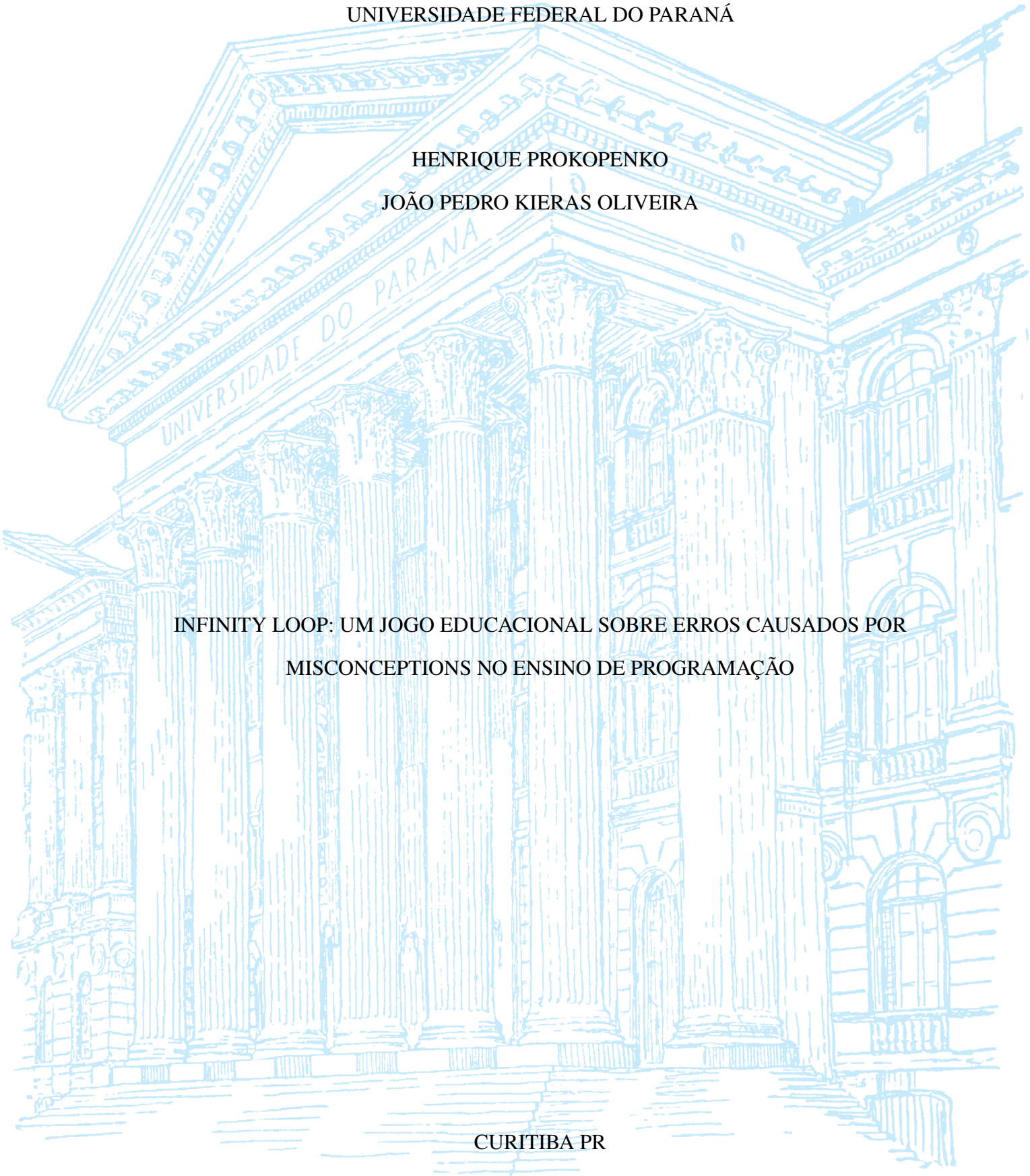
UNIVERSIDADE FEDERAL DO PARANÁ

HENRIQUE PROKOPENKO

JOÃO PEDRO KIERAS OLIVEIRA



INFINITY LOOP: UM JOGO EDUCACIONAL SOBRE ERROS CAUSADOS POR MISCONCEPTIONS NO ENSINO DE PROGRAMAÇÃO

CURITIBA PR

2024

HENRIQUE PROKOPENKO
JOÃO PEDRO KIERAS OLIVEIRA

INFINITY LOOP: UM JOGO EDUCACIONAL SOBRE ERROS CAUSADOS POR MISCONCEPTIONS NO ENSINO DE PROGRAMAÇÃO

Trabalho apresentado como requisito parcial à conclusão do Curso de Bacharelado em Ciência da computação, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Computação*.

Orientador: Rachel Carlos Duque Reis.

CURITIBA PR

2024

Por Henrique Prokopenko: *À Consuelo, Sandro, Amanda, Ricardo, Rachel, Nayara, Natalio e Silvia .*
Por João Pedro Kieras Oliveira: *Aos meus pais, pelo apoio durante a graduação.*

AGRADECIMENTOS

Por Henrique Prokopenko: Primeiramente a Deus e Nossa Senhora Aparecida, pela vida e pelos privilégios. Aos meus pais, por todo o suporte e por me mostrarem o caminho (Essa conquista é de vocês). À minha irmã e meu cunhado, que são fonte de inspiração como acadêmicos. À minha namorada, que me apoiou e compartilha a vida comigo. Aos meus sogros, que me acolheram como família. E por fim, à minha orientadora, que teve muita paciência para nos ensinar. Um profundo e imenso obrigado!

Por João Pedro Kieras Oliveira: A nossa orientadora, Rachel Reis, por todo o apoio, paciência e dedicação durante o desenvolvimento deste trabalho. Ao meu colega e amigo, Henrique Prokopenko, por toda a parceria desde o começo do curso. Aos meus pais, Adriane Kieras Oliveira e Cassiano Drabeski Oliveira, por todo o incentivo e apoio ao longo dos anos de estudo. A minha companheira, Eduarda de Aguiar Freitas, por ser uma fonte de afeto, motivação e apoio ao longo da graduação e também na vida profissional. A Deus, pela minha vida e por tudo que tenho. E por fim, aos meus amigos que me incentivaram ao longo dessa jornada.

RESUMO

Trabalhos na literatura relatam os desafios enfrentados pelos estudantes durante o processo de aprendizagem das disciplinas de introdução a programação. Um desses desafios refere-se à dificuldade dos alunos no desenvolvimento da lógica de programação. No intuito de mitigar essas dificuldades, diversas estratégias e ferramentas de ensino têm sido adotadas para apoiar o ensino de programação. Dentre elas, destaca-se os jogos digitais educacionais, *softwares* projetados para apresentar desafios que incorporam abordagens pedagógicas e que buscam motivar o aprendiz, sem comprometer a diversão e o entretenimento. No entanto, a maioria desses jogos se concentra em trabalhar conceitos básicos de programação como, manipulação de variáveis, estruturas de repetição e decisão, desconsiderando os erros cometidos pelos alunos causados por *misconceptions*, ou seja, por falta de compreensão adequada dos conceitos de programação. Logo, este trabalho tem como objetivo apresentar o desenvolvimento de um jogo educacional para apoiar o processo de aprendizagem de estudantes de programação, com foco em erros causados por *misconceptions* em laços de repetição isolados e combinados a outras estruturas de dados como vetores e matrizes. O jogo foi desenvolvido utilizando o motor gráfico Godot 4, com cenários representando uma vila medieval. Além disso, o jogo é composto por cinco fases que exploram quatro erros causados por *misconception*. A primeira fase tem como foco o erro "acesso a posição inválida do vetor", a segunda fase "laço infinito", a terceira fase "uso incorreto de operadores relacionais", a quarta fase "uso incorreto de operadores lógicos" e, por fim, a quinta fase possui uma mistura de todos os erros anteriores, exceto "uso incorreto de operadores lógicos". O jogo oferece um ambiente lúdico e dinâmico, onde a interação com o aluno tem influência direta no mapa do jogo, ou seja, quando o estudante seleciona uma opção correta ou incorreta para os desafios, a execução desse erro é incorporada ao cenário do jogo. Após completar todos os desafios, o jogo disponibiliza um relatório com os erros cometidos pelo jogador e sugere os conceitos de programação que o estudante deve revisar.

Palavras-chave: *Misconception*. Jogo Educacional. Programação. Estrutura de Repetição

ABSTRACT

Studies in the literature report on the challenges faced by students during the learning process of introductory programming courses. One of these challenges is the difficulty students have in developing programming logic. In order to mitigate these difficulties, various teaching strategies and tools have been adopted to support the teaching of programming. These include educational digital games, software designed to present challenges that incorporate pedagogical approaches and seek to motivate the learner, without compromising on fun and entertainment. However, most of these games focus on working on basic programming concepts such as manipulating variables, repetition and decision structures, disregarding the errors made by students caused by misconceptions, i.e. a lack of proper understanding of programming concepts. Therefore, the aim of this work is to present the development of an educational game to support the teaching of programming to beginner students on undergraduate Computing courses, with a focus on errors caused by misconceptions in isolated repetition loops and combined with other data structures like vectors and matrices. The game was developed using the Godot 4 graphics engine, with scenarios representing a medieval village. In addition, the game consists of five stages that explore four errors caused by misconception. The first stage focuses on the error "access to invalid vector position", the second stage "infinite loop", the third stage "incorrect use of relational operators", the fourth stage "incorrect use of logical operators" and, finally, the fifth stage has a mixture of all of the above errors, except "incorrect use of logical operators". The game offers a playful and dynamic environment, where interaction with the student has a direct influence on the game map, i.e. when the student selects a correct or incorrect option for the challenges, the execution of this error is incorporated into the game scenario. After completing all the challenges, the game provides a report with the errors made by the player and suggests the programming concepts that the student should review.

Keywords: Misconception. Educational Game. Programming. Repetition Structure

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Neste capítulo serão apresentadas as principais motivações e objetivos deste trabalho. Além disso, será mostrado o contexto geral da pesquisa e suas principais contribuições.

1.1 CONTEXTO E MOTIVAÇÃO

Nos cursos universitários da área de Computação, o aprendizado de programação pode trazer muitos benefícios aos alunos, como estimular o pensamento crítico e a capacidade de resolução de problemas (Monclar et al., 2018). No entanto, diversos trabalhos na literatura relatam desafios enfrentados pelos alunos durante o processo de aprendizagem das disciplinas de programação (Morais et al., 2020; Bosse e Gerosa, 2015). Entre os principais desafios, destaca-se a dificuldade no desenvolvimento da lógica de programação, compreensão da sintaxe, falta de tempo para se dedicar à disciplina e dificuldade na interpretação das questões (Moreira et al., 2018).

