UNIVERSIDADE DE CAXIAS DO SUL
ÁREA DO CONHECIMENTO DE CIÊNCIAS EXATAS E ENGENHARIAS
BACHARELADO EM TECNOLOGIAS DIGITAIS

GABRIEL CARNIEL DE OLIVEIRA

DESENVOLVIMENTO DE UM SERIOUS GAME PARA AUXILIAR NO APRENDIZADO DE PROGRAMAÇÃO

CAXIAS DO SUL
2022

GABRIEL CARNIEL DE OLIVEIRA

DESENVOLVIMENTO DE UM SERIOUS GAME PARA AUXILIAR NO APRENDIZADO DE PROGRAMAÇÃO

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Tecnologias Digitais na Área do Conhecimento de Ciências Exatas e Engenharias da Universidade de Caxias do Sul.

Orientador: Profª. Dra. Elisa Boff

CAXIAS DO SUL

2022

GABRIEL CARNIEL DE OLIVEIRA

DESENVOLVIMENTO DE UM SERIOUS GAME PARA AUXILIAR NO APRENDIZADO DE PROGRAMAÇÃO

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Tecnologias Digitais na Área do Conhecimento de Ciências Exatas e Engenharias da Universidade de Caxias do Sul.

Aprovado em: __/__/____

Banca Examinadora

_____
Profª. Dra. Elisa Boff
Universidade de Caxias do Sul - UCS

_____
Profª. Dra. Carine Geltrudes Webber
Universidade de Caxias do Sul - UCS

_____
Prof. Dr. Marcelo Luís Fardo
Universidade de Caxias do Sul - UCS

RESUMO

A taxa de evasão em cursos de graduação na área de tecnologias é bastante alta quando comparada a outras áreas do conhecimento. Um dos fatores que contribuem para essa taxa elevada é a dificuldade dos alunos com o aprendizado de programação, principalmente em disciplinas introdutórias; essas dificuldades normalmente se apresentam nos conceitos básicos da disciplina, como abstração e lógica, para entendimento de algoritmos, e funções e estruturas de repetição, para entendimento da programação em si. Esses problemas poderiam ser amenizados com o ensino de pensamento computacional em turmas de educação básica. Propostas de currículos para essa finalidade, como o da CIEB, existem porém eles ainda não são aplicados. Tendo em vista as dificuldades dos alunos de graduação com o aprendizado de programação, esse trabalho visa propor e avaliar a criação de um jogo sério que possa auxiliar no ensino dessas disciplinas iniciais de cursos de tecnologia. Utilizando ferramentas para desenvolvimento e avaliação de jogos sérios, o trabalho busca, na sequência, desenvolver o jogo proposto e avaliá-lo como ferramenta de ensino para disciplinas de programação e pensamento computacional. As avaliações, realizadas utilizando o método MEEGA+, foram aplicadas em turmas de programação de computadores para validar a aplicação do jogo como ferramenta de auxílio no ensino de programação e estruturas condicionais.

Palavras-chave: Jogos sérios, aprendizagem de programação, pensamento computacional

ABSTRACT

The dropout rate in undergraduate courses in the area of technology is quite high when compared to other areas of knowledge. One of the factors that contribute to this high rate is the students' difficulty with learning programming, especially in introductory disciplines; these difficulties normally appear in the basic concepts of the discipline, such as abstraction and logic, for understanding algorithms, and functions and repetition structures, for understanding programming itself. These problems could be mitigated by teaching computational thinking in basic education classes. Proposals for curricula for this purpose, such as the CIEB, exist but they are not yet applied. In view of the difficulties of undergraduate students with learning programming, this work aims to propose and evaluate the creation of a serious game that can help in the teaching of these initial subjects in technology courses. Using tools for the development and evaluation of serious games, the work then seeks to develop the proposed game and evaluate it as a teaching tool for programming and computational thinking disciplines. The evaluations, carried out using the MEEGA+ method, were applied in computer programming classes to validate the application of the game as an aid tool in the teaching of programming and conditional structures.

Keywords: Serious Games, programming learning, computational thinking

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

TI	Tecnologia da Informação
SEMESP	Secretaria de Modalidades Especializadas de Educação
SGDA	*Serious Game Design Assessment*
MEEGA	*Model for the Evaluation of Educational GAmes*
CIEB	Centro de Inovação para a Educação Brasileira
UC	Unidade curricular
UFERSA	Universidade Federal Rural do Semi-Árido
RPG	*Role Playing Game*

SUMÁRIO

1. INTRODUÇÃO

Com a crescente transformação digital nas áreas do conhecimento, a busca por profissionais da área de TI vem aumentando nos últimos anos. A procura por profissionais da área teve um crescimento de 671% apenas em 2020 (CNN, 2021). Em resposta a isso, as instituições de ensino superior têm uma projeção de formar cinco vezes mais profissionais na área até 2025, em comparação aos dados de profissionais formados até 2019 (PETROPOULEAS, 2021). Além disso, de acordo com o Semesp, o curso de Sistemas de Informação está entre os 10 mais procurados na rede privada e entre os 20 mais procurados da rede pública (SEMESP, 2021). Outra estatística, porém, vai de encontro a essa. Este curso possui a maior taxa de evasão, na modalidade presencial, e está entre os 10 primeiros nessa mesma estatística para modalidades a distância (SEMESP, 2021).

Um dos grandes desafios para alunos que estão ingressando em cursos da área é o aprendizado de programação. Disciplinas introdutórias de programação normalmente encontram-se nos semestres iniciais dos cursos da área e acabam se tornando uma barreira para alguns alunos, por diversos fatores. Estudos feitos por Arimoto e Oliveira (2019), Moreira *et al.* (2018) e Souza *et al* (2016) buscam possíveis motivos e soluções para as dificuldades encontradas por alunos no aprendizado de programação. Essas explicações variam desde a dificuldade para compreender conceitos de programação, até a falta de motivação para o aprendizado da mesma. Possíveis soluções para esse problema também são bastante variadas, envolvendo projetos de ambientes de aprendizagem e jogos sérios para auxiliar na educação de programação.

Tendo como proposta principal um jogo sério para ensinar e desenvolver conceitos de programação, o trabalho também buscou modelos para desenvolvimento e avaliação de jogos educativos como ferramentas de ensino, a fim de fundamentar o jogo proposto. Essas ferramentas, como o SGDA e o MEEGA+, foram utilizadas como método de avaliação e validação da proposta, buscando entender os fundamentos de um jogo sério e como ele pode ser utilizado para o ensino.

A falta de conhecimento prévio sobre programação e algoritmos também pode ser uma barreira para alunos de graduação. Por isso este trabalho buscou entender, por meio do currículo desenvolvido pelo CIEB, como poderia ser aplicado o ensino de pensamento computacional para turmas de ensino básico e como o jogo desenvolvido pode ser aplicado nesse contexto.

Neste trabalho identificou-se como problema de pesquisa as dificuldades de estudantes com disciplinas de programação. O trabalho busca compreender onde essas dificuldades se encontram e como seria possível auxiliar o aprendizado desses alunos. Além disso, este trabalho buscou, por meio de trabalhos relacionados, entender quais são e como funcionam outras propostas para solucionar problemas similares, avaliando o desenvolvimento e aplicação de outros jogos sérios ou ambientes de aprendizagem para as áreas de programação e pensamento computacional.

Levando em conta o problema de pesquisa, o trabalho foi norteado pela questão de pesquisa: "*Como pode ser estruturado, em termos de conteúdo, estética e mecânicas, um jogo sério para desenvolver a aprendizagem de conceitos de programação?*"

1.1. OBJETIVOS

O objetivo geral deste trabalho é desenvolver e avaliar um jogo sério para auxiliar na aprendizagem de programação. Neste contexto, o trabalho tem como objetivos específicos:

- Pesquisar currículos que incluam o pensamento computacional;
- Identificar dificuldades dos alunos com o aprendizado de programação;
- Buscar exemplos de ferramentas que auxiliam no ensino de programação;
- Pesquisar métodos de desenvolvimento e avaliação de jogos sérios;

1.2. ESTRUTURA DO TRABALHO

O trabalho desenvolvido será estruturado da seguinte forma.

O Capítulo 2 trata de aprendizagem de programação, explorando currículos para ensino de pensamento computacional a turmas de ensino fundamental e médio; além de buscar entender, com base em outros trabalhos e pesquisas desenvolvidos, quais são as dificuldades encontradas por alunos de ensino superior quanto ao ensino de programação.

O Capítulo 3 aborda jogos sérios, buscando fundamentar, com trabalhos relacionados, o desenvolvimento de um jogo sério para ensino de programação. Além disso, o capítulo apresentará as ferramentas SGDA e MEEGA+, que serão utilizadas, respectivamente, para desenvolver e avaliar o jogo proposto.

O Capítulo 4 apresenta o jogo Code_Dungeon como proposta de solução, apresentando as dimensões do SGDA como base para a estrutura do jogo, demonstrando como ele foi desenvolvido e como será avaliado, tendo como base ferramentas exploradas no capítulo 3.

O Capítulo 5 apresenta os testes de validação do jogo desenvolvido como ferramenta de ensino. Descrevendo como ocorreram as etapas dos testes, qual foi a metodologia utilizada e avaliando os resultados obtidos a partir deles.

Por fim, no Capítulo 6, são apresentadas as considerações finais, levando em conta o jogo desenvolvido e as respostas dos questionários utilizados para a validação dele.

2. APRENDIZAGEM DE PROGRAMAÇÃO

Em cursos da área de tecnologia, disciplinas introdutórias de programação não assumem conhecimentos prévios dos alunos, portanto essas disciplinas devem explorar conceitos de pensamento computacional, como abstração e reconhecimento de padrões, além de desenvolver os conteúdos de programação e algoritmos. Nas seções seguintes deste capítulo, serão exploradas as dificuldades de alunos com o aprendizado em disciplinas de programação, além de entender como um currículo para ensino de pensamento computacional poderia ser aplicado, para que alunos de disciplinas iniciais de computação possam ingressar nessas disciplinas tendo um conhecimento básico sobre o conteúdo que será desenvolvido nelas.

2.1. PENSAMENTO COMPUTACIONAL

No cenário ideal para o ensino de programação em disciplinas iniciais de graduação, os alunos deveriam chegar na graduação tendo uma noção básica sobre pensamento computacional, que envolve ensino de abstração, algoritmos, decomposição de algoritmos e reconhecimento de padrões. Para avaliar esse cenário "ideal" do ensino de programação, este trabalho estudará currículos de pensamento computacional para turmas de ensino básico e médio, conforme proposto pelo CIEB (Centro de Inovação para a Educação Brasileira).

2.1.1. Pensamento Computacional no Ensino Fundamental

O Currículo de Referência em Tecnologia e Computação, desenvolvido pelo CIEB, propõe currículos para ensino infantil e médio, contemplando temas de tecnologia e computação. Ambos os currículos propostos possuem eixos que buscam o ensino de Pensamento Computacional.

Figura 1: Currículo de referência do CIEB, destinado à Educação Infantil e Ensino Fundamental

Fonte: Currículo de Referência em Tecnologia e Computação, CIEB

No currículo de referência para educação infantil e ensino fundamental (Figura 1), a proposta do eixo de Pensamento Computacional explora os conceitos de Abstração, Algoritmos, Decomposição e Reconhecimento de Padrões, conforme o Quadro 1, que demonstra, conforme currículo proposto, habilidades a serem desenvolvidas, e as práticas necessárias para o desenvolvimento dessas habilidades na Educação Básica, e os Quadros 2 a 10, para cada ano do Ensino Fundamental.

Quadro 1: Conceitos, habilidades e práticas para o desenvolvimento do Pensamento Computacional na Educação Básica

Conceito	**Habilidade**	**Prática**
Abstração	Identificar informações importantes e descartar informações irrelevantes	Procurar exemplos que podem ser convertidos em uma sequência de instruções, identificando os passos relevantes e irrelevantes para a solução do problema.
Algoritmos	Compreender o conceito de algoritmo como uma sequência de passos ou instruções, percebendo que existem diferentes algoritmos para resolver um mesmo problema	Trabalhar o conceito e execução de algoritmos, relacionando-os com movimentos do corpo. Discutir diferentes algoritmos para solucionar um mesmo problema.
Decomposição	Compreender o conceito de decomposição por meio do agrupamento de brinquedos	Identificar similaridades e diferenças entre brinquedos, separando-os de acordo com critérios definidos.
Reconhecimento de Padrões	Identificar padrões em um conjunto de objetos ou sons	Encontrar formas, cores ou melodias que se repetem em um conjunto.

Fonte: Currículo de Referência em Tecnologia e Computação, CIEB

Quadro 2: Conceitos, habilidades e práticas para o desenvolvimento do Pensamento Computacional no Primeiro ano do Ensino Fundamental

Conceito	**Habilidade**	**Prática**
Abstração	Compreender que os computadores não têm inteligência e apenas realizam o que é programado	Dialogar sobre o papel do ser humano na criação de programas de computador e sobre a importância de saber programar

Algoritmos	Compreender o conceito de algoritmo como uma sequência de passos ou instruções	Compreender pequenos algoritmos ou sequências de instruções, visualizando possíveis erros e fazendo os ajustes necessários
Decomposição	Exercitar a decomposição	Utilizar exemplos do mundo real para criar algoritmos, identificando uma sequência principal e outras menores
Reconhecimento de Padrões	Identificar que todos os softwares são programados	Definir softwares como sendo programas criados por pessoas e detalhar como eles são produzidos, exemplificar tipos de software

Fonte: Currículo de Referência em Tecnologia e Computação, CIEB

Quadro 3: Conceitos, habilidades e práticas para o desenvolvimento do Pensamento Computacional no Segundo ano do Ensino Fundamental

Conceito	**Habilidade**	**Prática**
Abstração	Reconhecer os diferentes tipos de dados	Classificar textos, números e valores booleanos, exemplificando-os com situações do dia-a-dia
Algoritmos	Compreender o uso de repetição com número fixo de iterações	Criar algoritmos que utilizem repetições
Decomposição	Decompor, identificar e explicar a função das partes e sensores encontrados em dispositivos digitais e seus usos em algoritmos	Apresentar a definição de sensores; inventar novos sensores para um dispositivo existente; simular o funcionamento de um dispositivo
Reconhecimento de Padrões	Identificar, entender e explicar em que situações o computador pode ou não ser utilizado para solucionar um problema	Identificar situações em que o computador pode ou não auxiliar na resolução de problemas

Fonte: Currículo de Referência em Tecnologia e Computação, CIEB

Quadro 4: Conceitos, habilidades e práticas para o desenvolvimento do Pensamento Computacional no Terceiro ano do Ensino Fundamental

Conceito	**Habilidade**	**Prática**
Abstração	Compreender a distinção entre dado e informação	Entender como informações são registradas em um dispositivo
Algoritmos	Descrever os algoritmos de operações aritméticas simples	Descrever os passos de operações de soma, multiplicação ou divisão como sendo algoritmos
Decomposição	Decompor um algoritmo em processos menores para representação em diagramas	Identificar as etapas e decisões de um algoritmo, representando-o por meio de um diagrama
Reconhecimento de Padrões	Identificar e propor novas maneiras de interação ou interface em dispositivos computacionais	Identificar diferentes maneiras de interagir com os computadores e propor novas maneiras de interação

Fonte: Currículo de Referência em Tecnologia e Computação, CIEB

Quadro 5: Conceitos, habilidades e práticas para o desenvolvimento do Pensamento Computacional no Quarto ano do Ensino Fundamental

Conceito	**Habilidade**	**Prática**
Abstração	Entender que cada letra, número ou símbolo é representado por um padrão de caracteres	Relacionar e representar caracteres por um padrão de código
Algoritmos	Executar algoritmos simples, em português estruturado	Criar e executar algoritmos que utilizam condições com operações relacionais e lógicas para decisões
Decomposição	Classificar dispositivos digitais de acordo com suas características, usos e funcionalidades	Encontrar semelhanças e diferenças entre dispositivos, e categorizá-los conforme seus atributos
Reconhecimento de Padrões	Identificar semelhanças e diferenças em situações que se repetem	Identificar situações no mundo real em que seja possível utilizar iterações na execução de tarefas

Fonte: Currículo de Referência em Tecnologia e Computação, CIEB

Quadro 6: Conceitos, habilidades e práticas para o desenvolvimento do Pensamento Computacional no Quinto ano do Ensino Fundamental

Conceito	**Habilidade**	**Prática**
Abstração	Conhecer representações concretas para listas, filas e pilhas	Apresentar a definição de listas, filas e pilhas; procurar conceitos do mundo real e digital que possam ser representados por estas
Algoritmos	Conhecer e utilizar algoritmos com repetição	Executar e criar algoritmos que usam condições para controlar o número de repetições
Decomposição	Identificar e decompor operandos, operações e prioridades em expressões aritméticas	Decompor expressões aritméticas com vários termos; fazendo uso de parênteses para alterar a prioridades das operações nessas expressões
Reconhecimento de Padrões	Reconhecer um padrão em um algoritmo e converter em uma função sem retorno	Encontrar um conjunto de instruções que se repetem em um algoritmo, e substituí-lo pela execução de um subalgoritmo

Fonte: Currículo de Referência em Tecnologia e Computação, CIEB

Quadro 7: Conceitos, habilidades e práticas para o desenvolvimento do Pensamento Computacional no Sexto ano do Ensino Fundamental

Conceito	**Habilidade**	**Prática**
Abstração	Interpretar um algoritmo em pseudolinguagem e transpor para uma linguagem de programação visual	Compreender um algoritmo, transpondo-o para outra linguagem
Algoritmos	Experienciar e construir algoritmos com desvios condicionais utilizando uma linguagem de programação visual	Programar algoritmos com desvios condicionais utilizando ambiente de programação com blocos
Decomposição	Identificar e categorizar elementos que compõem a interface de um ambiente de programação visual	Explorar um ambiente de programação visual e os elementos de sua interface
Reconhecimento de Padrões	Identificar padrões de instruções que se repetem em um algoritmo e utilizar um	Analisar algoritmos e reconhecer a presença de instruções que se repetem

	módulo ou função para representar estas instruções	

Fonte: Currículo de Referência em Tecnologia e Computação, CIEB

Quadro 8: Conceitos, habilidades e práticas para o desenvolvimento do Pensamento Computacional no Sétimo ano do Ensino Fundamental

Conceito	**Habilidade**	**Prática**
Abstração	Conhecer o conceito de grafo e identificar instâncias do mundo real e digital	Ilustrar problemas do mundo real e digital que podem ser representados por grafos
Algoritmos	Experienciar e construir diferentes algoritmos com repetição	Criar e testar algoritmos com repetição
Decomposição	Compreender que a automatização de um problema é composta pela definição dos dados e do processo	Criar algoritmos para problemas do cotidiano e identificar neles os dados e os processos
Reconhecimento de Padrões	Identificar elementos que se repetem em diferentes softwares e compreender a modularização ou reuso de algoritmos	Utilizar diferentes softwares para reconhecer os elementos ou rotinas em comum entre eles

Fonte: Currículo de Referência em Tecnologia e Computação, CIEB

Quadro 9: Conceitos, habilidades e práticas para o desenvolvimento do Pensamento Computacional no Oitavo ano do Ensino Fundamental

Conceito	**Habilidade**	**Prática**
Abstração	Interpretar um algoritmo em linguagem natural e convertê-lo em linguagem de programação	Analisar uma sequência de instruções em linguagem natural e descrevê-las em linguagem de programação
Algoritmos	Experienciar e construir algoritmos de média complexidade utilizando uma linguagem de programação	Desenvolver algoritmos para problemas diversos
Decomposição	Compreender o conceito de paralelismo	Entender que algumas tarefas podem ser executadas sem esperar o término da anterior
Reconhecimento de Padrões	Entender a importância da identificação de padrões para a compressão de dados	Ilustrar a compactação de um texto, por exemplo, trocando palavras repetidas

		por um código numérico

Fonte: Currículo de Referência em Tecnologia e Computação, CIEB

Quadro 10: Conceitos, habilidades e práticas para o desenvolvimento do Pensamento Computacional no Nono ano do Ensino Fundamental

Conceito	**Habilidade**	**Prática**
Abstração	Compreender e identificar em um algoritmo a necessidade de utilizar a recursividade	Criar soluções por meio de algoritmos que façam uso de recursão
Algoritmos	Desenvolver algoritmos que utilizem recursão	Criar programas que façam uso de funções que se chamem a si mesmas
Decomposição	Compreender o que são programas modulares e por que incentivar sua reusabilidade	Criar um módulo independente, que possa ser usado em dois algoritmos diferentes
Reconhecimento de Padrões	Compreender que existem diferentes linguagens de programação e encontrar elementos comuns entre elas	Compreender que cada linguagem tem uma estrutura sintática parecida

Fonte: Currículo de Referência em Tecnologia e Computação, CIEB

Os currículos apresentados apresentam diversas habilidades que, se propriamente desenvolvidas, podem auxiliar num futuro aprendizado de programação e algoritmos. O trabalho propõe-se a ajudar com o ensino de programação, porém, os currículos de pensamento computacional também podem servir como guias para possíveis aplicações e adaptações à solução apresentada por este trabalho.

2.1.2. Pensamento Computacional no Ensino Médio

Para a proposta curricular de Ensino Médio (Figura 2), o eixo de Pensamento Computacional possui oito Unidades Curriculares (UCs), sendo quatro essenciais, totalizando 280 horas, e quatro eletivas, totalizando 240 horas. As UCs essenciais envolvem Programação de Computadores, Tecnologias para Internet, Design de Aplicativos e Ciência de Dados. Já as eletivas envolvem Jogos Digitais e Analógicos, Internet das Coisas, Simulações de Fenômenos Naturais e Robótica.

