INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO RIO GRANDE DO SUL

CAMPUS PORTO ALEGRE

MESTRADO PROFISSIONAL EM INFORMÁTICA NA EDUCAÇÃO

GUILHERME SCHIRMER DA COSTA

CODINGJOB: UM JOGO SÉRIO PARA AUXILIAR NAS DISCIPLINAS DE LINGUAGEM DE PROGRAMAÇÃO C

PORTO ALEGRE

2023

Guilherme Schirmer da Costa

CodingJob: Um jogo sério para auxiliar nas disciplinas de Linguagem de Programação C

Dissertação apresentada junto ao Mestrado Profissional em Informática na Educação do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul – campus Porto Alegre, como requisito para obtenção do título de Mestre em Informática na Educação.

Orientador: Profa. Dra. Márcia Häfele Islabão Franco
Coorientador: Prof. Dr. Fabio Y. Okuyama

Porto Alegre
2023

C837 Costa, Guilherme Schirmer da

CodingJob: um jogo sério para auxiliar nas disciplinas de Linguagem de Programação C / Guilherme Schirmer da Costa – Porto Alegre, 2023.
84 f. : il., color.

Orientadora: Dra. Márcia Häfele Islabão Franco
Coorientador: Prof. Dr. Fabio Yoshimitsu Okuyama

Dissertação (mestrado) – Instituto Federal do Rio Grande do Sul Campus Porto Alegre, Mestrado Profissional em Informática na Educação, Porto Alegre, 2023.

1. Informática na Educação. 2. Linguagem de Programação. 3. Jogos digitais. 4. Aprendizagem por computador I. Franco, Márcia Häfele Islabão. II. Okuyama, Fabio Yoshimitsu. III. Título.

CDU: 004:37

Elaborada por Débora Cristina Daenecke Albuquerque Moura - CRB10/2229

Dedicado ao Ramon Barbosa, que partiu muito cedo.

AGRADECIMENTOS

Agradeço ao Programa de Pós-Graduação em Informática na Educação do Instituto Federal de Educação, Ciência e Tecnologia (IFRS), Campus Porto Alegre, pela oportunidade da realização do curso de mestrado.

Agradeço à professora Márcia Franco pela orientação e pelo apoio em momentos de dúvida em relação ao projeto e em momentos de dúvidas pessoais durante o período de isolamento social.

Agradeço ao professor Fábio Okuyama pela coorientação e pelo acesso aos alunos das turmas utilizadas na pesquisa.

Aos demais professores do curso pelos ensinamentos passados.

Aos colegas de curso pelo apoio e pelas risadas nos intervalos das aulas.

À minha família e amigos pelo apoio.

À Cristina e à nossa filha por existirem.

RESUMO

Estudos mostram que dentre as disciplinas dos cursos da área de computação, as que envolvem programação são vistas como vitais para o progresso acadêmico e profissional dos alunos. No entanto, essas disciplinas são destacadas em diversas pesquisas como complexas e, geralmente, possuem uma alta taxa de evasão e/ou reprovação. Sabe-se que são vários os fatores que levam a esse problema, desde uma educação básica deficitária em lógica e matemática, falta de interesse nos exercícios, falta de tempo para praticar fora da sala de aula, dificuldades de abstrair sintaxe de uma linguagem de programação dentre outros. Diante deste cenário, acredita-se que o desenvolvimento de uma ferramenta que motive os alunos a praticarem programação fora do horário das aulas e que tenha teor mais lúdico como por exemplo, um jogo digital, possa contribuir com o desenvolvimento da motivação e do engajamento dos alunos. Jogos digitais são utilizados na educação com diferentes níveis de sucesso por décadas. Atualmente, os jogos sérios são conhecidos por unir o lúdico e o conteúdo programático de diversas disciplinas, com a possibilidade de serem utilizados dentro e fora das salas de aula. Esta pesquisa apresenta o jogo sério denominado CodingJob, que objetiva auxiliar os alunos da disciplina de Linguagem de Programação I a praticarem o conteúdo da disciplina em um ambiente que simula uma pequena fábrica de software. Inspirado em conceitos construcionistas, o jogo utiliza características de um micro mundo, conceito desenvolvido por Seymour Papert, no qual o aluno tem um ambiente de estudo provido de ferramentas que permitem que o mesmo explore o conteúdo, produzindo soluções e construindo conhecimento. O jogo foi desenvolvido em ciclos de produção com o auxílio de alunos através da utilização da metodologia Design Science Research, onde um ciclo de produção culmina com testes com usuários que levantam novas sugestões de alterações para um novo ciclo de produção. Ao fim de 3 ciclos de testes, a partir dos resultados apresentados pelos alunos foi possível notar que o resultado em relação a motivação dos alunos. No entanto, os resultados apresentados em relação a aprendizagem através do jogo são inconsistentes, ficando esta questão a ser tratada em investigações futuras.

Palavras-chaves: Jogos Sérios, Linguagem de Programação C, Jogos Digitais

ABSTRACT

Studies show that among the subjects of courses in the area of computing, those that addresses the programming skills are seen as vital to students' academic and professional progress. Nonetheless, these disciplines are highlighted in several researches as complex and, generally, have a high dropout and/or failure rate. It is known that there are several factors that may lead to this problem, from a basic education deficient in logic and mathematics, lack of interest in exercises, lack of time to practice outside the classroom, difficulties in abstracting syntax of a programming language among others. In this scenario, it is believed that the development of a tool that motivates students to practice programming outside of class hours and that have a more playful and less formal content, such as, a digital game, can contribute to the development of motivation and engagement of students. Digital games have been used in education with different levels of success for decades. Currently, serious games are known for uniting the playful and programmatic content of different disciplines, with the possibility of being used inside and outside the classrooms. This work presents the application of the serious game CodingJob, which aims to help students of the Programming Language I course to practice the course content in a environment that simulates a small software factory. Inspired by constructionist concepts, the game aims to use features of a micro world, a concept developed by Seymour Papert, in which the student has a study environment provided with tools that allow the same explore content, producing solutions and building knowledge. The game was developed in production cycles with the help of students through the use of the Design Science Research methodology, where a production cycle culminates with user testing that raises new suggestions for changes for a new production cycle. After 3 test cycles, the results presented were positive, although they raised several questions regarding some of the author's design choices. It is possible to notice that the result in relation to student motivation was positive, however, the results presented in relation to learning through the game are inconsistent, since the difficulty of some questions has created frustration in some students

Key-words: Serious Games, C Programming Language, Digital Games

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
abnTeX	ABsurdas Normas para TeX

LISTA DE SÍMBOLOS

Γ	Letra grega Gama
Λ	Lambda
ζ	Letra grega minúscula zeta
$\in$	Pertence

SUMÁRIO

1 INTRODUÇÃO

Os estudos sobre a evasão de alunos dos cursos de graduação em computação e áreas correlatas e, em grande parte dessas pesquisas, observa-se que as disciplinas ligadas a programação são responsáveis pela desistência de grande parte dos alunos (GIRAFFA; MORA, 2016). Isso se torna um problema, tanto acadêmico, onde os recursos utilizados pelo aluno são desperdiçados e passa a ser um prejuízo para a instituição, quanto no meio empresarial que perde de ter um funcionário especializado numa área complexa e que carece de mão de obra (SOFTEX, 2013).

A partir de dados do INEP (Instituto Nacional de Estudos e Pesquisas Educacionais) sobre a educação superior, realizado em 2017, é possível notar que as graduações na área da computação possuem um número alto de desistências e poucas conclusões anuais em comparação com uma média baixa de novas matrículas, o que torna preocupante cada caso de evasão. Giraffa e Mora (2016) em uma pesquisa com alunos de uma instituição privada de Porto Alegre, observaram que a maior parte dos alunos na disciplina de Programação I trabalham em tempo integral, reduzindo o tempo desses alunos para se dedicarem aos estudos. O mesmo artigo ainda aponta que grande parte dos alunos não sente interesse em ler o material proposto pelos professores e muitas vezes têm dificuldade no entendimento das atividades.

Observa-se que não existe um único fator responsável pelo baixo rendimento dos alunos (RAMOS *et al*., 2015) porém, nota-se que um conjunto de fatores sociais (GIRAFFA; MORA, 2016) e estruturais (WATSON; LI, 2014) podem ser apontados como indícios do problema. Um desses fatores é o distanciamento do aluno em relação ao conteúdo, o que torna a aplicabilidade do conteúdo algo abstrato, já que mesmo os exercícios apresentados muitas vezes não são compreendidos pelos alunos devido a esse distanciamento com a realidade.

O autor dessa dissertação, docente de disciplinas onde a compreensão em programação é crucial, frequentemente encontra casos de alunos com baixa compreensão da utilização do que foi ensinado. Em alguns casos, os alunos ao observarem a utilização de situações práticas, passam a entender a utilidade das disciplinas. Em outros casos, passam a compreender a importância e a aplicabilidade, porém, sentem-se despreparados para solucionar questões simples, mas que pra eles parecem extremamente complexas, evidenciando a falta de confiança nos próprios conhecimentos.

Diante do contexto exposto, acredita-se que a abordagem construcionista possa auxiliar na elaboração de soluções para os problemas evidenciados. Papert (1985) utiliza o conceito de micromundos para representar áreas livres para a experimentação do aluno, sem a necessidade da avaliação do professor, apenas para a criação e experimentação livre. Assim, desenvolvendo a confiança na produção do conteúdo dos alunos através do processo experimental. Esse conceito é semelhante ao que autores como Squire (2008) e Prensky (2012) utilizam para o uso de jogos

na educação.

Embora existam jogos que apresentam o conteúdo de linguagem de programação C ((BATTISTELLA; WANGENHEIM, 2016)), não foi identificado nenhum ambiente que aborda o conteúdo através de uma simulação do ambiente de trabalho ou que forneça auxílio e avaliação do progresso do aluno. Sendo assim, essa pesquisa busca suprir essa demanda, desenvolvendo e aplicando um jogo sério que fornece um ambiente para prática dos conhecimentos da disciplina de Linguagem e Programação I utilizando linguagem C.

Observando o cenário das disciplinas iniciais de programação e os resultados apontados por diversas pesquisas que investigam os motivos da evasão e reprovação dos alunos, definiu-se a seguinte questão de pesquisa: "É possível que o uso de um jogo sério, inspirado por uma abordagem construcionista, que busca simular um ambiente de trabalho, possa contribuir no aprendizado da disciplina de linguagem C ao engajando os alunos a praticarem o conteúdo?".

1.1 Motivação

A motivação inicial se deve a dificuldade do próprio autor, docente de disciplinas de programação, de avançar em conteúdos no início dos semestres letivos. Isso geralmente ocorre porque os alunos muitas vezes não trabalham com desenvolvimento de software ou não têm o hábito de praticar os conhecimentos das disciplinas anteriores. Como resultado, é comum que os alunos esqueçam de conceitos básicos de programação, criando a necessidade de desenvolver aulas de revisão. Embora o resultado das revisões seja positivo, elas atrasam o cronograma planejado para a disciplina, removendo tempo de explicação de outros conteúdos que, provavelmente, serão utilizadas numa disciplina seguinte. Dessa forma, criando um efeito cascata que, muitas vezes, só será observado nos últimos semestres dos cursos de graduação.

Por meio de um estudo aprofundado na questão das dificuldades dos alunos em relação as disciplinas iniciais de programação, foi observado que o problema é mais complexo e composto de diversas variáveis sociais e técnicas. As disciplinas citadas desempenham papel importante nas taxas de evasão e reprovação e por consequência, dos cursos de computação. O autor, como desenvolvedor de outros jogos voltados a educação, até então nunca tinha desenvolvido nenhum jogo para a sua própria área de formação, o que gera uma motivação pessoal em relação ao que está sendo desenvolvido.

A reprovação nas disciplinas de programação é um problema observado por vários autores, mesmo de outros cursos de fora da área de computação. Por exemplo, em (GUEDES et al., 2017) é demonstrado que cursos de engenharia também possuem taxas de reprovação e evasão muito altos nas disciplinas relacionadas a programação. Em exemplos mais específicos como em (TOSTES, 2019), em dados levantados de dois campus da UFPR (Universidade Federal do Paraná) na disciplina inicial de programação entre os anos de 2015 e 2018, as taxas de aprovação variam entre 15 e 31% dos alunos.

Observa-se que esse cenário não é exclusivo do Brasil, (BENNEDSEN; CASPERSEN, 2007) apresentam um trabalho em que mostram que a taxa de reprovação na disciplina inicial de programação era de 33% nos Estados Unidos. Em uma nova versão do mesmo trabalho, produzido uma década depois, (BENNEDSEN; CASPERSEN, 2019) destacam uma redução nas taxas de reprovação para 28%, no entanto, apontam que o número de cursos e alunos matriculados nessas disciplinas disparou em relação ao primeiro trabalho. Dessa forma, fica evidente que esses problemas ocorrem em muitos países (RAMOS et al., 2015) (RAMOS *et al*., 2015) e que diversas soluções estão sendo propostas na busca por melhorar o desempenho dos alunos nas disciplinas de programação, por exemplo (KUNKLE; ALLEN, 2016), (LEONG; KOH; RAZEEN, 2011), (ESTEVES *et al*., 2009), (PORTER *et al*., 2013).

1.2 Objetivos

Essa pesquisa tem por objetivo investigar as contribuições do CodingJob no aprendizado da disciplina de linguagem C por meio do engajamento.

Como objetivos específicos estão:
1) Identificar e analisar os conteúdos onde os alunos apresentam maior dificuldades;
2) Investigar quais são os tipos de exercícios mais proveitosos para os alunos (em relação ao formato);
3) Aplicar e avaliar o desenvolvimento do CodingJob;

Esse trabalho tem como principal contribuição auxiliar os alunos em disciplinas iniciais de linguagem de programação C a alcançarem melhores desempenhos, contribuindo assim na redução da evasão e da reprovação. Em consequência dos resultados, essa pesquisa também visa servir de objeto de estudos para futuros pesquisadores e desenvolvedores de jogos sérios com foco no conteúdo de computação, especialmente em programação.

2 REFERENCIAL TEÓRICO

As seções deste capítulo apresentam os conceitos envolvidos no desenvolvimento dessa pesquisa, sendo estes: Construcionismo, Jogos Sérios e Aprendizagem baseada em Jogos.

2.1 Construcionismo e Jogos Digitais

Construcionismo é uma teoria educacional baseada no construtivismo que observa que a construção do conhecimento ocorre através da criação de objetos tangíveis e compartilháveis. Sua origem advém do trabalho de Seymour Papert (1985) sobre a teoria Construtivista, desenvolvida principalmente por Jean Piaget, com o qual Papert trabalhou por alguns anos. Enquanto a teoria de Piaget gira, primariamente, no processo epistemológico de como o processo de aprendizagem acontece, Papert buscava criar um sistema mais prático e ativo, buscando as condições ideais para o aluno produzir mais conhecimento do que no modelo educacional padrão de sua época (PAPERT, 1994).

Papert observou nos computadores uma oportunidade para desenvolver um ambiente para formação e prática de conhecimentos matemáticos e de lógica, a partir dos experimentos com a linguagem de programação Logo (PAPERT, 1985), onde o aluno tem um contato ativo com o computador, praticando o conteúdo de forma livre dentro dos limites da ferramenta, construindo objetos, criando ações e muitas vezes, ensinando o computador. Essa prática difere da utilização mais passiva do computador, que replica o ensino instrucionista, e nesse cenário, o computador pode ser visto como uma versão digital dos livros didáticos.

Através da aplicação do Logo foi possível observar que certos ambientes digitais são capazes de criar um ambiente de "caixa de areia", onde o aluno tem um conjunto de ferramentas disponíveis e pode utiliza-las para produzir de forma livre, sem ser confrontado por outras pessoas ou sistemas sociais. Para esse tipo de ambiente, Papert nomeou de *microworlds* (micromundos) (PAPERT, 1980). De acordo com Resnick (1997), micromundos são mundos simplificados, projetados para focar (e tornar acessível) conceitos particulares e formas particulares de pensar.

Inicialmente, Papert (1985) define que micromundos são compostos por três elementos básicos: i) a possibilidade de criar exemplos simples no próprio micromundo; ii) não deve haver obstáculos na manipulação do micromundo, com possibilidade de criação de atividades, jogos e arte dentro do micromundo; e iii) o conteúdo necessário para a aprendizagem deve poder ser definido dentro do micromundo.

Observando o próprio ensino de programação, é possível notar que a prática é amparada em métodos baseados em resolução de exercícios específicos de lógica e prática de listas de exercícios de programação (MCCRACKEN et al., 2001), (SANTOS; GOMES; MENDES, 2013).

As listas de exercícios são compostas de diversas questões que permitem que os alunos pratiquem, no entanto, muitos desses exercícios não representam o mundo do aluno e, geralmente, são totalmente abstratos. Além disso, as questões por diversas vezes apresentam um ciclo instrucional, ou seja, são aplicadas aos alunos e corrigidas no quadro pelo professor, dando a falsa impressão que apenas a solução encontrada é a correta.

Sob o ponto de vista construcionista, tais práticas podem ser funcionais, porém, não otimizam a construção do conhecimento. Sendo assim, o construcionismo foi aplicado nesta pesquisa de forma a criar um ambiente que aborda conceitos de micromundos e ao mesmo tempo é apresentado como uma narrativa com a qual os jogadores, ou seja, os alunos, conseguem criar uma relação empática com a situação dos personagens do jogo. Sendo assim, o construcionismo é utilizado como inspiração para a construção do projeto porém, como observamos anteriormente, ele não é aplicado de forma fiel ao mesmo.

Quadro 1 – Adaptações baseadas em Micromundos Construcionistas

Teoria	**Adaptação**
Área de Livre Criação	Ambiente de jogo que simula local de trabalho. Restringe o jogador a utilizar apenas comandos e ações disponíveis na fase selecionada
Narrativa aberta	Narrativa adaptativa conforme o progresso das fases
Liberdade de errar, erro relativo	Avisa o jogador quais os erros ocorrem conforme o objetivo da fase
Ambiente Sandbox (caixa de areia)	Seleção de fases através do conteúdo. Fases lineares após a seleção do conteúdo

Fonte: autor (2023)

O quadro 1 apresenta um conjunto de adaptações realizadas aplicadas na pesquisa para a produção do jogo CodingJob.

2.2 Jogos Sérios

Jogos digitais podem ser definidos como qualquer jogo que utilize um aparelho eletrônico para ser meio entre o jogador e o jogo. Nessa definição estão inclusos jogos para videogame, computadores, arcades e smartphones.

Os jogos digitais podem ser catalogados pelo seu gênero mecânico, por exemplo, jogos de plataforma são jogos onde o jogador deve se aventurar através da movimentação de um avatar (o personagem que o jogador controla) em um ambiente composto por plataformas, barreiras, precipícios e outros elementos, onde o mesmo deve realizar uma travessia buscando o melhor caminho para isso. Existem diversos estudos em relação a essa forma de catalogar jogos, no entanto, para este trabalho será utilizado o formato apresentado por Apperley (APPERLEY, 2006), onde podemos simplificar essa organização conforme apresentado no Quadro 1.

Dentro do universo de jogos, é possível categorizá-los de acordo com o seu propósito, sendo alguns focados no entretenimento e outros com finalidades específicas, seja informar, educar, treinar ou analisar. Esses jogos, com foco além do entretenimento, são conhecidos como *serious games* (jogos sérios) (ISOTANI; ROCHA; BITTENCOURT, 2015).

Segundo Djaouti e colegas (2011), jogos sérios podem ser classificados em três eixos de categorias: pelo *gameplay* [1], propósito e pelo escopo. Essa classificação é conhecida como G/P/S e foi utilizada para categorizar CodingJob como jogo sério e também para estudos em relação a alterações de *gameplay* em versões futuras.

O *gameplay* se refere a forma como o jogo é estruturado em relação a seu conjunto de regras e objetivos. É possível classificar o *gameplay* através da sua linearidade em relação aos objetivos principais ou do roteiro caso seja um jogo guiado apenas pela história, ou então por seu mundo e história completamente aberta, ficando a cargo do próprio jogador definir o que é mais importante e como o jogo deve encerrar.

Em *gameplay* podemos categorizar dois tipos de jogos [2]:

- Baseados em Objetivos: ocorre quando o jogo possui linearidade em relação a seus objetivos possuindo um princípio, meio e fim.
- *Sandbox*: é um tipo de jogo que é completamente aberto e não possui um objetivo principal, cabendo ao jogador criar seu próprio objetivo pessoal em relação ao jogo.

A próxima categoria descreve a classificação em relação ao propósito do jogo, dividindo-se em três tipos:

[1] Em Game Design, o gameplay é o conjunto de mecânicas que o jogador possui e a relação delas com os desafios que o jogo propõe.

[2] O autor alterou os nomes encontrados na obra de Djaouti 2011 devido a característica linguística dos termos utilizados na versão original. Sendo assim, Play-Based foi alterado para Sandbox e Game-Based para Baseados em Objetivos

- Transmissão de informação: todo jogo que tem como objetivo passar alguma informação para seu jogador, seja ela educacional (Edugame), informativa (Newsgame), persuasiva (Advergame e jogos com temática política) ou subjetiva (Art games e jogos com temática militar oficial).

- Treinamento: quando o jogo apresenta alguma base pedagógica, geralmente behaviorista ou construtivista, o jogo é considerado como de treinamento.

- Troca de dados: são um tipo de jogo mais recente que utiliza de tecnologias de internet, redes P2P, geolocalização e outras como parte importante do conjunto de mecânicas e, em alguns casos, da narrativa.

O último eixo para definir a categoria do jogo é o escopo, que pode ser visto não só como a narrativa do jogo, sendo estes: estado e governo, defesa e militar, saúde, educação, corporativo, religião, arte e cultura, ecologia, política, humanitário, publicidade e pesquisa científica.

Observando os três eixos temáticos, identifica-se que o CodingJob fica categorizado como: (1) gameplay - baseado em objetivos, (2) propósito - treinamento, e (3) escopo - educação.

2.3 Aprendizagem Basesada em Jogos

Aprendizagem Baseada em Jogos ou *Game-Based Learning* (GBL) é um tipo de *E-Learning*[3] onde a ferramenta educacional utilizada é um jogo digital (SQUIRE, 2008). Embora o conceito de aprendizagem com jogos analógicos não seja recente e tenha sido aplicada por outros pesquisadores como Fröebel e Dewey, a utilização de jogos digitais tem alguns diferenciais importantes que o tornam atraentes para a aplicação em sala de aula (SHAFFER et al., 2005).

Entre suas principais vantagens para os docentes podemos citar o fato de ser mais simples de configurar e preparar para uma aula, o custo envolvido tende a ser menor além de, em alguns casos, ser acessível de forma ubíqua podendo ser acessado em diversos dispositivos e até mesmo fora da sala de aula. Já em relação aos alunos, os argumentos mais comuns são motivação, engajamento, adaptabilidade e falha graciosa [4] (PLASS; HOMER; KINZER, 2015). O aproveitamento do aluno geralmente é verificado pelo próprio jogo, seja com a progressão, seja com sistemas de pontuação ou conquistas.

É importante notar que a aplicação de GBL e as técnicas de gamificação contrastam fortemente em suas aplicações. Enquanto na gamificação os sistemas de pontuação, conquistas, roteiro e ranqueamento fazem parte da experiência porém não estão conectados pelo *gameplay*, em GBL o *gameplay* é a parte centralizadora da experiência.

[3] E-Learning corresponde a qualquer forma de ensino baseada em tecnologia da informação.

[4] Falha Graciosa ou Graceful Failure ocorre quando a falha em uma tarefa não é punida de forma rigorosa e permite que o aluno utilize o aprendizado do erro para tentar novamente.

O conceito de falha graciosa tem paralelos com o conceito construcionista de micromundos, apresentado posteriormente neste trabalho.