Como consequência desses desafios, observa-se que a taxa de evasão e reprovação de estudantes nos cursos de Computação é muito alta (Medeiros et al., 2020). Um exemplo de indicador da alta taxa de evasão é o desempenho acadêmico no primeiro período letivo. É nesse período que os estudantes se deparam com as disciplinas de introdução à programação, as quais exercem impacto significativo no destino acadêmico do estudante, ou seja, se irão prosseguir ou desistir do curso (Carvalho et al., 2019). Para tentar mitigar as dificuldades dos alunos, diversas estratégias e ferramentas de ensino têm sido adotadas no intuito de apoiar o ensino de programação (Gomes et al., 2008). Como exemplos de estratégias, têm-se a gamificação e robótica (Duarte de Holanda et al., 2019). Como exemplos de ferramentas, destacam-se os ambientes de aprendizagem *online*, ferramentas lúdicas como o Scratch[1] e os jogos digitais educacionais (Duarte de Holanda et al., 2019).

Um jogo digital educacional é um tipo de *software* projetado para apresentar desafios que incorporam uma abordagem pedagógica, buscando estimular e motivar o aprendiz, sem comprometer a diversão e o entretenimento (Falkembach, 2006). Em geral, os jogos educacionais, para apoiar o ensino de programação, se concentram em conteúdos específicos. Por exemplo, o jogo Projeto Éden (Oliveira e Farias, 2019) explora a manipulação de variáveis e tipos de dados, já o jogo Klouro (de Azevêdo Silva e Dantas, 2014) tem como objetivo o uso dos operadores lógicos e aritméticos. Os jogos Catacombs (Ralph e Barnes, 2007) e MazeLogic (Oliveira et al., 2018), por sua vez, abordam outros fundamentos como lógica de programação, estruturas de repetição e decisão.

Apesar dos avanços no desenvolvimento de jogos educacionais para apoiar o ensino de programação, foi observado que, em geral, a maioria deles se concentra em ensinar conceitos básicos sem levar em consideração os erros conceituais cometidos pelos alunos. Em geral, esses erros são causados por *misconceptions* (Araujo et al., 2021), ou seja, pela falta de compreensão adequada de um conceito. Nesse sentido, este trabalho tem como proposta desenvolver um jogo educacional que utiliza erros desse tipo como uma oportunidade de aprendizagem para alunos iniciantes em programação.

[1] O Scratch é um exemplo de ferramenta de programação visual, baseada em blocos de código pré-moldados que o aluno manipula para escrever um programa de computador (Blatt et al., 2017).

1.2 OBJETIVO

Este trabalho tem como objetivo desenvolver um jogo educacional para estudantes de programação. O jogo tem como proposta apoiar o ensino de programação, se concentrando nos principais erros gerados por *misconceptions* no contexto das estruturas de repetição (*for* e *while*) utilizadas de forma isolada e combinadas com estruturas de dados como vetor e matriz. Um *misconception* na programação consiste em uma compreensão equivocada ou imprecisa de um conceito que pode levar o aluno a cometer erros na codificação (Araujo et al., 2021).

1.3 DESAFIOS

Durante o desenvolvimento do jogo educacional foram encontrados diversos desafios que exigiram criatividade, persistência e habilidade técnica. A seguir, são apresentados alguns desses desafios:

1.3.1 Desenvolver um jogo atrativo

Um dos maiores desafios foi garantir que o jogo educacional fosse atrativo para os jogadores. Integrar conteúdo educativo de maneira lúdica e interessante requer equilíbrio entre diversão e aprendizagem. Foram empregados mecanismos de recompensa e *feedback* imediato, com o intuito de manter o interesse dos jogadores em todos os desafios.

1.3.2 Trabalhar estruturas de repetição nos desafios

A implementação de desafios que trabalham com estruturas de repetição e exploram os erros causados por *misconceptions* foi crucial para garantir o diferencial do jogo. Foi feita uma pesquisa detalhada sobre os *misconceptions* mais recorrentes em estruturas de repetição e investigado como incorporá-los à mecânica do jogo.

1.3.3 Encontrar *tilesets* adequados

A seleção de *tilesets*[2] adequados para os desafios do jogo foi um processo minucioso. Foi feita uma busca por *tilesets* com permissão de uso e visualmente apelativos nos principais repositórios gratuitos, que pudessem ser usados nos desafios do jogo.

1.3.4 Criar o mapa do jogo

A criação do mapa do jogo envolveu não apenas o *design* visual, mas também a definição dos caminhos, áreas de interação e a integração de elementos do jogo para apoiar a narrativa e os desafios. Foram utilizados vários recursos disponíveis do Godot 4 para construir telas responsivas e um mapa interativo, além de recursos que pudessem ser reutilizados no jogo.

1.3.5 Inserir interatividade com os NPCs

Adicionar interatividade com NPCs[3] foi essencial para melhorar a experiência do jogador. Foram desenvolvidos diálogos dinâmicos e interações contextuais para enriquecer a narrativa e facilitar o aprendizado, ao mesmo tempo que mantém o jogador engajado. Essa tarefa

[2] Um *tileset* é um conjunto de texturas reunidas numa mesma imagem. Essas texturas são usadas para construir o cenários dos jogos 2D.

[3] NPC significa "*Non Playable Character*", personagem não jogável, em tradução livre.

envolveu a integração de bibliotecas de diálogo ao jogo, a busca por *spritesheets*[4] gratuitos de NPCs e o planejamento dos diálogos para introduzir o desafio ao jogador.

1.4 CONTRIBUIÇÃO

Este trabalho tem como principal contribuição o desenvolvimento de um jogo educacional que explora erros causados por *misconceptions*, com ênfase em laços de repetição. O jogo se encontra disponível no *link* https://hproko.itch.io/infinity-loops. Além disso no jogo, as soluções propostas para os desafios foram incorporadas à mecânica do jogo. Isso significa que, se o jogador seleciona uma opção que causa "laço infinito", por exemplo, esse erro é exibido no mapa do jogo por meio de animações dinâmicas. Dessa forma, o aluno consegue visualizar seus erros e entender como eles afetam a execução do programa, além de compreender as consequências práticas desses erros.

1.5 ORGANIZAÇÃO DO DOCUMENTO

Este documento é composto por seis capítulos, incluindo este capítulo de introdução que mostrou o contexto e a motivação para o desenvolvimento deste trabalho (Seção 1.1), o objetivo (Seção 1.2), os desafios (Seção 1.3) e a principal contribuição (Seção 1.4). O Capítulo 2 exibe uma base teórica formada pelos conceitos de *misconception* (Seção 2.2) e jogo educacional (Seção 2.1). O Capítulo 3 apresenta os trabalhos relacionados sobre jogos educacionais para apoiar o ensino de programação. O Capítulo 4 possui os materiais e métodos utilizados para o desenvolvimento do jogo. O Capítulo 5 possui os principais resultados e por fim, no Capítulo 6 tem-se as conclusões, seguidas das referências bibliográficas.

[4]Uma *spritesheet* é uma imagem contendo várias imagens em sequência do mesmo objeto, que se colocados em sequência gerarão uma animação.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão apresentados a definição de um jogo educacional (Seção 2.1) e mais detalhes sobre o termo *misconception* na programação bem como alguns exemplos práticos (Seção 2.2).

2.1 JOGO EDUCACIONAL

Segundo Kapp (2013), um jogo pode ser definido como: "Um sistema no qual jogadores se engajam em um desafio abstrato, definido por regras, interatividade e *feedback* e que gera um resultado quantificável, frequentemente elicitando uma reação emocional". Cada tipo de jogo tem suas próprias características e objetivos que pode ser explorada para apoiar o processo de aprendizagem.

Os jogos de entretenimento são concebidos com o propósito principal de divertir e entreter seus jogadores, ou seja, são jogos que não possuem viés pedagógico. Nesse contexto, existem diversos jogos, como videogames, cubo mágico e RPGs. Surpreendentemente, esse tipo de jogo também pode proporcionar resultados pedagógicos que podem apoiar o processo de aprendizagem (Costa, 2008). Por outro lado, tem-se os jogos sérios que tem como objetivo ensinar e/ou treinar seus jogadores (de Oliveira et al., 2016). Por exemplo, o jogo ABCLingo (Karlini e Rigo, 2014) é um jogo sério que atua na área de alfabetização e mineração de dados, que disponibiliza relatórios de avaliação e permite ao professor verificar o desempenho individual e coletivo dos seus alunos. De forma geral, esse tipo de jogo tem objetivo claro, porém não significa que não possa ser divertido (Abrantes et al., 2022).

Unindo os dois universos (jogos de entretenimento e jogos sérios), chega-se nos jogos educacionais que possuem os objetivos de entreter e ensinar. Esses jogos são utilizados no ambiente escolar e podem tornar o ensino e aprendizagem muito eficientes (Silva e Soares, 2023).

Os jogos educacionais podem ser do tipo analógico ou digital. Um jogo analógico consiste em um jogo físico regido por um conjunto de regras que são mais flexíveis que os jogos digitais, já que os jogos digitais acontecem no meio virtual e tem suas regras e normas determinadas no código fonte do jogo (Ramos et al., 2020).

Segundo Fernanda Zortea et al. (2017), os jogos educacionais digitais proporcionam aos alunos a oportunidade de aprender de maneira dinâmica e motivadora, além de promover o aumento da atenção, disciplina e também estimular o pensamento crítico. Esses jogos são elaborados para divertir, aumentar os conhecimentos e melhorar a aprendizagem do aluno sobre os conteúdos contidos no jogo e, também, estimular as habilidades de percepção, exploração, estratégia e tentativa e erro. Dessa forma, os jogos educacionais oferecem aos alunos uma maneira mais atrativa e interativa de aprender, assim, se tornando ferramentas que contribuem com o processo de ensino e aprendizagem.

Porém, existem dificuldades para criar um jogo educacional. Por exemplo, cada jogo tem sua própria forma de abordar didaticamente as dificuldades dos alunos. Outra dificuldade se concentra na distância da cultura de jogar por parte dos produtores de conteúdo e do público-alvo. Como exemplo, alguns docentes podem não estar familiarizados com os jogos digitais (Perry et al., 2007). Por esse motivo, diversos trabalhos discutem estratégias para construir um jogo educacional (de Oliveira et al., 2018), começando pelo *Game Design* que deve ser interessante e educativo (Fernandes et al., 2018).

A programação visual tem sido uma grande aliada no apoio de disciplinas introdutórias de programação, além de facilitar a aprendizagem, também contribui para a resolução de problemas (Ribas et al., 2016). Um exemplo de jogo que utiliza programação visual é o LightBot, nele o jogador deve criar programas que guiam o robô até o destino final (Cardoso e Antonello, 2015). Esses programas são criados por meio de comandos em blocos, por exemplo, andar para frente, virar à esquerda, pular e usar procedimentos repetitivos, onde o robô executa o mesmo comando mais de uma vez. Assim, o jogo trabalha tanto com estruturas sequenciais quanto estruturas de repetição, permitindo que o programa criado seja observado em execução no robô (Souza et al., 2018).