Figura 2: Currículo de referência do CIEB, destinado ao Ensino Médio

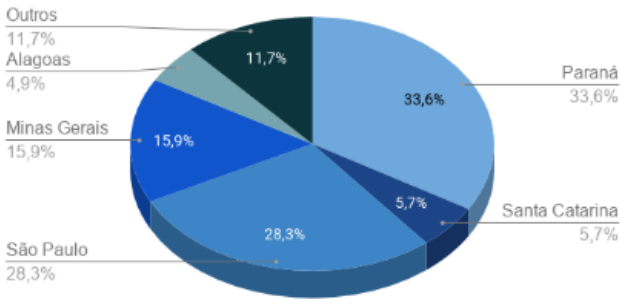
Fonte: Currículo de Referência em Tecnologia e Computação, CIEB

Como competências do eixo de Pensamento Computacional, a proposta curricular busca representar problemas complexos na forma de problemas menores, analisar os dados relevantes de um problema e representá-los em formatos que possam ser inseridos em um computador e compreender os processos e tecnologias envolvidas no desenvolvimento de programas para computadores.

2.2. DIFICULDADES NA APRENDIZAGEM

Arimoto e Oliveira (2019) realizaram uma pesquisa com estudantes de cursos da área da computação. O *survey*, que foi aplicado usando a plataforma *Google Forms*, teve como amostra mais de 200 alunos e recém-formados no ensino técnico e superior, em instituições de ensino públicas e privadas. Na Figura 3, pode-se observar o perfil dos respondentes.

Figura 3: perfil dos respondentes



Faixa Etária	Respondentes	(%)
17 a 20 anos	108	38%
21 a 24 anos	99	35%
25 a 28 anos	34	11%
Acima de 29 anos	43	15%
Sexo		
Masculino	218	77%
Feminino	66	23%
Instituição		
Pública	239	84%
Privada	45	16%
Período		
Diurno	20	7%
Noturno	182	64%
Integral	82	29%
Total	**284**	**100%**

Fonte: (Arimoto e Oliveira, 2019)

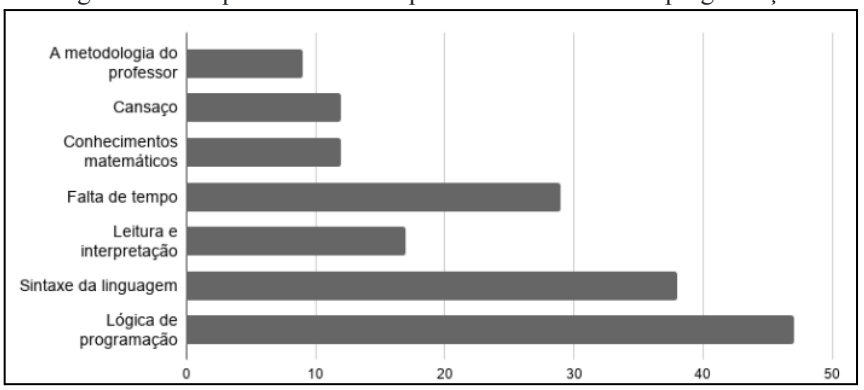
Na pesquisa, os autores observaram que grande parte dos respondentes, quanto aos conceitos e práticas de programação, indicaram possuir maior dificuldade com definição e uso de ponteiros, algoritmos recursivos e estrutura de dados. Quanto ao uso de ferramentas para auxiliar no ensino de programação, Arimoto e Oliveira (2019) afirmam que:

> Os estudantes foram questionados sobre a utilização ou não de ambientes e/ou ferramentas de apoio ao ensino e aprendizagem de programação. Mais da metade dos respondentes (60%) responderam que não era utilizado quaisquer ferramentas/ambientes de aprendizagem além da habitual sala de aula. Por outro lado, foram mencionadas a utilização de ambientes virtuais de aprendizagem como o Moodle, plataformas de cursos on-line como o Udemy, plataformas de envio e correção automática de exercícios de programação como URI e Run.codes [...] Em relação à efetiva contribuição das ferramentas/ambientes de aprendizagem no ensino e aprendizagem de programação, neste estudo 46% dos respondentes "concordam totalmente" que tais ambientes contribuem de maneira significativa para a melhoria da aprendizagem, enquanto 32% deles “concordam parcialmente” (ARIMOTO; OLIVEIRA, 2019, p.7)

Os resultados da pesquisa demonstraram que a maioria dos alunos respondentes possuem algum nível de dificuldade no aprendizado de programação e muitos deles sugeriram uma mudança na abordagem e metodologia de ensino para essas disciplinas, buscando mais atividades práticas ou ferramentas lúdicas para auxiliar com o aprendizado.

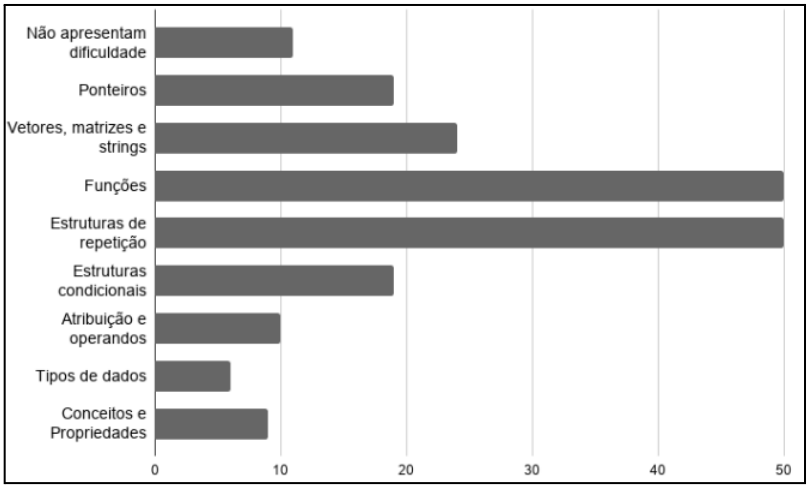
Outra pesquisa sobre o assunto foi aplicada por Moreira, Holanda, Coutinho e Chagas (2018) com alunos de cursos de TI da UFERSA. Os autores coletaram 110 respostas, entre alunos de disciplinas iniciais de programação, entre elas Algoritmos, Laboratório de Algoritmos e Informática Aplicada. Dentre os respondentes, 96 alunos nunca haviam utilizado linguagens de programação e a maioria deles afirmou possuir dificuldade em lógica de programação e na sintaxe da linguagem, conforme pode ser observado na Figura 4. Quanto aos conteúdos das disciplinas de programação, os alunos apresentaram uma maior dificuldade com funções e estruturas de repetição, conforme Figura 5.

Figura 4: Principais dificuldades para o entendimento de programação



Fonte: (Moreira, Holanda, Coutinho e Chagas, 2018)

Figura 5: Principais dificuldades com conteúdos de programação



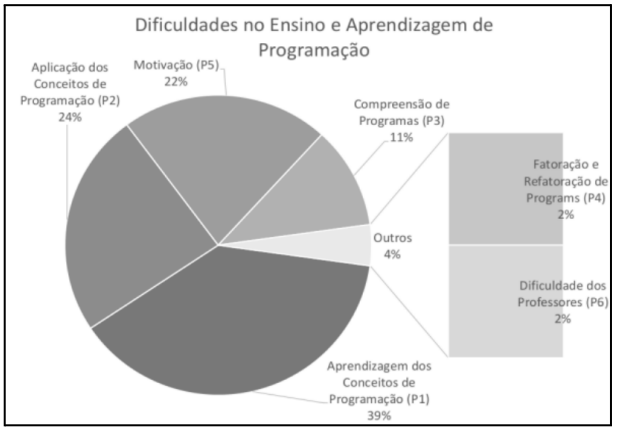
Fonte: (Moreira, Holanda, Coutinho e Chagas, 2018)

Com base nas respostas obtidas pelos autores, pode-se notar uma maior dificuldade com os conceitos e conteúdos iniciais das disciplinas de computação, sendo lógica ou sintaxe da linguagem quanto a conceitos, ou em funções e repetição, quanto a conteúdos.

Souza, Silva e Barbosa (2016) produziram um mapeamento sistemático com base em 70 artigos que apresentam problemas ou dificuldades no ensino ou aprendizagem de programação e sugerem possíveis soluções para esses problemas. Os artigos foram classificados de acordo com o tipo de problema apresentado, sendo eles aprendizagem de conceitos, aplicação dos conceitos, compreensão de programas, fatoração e refatoração, motivação e dificuldades relacionadas aos professores; e também quanto ao tipo de solução, que foram classificadas como visualização, *serious games*, ambientes de desenvolvimento pedagógico, colaboração, *scaffolding*, notações, reflexão, *feedback*, instrução ancorada, interatividade, problemas reais e representações semânticas. Os autores ainda afirmam que os artigos tinham a possibilidade de serem classificados simultaneamente em mais de uma das categorias; por exemplo, 25 dos artigos que discutiam problemas na aprendizagem de conceitos de programação também apresentavam problemas com a aplicação desses conceitos. Essa classificação em múltiplas categorias foi bem mais comum na esfera de problemas do que na esfera de soluções.

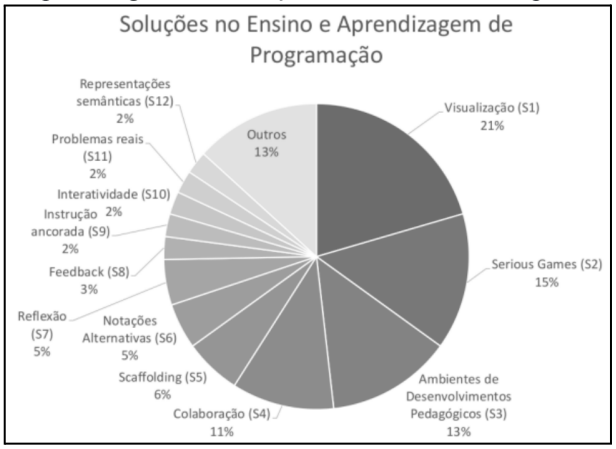
De acordo com a análise feita sobre os artigos com as categorias definidas, os autores apresentam gráficos com a quantidade de artigos que se encaixa em cada categoria, que podem ser vistos na Figura 6, para a representação quanto aos problemas, e na Figura 7, na classificação quanto às soluções.

Figura 6: Artigos classificados por categorias de problema, de acordo com mapeamento sistemático



Fonte: (Souza, Silva e Barbosa, 2016)

Figura 7: Artigos classificados por categorias de soluções, de acordo com mapeamento sistemático



Fonte: (Souza, Silva e Barbosa, 2016)

Com a análise dos dados obtidos pelo mapeamento sistemático, Souza, Silva e Barbosa (2016) afirmam que:

> Em geral, o mapeamento contribuiu para um melhor entendimento do atual estado da arte em ensino e aprendizagem de programação e para a identificação de limitações de pesquisa e futuras direções.[...] Foi possível concluir que os principais problemas no ensino e na aprendizagem de programação investigados atualmente são: (1) as dificuldades dos alunos em aprender os conceitos de programação, (2) a dificuldade dos alunos na aplicação desses conceitos durante a construção de programas e (3) a falta de motivação entre os alunos na realização da atividade de programação. Para amenizar tais problemas e dificuldades, foram identificadas tendências de pesquisa principalmente com relação à: (1) utilização de visualização de programas e algoritmos, (2) utilização de serious games (3) o desenvolvimento de ambientes pedagógicos para o ensino e aprendizagem de programação. (SOUZA et al., 2016, p.48)

Os dois surveys, por Arimoto e Oliveira (2019) e Moreira, Holanda, Coutinho e Chagas (2018) e o mapeamento sistemático de Souza, Silva e Barbosa (2016) chegam a conclusões similares quanto à dificuldades no ensino de programação, apresentando alunos que têm problemas nas fases iniciais do aprendizado, sejam esses problemas com entendimento e aplicação de conceitos da lógica de programação ou com conteúdos de funções e estruturas de repetição.

3. JOGOS SÉRIOS

No mapeamento sistemático (Souza, Silva e Barbosa, 2016), mencionado anteriormente, além de analisar os problemas com o ensino de programação, buscou-se também apresentar as propostas de solução dos artigos estudados. Dentre as soluções, as mais populares entre os artigos foram a utilização de técnicas de visualização, onde busca-se apresentar o ambiente de programação em um formato que seja mais divertido e não ameaçador; além de *serious games* e ambientes de programação pedagógicos, que são ferramentas que possibilitam a criação e execução de programas, mas com um maior foco no ensino e aprendizagem.

Apesar de ser um livro com foco em jogos sérios de carta e tabuleiros, as definições apresentadas em "*Serious Games*", escrito por Abt (1970), ainda se encaixam em jogos sérios digitais. No livro, Abt define que "O paradoxismo de *serious games* une a seriedade do pensamento e dos problemas que o exigem com a liberdade experimental e emocional do jogo ativo. *Serious games* combinam a concentração analítica e questionadora do ponto de vista científico com a liberdade intuitiva e as recompensas de atos artísticos imaginativos". Conforme afirmam Belloti, Berta e De Gloria (2010), sobre tecnologias para aprendizagem:

> Essas tecnologias devem ser desenvolvidas e exploradas em uma abordagem multidisciplinar orientada para o objetivo que coloca os benefícios do usuário no centro do processo. [...] Nesse sentido, acreditamos ser importante ressaltar que, embora os jogos sérios sejam frequentemente vistos como *de fato* instrucionais, a combinação de entretenimento e aquisição de conhecimento está longe de ser imediata. (BELLOTI et al., 2010, p.24)

Ou seja, o *game design* aplicado a jogos sérios foge um pouco dos conceitos de *game design* tradicionais.

3.1. TRABALHOS RELACIONADOS

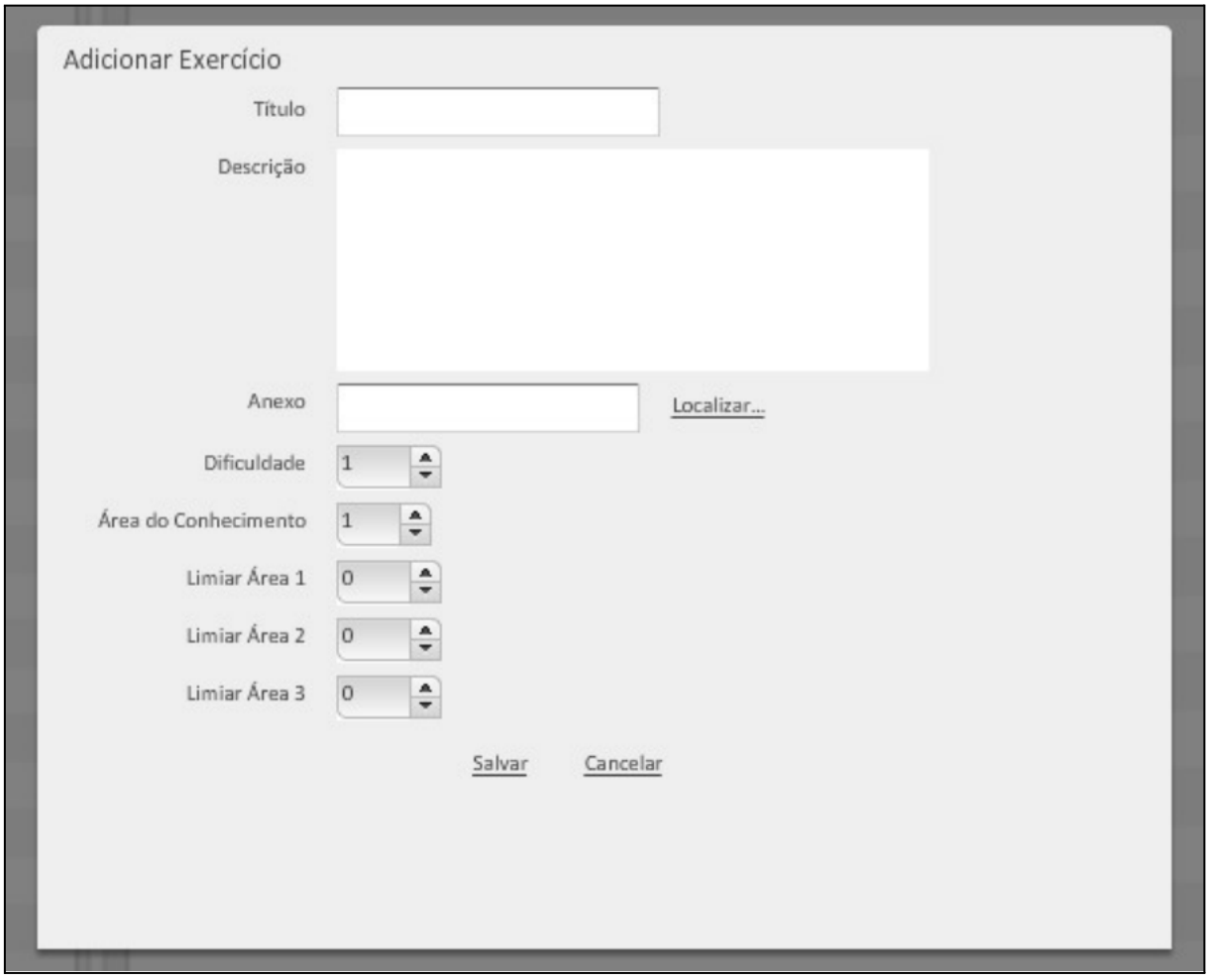
Outros trabalhos também abordam possíveis soluções para as dificuldades apresentadas no ensino e aprendizagem de disciplinas de programação. Entre eles, destacam-se os ambientes online BeeCrowd [1]e CodeCombat[2], além dos trabalhos de Zenato (2009), que apresenta um ambiente de aprendizagem, e de Severgnini (2016) e Lima (2017), que trazem propostas de *serious games*. Além desses, também é apresentado o jogo Final Fantasy XII.

[1] https://beecrowd.com.br/
[2] https://codecombat.com/

3.1.1. Ambiente de Aprendizagem para Programação

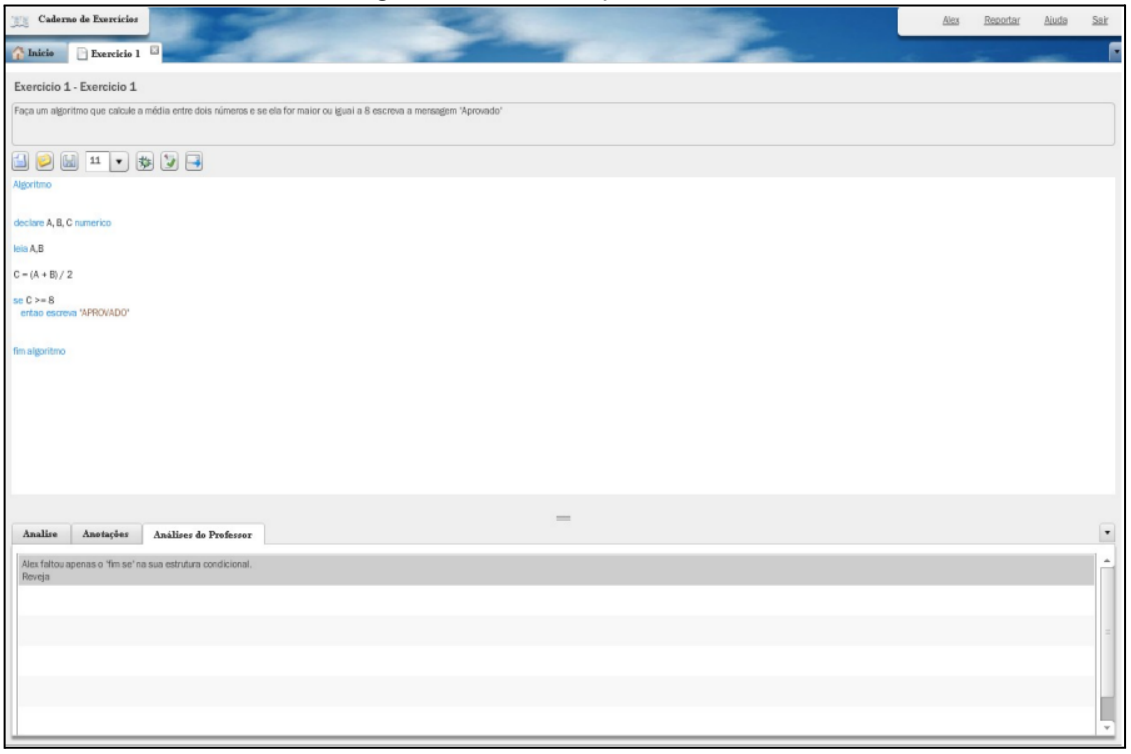
No trabalho de Zenato (2009), é proposto um ambiente de aprendizagem para programação. Nesse ambiente é possível ao professor cadastrar turmas, lançar conteúdos e exercícios para os alunos e, de acordo com a resolução desses exercícios, possibilita ao professor monitorar e avaliar o aprendizado dos alunos. Segundo Zenato, sobre o processo de aprendizagem: "O processo de aprendizagem é desencadeado a partir da motivação. [...] Nas situações escolares, o interesse é indispensável para que o aluno tenha motivos de ação no sentido de apropriar-se do conhecimento. [...] Cada aluno possui um potencial perceptivo próprio". As interfaces de cadastro e resolução de exercícios podem ser vistas nas Figuras 8 e 9.

