as instruções podem ser combinadas para realizar tarefas complexas.

É importante detalhar os planos de aulas de modo que o professor tenha segurança no momento de ministrar as aulas e aplicar as atividades, principalmente conhecer o conteúdo programático para alinhar os tópicos e objetivos da disciplina com as circunstâncias da construção de jogos.

2.6.2 Escolha dos jogos

A fase de planejamento das atividades também contempla a seleção dos jogos a serem criados pelos alunos em suas atividades de programação. Os professores podem preferir escolher um jogo existente ao invés de inventar um novo jogo, uma vez que escolher jogos já conhecidos pelos alunos exige menos tempo de explicação e entendimento das regras. No entanto, essa prática não inviabiliza a criação de novos jogos pelos alunos, principalmente quando estes forem variações de jogos já existentes. Jogos altamente complexos não encontram lugar nestas disciplinas iniciais de programação, uma vez que não há tempo

hábil necessário para o desenvolvimento desses jogos, além de exigir um nível de conhecimento de programação mais avançado, o que não é encontrado em alunos das séries iniciais.

Sendo assim, sugerimos que as regras sejam simples e de fácil implementação. Além do tempo de desenvolvimento, indicamos também que o tempo gasto para jogar o jogo também seja considerado. Ao escolher jogos que têm um tempo de jogabilidade relativamente curto, discentes e docentes têm um *feedback* mais rápido, além de requerer prazo menor para implementação.

Como forma de sugestão, listamos uma relação de jogos simples que podem ser utilizados durante o processo de elaboração de atividades. Os jogos listados são aplicações simples, com tempo de jogabilidade pequeno, e alguns com possibilidades de desenvolvimento de variações.

Jogo: Par ou ímpar.

Descrição: Brincadeira popular jogada entre duas pessoas, em geral usada para decidir um impasse. Os jogadores se preparam decidindo a quem será atribuído "par" e quem será "ímpar". Então, simultaneamente, mostram as mãos escondendo ou exibindo alguns dedos. A soma total dos dedos exibidos é par ou ímpar. Se o resultado é par, então a pessoa que foi atribuída "par" é o vencedor e pode decidir o impasse.

Possíveis variações: Caso seja necessário decidir um impasse entre três pessoas, uma variação pode ser aplicada. Conhecida como "dois-ou-um", a brincadeira consiste em os três jogadores simultaneamente mostrarem os dedos da mão, porém só é permitido mostrar dois ou um dedo. Dois jogadores mostrarão opções iguais, e um diferente. O jogador que revelou a opção diferente dos demais decide o impasse.

Jogo: Pedra-Papel-Tesoura (*Jokenpo*).

Descrição: Também conhecido como *Jokenpo*, é um jogo geralmente jogado entre duas pessoas que simultaneamente formam uma de três possíveis formas com uma mão estendida. Essas formas são "pedra" (um punho fechado), "papel" (uma mão aberta) e "tesoura" (uma mão com os dedos indicador e médio exibidos formando um V). A vitória é definida da seguinte forma: "pedra vence tesoura"; "papel vence pedra"; "tesoura vence papel". Se ambos os jogadores escolherem a mesma forma, o resultado é o empate.

Possíveis variações: Há diversas variações. A maioria consiste em simplesmente mudar as formas disponíveis e os seus nomes, mas a

essência do jogo permanece a mesma. A variação mais famosa é a Pedra-Papel-Tesoura-Lagarto-Spock que adiciona outras duas opções às três originais. Dessa forma, o critério de vitória fica definido assim: “pedra vence tesoura”; “pedra vence lagarto”; “papel vence pedra”; “papel vence Spock”; “tesoura vence papel”; “tesoura vence lagarto”; “lagarto vence papel”; “lagarto vence Spock”; “Spock vence tesoura”; “Spock vence pedra”.

Jogo: Disputa de pênaltis.

Descrição: Baseado em uma disputa de pênaltis, como no futebol, disputado por dois jogadores, sendo um chutador e um defensor/goleiro alternando as funções a cada rodada. O gol é dividido em 9 posições, três no topo, três no meio e três em baixo. As posições são numeradas de 1 a 9 sequencialmente. O chutador escolhe uma posição para chutar, e o goleiro escolhe uma posição para defender. Se o chutador escolher uma posição diferente do goleiro, ele marca um gol. Vence quem marcar mais gols.

Possíveis variações: Em sua forma original, não há possibilidade do chutador chutar na trave, ou para fora do gol. Uma variação pode ser desenvolvida implementando alguma probabilidade de erro do chutador durante a cobrança, chutando a bola na trave ou para fora do gol. Outra variação seria alterar o tamanho do gol de 9 para 4, 6, ou 12 posições por exemplo.

Jogo: Memória.

Descrição: Popular jogo de cartas em que todas as cartas são posicionadas sobre uma superfície com a face para baixo. Duas cartas são viradas para cima em cada turno. O objetivo do jogo é virar pares de cartas correspondentes. Pode ser jogado com qualquer número de participantes, um ou mais jogadores.

Possíveis variações: Em geral, as cartas correspondentes são cartas iguais. Como variação, as cartas podem ter relações entre si tais como: perguntas e respostas; operações matemáticas e seus resultados; imagens e suas descrições; países e suas capitais.

Jogo: Forca.

Descrição: É um jogo de adivinhação de palavras para dois ou mais jogadores. Um jogador pensa em uma palavra e o outro tenta adivinhar sugerindo letras dentro de um certo número de tentativas.

Possíveis variações: O jogo pode ter variações ao incluir limitações nas sugestões de letras, por exemplo limitar as sugestões de vogais, ou até mesmo excluir a sugestão de vogais.

Jogo: Torres de Hanói.

Descrição: Consiste em três hastes, e uma quantidade de discos de tamanhos diferentes que podem se mover em qualquer haste. O quebra-cabeça começa com os discos em uma pilha ordenada em ordem crescente de tamanho em uma haste, o menor no topo, tornando assim uma forma cônica. O objetivo do quebra-cabeça é mover a pilha inteira para outra haste, obedecendo as seguintes regras: (A) Apenas um disco pode ser movido de cada vez; (B) Cada movimento consiste em retirar o disco superior de uma das pilhas e colocá-lo sobre uma outra pilha, isto é, um disco só pode ser movido se for o disco mais superior numa pilha; (C) Nenhum disco pode ser colocado sobre um disco menor.

Possíveis variações: As variações do jogo Torres de Hanói consistem em alterar a quantidade de discos, e também a quantidade de hastes disponíveis.

Jogo: Jogo da Velha.

Descrição: Jogo para dois jogadores que recebem os símbolos "X" e "O", respectivamente. Alternadamente, os jogadores assinalam seus símbolos nos espaços de uma grade 3x3. O jogador que conseguir colocar três de seus símbolos em uma linha horizontal, vertical ou diagonal ganha o jogo.

Possíveis variações: As variações do jogo da velha são muitas. Uma delas consiste em alterar o tamanho da grade de 3x3 para 4x4. Outra variação consiste em adicionar outras duas grades de 3x3 e alterar a condição de vitória adicionando a possibilidade de um jogador assinalar o seu símbolo na mesma posição nas três grades, também conhecido como jogo da velha em três dimensões.

Jogo: Cobras e escadas.

Descrição: Jogo de tabuleiro jogado entre dois ou mais jogadores em um tabuleiro com quadrados (casas) numerados, dispostos em forma de grade. Uma casa do tabuleiro pode ter três situações: estar vazia, ter uma cobra ou ter uma escada. O objetivo do jogo é locomover-se da primeira até a última casa do tabuleiro. A cada rodada um jogador lança um dado comum, e o número sorteado no dado indica a quantidade de casas que serão percorridas. Se parar em uma casa vazia, o jogador permanece nesta casa vazia até a próxima rodada. Se parar em uma casa

com uma escada, o jogador se desloca para frente uma quantidade de casas indicada na escada. Se parar em uma casa com uma cobra, o jogador se desloca para trás uma quantidade de casas indicada na cobra. Vence quem chegar primeiro na última casa do tabuleiro.

Possíveis variações: A principal variação é o tamanho do tabuleiro e a quantidade de casas a serem visitadas. Outra variação é a introdução de temáticas diferentes de "cobras" e "escadas", utilizando outros símbolos para casas que avançam, ou atrasam, o jogador.

Jogo: Quizzes (Jogos de Perguntas e Respostas).

Descrição: Jogo de perguntas e respostas como visto em programas populares de televisão. Cada pergunta tem quatro ou mais alternativas de respostas. Se o jogador responder corretamente uma pergunta, ele avança para a próxima rodada. A cada rodada o grau de dificuldade das perguntas aumenta. O jogo termina com derrota para o jogador, se ele responder incorretamente alguma pergunta do jogo. E termina com vitória se o jogador responder corretamente a última pergunta do jogo.

Possíveis variações: São muitas as variações. Pode-se incluir temáticas nas perguntas e respostas ao invés de generalizar o tema das questões. Pode-se também fornecer recursos para que o jogador avance as rodadas, como dicas, eliminação de alternativas erradas, possibilidade de evadir algumas perguntas.

A relação de jogos apresentados serve como sugestão e não procura ser considerada como uma lista definitiva a ser aderida irrefletidamente pelo professor. Tal lista tem como objetivo funcionar como um guia que oriente o professor a escolher os jogos que serão desenvolvidos pelos alunos, levando em consideração os aspectos mencionados anteriormente, como tempo de jogabilidade e a simplicidade das regras.

Na maioria dos jogos, o conceito de aleatoriedade é importante, seja para evitar sucessivas e entediantes repetições, simular a presença de um outro jogador, ou ainda para implementar o lançamento de um dado. Por isso, sugerimos que o docente mostre aos alunos como gerar números aleatórios na linguagem de programação utilizada na disciplina. Algumas linguagens de programação já possuem comandos simples específicos para geração de números randômicos, contudo em outras será necessário implementar um pequeno conjunto de instruções.

2.6.3 Elaboração de atividades

Entenda-se por atividades as práticas de programação que serão propostas pelo docente após a ministração de um tópico ou assunto da disciplina com o objetivo de fixação de conteúdo e avaliação formativa do discente. As considerações acerca da elaboração de atividades dependem da neutralidade e dos níveis de desafio que o professor deseja para sua classe. É importante observar a neutralidade do material para evitar a alienação de grupos e minorias, uma vez que as habilidades dos alunos variam amplamente dentro e entre as classes. Por isso, recomendamos que os materiais elaborados sejam neutros em termos de gênero e experiência. Tópicos violentos, competição desnecessária e imagens supérfluas não são encorajados.

Para evitar sobrecargas de atividades, é essencial que os níveis de desafio projetados para as tarefas de programação de jogos sejam ajustados na mesma medida da quantidade de trabalhos desempenhados pelos alunos. O nível de desafio pode ser determinado desde o básico, como fazer pequenas modificações em um jogo existente, até o avançado, como o desenvolvimento de um jogo completo do início ao fim.

Com o objetivo de exemplificar a elaboração de uma atividade de programação baseada na construção de jogos, relacionamos os conceitos essenciais descritos na seção 2.6.1 com as sugestões de jogos listados na seção 2.6.2, como forma de atividades práticas.

Inicialmente provemos uma visualização por conceitos, onde elencamos os tópicos de programação na ordem em que, geralmente, aparecem nas ementas das disciplinas. Nesta visualização por conceitos, exibimos cada assunto seguido de possíveis aplicações em jogos, e algumas recomendações acerca da sugestão de relação entre tópicos de programação e desenvolvimento de jogos.

Conceito: Entrada e saída.

Aplicações em jogos: Comandos de entrada e saída serão utilizados para receber dados do jogador e exibir informações diversas sobre o jogo.

Comandos de saída: Atividades práticas simples, como exibir o nome do jogo na tela, apresentar um texto com as regras do jogo e mostrar textos com orientações sobre as ações requeridas para que o jogador possa prosseguir em um jogo.

Comandos de entrada: Ao receber informações do jogador, tais dados são armazenados em variáveis para posterior processamento. Por isso, sugerimos que as atividades práticas inerentes aos comandos de

entrada sejam propostas em conjunto com as atividades referentes às variáveis.

Recomendações: neste primeiro momento é relevante que as atividades não contenham processamentos de informações. Além disso, é essencial que as práticas de programação se atenham a instruir o aluno sobre como definir a exibição de mensagens na tela.

Conceito: Variáveis.

Aplicações em jogos: O conceito de variáveis é um dos primeiros conceitos a serem introduzidos na disciplina, sendo assim sugerimos o uso de atividades mais simples, sem processamento de informações, para exemplificar este conceito.

Textos personalizados: A impressão na tela de textos personalizados pode ser um bom precursor na construção de jogos. Como sugestão de atividades: 1) Receba o nome de um personagem do jogo, e exiba esse nome na tela dentro de uma frase de boas-vindas ao jogador; 2) Receba o nome do personagem, e sua cidade dentro do jogo, e exiba na tela uma frase de orientação contendo seu nome e a cidade.

Recomendações: Considerando que variáveis constituem um conceito básico em programação, elas serão utilizadas em praticamente todos os jogos desenvolvidos em sala de aula. Devido ao fato de que esse conceito é um dos primeiros a serem ministrados na disciplina, é importante que as atividades continuem sem os processamentos de informações. Por enquanto, as práticas se resumem em instruir os alunos a receberem dados do jogador e personalizar a exibição de mensagens na tela.

Conceito: Comandos condicionais.

Aplicações em jogos: Estruturas condicionais controlam o fluxo de instruções de um jogo. Sugerimos iniciar com os jogos mais simples e, posteriormente, avançar para jogos mais complexos.

Par-ou-Ímpar: É o mais simples dos jogos listados anteriormente. Neste caso, as variáveis serão utilizadas para armazenar os valores escolhidos pelos jogadores. Uma vez que os valores estão armazenados em variáveis, permite a comparação dos valores para definir o vencedor. Essa comparação é realizada pelos comandos condicionais que definirão o fluxo de instruções que será seguido pelo jogo de acordo com as regras estabelecidas pelo desenvolvedor.

Pedra-Papel-Tesoura: Também é um jogo simples de ser desenvolvido devido a simplicidade das regras. Assim como no jogo do

Par-ou-Ímpar, as variáveis são utilizadas para armazenar as escolhas dos jogadores e também o placar com a pontuação da partida.

Todos os jogos: As variáveis também podem ser utilizadas para armazenar e alterar o placar das partidas. Os comandos condicionais serão usados para definirem em que ocasião o placar será alterado.

Recomendações: Ambos os jogos mencionados são para dois jogadores. Porém, nem sempre será possível a presença de dois jogadores, especialmente durante a fase de desenvolvimento do jogo.

Números aleatórios: Conforme sugerido anteriormente, a geração de números aleatórios é um importante recurso na construção de jogos. Para evitar o entediante acontecimento do jogador se encontrar jogando Par-ou-Ímpar ou Pedra-Papel-Tesoura contra si mesmo, os números gerados aleatoriamente pelo próprio jogo proporcionarão ao jogador poder jogar contra o computador.

Conceito: Comandos de repetição.

Aplicações em jogos: Estruturas de repetição controlam a quantidade de execução de um conjunto de instruções em um jogo. Tais instruções podem conter conceitos já ministrados anteriormente, como entrada e saída de dados, variáveis e comandos condicionais. É um importante conceito no desenvolvimento de jogos, pois possibilita a implementação do laço principal do jogo. Ou seja, o funcionamento do jogo é feito por um conjunto de instruções que será executado repetidamente até que o jogador opte por sair desse laço, o que, consequentemente, encerra o jogo.

Pênaltis: Os comandos de entrada são usados para receberem as escolhas dos jogadores (posição do chute e posição do goleiro), e armazenarem nas variáveis. Dessa forma, os comandos condicionais são utilizados para comparar se as posições escolhidas são iguais (defesa do goleiro) ou diferentes (gol do chutador). Os comandos de saída são utilizados para exibir na tela o resultado das ações dos jogadores no jogo. Todas essas instruções estarão dentro de um laço repetitivo criado por um comando de repetição. Essas instruções serão repetidas até que uma condição de finalização seja satisfeita. Nesse caso, a condição de finalização poderia ser uma quantidade fixa de repetições ou rodadas.

Forca: Os comandos de entrada recebem a letra escolhida pelo jogador e a armazenam em uma variável. Então, verifica-se se a letra está presente na palavra escondida na forca, usando comandos condicionais. Em sequência, são exibidas na tela informações como: se a letra escolhida está presente na palavra escondida e em qual posição ela está; ou se a palavra escondida não contém a letra escolhida; e a quantidade de

rodadas, ou chances, restantes para o jogador acertar a palavra. Esse conjunto de instruções será repetido utilizando os comandos de repetição até que uma condição de finalização seja satisfeita.

Recomendações: Para o jogo de Pênaltis, podemos sugerir o uso de números aleatórios, conforme explicado anteriormente, para evitar o frustrante fato de um jogador precisar chutar e ele mesmo defender um pênalti. Para o jogo da Forca, sugerimos que dentro do código do jogo seja adicionado uma lista de palavras para servir como uma simples base de dados. A cada jogo, uma palavra diferente é selecionada por meio de um sorteio com números aleatórios.

Conceito: Vetores e matrizes.

Aplicações em jogos: Vetores e matrizes são variáveis que armazenam vários valores do mesmo tipo que são acessados com um mesmo nome (identificador), porém com índices diferentes. Podem ter uma dimensão (unidimensionais) ou várias dimensões (multidimensionais).

Jogo da Velha: Jogo simples jogado em uma matriz de tamanho 3x3. Comandos de entrada para receber a posição na matriz que o jogador deseja assinalar seu símbolo. Com as estruturas de repetição e condicionais, percorrer a matriz e verificar se o índice escolhido está vazio (disponível). Armazenar o símbolo do jogador ('X' ou 'O') na matriz no índice que ele escolheu. Comandos de saída para exibir na tela a situação atual da matriz 3x3 usada no jogo. Repetir as instruções até que algum jogador satisfaça alguma condição de vitória no jogo.

Cobras e Escadas: É um jogo jogado em um tabuleiro tipo grade (matriz) por dois jogadores com o uso de um dado de seis lados. Será imprescindível a implementação de uma instrução para gerar um número inteiro aleatório entre 1 e 6. Comando de entrada para receber o nome dos jogadores e armazena-los em uma variável. Lançar o dado (gerar um número aleatório entre 1 e 6) para o primeiro jogador. Mover o personagem para a casa do tabuleiro de acordo com o número gerado pelo dado. Com comandos condicionais, verificar se a casa está vazia, com cobra ou com escada, e mover o personagem de acordo com o estado da casa. Exibir na tela a situação atual do tabuleiro com os personagens em suas novas posições. Repetir as instruções, com comandos de repetição, até que a condição de finalização seja satisfeita, ou seja, até que um dos jogadores chegue na última casa do tabuleiro.

Memória: Neste tipo de jogo, as cartas estão dispostas em um formato de grade ou matriz. O uso de vetores e matrizes nessa implementação é essencial. Além disso, usamos comandos de entrada

para receber a posição da carta escolhida a ser virada em cada rodada. Os comandos condicionais são usados para comparar se as duas cartas viradas em cada rodada são iguais (ou correspondentes). Os comandos de saída são usados para exibir na tela a situação atual do jogo (cartas viradas, cartas removidas do jogo), e as estruturas de repetição são empregadas para repetir esses conjuntos de instruções a cada rodada, para cada jogador, até que todas as cartas sejam removidas do jogo.

Recomendações: Neste ponto da disciplina, já é necessário que os alunos dominem os comandos para gerar números aleatórios. Alguns jogos precisam de dados de seis lados para determinar a quantidade de passos, ou outras ações inerentes a cada jogo. Outros jogos demandam randomizar as posições iniciais de elementos, como as cartas do jogo da memória. Os conceitos de estruturas de repetição e condicionais também precisam estar claros no processo de aprendizagem dos alunos, uma vez que a navegação em vetores e matrizes é geralmente feita por meio de estruturas de repetição e a comparação/verificação de valores é comumente realizada por meio das estruturas condicionais.

Conceito: Funções e procedimentos.

Aplicações em jogos: Na maior parte dos casos, funções e procedimentos são os últimos tópicos a serem ministrados em uma disciplina inicial de programação. Também conhecida como modularização, esse conceito facilita bastante o desenvolvimento de jogos maiores e mais complexos.

Torres de Hanói: A modularização de um código neste jogo é fortemente recomendada devido a grande quantidade de movimentos repetitivos. Embora seja um jogo de raciocínio, não há muita liberdade de movimentos neste caso. A criação de uma função, ou de um procedimento, que mova um disco de uma haste para outra facilita o desenvolvimento do jogo, simplificando a sua manutenção e favorecendo uma melhor compreensão do código da aplicação.

Quizzes: Por se tratar de um jogo de múltiplas perguntas com várias alternativas, a modularização é altamente favorável neste tipo de *software*. É interessante criar um procedimento geral que tratará de todas as perguntas e suas respectivas alternativas de respostas. O emprego dos conceitos anteriores é certo. Variáveis, condicionais, repetições, serão certamente empregados no desenvolvimento de jogos do tipo quizzes. Assim como no jogo Torres de Hanói, o uso de funções e procedimentos auxilia o desenvolvimento do jogo, contribui na manutenção do código e propicia uma maior clareza das linhas de código da aplicação.

Recomendações: Em geral, este é o último conceito a ser ministrado nas disciplinas iniciais de programação. Neste ponto, o docente já estaria planejando o trabalho final de disciplina, ou seja, um projeto maior que aborde os conceitos ministrados ao longo da unidade curricular. Apresentamos aqui duas sugestões de jogos mais complexos, o que não limita as opções do docente na proposição de atividades neste ponto. É interessante que o docente não se limite a essas sugestões e procure propor o desenvolvimento de outros jogos, ou outros tipos de jogos, como trabalho final de disciplina.

Esta visualização baseada em conceitos tem como objetivo a orientação quanto à elaboração de atividades práticas de jogos, tendo como ponto de vista os tópicos abordados na disciplina de programação.