Conhecendo os principais benefícios que os jogos podem oferecer à educação, torna-se evidente a importância de seu estudo e desenvolvimento. Logo, será apresentado na próxima seção, o conceito central que fundamenta este trabalho: os *misconceptions*.

2.2 MISCONCEPTION

Um *misconception* é uma falsa concepção de um conceito essencial que pode ocasionar dificuldades na resolução de um problema (Qian e Lehman, 2017). No contexto da programação, *misconception* é a compreensão imprecisa ou incompleta de um conceito de programação que pode gerar erros. No entanto, um erro gerado por um *misconception* é diferente de um erro comum, pois um erro comum é apenas uma configuração incorreta de elementos de código (Gusukuma et al., 2018). Um erro ocasionado por um *misconception* pode ser difícil de ser detectado e corrigido (Araujo et al., 2021), já que o programa muitas vezes executa sem gerar mensagens de erro.

A seguir são apresentados alguns exemplos de *misconceptions* na programação. Os exemplos foram baseados no artigo de Araujo et al. (2021) que mapeou os *misconceptions* mais comuns em programação no contexto da linguagem Python. Como este trabalho irá utilizar códigos na linguagem de programação C, os exemplos mostrados no artigo de Araujo et al. (2021), em forma de texto, foram adaptados para trechos de códigos na linguagem C.

Exemplo 1: problema do laço infinito. Geralmente, esse tipo de erro ocorre quando o aluno “esquece” de incrementar (ou decrementar) a variável de controle do laço e a condição de parada nunca é satisfeita, gerando assim, um laço infinito. Um exemplo de código, utilizando a estrutura de repetição *while* é apresentado na Figura 2.1.

```c
#include <stdio.h>

int main()
{

   int i = 0;

   while(i < 10){
      printf("Ola, Mundo!\n");
   }

   return 0;
}
```

Figura 2.1: Exemplo de *misconception* em programação: laço infinito. Adaptado de Araújo et al. (2021), Tabela 1, ID X11.

Exemplo 2: acesso a posições inválidas de um vetor. Para melhor entendimento deste *misconception*, suponha que foi declarado um vetor com dez posições. Para percorrer todas as posições desse vetor na linguagem C, é necessário utilizar um laço de repetição que vai da posição de índice 0 (zero) até a posição de índice 9 (nove). Entretanto, ocorrem casos onde o aluno tenta realizar a iteração fora da faixa de valores permitida (por exemplo, no intervalo de 1 à 10), fazendo com que não seja possível acessar a última posição deste vetor. Um exemplo de código que representa esse *misconception* é mostrado na Figura 2.2.

```
#include <stdio.h>

int main()
{

   int vetor[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

   int i;

   for(i = 1;i <= 10;i++)
      printf("vetor[%d] = %d\n", i, vetor[i]);

   return 0;
}
```

Figura 2.2: Exemplo de *misconception* em programação: acesso a posição inválida de um vetor. Adaptado de Araújo et al. (2021), Tabela 1, ID PB1.

Exemplo 3: variável fora de escopo. Nesse caso, conforme mostrado na Figura 2.3, o aluno define localmente uma variável 'i' para realizar a iteração dentro do laço *for*, mas posteriormente tenta acessar essa variável fora do escopo da repetição. Vale ressaltar que, no caso da Figura 2.3, o código não compila.

```
#include <stdio.h>

int main()
{

   for(int i = 1;i <= 10;i++) {
      printf("Numero: %d\n", i);
   }

   printf("Ultimo valor: %d", i);

   return 0;

}
```

Figura 2.3: Exemplo de *misconception* em programação: variável fora do escopo. Adaptado de Araújo et al. (2021), Tabela 1, ID X1.

Exemplo 4: uso incorreto dos operadores lógicos E (&&), OU (||) e/ou NOT (!). No exemplo da Figura 2.4, é feita a leitura de um número inteiro, representando a idade de uma pessoa. Na sequência, é feita a verificação, por meio do comando de decisão *if*, ou seja, se a idade da pessoa é maior que zero OU menor que 18. Ao utilizar o operador lógico OU, se qualquer

uma das comparações for verdadeira, toda a condição é satisfeita. Logo, qualquer valor digitado (positivo ou negativo), fará com que a condição seja verdadeira, e, com isso, a mensagem "Menor de idade" será sempre impressa na tela do usuário.

```c
#include <stdio.h>

int main()
{

   int idade;

   printf("Entre com a idade: ");

   scanf("%d", &idade);

   if(idade > 0 || idade < 18)
      printf("Menor de idade");
   else
      printf("Maior de idade");

   return 0;
}
```

Figura 2.4: Exemplo de *misconception* em programação: operadores lógicos. Adaptado de Araújo et al. (2021), Tabela 1, ID A2.

Exemplo 5: uso incorreto de operadores relacionais como maior que (>), menor que (<), igual a (==), entre outros. No exemplo da Figura 2.5, o operador lógico E (&&) está sendo utilizado corretamente, contudo, a forma como os operadores "<" (menor que) e ">" (maior que) foram aplicados faz com que a condição da instrução *if* seja sempre falsa. Com isso, a mensagem "Menor de idade" nunca será impressa na tela do usuário.

```c
#include <stdio.h>

int main()
{

   int idade;

   printf("Entre com a idade: ");

   scanf("%d", &idade);

   if(idade < 0 && idade > 18)
      printf("Menor de idade");
   else
      printf("Maior de idade");

   return 0;
}
```

Figura 2.5: Exemplo de *misconception* em programação: uso incorreto dos operadores relacionais. Adaptado de Araújo et al. (2021), Tabela 1, ID A1.

3 TRABALHOS RELACIONADOS

Jogos digitais podem ser ferramentas educacionais eficientes, pois além de divertir, facilitam o aprendizado e aumentam a retenção de conteúdo pelos alunos (Tarouco et al., 2004). Logo, nesta seção, serão apresentados seis jogos educacionais que tem como objetivo apoiar estudantes que estejam cursando disciplinas introdutórias de programação e/ou algoritmos.

O jogo Projeto Éden (Oliveira e Farias, 2019) tem como foco principal a manipulação de variáveis e tipos de dados, considerando as linguagens de programação mais utilizadas por estudantes iniciantes, como: VisualG, Java, C# e Pascal. O jogo possui dez fases. Cada fase trabalha uma habilidade nova e essencial para a resolução da fase seguinte, ou seja, o jogador deve declarar variáveis, atribuir valores a elas e usar operações lógicas e aritméticas para passar de fase. O jogo foi projetado para rodar no ambiente *web* e foi construído utilizando a ferramenta Construct2, um aplicativo que permite a criação rápida de jogos sem a necessidade de utilizar uma linguagem de programação. Além disso, essa ferramenta permite a exportação do código fonte em HTML5.

Assim como o Projeto Éden, o World Prog (de Holanda e da Silva Coutinho, 2022) é um jogo educacional que tem como objetivo ensinar os conceitos básicos de programação: variáveis, tipos de dados e operações aritméticas. Nesse jogo, caracterizado como exploratório, o personagem principal viaja entre planetas resolvendo desafios de programação. Os desafios são caixas com perguntas e respostas em que o jogador deve selecionar a resposta correta para avançar de fase. O jogo foi desenvolvido utilizando o motor gráfico Unity e a linguagem de programação C#.

O Klouro (de Azevêdo Silva e Dantas, 2014), por sua vez, é um jogo com o objetivo de ajudar o aluno a compreender os operadores lógicos (ex.: AND e OR) e operadores aritméticos (ex.: multiplicação e soma) da linguagem de programação Python. O jogo possui duas fases, cada fase consiste em um trecho de código que testa o valor de alguma variável, por exemplo, se X é igual a cinco. A partir disso, o jogador deve ser capaz de interpretar o código e incrementar ou decrementar o valor dessa variável para que o teste dê verdadeiro, assim o jogo avança para a próxima fase. O jogo foi construído utilizando a ferramenta Construct2.

O Catacombs (Ralph e Barnes, 2007), um jogo desenvolvido com o motor gráfico Aurora, tem como proposta ensinar o básico de lógica de programação, desde variáveis até estruturas condicionais e laços de repetição. O jogo utiliza a temática de RPG (*Role-Playing Game* ou jogo de interpretação de personagem em Português) onde o jogador é um mago que deve responder perguntas a fim de preencher um código corretamente. O jogo possui uma linguagem própria em que o jogador escreve as respostas e foi pensado para o público universitário que já possui conhecimentos básicos em Ciência da Computação. Neste jogo existem três principais desafios: no primeiro, o jogador deve usar a lógica de dois blocos condicionais para destrancar uma porta, no segundo, ele deve usar laços de repetição para construir uma ponte; e, no último, deve usar laços de repetição aninhados para resolver um criptograma.

O Super Mario Logic (Panegalli et al., 2019) é um jogo que também se concentra no ensino de lógica de programação no nível sequencial, repetição e decisão. As ações do jogo se dão por meio de comandos, como andar para frente, pular, abaixar, entre outros. Os comandos são digitados pelo jogador que deve escolher a sequência de ações mais adequada para cada situação. O jogo foi feito para auxiliar estudantes de graduação em Computação, que estejam cursando disciplinas de programação e algoritmos. No desenvolvimento, foram utilizados o motor gráfico Unity e a linguagem de programação C#.

O MazeLogic (Oliveira et al., 2018), é um jogo desenvolvido utilizando o motor gráfico Unity cujo propósito é ensinar os conceitos de lógica de programação e raciocínio lógico computacional. O jogo possui um *ranking* de pontuação entre os jogadores, um menu de opções e um sistema de *login*. O menu de opções contém uma tela para os créditos e para a alteração do volume de áudio do jogo. O *login* é realizado pela conta do dispositivo móvel do jogador como Google ou iCloud, *login* esse, onde a pontuação de *ranking* do jogador são salvos. O jogo propriamente dito tem dois modos: nivelamento e labirinto. O nivelamento busca testar os conhecimentos do jogador a partir de perguntas de múltipla escolha, possuindo três níveis de dificuldade, do mais básico ao mais avançado (neste último podem aparecer questões discursivas). O modo labirinto, por outro lado, exibe ao jogador o mapa do labirinto e, para avançar no jogo, deve responder perguntas. Se responder corretamente, o jogador avança em direção ao final do labirinto, caso contrário, uma rota mais longa deve ser percorrida.

Também é importante citar a plataforma CodinGame [1], uma ferramenta *online* que oferece uma variedade de jogos e desafios dedicados ao aprendizado de programação. Nela é possível estudar diversas linguagens de programação, como Python, Java, C++, entre outras, e pode ser utilizada tanto por alunos iniciantes quanto por programadores mais experientes.