Figura 8: tela de cadastro de exercícios



Fonte: (Zenato, 2009)

Figura 9: tela de resolução de exercício



Fonte: (Zenato, 2009)

O ambiente de aprendizagem proposto por Zenato (2009) tem como objetivo aproximar o professor de sua turma, procurando trazer um acompanhamento individualizado da aprendizagem de cada aluno. Ainda, sobre resolução de problemas e ensino de programação, Zenato afirma:

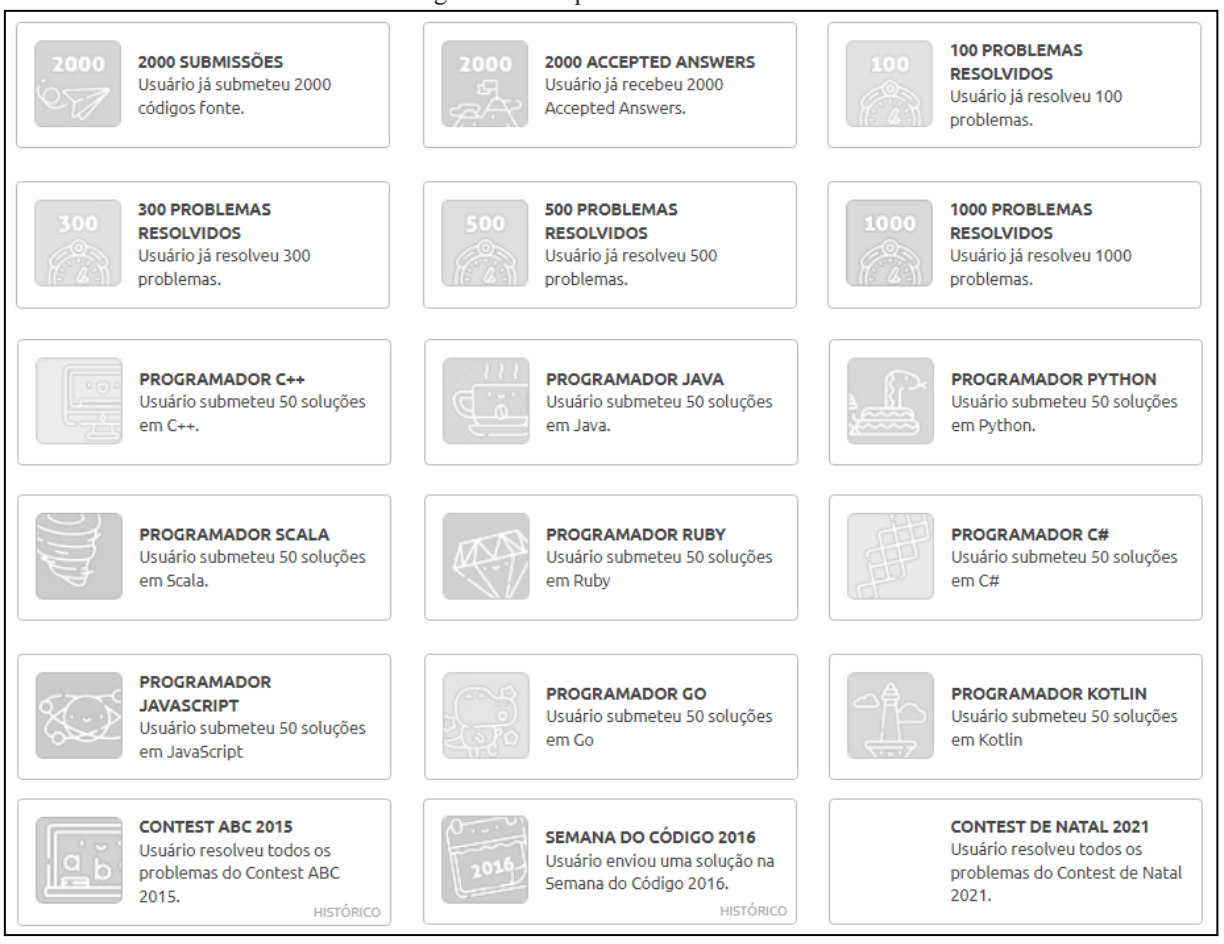
> Programar é uma atividade que tem como fundamento o desenvolvimento de algoritmos, que é a parte criativa do processo de programação, ficando a execução (reprodução de passos) para ser realizada pelo computador. Construir algoritmos requer raciocínio lógico, reflexão, pesquisa e envolvimento por parte do aluno para com o problema a ser modelado. [...] A habilidade de programação só pode ser desenvolvida através da prática. Porém, ao tentar escrever os primeiros programas, os alunos iniciantes passam por uma série de dificuldades [...] A falta de sucesso principalmente no estágio inicial de aprendizado levam a queda da auto-estima do aluno e dá a ele uma má impressão sobre a programação. (ZENATO, 2019, p.14)

Zenato, na conclusão de seu trabalho, apresenta os resultados dos testes realizados com o ambiente de aprendizagem desenvolvido, considerando-os aceitáveis de acordo com o esperado pelo autor; além disso, é destacado que, apesar do ambiente não substituir ferramentas já existentes, ele pode ser usado para auxiliar o acompanhamento da aprendizagem de alunos pelo professor.

3.1.2. BeeCrowd

O BeeCrowd, anteriormente conhecido como URI Online Judge, é uma plataforma online para auxiliar no aprendizado de programação, utilizando um conjunto com mais de 2000 exercícios em 20 diferentes linguagens de programação, que podem ser resolvidos diretamente na interface do sistema. Esse ambiente possibilita conectar professores e alunos por meio do BeeCrowd acadêmico, onde professores podem atribuir problemas de programação para os alunos, definindo grupos e limites de tempo para a resolução. Para incentivar os usuários e alunos, o BeeCrowd aplica conceitos de gamificação em sua plataforma, possuindo um sistema de *rankings*, tanto geral quanto para a resolução de cada problema, e de conquistas que incluem quantidade de respostas enviadas, problemas resolvidos e participação em eventos da plataforma, como pode ser observado na Figura 10.

Figura 10: conquistas do BeeCrowd



Fonte: BeeCrowd

O sistema de *rankings* da plataforma também possibilita aos alunos comparar sua pontuação geral no site com outros usuários e até mesmo comparar a pontuação de sua universidade com outras, como mostra a Figura 11, com o *ranking* de usuários, e a Figura 12, com o *ranking* de universidades.

Figura 11: *ranking* de usuários

RANKING	USUÁRIO	UNIVERSIDADE	PONTOS	ATIVO
1	[illegible]	FATEC-SJC	25.302,15	
2	[illegible]	-	24.669,64	
3	[illegible]	-	24.428,10	
4	[illegible]	MSU RU	21.927,31	
5	[illegible]	INATEL	21.296,75	
6	[illegible]	UFF	20.750,73	
7	[illegible]	UNICAMP	17.971,37	
8	[illegible]	UITS	17.928,33	
9	[illegible]	-	17.098,27	
10	[illegible]	INATEL	16.675,73	
11	[illegible]	-	16.507,93	
12	[illegible]	UNIOESTE	16.378,26	
13	[illegible]	UEM	15.539,75	
14	[illegible]	UFMG	15.513,59	
15	[illegible]	UNIVASF	14.675,40	
16	[illegible]	DCC	14.114,05	
17	[illegible]	UFMS	14.074,16	
18	[illegible]	UFABC	14.044,23	
19	[illegible]	FATEC-FRV	13.998,86	
20	[illegible]	USP	13.809,92	
21	[illegible]	IFCE	13.369,94	
22	[illegible]	-	13.297,98	
23	[illegible]	UTFPR	12.988,79	
24	[illegible]	FATEC-PG	12.752,00	
25	[illegible]	UERJ	12.742,68	
26	[illegible]	IFCE	12.696,07	

Fonte: BeeCrowd

Figura 12: *ranking* de universidades

RANKING	ACRÔNIMO	INSTITUIÇÃO	RESOLVIDO	ESTUDANTES
1	DIU	Daffodil International University	350.211	7.629
2	UNIFEI	Universidade Federal de Itajubá	188.080	2.828
3	UFU	Universidade Federal de Uberlândia	131.433	2.159
4	UFMS	Universidade Federal de Mato Grosso do Sul	123.239	2.668
5	UNB	Universidade de Brasília	117.763	2.393
6	INATEL	Instituto Nacional de Telecomunicações	114.706	1.794
7	UNINOVE	Universidade Nove de Julho	109.359	3.319
8	UTFPR	Universidade Tecnológica Federal do Paraná	108.488	3.498
9	IFSULDEMINAS	Instituto Federal do Sul de Minas Gerais	105.488	1.482
10	UNB-GAMA	Universidade de Brasília - Faculdade do Gama	103.334	1.439
11	UFF	Universidade Federal Fluminense	103.118	3.532
12	IFCE	Instituto Federal do Ceará	87.424	2.063
13	IFPB	Instituto Federal da Paraíba	80.684	1.729
14	FMM	Fundação Matias Machline	73.620	873
15	URI	Universidade Regional Integrada do Alto Uruguai e das Missões	72.776	878
16	IFRN	Instituto Federal do Rio Grande do Norte	71.413	1.582
17	PUCGO	Pontifícia Universidade Católica de Goiás	69.767	1.139
18	UFABC	Universidade Federal do ABC	66.141	1.646
19	BRACU	BRAC University	61.393	2.151
20	UFCG	Universidade Federal de Campina Grande	59.964	972
21	UNIOESTE	Universidade Estadual do Oeste do Paraná	57.180	717
22	UFC	Universidade Federal do Ceará	56.746	1.647
23	UFPB	Universidade Federal da Paraíba	55.097	1.404
24	IFTM	Instituto Federal do Triângulo Mineiro	54.136	1.294
25	UFSCAR	Universidade Federal de São Carlos	52.818	1.275
26	UFRPE	Universidade Federal Rural de Pernambuco	52.680	821

Fonte: BeeCrowd

3.1.3. CodeCombat

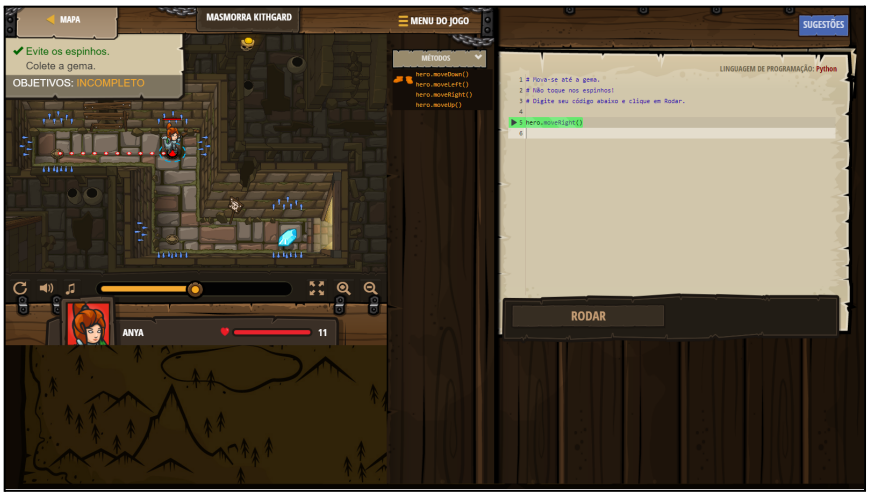
CodeCombat é uma plataforma *online* de aprendizagem de programação, disponibilizando aos professores a criação de aulas que utilizam uma série de *serious games*

para que o aluno possa aprender a programar em linguagens como Python, JavaScript e C++. De acordo com a própria equipe da plataforma (CodeCombat Teacher Getting Started Guide):

> CodeCombat é um jogo de codificação educacional que usa código digitado real e aprendizado personalizado para ensinar ciência da computação a alunos sem experiência anterior em ciência da computação. Com nossa abordagem exclusiva, os alunos adotam o aprendizado enquanto jogam e escrevem códigos desde o início de sua aventura, promovendo o aprendizado ativo e uma mentalidade de crescimento. O CodeCombat é recomendado para distritos, escolas, clubes e programas com alunos de 9 anos ou mais. Nossa missão é envolver todos os alunos, independentemente da experiência de codificação, e ajudá-los a descobrir um caminho para o sucesso em Ciência da Computação. (CODECOMBAT, 2021)

A plataforma utiliza uma interface simples, porém atrativa, para que os alunos comecem a escrever código desde o início do jogo. A interface pode ser observada na Figura 13. A tela do jogo se encontra do lado esquerdo, com os objetivos de cada fase aparecendo acima dessa tela. No lado direito da tela, o jogador tem o espaço onde ele pode programar o código que será rodado na fase. Por fim, na parte central da interface, o jogo lista os possíveis métodos que o jogador pode usar para programar e, ao selecionar um desses métodos, o jogo apresenta uma documentação que descreve o método, os parâmetros necessários e exemplifica como ele pode ser usado no código. Um exemplo dessa documentação pode ser visto na Figura 14.

Figura 13: interface de jogo da plataforma CodeCombat



Fonte: CodeCombat

Figura 14: documentação dos métodos disponíveis para o jogador

Fonte: CodeCombat

3.1.4. Serious Game “Alice e o Mistério dos Algoritmos”

Severgnini (2016), em seu trabalho sobre *serious games* para ensino de lógica de programação, afirma, em relação a aprendizagem em jogos sérios: “Para desenvolver um serious game, um jogo que tem como foco principal ensinar ao invés de entreter, são necessárias habilidades além do game design propriamente dito. Uma dessas habilidades é saber ensinar”. Em seu trabalho, Severgnini desenvolveu um jogo sério chamado de “Alice e o Mistério dos Algoritmos”. O jogo é baseado em *puzzles* de programação em papel quadriculado, onde o jogador deve fornecer uma sequência de instruções para reproduzir o resultado esperado, conforme pode ser observado na Figura 15.

Figura 15: exemplo de um puzzle resolvido em *Alice e o Mistério dos Algoritmos*

Fonte: (Severgnini, 2016)

O autor também levanta a importância do jogo sério fazer o mesmo papel de um professor no processo de aprendizagem, ensinando ao jogador os conceitos desenvolvidos pelo jogo até o ponto em que esse jogador consiga aplicar os conhecimentos desenvolvidos sem que o jogo precise guiá-lo através do processo. Além disso, sobre a possibilidade de quantificar a diversão em jogos sérios, Severgnini afirma que:

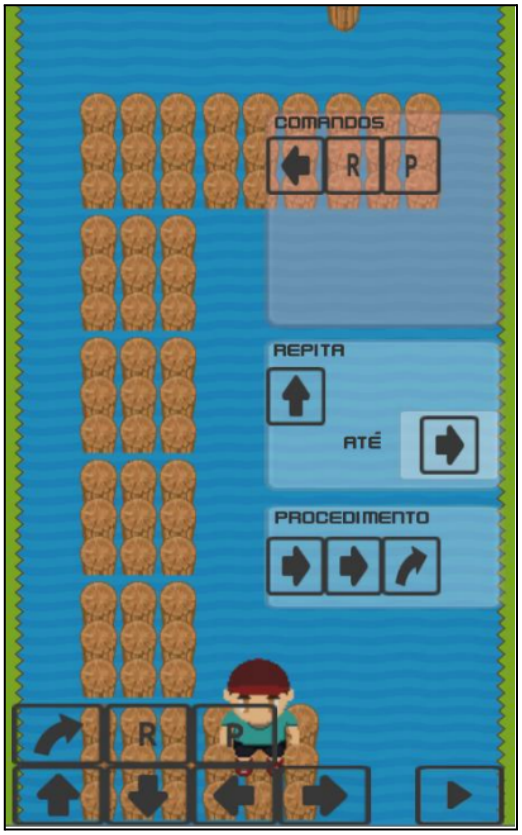
> O primeiro ponto é: *o jogo é divertido?* Conforme discutido no referencial teórico, a diversão é subjetiva. Logo, foi necessário procurar por um método que, de alguma forma, pudesse estimar e mensurar a diversão proporcionada pelo jogo. [...] Para verificar a resposta para essa pergunta, foi realizada uma sessão de *playtesting*. O teste consistia em jogar os quatro níveis do jogo e, em seguida, responder a um questionário. As perguntas do questionário eram diretamente ligadas aos fatores de diversão que compõem os limiares do modelo de três níveis de diversão. Os resultados do *playtesting* indicaram que o jogo conseguiu atingir os limiares de jogabilidade e diversão. Desta forma, pode-se dizer que a questão de pesquisa foi parcialmente respondida, uma vez que foi possível verificar que o jogo é divertido. (SEVERGNINI, 2016, p.57)

Por ser um conceito subjetivo, que varia de jogador para jogador, a diversão em um jogo, seja ele sério ou não, precisa levar em consideração o contexto geral, pensando em como levar entretenimento ao maior número de jogadores, sem que o jogo esqueça que sua ideia principal ainda deve ser a aprendizagem. Para isso, um jogo sério deve considerar aspectos de *game design*, estética e usabilidade em seu desenvolvimento.

3.1.5. Serious Game “As aventuras de Salazar”

No trabalho de Lima (2017), é proposto um *serious game* para aprendizagem de pensamento computacional. Para seu trabalho, Lima utilizou a ferramenta Unity3D para desenvolver um jogo voltado a plataformas *mobile*. No jogo, o jogador deve fornecer uma sequência de instruções, desenvolvendo um algoritmo para que o personagem possa se mover pelos níveis do jogo, como pode ser observado na Figura 16. Na interface do jogo, os comandos repita e procedimento são, inicialmente, pré-definidos e armazenados nas variáveis R e P, para que o jogador possa usá-los no campo de Comandos. O jogo busca, a partir da definição destas variáveis, ensinar ao jogador conceitos de funções e procedimentos.

Figura 16: tela do jogo desenvolvido por Lima



Fonte (Lima, 2017)

O autor afirma, sobre o desenvolvimento de um jogo sério:

> Apesar dos serious games terem seu propósito no intuito de agregar o conhecimento, o entretenimento é tão necessário para o mesmo quanto para jogos comerciais, uma vez que o seu propósito é utilizar as qualidades de um jogo para tornar a aprendizagem um processo agradável. [...] O foco de um serious game sempre deve ser a aprendizagem. No entanto, o entretenimento também deve ser integrado de forma satisfatória, para que o serious game não perca a sua essência: um jogo. (LIMA, 2017, p.25)

Na conclusão de seu trabalho, Lima apresenta os resultados do *playtesting* com o público alvo, destacando a importância da fase de *playtest* e da organização do código-fonte para o desenvolvimento do projeto, para facilitar novas implementações que não foram previamente planejadas.

3.1.6. Final Fantasy XII

Final Fantasy XII é um RPG desenvolvido e publicado pela Square Enix, lançado originalmente em 2006 exclusivamente para o PlayStation 2 e, posteriormente, remasterizado sob o nome Final Fantasy XII: Zodiac Age para PlayStation 4 e Xbox One. Ao contrário de outros jogos da franquia, Final Fantasy XII não utiliza encontros aleatórios ou combate por turnos, optando por combate em tempo real com inimigos que aparecem no mundo aberto do jogo. O grande diferencial de Final Fantasy XII é o sistema de *gambits*, que possibilita ao jogador programar os personagens de seu grupo para que estes executem determinadas ações em resposta a situações específicas. Conforme escrito por Fabro (2019) em artigo para o dev.to, "Sinto que isso [o sistema de *gambits*] é uma boa introdução à programação sem realmente conhecer o código [...]. É como se estivéssemos programando nossos próprios aliados de IA do modo que queremos que eles ajam". Com toda a customização disponível através do sistema de *gambits*, o jogo pode facilmente chegar a um ponto onde ele estará "jogando sozinho", sem a necessidade do jogador controlar diretamente qualquer personagem. Esse sistema de *gambits*, que serviu de inspiração para o jogo a ser desenvolvido como proposta de solução, pode ser observado na Figura 17.

Figura 17: o sistema de *gambits* de Final Fantasy XII

Fonte: Final Fantasy XII - The Zodiac Age

O sistema de *gambits* de Final Fantasy XII, conforme pode ser visto na Figura 17, funciona como uma sequência de declarações no formato "se… então…", com a condição

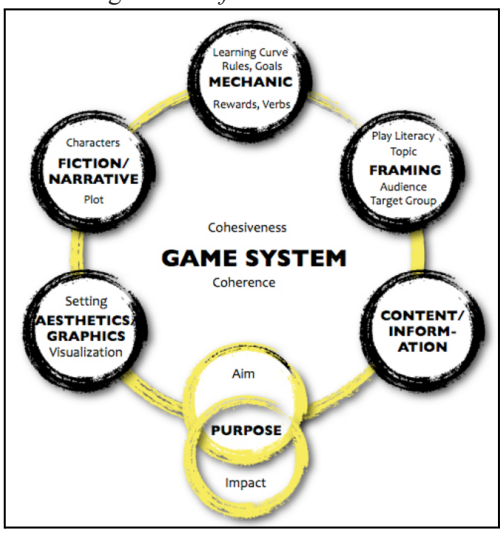
ficando na parte esquerda da linha, e a ação no lado direito. Cada personagem tem uma série de instruções numeradas de 1 a 12, que, depois de configuradas pelo jogador, podem ser ligadas ou desligadas no botão "ON" à esquerda de cada linha. Essas linhas possuem prioridade de cima para baixo, com o personagem executando uma ação apenas se a anterior não for possível. Por exemplo, entre as instruções 6 e 11 da Figura 17, o personagem vai executar uma série de testes sobre o inimigo, tentando descobrir se ele é fraco contra magias dos elementos fogo (6), eletricidade (7), gelo (8), vento (9) e sagrado (10); caso todos esses testes falharem, o personagem ataca o inimigo com os pontos de vida mais baixos (11).