Por outro lado, provemos também uma visualização por jogos, na qual elencamos as sugestões de jogos a serem desenvolvidos no decorrer da disciplina. Nesta visualização por jogos, exibimos cada jogo sugerido acompanhado dos conceitos de programação utilizados em sua criação, e algumas oportunidades de variações dos jogos, bem como oportunidades de conexões com outras unidades curriculares.

Jogo: Par-ou-Ímpar.

Conceitos utilizados: Entrada e saída de dados. Variáveis. Estruturas condicionais. Se o jogo possuir múltiplas rodadas, pode-se utilizar os comandos de repetição.

Possibilidades de variações: Utilizando o mesmo conceito do jogo do Par-ou-Ímpar, é possível implementar um jogo de Cara-ou-Coroa, que também pode ser utilizado para decidir impasses como qual jogador inicia fará a primeira jogada, ou primeiro movimento em um jogo mais complexo.

Jogo: Pedra-Papel-Tesoura.

Conceitos utilizados: Entrada e saída de dados. Variáveis. Estruturas condicionais. Se o jogo contar com múltiplas rodadas, pode-se fazer uso dos comandos de repetição.

Possibilidades de variações: A principal variação é a adição de mais dois elementos (Lagarto e Spock) além dos três originais, conforme vimos anteriormente.

Jogo: Pênaltis.

Conceitos utilizados: Entrada e saída de dados. Variáveis. Estruturas condicionais. Estruturas de repetição. Embora a visualização

do gol aparente uma grade 3x3, ou seja, uma matriz, não há a necessidade da implementação de vetores ou matrizes neste jogo.

Possibilidades de variações e conexões: As variações podem ser no tamanho do gol, alterando de 3x3 para 3x4 ou 4x4. Quanto maior o gol, maior será a chance de vitória do chutador e isso trará um balanceamento negativo no jogo. Para tornar o jogo um pouco mais balanceado, pode-se implementar a aplicação de uma determinada porcentagem de chance do goleiro defender o chute caso ele não acerte o local exato da bola, mas que tenha escolhido uma posição adjacente. A aplicação dessa porcentagem de chance de defesa pode ser discutida nas aulas de Matemática ou Estatística.

Jogo: Forca.

Conceitos utilizados: Entrada e saída de dados. Variáveis. Estruturas condicionais. Estruturas de repetição.

Possibilidades de variações e conexões: Os temas das palavras escondidas podem ser definidos anteriormente e ter relação com outras unidades curriculares. Exemplos de temas: Objetos que estão no interior de um computador (componentes de *hardware*); Mamíferos encontrados no continente africano (Biologia, Geografia); Ex-presidentes do Brasil (História); Componentes essenciais de um sistema de gestão empresarial (Sistemas de Informação).

Jogo: Jogo da Velha.

Conceitos utilizados: Entrada e saída de dados. Variáveis. Estruturas condicionais. Estruturas de repetição. Vetores e matrizes.

Possibilidades de variações e conexões: Uma variação que pode ser implementada consiste em adicionar uma condição restritiva para assinalar o símbolo do jogador no tabuleiro. Essa condição poderia ser a seguinte: somente terá o seu símbolo assinalado no local desejado se, e somente se, responder corretamente uma pergunta. Os temas das questões podem estar relacionados com outras unidades curriculares cursadas simultaneamente pelos discentes.

Jogo: Cobras e escadas.

Conceitos utilizados: Entrada e saída de dados. Variáveis. Estruturas condicionais. Estruturas de repetição. Vetores e matrizes.

Possibilidades de variações e conexões: Umas das variações comuns neste jogo é alterar o tamanho da matriz do tabuleiro e, consequentemente, a quantidade de casas a serem percorridas para vitória. Outra variação consiste em conceder aos jogadores habilidades de se

desviarem das casas que atrasam a sua jornada. Exemplo: ao cair em uma casa com uma cobra, o jogador poderia usar uma habilidade na qual ele tem uma determinada porcentagem de chance de vencer a armadilha e não regressar no caminho já percorrido.

Jogo: Memória.

Conceitos utilizados: Entrada e saída de dados. Variáveis. Estruturas condicionais. Estruturas de repetição. Vetores e matrizes.

Possibilidade de variações e conexões: As variações são diversas e vão desde alterar a quantidade de cartas, modificar o teor e o contexto das cartas, mudar de pares iguais para pares correspondentes (ou correlacionados). O teor e o contexto podem ser alterados com o objetivo de realizar conexões com outras disciplinas. Exemplo: Jogo da memória com contexto de Física, Química, Biologia, Geografia, ou Arquitetura de Computadores, Banco de Dados, Língua Inglesa, dentre quaisquer outras unidades curriculares que os discentes possam estar cursando paralelamente.

Jogo: Torres de Hanói.

Conceitos utilizados: Entrada e saída de dados. Variáveis. Estruturas condicionais. Estruturas de repetição. Vetores e matrizes. Procedimentos e funções.

Possibilidade de variações e conexões: A quantidade de discos e hastes pode ser alterada. Em geral, o número de hastes é fixo em três e a quantidade de discos é mais frequentemente alterada para aumentar o nível de dificuldade do jogo. A conexão pode ser feita quando for ministrado o assunto de análise combinatória dentro da disciplina de Matemática ou Matemática Discreta.

Jogo: Quizzes.

Conceitos utilizados: Entrada e saída de dados. Variáveis. Estruturas condicionais. Estruturas de repetição. Vetores e matrizes. Procedimentos e funções.

Possibilidades de variações e conexões: As variações são diversas e vão desde alterar a quantidade de perguntas, modificar o teor e o contexto das questões até adicionar recursos facilitadores ao jogador. Como no jogo da memória, o teor e o contexto podem ser alterados com o objetivo de realizar conexões com outras disciplinas. Exemplo: Quizzes com perguntas em um contexto de Física, Química, Biologia, Geografia, ou Arquitetura de Computadores, Banco de Dados, Língua Inglesa, dentre

quaisquer outras unidades curriculares que os discentes possam estar cursando paralelamente.

Esta visualização baseada em jogos tem como intuito oferecer um outro ponto de vista quanto à elaboração de atividades práticas de jogos. Desta vez, a orientação tem as sugestões de jogos como ponto de vista. Sendo assim, o docente tem a disposição um modelo com dois pontos de vista distintos, sobre como elaborar as atividades práticas de programação. É importante entender que apresentamos um modelo e que este pode ser adaptado conforme a realidade do docente e da instituição onde ele leciona.

2.7 FASE 3 – ENSINO E APRENDIZAGEM

A terceira fase refere-se ao período do ensino e aprendizagem propriamente ditos. Uma vez que os professores tenham concluído a segunda fase, as aulas, lições e atividades planejadas já podem ser colocadas em prática. Sugerimos que, antes da realização de atividades práticas de criação de jogos, o docente apresente e esclareça a definição de jogo, e a estrutura básica de um jogo do ponto de vista técnico. É importante que o docente use os exemplos de criação de jogos para cada tópico abordado. Como ressaltado anteriormente, é primordial que o docente encoraje seus alunos a participarem ativamente da disciplina, de modo a fornecer um suporte a construção de seus conhecimentos, conforme ressalta Zorzo et al. (2017). Como forma de evitar interrupções no processo de aprendizagem, sugerimos que – quando possível – as atividades sejam auto-suficientes, ou seja, que não haja dependências de atividades anteriores. Depois de terem sido ministradas as aulas e aplicadas as atividades, recomendamos a aplicação de uma avaliação de aprendizagem. Isso é importante para permitir que os professores compreendam como os alunos estão progredindo e se estão alcançando seus objetivos de aprendizagem.

2.7.1 Aulas e tutoriais

Durante a ministração das aulas, exposição e discussão dos conteúdos programáticos, o docente tem a responsabilidade de cumprir as políticas institucionais, bem como os objetivos da disciplina conforme visto na fase de Auditoria Preliminar. Em cada tópico abordado, o uso de exemplos inerentes ao desenvolvimento de jogos também é encorajado. Sendo assim, o docente estabelece uma conexão inicial com os alunos preparando-os para as atividades práticas de desenvolvimento de jogos.

Como parte dos esclarecimentos iniciais de uma disciplina, consideramos importante que o professor apresente alguns conceitos básicos inerentes aos jogos, bem como seu processo de desenvolvimento, antes mesmo de propor as atividades práticas de criação de jogos. Sendo assim, sugerimos que nas primeiras aulas o docente apresente o que é um jogo e quais são seus elementos essenciais, alinhando com os conceitos básicos da estrutura de desenvolvimento de um *software* do tipo jogo.

O que é um jogo? Quais são os elementos essenciais de um jogo? O que um jogo precisa ter para ser considerado um jogo? Vários pesquisadores (HUIZINGA; HULL, 1949; CAILLOIS, 1961; SUITS, 2014; CRAWFORD, 1984; AVEDON; SUTTON-SMITH, 1971; JUUL, 2011) já tentaram responder essas questões e as suas respostas são diversificadas entre si, ou seja, não há um consenso. Diante de diferentes definições de jogo, Salen e Zimmerman (2003) analisaram as contribuições de diversos autores para este tema, na tentativa de encontrar elementos em comuns nessas variadas definições disponíveis. Após realizar uma análise comparativa entre oito autores diferentes, Salen e Zimmerman (2003) harmonizaram os elementos das definições pesquisadas, reduziram as partes que eles julgaram desnecessárias, formularam a seguinte definição: "Um jogo é um sistema no qual os jogadores se envolvem em um conflito artificial, definido por regras, que resulta em um resultado quantificável" (SALEN; ZIMMERMAN, 2003, p. 50). Para facilitar o entendimento, Salen e Zimmerman (2003) ainda detalham cada um dos elementos essenciais de sua definição de jogo, conforme Quadro 8 a seguir.

Quadro 8: Detalhamento dos elementos essenciais de um jogo

Elemento	**Descrição**
Sistema	Um sistema é um conjunto de partes que se inter-relacionam para formar um todo complexo.
Jogadores	Um jogo é algo que um ou mais participantes jogam ativamente. Os jogadores interagem com o sistema de um jogo para experimentar o jogabilidade do jogo.
Artificialidade	Os jogos mantêm um limite da chamada "vida real" tanto no tempo quanto no espaço. Embora os jogos, obviamente, ocorram no mundo real, a artificialidade é uma das suas características definidoras.
Conflito	Todos os jogos incorporam uma disputa de poderes. Essa disputa pode assumir várias formas, desde a cooperação até a competição, desde conflito solo até conflitos sociais multijogadores. O conflito é essencial para os jogos.

Elemento	Descrição
Regras	As regras são uma parte crucial dos jogos, uma vez que fornecem a estrutura da qual o jogo emerge, delimitando o que o jogador pode e não pode fazer.
Resultado quantificável	Os jogos têm um objetivo ou resultado quantificável. Ao fim de um jogo, um jogador ganhou ou perdeu ou recebeu algum tipo de pontuação numérica. Um resultado quantificável é o que geralmente distingue um jogo de atividades lúdica menos formal (por exemplo: uma brincadeira).

Fonte: Salen e Zimmerman (2003, p. 50). Tradução nossa.

A definição de jogo e seus elementos essenciais (Quadro 8), de Salen e Zimmerman (2003), são importantes esclarecimentos para que tanto professores quanto alunos tenham em mente o que é um jogo e quais são seus componentes cruciais. Além disso, é importante levar em consideração o assunto deste guia, ou seja, desenvolvimento de jogos. E para isso, é essencial que o professor apresente e esclareça a estrutura básica de um jogo do ponto de vista técnico, ou seja, do seu desenvolvimento algorítmico.

Do ponto de vista da programação, qual é a estrutura de jogo? Como alinhar os elementos essenciais de um jogo com a estrutura algorítmica de um *software*? As regras de jogo definem uma lista de ações que os jogadores estão – ou não – autorizados a fazer. De acordo com Sicart (2008), essas regras são avaliadas por um ciclo de jogo (do inglês, *game loop*), que é um algoritmo que relaciona o estado atual do jogo e as propriedades dos objetos com uma série de condições que consequentemente podem modificar o estado do jogo. Por exemplo, a condição para vencer, a condição para perder e os efeitos de ação na saúde do personagem do jogador são calculados ao executar o ciclo do jogo. Este algoritmo relaciona as regras com a mecânica, alinhando os elementos essenciais de um jogo com os aspectos técnicos inerentes a programação.

Do ponto de vista da programação, o componente central de qualquer jogo é este ciclo do jogo. Este ciclo permite que o jogo funcione sem problemas, independentemente se o usuário fornece algum comando de entrada, ou não. A maioria dos programas de *software* tradicionais responde à entrada do usuário e não faz nada sem ele. Por exemplo, um processador de texto forma palavras e textos conforme a digitação do usuário no teclado. Se o usuário não digitar nada, o processador de texto não fará nada. Algumas operações podem demorar muito para ser

concluídas, mas todas são iniciadas por um usuário que ordena o programa a fazer algo.

Os jogos digitais, por outro lado, devem continuar a operar independentemente da entrada de um usuário. O ciclo do jogo permite isso. Como exemplo, um ciclo de jogo altamente simplificado, em pseudocódigo, pode parecer com o conteúdo do Quadro 9:

Quadro 9: Pseudocódigo simplificado de um ciclo de jogo

```
enquanto (o jogo está rodando)
   apresentar um cenário ou situação
   receber/aceitar a entrada do usuário
   executar inteligência artificial
   mover inimigos
   checar colisões
   desenhar gráficos
   tocar sons
termine enquanto
```

Fonte: Próprio autor.

As especificidades desse ciclo dependem do jogo, isto é, o ciclo pode ser refinado e modificado à medida que o desenvolvimento do jogo progride, mas a maioria dos jogos são baseados nesta estrutura básica. Podemos dizer que a estrutura rudimentar de um jogo digital consiste em: Apresentar, Aceitar, Interpretar, Calcular, Repetir.

O escopo de cada jogo digital é **apresentar** ao usuário uma situação, **receber**/**aceitar** sua entrada, **interpretar** esses sinais em ações, **calcular** uma nova situação resultante desses atos, **repetir** até que uma condição de encerramento seja satisfeita. Este escopo geral pode ser exemplificado pelo diagrama presente na Figura 5. Os jogos estão constantemente percorrendo esses estágios, diversas vezes, até ocorrer alguma condição final (por exemplo: ganhar, perder ou sair do jogo).

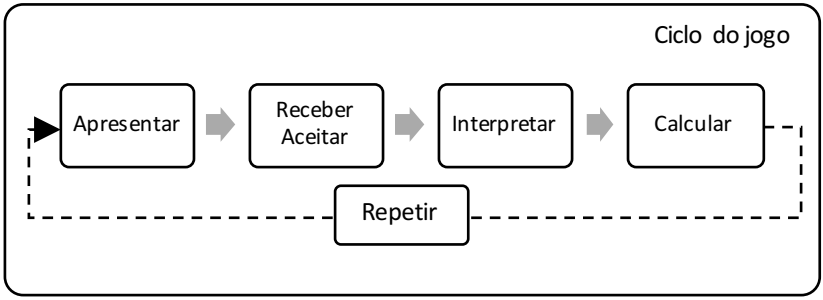

Figura 5: Estrutura de um ciclo de jogo
Fonte: Próprio autor.

Alguns jogos conduzem este ciclo conforme as entradas do usuário. Imagine que você está desenvolvendo um "jogo da memória". Esses jogos **apresentam** um conjunto de cartas (todas posicionadas com a face para baixo) para o usuário; eles **recebem/aceitam** seus cliques (ou toques); eles **interpretam** a entrada como um sucesso, falha, pausa, interação de menu, etc.; finalmente, eles **calculam** um cenário atualizada resultante dessa entrada. O ciclo do jogo avança conforme a entrada do usuário e aguarda até que essa entrada seja fornecida.

Com esses esclarecimentos iniciais sobre a definição de jogo, seus elementos principais, e as suas relações com a estrutura algorítmica d e um software, o docente estabelecerá um alicerce, uma base, para as atividades de ensino de programação baseado em desenvolvimento de jogos. De acordo com os Referenciais de Formação em Computação da SBC (ZORZO et al., 2017, p 37), a metodologia de ensino deve ser centrada no aluno como sujeito da aprendizagem e apoiada no professor como facilitador do processo de ensino-aprendizagem. Além disto, buscando promover a explicitação das relações entre os conteúdos abordados e as competências previstas para o egresso dos cursos.

Conforme mencionado na seção 2.5.1, é interessante que o docente tenha conhecimento sobre exigências das políticas e normas institucionais a respeito do uso de uma determinada linguagem de programação durante a ministração da disciplina. Para as aulas iniciais da disciplina, durante a ministração de conceitos introdutórios, é importante que os alunos compreendam os comandos básicos que a linguagem de programação escolhida oferece. Nessas primeiras aulas, a criação de pequenos programas é feita por meio da inserção de códigos em texto, e o retorno, ou seja, o programa/*software* resultante desse código é também exibido em modo texto conforme a Figura 6.

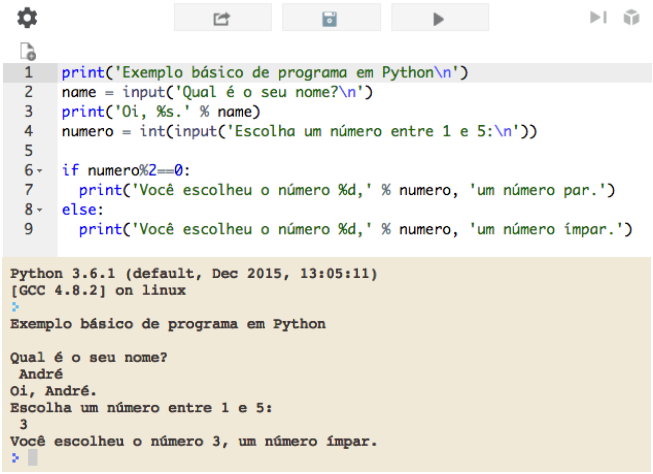

Figura 6: Programa básico em Python
Fonte: Próprio autor.

Este é o modo mais simples criar um programa e de visualizar rapidamente o resultado de um código. Os primeiros jogos produzidos pelos alunos terão uma aparência semelhante ao jogo da Memória exibido na Figura 7, com aspecto monocromático, somente texto, sem imagens, sem sons. É importante que esse primeiro contato dos alunos com a linguagem de programação forneça uma base sólida de conhecimentos sobre os principais comandos e funções disponíveis por padrão na linguagem estudada, o que permitirá, ocasionalmente, a inserção de outros elementos adicionais e ferramentas aliadas desta linguagem.

```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
>
Jogo da Memoria em Python

Acertos: 2

   1  2  3  4  5

1  x  x  x  x  A

2  x  A  x  x  x

3  x  x  x  B  x

4  x  x  x  x  B

5  x  x  x  x  x

Digite uma linha e coluna: 3 , 4
>
```

Figura 7: Jogo da Memória em Python (modo texto)
Fonte: Próprio autor.

Muito útil no início da disciplina, o aspecto monocromático aliado à falta de elementos gráficos e sonoros pode, eventualmente, tornar enfadonhas as aulas e atividades de construção de jogos. Sugerimos usar esse modo texto nas primeiras aulas e contatos iniciais dos alunos com a linguagem de programação estudada, ou enquanto houver necessidade. Caso o docente perceba que os discentes já detêm certo domínio das funções e dos comandos básicos da linguagem, e deseja aprimorar os jogos que estão produzidos pelos alunos, há a opção de trazer para a disciplina algumas ferramentas e elementos adicionais. Uma dessas ferramentas é, na verdade, um conjunto de elementos, ou recursos, também conhecido como biblioteca de funções. Trata-se de um conjunto de códigos e funções prontas para a linguagem de programação estudada que tem como objetivo facilitar a inserção/manipulação de elementos gráficos e sonoros nos jogos produzidos pelos alunos. Com a utilização dessas bibliotecas, os jogos produzidos pelos alunos são transformados em artefatos mais interessantes como exemplificado na Figura 8.

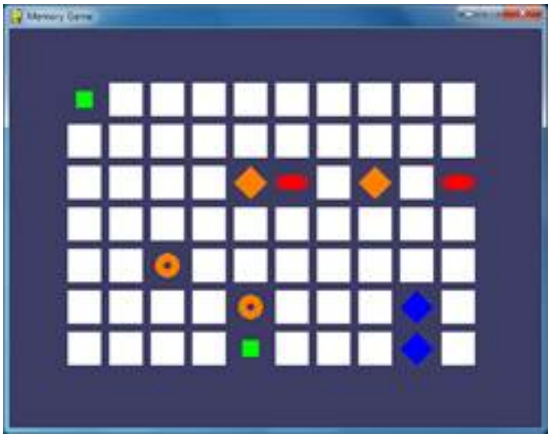

Figura 8: Jogo da Memória em Python (modo gráfico)
Fonte: Sweigart (2012, p. 33).

Existe um grande número de bibliotecas de funções de programação para os mais diversos fins, que vão desde 'Computação Gráfica', 'Bioinformática', 'Aprendizado de máquina', 'Jogos' até 'Redes de comunicações', dentre outras. São muitas as bibliotecas direcionadas para a programação de jogos, cada uma com suas próprias características e peculiaridades. Como sugestão apresentamos a seguir uma relação de bibliotecas, também conhecidas como *frameworks*, que podem ser utilizadas pelo docente em suas aulas e atividades práticas de programação de jogos. Restringimos aqui apenas as bibliotecas gratuitas, ou seja, que não tem custo algum em sua utilização.

Biblioteca
SDL: *Simple DirectMedia Layer*
Descrição
É uma biblioteca multi-plataforma destinada a prover acesso de baixo nível ao áudio, teclado, *mouse*, *joystick* e gráficos por meio de comandos e funções prontas. É largamente utilizada na produção de jogos comerciais e não-comerciais.
Linguagens de programação
Originalmente, a biblioteca SDL suporta as linguagens C/C++, porém é possível fazer ajustes e adaptações (*bindings*) que possibilitam a utilização dessa biblioteca em outras linguagens como Python, C# e Pascal.
Licença de uso
SDL é distribuída sob a licença *zlib* que permite que a biblioteca seja utilizada livremente em quaisquer *softwares*.