Apesar das importantes contribuições dos trabalhos apresentados neste capítulo, para apoiar o ensino dos conceitos introdutórios de programação, é importante observar que nenhum deles se concentrou especificamente nos erros causados por *misconceptions* cometidos por alunos iniciantes em programação. No intuito de contribuir com as pesquisas em jogos educacionais, direcionadas a apoiar o ensino de programação, este trabalho tem como proposta utilizar uma abordagem baseada em *misconceptions* de programação para estudantes iniciantes. Para isso, serão propostos desafios que envolvem o uso de laços de repetição, tanto de forma isolada quanto em combinação com outras estruturas de dados, como vetores e matrizes. Para isso será utilizada a linguagem de programação C como base.

[1] Disponível em https://www.codingame.com/start/.

4 MATERIAIS E MÉTODOS

Para atingir o objetivo proposto, este trabalho foi dividido em duas fases. A primeira fase consistiu na realização de uma revisão bibliográfica, fundamentada em artigos acadêmicos e trabalhos de conclusão de curso, com o objetivo de identificar jogos educacionais voltados para o ensino de programação e definir o público-alvo. Durante essa pesquisa não foram encontrados estudos que investigassem como erros causados por *misconceptions* em laços de repetição poderiam ser abordados em um jogo educacional.

Na segunda fase, procedeu-se ao desenvolvimento do jogo. Inicialmente, foi feito o esboço das telas em papel, permitindo a pré-visualização das interações entre os personagens no jogo e como os desafios seriam apresentados ao jogador. A ideia inicial era que o jogador corrigisse códigos escritos na linguagem C, digitando diretamente em um editor de texto incorporado ao jogo. Porém, após uma análise, verificou-se que essa abordagem seria inviável, pois demandaria o desenvolvimento de um analisador léxico para validar o que fosse digitado pelo jogador. Logo, optou-se por uma abordagem com botões *drop-down*[1] contendo partes de código já escritas. Dessa forma, o jogador deveria apenas selecionar a opção de código que considerasse correta para solucionar o desafio proposto.

Com o protótipo definido, foi realizada uma pesquisa para identificar quais ferramentas de desenvolvimento de jogos poderiam ser usadas para o desenvolvimento do jogo educacional "*Infinity Loop*". Inicialmente, foram selecionadas duas ferramentas: Godot 4 e Construct 2. Apesar de ambas atenderem às necessidades do projeto, o Construct 2 apresentava algumas limitações, como número máximo de *scripts* na versão gratuita. Por esse motivo, optou-se pelo Godot 4, uma *engine* muito poderosa com uma documentação abrangente disponível *online*. Além disso, o Godot 4 oferece suporte para exportar o jogo para Windows, Linux, *Mobile* e *web*.

O mapa do jogo foi desenvolvido com um *tilemap* em 2D. O *tileset* usado para criar o *design* visual foi o "12x12 RPG Tileset"[2], que possui licença personalizada que pode ser usada tanto em projetos gratuitos quanto comerciais. Após a criação do mapa, o próximo passo foi desenvolver a cena do jogador, utilizando *sprite* e animações próprias, bem como codificar as leituras de entradas do teclado para que o jogador pudesse realizar ações no mapa. Na sequência, foi desenvolvida uma cena do NPC que pudesse ser replicada sempre que fosse necessário adicionar um novo NPC ao mapa. Cada NPC é capaz de interagir com o jogador quando requisitado e apresentar diferentes falas. Por meio desses diálogos, o NPC explica ao jogador o que deve ser feito no desafio.

O jogo foi dividido em cinco desafios de programação. A cada desafio concluído uma nova área do mapa é liberada, e um novo desafio é apresentado por um NPC diferente. As figuras dos NPCs foram extraídas do *tileset* "Roguelike/RPG Pack"[3] que se encontra sob a licença Creative Commons CC0, que permite o uso livre em projetos gratuitos e comerciais.

Na implementação dos desafios do jogo, foi utilizado o modelo Incremental[4], em que a cada desafio foi necessário levantar os requisitos como: qual *misconception* será explorado, quais objetos de mapa podem ser usados, qual obstáculo do mapa será removido após passar o desafio. Com os requisitos definidos, foi feito o planejamento dos prazos (cada desafio demorava

[1] Um botão drop-down é um botão que, ao ser pressionado, abre uma lista de opções.

[2] Disponível em https://cypor.itch.io/12x12-rpg-tileset.

[3] Disponível em https://kenney.nl/assets/roguelike-rpg-pack.

[4] O modelo Incremental é um modelo de processo de *software* que trabalha com incrementos, com pequenos pedaços de *software* entregues de cada vez (Braatz et al., 2018).

cerca de duas semanas para ser completamente desenvolvido) e, por fim, foi feita a codificação e teste do desafio. Visando maior produtividade, foi utilizado o sistema de versionamento de código Git[5]. Isso permitiu trabalhar em paralelo nos arquivos do jogo. Cada integrante da equipe tinha sua própria *branch*, o que possibilitou salvar versões funcionais e também testar novas funcionalidades sem comprometer a *Branch Main*, que contém o código fonte principal e funcional.

[5]O Git é um sistema de controle de versões distribuído, usado principalmente no desenvolvimento de *software*.

5 RESULTADOS

Neste capítulo, serão apresentadas as principais telas do jogo educacional "*Infinity Loop*", incluindo os cinco desafios que exploram erros causados por *misconceptions* relacionados ao uso das estruturas de repetição na linguagem C. Os *misconceptions* abordados são: "acesso a posição inválida de um vetor", " laço infinito", "uso incorreto dos operadores lógicos" e "uso incorreto dos operadores relacionais". É importante mencionar que o jogo baseia-se em uma narrativa onde o jogador é representado por um "Mago do Código", que encontra vários NPCs ao longo do caminho. Cada NPC enfrenta dificuldades de realizar suas tarefas, como consertar uma ponte, consertar uma cerca, plantar árvores, dentre outros. Além disso, esses NPCs também apresentam desafios ao jogador, que só poderá avançar no jogo após concluí-los. Os diálogos completos dos NPCs com o jogador e a descrição dos desafios, são apresentados no Apêndice A. Conforme mostrado na Figura 5.1, a tela inicial exibe o nome do jogo e um botão para jogar. As demais telas, com a explicação dos desafios, são mostradas nas próximas subseções.

Figura 5.1: Tela inicial do jogo educacional "*Infinity Loop*".

5.1 DESAFIO 1

No primeiro desafio, o jogador encontra um NPC camponês que pede ajuda para consertar uma ponte quebrada (Figura 5.2). O camponês explica que a ponte pode ser reconstruída utilizando um código em C e pergunta se o jogador que é um "Mago do Código" pode ajudá-lo.

Figura 5.2: O jogador encontra uma ponte quebrada e o NPC camponês.

No desafio apresentado nas Figuras 5.3 e 5.4, o *misconception* explorado é o de "acesso a posição inválida de um vetor". Nesse caso, a ponte quebrada é representada por um vetor de sete posições. O objetivo é que o jogador adicione tábuas a ponte e, para isso, ele deve completar os campos do laço *for* (linha 6), respeitando as regras da linguagem C em que um vetor começa na posição zero e termina na posição "tamanho_do_vetor_menos_1".

Figura 5.3: Tela do jogo exibindo as instruções para resolução do Desafio 1 sobre o *misconception* de "acesso a posição inválida do vetor".

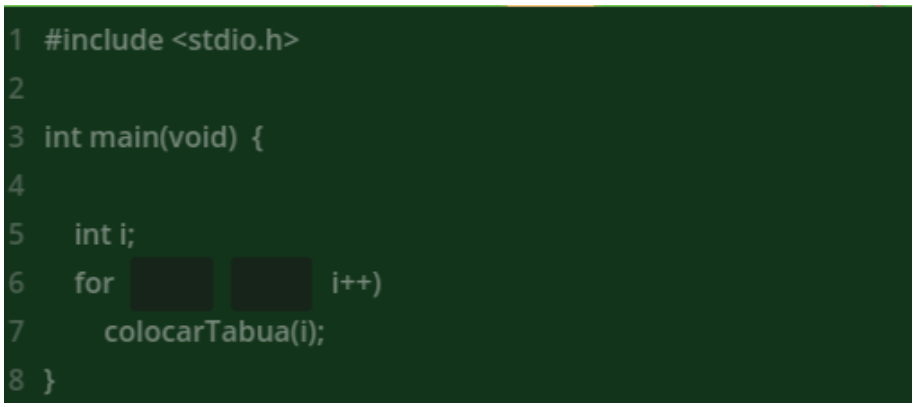


Figura 5.4: Código do Desafio 1 sobre o *misconception* de "acesso a posição inválida de um vetor".

Na Tabela 5.1 são mostradas todas as possibilidades de respostas para o Desafio 1. Nesse caso, as opções 5 e 6 são as corretas.

Tabela 5.1: Opções de resposta para o Desafio 1.

Opção	Inicialização do laço *for*	Condição do laço *for*
1	i = -1	i <= 7
2	i = -1	i <= 6
3	i = -1	i < 7
4	i = 0	i <= 7
5	**i = 0**	**i < 7**
6	**i = 0**	**i <= 6**

Para verificar se a resposta selecionada está correta, o jogador deve clicar no botão "Run Code" mostrado na Figura 5.3. A Figura 5.5 exibe a mensagem de *feedback* quando o jogador preencheu os campos do desafio com a opção 2 da Tabela 5.1. Observe que além da mensagem de *feedback*, uma parte da ponte já existente é destacada em vermelho para indicar que uma posição inválida do vetor foi acessada. As mensagens de *feedback* para as demais opções de resposta da Tabela 5.1 podem ser encontradas na Tabela B.1 - Apêndice B.

Figura 5.5: Mensagem de *feedback* quando uma opção incorreta é selecionada no Desafio 1.

5.2 DESAFIO 2

No segundo desafio, o jogador se depara com uma cerca quebrada e com o NPC construtor, que solicita ajuda para consertá-la (Figura 5.6). Neste desafio, o *misconception* abordado é o de "laço infinito", onde o jogador precisa iterar sobre um laço de repetição *while* a fim de adicionar cinco estacas a cerca (Figura 5.8). Para isso, ele deve preencher os campos que faltam no laço *while* (linhas 4 e 7) e, na sequência, clicar no botão "Run Code" (Figura 5.7).

Figura 5.6: O jogador encontra uma cerca quebrada e o NPC construtor.

Figura 5.7: Tela do jogo exibindo as instruções para a resolução do Desafio 2.