A aplicação de GBL, segundo Van Eck (2006), pode ocorrer através do desenvolvimento de jogos focados no conteúdo das aulas, na utilização de jogos comerciais focados no lúdico, no entanto, que abordem o conteúdo da aula em que será utilizado ou então através do desenvolvimento de jogos pelos próprios alunos. Nessa pesquisa, a aplicação de GBL ocorre através da aplicação de um jogo sério desenvolvido com inspiração nos conceitos do construcionismo.

3 TRABALHOS RELACIONADOS

Para o projeto e implementação do jogo proposto, foram analisados jogos educacionais, analógicos e digitais, gratuitos e comerciais, utilizados para fins educacionais e que envolvessem os conteúdos de e/ou Linguagem C ministrados em cursos de ensino superior.

Foram utilizadas as bases de artigos da SciELO e do IEEE Xplore, além do Google Scholar para identificação de artigos. Além dos repositórios, foram buscados jogos comerciais através de sites de busca e de repositórios de aplicativos (Google Play e Microsoft Store) e jogos digitais (Steam, Itch.io e Game Jolt).

Durante essa pesquisa, foram identificados diversos jogos que utilizam a Linguagem C como parte da mecânica de jogo, porém, devido as particularidades do currículo da disciplina de Programação, nenhum dos jogos encontrados aborda o conteúdo de forma completa. A pesquisa tem recorte temporal de 2019, ou seja, os trabalhos e jogos pesquisados foram publicados até o ano de 2019.

Codewars (2012) é um ambiente onde os jogadores tem a disposição diversos desafios de programação em várias linguagens de programação, incluíndo em C. A progressão é medida através de um sistema de níves baseados nos conceitos de Kyu e Dan[1] das artes marciais japonesas. Nesse sistema, os jogadores precisam acertar determinados desafios sobre conteúdos diversos de programação. O conteúdo não segue uma ordem didática como na disciplina de Programação, ela é baseada em complexidade dos desafios, nem sempre adicionando novos conteúdos. A falta de ordenação no conteúdo e a possibilidade do usuário pular completamente alguns conteúdos dificultaria a utilização do Codewars como ferramenta de ensino na disciplina, embora possa ser utilizado como um meio de prática para os alunos fora de sala de aula.

Já o CodinGame (2012) apresenta desafios de programação com uma aparência mais lúdica, onde o jogador deve resolver problemas de programação para executar alguma ação em um jogo. Esse jogo varia conforme o desafio, podendo ser desde o controle de uma torre que destrói naves inimigas até criar um algoritmo que faça um carro percorrer uma pista no menor tempo possível. Diferente do Codewars, o Codingame não apresenta nenhum sistema de nivelamento que limite a escolha dos desafios. Isso ocorre porque um dos objetivos do CodinGame é encontrar e selecionar perfis de programadores para tarefas específicas, sendo a plataforma utilizada por diversas empresas de software. Com esse perfil, o CodinGame não apresenta uma ambiente propício para a aprendizagem de de forma estruturada porém, assim como Codewars, pode ser utilizado como uma ferramenta para prática de e programação C.

[1] Kyu são níveis de graduação decrescente que são baseados em faixas coloridas utilizadas na cintura do aluno, geralmente iniciando em 10 e diminuindo de forma gradual até chegar ao 1º Kyu. Após o 1º Kyu ser ultrapassado, o estudante passa ao nível de 1º Dan onde recebe uma faixa preta com marcadores em sua ponta, representando o nível de Dan. A faixa preta ou nível representa que o aluno atingiu um nível onde pode se tornar um instrutor da arte marcial.

Wong e Chou (2007) apresentam a aplicação de código em Linguagem C para gerar ações nos personagens em tela. O jogo se assemelha ao jogo Bomberman[2], onde os personagens devem colocar bombas de forma estratégica para quebrar paredes de um labirinto e assim tentar eliminar outros personagens. O conceito de um ambiente embutido no jogo foi utilizado para gerar a IDE do CodingJob.

O CeeBot (2002) é um conjunto de jogos comerciais para ensinar C/C++ ou Java. Utiliza diversas atividades para ensinar crianças e adolescentes. Todas as atividades compartilham um ambiente de desenvolvimento reduzido, semelhante ao do Bomberman, porém mais estilizado e com uma gama de tutoriais e vídeos de ajuda. No caso de CodingJob, as atividades apresentadas ao jogador seguem características de aplicações voltadas ao ambiente de trabalho de um desenvolvedor de software profissional. Já o conteúdo adicional do jogo é diversificado através da conversação com os demais personagens disponíveis.

O trabalho de Leong *et al*. (2011) apresenta um jogo de exploração via navegador de internet. Baseando-se em um ambiente de aventura espacial (principalmente no universo de Star Wars), JFDI Academy é um jogo que apresenta desafios envolvendo programação básica em um ambiente que simula diversos cenários lúdicos através de problemas simples de programação. Diferente de todos os outros projetos testados, JFDI Academy utiliza uma linguagem de programação semelhante a Scheme. Esse trabalho inspirou os autores em realizar a interação do roteiro do jogo com o conteúdo da disciplina no CodingJob através da estrutura das missões.

Outro caso de jogo analógico é o DexHA (2019), onde os jogadores utilizam um tabuleiro e um conjunto de cartas para realizar ações envolvendo programação. Nesse jogo, os jogadores são divididos em dois grupos que se opõem, Desenvolvedores e Hackers. Enquanto os Desenvolvedores devem produzir o código do algoritmo proposto, os Hackers por sua vez devem impedir a produção do código, anulando linhas de código e roubando partes do algoritmo desenvolvido. Em comparação com CodingJob, o jogo abre espaço para a produção dinâmica e criativa devido a participação de outros jogadores, porém, a participação de jogadores também pode representar um problema, já que será necessário no mínimo dois jogadores para iniciar uma partida. O Quadro 2 apresenta o comparativo entre os jogos analisados e o CodingJob.

[2] Bomberman é uma série de jogos onde a mecânica básica é baseada em utilizar bombas para destruir paredes e inimigos num labirinto 2D

Quadro 2 – Comparativo entre os jogos

Nome	Conteúdo de Linguagem de Programação	Linguagem de Programação	Simula ambiente de fábrica de software?	Funciona em sistema de fases	Dicas dos personagens	Indicativo de erros
CodeWars	Sim	C	Não	Sim	Não	Não
CodinGame	Sim	C	Não	Não	Não	Sim
Bomberman Based Game	Parcialmente	C	Não	Sim	Não	Sim
CeeBot	Parcialmente	C	Não	Não	Não	Não
JFDI Academy	Parcialmente	Scheme	Não	Não	Não	Não
DexHA	Parcialmente	C	Não	Não	Não	Não
CodingJob	Sim	C	Sim	Sim	Sim	Sim

Fonte: Autor (2023)

Em contrapartida aos jogos observados, CodingJob se diferencia em três aspectos: (1) foca no conteúdo da disciplina de Linguagem de Programação I, utilizando a linguagem de programação C, (2) simula o ambiente de trabalho, apresentando tarefas menos abstratas e mais objetivas, (3) trata de outros conteúdos que também são comuns no ambiente da programação profissional, como por exemplo, práticas de qualidade de software. Observa-se que a aplicação desses três aspectos é essencial para a formação de um micromundo conforme definido por Papert 1980, uma vez que simula a realidade de forma clara, apresentando elementos comuns do mundo real, porém, mediados por mecânicas de jogo.

4 CENÁRIO DE USO

Este capítulo apresenta o cenário de uso do jogo CodingJob. Inicialmente, é apresentado o resultado da pesquisa realizada que tinha como objetivo identificar os conteúdos presentes na disciplina de programação C de diversas instituições. O cenário de aplicação do jogo foi a disciplina de Linguagem de Programação I do curso Superior de Sistemas para Internet do IFRS - Campus Porto Alegre.

4.1 A Disciplina de Linguagem de Programação I

O ensino de programação é comum em diversos cursos de graduação, em especial nos cursos de graduação na área computacional. O processo do ensino de programação tem sido criticado (DIJKSTRA, 1988) e alterado com o passar dos anos. No entanto, embora as linguagens de programação utilizadas em disciplinas de programação evoluam com o passar dos anos, o ensino básico de programação segue o mesmo, o que cria um dilema sobre a aplicação ou não de linguagens de programação no nível inicial.

No Brasil, a disciplina inicial de programação está presente nos currículos de cursos ligados a computação como Bacharel e Licenciatura em Ciência da Computação, Bacharel em Sistemas de Informação, Tecnólogo em Sistemas para Internet e Tecnólogo em Desenvolvimento de Jogos Digitais, assim como em grande parte de cursos de outras áreas, como Engenharia Elétrica, Matemática e Física.

A disciplina é a introdução aos conceitos mais básicos do desenvolvimento de sistemas e por isso, essencial para o progresso do aluno nos cursos da área de computação, já que os mesmos apresentam disciplinas que envolvem programação nos semestres seguintes. Geralmente é composta por conteúdos básicos de lógica e momentos práticos com uma linguagem de programação. Os momentos práticos costumam incluir listas de exercícios, um compilado de exercícios com os conteúdos aprendidos no dia. Devido a progressão do conteúdo estar ligada com a complexificação dos algoritmos, o conteúdo das aulas anteriores segue sendo utilizado nas aulas seguintes.

O conteúdo e o formato da disciplina varia conforme a instituição, no entanto, existe uma recomendação da Sociedade Brasileira da Computação (SBC 2005) em relação ao conteúdo obrigatório. Algumas instituições dividem o ensino inicial de programação entre duas disciplinas, uma focada em lógica e geralmente utilizando pseudocódigo [1] e diagramas, seguida de uma disciplina que inicia a prática de programação no computador, utilizando uma linguagem de programação para aplicar os conhecimentos. Para esse trabalho o interesse é apenas nas disciplinas

[1] Pseudocódigo é uma forma de abstração que simula o funcionamento de uma linguagem de programação, geralmente utilizada de forma analógica

Quadro 3 – Conteúdo das disciplinas de linguagem de programação

Conteúdo	**Quantidade de cursos com o conteúdo**
Operadores Matemáticos	20
Variáveis, constantes	20
Expressões Condicionais (If, Else, Switch – Case)	20
Laços de Repetição (For, While, Do-While)	20
Introdução Linguagem de Programação* (C)	9
Introdução Linguagem de Programação (Python)	5
Introdução Linguagem de Programação (Outra)	6
Funções / Métodos	20
Estruturas	9
Ponteiros	9
Vetores	20

Fonte: Autor (2023)

que utilizam alguma linguagem de programação, já que essas são padronizadas, ao contrário do pseudocódigo que pode variar de professor para professor.

Para padronizar o conteúdo utilizado na pesquisa, o autor analisou o currículo de vinte cursos de graduação, conforme descrito no Quadro 3.

A linguagem mais comum utilizada é a linguagem C, seguida de Python, Java e C++. A utilização de uma linguagem de programação com vasta documentação e relevancia de mercado parece ser a chave para escolha da linguagem utilizada em Linguagem de Programação I. As linguagens de programação que foram apresentadas nesse levantamento coincidem com as mais utilizadas segundo o indice TIOBE (2021), que realiza o levantamento anual das linguagens de programação mais utilizadas no mundo.

Devido à diferença entre o perfil dos egressos dos cursos e por consequência o conteúdo dos cursos, o Quadro 4 apresenta quais cursos e quais as linguagens de programação são utilizadas em cada disciplina. Os dados dos cursos e disciplinas foram investigados no primeiro semestre de 2022 nos sites das instituições. O conteúdo das disciplinas foi levantado a partir dos Planos Pedagógicos dos Cursos. O resultado da pesquisa trouxe a linguagem C como a mais utilizada dentro do grupo pesquisado.

Quadro 4 – Cursos analisados

Curso	**Disciplina**	**Instituição**
Bacharel em Sistemas de Informação	Fundamentos de Programação	PUCRS (Java)
Bacharel em Ciência da Computação	Algoritmos e Programação	UFRGS (C)
Bacharel em Ciência da Computação	Algoritmos e Programação	USP (C)
Bacharel em Ciência da Computação	Algoritmos e Programação de Computadores	UNICAMP (C)
Bacharel em Sistemas de Informação	Programação de Computadores I	UFF (Python)
Tecnólogo em Desenvolvimento de Jogos Digitais	Algoritmos e Programação	FACCAT (Python)
Tecnólogo em Desenvolvimento de Jogos Digitais	Algoritmos e Programação com C++	UNISINOS (C++)
Bacharel em Ciência da Computação	Algoritmos e Estruturas De Dados I	UFAM (C)
Bacharel em Ciência da Computação	Introdução a Programação	UFPE (Java)
Bacharel em Engenharia da Computação	Introdução a Lógica de Programação	UFBA (Pascal)
Bacharelado em Ciência da Computação	Programação I	UFPR (C)
Bacharelado em Ciência da Computação	Computação I	UFRJ (C)
Bacharelado em Ciência da Computação	Introdução à Ciência de Computação I	USP/ICMC (C)
Bacharelado em Sistemas de Informação	Introdução à Computação	UFG (Java)
Computer Science BSc	Principles of Programming	UCL (C/Haskell)
Computer Science and Engineering	Fundamental Programming A	Waseda University (C/C++)
Computer Science and Engineering	Thinking and Approach of Programming	Shangai Jiao Tong University (Python)
Electrical Engineering and Computer Science	Fundamentals of Programming	MIT (Python)
Licenciatura en Ingenieria en Ciencia de Datos	Introducción a la Programación	PUC Chile (Python)
Computer Science	Computer Science A	FU Berlin (Haskell)

Fonte: Autor (2023)

É importante notar que as disciplinas apresentadas no quadro 4 são compartilhadas com outros cursos, por exemplo, a disciplina de Fundamentos de Programação da PUCRS é uma disciplina transversal existindo também nos cursos de Engenharia de Software, Engenharia de Computação, Ciência da Computação e Ciência de Dados. A linguagem de programação C de

propósito geral, é conhecida como linguagem de programação de sistemas devido a seu histórico de utilização para criação de sistemas operacionais e compiladores ((KERNIGHAN; RITCHIE, 1988)). Segundo o índice TIOBE (2021), C é a linguagem de programação mais popular no início do ano de 2021, sendo amplamente utilizada no mercado de software embarcado e na construção de aplicativos de segurança. C é uma linguagem que é utilizada com frequência nas disciplinas iniciais de programação, pode-se supor que seja pelas características sintaxicas que são aproveitadas em diversas linguagens de programação modernas, como C++, C# e Java.

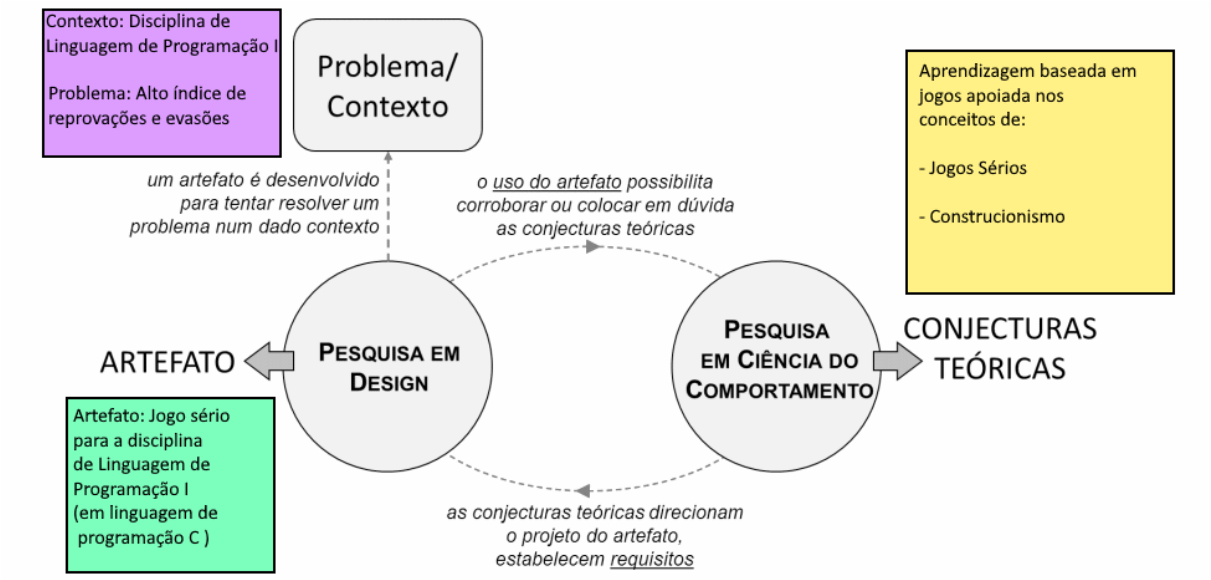
5 PERCURSO METODOLÓGICO

Esta pesquisa é de natureza aplicada, sendo utilizada a *Design Science Research* (DSR) como objeto norteador da produção científica. A escolha pelo DSR se deve pela necessidade de aplicar um método que abrangesse o problema de pesquisa e a produção e aplicação do artefato produzido para a mesma.

Na DSR, o pesquisador está comprometido com dois objetivos: (1) resolver um problema prático num contexto específico por meio de um artefato e (2) gerar novo conhecimento científico (PIMENTEL; FILIPPO; SANTORO, 2019). Nesse sentido, a DSR divide sua atenção para a solução de um problema a partir da relação entre a pesquisa em design e a pesquisa em ciência do comportamento. Na pesquisa em design o foco é mantido na análise e produção do artefato que será desenvolvido, enquanto na pesquisa em ciência do comportamento é focado no conjunto de conjecturas teóricas, ou seja, no conteúdo que embasa a construção do artefato para solucionar o problema proposto.

A Figura 1 apresenta a organização da pesquisa e a relação entre a pesquisa em design e a pesquisa em ciência do comportamento.

Figura 1 – *Design Science Research* – Planejamento Inicial



Fonte: autor (2023). Adaptado de Pimentel *et al*. (2019)

O problema definido para o trabalho foi o dos altos índices de reprovações e evasões nas disciplinas de Linguagem de Programação I. O contexto, disciplina de Linguagem de Programação I, limita a abrangência da pesquisa apenas para a disciplina inicial de programação, levando em consideração que um curso de computação costuma ter pelo menos uma disciplina de programação por semestre. O artefato proposto é um jogo sério que aborda o tema de

programação C simulando o ambiente de uma fábrica de software de pequeno porte. Por fim, a pesquisa em ciência do comportamento aborda a ludificação da aprendizagem (SQUIRE, 2008) e construcionismo (PAPERT, 1985).

O problema de pesquisa abordado nessa dissertação é o alto índice de reprovação e evasão nas disciplinas de linguagem de programação. O contexto, limita a abrangência da pesquisa apenas para a disciplina inicial de programação, levando em consideração que os cursos da área geralmente têm pelo menos uma disciplina de programação por semestre. O artefato proposto é um jogo sério que aborda o tema de programação C simulando o ambiente de uma fábrica de software de pequeno porte. Por fim, a pesquisa em ciência do comportamento aborda a ludificação da aprendizagem (SQUIRE, 2008) e construcionismo (PAPERT, 1985).

Durante o processo de desenvolvimento, o jogo foi aplicado em quatro turmas, sendo duas turmas de Linguagem de Programação I (primeiro semestre do curso), uma de Estrutura de Dados I (segundo semestre do curso) e uma de Engenharia de Software III (quarto semestre do curso), todas pertencentes ao Tecnólogo em Sistemas para Internet do IFRS (Campus Porto Alegre).

CodingJob foi construído conforme o relato de alunos que já cursaram a disciplina, elencando suas dificuldades e opiniões em relação ao conteúdo. Esses relatos foram obtidos por meio de um questionário que tinha por objetivo obter informações e auxiliar a quantificar questões de importância sobre o conteúdo. Com os resultados foi possível desenvolver as atividades do jogo focando nos conteúdos mais críticos segundo os dados obtidos através das respostas dos alunos.

Os alunos que utilizaram o jogo também responderam um questionário para auxiliar na identificação do perfil da turma. Essas informações serviram para identificar: i) para quais perfis de alunos o jogo poderá ser mais benéfico?; e ii) quais perfis de alunos são mais propensos a não utilizarem o jogo?.

Por fim, o jogo foi avaliado conforme uma adaptação no modelo MEEGA+ (PETRY 2019), que é um modelo de avaliação de jogos digitais que visa identificar o aproveitamento do aluno em relação ao conteúdo apresentado pelo jogo, assim como o jogo em si (avaliação das mecânicas, apresentação, etc.).

6 CODINGJOB

Este capítulo apresenta os passos de desenvolvimento do jogo CodingJob, desde o projeto até sua validação pelos alunos.

6.1 Contexto

O jogo CodingJob foi construído com base em duas premissas

1. Um jogo que simule o ambiente de uma fábrica de software de forma lúdica, utilizando de características fictícias para transformar as ações do jogador em algo rápido e simples de ser solucionado.
2. A utilização de código em linguagem C como base das atividades do jogo.

É importante notar que enquanto as atividades utilizam linguagem C e um cenário baseado em fábrica de software, o jogo não visa apresentar outros conteúdos, como segurança de software ou práticas avançadas de programação. Sendo assim, ele não replica o cenário de uma fábrica de jogos e sim, simula de forma lúdica. Ainda mantendo o cenário de lúdico, a linguagem C não é compilada pelo jogo, mas analisada através de uma validação de strings[1].

No planejamento inicial desta pesquisa, estava previsto para o jogo ser aplicado em sala de aula, em turno invertido, com a presença do autor, onde a pontuação dos alunos seria validada durante o período de testes através de um sistema de pontuação simples. No entanto, com o cenário imposto pela pandemia, onde as aulas tiveram seu modo alterado para o formato remoto, foi necessário alterações para que o jogo fosse aplicado de forma independente por parte dos alunos, sem a presença de um tutor *in loco*, apresentando um maior número de informações sobre o conteúdo e dicas sobre as atividades através de personagens interativos, além de um histórico com mais informações sobre o desempenho do aluno. Em 2022, com o retorno das aulas presenciais, a pesquisa original foi retomada e o jogo foi novamente alterado para ser aplicado em sala de aula com os alunos, no entanto, seria aplicado durante um período de aula, levando aproximadamente 30 a 40 minutos para ser aplicado.

[1] Validação de string é uma forma de verificar se a string (cadeia de caracteres inserida em um campo) informada pelo usuário é semelhante a string da resposta da questão / exercício

6.2 Projeto

O projeto do jogo foi elaborado de acordo com o framework MDA (HUNICKE; LEBLANC; ZUBEK, 2004). O MDA foca em três elementos na construção de jogos:

- Mecânica: componentes particulares do jogo como o conjunto de ações do personagem por exemplo.
- Dinâmica: o comportamento das mecânicas conforme as ações do jogador e do próprio jogo, por exemplo a ação de construir e testar um algoritmo no jogo.
- Estética: descreve a resposta emocional que o jogo deve proporcionar ao jogador, por exemplo o resultado positivo da ação de entregar um programa funcionando dentro do jogo.

A partir da utilização do MDA, foi possível categorizar as prioridades do jogo conforme apresentado no Quadro 5.

Quadro 5 – Modelo proposto de acordo com o framework MDA

Mecânica	**Dinâmica**	**Estética**
- Movimentação; - Interagir com outros personagens; - Interagir com o cenário; - Utilizar um computador para acessar internet (dicas sobre conteúdo), IDE (completar códigos C) e acessar e-mails (dicas de C).	- A movimentação e a interação é realizada através de um pseudocódigo que faz o personagem trocar de salas; - A utilização do computador é via uma interface visual; - O sistema de e-mail é apresentado de forma semelhante a um aplicativo de e-mail do início dos anos 2000; - A IDE apresentada apenas simula o código a ser desenvolvido, apresentando caixas de seleção para introduzir o trecho de código adequado ao algoritmo e simulando sua compilação e resultado.	- O jogo é narrativo (apresenta um roteiro que progride conforme as ações do jogador), simulador (simula uma realidade do mercado de trabalho) e desafiador (progride conforme os códigos da empresa são entregues).