Plataformas e sistemas alvos
SDL pode ser utilizada para desenvolvimento de jogos para os sistemas *desktop Windows*, *Mac OS X*, *Linux*, e sistemas móveis *iOS*, e *Android*.
Site (*download* e maiores informações)
https://www.libsdl.org/

Biblioteca
Monogame
Descrição
Sucessora de outra biblioteca, a XNA descontinuada em 2013, a Monogame utiliza linguagem C# para desenvolvimento de jogos com a grande vantagem da possibilidade de criar jogos para plataformas de videogames.
Linguagens de programação
C#
Licença de uso
Está licenciada sob a Ms-PL (*Microsoft Public License*) que é a menos restritiva das licenças da *Microsoft* e permite a distribuição de *softwares* para fins comerciais ou não-comerciais sob qualquer licença que esteja em conformidade com a Ms-PL.
Plataformas e sistemas alvos
Pode ser utilizada para desenvolvimento de jogos para os sistemas *desktop Windows*, *Mac OS X*, *Linux* e sistemas móveis *iOS* e *Android*. Além disso, também é possível desenvolver jogos para videogames como *PS4* (*Sony*), *PSVita* (*Sony*), *Xbox One* (*Microsoft*), e *Switch* (*Nintendo*).
Site (*download* e maiores informações)
http://www.monogame.net/

Biblioteca
LibGDX
Descrição
Libgdx é uma biblioteca de desenvolvimento de jogos em Java que fornece um conjunto de funções unificadas que podem ser utilizadas em diversas plataformas e sistemas
Linguagens de programação
Java
Licença de uso
Está licenciada sob a *Apache Licence* que permite ao usuário da biblioteca a liberdade de usá-la para qualquer propósito, distribuir versões modificadas do produto.
Plataformas e sistemas alvos

Pode ser utilizada para criação de jogos em sistemas *desktop Windows*, *Mac OS X*, *Linux* e sistemas móveis *iOS*, *BlackBerry* e *Android*.

Site (*download* e maiores informações)

https://libgdx.badlogicgames.com/

Biblioteca

Cocos2d-x

Descrição

O cocos2d-x é uma biblioteca para a construção de jogos em duas dimensões (2D) e outras aplicações gráficas interativas. Permite também manipulação de áudio, edições de cenas e publicação do jogo em múltiplas plataformas.

Linguagens de programação

Esta versão Cocos2d-x permite programação em C++, Lua, Javascript. Porém existe a versão Cocos2d (sem a letra 'x') na qual a programação é feita em Python ou Objective-C.

Licença de uso

Está licenciada sob a licença MIT (*Massachusetts Institute of Technology*) que permite a reutilização dentro do *software* proprietário, desde que todas as cópias do *software* licenciado incluam uma cópia dos termos da licença MIT e o aviso de direitos autorais.

Plataformas e sistemas alvos

Esta biblioteca tem como plataforma e sistemas alvos *iOS*, *Android*, *Windows 8*, *Windows Phone 8*, *Linux*, *Mac OS X*.

Site (*download* e maiores informações)

http://www.cocos2d-x.org/

Biblioteca

Corona SDK

Descrição

É uma biblioteca gratuita e multi-plataforma direcionada para criação de jogos e aplicativos para dispositivos móveis e *desktop*. Possui uma vasta seleção de ferramentas e extensões adicionais, além de todo um ecossistema de aplicativos voltados para facilitar a gestão do desenvolvimento de jogos e aplicações.

Linguagens de programação

Utiliza linguagem Lua em sua versão nativa, porém pode-se estender para Java, C/C++, Objective-C, por meio de uma ferramenta adicional fornecida pelo próprio desenvolvedor.

Licença de uso

Não informado.

Plataformas e sistemas alvos
Esta biblioteca permite desenvolver jogos e aplicações para dispositivos móveis com sistemas *iOS*, *Android*, *Amazon Kindle*, além dos desktops *Windows* e *macOS*, das plataformas de TV *Apple TV* e *Android TV* e da distribuidora de jogos *Steam*.
Site (*download* e maiores informações)
https://coronalabs.com/

A utilização dessas bibliotecas é opcional, e o seu uso somente agrega valor aos artefatos produzidos pelos alunos, evitando que fiquem entediados com o contínuo resultado estático e monocromático de jogos em modo texto. Além disso, consideramos que seja uma oportunidade para que os alunos tenham um primeiro contato com a utilização de bibliotecas de programação, habilidade que certamente será um diferencial em seu currículo profissional.

Vale ressaltar que essas bibliotecas não são ambientes visuais de programação em blocos, conforme vimos na seção 1.6. Diferente da programação em blocos, fazem uso da programação convencional em modo texto, porém o resultado final, o artefato jogo produzido, terá um aspecto visual melhor e mais atrativo (Figura 8), o que pode motivar ainda mais os alunos.

2.7.2 Atividades práticas de programação

Após a apresentação de um determinado tópico de programação, o docente pode propor a realização de atividades práticas de construção de jogos. A fase anterior, Planejamento e Projetos, é parte essencial neste guia para evitar improvisos e imprevistos no momento das aulas e práticas de programação. Salientamos que os jogos a serem propostos para cada tópico são selecionados previamente durante a fase de Planejamento e Projetos. É essencial que o docente se prepare no sentido de entender os jogos propostos e compreender as suas regras, além de ter uma clara noção de como implementá-los na linguagem de programação utilizada na unidade curricular. Caso o jogo selecionado para um determinado tópico não seja tão conhecido pelos alunos, sugerimos que o docente jogue o jogo em conjunto com os discentes antes do início da atividade prática. Por exemplo, se alguns alunos não conhecerem o jogo Torres de Hanói, é possível jogá-lo virtualmente em alguns *sites* de jogos para familiarização. Além disso, o docente pode, quando possível, apresentar o jogo fisicamente para os alunos. Isso é possível de ser feito com praticamente todos os jogos sugeridos anteriormente (par-ou-ímpar, pedra-papel-tesoura, forca, memória, torres de Hanói, quizzes, cobras e

escadas, jogo da velha). É importante lembrar que os jogos listados na fase anterior são apenas sugestões, ou seja, um guia para o docente elaborar suas atividades práticas de construção de jogos.

Estas atividades práticas não são necessariamente avaliativas no sentido de compor a nota final do aluno na disciplina. Mas é importante que o docente utilize essas atividades como forma de analisar o progresso dos discentes durante o processo de ensino/aprendizagem, de modo a verificar se os alunos estão compreendendo e assimilando os conceitos de programação ensinados.

2.7.3 Avaliação de aprendizagem

As avaliações de aprendizagem que compõem a nota final dos discentes na disciplina, em geral são definidas por documentos institucionais, seja ementa, projeto político pedagógico, ou regimento interno. Normalmente, essas atividades avaliativas consistem de trabalhos práticos e provas.

Seymour Papert (PAPERT, 1991), assim como Jean Piaget (1972), via a aprendizagem como a criação de estruturas de conhecimento, porém Papert (1991) acrescentou um aspecto importante a este pensamento: os artefatos da mente têm de se deslocar para um ambiente público, onde podem ser examinados, compartilhados e avaliados por outros. De muitas maneiras, essa visão já antevia as vastas redes distribuídas de objetos digitais e comunidades *online* que compõem a Internet atualmente. Fundamentalmente, isto reflete a base para fazer, pensar e interagir, ou seja, aprender com quaisquer meios ou mídias.

Essa concepção de Papert (1991), de tornar público os artefatos produzidos pelos alunos, aproxima-se visivelmente ao conceito de portfólios. Basicamente, portfólios são uma reunião de serviços e produções criativas, artísticos ou culturais, criados por um profissional ao longo de sua carreira, com o objetivo de retratar suas habilidades e competências em uma determinada área do conhecimento.

Cabe registrar que muitos educadores já fazem uso de portfólios como instrumentos apoiadores no processo de ensino e aprendizagem. Nascimento, Nery e Silva (2013), e Pontes e Castanho (2015), recomendam a manutenção de portfólios pelos alunos para reunirem os jogos produzidos por eles em disciplinas de cursos superiores (PONTES; CASTANHO, 2015) e em iniciativas governamentais de assistência e formação de jovens carentes (NASCIMENTO; NERY; SILVA, 2013).

Em outra direção, Santos e Freitas (2015) fazem uso de uma metodologia baseada em aprendizagem colaborativa no ensino de ciências físicas e biológicas em disciplinas do ensino médio. Os autores

fizeram uso de abordagens que foram desde produções de textos, criação de jogos, até elaboração de relatórios. O aspecto principal do estudo de Santos e Freitas (2015) foi a utilização de portfólios para reunir e manter os artefatos produzidos pelos alunos durante a disciplina.

Retomando as recomendações acerca do nosso guia, no que se refere aos trabalhos práticos, sugerimos o uso de uma avaliação baseada em portfólios pelo fato de harmonizarem bem com as ideias construcionistas. Hazzan, Lapidot e Ragonis (2015) destacam que a avaliação baseada em portfólio pode ser empregada nas disciplinas dos cursos de TI como forma de registrar as produções desenvolvidas pelos discentes, sejam elas jogos ou *softwares* de qualquer categoria.

Hazzan, Lapidot e Ragonis (2015) definem um portfólio como um conjunto de produções do aluno que conta a história sobre seus esforços, progressos, realizações e autorreflexão sobre seu processo de aprendizagem, em uma ou mais áreas de conhecimento. Diante disto, o portfólio pode ser um recurso pedagógico que:

- Integra a aprendizagem com a avaliação;
- Cria um canal contínuo de comunicação e colaboração entre professores e alunos;
- Fornece uma visão abrangente das conquistas dos alunos em relação a uma variedade de conceitos;
- Permite que os alunos identifiquem seus pontos fracos e fortes;
- Encoraja os alunos a assumirem a responsabilidade em seus processos de aprendizagem;
- Melhora as habilidades reflexivas dos alunos.

No caso da educação em TI, é relevante criar um portfólio virtual, ou seja, *online* o qual pode incluir:

- Projetos individuais e em grupo de alunos;
- Versões anteriores destes projetos;
- Descrições dos processos de desenvolvimento destes projetos;
- Avaliações por colegas de turma;
- Apresentação dos alunos de suas produções;
- Observações dos professores sobre o processo de aprendizagem.

O processo de confecção de um portfólio geralmente requer a síntese de ideias, reflexão sobre realizações, autoconsciência e planejamento, o que trará potencialmente benefícios educacionais inerentes à jornada de aprendizado do aluno. Um elemento-chave da avaliação baseada em portfólios é a motivação para o registro destas

produções discentes, levando-os a compreender que os portfólios não são apenas "arquivos e pastas" de toda a produção discente, mas sim um modo de registrar um histórico e uma forma de dar sentido e comunicar sobre a sua aprendizagem.

Em relação às provas, estas podem ser testes escritos ou testes práticos realizados em um laboratório de informática. Sugerimos que, sempre que possível, o professor opte pelos testes práticos como forma de avaliação, uma vez que a disciplina de programação é uma unidade curricular com alto teor prático, as provas práticas certamente avaliarão melhor o aprendizado dos alunos nestas circunstâncias. Elencamos a seguir sugestões de tipos de questões que podem ser utilizadas como componentes de uma prova de programação:

1. Desenvolvimento de um novo recurso para um jogo;
2. Desenvolvimento de uma solução utilizando um comando pré-determinado;
3. Análise de execução de um código e seu resultado;
4. Correção de erros em códigos;
5. Conclusão de códigos não-finalizados;
6. Elaboração de questões;
7. Modificação de um código de um jogo;
8. Perguntas fechadas (múltipla escolha).

Os tipos de questões listados acima são sugestões para o docente no momento de confecção das provas. Não há a necessidade de empregar todos estes componentes em uma única prova. Os tipos de questões utilizadas nos testes não podem se limitar às sugestões fornecidas. Contudo, é importante que docente se municie de outras fontes para elaboração de suas atividades.

Uma vez que diferentes professores podem enfatizar diferentes aspectos da educação em TI, não especificamos um esquema de avaliação específico para a disciplina de programação. Em vez disso, sugerimos uma lista de componentes que podem ser analisados como parte da avaliação dos alunos. Outros aspectos, bem como diferentes pesos (notas) atribuídos a cada componente, e diferentes mecanismos de avaliação (pelos próprios alunos, ou pelo professor da disciplina) são opcionais para a avaliação dos discentes. Entretanto, em cada caso recomendamos explicar aos alunos as formas de avaliação e divulgá-las previamente.

2.8 FASE 4 – REFLEXÕES E *FEEDBACKS*

A quarta fase consiste na análise do emprego da metodologia. Nesta fase, os *feedbacks*, tanto do professor quanto dos alunos, são recolhidos e analisados. Todo o processo de implementação é revisado

para melhorarias na implementação futura. Essa fase final do modelo visa rever a implementação, pois examina como os professores se sentiram em relação às aulas ministradas, e como os alunos as receberam. Nesta fase de reflexões e *feedbacks*, sugerimos que os professores coletem dados usando pesquisas ou conversas informais com os discentes.

2.8.1 *Feedback* do docente

O *feedback* exercido pelo próprio docente visa, em geral, obter dados e informações sobre as Fases 2 – Planejamento e Projetos, e 3 – Ensino e Aprendizagem, do guia. Funcionará como uma autorreflexão do docente sobre os processos de planejamento e ensino da disciplina. Para obter informações sobre o andamento e conclusão de tais fases, o docente pode se equipar de um questionário com questões sobre efetivo engajamento dos alunos, sobre as dificuldades e facilidades encontradas ao longo da disciplina, dentre outras.

Como exemplo, apresentamos uma lista de sugestões de questões para compor esse questionário de *feedback* do docente.

1. Houve um nítido aumento de engajamento dos alunos?
2. O uso da metodologia de construção de jogos despertou interesse de todos os alunos? Se não, qual a porcentagem de interessados e não-interessados?
3. Quais foram os principais obstáculos durante a disciplina?
4. Se pudesse repetir a metodologia, o que faria diferente?
5. Faltou algum recurso pedagógico ou tecnológico no transcorrer da disciplina?
6. Quais foram os conceitos mais difíceis de exemplificar com criação de jogos? Por quê?
7. É possível usar construção de jogos para exemplificar todos os conceitos determinados na ementa da disciplina? Se não, qual conceito não foi possível?
8. O material elaborado foi suficiente para atender as demandas dos alunos?
9. A ementa, ou o conteúdo programático, da disciplina atendeu às expectativas de aprendizagem dos alunos?
10. A seleção dos jogos para atividades práticas foi bem-sucedida?
11. Os alunos forneceram opiniões acerca da escolha dos jogos? Os alunos desenvolveram suas próprias variações dos jogos escolhidos?

As questões apresentadas são exemplos para o docente no momento da sua autoavaliação sobre o emprego da metodologia de construção de jogos. As respostas das questões servirão para identificação de pontos fracos e pontos fortes, tanto na fase de planejamento (Fase 2) quanto na fase de execução (Fase 3). Identificar fragilidades e pontos positivos possibilitará ao docente remodelar tais aspectos de modo a evitar possíveis equívocos na próxima oportunidade de ministrar a disciplina.

2.8.2 *Feedbacks* dos alunos

O *feedback* provido pelos alunos visa obter dados e informações sobre a Fase 3 – Ensino e Aprendizagem – do guia. Há duas formas de coletar as impressões dos alunos sobre a disciplina. A primeira consiste em aplicar questionários impressos em papel em sala de aula, e a segunda refere-se à aplicação de questionários virtuais que serão respondidos pelos alunos por meio de uma ferramenta *online*.

Com questionários impressos é possível coletar anonimamente os *feedbacks* dos alunos durante uma aula gastando cerca de quinze minutos ou pouco mais. Um questionário de opiniões pode conter uma combinação de perguntas abertas e quantitativas (perguntas com escalas de valores). O docente pode elaborar suas próprias perguntas ou usar as sugestões de questões elencadas a seguir.

Os questionários impressos aplicados em sala de aula têm a vantagem de uma alta taxa de resposta. Uma desvantagem é que o docente pode reconhecer a caligrafia de seus alunos e, por isso, suas respostas podem não ser tão anônimas como poderiam ser.

Outra maneira de coletar *feedback* dos discentes é fazer com que eles respondam anonimamente uma pesquisa *online* sobre a disciplina. Os tipos de questões são os mesmos utilizados em um questionário impresso, por isso, as sugestões de questões disponíveis a seguir podem ser utilizadas também neste tipo de questionário.

Considerando que os alunos digitam suas respostas, ao invés de escrever à mão, as pesquisas *online* têm a vantagem de preservar o anonimato dos alunos. No entanto, as pesquisas *online* geralmente têm taxas de respostas mais baixas do que pesquisas em sala de aula.

Como forma de sugestões, apresentamos alguns de tipos de questões para coleta de *feedbacks* dos alunos:

1. As aulas da disciplina foram interessantes (1 = Discordo totalmente; 5 = Concordo totalmente):
    a. 1(__). 2(__). 3(__). 4(__). 5(__).
2. As atividades práticas da disciplina são desafiadoras (1 = Discordo totalmente; 5 = Concordo totalmente):

   a. 1(__). 2(__). 3(__). 4(__). 5(__).
3. A quantidade de trabalhos e atividades práticas é justa (1 = Discordo totalmente; 5 = Concordo totalmente):
   a. 1(__). 2(__). 3(__). 4(__). 5(__).
4. O professor tem conhecimento do conteúdo de programação (1 = Discordo totalmente; 5 = Concordo totalmente):
   a. 1(__). 2(__). 3(__). 4(__). 5(__).
5. Em relação à qualidade do material apresentado/proposto pelo professor, você está: (1 = Muito insatisfeito; 5 = Muito satisfeito):
   a. 1(__). 2(__). 3(__). 4(__). 5(__).
6. O quão satisfeito você está com o seu aprendizado durante a disciplina (1 = Muito insatisfeito; 5 = Muito satisfeito):
   a. 1(__). 2(__). 3(__). 4(__). 5(__).
7. Considerando o método de ensino do professor, você considera que ele poderia usar mais tempo para:_____; e menos tempo para: _____.
8. Em relação às atividades práticas propostas pelo professor, você:
   a. Fez todas as atividades dentro do prazo;
   b. Fez a maioria das atividades dentro do prazo;
   c. Fez poucas atividades dentro do prazo;
   d. Fez nenhuma atividade dentro do prazo.
9. Dê uma razão à, ou justifique, sua resposta da questão anterior.
10. O que mais te agradou durante a disciplina? Por quê?
11. O que mais te desagradou durante a disciplina? Por quê?
12. O professor se mostrou disponível mesmo no período fora de aula?
13. Sinta-se livre para elaborar quaisquer comentários acerca da disciplina e/ou da conduta do professor.

As questões aqui expostas estão listadas como sugestões. É importante que o *feedback*, impresso ou *online*, dos alunos seja anônimo de modo a garantir o máximo de honestidade nas respostas.

2.8.3 *Feedback* da instituição

O *feedback* institucional consiste em averiguar se os documentos institucionais estão de acordo com as atuais necessidades de aprendizagem dos alunos. Os elementos que foram analisados durante a

Fase 1 – Auditoria Preliminar –, podem ser revistos neste momento de modo a fornecer à instituição um quadro geral das condições atuais dos recursos, e documentos institucionais, inerentes a disciplina ministrada.

1. A infraestrutura disponível é adequada para atender à demanda da disciplina e dos alunos?
2. A instituição possui laboratórios de informática adequados?
3. O número de computadores disponíveis por aluno é suficiente?
4. Os recursos computacionais disponíveis (ferramentas de programação) são adequados para atender à demanda de ensino da disciplina?
5. O projeto político pedagógico está atualizado para atender as principais exigências do mercado de TI brasileiro?
6. O conteúdo programático determinado nas ementas de programação está atualizado? As recentes tecnologias estão presentes na ementa? As tecnologias já ultrapassadas, ou que entraram em desuso, ainda continuam presentes na ementa?
7. Os recursos disponíveis virtualmente, ferramentas *online*, ainda estão disponíveis e acessíveis pelos alunos?

As respostas para tais questões podem ser utilizadas para uma possível readequação de normas e políticas institucionais que visam constantemente manter o nível de qualidade elevado do curso.

Em suma, todas as questões apresentadas são apenas sugestões para o docente ter como guia no momento da elaboração dos seus próprios questionários de *feedbacks*. É essencial que o professor elabore seus questionários (impressos ou *online*) que, de forma geral, respondam às seguintes indagações:

1. Os alunos estão satisfeitos com as aulas ministradas?
2. Quais mudanças o docente poderia realizar que seriam mais úteis aos alunos?
3. Como o docente pode melhor atender às necessidades dos alunos?
4. A instituição atende aos alunos adequadamente?

Com isso, o docente terá uma visão geral de como adaptar, renovar, remodelar seu plano de ação para implementação da metodologia baseada em construção de jogos em uma próxima oportunidade.

Baseado na análise das experiências coletadas, os professores podem melhorar a estrutura do próximo processo de implementação. Para

uma compreensão geral da implementação do guia, é importante realizar a coleta de dados em todos os três estágios anteriores, não apenas a etapa de Ensino e Aprendizagem.

2.9 VISÃO GERAL DO GUIA PROPOSTO

Como mencionado, este guia pode ser usado como um instrumento de orientação para implementação de uma metodologia baseada na construção de jogos digitais em disciplinas iniciais de programação. Este guia pode ser útil para o ensino introdutório de programação em várias estruturas, organizações e níveis de educação em TI (universidades, institutos federais, cursos independentes). Para isso, este guia apresenta alguns exemplos de atividades que abordam o aspecto pedagógico desde a preparação da disciplina até o momento de seu encerramento.

As atividades de ensino são apresentadas em detalhes, incluindo orientações e recomendações, de modo a permitir a sua aplicação imediata. A fim de tornar o material aplicável a todos os níveis de ensino de informática, não indicamos explicitamente a duração de cada atividade. Sugerimos que cada professor, ou instrutor, determine o tempo necessário para cada atividade de acordo com as características de seus alunos e de sua instituição. Como forma de apresentação geral, o Quadro 10 expõe um sucinto resumo dos pontos a considerar durante a implementação do guia proposto.