3.2. O *FRAMEWORK SGDA*

Tendo em vista a falta de ferramentas para avaliação de *design* focadas em jogos sérios, Mitgutsch e Alvarado (2012) desenvolveram um *framework* para avaliação de *design* para jogos sérios, o SGDA (*Serious Game Design Assessment*). O framework SGDA propõe uma ferramenta para que os conceitos de *game design* possam ser melhor estudados e avaliados quando aplicados a um jogo sério. Conforme ilustrado pela Figura 18, a pesquisa desenvolvida por Mitgutsch e Alvarado (2012) identificou seis principais componentes para a estrutura conceitual de um jogo sério:

> Se considerarmos os jogos sérios como sistemas de jogos baseados em propósitos, a força motriz que funciona como a influência central sobre os elementos do design do jogo deve ser o *propósito* (1) do jogo. Assim, o objetivo deve ser refletido em todos os elementos que suportam o sistema de jogo: *conteúdo* (2), a *ficção e narrativa* (3), a *mecânica* (4), a *estética e gráficos* (5) e o *enquadramento* (6). A relação entre esses seis componentes principais impacta a coerência e coesão do projeto conceitual formal da holística do sistema do jogo. (MITGUTSCH; ALVARADO, 2012, p.123)

Figura 18: o *framework* SGDA



Fonte: Mitgutsch e Alvarado (2012)

Explorando o *framework*, os autores explicam os seis elementos fundamentais para a coerência de um sistema de jogo sério da seguinte forma:

a) Propósito: a primeira etapa que deve ser levada em conta na hora de projetar um jogo sério é o propósito dele, identificando qual ideia ou mensagem o jogo deve passar para o jogador.

b) Conteúdo: o conteúdo e informação de um jogo sério são os dados, estatísticas e outras informações pertinentes que são apresentadas ao jogador. Além disso, esses dados podem ser separados e examinados de acordo com o contexto em que se apresentam

c) Mecânica: as mecânicas são o principal meio pelo qual o jogador vai interagir com o jogo, elas englobam o objetivo central do jogo, como funciona o sistema de recompensas, quais os principais obstáculos/desafios, como funciona o balanceamento de dificuldade e quais as condições de vitória

d) Ficção e Narrativa: A narrativa e o ambiente de um jogo sério devem levar em conta o propósito do jogo, buscando elevar a mensagem ou ideia que o jogo quer levar aos jogadores.

e) Estética e Gráficos: a estética de um jogo sério não é o primeiro elemento a ser projetado, porém é o primeiro elemento que vai ser apresentado ao jogador, portanto "desempenha um papel fundamental na introdução do propósito do jogo e seu impacto no jogador". Esse elemento não leva em conta apenas os gráficos do jogo, mas também toda a linguagem audiovisual, desde o som, imagem e a presença ou não de animações *in-game* que apresentam o contexto ficcional.

f) Enquadramento: o "enquadramento" de um jogo sério deve levar em conta todos os outros elementos citados e pensar na forma como esses elementos vão ser apresentados, levando em conta quem é o público alvo do jogo. Para isso, o *framework* levanta questões como "O público alvo tem problemas com controles, interface ou com a ficção do jogo? Quais são as habilidades necessárias? [...] Os níveis de dificuldade estão balanceados de acordo com as necessidades do público?" Um jogo sério deve ser de fácil acesso ao público, mas ainda assim precisa engajar o jogador com um *gameplay* balanceado.

O objetivo do *Framework* SGDA não é somente apresentar e analisar os elementos fundamentais para um jogo sério, mas também mostrar como esses sistemas interagem com o jogo e entre si para levar a melhor experiência para o jogador. Circulando ao redor do propósito do jogo, todos esses elementos devem coexistir de forma a introduzir ao jogador as ideias e conceitos que o jogo sério pretende abordar. De acordo com Mitgutsch e Alvarado, na

conclusão do artigo sobre o SGDA, “Se os jogos sérios são projetados para serem projetos baseados em propósitos e pretendem impactar seus jogadores, seu propósito precisa ser considerado em todos os componentes de design.”

3.3. O MODELO MEEGA+ PARA A AVALIAÇÃO DE JOGOS EDUCACIONAIS

O modelo de avaliação MEEGA+ se concentra em jogos educacionais. Na proposta de Petri, Wangenheim e Borgatto (2019), afirma-se que jogos educacionais são uma estratégia bastante válida cientificamente porém existem poucas ferramentas que possibilitem a avaliação desses jogos. Uma dessas possíveis avaliações é o modelo MEEGA (*Model for the Evaluation of Educational GAmes*), proposta em 2011. O modelo MEEGA+ é uma proposta de evolução com base na proposta original.

Com base na avaliação de outros oito trabalhos que avaliam *design* de jogos sérios ou educacionais, sendo três deles *frameworks*, dois modelos, duas escalas e uma metodologia, onde foram considerados os fatores analisados por esses trabalhos, o modelo MEEGA+ desenvolveu sua própria análise para esses fatores, levando em consideração possíveis melhorias identificadas. Levando em conta a análise feita, o modelo MEEGA+ apresenta um questionário que apresenta respostas em uma escala Likert de 5 pontos que varia de “discordo totalmente” a “concordo totalmente”, com objetivo de permitir a expressão de opinião com uma maior precisão. Esse questionário pode ser observado no Quadro 11.

Quadro 11: Itens do Questionário do modelo MEEGA+

Dimensão/Subdimensão		**Item**	**Descrição do Item**
Usabilidade	Estética	1	O design do jogo é atraente (interface, gráficos, tabuleiro, cartas, etc.).
		2	Os textos, cores e fontes combinam e são consistentes.
	Aprendizibilidade	3	Eu precisei aprender poucas coisas para poder começar a jogar o jogo.
		4	Aprender a jogar este jogo foi fácil para mim

		5	Eu acho que a maioria das pessoas aprenderiam a jogar este jogo rapidamente.
	Operabilidade	6	Eu considero que o jogo é fácil de jogar.
		7	As regras do jogo são claras e compreensíveis.
	Acessibilidade	8	As fontes (tamanho e estilo) utilizadas no jogo são legíveis.
		9	As cores utilizadas no jogo são compreensíveis
		10	O jogo permite personalizar a aparência (fonte e/ou cor) conforme a minha necessidade.
	Proteção contra erros do usuário	11	O jogo me protege de cometer erros.
		12	Quando eu cometo um erro é fácil de me recuperar rapidamente
Confiança		13	Quando olhei pela primeira vez o jogo, eu tive a impressão de que seria fácil para mim.
		14	A organização do conteúdo me ajudou a estar confiante de que eu iria aprender com este jogo.
Desafio		15	Este jogo é adequadamente desafiador para mim.
		16	O jogo oferece novos desafios (oferece novos obstáculos, situações ou variações) com um ritmo adequado.
		17	O jogo não se torna monótono nas suas tarefas (repetitivo ou com tarefas chatas)
Satisfação		18	Completar as tarefas do jogo me deu um sentimento de realização

	19	É devido ao meu esforço pessoal que eu consigo avançar no jogo.
	20	Me sinto satisfeito com as coisas que aprendi no jogo.
	21	Eu recomendaria este jogo para meus colegas.
Interação social	22	Eu pude interagir com outras pessoas durante o jogo.
	23	O jogo promove momentos de cooperação e/ou competição entre os jogadores.
	24	Eu me senti bem interagindo com outras pessoas durante o jogo.
Diversão	25	Eu me diverti com o jogo.
	26	Aconteceu alguma situação durante o jogo (elementos do jogo, competição, etc.) que me fez sorrir
Atenção focada	27	Houve algo interessante no início do jogo que capturou minha atenção.
	28	Eu estava tão envolvido no jogo que eu perdi a noção do tempo.
	29	Eu esqueci sobre o ambiente ao meu redor enquanto jogava este jogo.
Relevância	30	O conteúdo do jogo é relevante para os meus interesses
	31	É claro para mim como o conteúdo do jogo está relacionado com a disciplina.
	32	O jogo é um método de ensino adequado para

		esta disciplina.
	33	Eu prefiro aprender com este jogo do que de outra forma (outro método de ensino).
Aprendizagem percebida	34	O jogo contribuiu para a minha aprendizagem na disciplina.
	35	O jogo foi eficiente para minha aprendizagem, em comparação com outras atividades da disciplina

Fonte: Petri, Wangenheim e Borgatto (2019)

O modelo MEEGA+ foi utilizado para avaliar o jogo desenvolvido como proposta de solução para este trabalho.

4. O JOGO *CODE_DUNGEON*

Como solução, esse trabalho buscou planejar um jogo sério para ser desenvolvido na disciplina de Trabalho de Conclusão de Curso II. O jogo é um RPG (*Role Playing Game*) de turnos, ou seja, um jogo onde o jogador controla personagens com pontos fortes, fracos e habilidades pré-definidas por classes; e onde o combate funciona com uma ordem de turnos, onde, durante o combate, cada personagem toma suas ações de acordo com a ordem de turnos da batalha. Além disso, o jogo proposto tem mecânicas de combate inspiradas no sistema de *gambits* do jogo Final Fantasy XII, que foram explicadas na seção 3.1.6. No jogo desenvolvido, o jogador tem como objetivo programar os personagens jogáveis, buscando automatizar o combate de acordo com métodos pré-definidos.

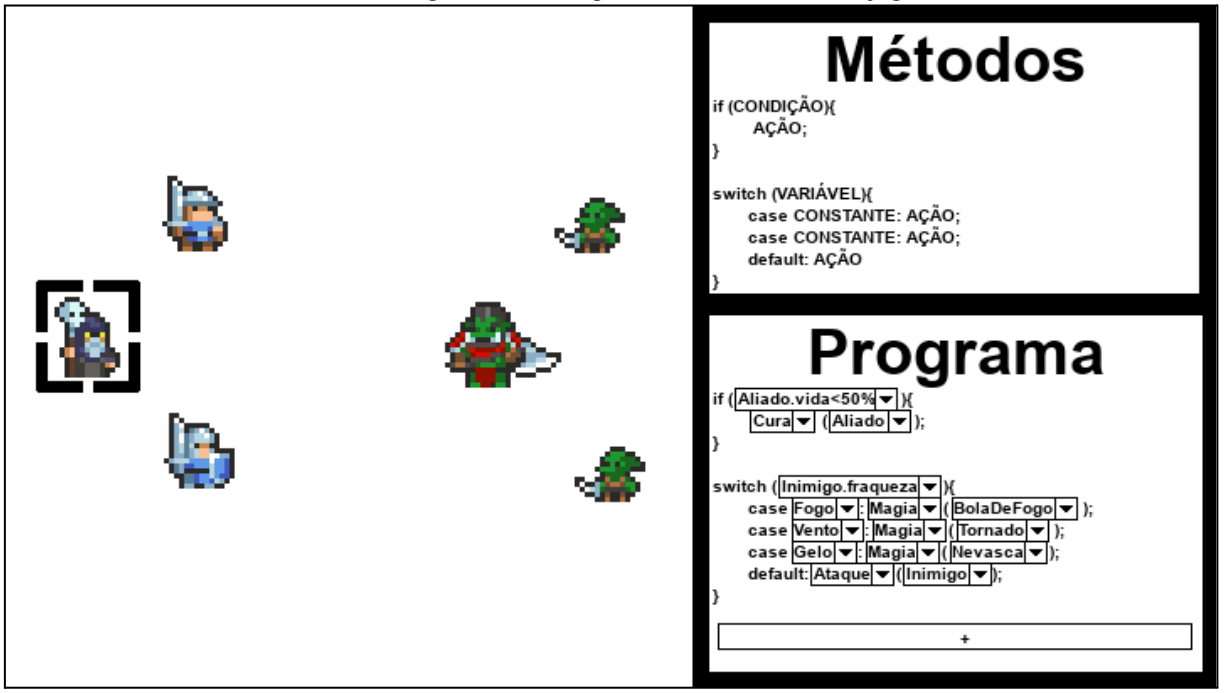
4.1. METODOLOGIA

Para o desenvolvimento do jogo, foi usada a ferramenta Unity3D, tendo como base para o *design* do jogo os elementos propostos pelo *framework* SGDA.. Para a avaliação do jogo desenvolvido, foram feitos *playtests* do jogo com o público alvo; utilizando o questionário desenvolvido pelo método MEEGA+ para poder quantificar a satisfação dos jogadores que participaram da sessão de testes do jogo.. Além do MEEGA+, foi desenvolvido um segundo questionário para os jogadores, que tem como objetivo avaliar o aprendizado dos conteúdos desenvolvidos pelo jogo.

4.2. DESENVOLVIMENTO

Neste capítulo são detalhadas as etapas de desenvolvimento do jogo proposto, de acordo com o *framework* SGDA, cujas etapas são: propósito, conteúdo, mecânicas, ficção e narrativa, estética e gráficos e enquadramento. Na Figura 19 é apresentado o protótipo de tela da interface de combate do jogo, que será usada como base para explorar os elementos do SGDA.

Figura 19: Protótipo de tela da interface do jogo



Fonte: Elaborado pelo autor (2022)

4.2.1. Propósito

O propósito principal do jogo gira em torno do aprendizado de programação. O jogo desenvolvido busca levar o jogador a aprender conceitos de programação, como estrutura de condicionais. Para isso, o *game design* foi feito de forma a ensinar esses conceitos ao jogador, mas também deixá-lo solucionar alguns dos problemas apresentados de forma livre, balanceando a quantidade de níveis que vão servir de tutorial para as mecânicas com níveis que ficarão abertos para o jogador aplicar os conhecimentos desenvolvidos até aquele ponto.

4.2.2. Conteúdo

O conteúdo é toda a informação apresentada ao jogador através da interface do jogo. No nível mais básico, é apresentado a classe de cada personagem, seus pontos de vida e quais habilidades esse personagem possui; além disso, os métodos utilizáveis para o jogador desenvolver a programação de cada personagem também estão disponíveis na interface de jogo. Quanto a estatísticas presentes, o jogo poderá apresentar a quantidade de turnos que o jogador demorou para concluir cada nível, além de quantos métodos foram usados, tentando levar ao jogador a ideia de otimização do programa desenvolvido.

4.2.3. Mecânica

O objetivo principal de cada nível do jogo é derrotar todos os inimigos presentes no menor número de turnos possíveis. Esse *feedback* sobre a quantidade de turnos é apresentado ao jogador no final de cada nível, deixando aberta a possibilidade do jogador repetir um nível

para tentar otimizar o código desenvolvido, de forma a conseguir terminar em uma quantidade menor de turnos.

4.2.4. Ficção e Narrativa

A narrativa do jogo baseia-se numa batalha de fantasia medieval, procurando guiar o jogador pelas fases. Para isso, tutoriais e dicas ficam disponíveis para o jogador no início de cada fase, apresentando o objetivo ao jogador e como cada função pode ser utilizada na programação dos personagens.

4.2.5. Estética e Gráficos

Por se tratar de um jogo do gênero RPG, o jogo tem como tema principal da estética elementos medievais e de fantasia, se utilizando de pacotes de *assets* adquiridos para o desenvolvimento do jogo. Os elementos gráficos do jogo, sejam eles personagens, cenários ou fontes, usam a estética de pixel art.

Para os personagens, foram utilizados os pacotes de *assets* MiniFolks, produzidos por LYASeeK (2022), que podem ser vistos na Figura 20. Os cenários utilizam o pacote Tiny Tales: Overworld, desenvolvido por megatiles (2019), conforme Figura 21. As fontes utilizadas para as interfaces do jogo foram as fontes de pixel dos pacotes Inkbit e New Hi-Score, produzidos por Rodrigo BSL (2022), conforme Figura 22 e Figura 23.

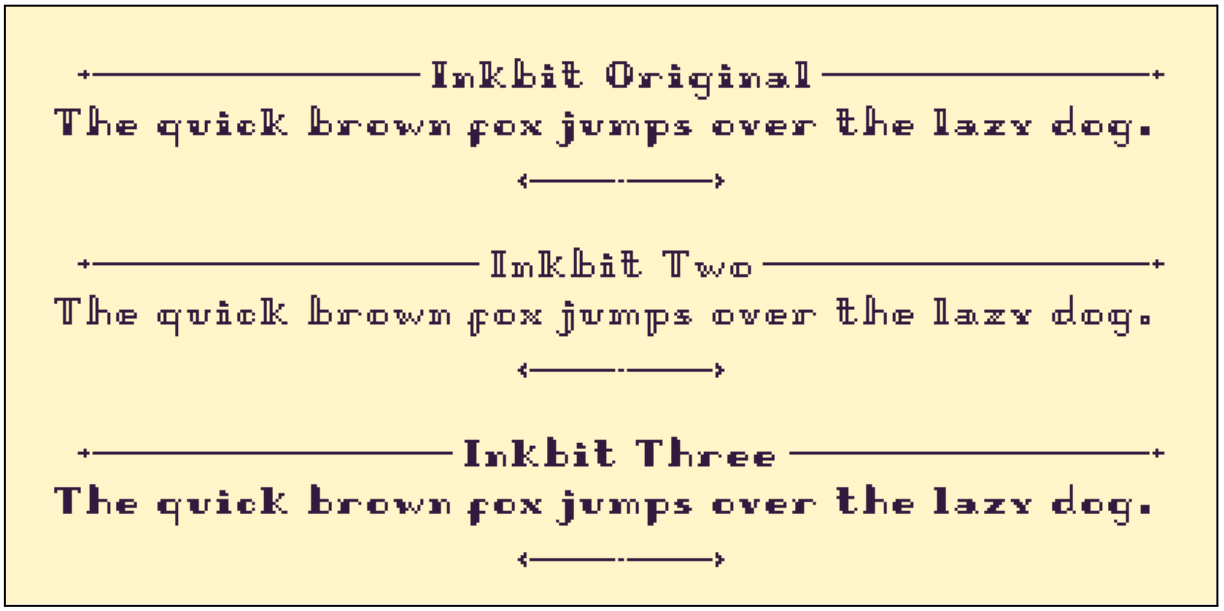
Figura 20: os três pacotes de *assets* MiniFolks

Fonte: LYASeek

Figura 21: o pacote de *assets* Tiny Tales: Overworld Tiles

Fonte: megatiles

Figura 22: o pacote de fontes Inkbit



Fonte: Rodrigo BSL

Figura 23: o pacote de fontes New Hi-Score

Fonte: Rodrigo BSL

4.2.6. Enquadramento

O público alvo do jogo são pessoas com conhecimento iniciante em programação, ou que já possuem interesse no assunto. Portanto, o jogo assume que o jogador tenha um conhecimento básico sobre programação, mas ainda assim explica os conceitos a serem desenvolvidos de forma a ajudar o jogador no entendimento dos conteúdos. O balanceamento dos níveis do jogo é feito levando em consideração se o nível é um tutorial sobre uma nova mecânica ou um nível livre para o jogador desenvolver seu código. Em níveis de tutorial o jogo deve "segurar a mão" do jogador e ensinar como determinada mecânica funciona no ambiente do jogo. Já em níveis abertos ao jogador, os desafios são apresentados de forma mais aberta, com algumas dicas sobre como cada código pode ser usado em determinadas situações.

4.2.7. Implementação

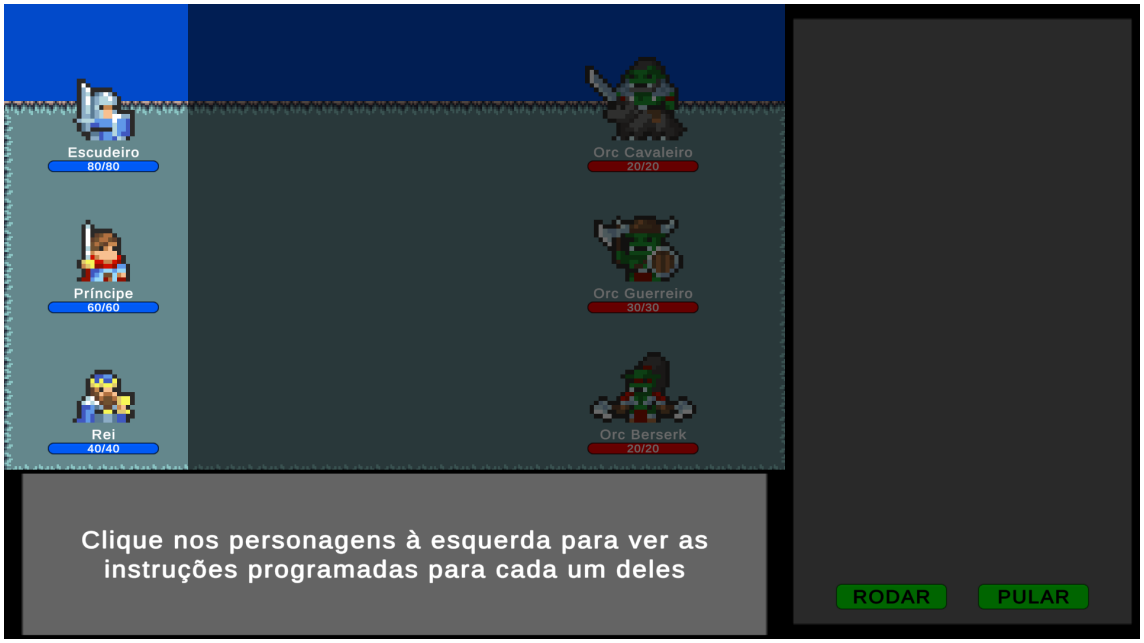
O jogo Code_Dungeon foi desenvolvido utilizando a *engine* Unity3D, usando a linguagem de programação *C#* e os pacotes de *assets* e fontes apresentados nos capítulos anteriores. A tela do menu inicial do jogo pode ser vista na Figura 24.