Fonte: Autor (2023)

O jogo foi desenvolvido através da metodologia Scrum (KEITH, 2010), e foi adaptado conforme o modelo PROGame (ALCOVER; CAPó; MOYá-ALCOVER, 2018) que foca na produção de jogos sérios. Nessa adaptação, o autor divide as fases de produção do projeto em Ideação, Pré-Produção, Produção e Pós-Produção.

Na fase de Ideação, é levantado o problema que será trabalhado e as opções de mecânicas que podem ser aplicadas para o cenário. Na fase de Pré-Produção, é desenvolvido um protótipo do jogo que é aplicado de forma rápida com indivíduos que já conhecem o conteúdo. Na fase de Produção o jogo começa a ser produzido e aplicado junto ao público alvo. A Pós-Produção encerra o projeto através da análise dos dados obtidos durante a fase anterior.

Segundo Keith (2010), projetos de desenvolvimento de jogos são divididos em diversas áreas de conhecimento distintas e em um modelo de desenvolvimento ágil como Scrum, é necessário identificar os papéis da equipe, no caso desse projeto apenas uma pessoa é responsável por todas as áreas e, sendo assim, as tarefas foram divididas por área, especificamente em Game Design, Arte e Programação.

CodingJob é um jogo sério, sendo assim, a representação do conteúdo deve possuir características realistas na forma que é apresentado. Porém, como ainda é um jogo e não um simulador, é necessário que exista equilíbrio entre o conteúdo e o lúdico. Para isso, optou-se por produzir o jogo com visual estilizado, semelhante a jogos para o público mais jovem além da utilização de uma linguagem mais leve.

Para organização do projeto, foi desenvolvido um *Game Design Document* (GDD) para

apresentar todos as informações relevantes ao design do jogo. A próxima seção apresenta o GDD do jogo CodingJob.

6.3 Game Design Document

O GDD é um documento utilizado na indústria de desenvolvimento de jogos para documentar todas as características de design do jogo em desenvolvimento. O documento geralmente é o segundo a ser desenvolvido após a apresentação do *Game Concept* [2].

Para esse trabalho, o GDD, apresentado na Tabela 1, foi resumido para simplificar a visualização e evitar a repetição de informações mencionadas anteriormente no texto.

Tabela 1 – Game Design Document - CodingJob

Game Design Document - CodingJob
Título do Jogo: CodingJob
Gênero: Jogo Sério
Categoria: Quebra Cabeça
Plataforma: Windows, MacOS e Linux
Tipo: Monousuário
Cenário: Temática de Fábrica de Software
Público-alvo: Alunos das disciplinas iniciais de programação que utilizam linguagem C.
Programação: Foi utilizada a game engine Unity em sua versão 2019, além da linguagem de programação C# para codificar o jogo.
Descrição:

[2] Game Concept é um documento, geralmente desenvolvido para vender a ideia do jogo para um investidor.

CodingJob é um jogo em estilo Quebra Cabeça que simula o trabalho de um programador de computadores recém-contratado para uma pequena fábrica de software. O jogador controla o personagem Alan (em homenagem a Alan Turing) e participa de projetos de software baseados em casos prováveis de programas de computador encomendados por uma empresa do ramo. O foco do jogador será o conteúdo da disciplina de linguagem de programação C e a dificuldade do jogo progride em paralelo com o conteúdo que é apresentado. Para auxiliar o jogador, é possível visitar um colega de trabalho chamado Pascal (em homenagem a linguagem de programação Pascal, e por consequência, do matemático Blaise Pascal) que fornece dicas sobre o conteúdo da fase atual por meio de um sistema semelhante a um aplicativo de conversa. Outro personagem que simula o papel de um colega virtual é Haskell (em homenagem a linguagem de programação Haskell, e por consequência, ao matemático Haskell Curry) que irá fornecer dicas sobre a construção do código da fase, explicando quais variáveis e comandos devem ser utilizados. O jogo funciona via arquivo executável em resolução 800x600 para manter o jogo funcionando de forma eficiente e, se o jogador desejar, facilitar a troca de telas com o conteúdo disponibilizado pelo professor ou pesquisar outros materiais.

Sistema do Jogo:

O jogo inicia com a apresentação dos personagens e das mecânicas básicas. Ao terminar a introdução, o jogador poderá selecionar qual conteúdo deseja trabalhar. Para cada conteúdo, será apresentado uma tela inicial de um computador, onde é possível selecionar três ícones, sendo estes: Email, Super IDE e chatApp. No ícone Email, o jogador poderá ler um email de Haskell que explicará o cenário da fase e quais detalhes sobre a fase são importantes atentar. O chatApp simula uma ferramenta de conversação por texto, onde o jogador poderá verificar informações sobre o conteúdo utilizado na fase atual. O jogador pode escolher qual o conteúdo irá utilizar em um menu dividido por perguntas pré-estabelecidas, por exemplo: "Como funciona o comando printf() em C?"e o personagem Pascal irá responder de forma textual, muitas vezes trazendo exemplos de aplicação. Por fim, o ícone de Super IDE leva o jogador para a tela da fase atual, onde o jogador deverá ler o código e completar os campos vazios com o comando correto. Caso o código esteja correto, o jogador receberá uma mensagem informando que o programa está correto. Caso contrário, a interface irá apresentar qual campo está incorreto.

Estrutura Narrativa:

O jogo inicia com uma apresentação dos personagens e da história por trás do personagem principal. Ao iniciar uma fase, o jogador obtém mais informações sobre os projetos, seus clientes e outras informações através da leitura dos emails enviados por Haskell ou pelas suas mensagens de dicas, que podem ser lidas na tela da IDE.

Mídia:

O jogo apresenta personagens e cenários produzidos em voxel, texturas e ícones utilizados nas interfaces, além da trilha sonora.

Fluxo de Jogo:

O jogo é dividido em cinco cenários que estão relacionados aos conteúdos da disciplina de Linguagem de Programação I. Foram desenvolvidas nove (09) questões compostas por um conjunto de campos que permitem edição do código. Nesses campos o aluno deve utilizar o conhecimento do conteúdo para dar continuidade ao algoritmo desenvolvido em tela. Caso a resposta esteja correta, o aluno progridirá para outra questão mais complexa ou poderá seguir para o próximo conteúdo.
Mapa de Ambientes:
O jogo apresenta os seguintes ambientes:
Tela 1 a 5 - Abertura do jogo e apresentação da história;
Tela 6 - Tela Principal, seleção de fases;
Tela 7 - Email (atualizada conforme a seleção da fase);
Tela 8 - chatApp (atualizada conforme a seleção da fase);
Tela 9 - IDE (atualizada conforme a seleção da fase).

Fonte: Autor (2023)

6.4 Produção

Esta seção apresenta o processo de produção do jogo CondingJob. É importante notar que o jogo segue um modelo de produção adaptado do modelo de desenvolvimento ágil Scrum apresentado em (KEITH, 2010) de forma que todas as ações são produzidas por uma pessoa de forma intercalada conforme as necessidades do projeto. Sendo assim, o jogo é produzido por cenários (conteúdos de linguagem de programação C) e, em seguida, são produzidas mais fases para complementar esses cenários. As subseções seguintes apresentam as fases de produção.

6.4.1 Ideação

A ideação é o processo de produzir ideias de projeto. Devido a definição de um tema central (ensino da linguagem de programação C para alunos iniciais de cursos de graduação em computação), a ideação foi centrada no tema, permintido um foco em relação ao tema e liberando as possibilidades de mecânicas.

O autor avaliou jogos e artigos que abordam tema semelhante e observou os comentários dos jogadores em relação aos jogos comerciais, de forma a identificar possíveis problemas em relação a escolha das mecânicas e ambientação. Observou-se que jogos que utilizam linguagens de programação através de um interpretador de uma linguagem de programação próprio, geralmente, são vistos como jogos mais adultos e sérios. No entanto, são considerados mais difíceis, onde raramente seus jogadores conseguem concluir o jogo. Já os jogos que apresentam visual mais colorido e mecânicas baseada em programação em blocos atraem um público mais amplo, porém, costumam ser jogos em formato sandbox, ou seja, não possuem sequencialidade e geralmente não abordam conteúdos de lógica mais complexos.

Em relação a plataforma, a maior parte dos jogos analisados são disponibilizados para PC, Web e Android, sendo que os jogos para PC e Android possuem características mais voltadas ao lúdico, enquanto que para Web os jogos variam muito entre o lúdico e o educativo, muitas vezes sendo definidos como simuladores ou ferramentas de treino ao invés de jogos em si. Sendo assim, a partir dessas análises, foram definidas as seguintes características para o jogo:

- Atender as plataformas PC, Linux e MacOS, mas adaptado facilmente para mobile ou web;
- Apresentar uma aparência animada, mas tratar de assuntos profissionais da área de programação;
- A mecânica principal deve ser baseada em validação de campos com pedaços do código C que devem estar corretos para o jogador prosseguir no jogo;
- Possuir uma narrativa baseada em algo que os jogadores criem laços afetivos e que se observem no lugar do protagonista. Nesse caso, o jogo visa simular um jovem entrando no mercado de trabalho na área de programação de software;
- Conter personagens de apoio que interagem e utilizam ferramentas semelhantes as encontradas no mercado de trabalho;
- Ser focado e com estágios breves, evitando que o jogador fique preso em um conteúdo e não possa prosseguir para outro mais avançado, caso já conheça ou não queira passar por estágios iniciais sempre que começar o jogo.

6.4.2 Pré-Produção

Nesta fase o autor selecionou as ferramentas que foram utilizadas no processo de produção do jogo, além de definir as mecânicas e o design (visual) do jogo. No desenvolvimento foi utilizada a *game engine*[3] Unity. Um dos principais motivos para essa escolha é a facilidade de portar o produto final para diversas plataformas. O jogo foi mantido em versões para computadores pessoais com sistema operacional Windows, Linux e MacOS, para isso, a Unity fornece a opção de desenvolver o jogo apenas uma vez e criar executáveis nas versões necessárias. Outro motivo que levou à escolha foi o fato do autor já ter experiência com a ferramenta.

A criação de personagens foi desenvolvida com a *Magica Voxel*, que é ferramenta de produção de modelos em voxel [4]. Para texturas e interfaces gráficas, foram utilizadas as ferramentas *Asesprite* e *Paint.Net*. Em relação ao estilo gráfico escolhido, o autor optou por trabalhar com gráficos baseados em voxel, semelhante a jogos como Minecraft, além de um

[3] Game Engine são ambientes de produção de jogos digitais que unem diversas ferramentas para simplificar o processo de produção.

[4] Voxel é o menor polígono dentro de uma grade tridimensional.

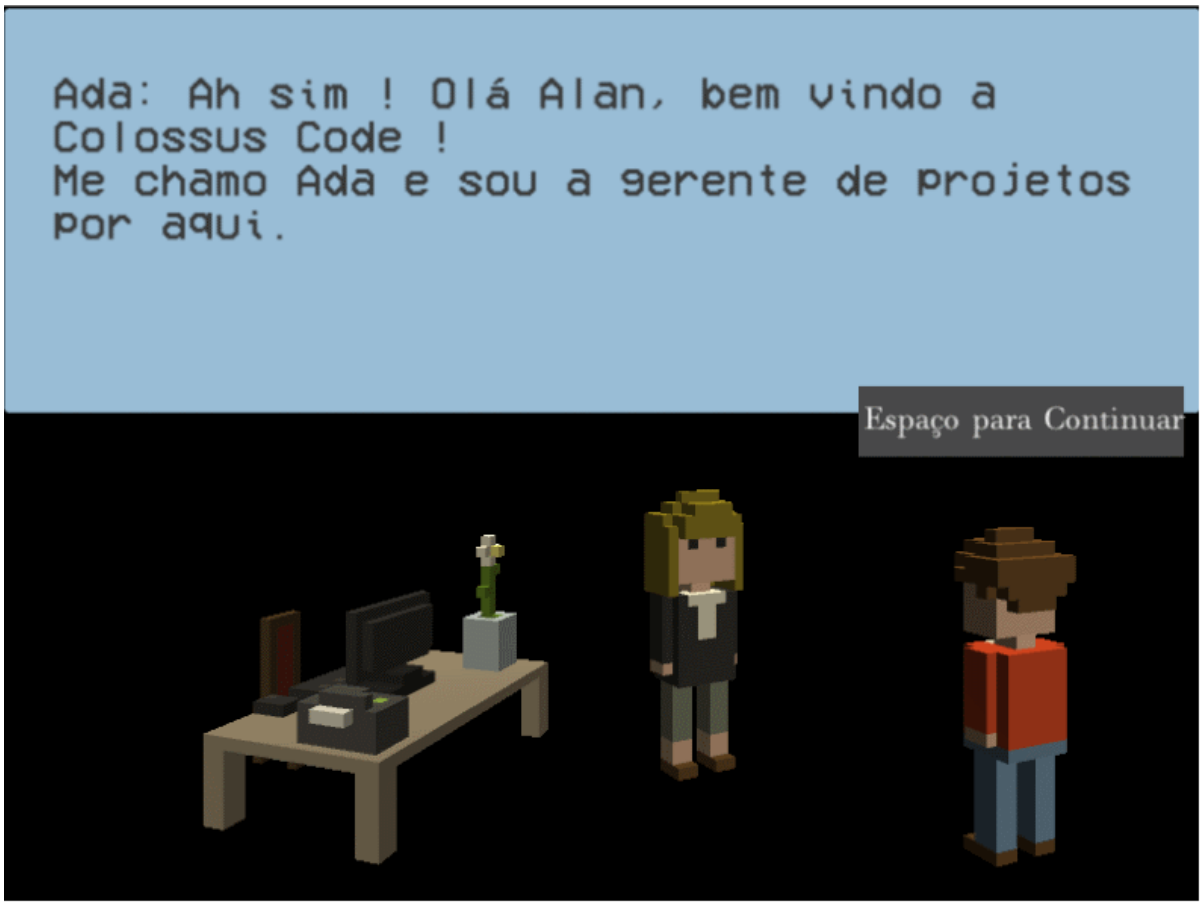
conjunto de texturas cartunizadas, dando uma aparência leve ao jogo. As interfaces tentam simular softwares conhecidos pelo grande público, como ambientes de email e de conversa instantânea, porém, apresentam um exagero em relação aos botões e fontes, de forma a ter uma aparência focada na diversão e não na utilidade das ferramentas reais.

6.4.3 Produção

O primeiro passo para a produção do projeto é a construção das mecânicas mais básicas do jogo, nesse caso a produção das fases que utilizam o conteúdo abordado na pesquisa. O planejamento de produção foi dividido em quatro fases:

1. Protótipo: primeira versão desenvolvida para ser levada para testes com usuário. Nesses testes são verificados se a mecânica e as características visuais combinam e agradam os jogadores. Na Figura 2 observa-se um exemplo de tela de conversação entre os personagens.

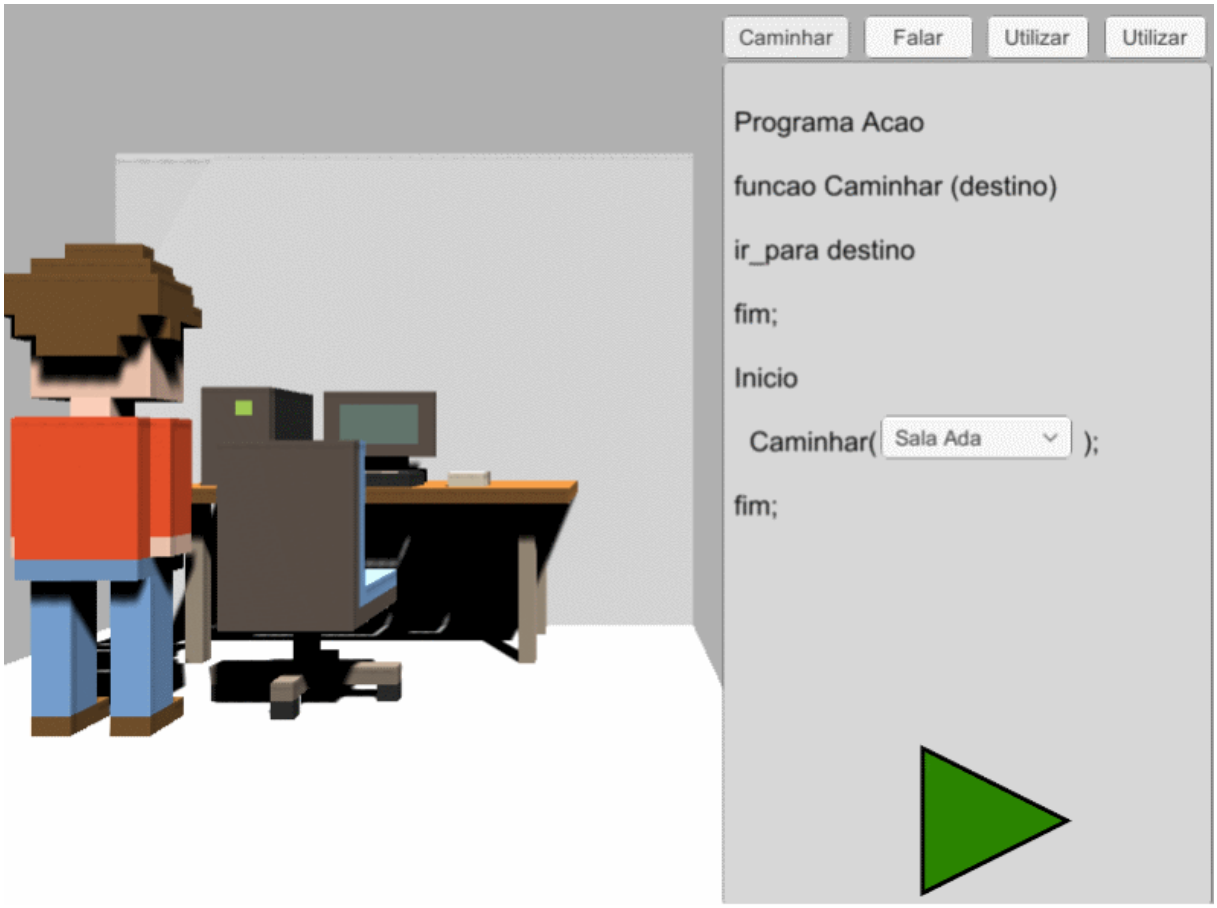
Figura 2 – Protótipo da tela de conversação



Fonte: Autor (2023)

Conforme ilustra a Figura 2, além dos personagens, nota-se que o cenário se mantém escuro, apenas com uma mesa sendo disponibilizada. Os cenários foram produzidos para versões mais avançadas do projeto. A mecânica desenvolvida inicialmente permitia que o jogador andasse entre salas dos personagens completando um código simples de movimentação. Também era possível utilizar outras ações como falar e pegar itens através dessa mecânica.

Figura 3 – Protótipo da mecânica de interação



Fonte: autor (2023)

Na Figura 3, observa-se como a mecânica de interação era executada. A mesma foi rejeitada no primeiro teste com os primeiros jogadores, que relataram que ela não acrescentava muito no objetivo proposto. Os exercícios com o conteúdo de linguagem de programação C são apresentados em um ambiente que simula um software do tipo ambiente de programação.

Figura 4 – Protótipo da tela do ambiente de programação

Fonte: autor (2023)

Conforme ilustra a Figura 4, o ambiente de programação é simples, apresentando apenas uma área de retorno e um botão para compilar o código construído. O jogador deve completar os campos específicos do código para prosseguir no jogo. Caso um dos campos esteja errado, o jogador receberá uma informação sobre o erro e deverá corrigir e compilar novamente até o código ser compilado.

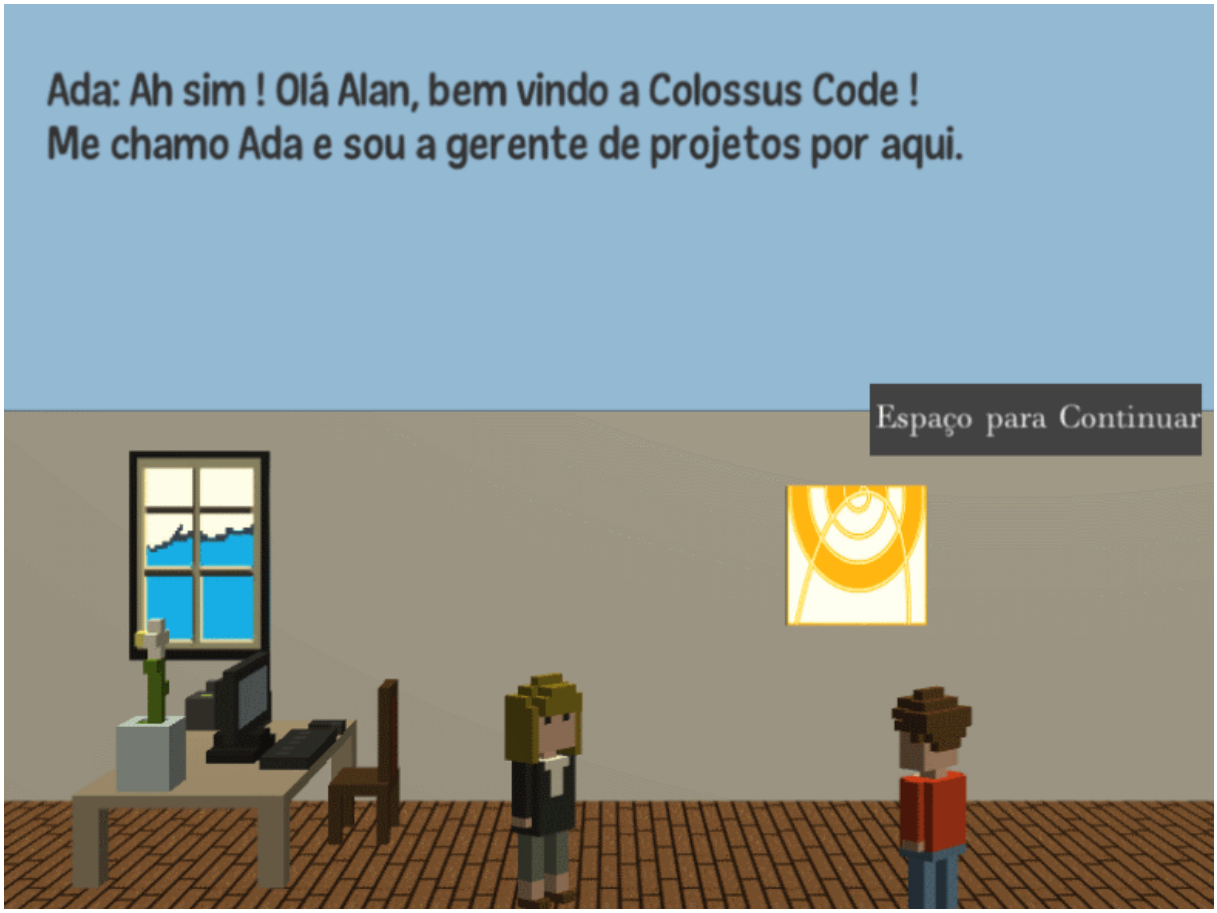
Nessa fase, o jogo foi apresentado a um grupo de jogadores para validação do jogo como um objeto lúdico. O perfil dos jogadores foi o de ex-alunos da disciplina inicial de programação, sendo que alguns desses jogadores não realizaram a disciplina com linguagem C, mas todos tiveram pelo menos uma disciplina com linguagem C durante o ensino superior.

2. Fase Alfa: primeiro teste que verifica questões relacionadas ao conteúdo. Nessa fase o jogo é testado pelo público alvo e apresenta características de um jogo em sua forma final, como apresentação da arte de forma completa, sons e efeitos sonoros. No entanto, ele é apresentado em formato limitado, com apenas algumas fases disponíveis para testar. Durante essa fase, diversas alterações são realizadas em relação a fase anterior.