Quadro 10: Pontos a considerar durante a implementação

1) Auditoria Preliminar	
Componente	**Pontos a considerar**
Políticas e normas institucionais	As políticas e normas serão verificadas para evitar violações relacionadas a mudanças significativas nos cursos.
Objetivos da disciplina	Os objetivos da disciplina serão analisados de modo a preservar a integridade acadêmica do curso.
Infraestrutura disponível	As ferramentas e recursos a serem utilizados devem ser simples, de fácil manuseio e estarem disponíveis.
Padrões de codificação e estilo	Um manual para melhores práticas e padrões de codificação pode ser definido neste momento.
2) Planejamento e Projetos	
Componente	**Pontos a considerar**
Planos de aula	São necessários planos de aula detalhados, considerando contexto, pedagogia e neutralidade do material.

Escolha dos jogos	Na escolha dos jogos é importante considerar familiaridade, tempo de jogabilidade e implementação.
Elaboração de atividades	É essencial relacionar os conceitos ministrados com os jogos selecionados a serem desenvolvidos.
3) Ensino e Aprendizagem	
Componente	**Pontos a considerar**
Aulas e tutoriais	Apresente conceitos básicos sobre jogos e estrutura básica de um jogo. Use exemplos de criação de jogos para cada tópico abordado.
Atividades práticas de programação	Importante forma de analisar o progresso dos discentes durante o processo de ensino aprendizagem.
Avaliação de aprendizagem	Testes, provas e trabalhos que compõem a nota final do aluno na disciplina. Sugerimos a avaliação baseada em portfólios.
4) Reflexões e *feedbacks*	
Componente	**Pontos a considerar**
Feedback do docente	Revisar todas as fases do guia e fazer as mudanças necessárias para implementar novamente.
Feedback dos alunos	Os alunos respondem questionários que podem ajudar os professores a fazerem as mudanças para as próximas disciplinas.
Feedback da instituição	Averiguação da conformidade dos documentos institucionais com as atuais necessidades de aprendizagem dos alunos.

Fonte: Próprio autor.

Em suma, destacamos que este guia não tem o objetivo de ensinar programação. De outro modo, o guia se concentra no ensino inicial de programação. O modelo apresentado não se limita ao ensino de uma linguagem específica nem tampouco a um paradigma de programação específico. De tal maneira, é fundamental que o docente esteja atento a cumprir as exigências dos documentos institucionais quanto à linguagem de programação usada na disciplina.

Este guia pode ser implementando em uma disciplina de programação em sua totalidade, ou seja, todos os tópicos e conceitos de programação inicial teriam suas atividades práticas relacionadas com o desenvolvimento de jogos. Entretanto, é possível que o docente interessado neste guia não se sinta familiarizado com os conceitos de jogos em geral, ou com o desenvolvimento de jogos em si. Neste caso,

sugerimos que o docente inicialmente tente algo mais simples, uma abordagem que contemple somente um tópico da disciplina com atividades práticas de criação de jogos, enquanto os demais tópicos sejam ensinados a partir das atividades convencionais que o docente já propunha anteriormente. Alguns professores podem escolher começar com pequenas atividades em tópicos específicos, enquanto outros podem aplicar a metodologia na disciplina em sua totalidade. Essa escolha vai depender da autoanálise do docente acerca de fatores como familiaridade com o tema, tempo disponível para planejamento (total ou parcial), e das diretrizes da instituição inerente a mudanças bruscas na condução da unidade curricular.

Este guia está apresentado e esclarecido em detalhes ao longo deste Capítulo 2, entretanto uma versão sintética do instrumento está disponível no Apêndice D desta tese. O guia pode conter mais, ou menos, material do que uma disciplina de programação pode exigir. Portanto, alertamos que o professor selecione os tópicos e atividades que se encaixam no contexto em que a sua disciplina será ministrada. Ao colocar este guia em prática, o docente deve ter em mente que isto é um modelo para ensino baseado no construcionismo, em atividades práticas, ou seja, a programação de computadores não pode ser aprendida somente lendo este guia, mas sim colocando-o em funcionamento.

3 IMPLEMENTAÇÃO EXPERIMENTAL DO GUIA

O processo de desenvolvimento desta pesquisa propôs a elaboração de um guia para ensino de programação por meio da construção de jogos, utilizando linguagens de programação convencionais, tendo como objetivo a orientação do planejamento, do desenvolvimento e da análise da metodologia no âmbito de uma disciplina introdutória de programação. Além disto, por meio de um estudo de caso procuramos examinar as contribuições que essa proposta metodológica pode proporcionar no sentido de atender às demandas existentes no contexto das dificuldades e barreiras das unidades curriculares de programação dos cursos de TI.

3.1 DELINEAMENTO DO EXPERIMENTO

Escolhemos o "estudo de caso" como forma de compreender o assunto em estudo e ao mesmo tempo desenvolver teorias mais genéricas a respeito do fenômeno observado. Além de descrever fatos ou situações, o estudo de caso busca proporcionar conhecimento acerca do fenômeno estudado e comprovar ou contrastar relações evidenciadas no caso.

Na posição de Lüdke e André (1986), o estudo de caso como estratégia de pesquisa é simples e específico ou complexo e abstrato e deve ser sempre bem delimitado. Pode ser semelhante a outros, mas é também distinto, pois tem um interesse próprio, único, particular e representa um potencial na educação. Destacam em seus estudos as características de casos naturalísticos, ricos em dados descritivos, com um plano aberto e flexível que focaliza a realidade de modo complexo e contextualizado.

Tendo em conta a posição de Lüdke e André (1986), o estudo de caso como modalidade de pesquisa é entendido como uma metodologia ou como a escolha de um objeto de estudo definido pelo interesse em casos individuais. Em outras palavras, visa à investigação de um caso específico, bem delimitado, contextualizado em tempo e lugar para que se possa realizar uma busca circunstanciada de informações.

Este estudo de caso tem como objetivo analisar e compreender a aplicação real do guia proposto em uma disciplina introdutória de programação. O objetivo desta investigação em sala de aula foi verificar a existência de pontos falhos e investigar a coerência entre teoria e prática.

Realizamos este estudo em sala de aula no campus Ituiutaba do Instituto Federal do Triângulo Mineiro (IFTM), durante a disciplina de Programação Estruturada do Curso Superior de Análise e Desenvolvimento de Sistemas. A disciplina de Programação Estruturada deste curso é ministrada utilizando a linguagem de programação C e contempla os tópicos introdutórios de programação de computadores,

como: entrada e saída de dados; variáveis; estruturas condicionais; estruturas de repetição; vetores e matrizes. O conteúdo programático desta disciplina está de acordo com aquilo que é recomendado em nossa proposta do guia em pauta.

Para isso, solicitamos autorização da direção geral do campus Ituiutaba do IFTM para realização desta intervenção, bem como do coordenador do curso, e do professor responsável pela disciplina. Todas as autorizações foram concedidas. Além disto, também foi solicitado de cada participante da pesquisa que consentisse com a sua participação por meio do termo de livre consentimento.

O processo de implementação experimental do guia foi conduzido pelo professor responsável pela disciplina, de maneira que nós como autores do guia nos distanciamos deste experimento a fim de não enviesar os resultados. O docente teve acesso ao guia otimizado (Capítulo 2 desta tese) para que pudesse colocar as ideias e práticas em execução no ambiente da unidade curricular de Programação Estruturada. Fundamentalmente, o processo consistiu na implementação do guia, observando as fases 3 (Ensino e Aprendizagem) e 4 (Reflexões e *Feedbacks*) do mesmo, realizando a ministração das aulas, proposição de atividades práticas de construção de jogos, uso de portfólios, coleta de *feedbacks* dos discentes. Durante o processo de implementação, era essencial que o docente aplicador coletasse dados e informações para subsidiar a pesquisa e a investigação acerca desta implementação experimental. Sendo assim, foram utilizados os seguintes recursos para coleta de dados:

Aplicação de questionários: Total de dois, sendo o primeiro (Apêndice A) aplicado antes da intervenção do professor, que teve como objetivo traçar um perfil do sujeito (idade, gênero, grau de interesse em jogos), também conhecer o grau de motivação e interesse do discente em relação à metodologia tradicional de ensino de programação, assim como o nível de conhecimento do discente em relação aos conteúdos já ministrados; o segundo (Apêndice B) aplicado após a intervenção docente, que teve como objetivo analisar novamente o grau de motivação, interesse e engajamento na disciplina, de modo a conhecer se houve alterações nestes aspectos. A aplicação do segundo questionário também analisou as sugestões dos discentes sobre o uso da metodologia de criação de jogos nesta disciplina de programação.

Análise das observações do docente aplicador: Por meio de um questionário (Apêndice C), foi proposto ao docente aplicador que fizesse suas próprias observações sistemáticas para compreender o comportamento dos discentes e verificar os problemas e as dificuldades

que se apresentaram durante a aplicação da metodologia, bem como o desenvolvimento das atividades práticas propostas, de forma que o docente pudesse sugerir modificações e melhorias nas diretrizes do guia proposto inicialmente. Neste questionário destinado ao docente, também solicitamos que o mesmo fizesse uma breve *análise da produção discente.* Os alunos reuniram suas produções em portfólios pessoais e a análise dos jogos produzidos possibilitou uma análise do desempenho discente frente ao uso desta metodologia, bem como averiguar a quantidade de tópicos abordados durante o processo.

A análise dos dados coletados foi realizada por meio da estratégia de análise de conteúdo. Segundo Bardin (1979), a expressão análise de conteúdo descreve uma coleção de métodos de análise das comunicações com a intenção de colher indicadores (quantitativos ou não) que possibilitem a inferência de compreensões e percepções inerentes às condições de produção destas comunicações. Tais análises poderão ser utilizadas em uma possível revisão do guia de modo a otimizá-lo ainda mais, corrigindo algumas falhas e agregando as sugestões de melhorias sugeridas pelo professor aplicador.

3.2 ANÁLISE E DISCUSSÃO DOS RESULTADOS

A implementação em caráter experimental contou com a colaboração de um docente do Instituto Federal do Triângulo Mineiro – campus Ituiutaba, que colocou este guia em prática em uma turma de programação de computadores composta por 14 alunos do curso superior em Análise e Desenvolvimento de Sistemas.

Conforme mencionado na seção 3.1, foram aplicados dois questionários (um inicial e outro final) aos alunos participantes, e um questionário ao docente colaborador. O questionário inicial (Apêndice A) foi elaborado com questões com o objetivo de traçar um perfil, mesmo que rudimentar, do aluno participante da implementação deste guia de ensino de programação. O questionário final (Apêndice B) foi confeccionado com o intuito de assimilar as impressões dos alunos participantes acerca da abordagem de desenvolvimento de jogos em uma disciplina de programação. O questionário ao docente (Apêndice C) foi idealizado com o propósito de captar as percepções, contribuições e sugestões do colaborador ao colocar este guia em prática em uma turma de programação.

3.2.1 Questionário inicial: discussão e análises

Investigamos inicialmente a faixa etária dos participantes da implementação, para então identificar se a turma é homogênea ou

heterogênea no que diz respeito a idade dos alunos. Sobre este aspecto – a faixa etária dos indivíduos participantes – identificamos que coletivo de alunos tem idade mínima de 20 anos e máxima de 40 anos conforme pode ser visto no Gráfico 1 a seguir.

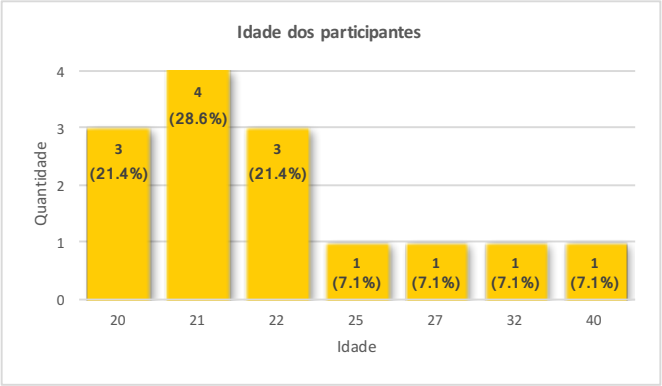

Gráfico 1: Idade dos participantes da implementação
Fonte: Resultados estatísticos dos questionários.

Como podemos observar, 71,4% dos participantes tem idade entre 20 e 22 anos, o que mostra uma turma predominantemente jovem. Podemos notar também a presença de alunos com idades de 25, 27, 32 e 40 anos, tornando este coletivo de participantes uma turma heterogênea concernente à faixa etária. Tal fato não impossibilitou, tampouco foi motivo de barreira, para implementação das atividades de programação de jogos.

Procuramos também investigar o tempo investido pelos participantes semanalmente em jogos, para que pudéssemos identificar o grau de interesse, ou familiaridade, dos participantes em relação aos jogos. Além disso, poderíamos verificar a existência, ou não, de alguma relação entre idade do participante e o tempo investido em jogos pelo mesmo. Para tanto, estabelecemos uma questão do tipo fechada com cinco alternativas: uma hora por semana ou menos; entre uma e três horas por semana; entre três e cinco horas por semana; entre cinco e dez horas por semana; e, por fim, mais de dez horas por semana. Os resultados desta questão estão no Gráfico 2 exibido a seguir.

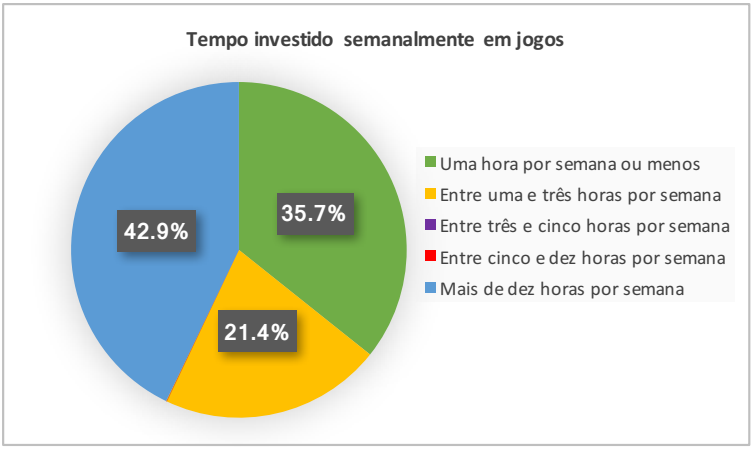

Gráfico 2: Tempo investido semanalmente em jogos
Fonte: Resultados estatísticos dos questionários.

O Gráfico 2 nos mostra que 42,9% dos participantes declararam que investem um tempo maior do que dez horas por semana em jogos. Por outro lado, 35,7% afirmaram que não dedicam mais do que uma hora semanalmente em jogos. Podemos considerar que em relação ao tempo gasto com jogos, este grupo é diversificado. Há participantes que despendem um tempo considerável com jogos, e há aqueles que não ocupam muito do seu tempo com tais atividades. Contudo, esta diversidade não provocou empecilhos, nem dificuldades, durante a implementação do guia de ensino de programação. Neste grupo, nenhum participante declarou que joga entre três a cinco horas, ou entre cinco a dez horas.

Ainda referente a análise do Gráfico 2, notamos a possível existência de uma polaridade neste comportamento, ou seja: de um lado participantes que dedicam muito tempo (mais de dez horas semanais) aos jogos, e de outro lado alunos que dedicam pouco tempo (uma hora ou menos). Isto pode indicar que, talvez, uma parte desse grupo tem ampla familiaridade com jogos, enquanto outra parte do grupo tem pouca experiência com estes artefatos. O surgimento desta questão da familiaridade dos alunos com jogos, remete-nos a um alerta feita na Fase 2 do guia, acerca do cuidado do docente durante a elaboração de atividades para observar a adequação e os níveis de desafio que este deseja para sua classe. É importante observar a neutralidade do material para evitar a alienação de grupos e minorias.

Procuramos identificar dentro deste coletivo de alunos a existência de uma relação entre idade dos participantes e o tempo investido em jogos por eles. Traçamos, então, um gráfico de barras com as idades dos 14 participantes e, em seguida, atribuímos valores de um 1 a 5 para cada uma das alternativas da questão sobre o tempo investido em jogos, e traçamos uma linha com esses valores sobrepondo o gráfico das idades. O Gráfico 3, a seguir, ilustra esse relacionamento entre as respostas das duas questões.

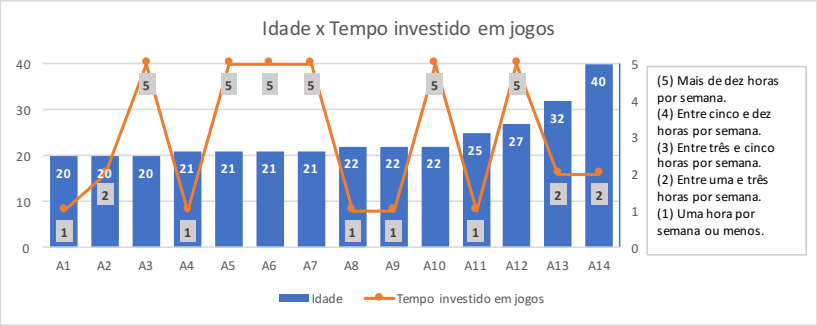

Gráfico 3: Relação entre idade e tempo investido em jogos
Fonte: Resultados estatísticos dos questionários.

O Gráfico 3 nos apresenta a relação encontrada entre a idade dos participantes e o tempo investido em jogos por cada um deles. Percebemos que 5 dos 10 alunos com idade de até 22 anos despendem mais de dez horas por semana em jogos eletrônicos. Por outro lado, 4 dos 10 alunos desse mesmo subgrupo (A1, A4, A8 e A9) declararam que não ocupam mais do que uma hora com jogo no período de uma semana. Dos alunos restantes, subgrupo acima de 25 anos, um deles (A12) declarou passar mais de dez horas por semana jogando, enquanto outro (A11) declarou passar uma hora ou menos por semana jogando, e os dois demais (A13 e A14) asseveraram investir entre uma e três horas por semana com jogos.

A análise deste Gráfico 3 nos mostra que dentro desse grupo pesquisado, os alunos mais jovens tendem a investir mais do seu tempo semanal com jogos do que os alunos com idade acima de 30 anos. Embora alguns poucos alunos (A1, A4, A8, A9 e A11) tenham declarado que passam uma hora ou menos jogando, os dez demais alunos com idade até 27 anos dedicam uma porção considerável de seu tempo semanal com jogos. Esta análise específica não deve ser interpretada como uma regra geral, portanto consideramos adequado que o docente – interessado em

usar este guia – antes de adotá-lo em suas aulas, investigue, mesmo que a turma seja composta em sua maioria por jovens, se estes têm interesse (ou familiaridade) com jogos.

Examinamos similarmente o tempo investido pelos participantes semanalmente nos estudos de programação, além do horário em sala de aula, com o objetivo de identificar o grau de dedicação dos participantes em relação à prática de programação. Além disso, poderíamos também verificar a existência de alguma relação entre idade do participante e o tempo de estudo de programação de cada um. Da mesma forma da pergunta anterior, estabelecemos assim uma questão do tipo fechada com cinco alternativas: uma hora por semana ou menos; entre uma e três horas por semana; entre três e cinco horas por semana; entre cinco e dez horas por semana; e, por fim, mais de dez horas por semana. As respostas a essa questão são apresentadas no Gráfico 4 exibido a seguir.

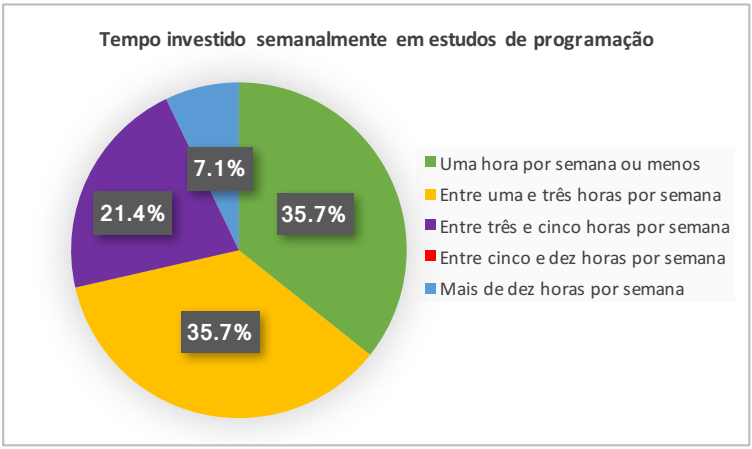

Gráfico 4: Tempo dedicado semanalmente em estudos de programação
Fonte: Resultados estatísticos dos questionários.

Observamos que 71,4% dos participantes da pesquisa declararam que dedicam três horas, ou menos, por semana aos estudos de programação, ou seja, em média não mais que 25 minutos por dia, além do horário de aula. Neste grupo, nenhum participante declarou que estipula entre cinco a dez horas por semana para estudar programação. Três alunos (21,4%) responderam que investem entre três e cinco horas por semana, e apenas um participante (7,1%) afirmou que destina mais dez horas por semana ao aprendizado de programação, ou seja, mais de 85 minutos diários. Considerando a alta carga de atividades práticas de

uma disciplina de programação, qual será o motivo de mais de 70% destes alunos investirem menos de 25 minutos diários para estudar programação? Porventura seria uma dificuldade na gestão do tempo?

A carga de atividades práticas em uma disciplina de programação é alta e, por isso, eventualmente, não é viável destinar o escasso tempo dentro de sala de aula para realização de todas as atividades práticas propostas. Em virtude disso, é importante que o professor disponha do seu tempo de aula buscando ser um intermediador na jornada de aprendizado que o aluno está percorrendo, ministrando conteúdos, sanando dúvidas e guiando o discente em sua busca. Desse modo, para que a disciplina não tenha o seu andamento negativamente afetado, pode ser proveitoso que as atividades práticas propostas, ocasionalmente, sejam realizadas pelos alunos em um momento extra-classe. É importante que o docente administre bem o tempo entre teoria e prática dentro de sala de aula, como também administrar as atividades, projetos e trabalhos que serão realizados pelos alunos em um momento não conflitante com o horário de aula. Estas sugestões estão de acordo com os Referenciais de Formação em Computação (RF) da SBC, de acordo com Zorzo et al. (2017, p. 37)

Procuramos reconhecer dentro desse específico grupo de alunos a presença de uma associação entre idade dos participantes e o tempo investido em seus estudos de programação. Do mesmo modo que estabelecemos anteriormente, esboçamos um gráfico de barras com as idades dos 14 participantes e, em seguida, atribuímos valores de um 1 a 5 para cada uma das alternativas da questão sobre o tempo investido em estudos de programação, e traçamos uma linha com estes valores sobrepondo o gráfico das idades. O Gráfico 5, a seguir, ilustra esse relacionamento entre as respostas das duas questões.