Figura 24: Menu inicial do jogo Code_Dungeon

Fonte: Elaborado pelo autor (2022)

O jogo possui cinco fases, sendo a primeira delas um tutorial para apresentar as mecânicas do jogo. Mostrando como funcionam as instruções programadas, e ensinando ao jogador como criar uma nova instrução para o personagem, conforme as figuras 25, 26, 27 e 28.

Figura 25: Etapa do tutorial introduzindo o jogador às instruções de personagens

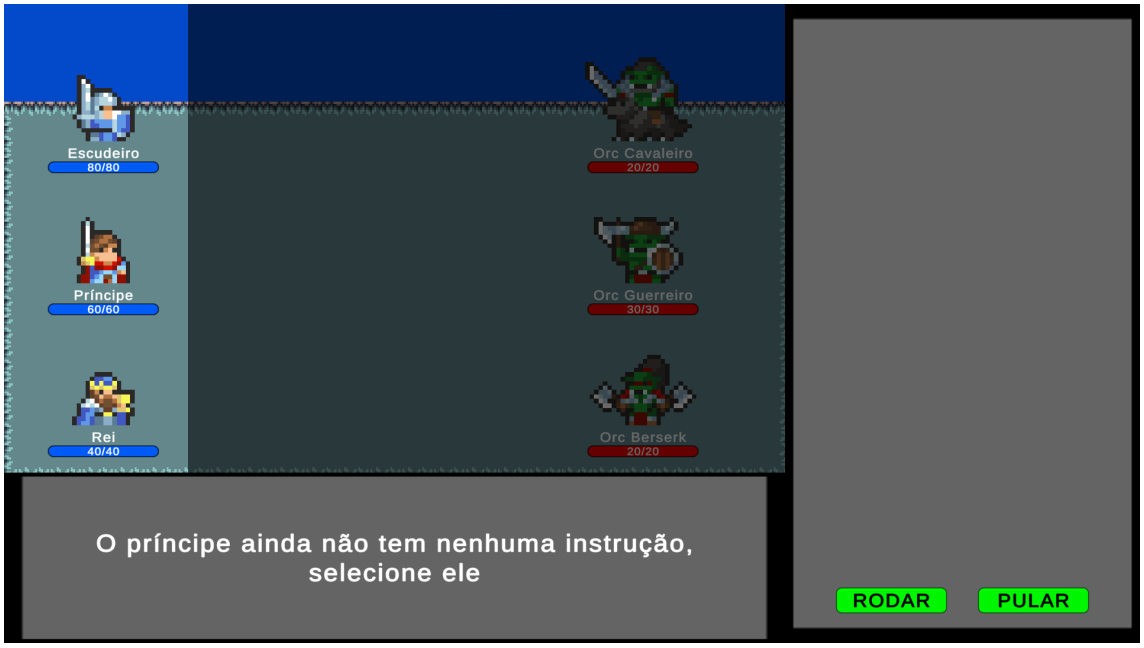


Fonte: Elaborado pelo autor (2022)

Figura 26: Etapa do tutorial demonstrando instruções já programadas aos personagens

Fonte: Elaborado pelo autor (2022)

Figura 27: Etapa do tutorial introduzindo o jogador à criação de instruções



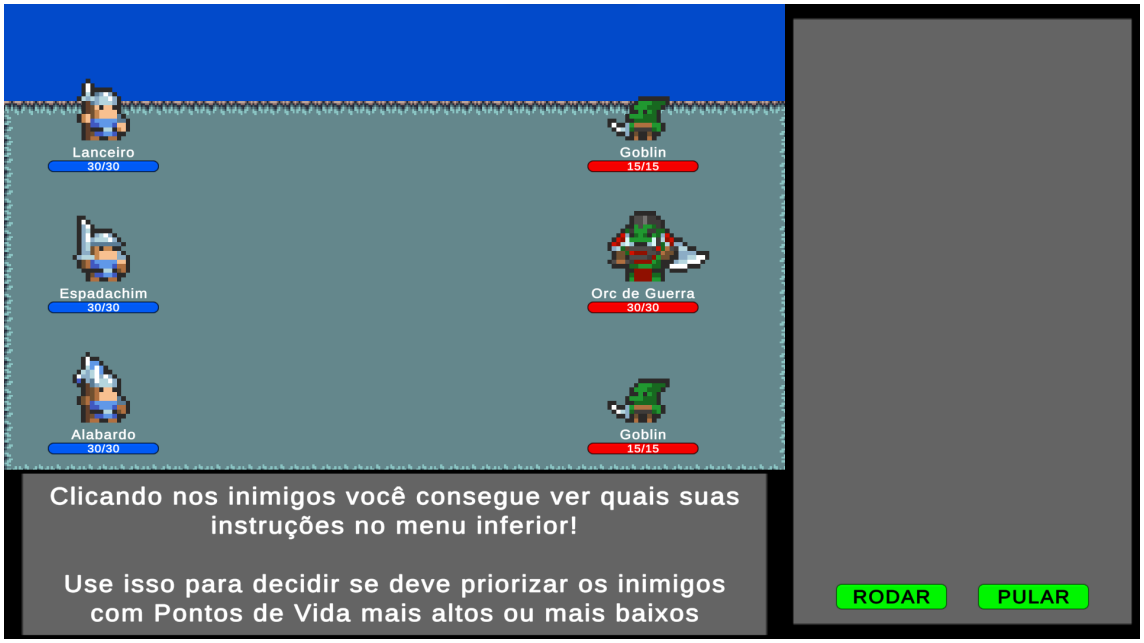
Fonte: Elaborado pelo autor (2022)

Figura 28: Interface de criação de instruções

Fonte: Elaborado pelo autor (2022)

A partir da segunda fase, o controle de criação de instruções para os personagens fica totalmente sob controle do jogador, porém o jogo ainda dá dicas no início de cada fase, além de introduzir novas mecânicas e possíveis instruções para o jogador, conforme figuras 29, 30 e 31.

Figura 29: Início da segunda fase, com dicas sobre comportamento dos inimigos e possíveis instruções

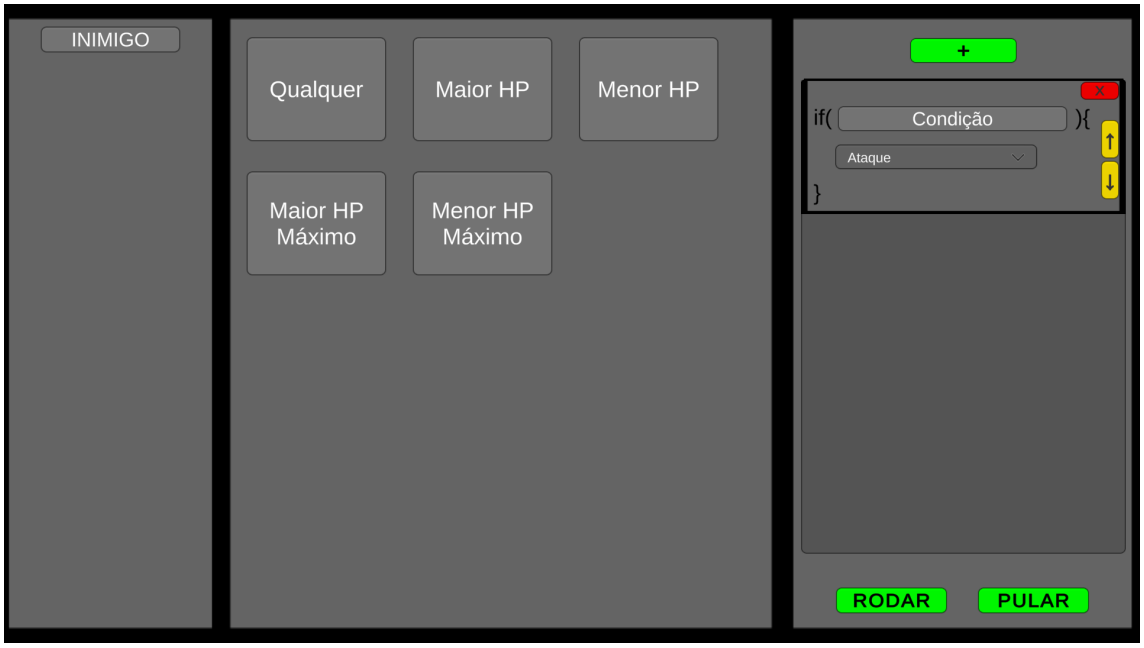


Fonte: Elaborado pelo autor (2022)

Figura 30: Interface com as instruções do inimigo selecionado

Fonte: Elaborado pelo autor (2022)

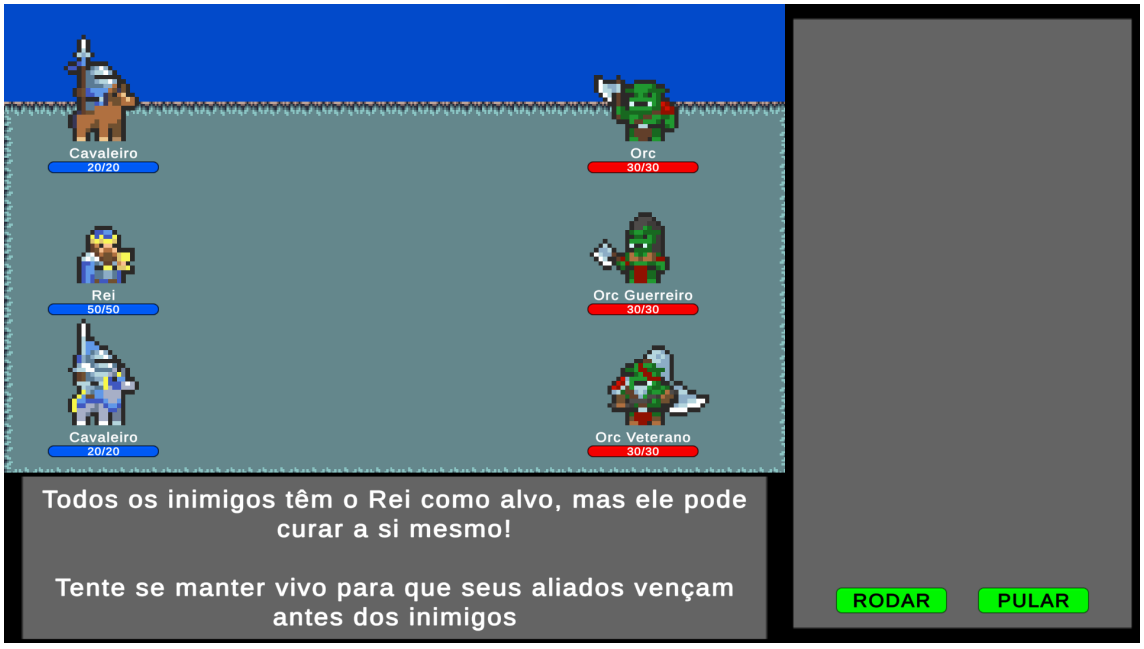
Figura 31: Criação de instruções na segunda fase do jogo



Fonte: Elaborado pelo autor (2022)

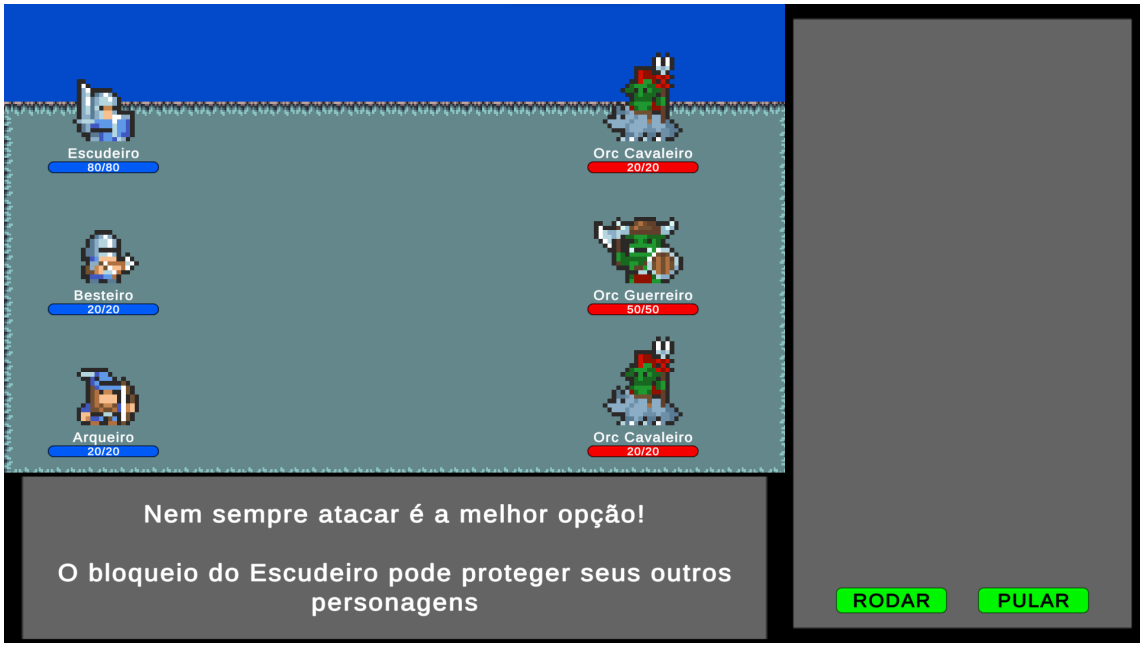
Durante a terceira e quarta fase, o jogador é apresentado a novas mecânicas, de cura e bloqueio, respectivamente. Para que possa utilizar essas mecânicas, o jogador tem a possibilidade de usar novas instruções, tendo seus aliados como alvo, como pode ser observado nas figuras 32, 33 e 34.

Figura 32: Início da terceira fase, com dica sobre a mecânica de cura



Fonte: Elaborado pelo autor (2022)

Figura 33: Início da quarta fase, com dica sobre a mecânica de bloqueio



Fonte: Elaborado pelo autor (2022)

Figura 34: Criação de instruções a partir da terceira fase do jogo

Fonte: Elaborado pelo autor (2022)

A quinta fase serve como um teste para os conhecimentos desenvolvidos pelo jogador nas fases anteriores. Ela não apresenta nenhuma dica ao jogador, apenas o objetivo de derrotar todos os inimigos, permitindo ao jogador utilizar todas as mecânicas desenvolvidas nas fases anteriores.

5. VALIDAÇÃO E RESULTADOS

Para a validação do jogo, foram feitos dois testes utilizando a ferramenta MEEGA+: no primeiro teste, com um grupo fechado de cinco pessoas, foi testada uma versão inicial do jogo, para identificar mudanças a serem feitas nas mecânicas e tutoriais, além de buscar um melhor balanceamento no sistema de combate; no segundo teste, aplicado com duas turmas de Programação de Computadores I, o jogo foi avaliado como ferramenta de ensino. Neste capítulo, será descrita a aplicação dos questionários do MEEGA+, além de apresentar os resultados dos testes realizados, utilizando algumas das respostas coletadas no teste de validação. As respostas para todas as perguntas do questionário podem ser visualizadas no Apêndice A.

5.1. PRÉ-TESTES

A etapa de pré-testes foi aplicada utilizando uma versão ainda incompleta do jogo, que possuía apenas a fase de tutorial e uma fase "livre" para o jogador explorar as mecânicas. Essa etapa foi aplicada com um grupo de cinco pessoas que já têm uma certa experiência com desenvolvimento de jogos e programação, e tinha como objetivo identificar *bugs* e falhas no jogo, além de coletar um primeiro *feedback* sobre possíveis alterações nas mecânicas do jogo.

Além da correção dos *bugs* identificados na etapa de pré-testes, os principais *feedbacks* desse primeiro teste foram sobre a dificuldade de compreender totalmente as mecânicas do jogo que, por estar numa versão inicial, não eram introduzidas a cada fase, como é na versão final do jogo. Para auxiliar na compreensão do jogador, além de introduzir mecânicas e instruções novas a cada fase, também foram adicionadas as dicas ao início de cada fase para guiar o jogador à solução, mas ainda o desafiando a desenvolver a resposta.

Outros *feedbacks* que também foram aplicados na versão final do jogo foram: o balanceamento do combate, para o qual foram alterados os números dos pontos de vida e ataque dos personagens; fases muito demoradas, por conta das animações de cada personagem, para o qual foi adicionado o botão de "pular", que é apresentado ao jogador no tutorial e acelera as ações de cada turno da batalha.

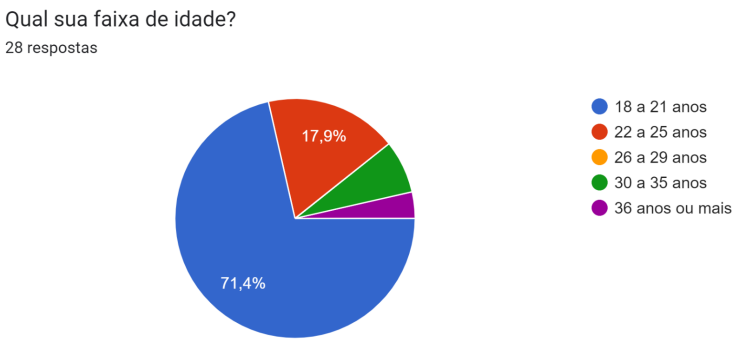
5.2. VALIDAÇÃO COMO FERRAMENTA DE ENSINO

Para a validação do jogo como ferramenta de ensino, o formulário MEEGA+ foi aplicado com duas turmas de Programação de Computadores I. No total, entre as duas turmas, 28 alunos participaram do processo de validação do jogo. Os testes foram realizados em trinta minutos e divididos em duas etapas. Na primeira etapa, durante os primeiros vinte minutos da

aplicação, os alunos jogaram o jogo individualmente. Durante a etapa do jogo, não foi feita nenhuma interrupção à experiência do aluno, para que estes pudessem jogar o jogo e explorar as mecânicas sem intervenção externa. Para a segunda etapa, nos últimos dez minutos da aplicação, os alunos foram convidados a preencher o questionário de participação, disponível via Google Forms.

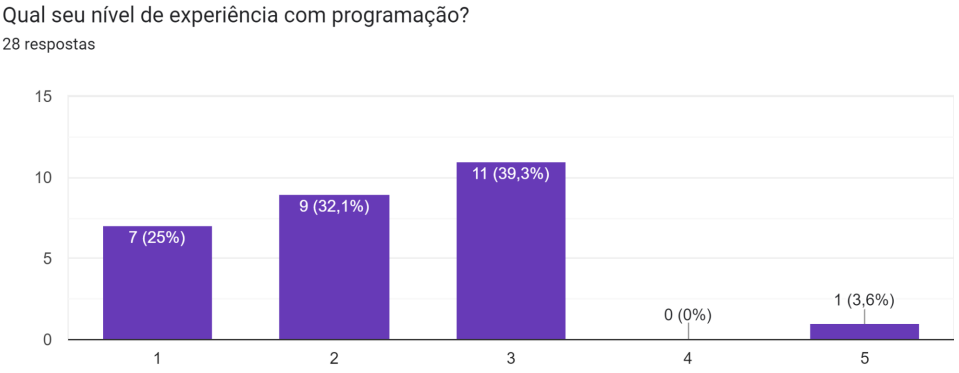
As primeiras perguntas do questionário tinham como objetivo identificar o público ao qual estava sendo aplicado o teste, contendo duas perguntas: a faixa etária dos jogadores e qual a experiência deles com programação. Essa segunda pergunta foi aplicada com uma escala Likert de 5 pontos, variando de "Nenhuma experiência" a "Muita experiência". As respostas a essas primeiras perguntas podem ser observadas nas Figuras 35 e 36.