Os resultados da fase de prototipagem geraram diversos questionamentos, assim como o

cenário imposto pela pandemia, que serviram para produzir um conjunto de alterações no jogo.

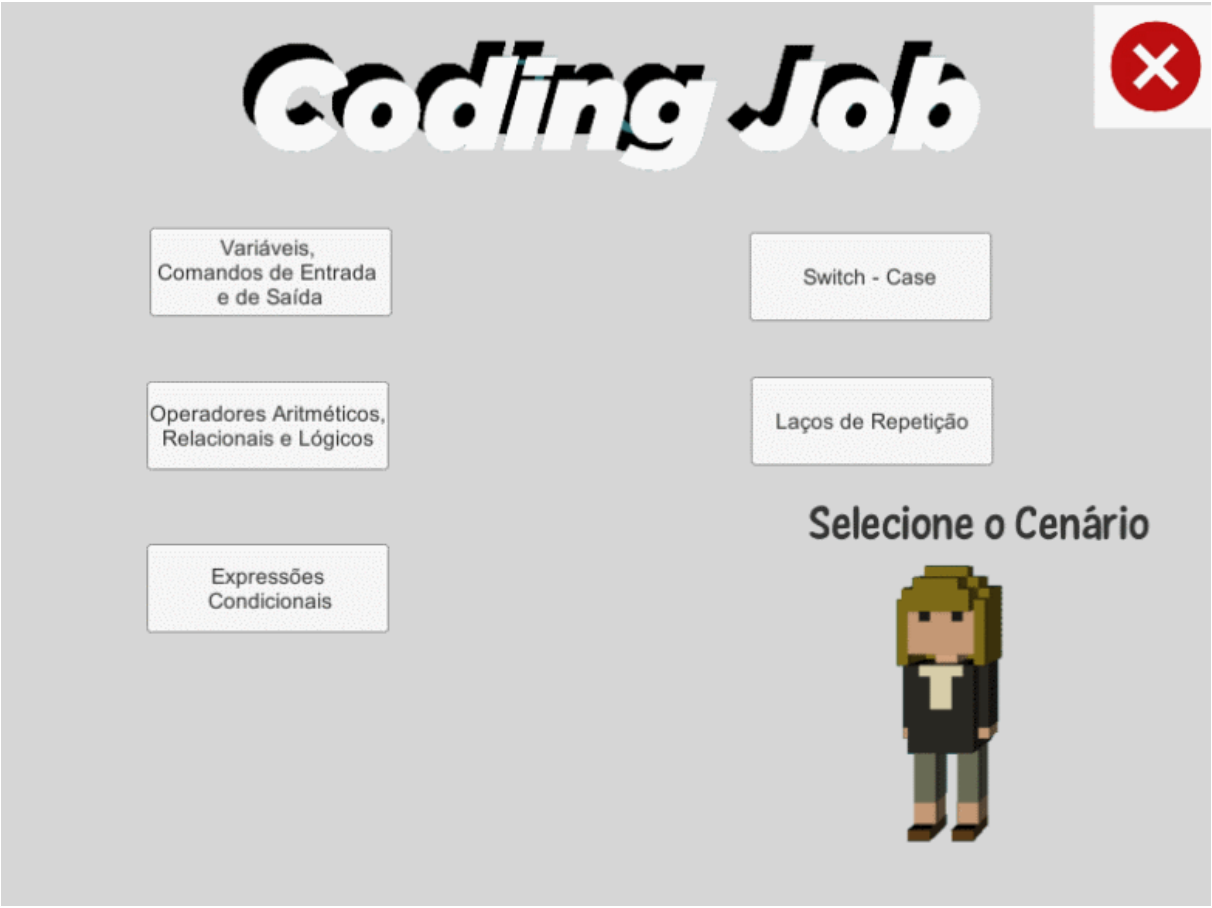
Figura 5 – Tela de abertura da Fase Alfa



Fonte: autor (2023)

A primeira alteração realizada foi estética, onde um maior número de elementos visuais foram produzidos para criar ambientes mais claros e com maior variedade de elementos. A fonte utilizada foi alterada, sendo trocada por uma fonte de melhor visualização. A Figura 5 apresenta a mesma tela de abertura, agora com a inserção de elementos visuais. Esses elementos foram incluídos em outras telas onde o jogador interage com os personagens.

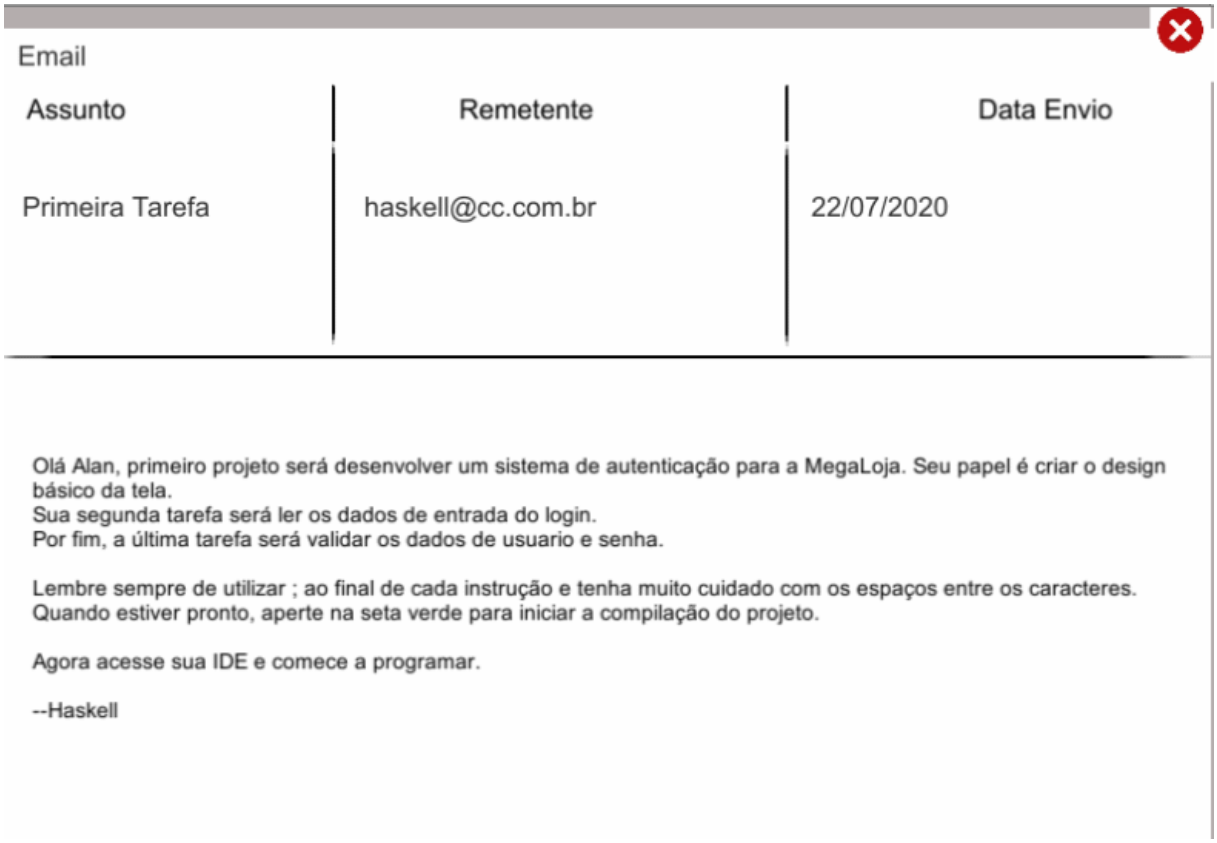
Figura 6 – Tela de seleção de cenário da Fase Alfa



Fonte: autor (2023)

A segunda alteração ocorre através da remoção da narrativa linear por uma baseada em cenários. Essa alteração ocorreu quando foi observado que o jogador poderia ficar preso a um exercício, não conseguindo avançar a um cenário de seu interesse. A divisão por cenários foi escolhida como uma forma de simplificar esse avanço e evitar a frustração do jogador. Também é uma forma de focar em conteúdos específicos, auxiliando o jogador a praticar apenas no que ele necessita de reforço. A Figura 6 apresenta o menu inicial com as alterações, permitindo que o jogador escolha o cenário.

Figura 7 – Tela de Email do Haskell da Fase Alfa

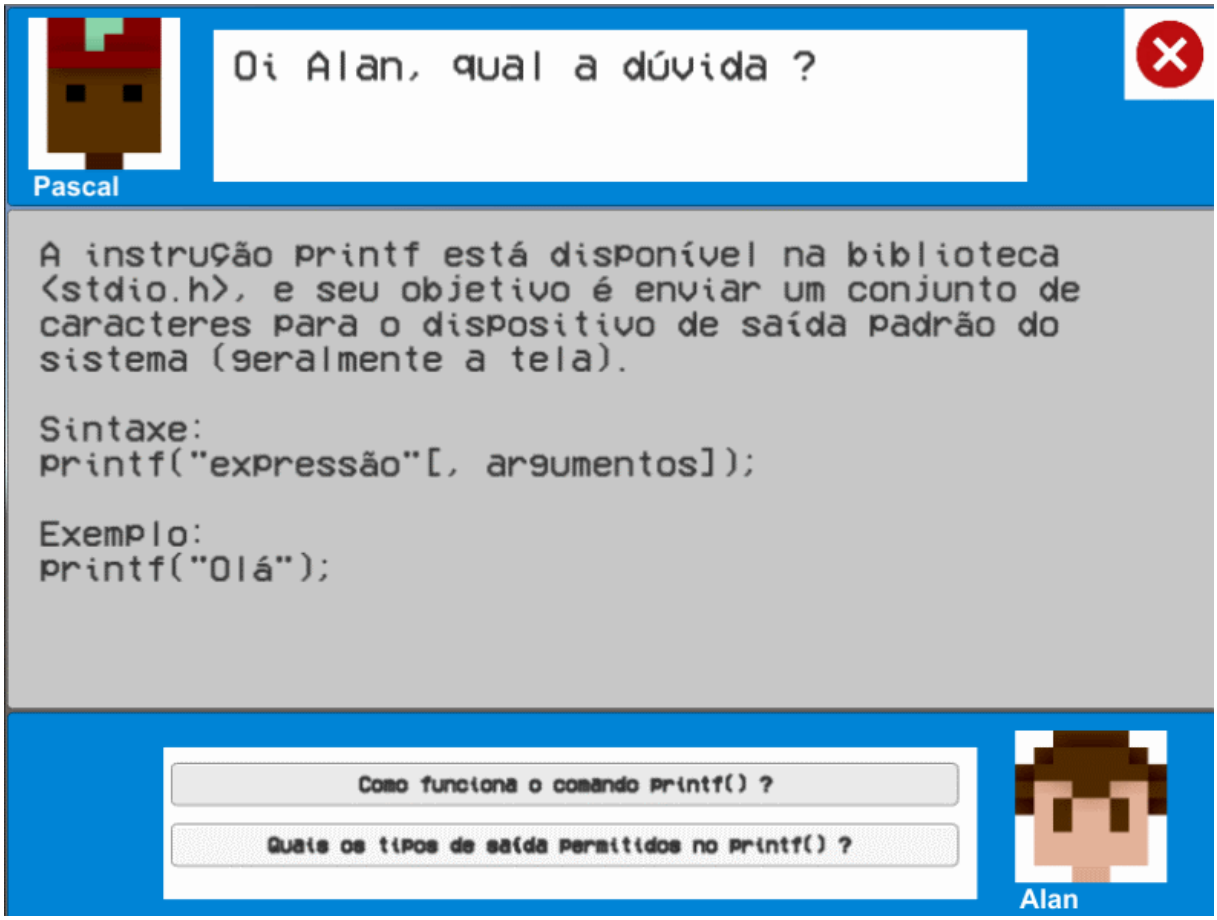


Fonte: autor (2023)

Outras alterações importantes foram em relação a apresentação dos objetivos de cada cenário para o jogador. Inicialmente essa informação era apresentada através de um texto explicativo e escrito de forma formal. Na versão alfa, além de alterações na interface (apresentado em formato de email), os textos são descritos de forma sucinta e menos formal. Algumas informações e dicas foram removidas dos emails para se tornar um conjunto de notas que podem ser acessadas durante o momento que o jogador estiver realizando a atividade.

Conforme observa-se na Figura 7, a interface do email simula uma ferramenta de email, apresentando informações básicas do remetente. O texto apresentado na versão inicial do projeto foi ampliado para informar detalhes mais específicos do software em produção na fase do jogo.

Figura 8 – Tela do chatApp da Fase Alfa



Fonte: autor (2023)

Como o jogo foi projetado inicialmente para praticar o conteúdo de linguagem de programação C em paralelo a disciplina, o protótipo foi desenvolvido sem nenhuma explicação sobre o conteúdo, apenas sobre as fases. Após os testes, foi verificado que os jogadores que testaram consideraram frustrante o movimento de sair do jogo (ou alterar a janela) para ler o conteúdo em outra fonte no computador. Para sanar esse problema, foi criado uma nova tela chamada chatApp. A Figura 8 apresenta o chatApp, uma simulação de um aplicativo de conversação instantânea e que pode ser utilizado para buscar informações sobre os conteúdos trabalhados no cenário. Nessa interface, o jogador pode selecionar dúvidas comuns em relação ao conteúdo da fase que está jogando. Ao contrário do email e das dicas dadas na tela da IDE, o chatApp auxilia apenas sobre informações do conteúdo da disciplina, apresentando uma pequena descrição seguida de um exemplo.

Figura 9 – Tela IDE da Fase Alfa

Fonte: autor (2023)

Por fim, a interface de desenvolvimento, chamado no jogo de Super C IDE, foi alterada para suportar as anotações com dicas. Na Figura 9 é possível observar que a área de saída passou para o rodapé da tela. Outra alteração realizada ocorre na verificação do código, onde os erros são detalhados conforme uma lista de erros mais comuns apresentados em (BRAGA, 2020).

3. Fase Beta: fase apresentada para os testes finais e serve para validar a questão de pesquisa. Essa versão aborda todo o conteúdo planejado e também possui a revisão de problemas técnicos e de conteúdo levantados na fase Alfa.

4. Fase Gold: representa a fase utilizada para os testes finais do jogo com o público alvo. Essa é a versão com o maior número de correções e alterações e serve como o produto final da pesquisa. Após o teste aplicado, o jogo passa para a fase de pós-produção.

6.4.4 Pós-Produção

Esta fase ocorre ao final do período de desenvolvimento e testes, passando a ser visto como um produto finalizado. Nesta fase, as últimas alterações no projeto são aplicadas e todos

os erros identificados pelos jogadores são corrigidos. Assim, podendo encerrar o projeto e disponibilizá-lo para um público mais amplo, fora da pesquisa.

6.5 Avaliação do CodingJob

Para a avaliação do jogo, foi realizado um conjunto de testes, sendo o primeiro utilizado como análise de requisitos para construção da primeira versão do jogo. Esse primeiro teste utilizou os questionários disponibilizados nos anexos 1 e 2. Os testes seguintes foram realizados conforme a metodologia DSR, sendo o jogo testado com os alunos e incrementado para cada teste posterior. Além dos testes com o jogo, os alunos devem responder um questionário antes de testar o jogo e um após o teste. Ambos questionários se encontram nos anexos 3 e 4 respectivamente.

Inicialmente, o jogo foi planejado para utilização em sala de aula, com apoio do professor respondendo dúvidas pontuais sobre o conteúdo, mas durante o período de aulas remotas o jogo foi repensado para ser utilizado como uma ferramenta com mais recursos de apoio de forma individual, como por exemplo a utilização da opção de trazer informações do conteúdo através da interação com personagens do jogo. Finalmente, o jogo foi testado em sala de aula após o período de isolamento social, misturando ambas características, abrindo uma possibilidade do aluno utilizar o jogo apenas utilizando as ferramentas disponíveis no jogo para tirar dúvidas em relação ao conteúdo ou requisitando auxílio para o pesquisador e para o professor em sala de aula em caso de alguma dúvida em relação ao jogo.

7 ANÁLISE DOS RESULTADOS

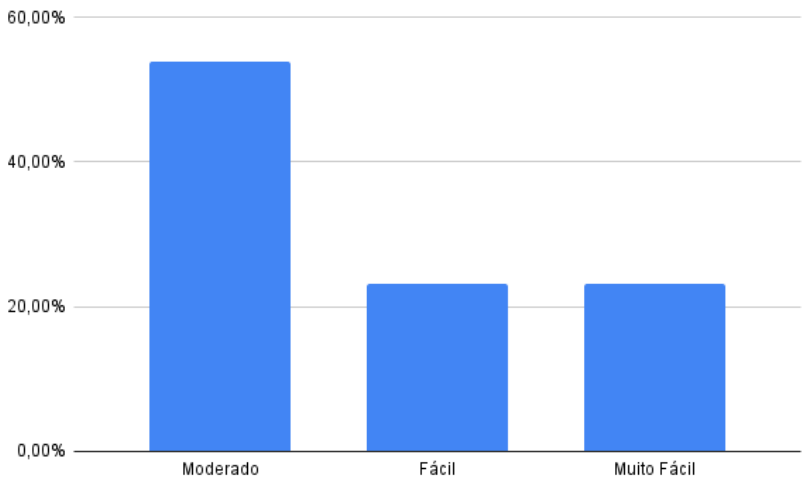
Seguindo o processo estabelecido pela metodologia DSR, o projeto foi testado em ciclos compostos por testes com alunos, análise de dados e reestruturação do jogo. Os ciclos de testes ocorreram da seguinte forma:

- Os alunos receberam o primeiro questionário que tinha por objetivo obter informações relacionadas a performance do aluno na disciplina. O questionário investigou quais os pontos mais sensíveis em relação ao aproveitamento do aluno, observando quais conteúdos eram problemáticos na disciplina;
- O jogo foi apresentado aos alunos. Estes tinham um determinado tempo (conforme a disponibilidade do professor) para testarem o jogo de forma livre;
- Um segundo questionário foi aplicado ao final do período de testes, buscando identificar as experiências dos alunos com o jogo. Os alunos poderiam dar sugestões de alterações. Estas foram analisadas e as consideradas relevantes para o contexto do jogo foram implementadas na versão seguinte.

7.1 Resultados do Primeiro Teste

O primeiro teste foi aplicado com 13 alunos da disciplina de Estrutura de Dados, do curso Superior de Sistemas para Internet (SSI) do IFRS. A disciplina de Linguagem de Programação I (LPI) é pré-requisito para a disciplina de Estrutura de Dados, ou seja, todos os alunos que participaram do teste haviam sido aprovados em LPI.

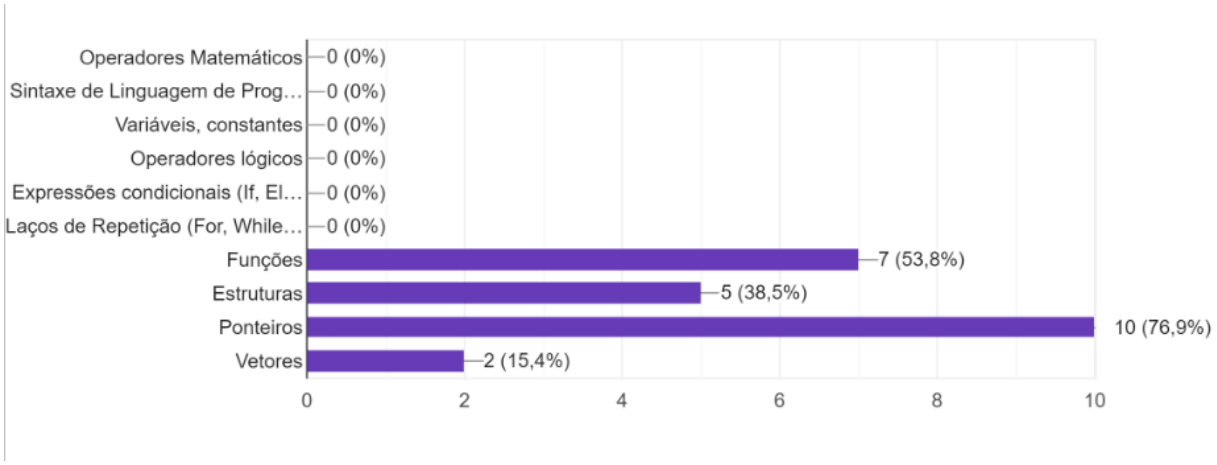
Figura 10 – Teste 1 - Percepção da disciplina de Linguagem de Programação I



Fonte: autor (2023)

Observa-se na Figura 10 que mais da metade dos alunos considerava a disciplina de dificuldade moderada.

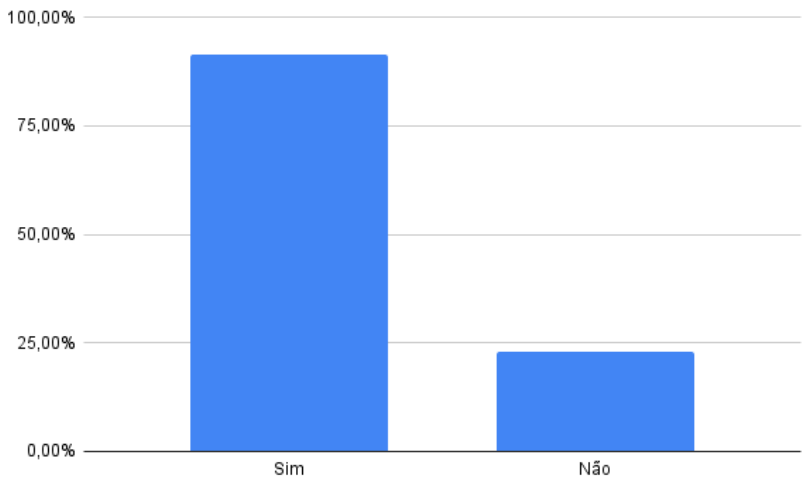
Figura 11 – Teste 1 - Conteúdos considerados de maior dificuldade



Fonte: autor (2023)

A Figura 11 apresenta os conteúdos em que os alunos consideravam de maior dificuldade. Observa-se que o conteúdo de ponteiros, geralmente apresentado no final da disciplina de LPI e de extrema importância para a disciplina de Estrutura de Dados, foi considerado o mais complexo. Em relação aos outros conteúdos apontados como de maior dificuldade de compreensão, é importante notar que nenhum foi abordado na versão aplicada neste teste, já que o foco de CodingJob é nos conteúdos iniciais da disciplina, considerados estrutura básica para programação.

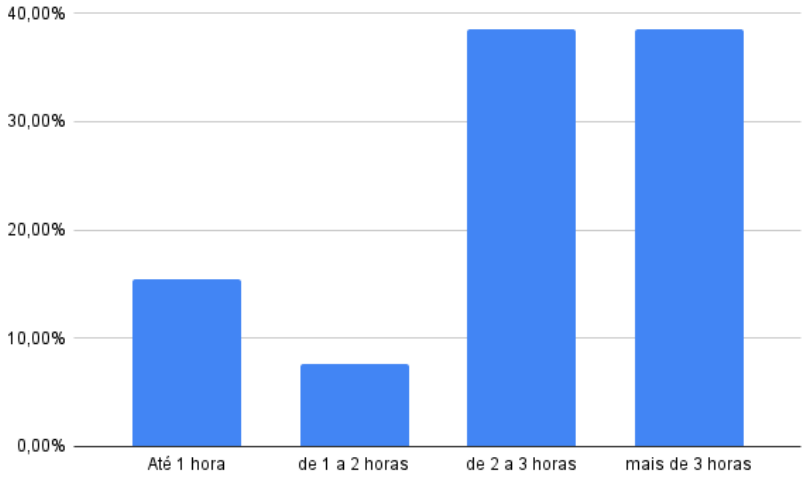
Figura 12 – Teste 1 - Alunos com hábito de jogar



Fonte: autor (2023)

A Figura 12 apresenta o resultado em relação ao hábito dos alunos de jogar jogos digitais.