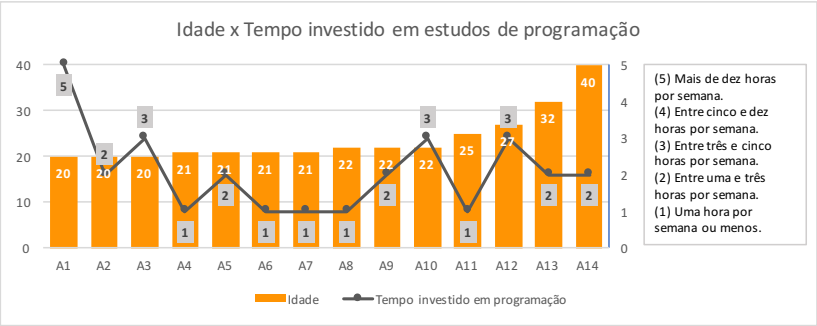

Gráfico 5: Relação entre idade e tempo investido em estudos de programação

Fonte: Resultados estatísticos dos questionários.

O Gráfico 5 nos apresenta a relação encontrada entre a idade dos participantes e o tempo investido por cada um deles em estudar programação além do horário de aula. Como vimos antes, em geral, o tempo médio de estudos de programação, nesta turma, é baixo. A exceção é um aluno (A1) que disse estudar mais de dez horas semanais. Os alunos com idade acima de 30 anos (A13 e A14), também declararam que suas cargas horárias de estudos são baixas, entre uma e três horas semanais, equiparando-os (neste aspecto) aos alunos mais jovens (faixa etária de 20 a 25 anos). Devido a essa equiparação, não podemos distinguir de fato uma conexão entre a idade dos participantes e o tempo dedicado ao aprendizado de programação extra-classe.

Diante do exposto, dispomo-nos a identificar alguma associação entre idade dos participantes, o tempo dedicado aos jogos e o tempo investido em seus estudos de programação. De similar forma que estabelecemos previamente, esboçamos um gráfico de barras com as idades dos 14 participantes e, na sequência, atribuímos valores de um 1 a 5 para cada uma das alternativas da questão sobre o tempo investido em estudos de programação, bem como para cada uma das respostas da pergunta sobre tempo dedicado aos jogos, e assim traçamos duas linhas com estes valores sobrepondo o gráfico das idades. O Gráfico 6, a seguir, mostra esse relacionamento entre as respostas das três questões.

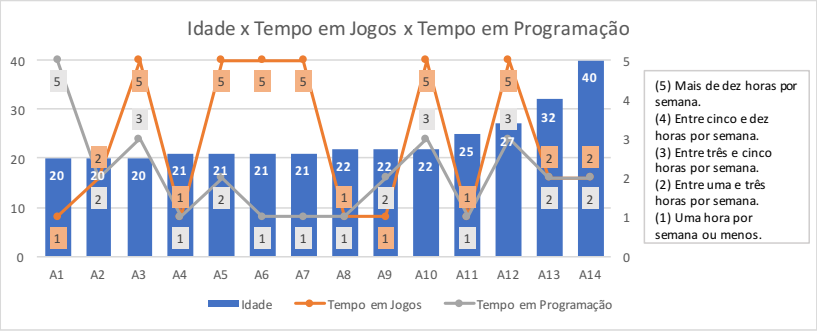

Gráfico 6: Relação entre idade, tempo investido em jogos, e tempo dedicado aos estudos de programação
Fonte: Resultados estatísticos dos questionários.

O que vemos no Gráfico 6 é a relação entre a idade, o tempo dedicado aos estudos de programação e o tempo investido em jogos pelos alunos participantes da pesquisa. Nos chama a atenção inicialmente as

distâncias e proximidades dos pontos das curvas referentes aos tempos de jogos e estudos de programação. Os alunos com idades acima de 30 anos, deste grupo especificamente, parecem equilibrar o tempo de estudos com o tempo para jogar, devido à proximidade dos pontos nas duas curvas. Também podemos perceber esse comportamento em quatro alunos com idade abaixo de 25 anos (A2, A4, A8 e A11). Por outro lado, identificamos alunos com um distanciamento entre as duas curvas, seja privilegiando os estudos (A1), ou favorecendo o tempo despendido com jogos (A5, A6 e A7). Por que razão estes alunos deste grupo específico decidem favorecer seu tempo de diversão com jogos em detrimento do tempo de estudos? Porventura esses participantes têm um conhecimento mais amplo de programação e não necessitam de uma carga horária de estudos maior? Ou porventura as atividades práticas propostas não são desafiadores o bastante para tais alunos? Encontrar um ponto de equilíbrio acerca do nível de desafio das atividades práticas é um ponto importante a ser considerado pelo docente. De igual modo para os estudantes, encontrar um ponto de equilíbrio entre tempo de entretenimento com jogos e o tempo de estudo é de essencial importância para sua jornada de aprendizado.

O nível de conhecimento em programação antes de ingressar no curso também foi alvo do nosso questionamento aos participantes da pesquisa. Indagamos essa informação para que tivéssemos uma percepção sobre qual influência o grau de experiência em programação de computadores pode exercer sobre a implementação dessa metodologia. Estipulamos, assim, uma questão do tipo fechada com cinco alternativas a respeito do grau de conhecimento em programação de cada um: não sabia nada; pouco conhecimento; conhecimento razoável; sabia programar; e, por fim, sabia programar bem. As respostas a essa questão estão no Gráfico 7 exposto a seguir.

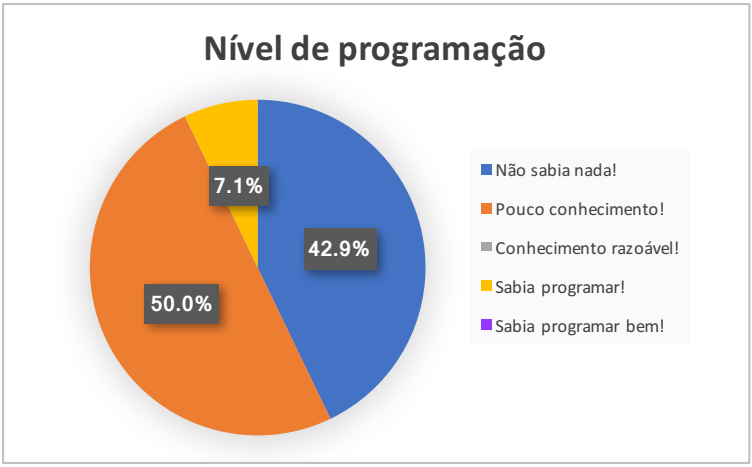

Gráfico 7: Nível de conhecimento em programação antes de entrar no curso
Fonte: Resultados estatísticos dos questionários.

Podemos ver no Gráfico 7 as asserções dos participantes acerca de seus conhecimentos prévios em programação antes de ingressarem no curso. Inferimos, a partir desse gráfico, que quase todos os alunos dessa turma tinham pouco ou nenhum conhecimento de programação antes de se vincularem ao curso. Apenas um aluno declarou que já sabia como programar, enquanto os demais se dividiram em um grupo que não possuía experiência em programação e outro grupo que tinha uma pequena experiência no assunto. Considerando que essa é uma turma inicial de programação, esse é um resultado esperado, entretanto há um aluno nessa turma que já sabia programar. Sendo assim, esse fato pode se tornar uma barreira no aprendizado? Esse aluno, eventualmente, pode se encontrar desmotivado caso ele julgue que o ritmo e o andamento da disciplina estão aquém de suas expectativas. O ponto que queremos ressaltar refere-se ao nível de desafio proposto durantes as atividades práticas e também às novidades acerca do conteúdo ministrado. Enquanto para seus colegas as atividades têm um alto grau de desafio, e os conteúdos ministrados são informações novas, para esse aluno que já sabe programar, em especial, a medida de desafio pode parecer menor, e o conteúdo ministrado pode parecer repetitivo.

Na sequência fizemos duas perguntas acerca do entendimento que os participantes tinham a respeito das atuais atividades práticas de programação nessa disciplina. Na primeira, perguntamos "*Você considera que as atividades práticas e exercícios de programação propostos nesta*

disciplina atualmente são adequadas para aprender programação?", e na segunda questão indagamos "*Você pensa que as atividades práticas e exercícios de programação propostos nesta disciplina atualmente, estão de acordo com os desafios que espera enfrentar no mercado de trabalho, ou em estudos futuros, após concluir este curso?*". Ambas as perguntas eram do tipo fechadas utilizando uma escala de Likert[3] que variou de 1 até 5, sendo (1) equivalente a "inadequadas" e (5) correspondente a "totalmente adequadas". Acerca dos entendimentos dos alunos sobre essas duas questões, suas respostas estão no Gráfico 8 a seguir.

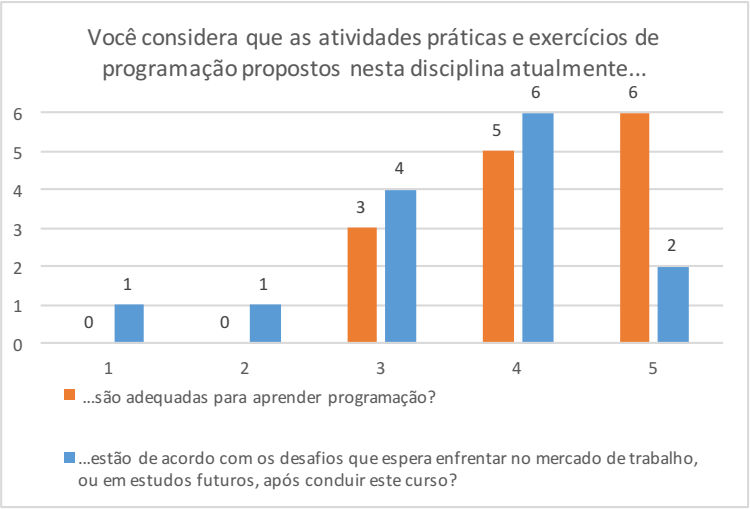

Gráfico 8: Percepções acerca das atividades práticas de programação
Fonte: Resultados estatísticos dos questionários.

Vale ressaltar que essas perguntas foram feitas no questionário inicial (Apêndice A), ou seja, essas questões referem-se às atividades práticas antes da aplicação da metodologia de criação de jogos, ou seja, dizem respeito às atividades práticas tradicionais que não abordam a criação de jogos ainda. Optamos por plotar as respostas das duas perguntas no mesmo Gráfico 8 para facilitar a comparação entre as asserções as duas questões. Por um lado, vemos que a turma considera as atividades práticas propostas adequadas para o aprendizado de

[3] Escala Likert mede atitudes e comportamentos utilizando opções de resposta que variam de um extremo a outro (exemplo, de nada provável para extremamente provável). Fonte: http://pt.surveymonkey.com/mp/likert-scale

programação, uma vez que nenhum participante declarou que as atividades são totalmente inadequadas (níveis 1 e 2) para o aprendizado de programação, ao contrário, suas respostas se concentram nos níveis 4 e 5 (totalmente adequadas). Contudo, quando indagados sobre a existência da relação real entre essas atividades e os desafios do mercado de trabalho, as respostas se diluíram ao longo das alternativas. Dois participantes responderam que as atividades não estariam de acordo (níveis 1 e 2), enquanto apenas dois alunos declararam que as atividades estariam plenamente de acordo com o mercado de trabalho, ou estudos futuros. Estas são as percepções dos estudantes acerca das atividades práticas antes da implementação da metodologia de criação de jogos. Posteriormente, tais percepções serão comparadas com as impressões dos participantes relatadas no questionário final (Apêndice B) que investigou o cenário após aplicação da metodologia de desenvolvimento de jogos.

Ainda no questionário inicial, solicitamos aos participantes que relatassem, e justificassem se possível, as suas maiores dificuldades concernentes ao aprendizado de programação. O relato de complicações enfrentadas pelos participantes é importante para que possamos analisar quais tipos de barreiras e obstáculos podem eventualmente surgir durante a jornada de aprendizado de um aluno de programação. Ciente dessas possíveis complicações, o docente pode auxiliar seus alunos a desfazerem tais embaraços e assim ajuda-los a prosseguirem em sua caminhada. A seguir, listamos o conjunto de apontamentos dos participantes a respeito de suas dificuldades.

> Nenhuma, apenas acho o conteúdo passado bem raso (A1).
> Questão de horário para aprendizado, e para encontrar conteúdos também (A2).
> Nenhum (A3).
> Geralmente onde deve-se usar cada conceito, exemplo: quando se deve usar, como se deve usar, onde se deve usar (A4).
> Foco (A5).
> A parte mais difícil é ficar motivado e não ter preguiça (A6).
> Sintaxe em geral é o problema, sou razoável em lógica (A7).
> Utilizar a lógica de programação é complicado, consigo saber o que fazer, porém não sei implementar bem (A8).

> Maior dificuldade nas aulas em si muitas vezes e por parte dos professores que não costumam preparar bem as aulas, deixando o conteúdo a desejar (A9).
> De princípio eu não tinha muito interesse na área de programação, tinha um raciocínio muito lento para pensar uma solução para determinado problema (A10).
> Pouco tempo 'pra' estudar o conteúdo em casa (A11).
> Adaptação a uma nova linguagem (A12).
> Acompanhar o ritmo dos colegas (A13).
> Muito conteúdo ministrado, muito conteúdo para pouco tempo de aula (A14).

À primeira vista identificamos barreiras específicas, e em alguns casos são situações compartilhadas por dois ou mais participantes como, por exemplo, a falta de tempo ou dificuldade com a lógica de programação. As dificuldades relatadas são diversificadas, enquanto uns queixam da superficialidade do conteúdo (A1, A9), outros lamentam a falta de tempo para estudar (A2, A11, A14), alguns encontram impasses na aplicação da lógica ou sintaxe de programação (A4, A7, A8, A10), ao passo que a dificuldade de outros é foco ou motivação (A5, A6), ao mesmo tempo que outros não veem obstáculos na caminhada (A3).

Para cada um dos tipos de problemas comunicados, o docente pode auxiliar seus alunos a encontrar uma solução, ou um caminho mais adequado, para que tais barreiras não perturbem demasiadamente a sua jornada em busca do conhecimento. Em relação à superficialidade de conteúdo, o docente poderia propor atividades práticas com níveis de desafios diferenciados (dos mais simples aos mais complexos) entre os alunos. Acerca da falta de tempo para estudar, o educador poderia encontrar uma nova harmonização entre teoria e prática dentro do horário de aula, de modo que os alunos possam realizar atividades práticas na presença do professor para que este tenha possibilidade de sanar possíveis dúvidas *in loco*.

Concernente aos impasses na aplicação da lógica de programação, problemas de adaptação a uma nova linguagem e também dificuldades para acompanhar o ritmo da turma, o docente poderia propor atividades extras (além das práticas previstas), como por exemplo lista de práticas complementares, ou desenvolvimento de várias aplicações menores ao invés uma única aplicação de médio, ou grande, porte. Desse modo, os

alunos que se deparam com esse tipo de empecilho terão um contato maior com os estudos de programação.

Alguns participantes da turma relataram a falta de foco e/ou motivação como contratempos na aprendizagem de programação. Uma possível maneira de contornar essa situação seria se o docente trouxesse para a disciplina atividades que despertem o interesse dos alunos, ou atividades que estejam de certa forma mais próximas à realidade desses alunos desmotivados, como por exemplo o desenvolvimento de jogos, ideia central deste guia.

E ainda temos o aluno que declara não existir problemas, ou dificuldades em seu aprendizado de programação. Aparentemente é a resposta perfeita que um docente gostaria de ouvir: "nenhuma dificuldade de aprendizado", ou "nenhum problema nesta disciplina". Porém, esta resposta em particular nos suscitou muitas inquietações. Por que este aluno pensa assim? Ele não vê problemas em seu aprendizado por que ele está aprendendo tudo que está sendo ensinado? Estaria ele plenamente satisfeito com o andamento da disciplina e com o ritmo de sua jornada de aprendizado? Ou estaria ele acomodado em sua caminhada sem ter um objetivo final bem definido? Será que ele está desinteressado nos estudos de programação, a ponto de declarar que não há problemas por que ele não se importa com o conteúdo da disciplina?

E, por fim, na última pergunta do questionário inicial solicitamos que os participantes elaborassem sugestões para tornar as aulas de programação mais ricas em termos de aprendizagem e interesse. A ideia de co-construir conhecimento com estudantes pode ser uma coisa infactível para alguns professores. Para Freire (1974), aprender é entendido como um ato social, no qual professores e alunos dialogam e todos criam conhecimento em conjunto, contrapondo o cenário no qual professores "depositam" conteúdos e informações nos alunos como se estes fossem caixas vazias, fazendo referência ao método de "educação bancária". Freire esclarece:

> [...] e já que o diálogo é o encontro em que a reflexão e a ação unidas de quem dialoga são dirigidas ao mundo que deve ser transformado e humanizado, esse diálogo não pode ser reduzido ao ato de uma pessoa "depositar" ideias nas outras, nem tornar-se um simples intercâmbio a ser "consumido" pelos debatedores [...]. Pelo fato de o diálogo ser um encontro entre [os humanos] que dão nome ao mundo, não deve ser uma situação em

> que alguns [humanos] o nomeiam pelos outros. (Freire, 1974, p. 77)

De tal modo, quando estamos em um diálogo, não devemos "depositar" nossas ideias nos outros indivíduos, uma vez que tal comportamento seria defini-las como objetos determinados e manipulados. Aprender por meio da interlocução significa nomear o mundo em sintonia com os outros, em um ato social, processo que o auxilia a compreende-lo por conta própria.

Concordamos que os professores têm a responsabilidade de serem tomadores de decisões sobre muitas coisas. Não estamos propondo compartilhamento de autoridade e controle com os alunos, contudo considere um convite para dar um passo ao lado e investigar o que pode acontecer quando mudamos nossa perspectiva e aprofundamos nesta experimentação. Na sequência, apresentamos as asserções dos participantes na última pergunta do primeiro questionário quando solicitamos aos participantes que apontassem sugestões para tornar as aulas de programação mais ricas em termos de aprendizagem e interesse.

> Ter foco na qualidade do *software*, além de mais foco nas áreas *web* e *mobile* (A1).
> Ter mais aulas práticas, e aulas mais voltadas ao aprendizado para o mercado de trabalho, visando mais aplicar conhecimento na área e não só teorias (A2).
> Assuntos que despertem o interesse dos alunos (A3).
> Colocar sempre temas atualizados, só que se deve ensinar desde como instalar ao processo final, além de oferecer ferramentas que funcionem não só em um sistema operacional mais na maioria (A4).
> Projetos a médio prazo (A5).
> ¯_(' n')_/¯ (A6).
> Deixar as aulas mais dinâmicas ajudaria bastante, não sendo muito sério com os alunos, mas ainda assim sem perder a autoridade, a maioria dos professores já explica com cara de cansado (A7).
> Minha sugestão na verdade seriam os cursos de Ciência da Computação não focarem somente em programação, possui diversas outras áreas de constante crescimento como: virtualização de *data centers*, coisas que não vemos na faculdade (A8).

> Para se tornar as aulas mais interessantes e mais agregativas, penso que as aulas devem ser baseadas e conteúdos baseados na vida real e no mercado de trabalho. Devemos aprender algo que seja relevante para nossa vida e para nossas futuras carreiras profissionais (A9).
> Trazer algo envolvido mais com a realidade, com o mercado de trabalho (A10).
> Simular o desenvolvimento de um sistema como no mercado de trabalho (A11).
> Miniprojetos práticos de diferentes plataformas de jogos (*arcade*, aventura, *puzzle*, quebra-cabeça) (A12).
> Está 100% (A13).
> Mais exemplos práticos (A14).

Ao analisar inicialmente todas as sugestões apresentadas, podemos perceber que a maioria (10 participantes, 71,4%) apontam como sugestão a adesão de temas mais próximos à realidade do mercado de trabalho (A1, A2, A4, A5, A8, A9, A10, A11, A12 e A14). A preocupação desse grupo de alunos é relevante, considerando o fato de que eles ingressaram em um curso buscando se posicionar no mercado de trabalho e para isso eles esperam ser preparados para que tal posicionamento seja bem-sucedido. Dos quatro demais participantes, um sugeriu o uso de assuntos que despertem o interesse dos alunos (A3); outro apresentou um desenho que, no nosso entendimento, significa "não tenho sugestões", ou "não quero apresentar sugestões" (A6); outro reclamou da postura de alguns professores, não se queixou da disciplina, ou do conteúdo, mas sim do comportamento de alguns docentes sem identifica-los (A7); e, por fim, um participante considera que que não há melhorias a serem feitas na disciplina de programação (A13).

Como já discutimos anteriormente, muitos alunos em aulas de programação introdutória não conseguem entender o significado e a utilidade dos conceitos que estão sendo ensinados. Sua baixa motivação afeta sua aprendizagem. Uma estratégia tradicional muito utilizada por professores refere-se a uma abordagem que enfatiza o ensino de programação voltado para a própria computação, bem como as atribuições abstratas, como a computação da função fatorial. Contudo, alguns educadores aprimoram essas abordagens tradicionais ao ensinar conceitos em contextos que os alunos consideram mais relevantes, como jogos, robótica e meios de comunicação. Essa abordagem é mais do que apenas uma coleção de "atividades práticas cativantes", ao invés disso é uma

atitude que afeta a escolha de tópicos e pedagogia, que, em conjunto, podem conduzir a uma maior satisfação dos alunos.