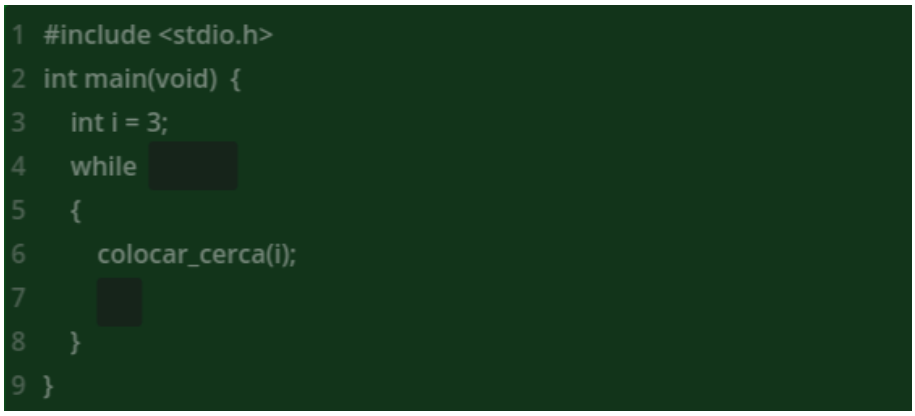


Figura 5.8: Código do Desafio 2 sobre o *misconception* de "laço infinito".

Conforme mostrado na Tabela 5.2, o erro de laço infinito pode ocorrer de diversas formas. No Desafio 2, as opções de respostas corretas são 8 e 11.

Tabela 5.2: Opções de resposta para o Desafio 2.

Opção	Condição do laço *while*	Iteração do laço *while*
1	(i <= 7);	i--
2	(i <= 7)	i--
3	(i = 7)	i--
4	1 < 8	i--
5	(i < 8)	i--
6	(i = 8)	i--
7	(i <= 7);	i++
8	**(i <= 7)**	**i++**
9	(i = 7)	i++
10	1 < 8	i++
11	**(i < 8)**	**i++**
12	(i = 8)	i++

No caso da Figura 5.9, o jogador preencheu os campos do laço *while* (linhas 4 e 7) com a opção 2 da Tabela 2. Nesse caso, o erro ocorreu devido ao uso do operador de decremento i--. Além da mensagem de *feedback*, estacas em vermelho foram inseridas dinamicamente na direção contrária da cerca quebrada, para simular o laço infinito. As mensagens de *feedback* para as demais opções de resposta da Tabela 5.2 podem ser encontradas na Tabela B.2 - Apêndice B.

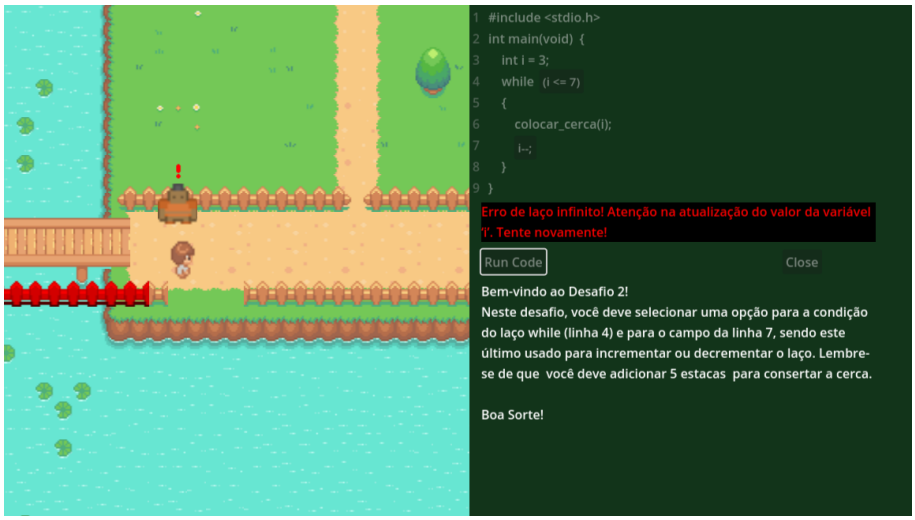


Figura 5.9: Mensagem de *feedback* quando uma opção incorreta é selecionada no Desafio 2.

5.3 DESAFIO 3

No terceiro desafio, o jogador deve ajudar a NPC lenhadora a plantar árvores para completar o percurso até uma floresta. Conforme mostrado na Figura 5.10, árvores foram plantadas apenas à direita do caminho. Logo, o objetivo é que o jogador plante a mesma quantidade de árvores no lado esquerdo do caminho, alinhadas às árvores do lado direito.

Figura 5.10: O jogador encontra a NPC lenhadora e observa que árvores foram plantadas apenas de um lado do caminho até a floresta.

O *misconception* explorado no Desafio 3 (Figura 5.11 e 5.12) é o de "uso incorreto de operadores relacionais". Para resolvê-lo, o jogador deve completar o laço *for* (linha 4) e escolher um operador relacional para a condição do *if* (linha 6).

Figura 5.11: Tela do jogo exibindo as instruções para a resolução do Desafio 3.

```
1 #include <stdio.h>
2 int main(void) {
3    int tam_floresta = 8;
4    for (int i = 1;          ; i++)
5    {
6       if (i % 2    0)
7          plantarArvore(i);
8    }
9 }
```

Figura 5.12: Código do Desafio 3 sobre o *misconception* “uso incorreto de operadores relacionais”.

Na Tabela 5.3 são mostradas todas as possibilidades de respostas para o Desafio 3. Nesse caso, as opções 8 e 10 são as corretas.

Tabela 5.3: Opções de resposta para o Desafio 3.

Opção	Condição do laço *while*	Condição do comando *if*
1	i < tam_floresta	>=
2	i < tam_floresta	<
3	i < tam_floresta	==
4	i < tam_floresta	!=
5	i < tam_floresta	<=
6	i <= tam_floresta	>=
7	i <= tam_floresta	<
8	**i <= tam_floresta**	**==**
9	i <= tam_floresta	!=
10	**i <= tam_floresta**	**<=**
11	i < (tam_floresta - 1)	>=
12	i < (tam_floresta - 1)	<
13	i < (tam_floresta - 1)	==
14	i < (tam_floresta - 1)	!=
15	i < (tam_floresta - 1)	<=
16	i <= (tam_floresta - 1)	>=
17	i <= (tam_floresta - 1)	<
18	i <= (tam_floresta - 1)	==
19	i <= (tam_floresta - 1)	!=
20	i <= (tam_floresta - 1)	<=

Na Figura 5.13, o jogador completou a condição do laço *for* e da estrutura de decisão *if* com a opção 6 da Tabela 5.3. Nesse caso, a resposta está incorreta por causa do operador relacional selecionado para a condição do *if*. Ao utilizar o operador maior ou igual ">=", a condição do *if* será sempre verdadeira e árvores em vermelho serão plantadas dinamicamente em todas as posições. As mensagens de *feedback* para as demais opções de resposta da Tabela 5.3 podem ser encontradas na Tabela B.3 - Apêndice B.

Figura 5.13: Mensagem de *feedback* quando uma opção incorreta é selecionada no Desafio 3.

5.4 DESAFIO 4

No quarto desafio, o jogador encontra um NPC fazendeiro que pede ajuda para plantar sementes em um terreno (Figura 5.14). O *misconception* explorado é o de "uso incorreto de operadores lógicos". Nesse caso, o jogador deve indicar dois operadores na condição do laço *while* (linha 6 das Figuras 5.15 e 5.16), para que as sementes sejam plantadas de forma correta. Para verificar se a resposta está correta, o jogador deve clicar em "Run Code" (Figura 5.15).

Figura 5.14: O jogador encontra uma plantação vazia e o NPC fazendeiro.

Figura 5.15: Tela do jogo exibindo as instruções para a resolução do Desafio 4.

```
1  #include <stdio.h>
2  int main(void) {
3    int qtde_milho = 5;
4    int qtde_trigo = 7;
5    int tam_plantacao = 0;
6    while ((qtde_milho > 0)    (qtde_trigo > 0)    (tam_plantacao < 16))
7    {
8      plantarMilho();
9      plantarTrigo();
10     qtde_milho--;
11     qtde_trigo--;
12     tam_plantacao += 2;
13   }
14 }
```

Figura 5.16: Código do Desafio 4 sobre o *misconception* “uso incorreto de operadores lógicos”.

Na tabela 5.4 são mostradas todas as possibilidades de respostas para o Desafio 4. Nesse caso, apenas a opção 4 é a correta.

Tabela 5.4: Opções de resposta para o Desafio 4

Opção	Primeiro operador	Segundo operador
1	||	||
2	||	&&
3	&&	||
4	**&&**	**&&**

A Figura 5.17 apresenta o *feedback* após o jogador selecionar a opção 1 da tabela 5.4. Observe que, além da mensagem de *feedback*, algumas sementes foram plantadas com a cor marrom escuro, indicando que o código tentou plantar mesmo não havendo mais sementes disponíveis. As mensagens de *feedback* para as demais opções de resposta da tabela 5.4 podem ser encontradas na Tabela B.4 - Apêndice B.

Figura 5.17: Mensagem de *feedback* quando uma opção incorreta é selecionada no Desafio 4.

5.5 DESAFIO 5

No quinto desafio, o jogador encontra um NPC jardineiro que pede ajuda para plantar flores em um jardim, representado por uma matriz 10x10 (Figura 5.18). Neste desafio são explorados vários erros causados por *misconception*: "laço infinito", "uso incorreto de operadores relacionais" e "acesso a posição inválida de vetor". Nesse caso, o jogador deve selecionar cinco opções para completar o desafio: duas para os laços "*for*" (interno e externo) e três para os blocos condicionais "*if*" (Figuras 5.19 e 5.20). Após preencher os cinco campos, o jogador deve clicar em "Run Code" e verificar se sua resposta está correta.

Figura 5.18: O jogador encontra o jardim vazio e o NPC jardineiro.

Figura 5.19: Tela do jogo exibindo as instruções para a resolução do Desafio 5.

```
1  #include <stdio.h>
2  int main(void) {
3      int jardim[10][10];
4      for (int i = 0;        i++)
5      {
6          for (int j = 0;        j++)
7          {
8              if (      ) jardim[i][j] = flor_branca;
9              if (      ) jardim[i][j] = flor_vermelha;
10             if (      ) jardim[i][j] = flor_amarela;
11         }
12     }
13 }
```

Figura 5.20: Código do Desafio 5 sobre diversos *misconceptions*.

Na Tabela 5.5 são mostradas, algumas possibilidades de resposta para o Desafio 5, onde apenas a opção 4 é a correta. Todas as mensagens de *feedback* para as demais opções de resposta da Tabela 5.5 estão disponíveis na Tabela B.5 - Apêndice B.

Tabela 5.5: Opções de resposta para o Desafio 5

Opção	Condição do primeiro laço*for*	Condição do segundo laço *for*	Condição do primeiro comando *if*	Condição do segundo comando *if*	Condição do terceiro comando *if*
1	i <= 10	j <= 10	i == j	i > j	i < j
2	i <= 10	j < 10	i > j	i == j	i < j
3	i < 10	j < 10	i < j	i == j	i > j
4	**i < 10**	**j < 10**	**i == j**	**i > j**	**i < j**

Na Figura 5.21 tem-se o *feedback* dado ao jogador após a seleção da opção 3 da Tabela 5.5. Neste caso, as flores são plantadas fora da ordem especificada no enunciado. Embora as condições dos laços *for* estejam corretas, as condições dos *ifs* não estão. Isso resulta em um erro devido ao "uso incorreto de operadores relacionais".