Figura 35: Respostas à primeira pergunta do questionário



Fonte: Elaborado pelo autor (2022)

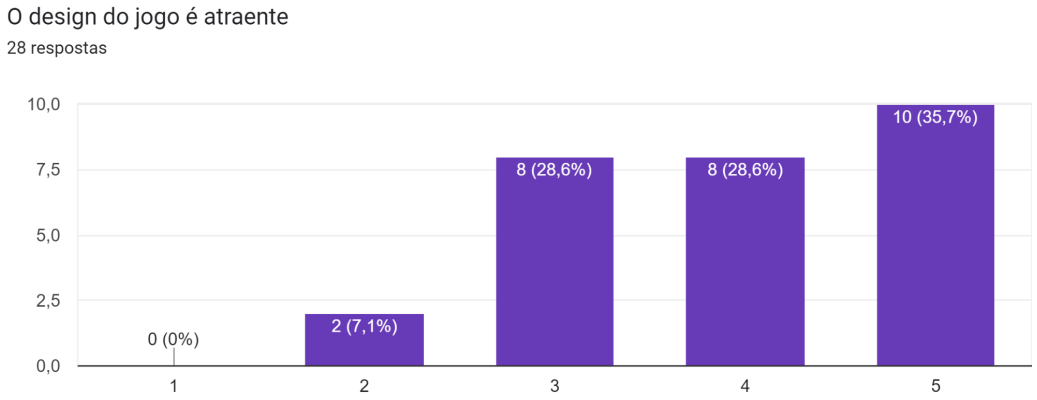
Figura 36: Respostas à segunda pergunta do questionário



Fonte: Elaborado pelo autor (2022)

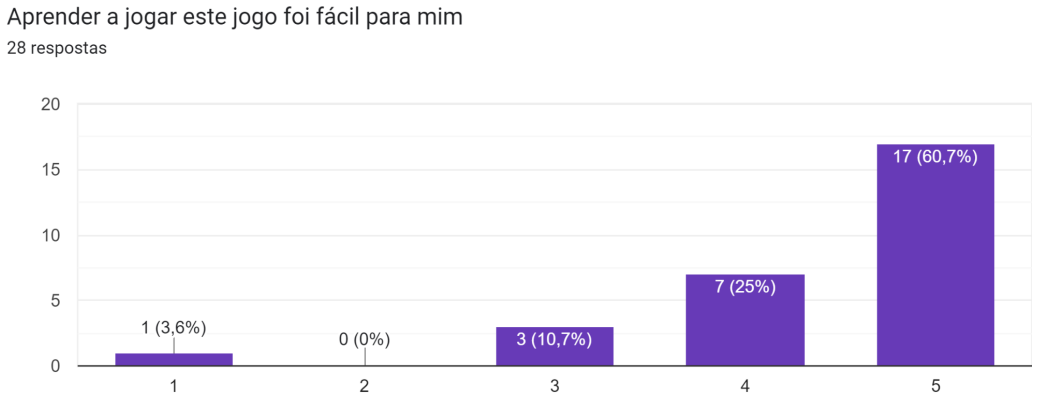
O público dos testes realizados identificou-se, portanto, como maioria na faixa de idade entre 18 a 25 anos e que identifica a experiência própria com programação de média ou inferior, com apenas três pessoas com idade superior a 25 anos e uma única resposta identificando-se como “Muita experiência” com programação. As perguntas seguintes foram divididas conforme as dimensões apresentadas pelo questionário do MEEGA+. No primeiro conjunto de perguntas, sobre Usabilidade e Confiança, os jogadores foram questionados sobre o design do jogo, a dificuldade em aprender o jogo, e a proteção contra erros.

Figura 37: Respostas à pergunta sobre o design do jogo



Fonte: Elaborado pelo autor (2022)

Figura 38: Respostas à pergunta sobre facilidade com o jogo

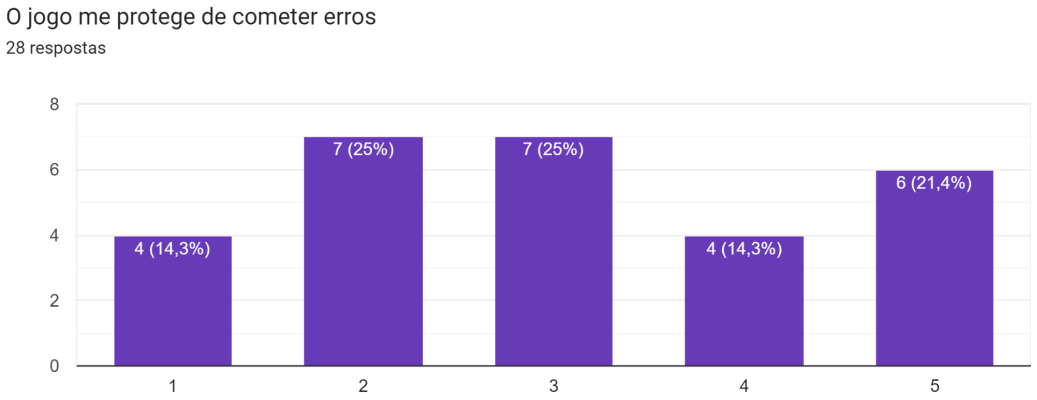


Fonte: Elaborado pelo autor (2022)

Conforme pode ser observado nos gráficos apresentados nas Figuras 37 e 38, o jogo teve uma boa aprovação quanto ao design, e aprender a jogar foi relativamente fácil para os

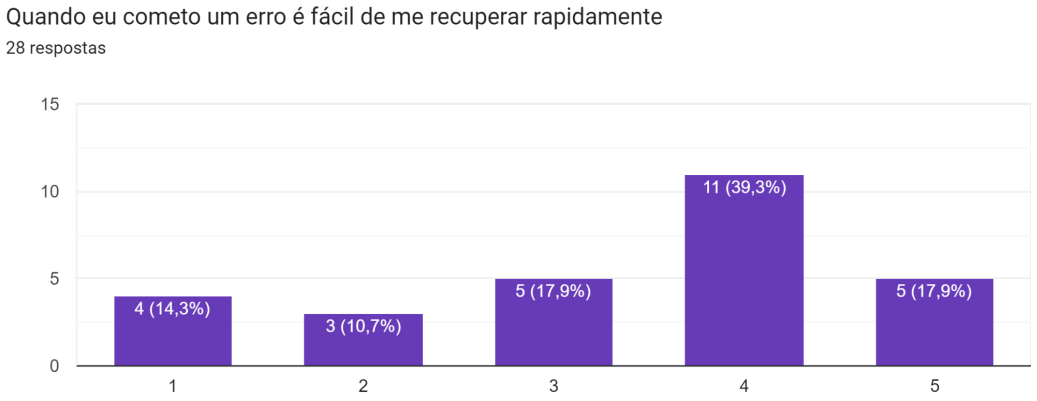
alunos. Essa facilidade de jogar se espelha também em outras perguntas que tiveram respostas com uma distribuição bastante similar, como “Considero o jogo fácil de jogar” e “As regras do jogo são claras e compreensíveis” que tiveram, respectivamente, 89,3% e 85,7% de respostas assinalando 4 e 5 na escala Likert. Essa facilidade de entender o jogo mudou significativamente em relação aos *feedbacks* da etapa de pré-testes, a partir de qual foram feitas alterações nas explicações do tutorial, nas dicas iniciais de cada fase, e na introdução de novas mecânicas em um ritmo constante.

Figura 39: Respostas à pergunta relacionada a proteção contra erros



Fonte: Elaborado pelo autor (2022)

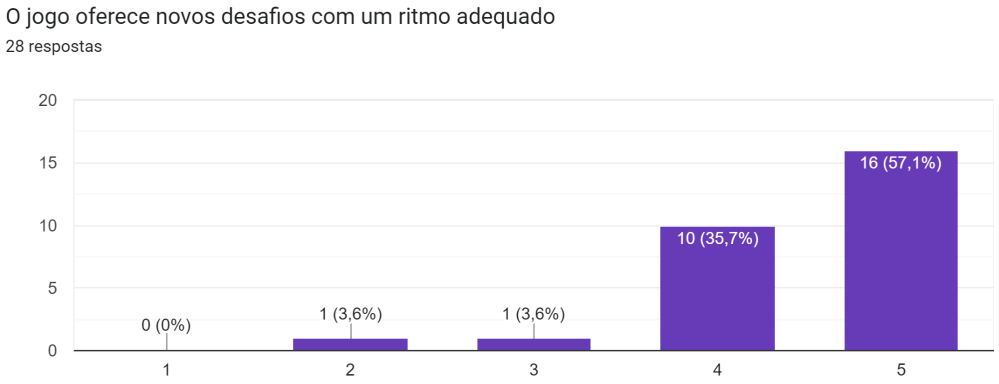
Figura 40: Respostas à pergunta relacionada a recuperação de erros



Fonte: Elaborado pelo autor (2022)

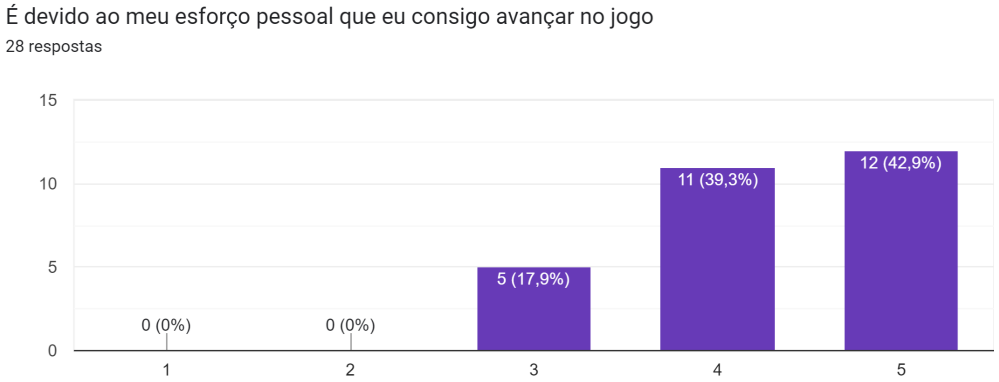
Quanto às perguntas sobre proteção contra erros, a distribuição de respostas foi bastante variada, conforme pode ser observado nas Figuras 39 e 40. Um *feedback* que apareceu diversas vezes na última pergunta do questionário, que foi deixada em aberto para comentários dos jogadores, foi sobre permitir ao jogador desfazer ou consertar algo que havia sido feito na etapa de programação dos personagens antes da fase terminar de rodar as instruções programadas. O segundo conjunto de perguntas abordou o Desafio e Satisfação proporcionados pelo jogo.

Figura 41: Respostas à pergunta relacionada ao ritmo dos desafios apresentados pelo jogo



Fonte: Elaborado pelo autor (2022)

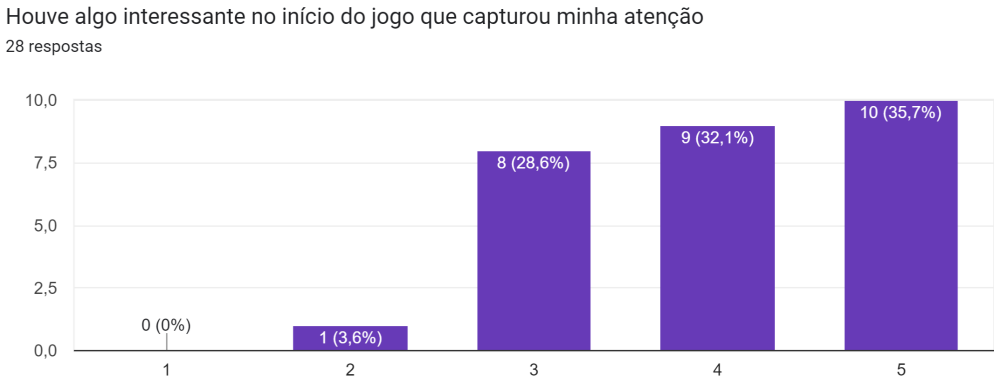
Figura 42: Respostas à pergunta relacionada ao esforço pessoal do jogador



Fonte: Elaborado pelo autor (2022)

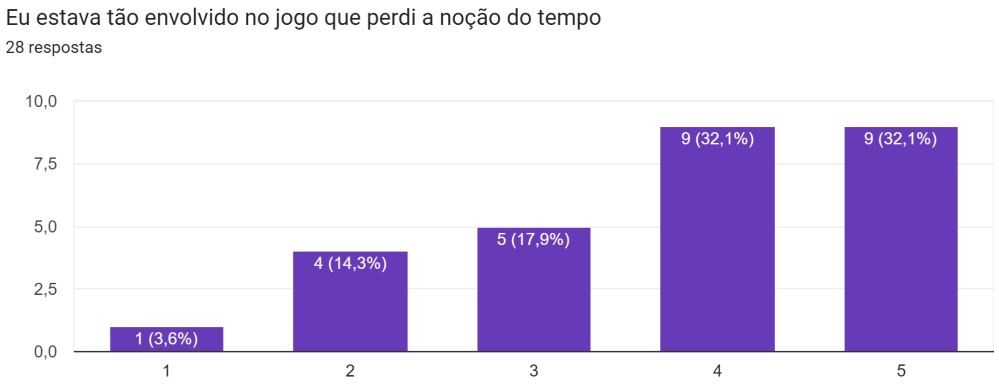
As respostas recebidas nas perguntas sobre desafio e satisfação foram bastante positivas, conforme Figuras 41 e 42, e também se devem ao *feedback* recebido no questionário de pré-teste. Introduzir mecânicas e possíveis instruções para os personagens conforme o jogador entende as mecânicas já apresentadas em fases anteriores facilitou o entendimento do jogo, mantendo o desafio ao jogador. O terceiro conjunto de perguntas do questionário aborda diversão e atenção focada, buscando entender se o jogador se divertiu com o jogo, o que é importante para manter o jogador interessado no jogo educativo, e se o jogador se manteve focado no jogo durante o período de teste. As Figuras 43 e 44 apresentam os gráficos de respostas relacionadas à Atenção Focada durante o período de testes.

Figura 43: Respostas à pergunta relacionada à atenção do jogador



Fonte: Elaborado pelo autor (2022)

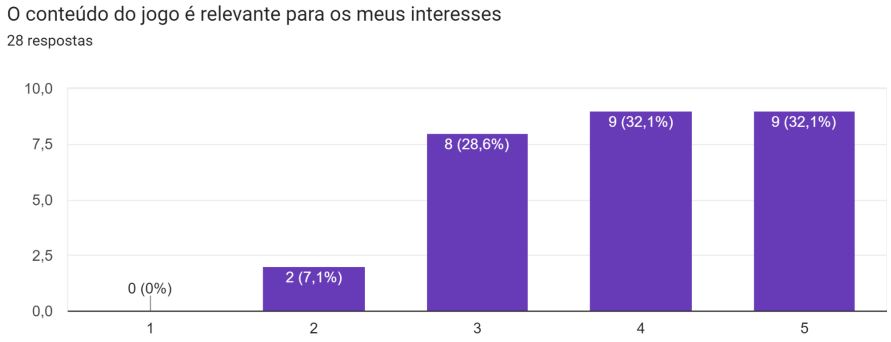
Figura 44: Respostas à pergunta relacionada ao foco do jogador



Fonte: Elaborado pelo autor (2022)

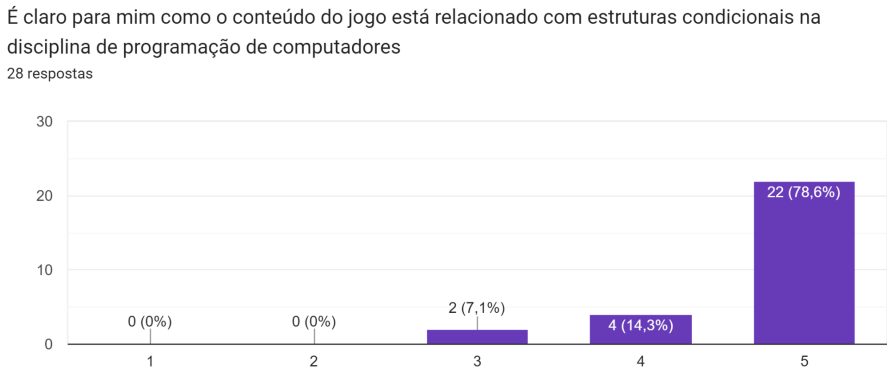
O quinto e último conjunto de perguntas foi relacionado à relevância e aprendizagem, e explora questões importantes para a validação do jogo como ferramenta de ensino. Entre elas, pergunta-se se o jogo é relevante aos interesses do jogador, se é claro como o jogo está relacionado à disciplina e se o jogo contribui com a aprendizagem. As respostas a essas perguntas podem ser vistas nas Figuras 45, 46 e 47, respectivamente. Pode-se observar pelos gráficos que os jogadores conseguiram entender a proposta do jogo e como ela se relaciona com o conteúdo estudado na disciplina.

Figura 45: Respostas à pergunta relacionada à relevância do jogo para os interesses do jogador



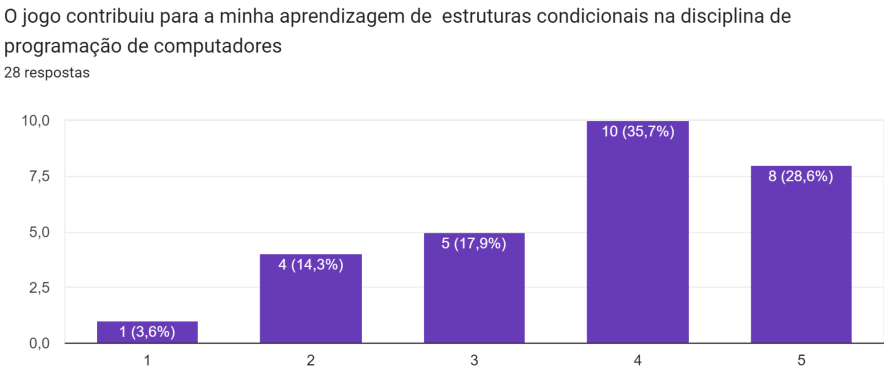
Fonte: Elaborado pelo autor (2022)

Figura 46: Respostas à pergunta relacionada a quão clara é a relação entre o conteúdo do jogo e o conteúdo da disciplina



Fonte: Elaborado pelo autor (2022)

Figura 47: Respostas à pergunta relacionada à contribuição do jogo com o aprendizado do jogador



Fonte: Elaborado pelo autor (2022)

A validação, aplicada por meio de questionários em turmas de Programação de Computadores I, proporcionou um bom feedback sobre o jogo como ferramenta de ensino, além de apresentar novas ideias de como o jogo poderia ser aplicado. O questionário finalizou com uma questão aberta sobre o *feedback* geral sobre a experiência com o jogo, por meio da qual pôde-se perceber uma boa aceitação por parte dos alunos, além de apresentar sugestões sobre novas mecânicas a serem exploradas. Como sugestões, destacam-se: criar um ranqueamento de acordo com um *score* gerado ao fim de cada fase, adicionar outros comandos relacionados à condicionais e possibilidade de alterar a velocidade das animações durante o combate.

6. CONSIDERAÇÕES FINAIS

A proposta inicial do trabalho, de planejar e desenvolver um jogo educativo com base nos métodos SGDA, para planejamento e desenvolvimento, e MEEGA+, para avaliação e validação, demonstrou-se possível. O jogo desenvolvido tem como principal objetivo o ensino de programação para estudantes de disciplinas iniciais no ensino superior, servindo como ferramenta de ensino, mas sem a intenção de substituir a aula, de forma com que o aluno possa reaprender ou fixar o aprendizado do conteúdo de estruturas de repetição em disciplinas de programação.

Ainda, de acordo com as aplicações de testes desenvolvidos com base no formulário do MEEGA+, o jogo demonstrou-se viável como ferramenta de ensino. Na aplicação dos questionários, com duas turmas de Programação de Computadores I, 28 alunos responderam, em grande parte, com um *feedback* bastante positivo sobre o jogo desenvolvido. O jogo foi baseado no conteúdo de estruturas condicionais, aplicada nas disciplinas de programação, porém pode também ser aplicado para currículos de pensamento computacional, conforme os apresentados no capítulo 2.1. O conteúdo de condicionais encaixa nas disciplinas propostas pelo CIEB para currículos de 4º e 5º ano do ensino fundamental, desenvolvendo as dimensões curriculares de Algoritmos e de Reconhecimento de Padrões. Conforme as propostas de currículo do CIEB, essas dimensões propõe ao aluno executar algoritmos simples e identificar semelhanças e diferenças em situações que se repetem, para turmas de quarto ano; e conhecer e utilizar algoritmos com repetição e reconhecer padrões em um algoritmo, para turmas de quinto ano.

Além disso, nos *feedbacks* recebidos a partir do questionário, surgiram várias sugestões de melhorias para o jogo, que podem ser levadas a trabalhos futuros utilizando-o. Para possíveis trabalhos futuros, poderia-se desenvolver mais disciplinas e mais conhecimentos relacionados a programação e pensamento computacional utilizando as mesmas mecânicas já existentes no jogo desenvolvido. Ainda levando em conta as sugestões geradas pelo questionário, o jogo pode evoluir com novas fases, novas mecânicas, ou um sistema de ranqueamento de jogadores. Além disso, novas aplicações do jogo tanto como ferramenta de ensino quanto como objeto de estudo. Como projeto já em andamento, por exemplo, há um teste de ferramentas de análise de garantia de qualidade desenvolvidas pelo UCSLabQA, que usará o jogo Code_Dungeon como um dos objetos de estudo para essa aplicação.

REFERÊNCIAS BIBLIOGRÁFICAS

ABT, C. **Serious Games.** 1. ed. New York: Viking Press, 1970.

ARIMOTO, M; OLIVEIRA, W. Dificuldades no Processo de Aprendizagem de Programação de Computadores: um Survey com Estudantes de Cursos da Área de Computação. **Anais do XXVII Workshop sobre Educação em Computação**. Porto Alegre: Sociedade Brasileira de Computação, 2019. p. 244-254.

BELLOTTI, F.; BERTA, R.; DE GLORIA, A.; Designing Effective Serious Games: Opportunities and Challenges for Research, **International Journal of Emerging Technologies in Learning (iJET)**. vol. 5, 2010. p. 22-35

CENTRO DE INOVAÇÃO PARA A EDUCAÇÃO BRASILEIRA. **Currículo de Referência em Tecnologia e Computação**. CIEB. 2018. Disponível em: <https://curriculo.cieb.net.br/> Acesso em: 30 abr. 2022.

Procura por profissionais de tecnologia cresce 671% durante a pandemia. CNN. 27 out. 2021. Disponível em: <https://www.cnnbrasil.com.br/business/procura-por-profissionais-de-tecnologia-cresce-671-durante-a-pandemia>.

CodeCombat Teacher Getting Started Guide. CodeCombat, 2021. Disponível em: <https://codecombat.zendesk.com/hc/en-us/articles/1500009108962> Acesso em: 15 mai. 2022.