Essas foram as análises e discussões sobre as respostas encontradas no questionário inicial proposto aos participantes antes da implementação da abordagem de desenvolvimento de jogos na disciplina de programação. Depois disso, o docente colaborador passou a propor atividades práticas inerentes à criação, desenvolvimento e programação de jogos, como proposto neste guia. Perto do encerramento da disciplina, foi solicitado aos alunos que respondessem o questionário final que objetivou coletar suas impressões, opiniões e percepções em relação à metodologia de desenvolvimento de jogos na disciplina de programação.

3.2.2 Questionário final: discussão e análises

Os alunos participantes da implementação deste guia em uma disciplina de programação tiveram a oportunidade de responder um questionário final. Inicialmente, foram feitas duas perguntas acerca do entendimento que os participantes tinham a respeito das atividades práticas de programação relacionadas com a criação de jogos nesta disciplina. Na primeira, perguntamos "*As atividades práticas e exercícios de programação relacionados com a criação de jogos são adequadas para aprender programação?*", e na segunda questão indagamos "*As atividades práticas e exercícios de programação relacionados com a criação de jogos, estão de acordo com os desafios que você espera enfrentar no mercado de trabalho, ou em futuros cursos, após concluir este curso?*". As questões deste questionário final são similares as perguntas questionário inicial, de maneira que a comparação entre ambas fosse facilitada. Estas perguntas eram do tipo fechadas utilizando uma escala Likert que variou de 1 até 5, sendo (1) equivalente a "inadequadas" e (5) correspondente a "totalmente adequadas". Acerca das opiniões dos alunos sobre essas duas questões, suas considerações estão no Gráfico 9 a seguir.

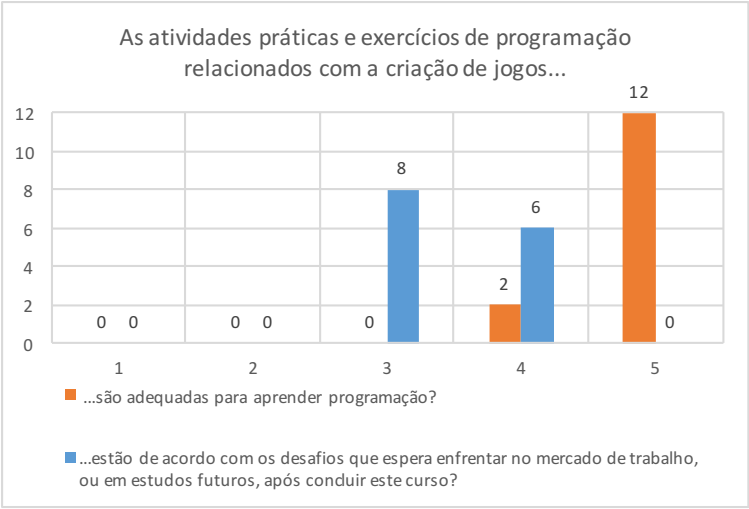

Gráfico 9: Percepções acerca das atividades práticas de programação relacionadas com a criação de jogos
Fonte: Resultados estatísticos dos questionários.

Vale ressaltar que essas perguntas constam no questionário final (Apêndice B), ou seja, referem-se às atividades práticas após da aplicação da metodologia de criação de jogos. São atividades práticas que já abordam a criação de jogos. Optamos por plotar as respostas das duas perguntas no mesmo Gráfico 9 para facilitar a comparação entre as asserções as duas questões. O fato que se destaca é que 12 alunos (85,7%) concordam plenamente que as atividades práticas relacionadas com criação de jogos são adequadas para aprender programação, enquanto os dois demais participantes colocaram suas respostas no nível 4, ou seja, concordam parcialmente.

No entanto, quando indagados acerca da relação entre as atividades de criação de jogos e os desafios vindouros (mercado de trabalho e futuros estudos) que enfrentarão, 8 alunos colocaram suas respostas no nível 3 (não concorda, nem discorda), e 6 alunos responderam que concordam parcialmente (nível 4). Essas respostas corroboram com a nossa visão acerca dessa metodologia, bem como concordam com o nosso propósito na elaboração deste guia. O ponto que queremos frisar é: este guia de ensino de programação tem como objetivo auxiliar o docente no despertar do interesse dos alunos ao aprendizado introdutório de programação utilizando uma temática que está próxima da realidade dos alunos e valorizando artefatos tangíveis (jogos) frutos da própria produção

discente. O uso exclusivo da temática de jogos em boa parte das disciplinas de um curso é inerente aos cursos específicos de produção de jogos. Utilizar o desenvolvimento de jogos durante todo um curso de TI pode ser prejudicial à formação do profissional, uma vez que este ingressou no curso buscando um preparo para se posicionar no mercado de trabalho. É importante que a formação desse profissional contemple – além do desenvolvimento de jogos – tópicos, temas, assuntos e demandas que ele defrontará futuramente no mercado de trabalho, ou em estudos posteriores. Diante disso, concordamos com as respostas expostas no Gráfico 9, pois o uso da temática de desenvolvimento de jogos é apropriado para o aprendizado inicial de programação, contudo não pretende substituir os demais assuntos, conteúdos, projetos e demandas necessários para boa formação e preparo de um profissional de TI.

De posse destas informações, faremos uma comparação entre as opiniões obtidas no questionário inicial com as coletadas do questionário final. Inicialmente, plotamos no mesmo gráfico as respostas das questões sobre quais atividades (tradicionais ou jogos) são adequadas para aprender programação. Vejamos o Gráfico 10.

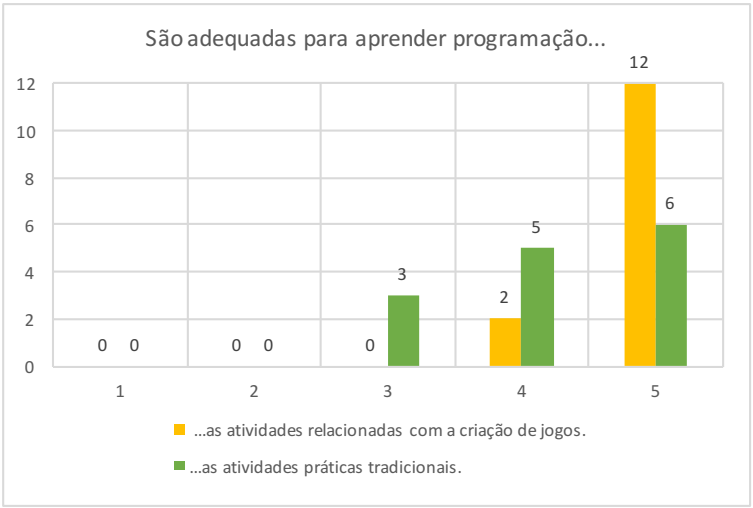

Gráfico 10: Opiniões sobre atividades adequadas para aprender programação
Fonte: Resultados estatísticos dos questionários.

O Gráfico 10 nos apresenta, dispostas lado a lado, as respostas do questionário inicial (atividades práticas tradicionais), bem como as

respostas do questionário final (atividades relacionadas com a criação de jogos). Podemos perceber, a partir dos resultados expostos no Gráfico 10, que esta turma em particular considera as práticas de desenvolvimento de jogos mais adequadas ao aprendizado de programação do que as atividades tradicionais. Tal fato é percebido quando analisamos que antes da implantação da abordagem as respostas se dividiam entre os níveis 3, 4 e 5, e após o contato deles com o desenvolvimento de jogos, as respostas então passaram a se concentrar nos níveis 4 e 5, com forte tendência para o último nível. Esta abordagem de desenvolvimento de jogos pode motivar os alunos a aprender programação, enquanto produzem artefatos (jogos) tangíveis e significativos, diferentemente das produções abstratas oriundas das atividades práticas tradicionais.

Continuado as comparações entre as opiniões obtidas no questionário inicial com as coletadas do questionário final, em seguida, plotamos no mesmo gráfico as respostas das questões sobre quais atividades (tradicionais ou jogos) seriam compatíveis com os desafios que os alunos esperam enfrentar ao concluir o curso. Vejamos o Gráfico 11.

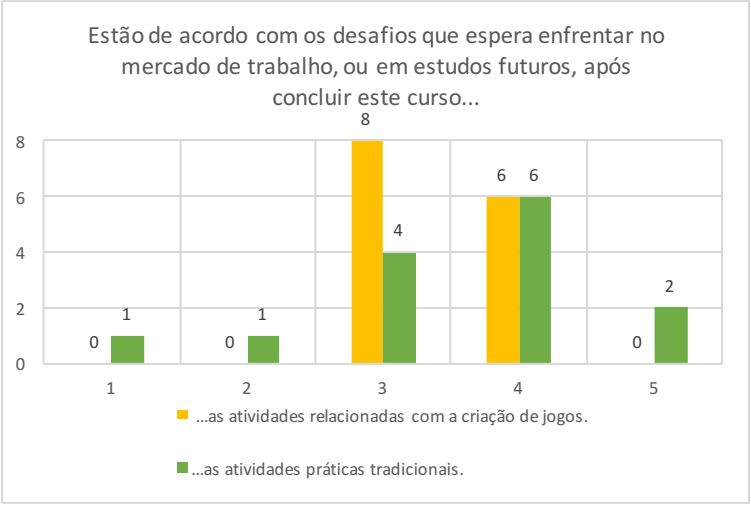

Gráfico 11: Opiniões sobre atividades compatíveis com desafios que esperam enfrentar após conclusão do curso
Fonte: Resultados estatísticos dos questionários.

O Gráfico 11 nos apresenta, dispostas lado a lado, as respostas do questionário inicial (atividades práticas tradicionais), bem como as respostas do questionário final (atividades relacionadas com a criação de

jogos). As opiniões da turma sobre as atividades tradicionais se distribuem em todo o espectro do gráfico, variando desde o "totalmente não apropriado" até o "totalmente apropriado", enquanto as opiniões sobre as atividades com jogos se concentram no meio do gráfico no nível 3 (não concorda, nem discorda) e 4 (concorda parcialmente). Esta turma tem o entendimento de que as demandas e desafios que os aguardam após a conclusão do curso vão além da produção de jogos. Ou seja, os participantes entendem que a criação de jogos pode ser útil no aprendizado de programação, contudo compreendem também que a sua formação profissional precisa contemplar mais tópicos, assuntos e temas além dos jogos e das atividades tradicionais com as quais tiveram contato.

Em linhas gerais, essas são as percepções dos estudantes acerca das atividades práticas após a implementação da metodologia de criação de jogos. Além das breves análises, comparamos essas opiniões com as impressões dos participantes relatadas no questionário inicial que investigou o cenário antes aplicação da metodologia de desenvolvimento de jogos.

Mais que do apenas números em alguns gráficos, interessou-nos ouvir a voz dos participantes a respeito de suas preferências entre as atividades tradicionais e o desenvolvimento de jogos. Por isso, apresentamos aos participantes a seguinte pergunta do tipo aberta: *"Qual tipo de atividades práticas lhe interessou mais? Tradicionais ou Criação de Jogos? Justifique se possível"*. Os comentários e opiniões dos participantes são elencados abaixo.

> Jogos, pois o mesmo tem um grande leque de opções (A1).
> Criação de jogos, o interesse e a motivação é maior (A2).
> Criação de jogos. Pois através disso é possível aplicar o conhecimento de programação de maneira divertida (A3).
> Criação de jogos, porque sai do padrão e isso instiga a criatividade (A4).
> Criação de jogos é divertido e não é repetitivo como os sistemas de CRUD[4] que os professores pedem 'pra' fazer (A5).
> Criação de jogos, por ter uma resposta mais rápida e visualizável (A6).

[4] CRUD (sigla para *Create*, *Read*, *Update* e *Delete* em inglês) para as quatro operações básicas utilizadas em aplicativos e interfaces de usuários para criação, consulta, atualização e remoção de dados.

> Criação *[de jogos]*, pois estimula a criatividade (A7).
> Criação de jogos, porque a gente pode fazer os nossos próprios jogos para jogar (A8).
> Criação de jogos é melhor pois eu me diverti mais (A9).
> Criação de jogos é mais interessante e faz a gente pensar mais, usar a criatividade (A10).
> Criação de jogos é mais criativo na minha opinião (A11).
> Criação de jogos motiva a gente a aprender mais (A12).
> Jogos porque incentiva a procurar mais assunto e motiva a ir 'pra' escola (A13).
> Criação de jogos é mais fácil visualizar o que cada código faz (A14).

Todos os alunos da turma afirmaram que o tipo de atividades de programação que lhes interessou mais foram as práticas de criação de jogos. As justificativas são diversificadas, alguns apontam o fator entretenimento como razão para sua preferência (A3, A5, A8, A9), outros alegam que o aspecto da criatividade é importante (A4, A7, A10, A11), há aqueles que se sentem mais motivados com essa experiência (A2, A12, A13), e também aqueles que destacam outras qualidades nos jogos, como maior variedade de opções, respostas rápidas e de fácil visualização (A1, A6, A14). As opiniões desses alunos confirmam nosso ponto de vista a respeito do uso da temática de jogos em uma disciplina de programação inicial. Ou seja, defendemos que os jogos podem motivar os alunos a estudar mais, pois possuem um alto grau de entretenimento, despertam a criatividade, ampliam as opções de produções de artefatos pelos próprios alunos, além de fornecerem respostas ágeis e de fácil visualização.

Para finalizar o questionário final, solicitamos aos participantes que registrassem suas sugestões para tornar as atividades práticas de criação de jogos mais ricas em termos de aprendizagem e interesse. Vejamos as sugestões apresentadas pelos alunos.

> Mostrar como que funciona as *engines* mais utilizadas 'pra' jogos (A1).
> Bons exemplos antes dos exercícios e maior liberdade em trabalhos nos trabalhos de desenvolver jogos mais complexos (A2).
> Não tenho sugestões (A3).
> Outro ambiente, e mais trabalhos (A4).

> Criar projetos maiores, que envolvam vários alunos em cada grupo (ou em apenas 1). Cada aluno será responsável por implementar determinados recursos no jogo. Além de se tornar um trabalho interdisciplinar, irá melhorar o trabalho em equipe, a competitividade, diversão, criatividade e mostrará ao aluno que fazer um simples jogo não é tão fácil e que precisa de muitas pessoas. Mas para isso acontecer, deve ser necessário pensar em regras para que cada aluno trabalhe igualmente, sem a possibilidade de gerar discussões (A5).
> Utilizar inicialmente plataformas simples para um resultado rápido, e após um período passar para plataformas mais completas 'pra' se ter resultados mais elaborados e profissionais (A6).
> Gostaria de ser mais avançado, mas com o tempo que é disponível, acho difícil ser mais avançado (A7).
> Podia mostrar mais coisas, usar o *joystick* por exemplo (A8).
> Se tivesse mais trabalhos para fazer mais jogos, ia ser legal (A9).
> Sem sugestões (A10).
> Acho que devia falar mais de jogos para celular e para videogames (A11).
> Gostei do jeito que foi, mas dava para ter avançado mais. Faltou tempo (A12).
> Usar mais outros ambientes de programação de jogos (A13).
> Colocar mais trabalhos práticos, tipo mostrar um jogo e pedir para fazer um 'igual mas diferente' [sic]. E assim ia ter mais jeito de fazer mais coisa com os jogos tipo colocar 'pra' vender (A14).

Os participantes A3 e A10 não apresentaram sugestões. Ao analisar as sugestões que foram registradas, percebemos que esta turma não apenas aprovou a temática de programação de jogos, como também ansiava por mais conteúdo (A1, A4, A5, A7, A8, A11, A13 e A14), e também por mais trabalhos e atividades mais complexas (A2, A4, A5, A9, A12 e A14). Interessante observar, no entanto, que o conteúdo ministrado não foi superficial, mas acreditamos que o tema despertou o interesse dos alunos em aprender mais, em conhecer mais sobre programação, em ter acesso a um conteúdo mais profundo sobre desenvolvimento de jogos. Porém, essas sugestões devem ser analisadas de forma crítica. Não

podemos esquecer de que esta é a unidade curricular de programação inicial, na qual os alunos estão aprendendo os conceitos básicos de programação. O ponto que queremos ressaltar é que esta não é uma disciplina de desenvolvimento de jogos, mas sim uma disciplina de programação introdutória e os objetivos elencados na ementa devem ser respeitados, ou seja, os alunos devem saber os conceitos básicos de programação ao concluírem esta matéria. O docente deve evitar privilegiar os conteúdos específicos dos jogos (sons, animações, *joysticks*, vídeos, *engines*) em detrimento dos conteúdos específicos de programação (variáveis, condicionais, repetição, estruturas de dados).

Uma outra sugestão apresentada refere-se ao ritmo da condução da disciplina, A6 sugere que o conteúdo seja ministrado de forma simples inicialmente e posteriormente progrida para algo mais avançado e complexo. Esta sugestão é interessante e pode ser considerada nas próximas implementações desta metodologia, pois facilitaria o entendimento e a assimilação de conteúdos para os alunos.

Como salientado anteriormente, essas são as sugestões apresentadas pelos participantes para tornar as atividades práticas de criação de jogos mais ricas em termos de aprendizagem e interesse. Boa parte das sugestões dos alunos referem-se ao interesse dos mesmos em aprofundar nos estudos sobre desenvolvimento de jogos, entretanto o docente deve ter cautela para não priorizar os aspectos específicos dos jogos em detrimento dos conceitos de programação estipulados na ementa da disciplina.

Apresentamos aqui as análises e discussões sobre as respostas registradas no questionário final proposto aos participantes após a implementação da abordagem de desenvolvimento de jogos na disciplina de programação. Em síntese, podemos perceber que os participantes da turma consideram a temática de desenvolvimento de jogos apropriada para aprender programação até mesmo almejando a ministração de conteúdo adicional e atividades extras, contudo entendem que para se posicionarem no mercado de trabalho, ou ingressarem em estudos posteriores, necessitam ter um contato com outros temas, tópicos e demandas além dos jogos.

3.2.3 Percepções e opiniões do docente colaborador

Além dos alunos, é essencial também ouvir as opiniões e considerações do docente colaborador que similarmente registrou suas impressões acerca do experimento.

Para fins de informação, traçamos um breve perfil do docente colaborador. Atualmente o colaborador é docente responsável pela

disciplina de programação no curso superior em Análise de Sistemas do Instituto Federal do Triângulo Mineiro – campus Ituiutaba. Em relação à sua formação acadêmica, o mesmo possui uma graduação em Engenharia da Computação e mestrado em Engenharia Elétrica com ênfase em Computação Gráfica. Sobre seu tempo como docente, ele acumula cinco anos de experiência em docência na área de TI.

O questionário destinado ao docente foi composto de quatro questões do tipo aberta, com o objetivo de coletar suas opiniões sobre o interesse dos alunos, sugestões para superação de dificuldades na implementação da metodologia, aquisição de conhecimento pelos alunos, e possibilidades de utilização da metodologia em futuras oportunidades.

Na primeira pergunta ao docente, procuramos identificar alterações referentes ao interesse e motivação dos alunos nas aulas de programação com a temática sobre desenvolvimento de jogos.

Pergunta 1: Verificou diferença no interesse, ou motivação, dos alunos ao propor as atividades de criação de jogos?

Resposta 1: *Sim, eles se sentem bem mais motivados.*

O aumento do interesse e motivação pode ser justificado, sobretudo, pelo potencial inerente dos jogos para impressionar, engajar e motivar os jovens. Contudo, para atingir esses resultados, é essencial que o processo de planejamento da inserção da temática no contexto educativo seja desempenhado com atenção, considerando os objetivos e conteúdos da disciplina, estratégias e resultados previstos.

Com o objetivo de ouvir as opiniões do colaborador, na segunda questão solicitamos ao docente que apresentasse suas sugestões para possíveis melhorias e aperfeiçoamentos nesta metodologia.

P. 2: Quais seriam suas sugestões para superar as barreiras e/ou dificuldades com o intuito de promover melhorias nesta metodologia de ensino de programação com criação de jogos?

R. 2: *Aplicar conceitos de gamificação*[5] *em outras disciplinas.*

Os jogos e seus fundamentos em geral oferecem momentos de entretenimento, bem como situações desafiantes e cenários envolventes. Além disso, instigam o interesse dos jogadores ao propor desafios,

[5] Gamificação (do original inglês *gamification*), refere-se ao uso de elementos inerentes aos jogos, porém fora do seu usual contexto de entretenimento, com o intuito de mobilizar os sujeitos à ação, auxiliar na solução de problemas e promover aprendizagens (KAPP, 2012).

problemas, questões e limites, bem como a possibilidade e oportunidade de solucioná-los e superá-los. Tais elementos podem ser aplicados em outras disciplinas, como sugeriu o docente, entretanto mais uma vez ressaltamos a importância do planejamento que envolva identificação de etapas e elementos fundamentais do processo.

Na terceira pergunta, indagamos ao docente sobre sua própria reflexão a respeito da aprendizagem e aquisição de conhecimento pelos alunos. Nosso propósito com essa questão é conhecer se, na opinião do colaborador, os alunos realmente alcançaram os objetivos educacionais da disciplina, ou seja, se eles aprenderam os conceitos de programação como previsto.

P. 3: Considerando a disciplina em andamento e a atividade desenvolvida com a criação de jogos, como você (docente) avalia a aprendizagem e a aquisição/apreensão de conhecimento pelos alunos?

R. 3: *Considero que temos um aumento considerável, pois o interesse em implementar recursos ainda não conhecidos os motiva a procurar mais sobre o assunto e, consequentemente, aprender mais.*

O objetivo da metodologia é motivar, aumentar o interesse e o envolvimento dos alunos na disciplina de programação, porém sem prejudicar o aprendizado. A verificação de aprendizado não é realizada somente nas notas de provas e trabalhos dos alunos, mas, sobretudo, na participação e envolvimento dos discentes nas atividades propostas, como pode ser visto nessa resposta (R. 3), em que o docente percebeu um aumento no interesse dos alunos que buscaram mais informações sobre o tema e assim aprenderam mais, conforme afirmou o professor colaborador.

Para saber da possibilidade de futuras implementações desta metodologia, perguntamos ao docente, na quarta pergunta, sobre seus projetos de educação continuada e suas expectativas de utilização desta abordagem em futuras disciplinas.

P. 4: Concernente aos seus projetos de educação continuada como docente, você considera promissora a aplicação desta metodologia da criação de jogos em suas disciplinas futuramente?