Figura 13 – Teste 1 - Horas semanais de estudo

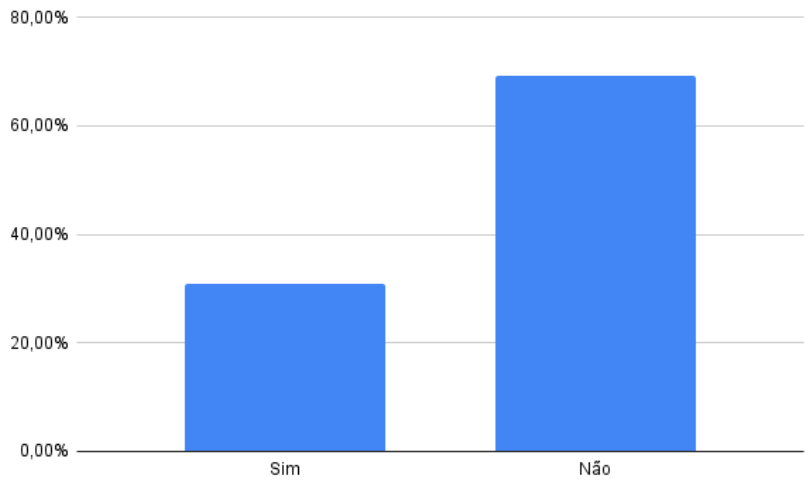


Fonte: autor (2023)

De acordo com o resultado apresentado na Figura 13, a maior parte da turma possui um regime de pelo menos 2 horas de estudo semanal aplicado em conteúdos de programação. Esse resultado é maior do que o esperado pelo autor, levando em consideração que parte dos alunos já está inserido no mercado de trabalho.

Na turma de Estrutura de Dados, todos os alunos que responderam ao questionário consideram ser possível aprender algum conteúdo através de jogos digitais. Nas justificativas, muitos alunos responderam ter aprendido conhecimentos de lingua estrangeira, história e até raciocínio lógico, através de jogos digitais.

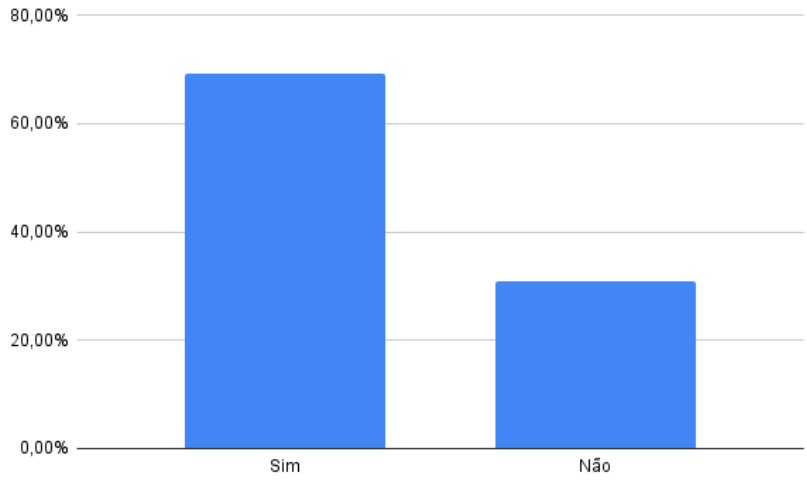
Figura 14 – Teste 1 - Experiência em desenvolvimento de software



Fonte: autor (2023)

A Figura 14 mostra que 30,8% dos alunos têm alguma experiência profissional com desenvolvimento de software. Acredita-se que esses alunos tenham melhor desempenho tanto no jogo aplicado quanto nas disciplinas de programação.

Figura 15 – Teste 1 - O jogo deve simular o ambiente de trabalho



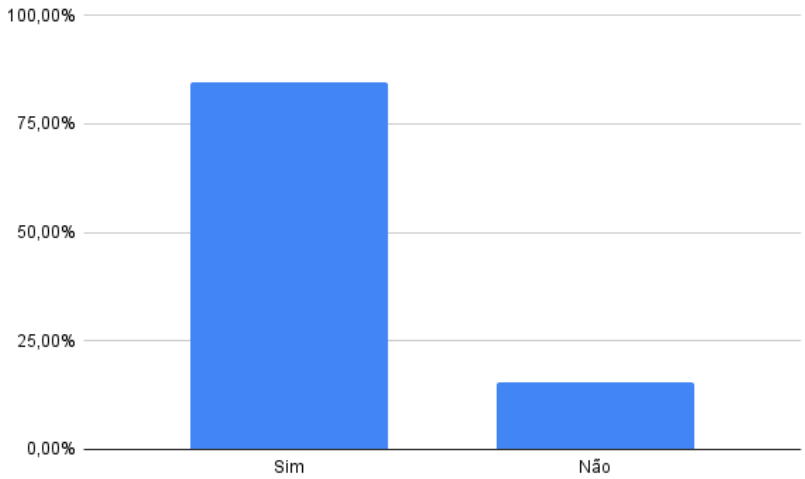
Fonte: autor (2023)

Conforme o resultado apresentado na Figura 15, 69,2% dos alunos considera importante que um jogo para o ensino de programação simule o ambiente de trabalho de um programador. Essa resposta foi positiva para todos os alunos que já trabalham na área.

Conversando informalmente com os alunos que responderam não para a questão, o autor observou que existe uma expectativa por jogos mais lúdicos e menos centrados no conteúdo, o que seria interessante para um cenário de educação não formal.

Após a aplicação desse questionário, os 13 alunos da disciplina de Estrutura de Dados testaram o jogo por aproximadamente 01 hora. As respostas a seguir foram obtidas através do segundo questionário aplicado após os alunos utilizarem o jogo.

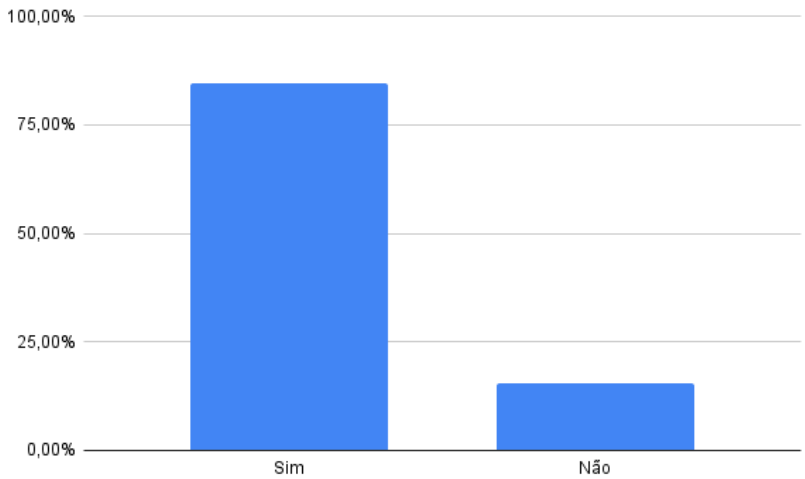
Figura 16 – Teste 1 - Utilização de um jogo para o ensino de programação



Fonte: autor (2023)

O resultado ilustrado na Figura 16 mostra que 84,6% da turma já utilizou jogos com objetivo de ensinar algum conteúdo de programação.

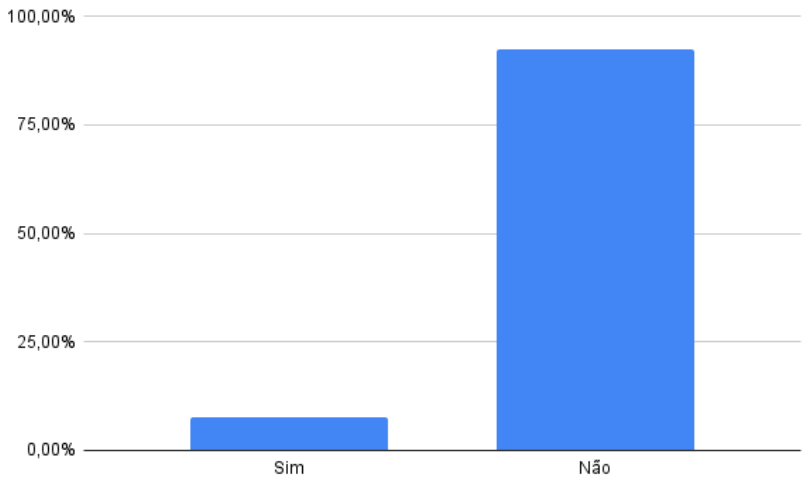
Figura 17 – Resultado da avaliação quanto à adequação dos exercícios - Teste 1



Fonte: autor (2023)

Observa-se no resultado apresentado na Figura 17 que a maior parte dos alunos consideram que os exercícios do CodingJob são adequados para estudo.

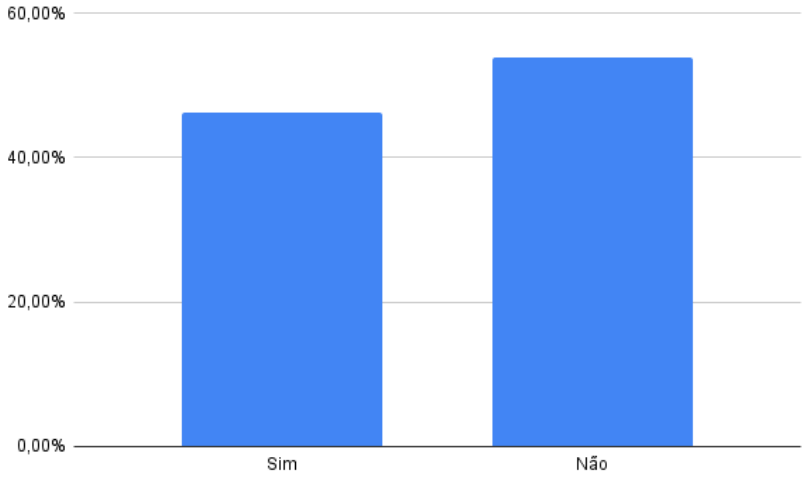
Figura 18 – Resultado referente a conclusão dos exercícios - Teste 1



Fonte: autor (2023)

A Figura 18 mostra que apenas 7,7% dos alunos encerrou todos os exercícios de CondingJob. Esse resultado já era esperado, considerando o número de exercícios aplicados e o tempo disponibilizado para os alunos.

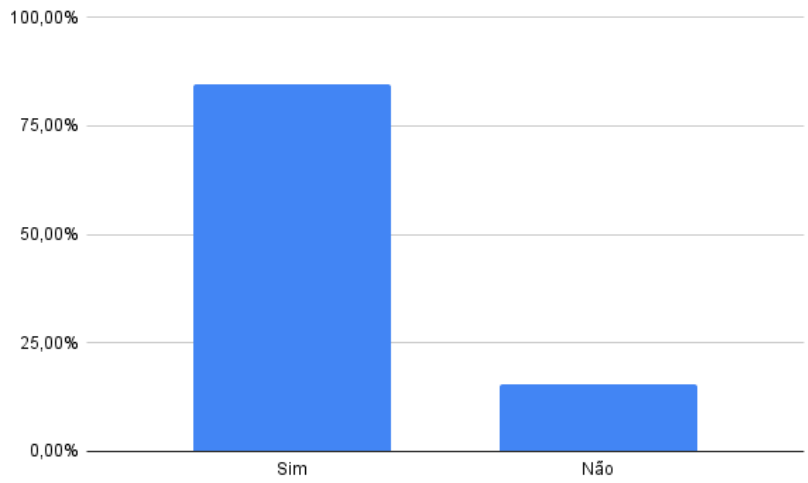
Figura 19 – Resultado da avaliação da motivação em realizar os exercícios - Teste 1



Fonte: autor (2023)

Na Figura 19 observa-se que 46,2% dos alunos se sentiram motivado para realizar os exercícios apresentados em CodingJob. Acredita-se que esse resultado seja pelo fato de que alguns exercícios possam ter confundido os alunos, levando a um número alto de tentativas incorretas. Isso pode ser comprovado após a análise dos resultados, onde alguns alunos repetiram o mesmo exercício quase 30 vezes sem sucesso.

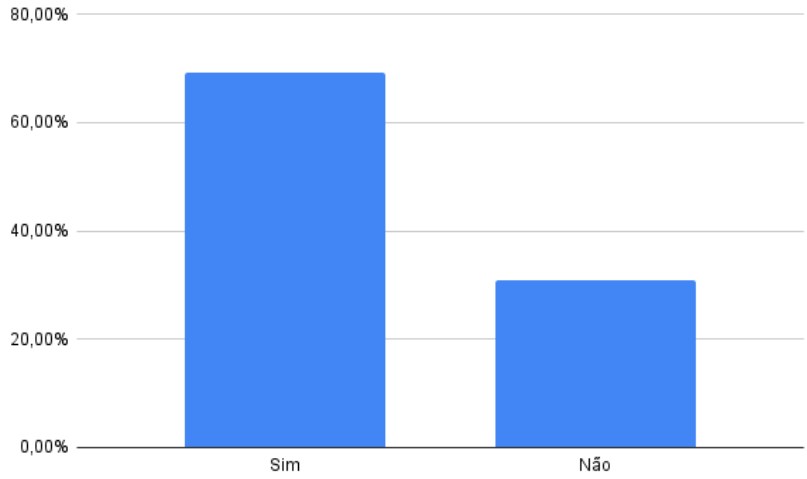
Figura 20 – Resultado da avaliação do personagem Haskell - Teste 1



Fonte: autor (2023)

O resultado ilustrado na Figura 20 mostra que 84,6% consideraram que os emails de dicas do personagem Haskell auxiliou de alguma forma da resolução dos exercícios.

Figura 21 – Resultado da avaliação das dicas do personagem Pascal - Teste 1



Fonte: autor (2023)

De acordo com a Figura 21, 69.2% conseguiram alguma ajuda com as informações apresentadas pelo personagem Pascal. Conforme relatado por alguns alunos nas sugestões, o menu de acesso para as dicas de Pascal ficava longe da área de jogo, assim prejudicando a concentração de alguns alunos.

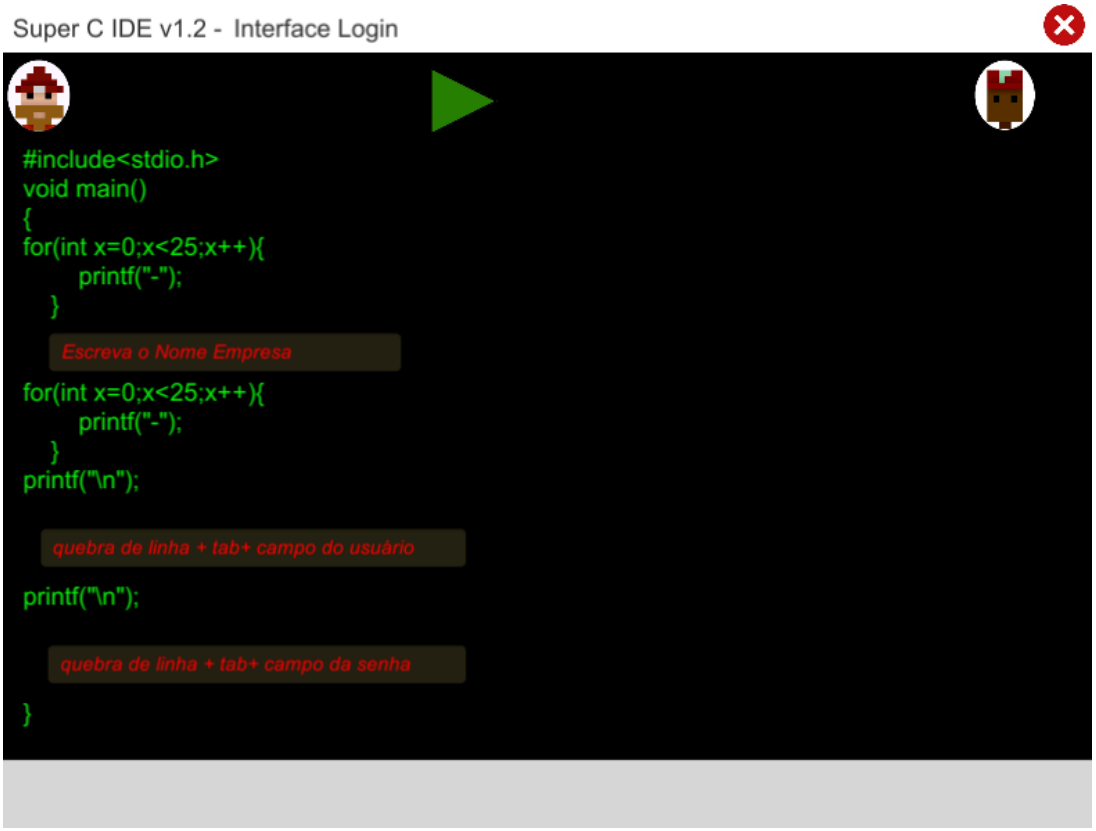
7.1.1 Sugestões e Comentários

No primeiro teste com o jogo CodingJob foram levantadas diversas sugestões de aprimoramento, tanto do conteúdo, quanto da interface. As principais correções relacionadas ao conteúdo foram nos textos apresentados nos emails e no próprio corpo do código.

As alterações realizadas na interface foram mais drásticas e alteraram a apresentação de vários elementos do jogo. A primeira alteração foi a do ícone de acesso ao texto do email de Haskell. Durante os testes, os alunos relataram que o ícone não era intuitivo, sendo que poucos compreenderam que ele simbolizava um bloco de anotações, representando as anotações do personagem do jogador (Adan). Sendo assim, os alunos sugeriram utilizar um ícone com a face do personagem Haskell.

Outra alteração proposta pelos alunos foi a de mover o conteúdo do personagem Pascal para dentro da IDE do jogo. Inicialmente, o acesso ao conteúdo de Pascal era disponibilizado apenas na interface do sistema operacional, sendo necessário o aluno sair da IDE (onde a resolução do problema se passa) e retornar para a tela anterior. A alteração implementada insere a interface do chatApp de Pascal para a IDE, reduzindo a mudança de telas. Para manter o padrão já estabelecido pelo botão de Haskell, um ícone com a face de Pascal foi inserido na parte superior da tela.

Figura 22 – Alterações realizadas na interface do CodingJob



Fonte: autor (2023)

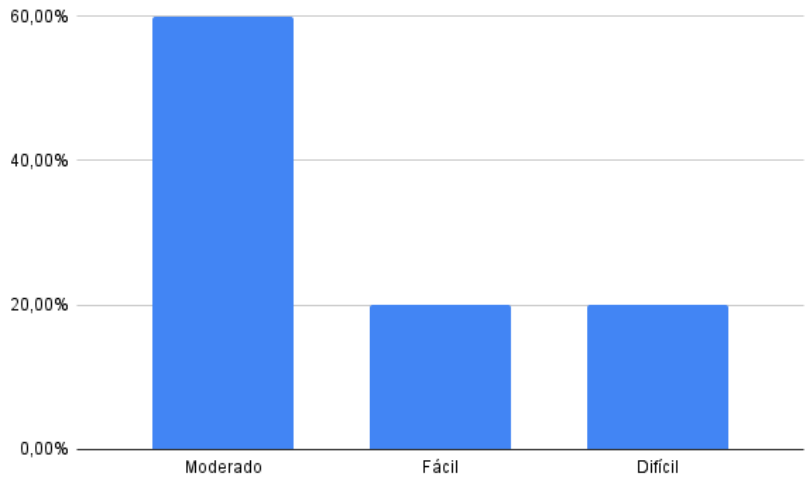
Na Figura 23 observa-se as alterações visuais aplicadas após o primeiro teste. É possível notar a alteração do ícone do acesso aos emails do personagem Haskell. No lado oposto o botão para acesso ao chatApp do personagem Pascal.

7.2 Resultados do Segundo Teste

O segundo teste foi aplicado duas semanas após a aplicação do primeiro teste com cinco alunos de uma turma de Engenharia de Software do curso SSI do IFRS. Esses alunos já haviam cursado a disciplina de LPI e todos tinham experiência profissional no desenvolvimento de software.

O teste foi conduzido na mesma forma que o primeiro, sendo inicialmente apresentado um formulário para obter uma visão geral do participante e depois um período para testes, seguido pela aplicação do segundo formulário composto por questões para a avaliação do CodingJob.

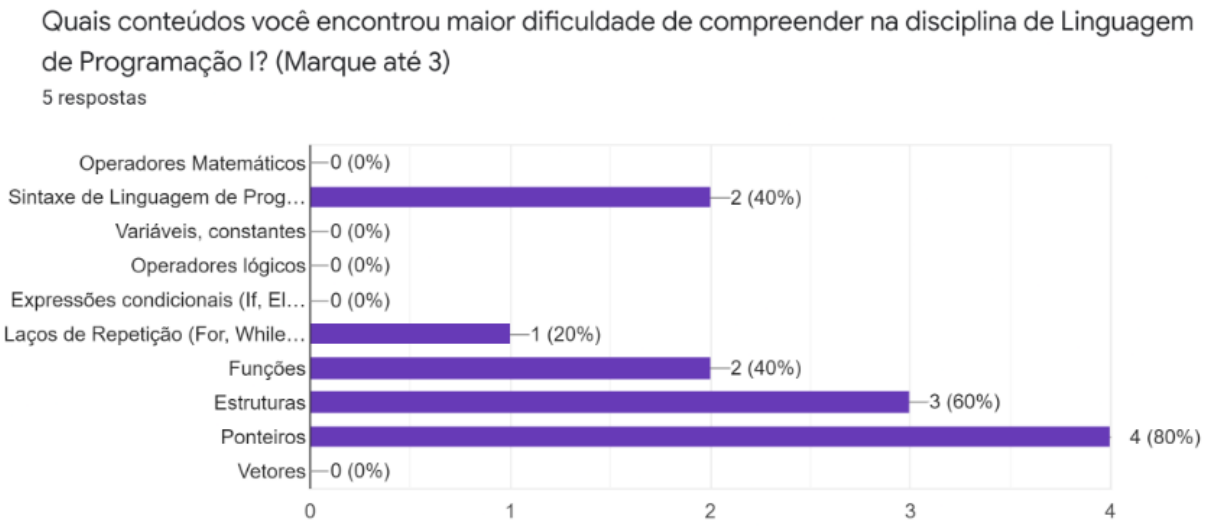
Figura 23 – Avaliação da percepção da disciplina de LPI - Teste 2



Fonte: autor (2023)

No resultado apresentado na Figura 23 observa-se que, assim como a Turma 1, a Turma 2 avaliou a disciplina de LPI como teve uma percepção moderada da dificuldade da disciplina de Linguagem de Programação 1. Considerando que o conteúdo aplicado é o mesmo, o resultado apresentado era esperado.

Figura 24 – Teste 2 - Conteúdos Mais Problemáticos para os Alunos

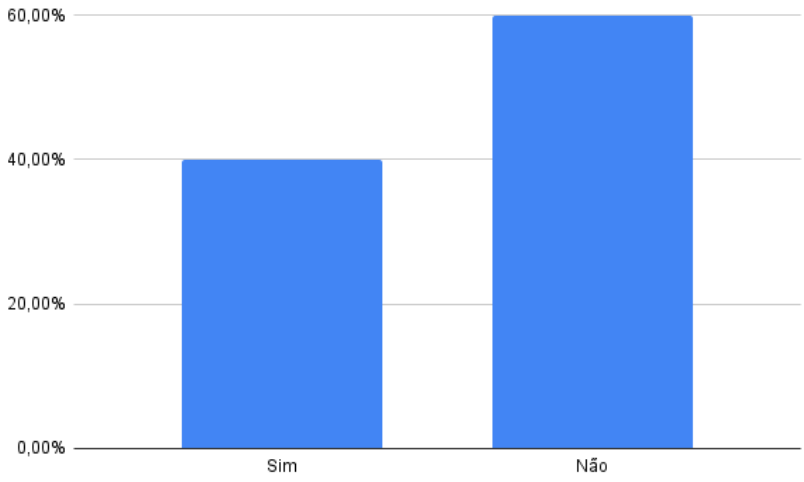


Fonte: Do Autor (2023)

Na Figura 24 é possível observar que ponteiros e estruturas foram os conteúdos com maior dificuldade de compreensão pelos alunos. É possível que esses conteúdos apresentem maior dificuldade pela sua abstração mais complexa e por serem conteúdos que geralmente ficam de fora da disciplina inicial de algoritmos.