Figura 5.21: Mensagem de *feedback* quando são selecionadas opções incorretas no Desafio 5.

5.6 RELATÓRIO DE ERROS

Após finalizar o Desafio 5, o NPC Jardineiro oferece a opção ao jogador de “Ver relatório de erros” (Figura 5.22). Ao clicar nessa opção, uma nova tela é aberta e são apresentados os erros cometidos pelo jogador, seguido pela quantidade de ocorrências (Figura 5.23). Os erros são classificados por tipo de ***misconception***. Ao final do relatório, são mostrados os conceitos que o jogador deve revisar.

Figura 5.22: NPC Jardineiro mostra opção de ver relatório de erros cometidos

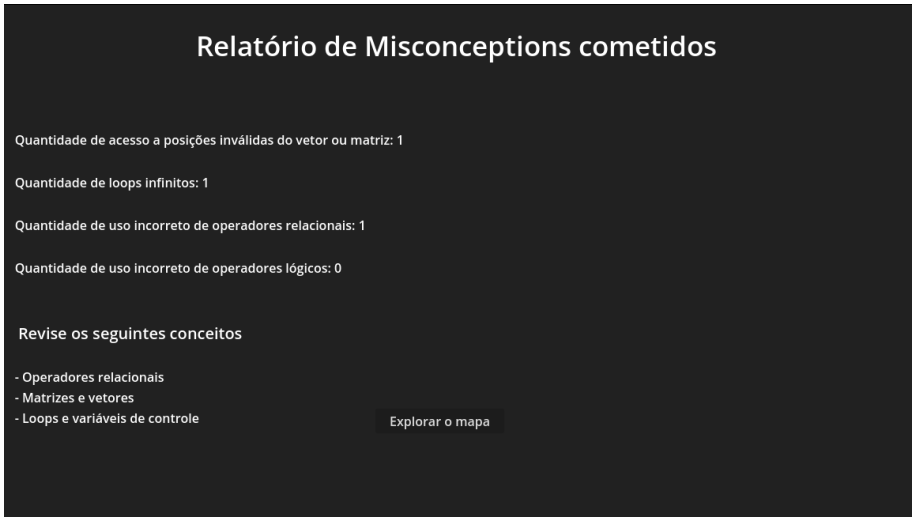


Figura 5.23: Relatório de erros causados por *misconceptions*

6 CONCLUSÕES

Este Trabalho de Conclusão de Curso apresentou o desenvolvimento do jogo educacional "*Infinity Loop*", que tem como objetivo apoiar o ensino de programação, focando em erros causados por *misconceptions* em estruturas de repetição. O principal resultado deste estudo é um jogo que aborda quatro erros causados por *misconceptions* em estruturas de repetição: "acesso a posição inválida do vetor", "laço infinito", "uso incorreto ao utilizar operadores lógicos" e "uso incorreto ao utilizar operadores relacionais". O jogo oferece um ambiente lúdico e dinâmico, onde a interação com o aluno tem influência direta no mapa do jogo, ou seja, quando o estudante seleciona uma opção correta ou incorreta para os desafios, a execução é incorporada ao cenário. Após completar todos os desafios, o jogo disponibiliza um relatório com os erros de *misconception* cometidos pelo jogador e sugere os conceitos de programação que o estudante deve revisar. O jogo encontra-se na *web* e seus respectivos arquivos fonte[1] foram disponibilizados na plataforma do Github.

Como trabalhos futuros, recomenda-se a inserção de músicas e efeitos sonoros, bem como a realização de testes funcionais nos desafios do jogo, além de testes com estudantes em contextos reais de aprendizagem. Sugere-se que os testes com os estudantes avaliem, por exemplo, a satisfação do aluno em relação à narrativa do jogo, bem como os ganhos de aprendizagem após a resolução dos desafios. Além disso, sugere-se também o desenvolvimento de um analisador léxico nos desafios do jogo, ao invés de utilizar os botões *dropdown*. Acredita-se que isso pode proporcionar aos estudantes uma experiência ainda mais enriquecedora ao resolver os desafios, possibilitando a escrita de códigos funcionais ao invés de selecionar opções pré-determinadas. Acredita-se que isso minimizaria a prática do *gaming the system* (Nunes e Jaques, 2014), onde o jogador escolhe as opções aleatoriamente na tentativa de avançar rapidamente para os próximos desafios.

[1] Disponível em https://github.com/Hproko/InfinityLoop.

REFERÊNCIAS

Abrantes, A. L. R. C., Lima, B. A. d., Angelo, F. d. M., Souza, M. d. F. d., Silva, R. J. d., Pinheiro, S. A. e Brito, C. A. F. (2022). Serious game: Tracking school knowledge. *Research, Society and Development*, 11(13):e352111335406.

Araujo, A., Filho, D., Oliveira, E., Carvalho, L., Pereira, F. e Oliveira, D. (2021). Mapeamento e análise empírica de misconceptions comuns em avaliações de introdução à programação. Em *Anais do Simpósio Brasileiro de Educação em Computação*, páginas 123–131, Porto Alegre, RS, Brasil. SBC.

Blatt, L., Becker, V. e Ferreira, A. (2017). Mapeamento sistemático sobre metodologias e ferramentas de apoio para o ensino de programação. Em *Anais do XXIII Workshop de Informática na Escola*, páginas 815–824, Porto Alegre, RS, Brasil. SBC.

Bosse, Y. e Gerosa, M. (2015). Reprovações e trancamentos nas disciplinas de introdução à programação da universidade de são paulo: Um estudo preliminar. Em *Anais do XXIII Workshop sobre Educação em Computação*, páginas 426–435, Porto Alegre, RS, Brasil. SBC.

Braatz, D., de Oliveira, K. d. S. e da Rocha, T. R. (2018). O modelo de ciclo de vida iterativo/incremental para desenvolvimento de software. Em *7ª MOEPEX*.

Cardoso, R. e Antonello, S. (2015). Interdisciplinaridade, programação visual e robótica educacional: relato de experiência sobre o ensino inicial de programação. Em *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, volume 4, página 1255.

Carvalho, L., Santos, A., Nakamura, F. e Oliveira, E. (2019). Detecção precoce de evasão em cursos de graduação presencial em computação: um estudo preliminar. Em *Anais do XXVII Workshop sobre Educação em Computação*, páginas 233–243, Porto Alegre, RS, Brasil. SBC.

Costa, L. D. (2008). O que os jogos de entretenimento tÊm que os jogos com fins pedagÓgicos nÃo tÊm: PrincÍpios para projetos de jogos com fins pedagÓgicos. Dissertação de Mestrado, PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO - PUC-RIO.

de Azevêdo Silva, M. A. e Dantas, A. (2014). Klouro: Um jogo educacional para motivar alunos iniciantes em programação. *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação - SBIE)*, 25(1):702.

de Holanda, W. D. e da Silva Coutinho, J. C. (2022). World prog: Um jogo educacional para aprendizagem de conceitos básicos de programação. *Revista Novas Tecnologias na Educação*, 20:213–222.

de Oliveira, H. C., da Silva Hounsell, M. e Gasparini, I. (2016). Uma metodologia participativa para o desenvolvimento de jogos sérios. *Proceedings do XV SBGames–Trilha Artes e Design-Full Papers*.

de Oliveira, R. N. R., Cardoso, R. P., Braga, J. C. e da Rocha Campos, R. V. (2018). Frameworks para desenvolvimento de jogos educacionais: uma revisão e comparação de pesquisas recentes. Em *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, volume 29, página 854.

Duarte de Holanda, W., de Paiva Freire, L. e Cássia da Silva Coutinho, J. (2019). Estratégias de ensino-aprendizagem de programação introdutória no ensino superior: uma revisão sistemática da literatura. *Revista Novas Tecnologias na Educação*, 17(1):527–536.

Falkembach, G. A. M. (2006). O lúdico e os jogos educacionais. Em *Centro Interdisciplinar de Novas Tecnologias na Educação*.

Fernanda Zortea, C., Kliszcz, S., José Parreira, F. e Renato Silveira, S. (2017). "super zid": Desenvolvimento de um jogo educacional digital para apoiar o combate ao aedes aegypti. *Revista Novas Tecnologias na Educação*, 15(1).

Fernandes, K., Aranha, E. e Lucena, M. (2018). Estratégias para elaboração de game design de jogos digitais educativos: uma revisão sistemática. *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação - SBIE)*, 29(1):585.

Gomes, A., Areias, C., Henriques, J. e Mendes, A. (2008). Aprendizagem de programação de computadores: dificuldades e ferramentas de suporte. *Revista Portuguesa de Pedagogia*, 42:161–179.

Gusukuma, L., Bart, A. C., Kafura, D. e Ernst, J. (2018). Misconception-driven feedback: Results from an experimental study. Em *Proceedings of the 2018 ACM Conference on International Computing Education Research*, páginas 160–168.

Kapp, K. (2013). *The Gamification of Learning and Instruction Fieldbook: Ideas into Practice*. Wiley.

Karlini, D. e Rigo, S. J. (2014). Abclingo: Integrando jogos sérios e mineração de dados educacionais no apoio ao letramento. *Proceedings of SBGames*, páginas 1149–1152.

Medeiros, R., ao, T. F. e Ramalho, G. (2020). Ensino e aprendizagem de introdução à programação no ensino superior brasileiro: Revisão sistemática da literatura. Em *Anais do XXVIII Workshop sobre Educação em Computação*, páginas 186–190, Porto Alegre, RS, Brasil. SBC.

Monclar, R. S., Silva, M. A. e Xexéo, G. (2018). Xvii simpósio brasileiro de jogos e entretenimento digital. Em *Jogos com Propósito para o Ensino de Programação*.

Morais, C. G. B., Mendes Neto, F. M. e Osório, A. J. M. (2020). Difficulties and challenges in the learning process of algorithms and programming in higher education: a systematic literature review. *Research, Society and Development*, 9(10):e9429109287.

Moreira, G. L., Holanda, W., Coutinho, J. C. d. S. e Chagas, F. S. (2018). 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 (ISSN 2526-7574)*, 1(2).

Nunes, T. e Jaques, P. (2014). Utilizando agentes pedagógicos animados como uma abordagem não restritiva ao gaming the system. *Revista Brasileira de Informática na Educação*, 22:1–17.

Oliveira, A. V., Ribeiro, F. S., Schalski, L., Silva, J. C. F., Camargo, J. O., Farion, V., da Silva, C. F. e Mueller, R. R. (2018). Mazelogic: Jogo educacional para ensino de lógica de programação. *Anais SULCOMP*, 9.