FABRO, F. **Final Fantasy XII's Gambit System was like programming for beginners**. DEV. 15 out. 2019. Disponível em: <https://dev.to/fihra/final-fantasy-xii-s-gambit-system-was-like-programming-for-beginners-7d1>

INSTITUTO SEMESP. *In*: Mapa do Ensino Superior. **Cursos mais procurados.** São Paulo, SP: Instituto SEMESP, 2021. Disponível em: <https://www.semesp.org.br/mapa/edicao-11/brasil/cursos-mais-procurados/> Acesso em: 12 jun. 2022.

INSTITUTO SEMESP. *In*: Mapa do Ensino Superior. **Evasão.** São Paulo, SP: Instituto SEMESP, 2021. Disponível em: <https://www.semesp.org.br/mapa/edicao-11/brasil/evasao/> Acesso em: 12 jun. 2022.

LIMA, H. P. **Desenvolvimento de um Serious Game para a aprendizagem do pensamento computacional**. Universidade de Caxias do Sul. Caxias do Sul. 2017.

MITGUTSCH, K.; ALVARADO, N. Purposeful by design?: a serious game design assessment framework. **FDG '12: Proceedings of the International Conference on the Foundations of Digital Games**. New York: Association for Computing Machinery, 2012. p. 121-128.

MOREIRA, G. L.; HOLANDA, W.; COUTINHO, J. C.; CHAGAS, F. S. Desafios na aprendizagem de programação introdutória em cursos de TI da UFERSA, campus Pau dos Ferros: um estudo exploratório. **Anais do Encontro de Computação do Oeste Potiguar ECOP/UFERSA**. [S.l.], n. 2, 2018. p 90-96.

PETROPOULEAS, S. **Nova leva de cursos tenta suprir aumento da demanda por profissionais de TI**. Folha de S.Paulo. 1 dez. 2021. Disponível em: <https://www1.folha.uol.com.br/mercado/2021/12/demanda-por-profissionais-de-ti-supera-projecoes-e-impulsiona-iniciativas-para-suprir-deficit.shtml>. Acesso em: 7 jun. 2022.

ROBINSON, M. **Final Fantasy 12's Gambits remain the greatest mechanic in JRPGs**. Eurogamer. 31 mai. 2017. Disponível em: <https://www.eurogamer.net/final-fantasy-12s-gambits-remain-one-of-jrpgs-best-ideas-in-years>

SEVERGNINI, L. F. **Serious Game como ferramenta de ensino de lógica de programação para crianças**. Universidade de Caxias do Sul. Caxias do Sul. 2016.

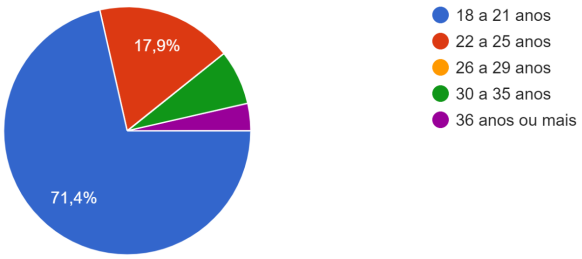
SOUZA, D.M.; BATISTA, M. H.; BARBOSA, E. F. Problemas e Dificuldades no Ensino e na aprendizagem de Programação: Um Mapeamento Sistemático. **Revista Brasileira de Informática na Educação**. [S.l.], v. 24, n. 01, 2016. p 40-52.

ZENATO, M. **Ambiente de aprendizagem de programação de computadores**. Universidade de Caxias do Sul. Caxias do Sul. 2009.

APÊNDICE A - RESPOSTAS DO QUESTIONÁRIO DE VALIDAÇÃO DO JOGO, APLICADO EM TURMAS DE PROGRAMAÇÃO I

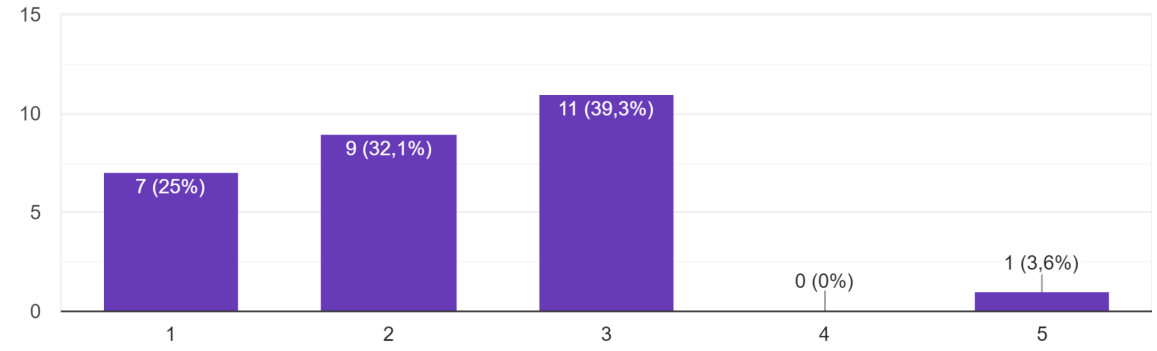
Qual sua faixa de idade?

28 respostas



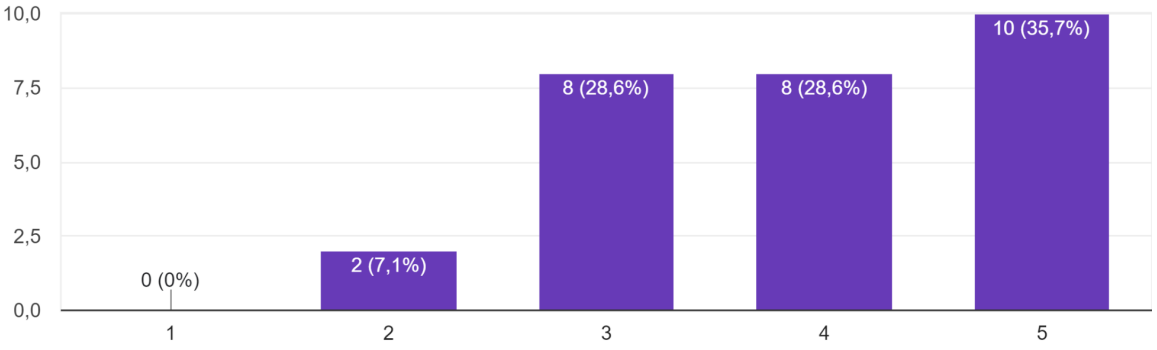
Qual seu nível de experiência com programação?

28 respostas



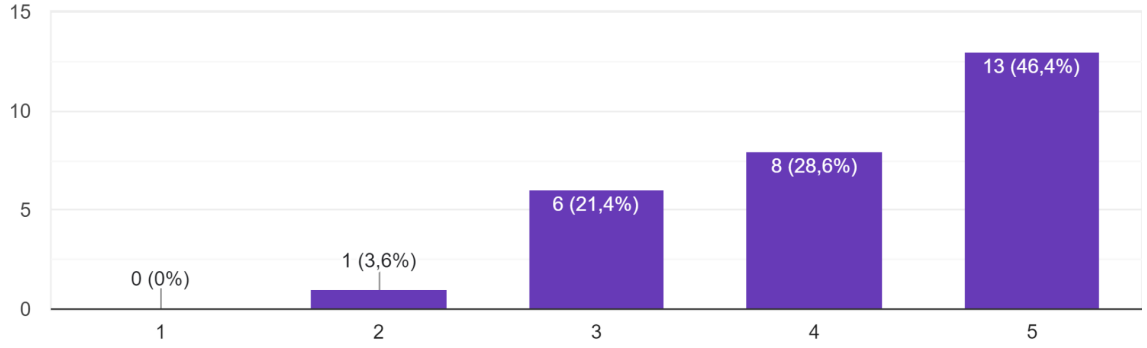
O design do jogo é atraente

28 respostas



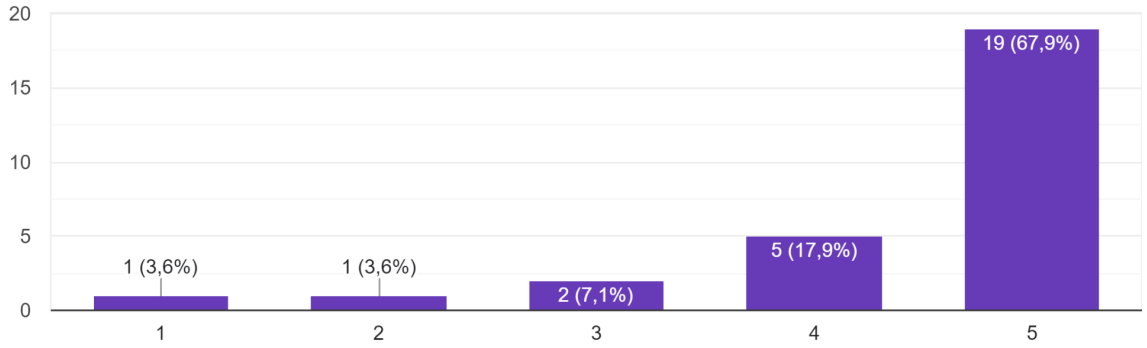
Os textos, cores e fontes combinam e são consistentes

28 respostas



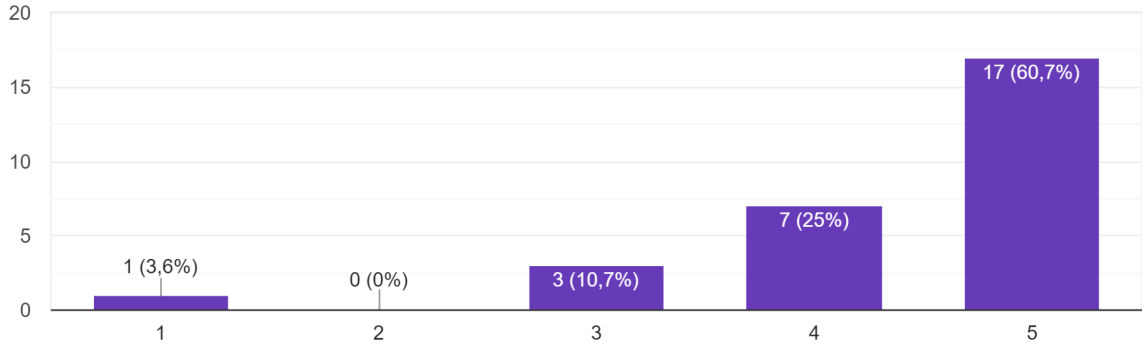
Eu precisei aprender poucas coisas para poder começar a jogar o jogo

28 respostas



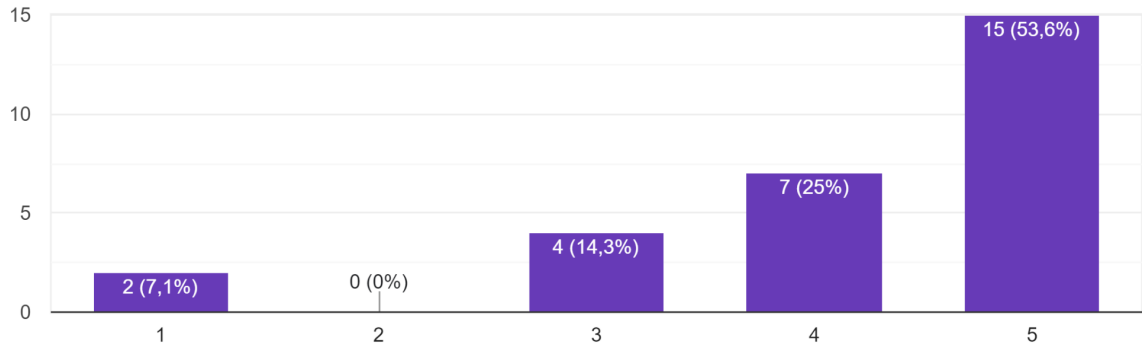
Aprender a jogar este jogo foi fácil para mim

28 respostas



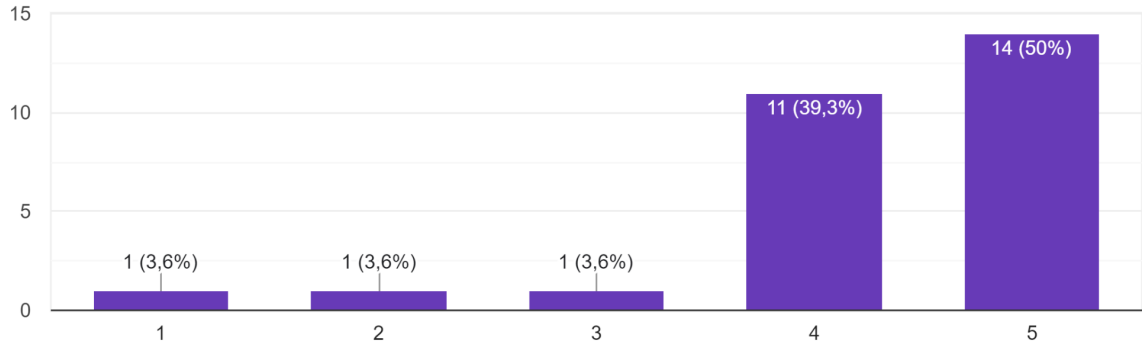
Eu acho que a maioria das pessoas aprenderiam a jogar este jogo rapidamente

28 respostas



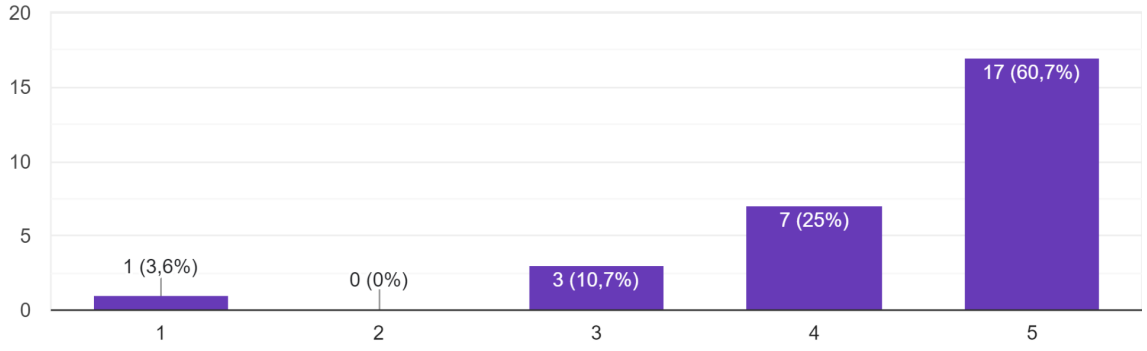
Eu considero que o jogo é fácil de jogar

28 respostas



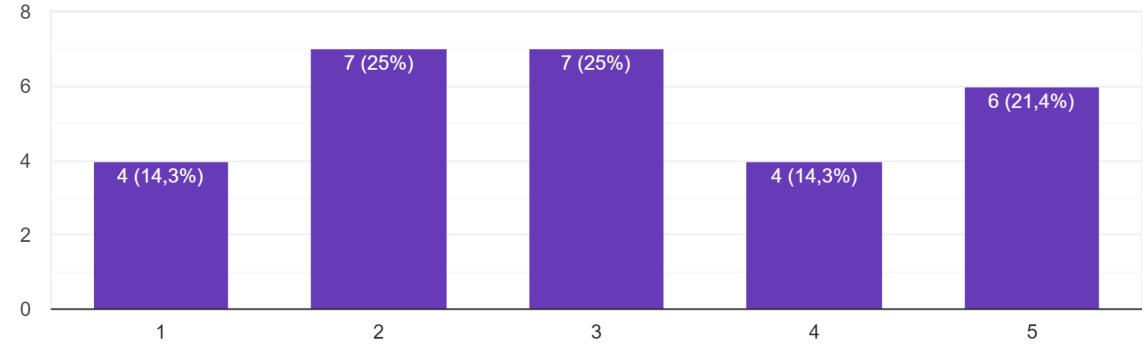
As regras do jogo são claras e compreensíveis

28 respostas



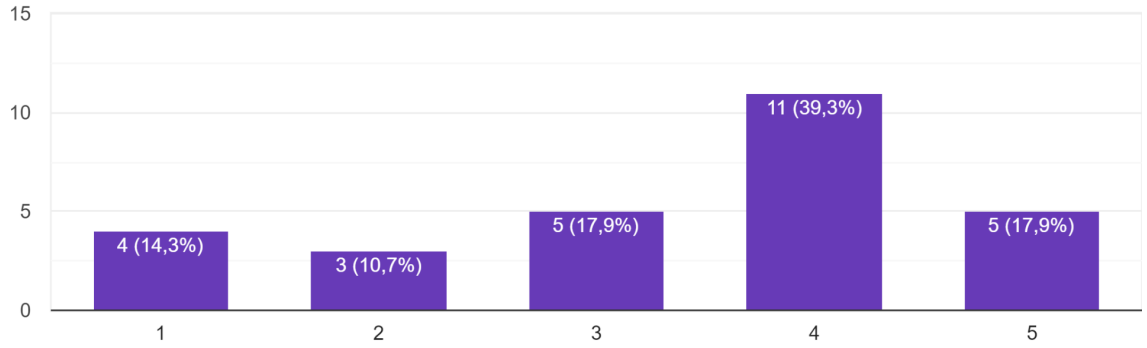
O jogo me protege de cometer erros

28 respostas



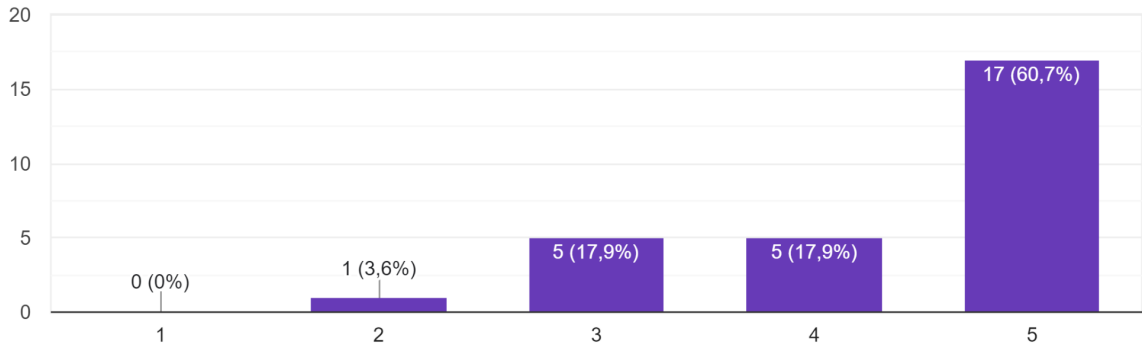
Quando eu cometo um erro é fácil de me recuperar rapidamente

28 respostas



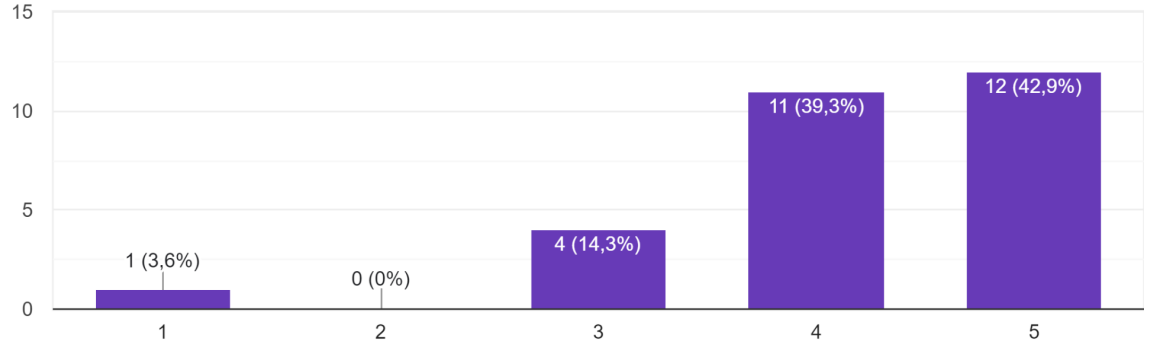
Quando olhei pela primeira vez o jogo, eu tive a impressão de que seria fácil para mim

28 respostas



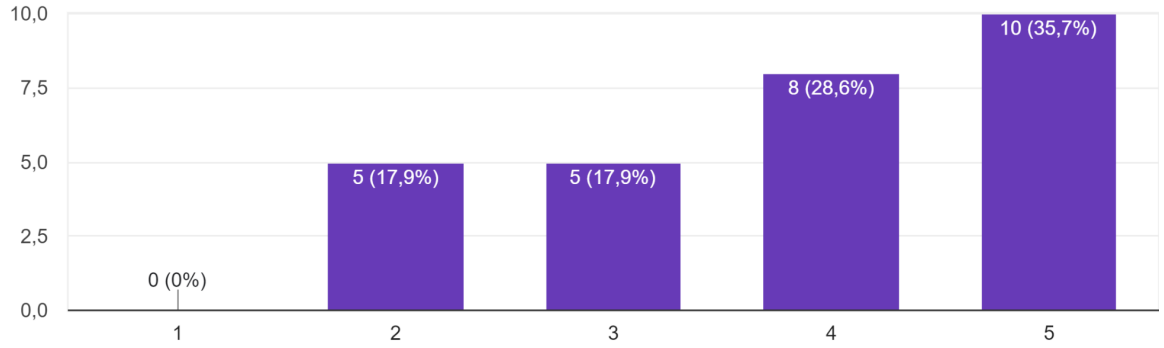
A organização do conteúdo me ajudou a estar confiante de que eu iria aprender com este jogo