Todos os alunos do teste 2 afirmaram possuir o hábito de jogar jogos digitais. Os mesmos alunos afirmam que acreditam ser possível aprender conteúdos de programação através de jogos digitais.

Figura 25 – Teste 2 - Experiência em Desenvolvimento de Software

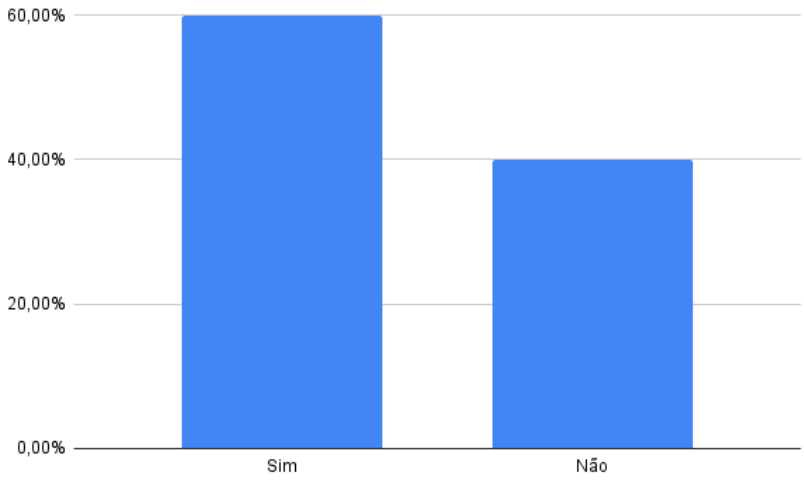


Fonte: Do Autor (2023)

Em relação aos alunos que já tiveram alguma experiência profissional como programadores, como pode ser visto pela Figura 25, 60% dos alunos não possuem experiência de

mercado.

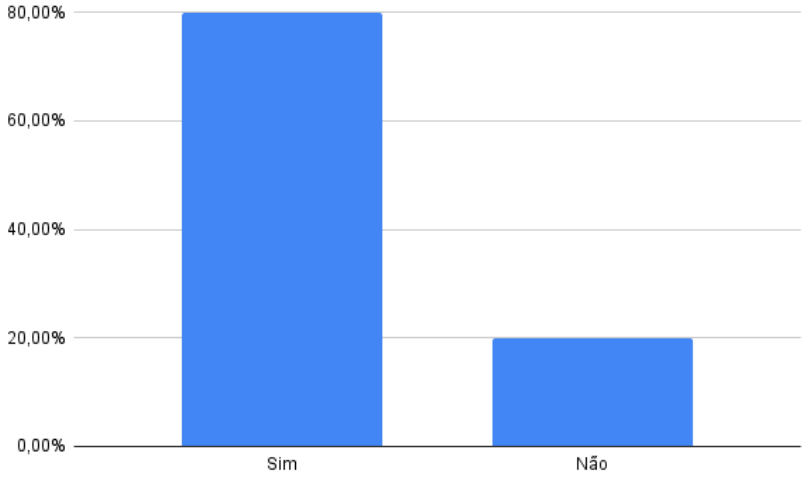
Figura 26 – Teste 2 - O Jogo deve Simular um Ambiente de Trabalho



Fonte: Do Autor (2023)

Segundo os resultado das avaliações apresentado na Figura 26, os alunos da disciplina de Engenharia de Software tem uma leve preferencia pela utilização de um cenário baseado em ambiente de trabalho da área de desenvolvimento de software.

Figura 27 – Teste 2 - Utilização de um Jogo para Ensino de Programação

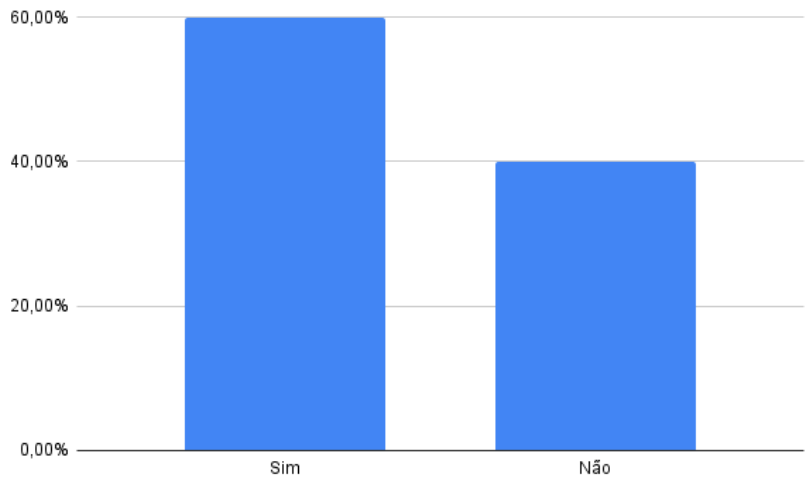


Fonte: Do Autor (2023)

A Figura 27, é identificado que a maior parte da turma já utilizou algum jogo que tem como objetivo ensinar conceitos de programação. Essa informação segue sendo uma surpresa, devido aos jogos que focam nesse conteúdo ainda não serem tão comuns.

Após a aplicação do questionário inicial, os alunos testaram o jogo CodingJob por alguns minutos dentro do horário da aula. A versão do jogo utilizada foi uma atualização da versão aplicada pela turma anterior. Correções na interface e nos textos foram implementadas, assim como algumas alterações propostas pelos jogadores da primeira sessão de testes. Nenhum aluno conseguiu terminar todas as questões do jogo durante o teste.

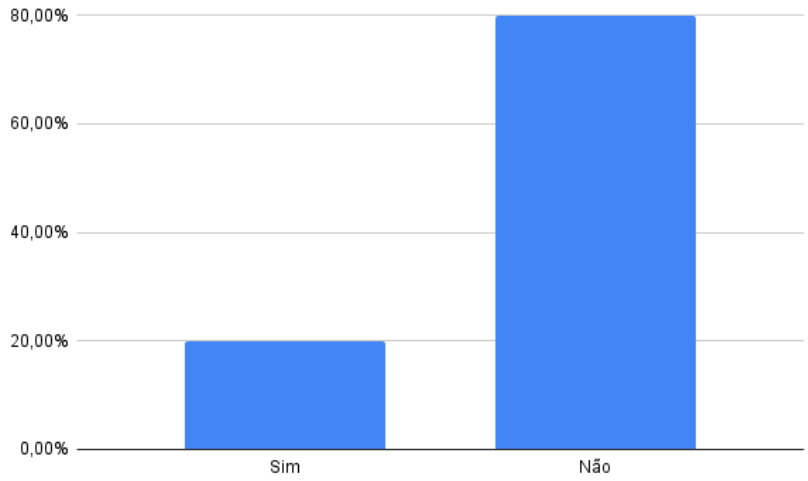
Figura 28 – Teste 2 - Exercícios Apresentados de Forma Adequada



Fonte: Do Autor (2023)

Na Figura 28, 60% dos alunos que utilizaram a versão atualizada de CodingJob consideraram que os exercícios foram aplicados da forma correta pelo jogo.

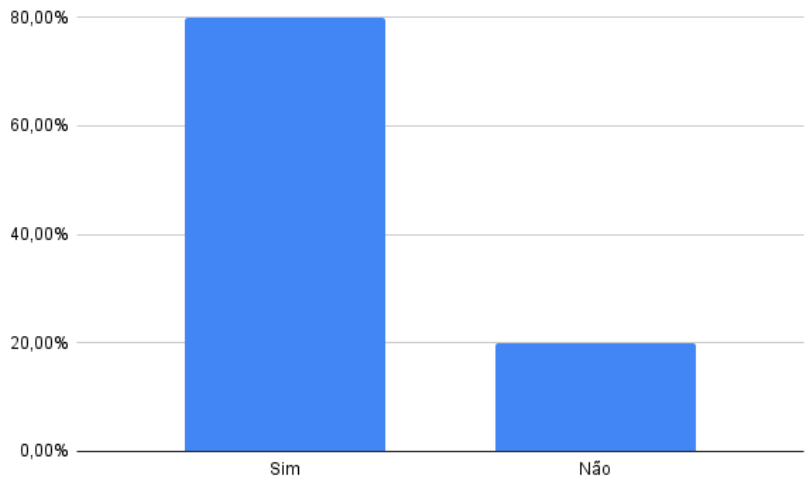
Figura 29 – Teste 2 - Motivação para Realizar os Exercícios do Jogo



Fonte: Do Autor (2023)

Os participantes do teste 2 se sentiram menos motivados em realizar os exercícios propostos pelo jogo. Segundo a figura 29, apenas 20% sentiram motivação para seguir utilizando o jogo.

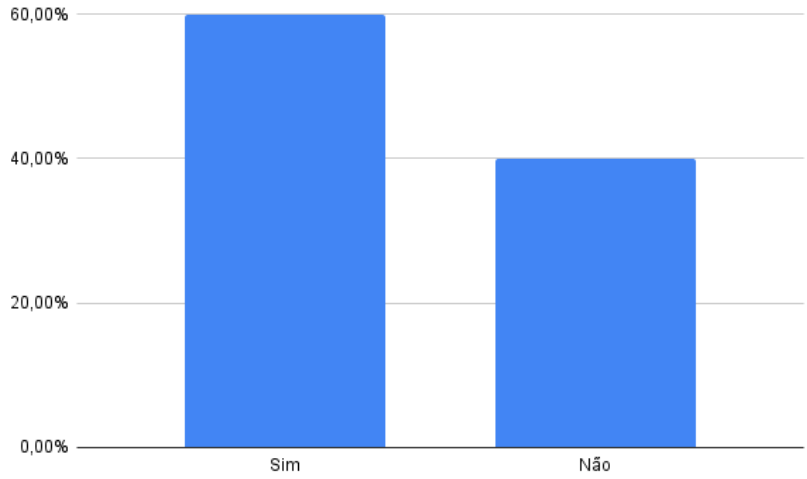
Figura 30 – Teste 2 - Dicas de Haskell Ajudaram o Aluno?



Fonte: Do Autor (2023)

Em relação aos personagens de apoio, como apresentado na figura 30, o personagem Haskell teve uma aprovação de 80% dos avaliadores do teste 2.

Figura 31 – Teste 2 - Dicas de Pascal Ajudaram o Aluno?



Fonte: Do Autor (2023)

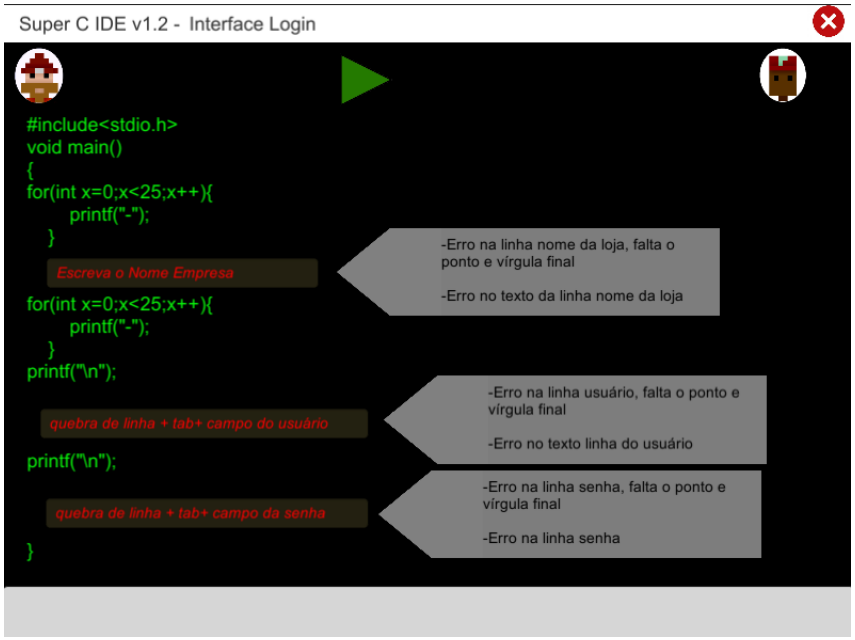
Já na figura 31, fica evidente que o personagem Pascal teve um menor aproveitamento do avaliadores, apenas 60%.

7.2.1 Sugestões e Comentários

O teste 2 apresentou comentários mais críticos em relação a interface do jogo, tanto na navegação dos menus quanto na apresentação dos exercícios. Uma sugestão implementada para o terceiro teste foi a inclusão de todos os erros de sintaxe existentes em uma linha. A justificativa

para essa alteração foi para tentar reduzir a quantidade de erros cometidos numa mesma linha, permitindo que o aluno identifique dois ou mais erros previstos pelo jogo.

Figura 32 – Alterações após a segunda avaliação



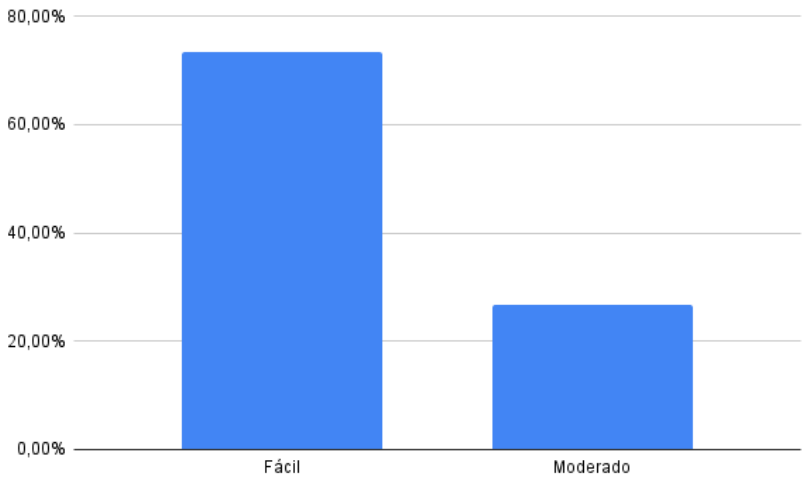
Fonte: Do Autor (2023)

A interface de identificação de erros foi aprimorada, de modo a apresentar os erros em sequência ao lado ou abaixo da linha onde o erro ocorreu.

7.3 Resultados do Terceiro Teste

O terceiro teste ocorreu duas semanas após o segundo teste e foi aplicado em uma turma de Linguagem de Programação I, ou seja, o grupo idealizado pela pesquisa. Dentre dos alunos da turma, 18 responderam de forma completa os questionários apresentados. É importante salientar que parte da turma cometou um equivoco ao executar a versão atualizada do jogo, utilizando a versão apresentada no segundo teste. Apenas três usuários utilizaram a versão atualizada. O autor optou por dividir o terceiro teste em 2, sendo a primeira parte uma apresentação dos resultados dos 15 alunos que testaram a versão 2 e em seguida os outros alunos que testaram a versão correta.

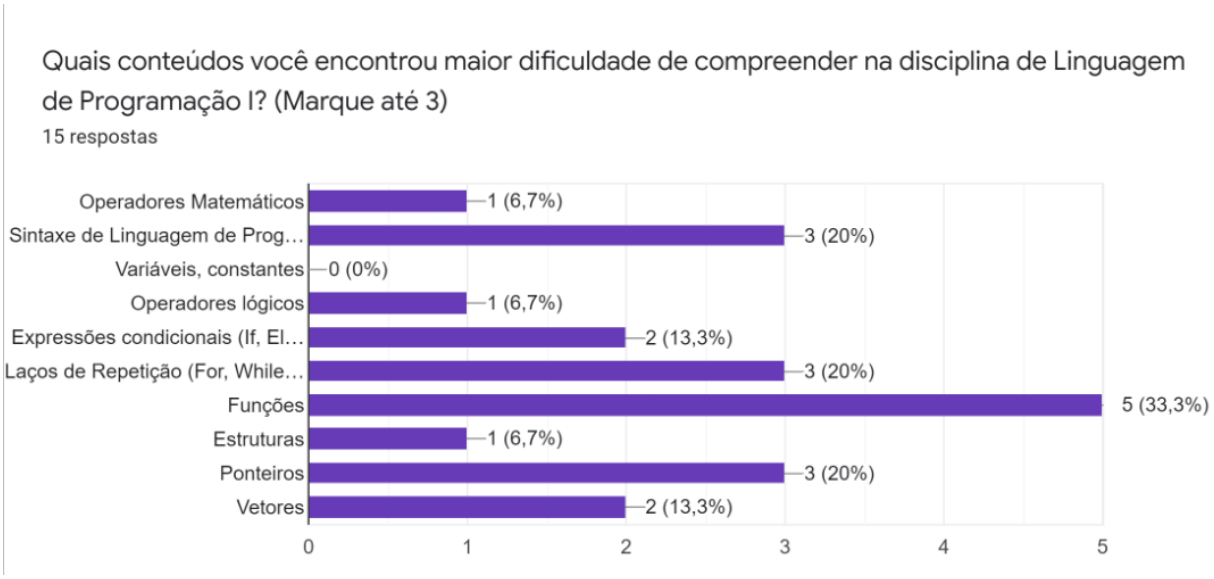
Figura 33 – Teste 3 - Percepção da Disciplina de Linguagem de Programação I



Fonte: Do Autor (2023)

Na Figura 33, podemos observar que a turma tem encontrado uma dificuldade moderada na disciplina. É importante notar que esses alunos estão na metade do semestre e sendo assim, a grande maioria ainda não chegou as conteúdos mais complexos da disciplina, como ponteiros e estruturas.

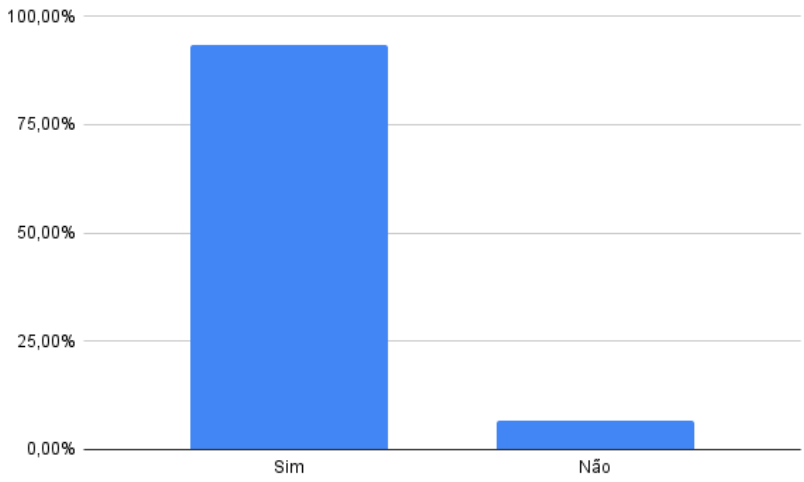
Figura 34 – Teste 3 - Conteúdos Mais Problemáticos para os Alunos



Fonte: Do Autor (2023)

Já a Figura 34 apresenta quais conteúdos os alunos possuem maior dificuldade. O conteúdo mais complexo para os alunos é o de funções, seguido da sintaxe da linguagem C, For/While e Ponteiros. É possível notar que mesmo no meio do semestre, os alunos marcaram vários conteúdos mais avançados da disciplina. O autor suspeita que isso ocorra devido a alguns alunos já tenham participado da disciplina e tenham falhado.

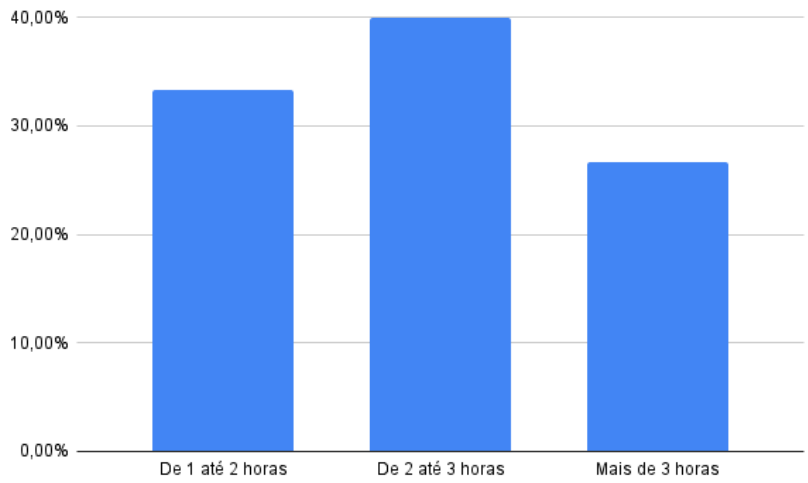
Figura 35 – Teste 3 - Alunos com Hábito de Jogar



Fonte: Do Autor (2023)

Novamente, a imensa maioria dos alunos tem o hábito de jogar algum jogo digital. Conforme a Figura 35, 93,3% dos alunos.

Figura 36 – Teste 3 - Total de Horas de Estudo por Semana

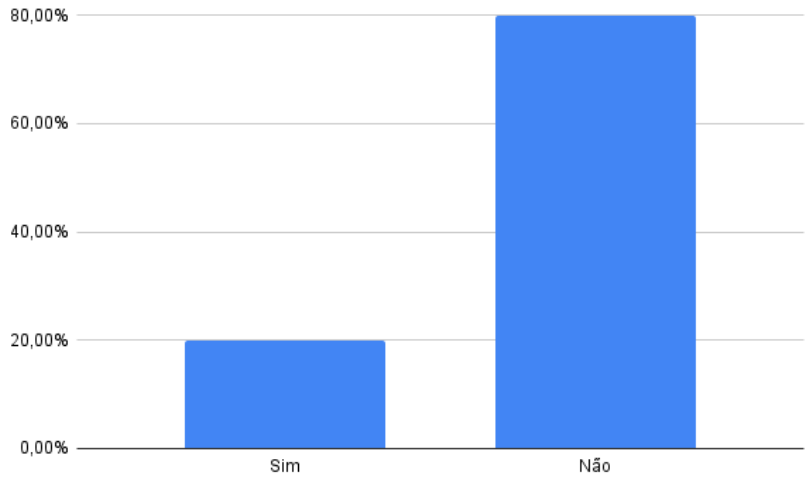


Fonte: Do Autor (2023)

Na figura 36, observamos que a maior parte dos alunos estudam o conteúdo de 2 a 3 horas semanais.

Todos os alunos do teste acreditam que é possível aprender algum conteúdo através de jogos digitais.

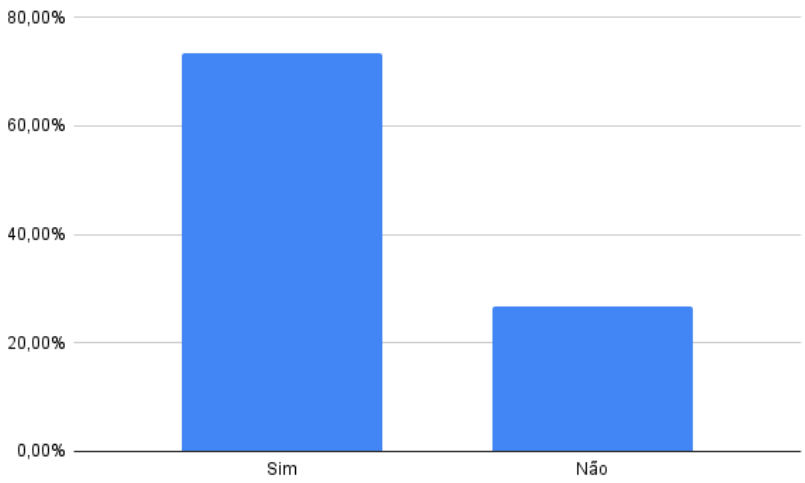
Figura 37 – Teste 3 - Experiência em Desenvolvimento de Software



Fonte: Do Autor (2023)

Já na Figura 37, apenas 20% possuem alguma experiência profissional em desenvolvimento de software.