Oliveira, Y. e Farias, C. (2019). Desenvolvimento e avaliação do jogo sério projeto Éden sobre variáveis e tipos de dados. Em *Anais da XIX Escola Regional de Computação Bahia, Alagoas e Sergipe*, páginas 615–624, Porto Alegre, RS, Brasil. SBC.

Panegalli, F. S., Bernardi, G. e Cordenonsi, A. Z. (2019). Super mario logic: um jogo sério para auxiliar no processo de ensino e aprendizagem de lógica de programação. *Revista Novas Tecnologias na Educação*, 17:244–253.

Perry, G. T., Timm, M. I., Silvestrim, F. G. e Schnaid, F. (2007). Necessidades específicas do design de jogos educacionais. *SBGames 2007*, páginas 7–9.

Qian, Y. e Lehman, J. (2017). Students' misconceptions and other difficulties in introductory programming: A literature review. *ACM Transactions on Computing Education (TOCE)*, 18:1–24.

Ralph, T. e Barnes, T. (2007). The catacombs: A study on the usability of games to teach introductory programming.

Ramos, D. K., Knaul, A. P. e Rocha, A. (2020). Jogos analógicos e digitais na escola: uma análise comparativa da atenção, interação social e diversão. *Revista Linhas*, 21(47):328–354.

Ribas, E., Dal Bianco, G. e Lahm, R. A. (2016). Programação visual para introdução ao ensino de programação na educação superior: uma análise prática. *RENOTE. Revista Novas Tecnologias na Educação*.

Silva, C. S. d. e Soares, M. H. F. B. (2023). Estudo bibliográfico sobre conceito de jogo, cultura lúdica e abordagem de pesquisa em um periódico científico de ensino de química. *Ciência & Educação (Bauru)*, 29:e23003.

Souza, D., Goulart, M., Guarda, G. e Goulart, I. (2018). Lightbot logicamente: um game lúdico amparado pelo pensamento computacional e a matemática. Em *Anais do XXIV Workshop de Informática na Escola*, páginas 61–69, Porto Alegre, RS, Brasil. SBC.

Tarouco, L. M. R., Roland, L. C., Fabre, M.-C. J. M. e Konrath, M. L. P. (2004). Jogos educacionais. *RENOTE: revista novas tecnologias na educação [recurso eletrônico]. Porto Alegre, RS*.

APÊNDICE A – DIÁLOGOS ENTRE O PERSONAGEM E OS NPCS

Neste apêndice são apresentados os diálogos dos NPCs no jogo educacional "*Infinity Loop*" e a descrição dos cinco desafios. Para facilitar o entendimento, foi utilizado o seguinte formato para representar a opção que o jogador escolhe e a resposta que o NPC reproduz: [Opção do diálogo escolhida] Resposta do NPC.

A.1 NPC TUTORIAL

Diálogo inicial: Olá... Em que posso ser útil?

[Créditos] Este jogo é resultado de um Trabalho de Conclusão de Curso em Ciência da Computação desenvolvido por Henrique Prokopenko e João Pedro Kieras Oliveira, pela Universidade Federal do Paraná. Obrigado por jogar nosso jogo!

[Como jogar?] Neste jogo você é um Mago que deve resolver desafios de programação em linguagem C, explorando os laços de repetição como '*for*' e '*while*'. Sua missão é completar os desafios de programação que encontrar, ajudando os personagens deste mundo a cumprir suas tarefas. Utilize sua magia (código!). A cada erro cometido você recebrá um *feedback* explicando o motivo do erro. Cada desafio que completar permitirá que você avance no mapa. Boa sorte!

A.2 DESAFIO 1

A.2.1 Diálogo com NPC Camponês

Diálogo inicial: Olá... Bem vindo ao seu primeiro desafio!

[Como a ponte quebrou?] Na verdade, está quebrada há anos... Dizem que ela pode ser reconstruida usando o código correto, mas apenas os magos podem decifrar os códigos. Há boatos de que ainda existem escritas antigas com este código, mas o último mago que passou por aqui acabou confundiu as coisas e... bom, quebrou tudo. Você poderia me ajudar a consertar a ponte?

[Sim] Precisamos colocar 7 tábuas nesta ponte em sequência, considerando que a ponte é um vetor, ela deve começar na posição zero. Boa sorte!

[Não] (Fim do diálogo).

A.2.2 Descrição do Desafio 1

Para jogar, clique nos dois primeiros campos não preenchidos do laço for (Linha 6) e selecione uma das opções. Em seguida, clique em "Run Code". Lembre-se de que faltam 7 tábuas em sequência nesta ponte, que está sendo representada como um vetor iniciando na posição zero. Considere que a parte inicial construída da ponte é irrelevante, ou seja, a contagem das posições deve começar após a parte quebrada. Boa Sorte!

A.3 DESAFIO 2

A.3.1 Diálogo com NPC Construtor

Diálogo inicial: Olá... Sou o construtor desta ilha! Fiquei sabendo que tem um novo mago andando por essas terras. É um prazer te conhecer! Estou tentando consertar esta cerca,

mas não consigo de maneira alguma. . . A ajuda de um mago seria ótima. Se você consertá-la, abrirei o portão para você passar. Você poderia me ajudar?

[Sim] Preciso colocar 5 estacas em sequência nesta cerca. Observe que as duas primeiras eu consegui consertar. Boa sorte!

[Não] (Fim do diálogo).

A.3.2 Descrição do Desafio 2

Bem-vindo ao Desafio 2! Neste desafio, você deve selecionar uma opção para a condição do laço while (linha 4) e para o campo da linha 7, sendo este último usado para incrementar ou decrementar o laço. Lembre-se de que você deve adicionar 5 estacas para consertar a cerca. Boa Sorte!

A.4 DESAFIO 3

A.4.1 Diálogo com NPC Lenhadora

Diálogo inicial: Olá... Sou a lenhadora desta ilha. Preciso plantar árvores do lado esquerdo deste caminho para completar a floresta, mas estou tendo dificuldade. Você consegue me ajudar?

[Sim] Preciso plantar 4 árvores, de forma que fiquem alinhadas com as árvores do lado direito, ou seja, nas posições 2, 4, 6 e 8.

[Não] (Fim do diálogo).

A.4.2 Descrição do Desafio 3

Bem vindo ao Desafio 3! Neste desafio, você deve selecionar uma opção para a condição do laço for (linha 4) e um operador relacional na linha 6, para completar a condição da estrutura de decisão if. Você deverá plantar 4 árvores do lado esquerdo do caminho, alinhadas com as árvores do lado direito, para completar a floresta. Boa Sorte!

A.5 DESAFIO 4

A.5.1 Diálogo com NPC Fazendeiro

Diálogo inicial: Olá... Sou o fazendeiro desta ilha. Poderia me ajudar a plantar sementes neste terreno?

[Sim] Preciso plantar trigo e milho neste terreno, mas apenas até que uma das sementes acabe. Além disso, não posso ultrapassar os limites do terreno, ou seja, só posso plantar nessa área retangular.

[Não] (Fim do diálogo).

A.5.2 Descrição do Desafio 4

Bem vindo ao Desafio 4! Neste desafio, você deve completar a condição do laço ‘while’ (linha 6) utilizando operadores lógicos. Sua tarefa é plantar dois tipos de sementes, trigo e milho. Além disso, você deve parar de plantar em duas situações: 1) quando uma das sementes acabar ou 2) quando o terreno estiver totalmente plantado. Boa sorte!

A.6 DESAFIO 5

A.6.1 Diálogo com NPC Jardineiro

Diálogo inicial: Olá... Sou o jardineiro desta ilha e gostaria de plantar flores para deixar meu jardim mais bonito. Você pode me ajudar?

[Sim] Gostaria de plantar as flores de uma forma organizada em uma área 10 x 10: as flores brancas devem ser plantadas na diagonal principal dessa matriz, as flores amarelas devem ocupar o canto superior direito do jardim e as flores vermelhas devem preencher toda a área restante.

[Não] (Fim do diálogo).

A.6.2 Descrição do Desafio 5

Bem vindo ao desafio 5! Neste desafio, você deve completar os campos de condição e iteração do laço ‘for’ nas linhas 4 e 6 (condição e iteração). O objetivo é preencher uma matriz ‘jardim’ de 10 linhas e 10 colunas com três tipos de flores: brancas, vermelhas e amarelas. As flores brancas devem ser plantadas na diagonal principal da matriz. As flores amarelas devem ocupar o canto superior direito da matriz, e as vermelhas devem preencher todas as posições restantes na matriz. Boa sorte!

APÊNDICE B – FEEDBACK PARA OS DESAFIOS

Neste apêndice são apresentados os *feedbacks* para os cinco desafios. Em cada tabela (B.1, B.2, B.3, B.4 e B.5) está descrito o *feedback* correspondente a cada opção.

B.1 DESAFIO 1

Tabela B.1: Opções de resposta para o Desafio 1.

Opção	Inicialização do laço *for*	Condição do laço *for*
1	i = -1	i <= 7
2	i = -1	i <= 6
3	i = -1	i < 7
4	i = 0	i <= 7
5	i = 0	i < 7
6	i = 0	i <= 6

Feedback das Opções 1, 2 e 3: Você tentou acessar a posição -1 do vetor, que é uma posição inválida! Na linguagem C, os vetores sempre começam na posição de índice zero. Tente novamente!

Feedback da Opção 4: Você tentou acessar a posição 7 do vetor, que é uma posição inválida! Na linguagem C, o índice da última posição do vetor é calculado como "tamanho-do-vetor - 1". Tente novamente

Feedback das Opções 5 e 6: Parabéns! Você conseguiu!

B.2 DESAFIO 2

Tabela B.2: Opções de resposta para o Desafio 2.

Opção	Condição do laço *while*	Iteração
1	(i <= 7);	i–
2	(i <= 7)	i–
3	(1 < 8)	i–
4	(i < 8)	i–
5	(i = 8)	i–
6	(i <= 7);	i++
7	(i <= 7)	i++
8	(1 < 8)	i++
9	(i < 8)	i++
10	(i = 8)	i++

Feedback da Opção 1: Erro de laço infinito! Lembre-se que não deve existir ponto e vírgula após a condição do laço while. Além disso, atenção na atualização do valor da variável 'i' na linha 7.

Feedback das Opções 2 e 4: Erro de laço infinito! Atenção na atualização do valor da variável ‘i’. Tente novamente!

Feedback da Opção 3: Erro de laço infinito! A condição (1 < 8) sempre é verdadeira. Além disso, atenção na atualização do valor da variável ‘i’ na linha 7. Tente novamente!

Feedback da Opção 5: Erro de laço infinito! O operador ‘=’ está atribuindo um valor a variável ‘i’ na linha 4. Além disso, atenção na atualização do valor da variável ‘i’ na linha 7. Tente novamente!

Feedback da Opção 6: Erro de laço infinito! Lembre-se que não deve existir ponto e vírgula após a condição do laço while. Tente novamente!