R. 4: *Sim, principalmente pela interação mais dinâmica com o software/jogo desenvolvido e o trabalho com gráficos, sons e imagens, fugindo do convencional caixas de texto e botões.*

Os alunos participantes desta pesquisa são jovens que estão cercados por uma ampla utilização de linguagem iconográfica e, às vezes, há até uma priorização das imagens em relação aos textos, seja dentro ou fora de suas redes sociais de interação. Diante desse peculiar cenário, a motivação e o estímulo são, por vezes, essenciais e tais fatores podem ser apoiados pela implementação de novas práticas pedagógicas pelo docente. Além disso, o jovem aluno também procura manter-se atualizado em relação a novas tendências tecnológicas, sendo assim, nessas novas práticas, é importante considerar como apresentar os conteúdos da disciplina ao estudante de modo eficiente.

Por fim, na quinta pergunta solicitamos ao docente que registrasse seus comentários acerca dos artefatos (jogos) produzidos pelos alunos durante a implementação da metodologia.

P. 5: Quais comentários você pode fazer sobre a produção de jogos pelos alunos durante a disciplina? São jogos simples ou mais elaborados? Poderiam ser colocados no mercado de jogos? Quanto à originalidade são jogos originais (ideia do aluno), ou são adaptados (jogos já existentes que os alunos refizeram)?

R. 5: *Os jogos produzidos são simples devido ao tempo limitado para o seu desenvolvimento, que é de apenas um semestre. Dificilmente poderiam ir para o mercado, pois apesar de interessante, falta refinamento, mais uma vez causado pelo pouco tempo disponível. Apesar disso, dois jogos produzidos por alunos destas turmas venceram a Feira de Novos Produtos*[6]*, o que demonstra que, na opinião dos avaliadores, os jogos possuíam qualidade o suficiente para ir para o mercado (mais uma vez, reafirmo, depois de um refinamento). Quanto à originalidade, creio que 100% dos jogos desenvolvidos foram baseados em algum jogo que os alunos já jogaram alguma vez.*

Em relação à complexidade dos jogos, o fator tempo é um limitador para produção de jogos mais complexos e elaborados, por isso indicamos que na escolha dos jogos (fase de planejamento) opte-se por artefatos mais simples, como fez o docente colaborador neste experimento. Em relação à comercialização dos jogos, o tempo para desenvolvimento aparece novamente como fator limitante, entretanto duas produções fruto deste

[6] Feira de Novos Produtos é um evento organizado anualmente pelo Instituto Federal do Triângulo Mineiro (IFTM) e tem como objetivo oportunizar a divulgação dos produtos, processos e serviços desenvolvidos pelos estudantes da instituição, em diferentes áreas do conhecimento.

experimento granjearam prêmios em uma Feira de Novos Produtos da própria instituição, o que segundo o docente "demonstra que, na opinião dos avaliadores, os jogos possuíam qualidade o suficiente para ir para o mercado" desde que passassem por um aprimoramento.

Em relação à originalidade dos jogos, todas as produções foram baseadas em jogos já conhecidos previamente pelos alunos. Este fato corrobora com o que sugerimos na seção 2.6.2, na qual sugerimos ao interessado neste guia escolher jogos já conhecidos pelos alunos pois demanda menos tempo de esclarecimento e entendimento das regras. Porém, isto não é um empecilho para os estudantes inventem os seus próprios jogos, principalmente quando estes forem variações de jogos já existentes.

Apresentamos aqui as perguntas feitas ao docente colaborador bem como as suas respostas após a implementação da abordagem de desenvolvimento de jogos na disciplina de programação. Em síntese, entendemos que do ponto de vista deste docente, essa turma se encontrou mais motivada e interessada nas aulas temáticas de criação de jogos, do que nas aulas convencionais de programação. Além disso, inferimos também que na opinião deste professor seus alunos aprenderam mais, pois diante do maior interesse na disciplina passaram a buscar mais conteúdos referentes à programação, especialmente, de jogos. Reconhecemos também que, à luz de seus comentários e *feedbacks*, este professor tem expectativas de utilizar esta metodologia futuramente em outras oportunidades. E sobre a análise da produção discente, os artefatos produzidos são jogos simples, não originais – adaptados de jogos já conhecidos – que se aprimorados poderiam, talvez, encontrar lugar no mercado de jogos.

Neste capítulo tratamos de apresentar o experimento, bem como seus resultados, que foi realizado com vistas a buscar informações sobre a implementação deste guia, ouvir as opiniões dos alunos e do professor, e também procurar por pontos falhos e consequentes melhorias e aperfeiçoamentos para implementações futuras. Encerramos aqui as análises e discussões dos resultados que obtivemos por meio dos questionários (aos discentes e ao docente) e diálogos com o docente colaborador. Nesta seção 3.2 divulgamos todas as respostas e *feedbacks* dos participantes, não omitindo nenhum registro, de modo que a transcrição está integralmente presente nesta seção e não vislumbramos a necessidade de repetir tais apontamentos na seção de Apêndices desta tese. No próximo capítulo apresentamos as considerações gerais acerca da metodologia diante de uma visão geral do texto desta tese.

CONSIDERAÇÕES FINAIS E ESTUDOS FUTUROS

A dinâmica evolução do campo da TI traz também desafios educacionais e pedagógicos. Esses desafios incluem questões relacionadas aos alunos de informática, tais como engajamento, motivação e apoio ao contínuo desenvolvimento educacional do discente. Além disso, o contexto pedagógico e instrucional dos materiais de ensino e aprendizagem também demandam atenção.

Nos últimos anos, a importância da programação tornou-se um tema de crescente consciência internacional, passando de um curto domínio dominado por poucos especialistas para um mundo mais amplo, incluindo crianças e adolescentes. Muitas iniciativas têm surgido com o objetivo de inserir o ensino de programação de computadores nas escolas cada vez mais cedo. Fica claro que a habilidade de codificar está assumindo um nível de importância sem precedentes.

Dessa forma, é preciso ter em mente que a programação não se tornou popular por acidente. Há uma crescente compreensão de que saber programar é essencial, especialmente para as gerações mais jovens. Entretanto, os métodos tradicionais de ensino de programação não têm motivado propriamente esse público-alvo mais jovem. Além disso, em diversas ocasiões registradas na literatura, a unidade curricular de programação tem sido o principal motivo de evasão dos cursos de TI. Diante desse fato, vemos que as pesquisas relacionadas a novos métodos de ensino de programação e novas ferramentas de apoio ao ensino de codificação tem constantemente ganhado adeptos no Brasil e no exterior.

Muitas das iniciativas para motivar e engajar jovens na aprendizagem de programação consistem no uso de ambientes virtuais de codificação em blocos. Essas são ferramentas de *software* que se utilizam de imagens que se encaixam umas às outras, como um quebra-cabeças, para criação de pequenos programas, jogos e estórias virtuais. Esses ambientes de programação têm se mostrado eficazes no princípio do aprendizado, porém as suas diversas limitações de recursos de programação os impossibilitam de construir aplicações industriais. O mercado de trabalho demanda cada vez mais *softwares* e aplicações de alta complexidade e robustez, e o desenvolvimento dessas aplicações só é possível com uso de linguagens de programação tradicionais que possuem um maior conjunto de recursos e funcionalidades do que os ambientes gráficos.

A motivação maior desta tese é enfrentar o desafio de engajar e preparar futuros profissionais de TI para o mercado de trabalho e suas reais demandas. Para isto, apropriamo-nos das ideias cativantes e atraentes dos jogos e os efeitos que eles causam nas gerações mais jovens,

juntamente com os conceitos da abordagem pedagógica construcionista. Harmonizar os jogos em uma perspectiva construcionista resultou na concepção de um guia para ensino de programação baseado na construção de jogos digitais utilizando linguagens de programação convencionais.

Inicialmente foi proposto uma versão inicial do guia para ensino de programação baseado na construção de jogos. Esse protótipo foi desenvolvido baseado na revisão de literatura inerentes a outros modelos pedagógicos baseados no desenvolvimento de jogos em outras áreas da educação. Após a concepção do modelo inicial, esse passou por uma avaliação de especialistas na área de Educação em TI que contribuíram com suas análises e considerações para refinamento do protótipo.

O guia otimizado pode ser usado como um modelo de ensino para disciplinas iniciais de programação. Considerando situações gerais de ensino de programação, o guia apresenta um quadro conceitual, com suporte de diretrizes de implementação detalhadas em fases. Seu modelo estrutural, composto por quatro fases, possibilita a implementação imediata de suas ideias em uma unidade curricular de programação.

A Fase 1 do guia consiste na Auditoria Preliminar dos documentos institucionais, infraestrutura disponível e possibilidade da aplicação de um padrão de codificação e estilos. A finalidade principal é evitar a violação de regras da instituição e manter a integridade dos objetivos da disciplina.

A Fase 2 do modelo compreende a etapa de Planejamento e Projetos. Nessa fase, o docente prepara as suas aulas de acordo com o conteúdo programático determinado na ementa, e também realiza a seleção dos jogos que serão utilizados na elaboração das atividades práticas da disciplina.

A Fase 3 do guia contempla o momento do Ensino e Aprendizagem. Durante as aulas, é essencial que o docente se utilize de exemplos de criação de jogos em cada assunto ensinado, e também acompanhe o progresso dos discentes durante o processo de ensino aprendizagem. Nesta fase, sugerimos o uso da avaliação baseada em portfólios para avaliação dos trabalhos e apresentamos algumas sugestões de tipos de questões para as provas e testes que integram a nota final do discente.

A Fase 4 do modelo aborda as Reflexões e *Feedbacks* dos alunos, bem como do docente e da instituição. Consiste de uma importante etapa do modelo pois servirá como base para possíveis readequações de práticas e condutas em uma próxima implementação do guia.

Após a otimização deste guia, oportunizada pelas contribuições dos especialistas, colocamos em prática um experimento no qual o guia já

otimizado foi implementado em uma disciplina de programação. Nessa implementação prática do guia, um professor de programação do Instituto Federal do Triângulo Mineiro – campus Ituiutaba – fez uso da metodologia de criação de jogos em sua disciplina de programação. Os resultados do experimento foram colhidos por meio de questionários aplicados aos alunos e ao professor, que por sua vez nos mostram que a turma se interessou mais pelo desenvolvimento de jogos do que pelas atividades convencionais de programação. Também mostrou que, na opinião dos discentes, essa abordagem é adequada para o aprendizado inicial de programação, contudo não é tão apropriada para preparar completamente estes futuros profissionais para os desafios e demandas que eles esperam encontrar no mercado de trabalho, ou posteriores estudos, após concluírem o curso.

O docente colaborador, que implementou este guia em uma turma de programação, demonstrou em suas opiniões que a turma se mostrou mais motivada e interessada nas aulas em que desenvolviam os jogos quando comparadas às aulas convencionais. De acordo com comentários do docente, seus alunos aprenderam mais, já que diante de maior simpatia pela disciplina, os discentes procuraram mais informações sobre programação. Nos comentários do colaborador, este declarou ter expectativas de usar esta metodologia futuramente em outras disciplinas, e analisou as produções discentes durante o experimento como jogos simples (não complexos), adaptados (não originais) e que se aprimorados encontrariam posicionamento no mercado de jogos.

Ao leitor que tenha interesse de implementar este guia em suas aulas, sugerimos que inicialmente tente uma abordagem simples, escolhendo um tópico para tematizar com jogos e, posteriormente, aprimore a implementação até abordar todos os assuntos da disciplina (se julgar conveniente).

No que diz respeito aos estudos futuros frutos desta tese, entendemos que ainda é necessário investigar se este guia pode ser adaptado e implantado em outras disciplinas dos cursos de TI que não tenham como foco a programação. Como exemplo, as disciplinas de Engenharia de *Software* que cuidam da produção de um *software* sem se ater às especificidades dos conceitos de programação. Outro esforço futuro consiste em uma investigação que busque perceber se este guia pode ser implantado em disciplinas avançadas de programação, como por exemplo Programação Orientada a Objetos (POO), Programação Funcional, Inteligência Artificial, Banco de Dados, dentre outras. Quais seriam as adaptações necessárias para implantação adequada deste guia em outras situações? Em disciplinas que tenham como foco a

programação (intermediária ou avançada; para computadores, para *web*, ou para dispositivos móveis), quais seriam as adequações exigidas para implementação do modelo?

No futuro próximo, os dispositivos computacionalmente aprimorados serão onipresentes, difundidos e conectados em rede entre si. Podemos ter praticamente certeza disso, porém ainda não sabemos (mas é criticamente importante) como as pessoas usarão e pensarão sobre esses dispositivos. Será que algumas pessoas se tornarão aptos a programarem esses dispositivos, usando-os para explorar o mundo ao seu redor e expressar-se de novas maneiras, ao mesmo passo que outras pessoas utilizarão apenas para jogar, assistir vídeos e fazer compras *online*?

Em suma, para apoiar a construção do conhecimento profissional dos discentes de TI, propomos que esses alunos tenham um ambiente de aprendizagem que apoie essa complexa construção intelectual. Portanto, defendemos que o aprendizado ativo, construcionista, motivador, é bastante adequado neste caso.

REFERÊNCIAS

ABES. Mercado Brasileiro de Software - Panoramas e Tendências. Relatório técnico. Associação Brasileira das Empresas de Software (ABES). http://www.abes.org.br. 2012.

ABREU, C. N. D.; KARAM, R. G.; GÓES, D. S.; SPRITZER, D. T. (2008). Dependência de Internet e de jogos eletrônicos: uma revisão. **Revista Brasileira de Psiquiatria**, 30(2), 156-167.

ALMEIDA, M. E. B.; SILVA, M. G. M. Currículo, tecnologia e cultura digital: espaços e tempos de *Web* currículo. **Revista e-Curriculum**, São Paulo, v. 7, n.1, Abril, 2011. Disponível em: http://revistas.pucsp.br/index.php/curriculum/article/viewFile/5676/4002 Acesso em: 14 mar. 2017.

ANDERSON, R. E.; ERNST, M. D.; ORDÓÑEZ, R.; Pham, P. and WOLFMAN, S. A. (2014). Introductory programming meets the real world: using real problems and data in CS1. In Proceedings of the 45th ACM Technical Symposium on Computer Science Education (pp. 465-466). ACM.

AURELIANO, V. C. O.; TEDESCO, P. C. A. R. Ensino-aprendizagem de Programação para Iniciantes: uma Revisão Sistemática da Literatura focada no SBIE e WIE. In: **Anais do Simpósio Brasileiro de Informática na Educação**. 2012.

AVEDON, E. M.; SUTTON-SMITH, B. (1971). **The study of games**. John Wiley & Sons.

BARDIN, L. **Análise de Conteúdo**. Persona Edições, Lisboa, 1979.

BATISTA, A. L. F.; BAZZO, W. A. Questões contemporâneas e desenvolvimento de aplicativos móveis: onde está a conexão? **Revista Brasileira de Ensino de Ciência e Tecnologia,** Curitiba/PR, v. 8, n. 4, p.27-38, 29 dez. 2015. Universidade Tecnológica Federal do Paraná (UTFPR).

BATISTA, A. L. F.; CONNOLLY, T. M.; ANGOTTI, J. A. P. A Framework for Games-Based Construction Learning: A Text-Based Programming Languages Approach. **European Conference on Games**

Based Learning; pp. 815-823. Academic Conferences International Limited. 2016.

BAYLISS, J. D. Using games in introductory courses: tips from the trenches. **In Proceedings of the 40th ACM technical symposium on Computer science education** (SIGCSE'09). ACM, New York, NY, 337-341. 2009.

BERMINGHAM, S.; CHARLIER, N.; DAGNINO, F.; DUGGAN, J.; EARP, J.; KIILI, K.; LUTS, E.; VAN DER STOCK, L. and WHITTON, N. (2013). Approaches to collaborative game-making for fostering 21st century skills. In European Conference on Games Based Learning (pp. 45-52). Academic Conferences International Limited.

BLIKSTEIN, P.; KRANNICH, D. The makers' movement and FabLabs in education: experiences, technologies, and research. In: **Proceedings of the 12th International Conference on Interaction Design and Children**. ACM, 2013. p. 613-616.

BOURGONJON, J.; De GROVE, F.; De SMET, C.; VAN LOOY, J.; SOETAERT, R.; VALCKE, M. (2013). Acceptance of game-based learning by secondary school teachers. **Computers & Education**, 67, 21-35.

BRASSCOM. Índice Brasscom de Convergência Digital - IBCD. Relatório técnico. Associação Brasileira das Empresas de Tecnologia da Informação e Comunicação (BRASSCOM). http://www.brasscom.org.br. 2012

CABRAL, M. I. C. et al. A Trajetória dos Cursos de Graduação da Área de Computação e Informática: 1969-2006. Rio de Janeiro: Editora SBC 2008. 136 p.

CAILLOIS, R. (1961). **Man, play, and games**. University of Illinois Press.

CARMO, H.; FERREIRA, M. **Metodologia da investigação: guia para a auto-aprendizagem.** Lisboa: Universidade Aberta, 2008.

CARR, N. G. A grande mudança. Reconectando o mundo, de Thomas Edison ao Google. Editora Landscape. São Paulo, 2008.

CARR, N. G. Será que TI é tudo? Repensando o papel da tecnologia da informação. Editora Gente. São Paulo, 2009.

CASTELLS, M. **Flows, Networks, Identities: a critical theory of the information society**, in M. Castells, R. Flecha, P. Freire, H. A. Giroux, D. Macedo & P. Willis (Eds) Critical Education in the New Information Age. Lanham: Rowman & Littlefield. 1999.

CASTELLS, M. **The Information Age: economy, society and culture**. Vol. 1. The Rise of the Network Society. Cambridge, MA: Blackwell. 1996.

CHANCE, P. (2013). **Learning and behavior**. Nelson Education.

CONNOLLY, T.; BOYLE, E. A.; MACARTHUR, E.; HAINEY, T.; BOYLE, J. M. A systematic literature review of empirical evidence on computer games and serious games. **Computers & Education**, 59 (2), 661–686, 2012.

CONNOLLY, T.; BOYLE, L.; HAINEY, T. A survey of students' motivations for playing computer games: A comparative analysis. In: **Proceedings of the 1st European conference on games-based learning (ECGBL)**. p. 71-78. 2007.

CORRAL, J. M. R.; BALCELLS, A. C.; ESTÉVEZ, A. M.; MORENO, G. J.; RAMOS, M. J. F. (2014). A game-based approach to the teaching of object-oriented programming languages. **Computers & Education**, 73, pp. 83-92.

COUTINHO, C. P. Metodologias de investigação em Ciências Humanas. Coimbra: Almedina, 2011.

CRAWFORD, C. (1984). **The art of computer game design**. Osborne/McGraw-Hill.

CRUZ, D. M.; de ALBUQUERQUE, R. M. (2013). Letramento digital através de criação de jogos eletrônicos: ensaio comparativo sobre dois contextos escolares. **Revista NUPEM**, 5(8), 123-143.

DA SILVA, H. M. et al. Uma reflexão sobre o crescente desinteresse e a constante evasão em cursos de computação e informática.

In: **Proceedings of International Conference on Engineering and Technology Education**. p. 166-170. 2013.

DENNER, J.; WERNER, L.; CAMPE, S.; ORTIZ, E. 2014. Using Game Mechanics to Measure What Students Learn from Programming Games. ***Int. J. Game-Based Learn.*** 4, 3 (July 2014), 13-22. 2014.

DEPRADINE, C. A. Using gaming to improve advanced programming skills. **The Caribbean Teaching Scholar**, v. 1, n. 2, 2012.

DIAS, C. M.; BORGES, C. F.; PEREIRA, A. M. "Construção de jogo como dispositivo para a aprendizagem colaborativa: algumas estratégias", p. 255-263. In: **Anais do V Simpósio sobre o Livro Didático de Língua Materna e Língua Estrangeira & do IV Simpósio sobre Materiais e Recursos Didáticos.** Blucher Design Proceedings, v.2, n.6. São Paulo: Blucher, 2016.

DICKSON, P. E. (2010). Experiences building a college video game design course. **Journal of Computing Sciences in Colleges**, 25(6), pp. 104-110.

DOMINGOS, D. C. A.; RECENA, M. C. P. "Elaboração de jogos didáticos no processo de ensino e aprendizagem de química: a construção do conhecimento." **Ciências & cognição** 15.1 (2010): 272-281.

DORN, B.; TEW, A. E. and GUZDIAL, M. (2007). Introductory computing construct use in an end-user programming community. VLHCC, 2007, **Visual Languages and Human-Centric Computing, IEEE Symposium on** 2007, pp. 27-32.

DRAKE, P. and SUNG, K. (2011). Teaching introductory programming with popular board games. In Proceedings of the 42nd ACM Technical Symposium on Computer Science Education (pp. 619-624). ACM.

FEIJÓ, B.; CLUA, E.; DA SILVA, F. S. C. **Introdução à Ciência da Computação com Jogos**. Editora Elsevier. Rio de Janeiro, 2009.

FERRARI, R.; RIBEIRO, M. X.; DIAS, R. L.; FALVO, M. **Estruturas de Dados Com Jogos**. Editora Elsevier. Rio de Janeiro, 2014.

FINCHER, S.; COOPER, S.; KÖLLING, M. and MALONEY, J. (2010). Comparing Alice, Greenfoot & Scratch. In Proceedings of the 41st ACM

Technical Symposium on Computer Science Education (pp. 192-193). ACM.

FORTE, A.; GUZDIAL, M. Motivation and nonmajors in computer science: identifying discrete audiences for introductory courses. **Education, IEEE Transactions** on 48, 248-253. 2005.

FREEMAN, S. et al. Active learning increases student performance in science, engineering, and mathematics. **Proceedings of the National Academy of Sciences**, v. 111, n. 23, p. 8410-8415, 2014.

FREIRE, P. **Pedagogia do oprimido.** 1.ed. Rio de Janeiro: Paz e Terra 1974.

GEE, J. P. **What Video Games Have to Teach Us About Learning and Literacy.** Second Edition: Revised and Updated Edition. 2nd. Palgrave Macmillan, 2007. ISBN 1403984530.