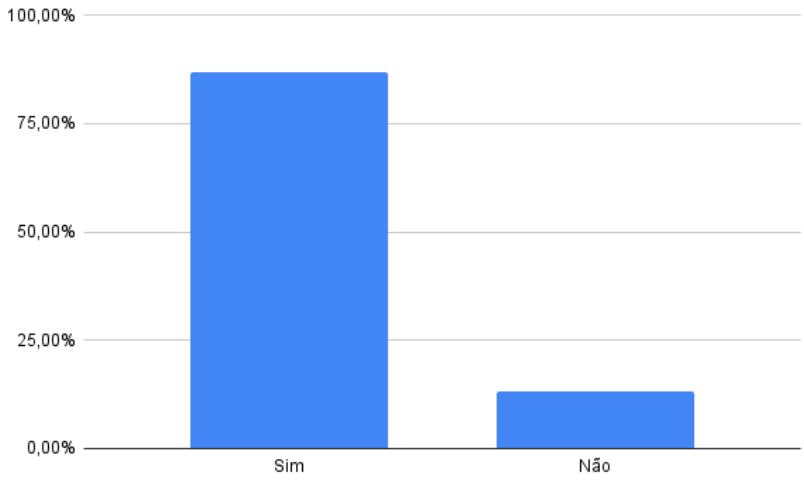
Figura 38 – Teste 3 - O Jogo deve Simular um Ambiente de Trabalho



Fonte: Do Autor (2023)

Na Figura 38, a maior parte dos alunos afirmam que a utilização de um cenário de desenvolvedor é importante para a experiência de jogo. Para o autor, foi uma surpresa, ja que contradiz os resultados apresentados em outros testes.

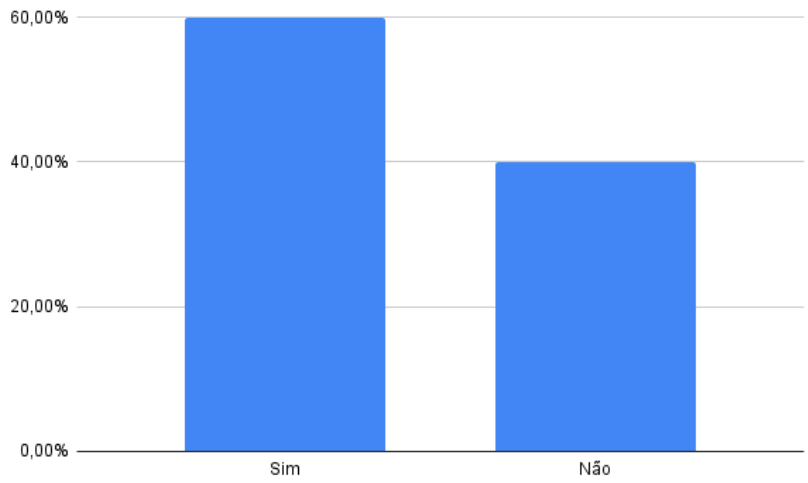
Figura 39 – Teste 3 - Utilização de um Jogo para Ensino de Programação



Fonte: Do Autor (2023)

Na Figura 39, 86,7% dos alunos afirmam que já utilizaram algum jogo para aprender algum conteúdo de programação.

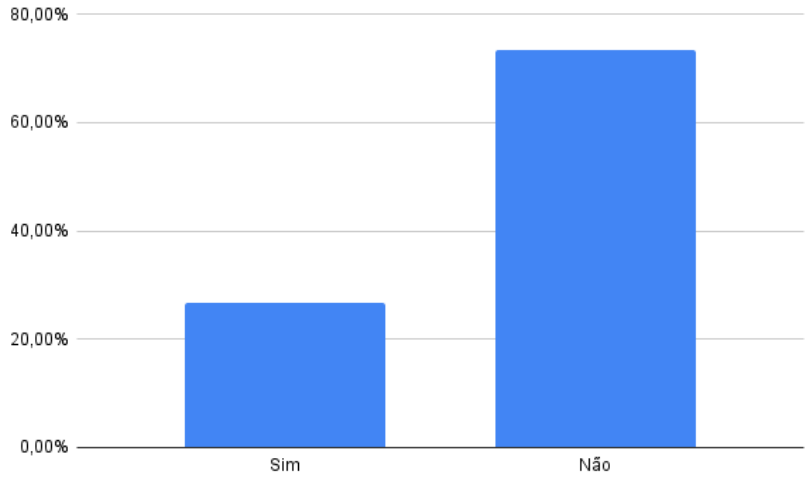
Figura 40 – Teste 3 - Exercícios Apresentados de Forma Adequada



Fonte: Do Autor (2023)

Na Figura 40, 60% dos alunos afirmam que acham os exercícios adequados para aprender o conteúdo de Linguagem de Programação I.

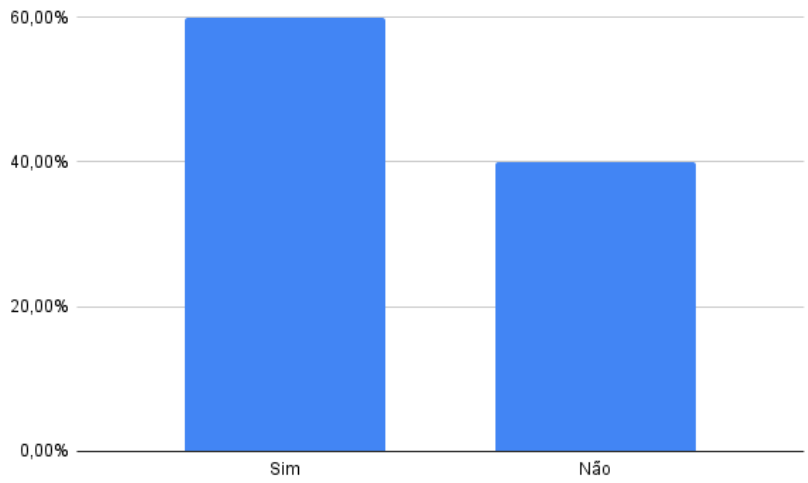
Figura 41 – Teste 3 - Conclusão dos Exercícios do Jogo



Fonte: Do Autor (2023)

Na Figura 41 é possível observar que apenas 73,3% dos alunos não conseguiram encerrar o jogo. Novamente, esse valor é previsivel devido ao tempo disponibilizado para encerrar o jogo.

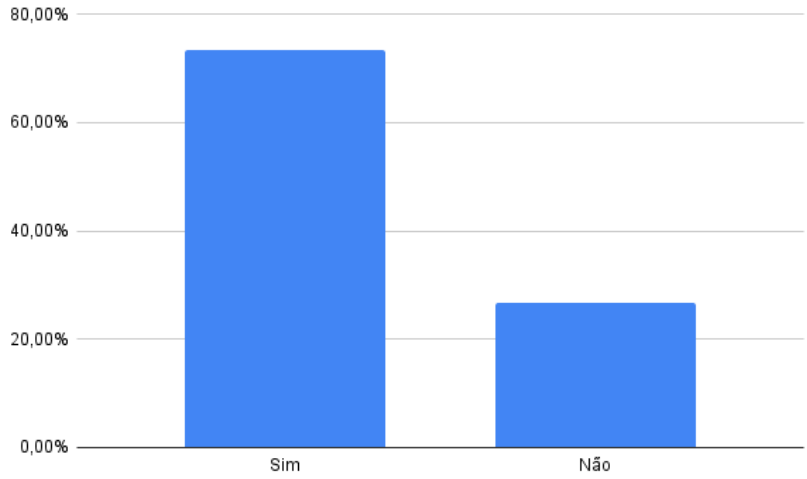
Figura 42 – Teste 3 - Motivação para Realizar os Exercícios do Jogo



Fonte: Do Autor (2023)

Na Figura 42, 60% dos alunos se sentiu motivados ao realizar os exercícios propostos pelo jogo.

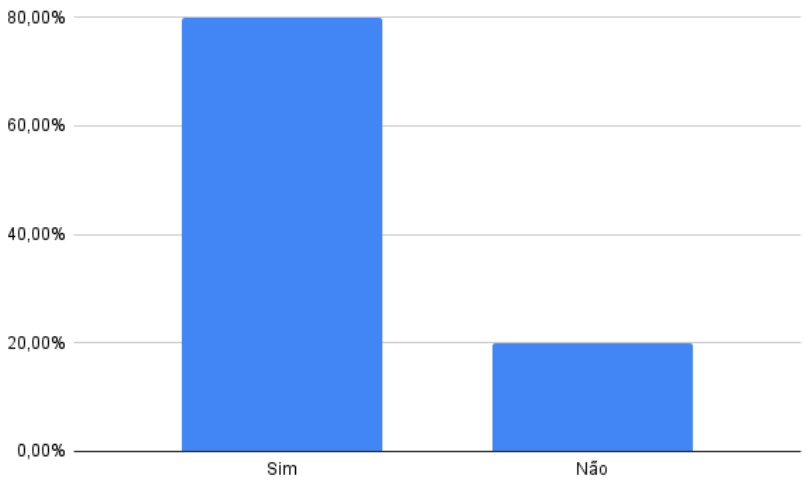
Figura 43 – Teste 3 - Dicas de Haskell Ajudaram o Aluno?



Fonte: Do Autor (2023)

Na Figura 43, 73,3% dos alunos conseguiram alguma ajuda através dos textos apresentados pelo personagem Haskell.

Figura 44 – Teste 3 - Dicas de Pascal Ajudaram o Aluno?



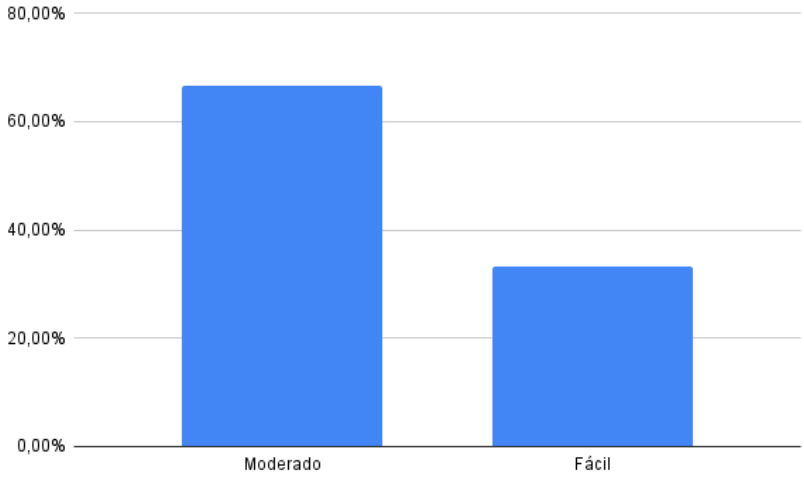
Fonte: Do Autor (2023)

Na Figura 44, 80% dos alunos conseguiram alguma ajuda através dos textos apresentados pelo personagem Pascal.

7.4 Resultados do Terceiro Teste - Utilização da Versão Correta

Como explicado no item anterior, apenas um grupo de 3 alunos utilizaram a versão modificada da versão 2 para a 3. Nessa versão existem atualizações dos textos dos personagens, melhor organização dos exercícios e uma correção de erros identificados no teste 2.

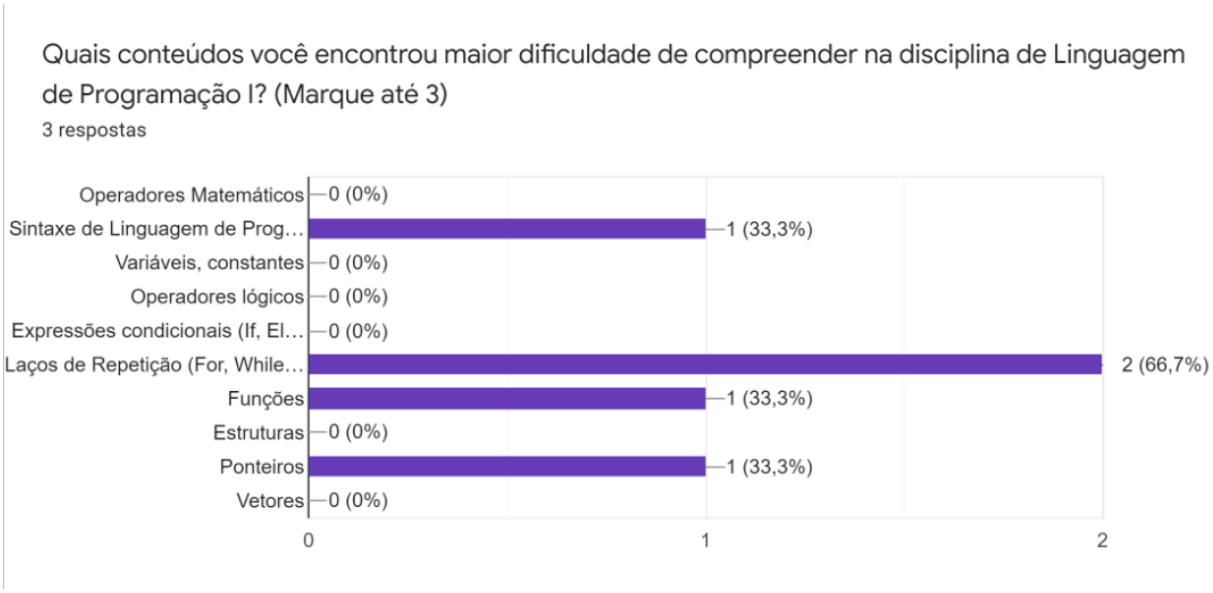
Figura 45 – Teste 3, versão correta - Percepção da Disciplina de Linguagem de Programação I



Fonte: Do Autor (2023)

Na Figura 45, é possivel observar que os alunos tem uma visão de dificuldade moderada para difícil da disciplina de Linguagem de Programação I.

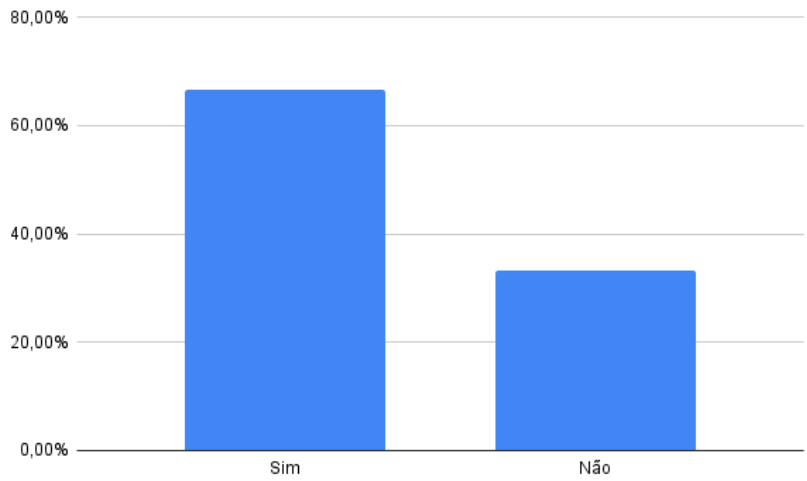
Figura 46 – Teste 3, versão correta - Conteúdos Mais Problemáticos para os Alunos



Fonte: Do Autor (2023)

Já a Figura 46 apresenta quais conteúdos foram percebidos como de maior dificuldade pelos alunos. Nessa avaliação, o conteúdo com percebepção mais problemática foi o de laços de repetição.

Figura 47 – Teste 3, versão correta - Alunos com Hábito de Jogar



Fonte: Do Autor (2023)

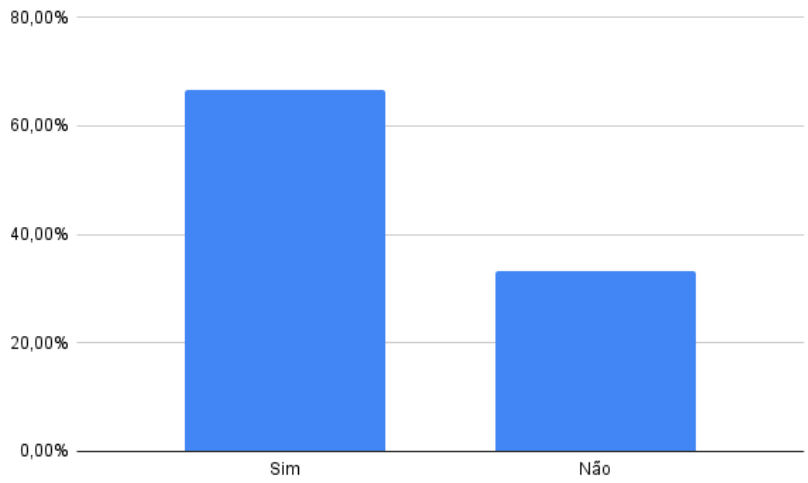
Na Figura 46, dos três alunos avaliados, dois possuem o hábito de jogar algum jogo digital no seu tempo livre.

Dentro do grupo de testes, todos os alunos afirmaram estudar programação em média de 1 a 2 horas semanais. O valor é baixo considerando os resultados obtidos em outros testes.

Semelhante aos outros testes, todos alunos acreditam que é possível aprender algum conteúdo através de jogos digiais. Os mesmos alunos afirmaram que já utilizaram algum jogo com objetivo de praticar conhecimentos de programação.

Na Figura 55, notamos que nesse teste, nenhum aluno tem experiência de trabalho em área de programação.

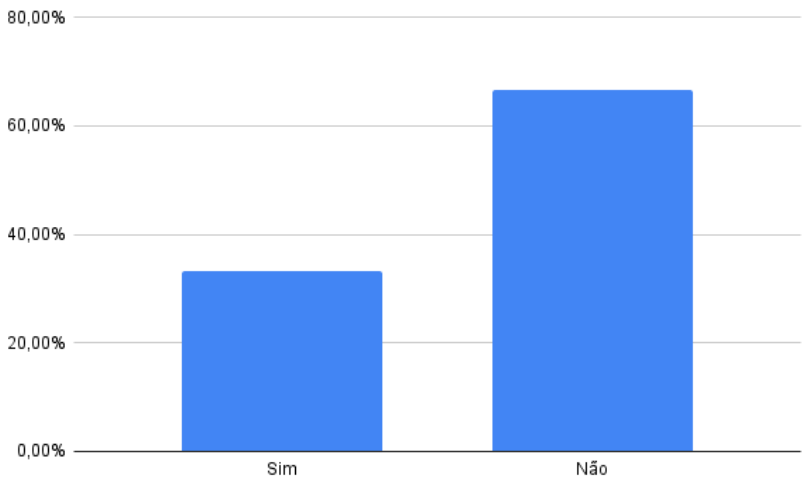
Figura 48 – Teste 3, versão correta - O Jogo deve Simular um Ambiente de Trabalho



Fonte: Do Autor (2023)

Na Figura 48, 2 alunos acham importante que o jogo simule um ambiente de trabalho de uma fábrica de software.

Figura 49 – Teste 3, versão correta - Conclusão dos Exercícios do Jogo

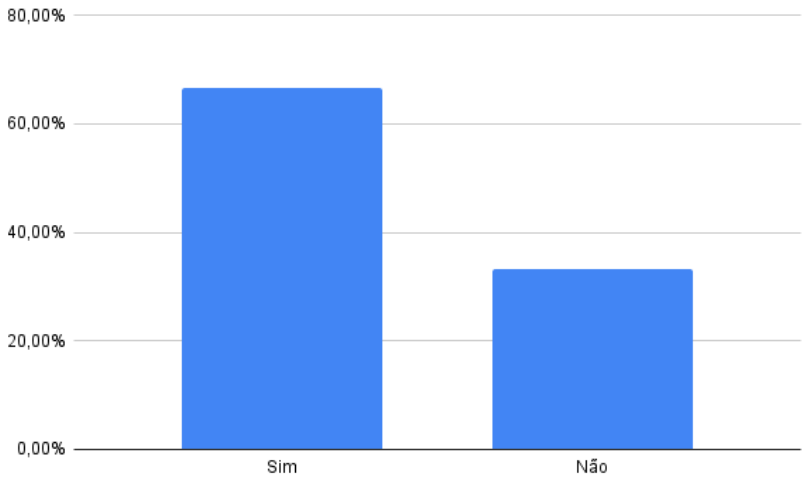


Fonte: Do Autor (2023)

Na Figura 49, observamos que apenas 1 dos alunos conseguiu encerrar todos os exercícios propostos. Todos os alunos que testaram esta versão afirmam que os exercícios do CodingJob foram adequados para praticar o conteúdo da disciplina de Linguagem de Programação I.

Em relação a motivação para realizar os exercícios propostos, todos os alunos se sentiram motivados com a versão atualizada.

Figura 50 – Teste 3, versão correta - Dicas de Haskell Ajudaram o Aluno?



Fonte: Do Autor (2023)

Na Figura 61, 2 alunos acharam as dicas do personagem Haskell positivas. Após a avaliação dos textos utilizados por Haskell, o autor notou que talvez seja necessário utilizar diagramas ou outros artifícios para chamar mais atenção do jogador para as tarefas aplicadas. Já o personagem Pascal auxiliou todos os alunos que participaram do teste.

7.4.1 Sugestões e Correções

Ao término da terceira sessão de testes, as sugestões dos alunos focaram principalmente na alteração de alguns conceitos mecânicos do jogo. A sugestão mais comum apresentada, era de um sistema de dicas após um número determinado de erros em uma questão. Outra sugestão que voltou a aparecer nos questionários, foi a de aumentar a carga narrativa do jogo (aumentar a historinha). No teste dos alunos que utilizaram a versão atualizada, as sugestões foram sobre melhorar as dicas, criar um sistema de progresso do personagem e um modo "difícil", onde o personagem poderia resolver os problemas apresentados de forma alternativa.

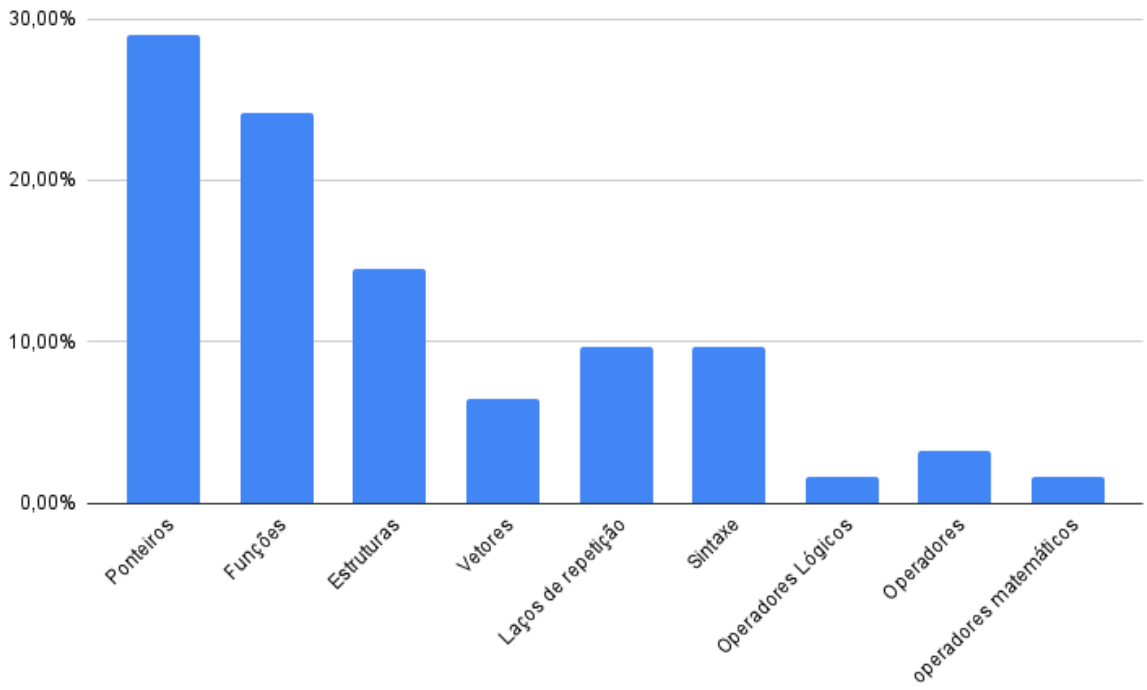
8 RESULTADOS E DISCUSSÃO

Após 3 sessões de testes com alunos do curso de Sistemas para Internet do IFRS, foi observado que o jogo obteve uma boa aceitação pelos alunos. O autor utilizou metodologia DSR para produção da pesquisa, sendo a metodologia focada na produção do produto final em pararelo a produção da pesquisa. O ciclo produzido pelo autor foi baseado em 3 pontos:

- Construção ou atualização de elementos do jogo com base na análise dos resultados de uma avaliação.
- Aplicação do jogo com alunos e a aplicação de 2 questionários, o primeiro para verificar o perfil dos alunos e o segundo sobre a experiência com o jogo CodingJob.
- Análise dos dados obtidos na aplicação do jogo e indentificação de problemas encontrados no jogo.