28 respostas



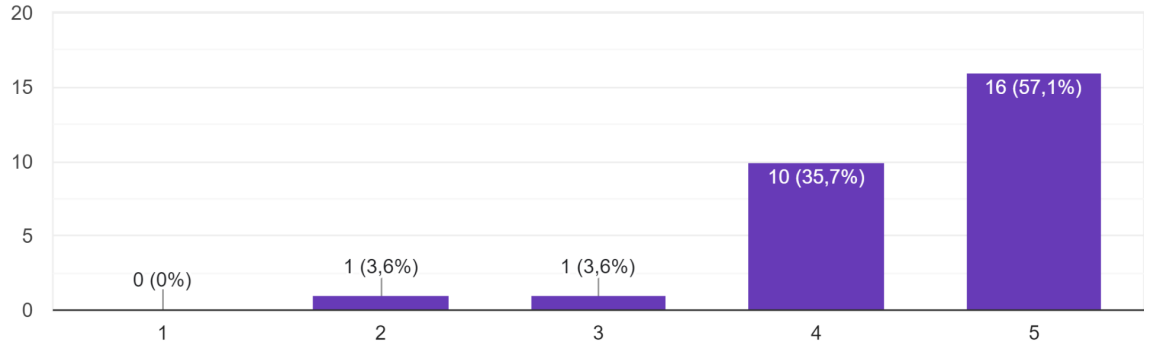
Este jogo é adequadamente desafiador para mim

28 respostas



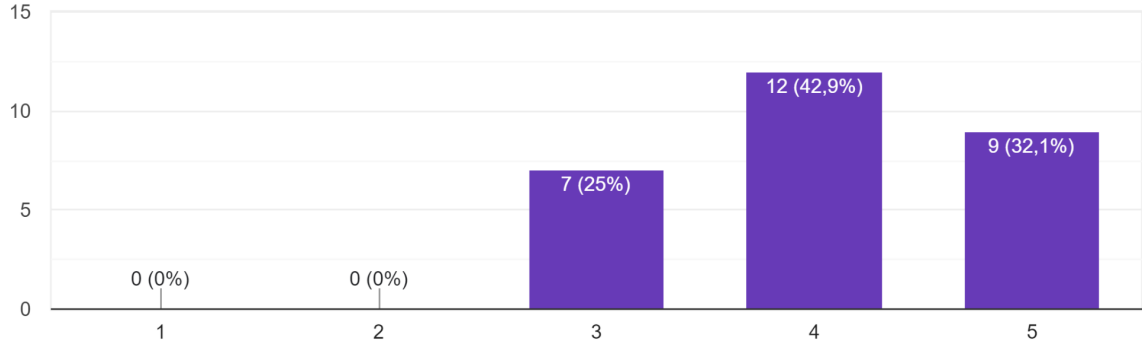
O jogo oferece novos desafios com um ritmo adequado

28 respostas



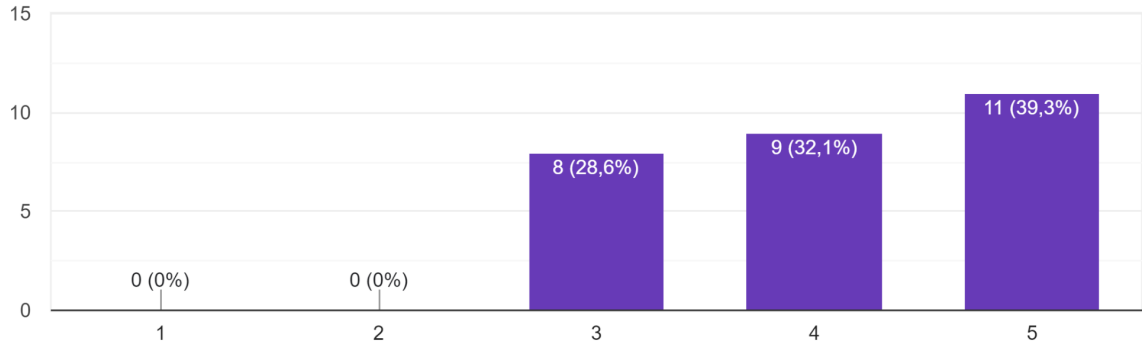
O jogo não se torna monótono nas suas tarefas (repetitivo ou com tarefas chatas)

28 respostas



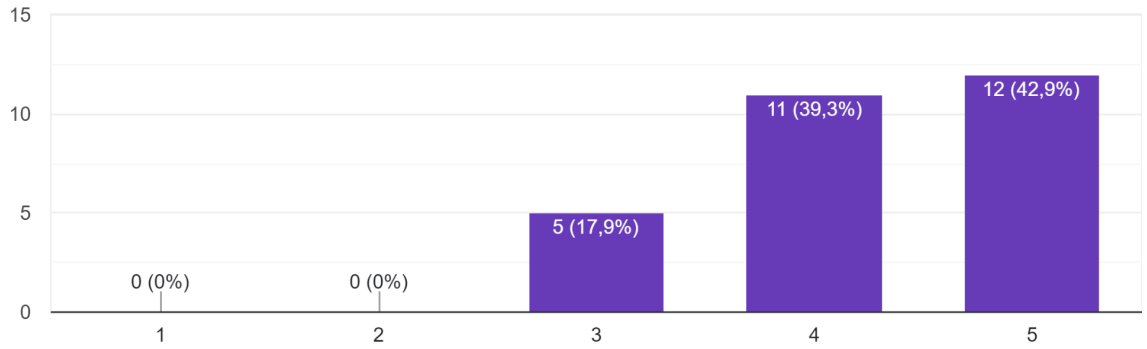
Completar as tarefas do jogo me deu um sentimento de realização

28 respostas



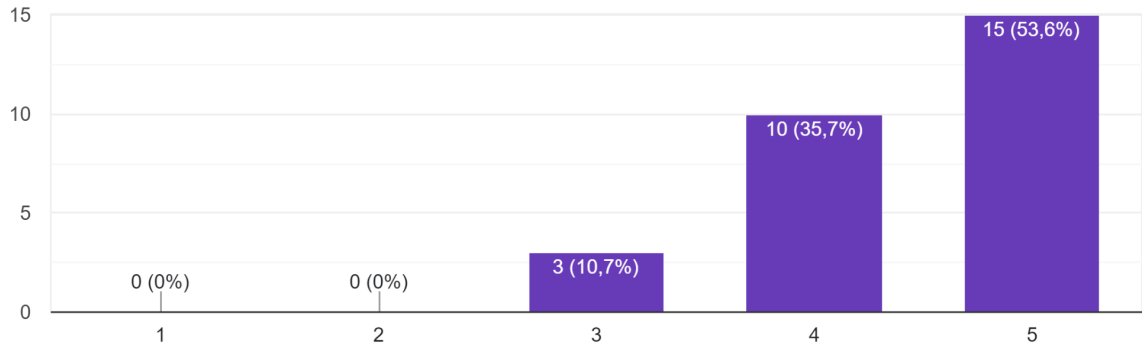
É devido ao meu esforço pessoal que eu consigo avançar no jogo

28 respostas



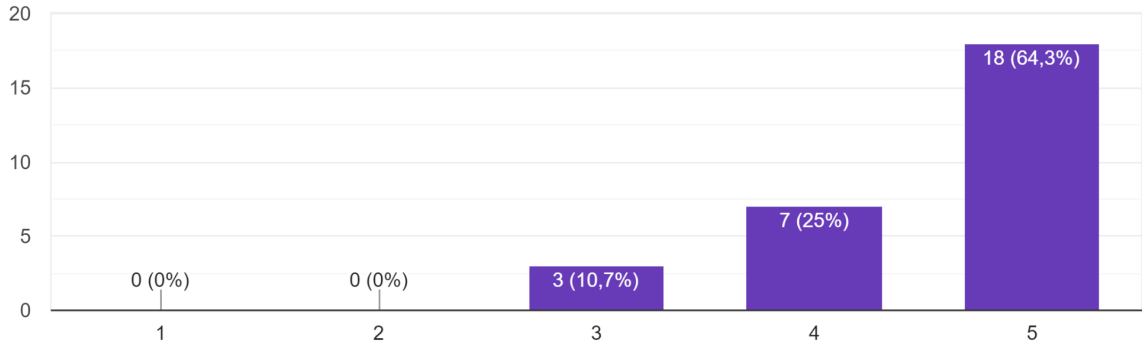
Eu recomendaria esse jogo para meus colegas

28 respostas



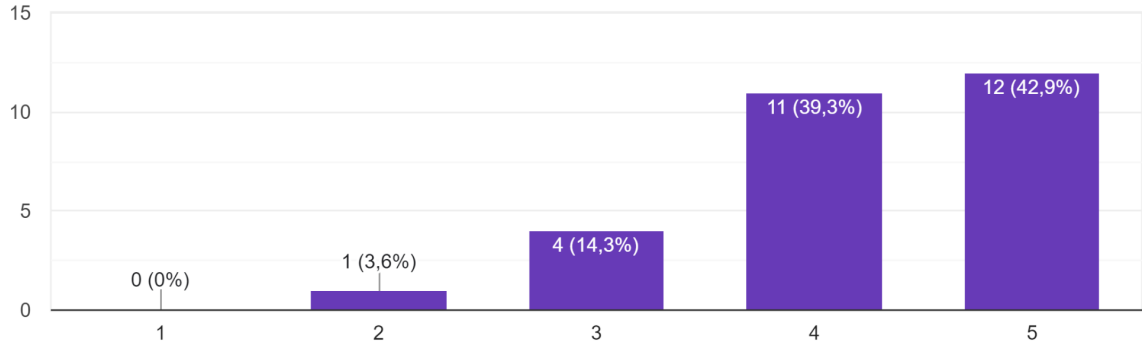
Eu me diverti com o jogo

28 respostas



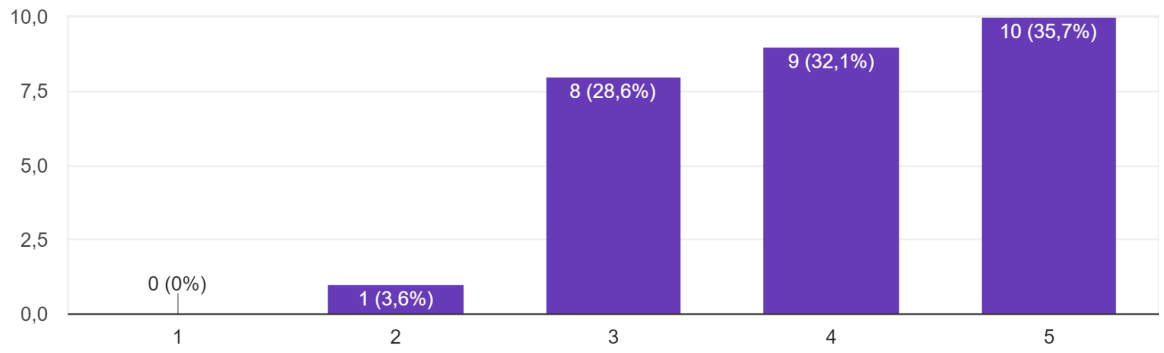
Aconteceu alguma situação durante o jogo que me fez sorrir

28 respostas



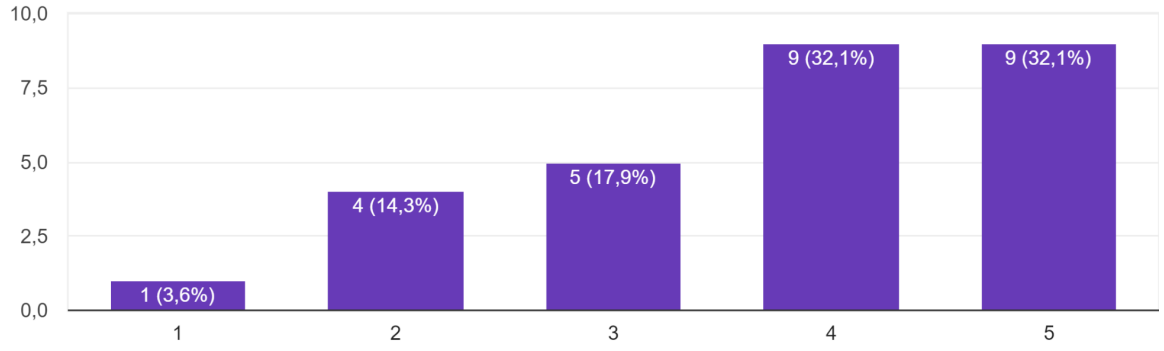
Houve algo interessante no início do jogo que capturou minha atenção

28 respostas



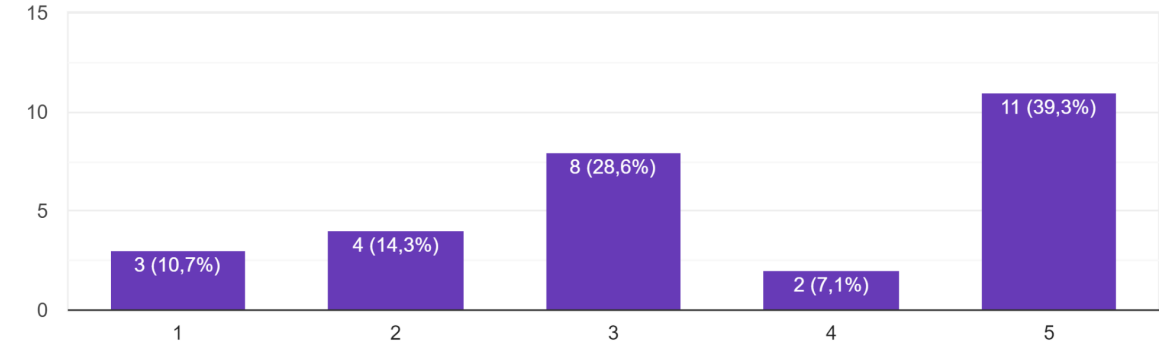
Eu estava tão envolvido no jogo que perdi a noção do tempo

28 respostas



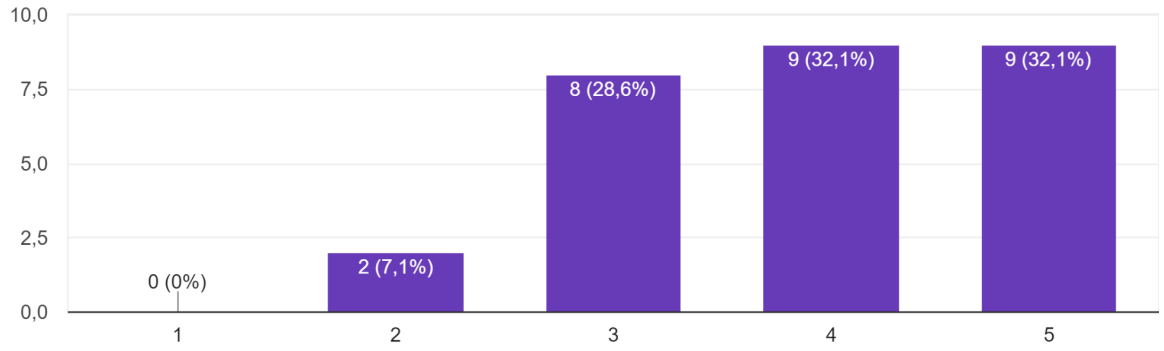
Eu esqueci sobre o ambiente ao meu redor enquanto jogava este jogo

28 respostas



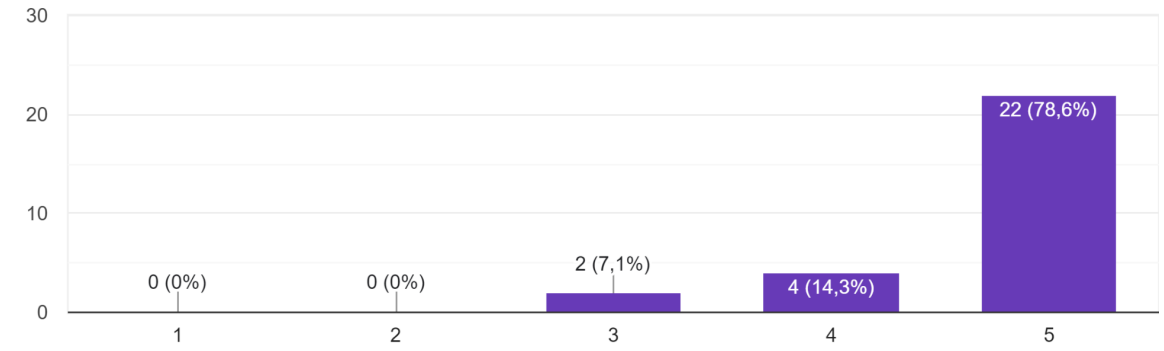
O conteúdo do jogo é relevante para os meus interesses

28 respostas



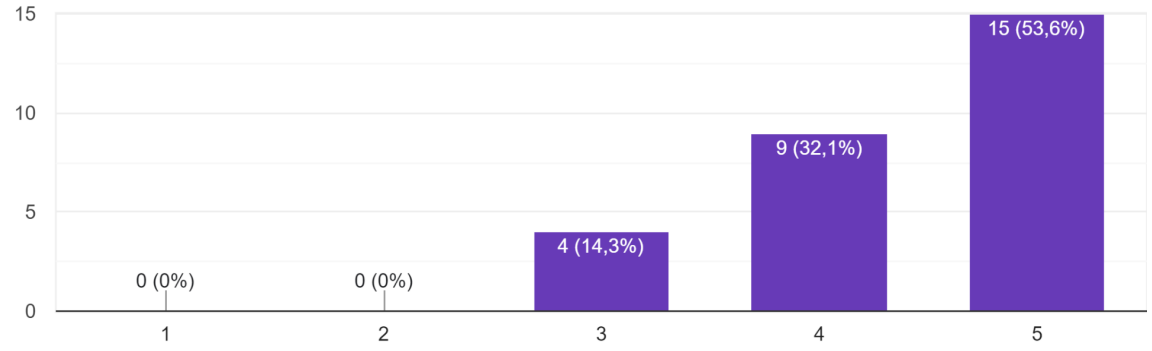
É claro para mim como o conteúdo do jogo está relacionado com estruturas condicionais na disciplina de programação de computadores

28 respostas



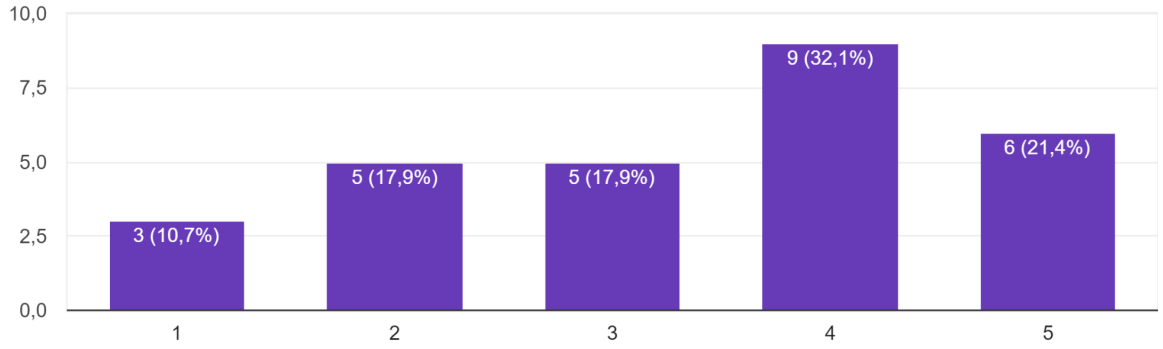
O jogo é um método de ensino adequado para estruturas condicionais na disciplina de programação de computadores

28 respostas



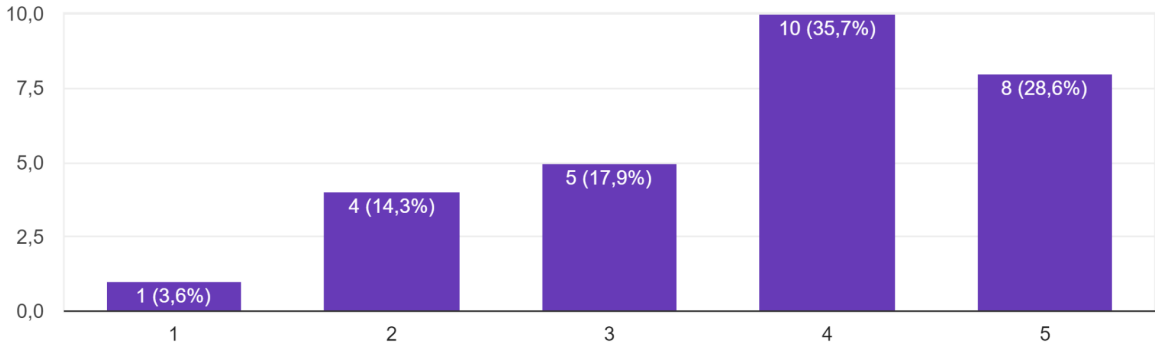
Eu prefiro aprender com este jogo do que de outra forma (outro método de ensino)

28 respostas



O jogo contribuiu para a minha aprendizagem de estruturas condicionais na disciplina de programação de computadores

28 respostas



O jogo foi eficiente para minha aprendizagem, em comparação com outras atividades relacionadas a estruturas condicionais na disciplina de programação de computadores

28 respostas