GILAKJANI, A. P; LEONG, L.; ISMAIL, H. N. Teachers' use of technology and constructivism. **International Journal of Modern Education and Computer Science (IJMECS)**, v. 5, n. 4, p. 49, 2013.

GONDIM, H. W. A. S.; AMBRÓSIO, A. P.; COSTA, F. M. Uma Experiência no Ensino de Algoritmos utilizando Ambientes Visuais de Programação 3D. In: **XVII Workshop sobre Educação em Computação**, Bento Gonçalves, Brasil, 2009.

GRANT, M. M. Getting a grip on project-based learning: Theory, cases and recommendations. **Meridian: A middle school computer technologies journal**, v. 5, n. 1, p. 83, 2002.

HALVERSON, E. R.; SHERIDAN, K. "The maker movement in education." **Harvard Educational Review** 84.4 (2014): 495-504.

HAZZAN, O.; LAPIDOT, T.; RAGONIS, N. **Guide to Teaching Computer Science:** An Activity-Based Approach. 2. ed. Londres: Springer, 2015. 260 p.

HUIZINGA, J.; HULL, R. F. C. (1949). **Homo Ludens. A Study of the Play-element in Culture**. [Traduzido por RFC Hull.]. Routledge & Kegan Paul.

HWANG, G.; WU, P. Advancements and trends in digital game-based learning research: a review of publications in selected journals from 2001 to 2010. **British Journal of Educational Technology**, v. 43, n. 1, p. E6-E10, 2012.

JUNIOR, G. P.; FECHINE, J. M.; COSTA, E. B. Analogus: Um Ambiente para Auxílio ao Ensino de Programação Orientado pelo Raciocínio por Analogia. In: **XVII Workshop sobre Educação em Computação**, Bento Gonçalves, Brasil, 2009.

JUUL, J. (2011). **Half-real: Video games between real rules and fictional worlds**. MIT press.

KAFAI, Y. B. (2006). Playing and making games for learning instructionist and constructionist perspectives for game studies. Games and Culture, 1(1), pp. 36-40.

KAFAI, Y. B.; BURKE, Q. (2015). Constructionist gaming: Understanding the benefits of making games for learning. Educational Psychologist, 50(4), pp. 313-334.

KAHN, R.; KELLNER, D. Paulo Freire and Ivan Illich: Technology, politics and the reconstruction of education. **Policy Futures in Education**, v. 5, n. 4, p. 431-448, 2007.

KAPP, K. M. **The Gamification of Learning and Instruction: Game-based Methods and Strategies for Training and Education**. John Wiley & Sons. 2012.

KASPERAVICIUS, L. C. C.; et al. "Ensino de desenvolvimento de jogos digitais baseado em metodologias Ágeis: o projeto primeira habilitação." **Anais do XXVIII Congresso da SBC-Workshop sobre Educação em Computação**. 2008.

KULLENBERG, C.; KASPEROWSKI, D. (2016). What is citizen science? – A scientometric meta-analysis. **PloS one**, 11(1), e0147152.

LAMPE, C.; RESNICK, P.; FORTE, A.; YARDI, S.; ROTMAN, D.; MARSHALL, T.; LUTTERS, W. Educational Priorities for Technology-Mediated Social Participation. **Computer 43**, 60-67. 2010.

LEUTENEGGER, S.; EDGINGTON, J. A games-first approach to teaching introductory programming. **In Proceedings of the 38th SIGCSE technical symposium on Computer Science Education** (SIGCSE '07). ACM, New York, NY, 115-118. 2007.

LEWIS, M. C.; MASSINGILL, B. (2006). Graphical game development in CS2: a flexible infrastructure for a semester long project. ACM SIGCSE Bulletin, 38(1), pp. 505-509.

LUDI, S. (2011). The benefits and challenges of using educational game projects in an undergraduate software engineering course. In Proceedings of the 1st International Workshop on Games and Software Engineering (pp. 13-16). ACM.

LÜDKE, M.; ANDRÉ, M. E. D. A. **Pesquisa em Educação: abordagens qualitativas.** São Paulo: Ed. Pedagógica e Universitária - EPU EPU, 1986.

MAJGAARD, G. Creating Games in the Classroom: from native gamers to reflective designers. In: **The 7th European Conference on Games Based Learning**. p. 253-258. 2013.

MALONEY, J., et al. "Scratch: a sneak preview [education]." Creating, connecting and collaborating through computing, 2004. Proceedings. Second International Conference on. IEEE, 2004.

MARTINS, L. C.; LOPES, D. A.; RAABE, A. Um Assistente de Predição de Evasão aplicado a uma disciplina Introdutória do curso de Ciência da Computação. In: **Anais do Simpósio Brasileiro de Informática na Educação**. 2012.

MCGONIGAL, J. (2011). **Reality is broken: Why games make us better and how they can change the world**. Penguin.

MEC (2016). Diretrizes Curriculares Nacionais para os Cursos de Graduação em Computação (DCN16). Disponível em: http://portal.mec.gov.br/index.php?option=com_docman&view=download&alias=52101-rces005-16-pdf&category_slug=novembro-2016-pdf&Itemid=30192. Resolução CNE/CES nº 5, de 16 de novembro de 2016. Último acesso: 27/03/2017.

MOLL, J. Educação Profissional e Tecnológica no Brasil Contemporâneo: desafios, tensões e possibilidades. Artmed, 2010.

MORAN, J. M. **A educação que desejamos:** novos desafios e como chegar lá. 2. ed. Campinas, SP: Papirus, 2007. 174 p.

MORAN, J. M. **Principais diferenciais das escolas mais inovadoras.** 2013. Disponível em: <http://www2.eca.usp.br/moran/wp-content/uploads/2013/12/diferenciais.pdf>. Acesso em: 14 mar. 2017.

MSELLE, L. J.; KONDO, T. S. Against the "Hello World". **International Journal of Computer Applications**, v. 95, 2014.

NASCIMENTO, M. N.; NERY, M. S.; SILVA, V. N. "Desenvolvimento de Jogos Digitais e sua Utilização na Educação Juvenil: Um Estudo de Caso Real em um Projeto Governamental." **Proceedings of the XII Simpósio Brasileiro de Games e Entretenimento Digital (SBGames)**. *São Paulo* (2013).

PAPASTERGIOU, M. Digital Game-Based Learning in high school Computer Science education: Impact on educational effectiveness and student motivation. **Computers & Education** 52, 1-12. 2009.

PAPERT, S. (1991) Preface, In: Harel, I. & Papert, S. (Eds), Constructionism, Research reports and essays, 1985-1990 (p. 1), Norwood NJ.

PAPERT, S. A máquina das crianças: repensando a escola na era digital. Porto Alegre: Artes Médicas, 1994.

PAPERT, S.; HAREL, I. (1991). Situating constructionism. Constructionism. Norwood.

PARALIC, M.; PIETRIKOVA, E. (2014). Learning by game creation in introductory programming course: 5-Year-long study. In Emerging eLearning Technologies and Applications (ICETA), pp. 391-396, 2014 IEEE 12th International.

PERROTTA, C.; FEATHERSTONE, G.; ASTON, H.; HOUGHTON, E. **Game-based learning: latest evidence and future directions**. Slough: NFER. 2013.

PHELPS, A. M.; EGERT, C. A.; BAYLISS, J. D. Games in the classroom: using games as a motivator for studying computing: part 1. - **IEEE MultiMedia**, 2009.

PIAGET, J. Development and learning. In LAVATTELLY, C. S. e STENDLER, F. Reading in child behavior and development. New York: Hartcourt Brace Janovich, 1972.

PIAGET, J. **Epistemologia genética**. Petropolis, RJ: Vozes, 1971. Tradução: Nathanael C. Caixeiro. L'èpistémologie Genétique. Paris: Universitaries de France, 1970.

PIAGET, J. **To understand is to invent.** New York: The Viking Press, Inc. 1972.

PICCOLO, H. L.; SENA, V. F.; NOGUEIRA, K. B.; SILVA, M. O.; Y. A. N. Maia. Ambiente Interativo e Adaptável para ensino de Programação. In: **XXI Simpósio Brasileiro de Informática na Educação**, João Pessoa, Brasil, 2010.

PONTES, H. P. "Desenvolvimento de Jogos no Processo de Aprendizado em Algoritmos e Programação de Computadores." **Proceedings of the XII Simpósio Brasileiro de Games e Entretenimento Digital (SBGames)**. *São Paulo* (2013).

PONTES, T. B.; CASTANHO, C. D. A experiência interdisciplinar na metodologia de desenvolvimento de jogos. **Blucher Design Proceedings** 2.2 (2015): 1118-1130.

PRENSKY, M. Aprendizagem baseada em jogos digitais. São Paulo: Senac, 2012.

PRENSKY, M. Não me atrapalhe, mãe–eu estou aprendendo. São Paulo: Phorte, 2010.

PRICE, T. W.; BARNES, T. "Comparing textual and block interfaces in a novice programming environment." *Proceedings of the eleventh annual International Conference on International Computing Education Research*. ACM, 2015.

PYPL. **PYPL:** PopularitY of Programming Language. 2017. Disponível em: <http://pypl.github.io/PYPL.html>. Acesso em: 3 fev. 2017.

RAABE, A. L. A.; SILVA, J. M. C. Um ambiente para atendimento as dificuldades de aprendizagem de algoritmos". In: **XIII Workshop de Educação em Computação – WEI**, p.2326-2335. 2005.

REBOUÇAS, A. D. D. S.; MARQUES, D. L.; COSTA, L. F. S.; SILVA, M. A. A. Aprendendo a Ensinar Programação Combinando Jogos e Python. In: **XXI Simpósio Brasileiro de Informática na Educação – SBIE**. 2010.

RESNICK, M., et al. "Scratch: programming for all." *Communications of the ACM* 52.11 (2009): 60-67.

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: **XVII Workshop de Informática na Escola**, Aracaju, Brasil, 2011.

ROBINS, A.; ROUNTREE, J.; ROUNTREE, N. Learning and teaching programming: A review and discussion. **Computer Science Education**, v. 13, n. 2, p. 137-172, 2003.

SALEN, K.; ZIMMERMAN, E. (2003). **Rules of play: Game design fundamentals**. MIT press.

SANTOS, E. M.; FREITAS, K. R. As apropriações ocorridas na implementação do portfólio educacional no ensino de ciências físicas e biológicas. **Revista Polyphonía** 25.2 (2015): 325-332.

SAVI, R. **Avaliação de jogos voltados para disseminação do conhecimento.** 2011. 238 f. Tese (Doutorado) - Curso de Pós-graduação em Engenharia e Gestão do Conhecimento, Universidade Federal de Santa Catarina, Florianópolis, 2011. Disponível em: <https://repositorio.ufsc.br/handle/123456789/96046>. Acesso em: 14 mar. 2017.

SBC, SOCIEDADE BRASILEIRA DE COMPUTAÇÃO. **Educação Superior em Computação:** Estatísticas 2015. [s.i.]: SBC, 2016. 52 p.

SCAICO, P. D., et al. "Ensino de programação no ensino médio: Uma abordagem orientada ao design com a linguagem Scratch." **Revista Brasileira de Informática na Educação** 21.02 (2013): 92.

SCHUSTER, D. L. (2010). CS1, arcade games and the free java book. In Proceedings of the 41st ACM Technical Symposium on Computer Science Education (pp. 549-553). ACM.

SHELDON, L. (2011). **The multiplayer classroom: Designing coursework as a game**. Cengage Learning.

SHUTE, V. J.; KE, F. (2012). Games, learning, and assessment. In **Assessment in game-based learning** (pp. 43-58). Springer New York.

SICART, M. (2008). Defining game mechanics. **Game Studies**, 8(2), p. 1-14.

SILVA, R. B.; MERKLE, L. E. Perspectivas educacionais FabLearn: conceitos e práticas maker no Brasil. **FabLearn Conference**. Universidade de São Paulo, Brasil. 2016.

SILVA, T. R., et al. Ensino-aprendizagem de programação: uma revisão sistemática da literatura. **Revista Brasileira de Informática na Educação,** [s.l.], v. 23, n. 01, p.182-196, 30 abr. 2015. Comissão Especial de Informática na Educação. http://dx.doi.org/10.5753/rbie.2015.23.01.182.

SILVA, V. N.; NASCIMENTO, M. N. Investigação da melhoria do aprendizado de alunos do ensino médio da rede pública de ensino através do uso de programação, robótica e jogos digitais. In: **Simpósio Brasileiro de Jogos e Entretenimento Digital**, p.176-179, 2012.

SOARES, A.; FONSECA, F.; MARTIN, N. L. (2015). Teaching introductory programming with game design and problem-based learning. Issues in Information Systems, 16(3), pp. 128-137.

SOARES, R. F.; BORGES, M. A. F. Robótica: aprendizado em informática de forma lúdica. In: **XIX Workshop sobre Educação em Computação**, Natal, Brasil, 2011

SOUZA, P. R. A.; DIAS, L. R. Kodu Game Labs: Estimulando o Raciocínio Lógico através de Jogos. . In: **XXIII Simpósio Brasileiro de Informática na Educação**, Rio de Janeiro, Brasil, 2012.

SUITS, B. (2014). **The Grasshopper: Games, Life and Utopia**. Broadview Press.

SUNG, K. Computer games and traditional CS courses. Communications of the ACM, v. 52, n. 12, p. 74-78, 2009.

SUNG, K.; HILLYARD, C.; ANGOTTI, R. L.; PANITZ, M. W.; GOLDSTEIN, D. S.; NORDLINGER, J. (2011). Game-themed programming assignment modules: a pathway for gradual integration of gaming context into existing introductory programming courses. Education, IEEE Transactions on, 54(3), pp. 416-427.

SUNG, K.; PANITZ, M.; WALLACE, S.; ANDERSON, R.; NORDLINGER, J. Game-themed programming assignments: The faculty perspective. In Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education. ACM Press, NY, 300–304. 2008.

SUNG, K.; SNYDER, L. (2014). A case of computer science principles with traditional text-based programming languages. Journal of Computing Sciences in Colleges, 30(1), pp. 161-172.

SWEIGART, A. (2012). **Making Games with Python & Pygame**. North Charleston: CreateSpace.

TIOBE. **TIOBE Software:** TIOBE Index. 2017. Disponível em: <http://www.tiobe.com/tiobe-index/>. Acesso em: 3 fev. 2017.

UTTING, I.; COOPER, S.; KÖLLING, M.; MALONEY, J. and RESNICK, M. (2010). Alice, Greenfoot, and Scratch - a discussion. ACM Transactions on Computing Education (TOCE), 10(4), pp. 17:1-11.

VALASKI, J.; PARAISO, E. C. Limitações da Utilização do Alice no Ensino de Programação para Alunos de Graduação. In: **XXIII Simpósio Brasileiro de Informática na Educação**, Rio de Janeiro, Brasil, 2012.

VAN STAALDUINEN, J.; DE FREITAS, S. A Game-Based Learning Framework: Linking Game Design and Learning. **Learning to play: exploring the future of education with video games**, v. 53, p. 29, 2011.

VIHAVAINEN, A.; AIRAKSINEN, J.; WATSON, C. A systematic review of approaches for teaching introductory programming and their influence on success. **In Proceedings of the tenth annual conference on International computing education research** (ICER '14). ACM, New York, NY, USA, 19-26. 2014.

WANG, A. I. (2011). Extensive evaluation of using a game project in a software architecture course. ACM Transactions on Computing Education (TOCE), 11(1), pp. 5:1-28.

WEINTROP, D.; WILENSKY, U. (2015). To block or not to block, that is the question: students' perceptions of blocks-based programming. In Proceedings of the 14th International Conference on Interaction Design and Children (pp. 199-208). ACM.

WILSON, A.; HAINEY, T.; CONNOLLY, T. M. Using Scratch with primary school children: an evaluation of games constructed to gauge understanding of programming concepts. **International Journal of Game-Based Learning (IJGBL)**, v. 3, n. 1, p. 93-109, 2013.

ZORZO, A. F.; NUNES, D.; MATOS, E.; STEINMACHER, I.; LEITE, J.; ARAUJO, R. M.; CORREIA, R.; MARTINS, S. “Referenciais de Formação para os Cursos de Graduação em Computação”. Sociedade Brasileira de Computação (SBC). 153p, 2017.

APÊNDICES

APÊNDICE A – QUESTIONÁRIO PROPOSTO AOS ALUNOS ANTES DA APLICAÇÃO DA METODOLOGIA

Questionário 01 (aluno)

1. Qual a sua idade?
2. Quanto tempo você investe por semana em jogos eletrônicos (videogame, celular, computador)?
3. Quanto tempo você dedica aos estudos desta disciplina de programação além das aulas?
4. Indique seu tempo de dedicação em estudo nas demais disciplinas do curso, além das aulas.
5. Como você avalia seu nível de conhecimento em programação de computadores antes de entrar neste curso?
6. Você considera que as atividades práticas e exercícios de programação propostos nesta disciplina atualmente são adequadas para aprender programação?
7. Você considera que as atividades práticas e exercícios de programação propostos nesta disciplina atualmente, estão de acordo com os desafios que você espera enfrentar no mercado de trabalho, ou em futuros cursos, após concluir este curso?
8. Aponte a sua maior dificuldade nas aulas de programação, e justifique se possível.
9. Qual seria a(s) sua(s) sugestão(ões) para tornar as aulas de programação mais ricas em termos de aprendizagem e interesse?

Agradecemos a colaboração e destacamos nosso compromisso com o anonimato dos participantes e confidencialidade dos dados.

Atenciosamente,
André Luiz França Batista – Doutorando
Prof. Dr. José André Peres Angotti – Orientador

APÊNDICE B – QUESTIONÁRIO PROPOSTO AOS ALUNOS APÓS A APLICAÇÃO DA METODOLOGIA

Questionário 02 (aluno)

1. Você acha que as atividades práticas e exercícios de programação relacionados com a criação de jogos são adequadas para aprender programação?
2. Você considera que as atividades práticas e exercícios de programação relacionados com a criação de jogos, estão de acordo com os desafios que você espera enfrentar no mercado de trabalho após concluir o curso?
3. Quais tipos de atividades práticas lhe interessou mais? Tradicionais ou Criação de Jogos? Justifique se possível.
4. Qual seria a(s) sua(s) sugestão(ões) para tornar as atividades práticas de programação vinculadas a criação de jogos mais ricas em termos de aprendizagem e interesse?
5. Seus comentários e sugestões sobre esta prática de criação de jogos nas disciplinas de programação (opcional).

Agradecemos a colaboração e destacamos nosso compromisso com o anonimato dos participantes e confidencialidade dos dados.

Atenciosamente,
André Luiz França Batista – Doutorando
Prof. Dr. José André Peres Angotti – Orientador

APÊNDICE C – QUESTIONÁRIO PROPOSTO AO DOCENTE

Questionário 03 (professor)

1. Verificou diferença no interesse, ou motivação, dos alunos ao propor as atividades de criação de jogos?
2. Quais seriam suas sugestões para superar as barreiras e/ou dificuldades com o intuito de promover melhorias nesta metodologia de ensino de programação com criação de jogos?
3. Considerando a disciplina em andamento e a atividade desenvolvida com a criação de jogos, como você (docente) avalia a aprendizagem e a apreensão de conhecimento pelos alunos?
4. Concernente aos seus projetos de educação continuada como docente, você considera promissora a aplicação desta metodologia da criação de jogos em suas disciplinas futuramente?
5. Quais comentários você pode fazer sobre a produção de jogos pelos alunos durante a disciplina? São jogos simples ou mais elaborados? Poderiam ser colocados no mercado de jogos? Quanto à originalidade são jogos originais (ideia do aluno), ou são adaptados (jogos já existentes que os alunos refizeram)?

Agradecemos a colaboração e destacamos nosso compromisso com o anonimato dos participantes e confidencialidade dos dados.

Atenciosamente,
André Luiz França Batista – Doutorando
Prof. Dr. José André Peres Angotti – Orientador

Guia	Pontos principais	Ações	Demandas e questões
1 Auditoria preliminar Obter uma visão geral do ambiente acadêmico disponível.	Políticas e normas institucionais. Objetivos da disciplina. Infraestrutura disponível. Padrões de codificação e estilos.	Verificar Conferir Checar Averiguar Pesquisar	Verifique quais são as políticas institucionais. Conferir quais são os objetivos da disciplina. Verifique qual é a infraestrutura (física e técnica) disponível. Considere pesquisar sobre o uso de padrões de codificação e estilos.
2 Planejamento e projetos Preparar a disciplina considerando os aspectos pedagógicos da implementação.	Planos de aula detalhados. Escolha dos jogos. Elaboração de atividades.	Planejar Escolher Elaborar Preparar	Planeje as aulas antecipadamente observando o contexto e pedagogia adequados. Escolha os jogos que serão desenvolvidos durante a disciplina. Os próprios alunos podem participar desse momento de escolha. Elabore as atividades práticas, alinhando os conceitos de programação com criação de jogos.
3 Ensino e aprendizagem Ministrar a disciplina colocando em prática a metodologia de desenvolvimento de jogos.	Aulas e tutoriais. Atividades práticas de programação. Avaliação de aprendizagem.	Ensinar Apresentar Orientar Propor Intermediar Avaliar	Apresente os conceitos básicos sobre jogo e sua estrutura básica de desenvolvimento. Proponha as atividades práticas de programação contextualizadas com a temática de criação de jogos. Considere a utilização de bibliotecas gráficas. Avalie a aprendizagem dos alunos por meio de avaliações tradicionais ou portfólios digitais.
4 Reflexões e feedbacks Revisar e analisar o andamento da disciplina buscando pontos de melhorias e ajustes para próxima implementação.	*Feedback* do docente. *Feedback* dos alunos. *Feedback* da instituição.	Perguntar Analisar Revisar Rever Refletir	Em relação ao andamento da disciplina, pergunte: o que deu certo? O que deu errado? Quais foram os pontos positivos e os negativos? Quais pontos precisam de ajustes para uma próxima implementação? Revise os pontos negativos. O que pode ser melhorado? Reflita sobre o aprendizado dos alunos. Os objetivos de aprendizagem propostos inicialmente foram alcançados?