Dentro da análise de dados, foi observado diversos pontos que levantaram novos questionamentos do autor. Em relação ao perfil dos alunos, por exemplo, é possível perceber que uma grande parte dos alunos observa o conteúdo de Linguagem de Programação I como uma disciplina de dificuldade moderada. Isso pode significar que de forma geral, os alunos da disciplina podem encontrar algum ponto de complexidade em relação ao conteúdo, mas não a todo. Por esse motivo, o autor buscou indentificar quais os conteúdos com percepção de maior complexidade. Os alunos poderiam marcar até 3 conteúdos nessa questão.

Figura 51 – Percepção sobre conteúdos mais complexos da disciplina



Fonte: Do Autor (2023)

Como observado na Figura 51, o conteúdo observado como de maior complexidade para compreensão dos alunos foi o de ponteiros, seguido por funções e estruturas. Infelizmente, os 3 conteúdos não foram abordados pelo jogo CodingJob. Foi possível perceber que ocorreu uma melhoria consideravel em relação a percepção dos exercícios apresentados. Mesmo com essa melhoria evolutiva em cada versão, o número de alunos que conseguiram encerrar o jogo foi baixo. Isso ocorreu devido a quantidade de exercícios disponíveis para resolução nas versões apresentadas. Inicialmente foram apresentadas 8 fases, transformadas em 11 fases alteradas em um formato mais simplificado. Já a motivação em relação a realizar os exercícios mudou de forma drástica conforme as adaptações nas fases foram aplicadas, sendo a última versão a melhor avaliada nesse quesito.

9 CONSIDERAÇÕES FINAIS

Este documento apresentou o processo de desenvolvimento de um jogo sério para auxiliar alunos da disciplina inicial de programação a praticar o conteúdo de linguagem de programação C. Apoiado pela metodologia DSR, o jogo foi construído em conjunto com os alunos da disciplina, utilizando várias sugestões disponibilizadas pelos alunos através dos questionários e conversas informais.

A pesquisa apresentou uma forma de como criar e aplicar jogos digitais como ferramentas educacionais. Embora foque no conteúdo de programação C, o autor acredita que outros conteúdos poderiam ser contemplados da mesma forma. A pesquisa ainda serviu como uma forma de apresentar ao autor novos, novos conceitos sobre construção de jogos digitais voltados para a educação e outras formas de como apresentar o conteúdo de disciplinas de programação para alunos do ensino superior.

O autor nota que diversos aprendizados nasceram da pesquisa, vários relacionados a produção de jogos sérios. Como observado nas pesquisas, a narrativa tem um peso muito maior do que o imaginado para os alunos. Em futuras produções (ou versões de CodingJob), a construção uma narrativa mais envolvente e mais ampla é uma necessidade primária, já que é ela parece ser um dos principais motivos dos jogadores se engajarem no jogo. A curiosidade pelo destino dos personagens que os alunos conheceram durante a jornada deve ser levada em conta no futuro.

Outra alteração necessária para CodingJob, é a simplificação da criação de novos conteúdos ou de novos exercícios sobre programação C. No projeto atual, é necessário que que o professor tenha alguma conhecimento básico de de Unity e C para conseguir incluir algum material novo.

CodingJob foi registrado pelo INPI e deverá ter uma página de apresentação baseada nos resultados da pesquisa, além da inclusão da dissertação e artigos baseados no projeto.

A versão para sistema operacional Windows pode ser obtida clicando aqui. Já o código do projeto se encontra em https://github.com/guilhermescosta/CodingJob.

REFERÊNCIAS BIBLIOGRÁFICAS

ALCOVER, E.; CAPó, A. Jaume-i; MOYá-ALCOVER, B. Progame: A process framework for serious game development for motor rehabilitation therapy. *PLoS ONE*, v. 13, 2018. Citado na página 33.

APPERLEY, T. H. Genre and game studies: Toward a critical approach to video game genres. *Simulation & Gaming*, v. 37, n. 1, p. 6–23, 2006. Disponível em: <https://doi.org/10.1177/1046878105282278>. Citado na página 19.

BASTIANI, E.; ROMAN, L. M.; BOTELHO, V. Proposta de um jogo sério e desplugado para o ensino de algoritmos. *Revista do CCEI*, URCAMP, v. 24, n. 32, 2019. Citado na página 23.

BATTISTELLA, P. E.; WANGENHEIM, C. Gresse von. Engaged: Um processo de desenvolvimento de jogos para ensinar computação. In: . [S.l.: s.n.], 2016. p. 380. Citado na página 15.

BENNEDSEN, J.; CASPERSEN, M. Failure rates in introductory programming. *SIGCSE Bulletin*, v. 39, p. 32–36, 2007. Citado na página 16.

BENNEDSEN, J.; CASPERSEN, M. E. Failure rates in introductory programming: 12 years later. *ACM Inroads*, Association for Computing Machinery, New York, NY, USA, v. 10, n. 2, p. 30–36, abr. 2019. ISSN 2153-2184. Disponível em: <https://doi.org/10.1145/3324888>. Citado na página 16.

BRAGA, K. C. *CODEIN'PLAY: UM AMBIENTE DE MEDIAÇÃO DO ERRO A PARTIR DA AVALIAÇÃO DE EXERCÍCIOS DE PROGRAMAÇÃO DE COMPUTADORES*. Dissertação (Mestrado) — Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul, 2020. Citado na página 46.

CEEBOT. 2002. <http://www.blupi.org/>. Accessed: 2021-07-19. Citado na página 23.

CODEWARS. 2012. <https://www.codewars.com/>. Accessed: 2021-07-19. Citado na página 22.

CODINGAME. 2012. <https://www.codingame.com/>. Accessed: 2021-07-19. Citado na página 22.

DIJKSTRA, E. *On the Cruelty of Really Teaching*. 1988. <https://www.cs.utexas.edu/users/EWD/transcriptions/EWD10xx/EWD1036.html>. Accessed: 2021-06-09. Citado na página 25.

DJAOUTI, D.; ALVAREZ, J.; JESSEL, J.-P. *Classifying Serious Games: G/P/S model*. [S.l.]: IGI Global, 2011. Citado na página 19.

ECK, R. V. Digital game based learning it's not just the digital natives who are restless. *EDUCAUSE*, v. 41, 01 2006. Citado na página 21.

GIRAFFA, L. M. M.; MORA, M. da C. Evasão na disciplina de algoritmos e programação: Um estudo a partir dos fatores intervenientes na perspectiva do aluno. In: *Tercera Conferencia sobre el Abandono en la Educación Superior (III CLABES)*. [S.l.: s.n.], 2016. Citado na página 14.

GUEDES, K. et al. Um retrato do ensino de algoritmos e programação de computadores em cursos de engenharia de produção. In: . [S.l.: s.n.], 2017. Citado na página 15.

HUNICKE, R.; LEBLANC, M.; ZUBEK, R. Mda: A formal approach to game design and game research. In: *In Proceedings of the AAAI Workshop on Challenges in Game AI*. [S.l.: s.n.], 2004. Citado na página 32.

ISOTANI, S.; ROCHA, R.; BITTENCOURT, I. Análise, projeto, desenvolvimento e avaliação de jogos sérios e afins: uma revisão de desafios e oportunidades. In: *XXVI Seminário Brasileiro de Informática na Educação*. [S.l.: s.n.], 2015. Citado na página 19.

KEITH, C. *Agile Game Development with Scrum*. [S.l.]: Pearson/Addison-Wesley Professional, 2010. Citado 2 vezes nas páginas 33 e 36.

KERNIGHAN, B. W.; RITCHIE, D. M. *The C Programming Language*. 2. ed. [S.l.]: Prentice Hall, 1988. Citado na página 28.

LEONG, B.; KOH, Z. H.; RAZEEN, A. Teaching introductory programming as an online game. 2011. Citado na página 23.

MCCRACKEN, W. et al. A multi-national, multi-institutional study of assessment of programming skills of first-year cs students. *SIGCSE Bulletin*, v. 33, p. 125–180, 12 2001. Citado na página 17.

PAPERT, S. *Mindstorms: Children, Computers and Powerful Ideas*. [S.l.]: Basic Books, inc., 1980. Citado 2 vezes nas páginas 17 e 24.

PAPERT, S. Different visions of logo: Computers in the schools. *Interdisciplinary Journal of Practice, Theory, and Applied Research*, v. 2, 1985. Citado 3 vezes nas páginas 14, 17 e 30.

PAPERT, S. *A Máquina das Crianças: Repensando a Escola na Era da Informática*. [S.l.]: Artes Médicas, 1994. Citado na página 17.

PIMENTEL, M.; FILIPPO, D.; SANTORO, F. M. Design science research: fazendo pesquisas científicas rigorosas atreladas ao desenvolvimento de artefatos computacionais projetados para a educação. *Metodologia de Pesquisa em Informática na Educação: Concepção da Pesquisa*, SBC, 2019. Citado na página 29.

PLASS, J.; HOMER, B.; KINZER, C. Foundations of game-based learning. *Educational Psychologist*, v. 50, p. 258–283, 10 2015. Citado na página 20.

PRENSKY, M. *Aprendizagem Baseada em Jogos Digitais*. [S.l.]: Editora SENAC, 2012. Citado na página 14.

RAMOS, V. et al. A comparação da realidade mundial do ensino de programação para iniciantes com a realidade nacional: Revisão sistemática da literatura em eventos brasileiros. In: *XXVI Simpósio Brasileiro de Informática na Educação (SBIE 2015)*. [S.l.: s.n.], 2015. Citado 2 vezes nas páginas 14 e 16.

RESNICK, M. *Turtles, termites and traffic jams: Explorations in massively parallel microworlds.* [S.l.]: MIT Press., 1997. Citado na página 17.

SANTOS, ; GOMES, A.; MENDES, A. A taxonomy of exercises to support individual learning paths in initial programming learning. In: . [S.l.: s.n.], 2013. p. 87–93. Citado na página 17.

SHAFFER, D. et al. Video games and the future of learning. *The Phi Delta Kappan*, v. 87, p. 104–111, 10 2005. Citado na página 20.

SOFTEX. *Mercado de Trabalho e Formação de Mão de Obra em TI*. 2013. <http://www.ftp.softex.br/Inteligencia/cadernos_tematicos/cadernos_tematico_mercado_de_trabalho.pdf>. Accessed: 2020-01-12. Citado na página 14.

SQUIRE, K. Video game–based learning: An emerging paradigm for instruction. *Performance Improvement Quarterly*, v. 21, p. 7–36, 2008. Citado 3 vezes nas páginas 14, 20 e 30.

TIOBE. *TIOBE PROGRAMMING LANGUAGE INDEX*. 2021. <https://www.tiobe.com/tiobe-index>. Accessed: 2021-02-13. Citado 2 vezes nas páginas 26 e 28.

TOSTES, L. K. e Carlos Beleti Jr e Robertino Santiago Jr e R. Ensino de programação: um estudo preliminar nos cursos de licenciatura em computação no brasil. *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, v. 8, n. 1, p. 21, 2019. ISSN 2316-8889. Disponível em: <https://www.br-ie.org/pub/index.php/wcbie/article/view/8943>. Citado na página 15.

WATSON, C.; LI, F. Failure rates in introductory programming revisited. In: *2014 Conference on Innovation 38; Technology in Computer Science Education*. [S.l.: s.n.], 2014. Citado na página 14.

WONG, W.-T.; CHOU, Y.-M. An interactive bomberman game-based teaching /learning tool for introductory c programming. technologies for e-learning and digital entertainment. *Technologies for E-Learning and Digital Entertainment*, Springer, 2007. Citado na página 23.

10 APÊNDICE A - QUESTIONÁRIO APLICADO PARA ALUNOS QUE JÁ CURSARAM A DISCIPLINA

1- Quando você cursou a disciplina de Algoritmos e Programação?

2- Qual o resultado? (Aprovado / Reprovado)

3- Na sua opinião, qual o conteúdo mais complexo de compreender?

4- Qual o conteúdo mais fácil?

5- Você tem o hábito de jogar algum jogo digital?

6- Se sim, qual seu estilo de jogo preferido
RPG – Ação – First Person Shooter – Corrida – Esporte – Estratégia – Outro

7- Em caso de evasão, qual o motivo? (Falta de compreensão da disciplina / Problemas pessoais)

8- Em qual período você evadiu? (1° a 2° mês / 3° a 4° mês / Final do semestre)

9 – Você obteve resultado positivo na disciplina de Lógica para Computação e/ou ?[1]

[1] Nos cursos de computação do Instituto Federal do Rio Grande do Sul existem 2 disciplinas paralelas a Algoritmos e Programação que servem como base do conteúdo da mesma. São intituladas Lógica para Computação e

11 APÊNDICE B – QUESTIONÁRIO APLICADO PARA ALUNOS NOVOS NA DISCIPLINA

1- Você tem costume de jogar algum tipo de jogo digital (no videogame, computador, smartphone)? (S/N)

2- Se sim, qual seu estilo de jogo preferido
RPG – Ação – Esporte (Futebol, Basquete, Corrida etc...) – Estratégia – Outro

3- Quantas horas por semana você costuma estudar fora do horário das aulas?

4- Você acredita que é possível aprender ou já aprendeu algo jogando algum jogo digital? (S/N)

5 – Você acredita que utilizaria mais horas de estudo se as atividades forem apresentadas como jogos? (S/N)

6- Você prefere jogos cooperativos ou competitivos?

7- Você prefere que as atividades em aula sejam em grupo ou individuais?

8- Você possui alguma experiência no mercado de trabalho ? (S/N)

12 APÊNDICE C – QUESTIONÁRIO APLICADO DURANTE OS TESTES. VERIFICAÇÃO DA PERCEPÇÃO DO ALUNO SOBRE A DISCIPLINA DE LINGUAGEM DE PROGRAMAÇÃO I

1- Qual foi a sua percepção da disciplina de Linguagem de Programação I?
( )Muito fácil - ( )Fácil - ( ) Moderado - ( ) Difícil - ( ) Muito Difícil

2- Quais os conteúdos você encontrou maior dificuldade para compreender na disciplina de Linguagem de Programação I? (Marque até 3)
( ) Operadores Matemáticos
( )Sintaxe de Linguagem de Programação
( )Variáveis e Constantes
( )Operadores Lógicos
( )Expressões Condicionais (If, Else, Switch-Case)
( )Laços de Repetição (For, While, Do-While)
( )Funções
( )Estruturas
( )Ponteiros
( )Vetores

3- Você costuma jogar algum tipo de jogo digital (no videogame, computador, smartphone ou tablet)?
( ) Sim ( ) Não

4- Quantas horas por semana você costuma estudar, fora do horário das aulas, para as disciplinas de programação?
( ) até 1 horas - ( ) de 1 até 2 horas - ( ) de 2 até 3 horas - ( ) Mais de 3 horas

5- Você acredita que é possível aprender algo através de um jogo digital?
( ) Sim ( ) Não
Justificativa:

6- Você possui alguma experiência no mercado de trabalho de desenvolvimento de software?

( ) Sim ( ) Não

7- Você considera importante que um jogo, que tenha o objetivo de ensinar programação, simule o ambiente de trabalho de um programador?
( ) Sim ( ) Não

8- Você já utilizou algum jogo que tivesse o objetivo de ensinar algum conteúdo de programação?
( ) Sim ( ) Não

9- Que características você considera importante para um jogo que tenha o objetivo de auxiliar na compreensão de conteúdos de linguagem de programação?

13 APÊNDICE D – AVALIAÇÃO DO JOGO CODINGJOB

1- Você considera adequada a forma como os exercícios foram apresentados no CodingJob?
( ) Sim ( ) Não

2- Você conseguiu terminar todos os exercícios do jogo CodingJob?
( ) Sim ( ) Não

3- Você se sentiu motivado em realizar os exercícios de CodingJob?
( ) Sim ( ) Não

4- Existe algum conteúdo da disciplina que gostaria que fosse abordado em versões futuras do jogo CodingJob?

5- Os emails e dicas do personagem Haskell ajudaram de alguma forma?
( ) Sim ( ) Não

6- As dicas do personagem Pascal ajudaram de alguma forma?
( ) Sim ( ) Não

7- O que você mais gostou no CodingJob?

8- Que melhorias você sugere para CodingJob?

14 APÊNDICE E - SUGESTÕES DO TESTE 1

aluno 1 -Ampliar o feedback dos erros do jogador, de modo que o erro específico seja mais claro, e possíveis soluções. -Fazer o chat app mais acessível pelo IDE, como é o caso do e-mail.

aluno 2 -Uma validação mais compreensível do código escrito (talvez com demonstração dos erros cometidos)

aluno 3 -Seria interessante adaptar o jogo para deficiências físicas como: o deficiente visual, implementando sons.

aluno 4 -Redução do tamanho da caixa de diálogo, introduzir dicas do pascal enquanto programo, o jogo é muito didático e interativo. Acho que isso pode melhorar bastante ele!

aluno 5 -A instrução que fico no canto esquerdo, mudar para um ícone em forma de ponto de interrogação no canto direito superior "?-Colocar o número de linhas, como nas IDEs. Facilita na dica do erro e na leitura do código. -Parabéns pelo projeto.

aluno 6 -Não me senti muito motivado a jogar, os testes de error não explicam muito bem o erro e não dá vontade de ficar brigando com o campo pra saber quer que você escreva. Assim como o jogo esta agora não tem nenhuma diferença de fazer um exercício proposto pelo professor no moodle, só que 20 vezes mais frustrante, sem pontos, tempo pra fazer a atividade etc. Boa sorte com o projeto.

aluno 7 -Mensagens de erro mais explicativas

aluno 8 -As dicas do personagem Pascal poderiam ser acessadas de forma que não precise começar o código novamente ao sair da IDE. -Os erros apresentados na IDE poderiam ser mais descritivos. Pra quem está começando a programar é importante. -Talvez pudesse ter um breve tutorial em cada tarefa, explicando o básico de cada tópico, sobre as estruturas, em vez de somente apresentar como montar.

aluno 9 -No 1º exercício tive dificuldade de entender o que precisava fazer, pois não vi que a parte branca no canto esquerdo superior eram as dicas. Acho que ícone poderia ser para algo parecido como um alerta [1]. -A mensagem de erro poderia ser mais explicativa.

aluno 10 -Não perde o progresso ao sair da IDE para falar com Pascal ou ler e-mails. -Aceitar todas as respostas que retornem o resultado esperado. -Retornar erros mais legíveis/interpretáveis.

aluno 11 -Correção de bugs e interação com personagens.

aluno 12 -No momento não me vem nada em mente.

aluno 13 -Revisar as respostas, algumas ele é bem chatinho com os espaços.

15 APÊNDICE F - SUGESTÕES DO TESTE 2

aluno 1 -Acho que os enunciados e as dicas nos campos de respostas acabaram sendo um pouco confusos em alguns exercícios, acredito que devem ser apresentados para o usuário de uma forma mais clara. -A interface não é muito intuitiva, por exemplo: demorei para perceber que os ícones do Haskell e do Pascal acima dos códigos eram botões para o email e para o chat app, e eu acabei evitando de clicar no botão "Desligar o PC"porque achei que fechava o jogo completamente, acabou não ficando claro que voltava para o menu de cenários. -As mensagens de erro podem ter melhorias também, pois em alguns momentos eu não sabia bem o que eu estava fazendo de errado. -Também sugiro que o jogo possa salvar o nosso progresso e que tenha opção de pular a introdução para quem já abriu o jogo anteriormente.

aluno 2 -Acredito que as mensagens de erro não estão claras e são indicadas incorretamente.

aluno 3 -Tooltips nos icones do Haskell e Pascal nas telas de código. Implementar boas práticas de código, exemplo quando usar switch e if. Algum tipo de dica quando se excede uma quantidade de erros, tirando o erro do ; os demais são bem genéricos e não ajudam a visualizar onde se esta errando.

aluno 4 -Explicações bem menos vagas nos exercícios. Muitos eu não consegui fazer pq ele só diz que tem erro, sendo que eu segui o que os personagens pediram. Ter dicas melhores e mostrar TUDO que é necessário para responder as questões seria uma melhoria que priorizaria. Errar e não saber porque é muito frustrante e vai fazer o jogador parar de jogar. -Alguns comandos não funcionam como o tab no primeiro exercício.

aluno 5 -Refazer a parte gráfica do jogo. Escolher outra paleta de cores. -Tornar a parte inicial de apresentação uma animação. -Repensar o design da IDE e do chatbot. Incluir mais exemplos de código e saídas na seção de chat. -Apresentar cada conteúdo antes de pedir a questão por e-mail.

16 APÊNDICE G - SUGESTÕES DO TESTE 3

aluno 1 -Mais explicações

aluno 2 -Mais conteúdos sobre a matéria, ou quem sabe, futuras versões para outras linguagens.

aluno 3 -Melhor detalhamento no início de o que fazer. -Começar com níveis mais fáceis e evoluir aos poucos.

aluno 4 -Todas as estruturas e funções deixou o programa interessante, conto com as atualizações futuras.

aluno 5 -Clareza dos objetivos e melhor retorno quanto aos erros.

aluno 6 -Maior fluidez.

aluno 7 -Flexibilização dos nomes e palavras, e um comando para voltar tipo ctrl+z

aluno 8 -Informar os erros.

aluno 9 -Opção para dimiuir o volume da trilha sonora, mais cores no design do menu, talvez uma opção de "dificultar tarefa"para que Alan resolva o mesmo desafio porém de outra forma e ainda sim aplicando a mesma lição aprendida.

aluno 10 -Na atividade "operadores aritméticos, relacionais e lógicos"há algumas inconsistências

aluno 11 -Ordens mais claras de Haskell

aluno 12 -Não entendi o que deveria ser feito

aluno 13 -Mais base para digitação do código, mais exemplos. -Botões de voltar, após as interações do Haskell. Eu cliquei no X para voltar e precisava reiniciar o jogo. -Interações de motivação quando acertar o código

aluno 14 -Progresso do personagem, melhoria dos níveis e melhoria de interface.

aluno 15 -Público alvo não adequado, acredito que para pessoas mais novas ou mesmo crianças o jogo funcione melhor.

aluno 16 -Dar sugestão quando errar 3 vezes.

aluno 17 -Alterar o botão "Desligar PC"para "Voltar"ou algo assim.

aluno 18 -Acredito que as dicas poderiam ser melhores