Feedback das Opções 7 e 9: Parabéns! Você conseguiu!

Feedback da Opção 8: Erro de laço infinito! A condição (1 < 8) sempre é verdadeira. Tente novamente!

Feedback da Opção 10: Erro de laço infinito! O operador ‘=’ está atribuindo um valor a variável ‘i’ na linha 4. Tente novamente!

B.3 DESAFIO 3

Tabela B.3: Opções de resposta para o Desafio 3.

Opção	Condição do laço *for*	Operador relacional do *if*
1	i < tamfloresta	≥
2	i <= tamfloresta	≥
3	i < (tamfloresta - 1)	≥
4	i <= (tamfloresta - 1)	≥
5	i < tamfloresta	<
6	i <= tamfloresta	<
7	i < (tamfloresta - 1)	<
8	i <= (tamfloresta - 1)	<
9	i < tamfloresta	==
10	i <= tamfloresta	==
11	i < (tamfloresta - 1)	==
12	i <= (tamfloresta - 1)	==
13	i < tamfloresta	!=
14	i <= tamfloresta	!=
15	i < (tamfloresta - 1)	!=
16	i <= (tamfloresta - 1)	!=
17	i < tamfloresta	≤
18	i <= tamfloresta	≤
19	i < (tamfloresta - 1)	≤
20	i <= (tamfloresta - 1)	≤

Feedback das Opções 1 e 4: Uso incorreto do operador relacional na condição do ‘if’. Você plantou 7 árvores. Tente novamente!

Feedback da Opção 2: Uso incorreto do operador relacional na condição do ‘if’. Você plantou 8 árvores. Tente novamente!

Feedback da Opção 3: Uso incorreto do operador relacional na condição do ‘if’. Você plantou 6 árvores. Tente novamente!

Feedback das Opções 5, 6, 7 e 8: Uso incorreto do operador relacional na condição do ‘if’. Nenhuma árvore foi plantada. Tente novamente!

Feedback das Opções 9, 11, 12, 17, 19 e 20: Uso incorreto do operador relacional na condição do ‘if’. Você plantou apenas 3 árvores. Tente novamente!

Feedback das Opções 10 e 18: Parabéns! Você completou o desafio!

Feedback das Opções 13, 14, 15 e 16: Uso incorreto do operador relacional na condição do ‘if’. Você plantou 4 árvores nas posições incorretas. Tente novamente!

B.4 DESAFIO 4

Tabela B.4: Opções de resposta para o Desafio 4.

Opção	Primeiro operador	Segundo operador
1	||	||
2	||	&&
3	&&	||
4	&&	&&

Feedback das Opções 1 e 3: Uso incorreto do operador lógico. Você tem 5 sementes de milho e 7 sementes de trigo e está tentando plantar 8 de cada. Tente novamente!

Feedback da Opção 2: Uso incorreto do operador lógico. Você tem 5 sementes de milho e está tentando plantar 7. Tente novamente!

Feedback da Opção 4: Parabéns! Você conseguiu!

B.5 DESAFIO 5

Tabela B.5: Opções de resposta para o Desafio 5.

Opção	Condição do primeiro laço *for*	Condição do segundo laço *for*	Condição do primeiro comando *if*	Condição do segundo comando *if*	Condição do terceiro comando *if*
1	i < 10	j <= 10	i == j	i == j	i == j
2	i < 10	j <= 10	i == j	i == j	i > j
3	i < 10	j <= 10	i == j	i == j	i < j
4	i < 10	j <= 10	i == j	i > j	i == j
5	i < 10	j <= 10	i == j	i > j	i > j
6	i < 10	j <= 10	i == j	i > j	i < j
7	i < 10	j <= 10	i == j	i < j	i == j
9	i < 10	j <= 10	i == j	i < j	i > j
9	i < 10	j <= 10	i == j	i < j	i < j
10	i < 10	j <= 10	i > j	i == j	i == j
11	i < 10	j <= 10	i > j	i == j	i > j
12	i < 10	j <= 10	i > j	i == j	i < j
13	i < 10	j <= 10	i > j	i > j	i == j
14	i < 10	j <= 10	i > j	i > j	i > j
15	i < 10	j <= 10	i > j	i > j	i < j
16	i < 10	j <= 10	i > j	i < j	i == j
17	i < 10	j <= 10	i > j	i < j	i > j
18	i < 10	j <= 10	i > j	i < j	i < j
19	i < 10	j <= 10	i < j	i == j	i == j
20	i < 10	j <= 10	i < j	i == j	i > j
21	i < 10	j <= 10	i < j	i == j	i < j
22	i < 10	j <= 10	i < j	i > j	i == j
23	i < 10	j <= 10	i < j	i > j	i > j
24	i < 10	j <= 10	i < j	i > j	i < j
25	i < 10	j <= 10	i < j	i < j	i == j
26	i < 10	j <= 10	i < j	i < j	i > j
27	i < 10	j <= 10	i < j	i < j	i < j

Continua...

Opção	Condição do primeiro laço *for*	Condição do segundo laço *for*	Condição do primeiro comando *if*	Condição do segundo comando *if*	Condição do terceiro comando *if*
28	i < 10	i < 10	i == j	i == j	i == j
29	i < 10	i < 10	i == j	i == j	i > j
30	i < 10	i < 10	i == j	i == j	i < j
31	i < 10	i < 10	i == j	i > j	i == j
32	i < 10	i < 10	i == j	i > j	i > j
33	i < 10	i < 10	i == j	i > j	i < j
34	i < 10	i < 10	i == j	i < j	i == j
35	i < 10	i < 10	i == j	i < j	i > j
36	i < 10	i < 10	i == j	i < j	i < j
37	i < 10	i < 10	i > j	i == j	i == j
38	i < 10	i < 10	i > j	i == j	i > j
39	i < 10	i < 10	i > j	i == j	i < j
40	i < 10	i < 10	i > j	i > j	i == j
41	i < 10	i < 10	i > j	i > j	i > j
42	i < 10	i < 10	i > j	i > j	i < j
43	i < 10	i < 10	i > j	i < j	i == j
44	i < 10	i < 10	i > j	i < j	i > j
45	i < 10	i < 10	i > j	i < j	i < j
46	i < 10	i < 10	i < j	i == j	i == j
47	i < 10	i < 10	i < j	i == j	i > j
48	i < 10	i < 10	i < j	i == j	i < j
49	i < 10	i < 10	i < j	i > j	i == j
50	i < 10	i < 10	i < j	i > j	i > j
51	i < 10	i < 10	i < j	i > j	i < j
52	i < 10	i < 10	i < j	i < j	i == j
53	i < 10	i < 10	i < j	i < j	i > j
54	i < 10	i < 10	i < j	i < j	i < j

Continua...

Opção	Condição do primeiro laço *for*	Condição do segundo laço *for*	Condição do primeiro comando *if*	Condição do segundo comando *if*	Condição do terceiro comando *if*
55	i < 10	j < 10	i == j	i == j	i == j
56	i < 10	j < 10	i == j	i == j	i > j
57	i < 10	j < 10	i == j	i == j	i < j
58	i < 10	j < 10	i == j	i > j	i == j
59	i < 10	j < 10	i == j	i > j	i > j
60	i < 10	j < 10	i == j	i > j	i < j
61	i < 10	j < 10	i == j	i < j	i == j
62	i < 10	j < 10	i == j	i < j	i > j
63	i < 10	j < 10	i == j	i < j	i < j
64	i < 10	j < 10	i > j	i == j	i == j
65	i < 10	j < 10	i > j	i == j	i > j
66	i < 10	j < 10	i > j	i == j	i < j
67	i < 10	j < 10	i > j	i > j	i == j
68	i < 10	j < 10	i > j	i > j	i > j
69	i < 10	j < 10	i > j	i > j	i < j
70	i < 10	j < 10	i > j	i < j	i == j
71	i < 10	j < 10	i > j	i < j	i > j
72	i < 10	j < 10	i > j	i < j	i < j
73	i < 10	j < 10	i < j	i == j	i == j
74	i < 10	j < 10	i < j	i == j	i > j
75	i < 10	j < 10	i < j	i == j	i < j
76	i < 10	j < 10	i < j	i > j	i == j
77	i < 10	j < 10	i < j	i > j	i > j
78	i < 10	j < 10	i < j	i > j	i < j
79	i < 10	j < 10	i < j	i < j	i == j
80	i < 10	j < 10	i < j	i < j	i > j
81	i < 10	j < 10	i < j	i < j	i < j

Continua...

Opção	Condição do primeiro laço *for*	Condição do segundo laço *for*	Condição do primeiro comando *if*	Condição do segundo comando *if*	Condição do terceiro comando *if*
82	i <= 10	j <= 10	i == j	i == j	i == j
83	i <= 10	j <= 10	i == j	i == j	i > j
84	i <= 10	j <= 10	i == j	i == j	i < j
85	i <= 10	j <= 10	i == j	i > j	i == j
86	i <= 10	j <= 10	i == j	i > j	i > j
87	i <= 10	j <= 10	i == j	i > j	i < j
88	i <= 10	j <= 10	i == j	i < j	i == j
89	i <= 10	j <= 10	i == j	i < j	i > j
90	i <= 10	j <= 10	i == j	i < j	i < j
91	i <= 10	j <= 10	i > j	i == j	i == j
92	i <= 10	j <= 10	i > j	i == j	i > j
93	i <= 10	j <= 10	i > j	i == j	i < j
94	i <= 10	j <= 10	i > j	i > j	i == j
95	i <= 10	j <= 10	i > j	i > j	i > j
96	i <= 10	j <= 10	i > j	i > j	i < j
97	i <= 10	j <= 10	i > j	i < j	i == j
98	i <= 10	j <= 10	i > j	i < j	i > j
99	i <= 10	j <= 10	i > j	i < j	i < j
100	i <= 10	j <= 10	i < j	i == j	i == j
101	i <= 10	j <= 10	i < j	i == j	i > j
102	i <= 10	j <= 10	i < j	i == j	i < j
103	i <= 10	j <= 10	i < j	i > j	i == j
104	i <= 10	j <= 10	i < j	i > j	i > j
105	i <= 10	j <= 10	i < j	i > j	i < j
106	i <= 10	j <= 10	i < j	i < j	i == j
107	i <= 10	j <= 10	i < j	i < j	i > j
108	i <= 10	j <= 10	i < j	i < j	i < j

Continua...

Opção	Condição do primeiro laço *for*	Condição do segundo laço *for*	Condição do primeiro comando *if*	Condição do segundo comando *if*	Condição do terceiro comando *if*
109	i <= 10	i < 10	i == j	i == j	i == j
110	i <= 10	i < 10	i == j	i == j	i > j
111	i <= 10	i < 10	i == j	i == j	i < j
112	i <= 10	i < 10	i == j	i > j	i == j
113	i <= 10	i < 10	i == j	i > j	i > j
114	i <= 10	i < 10	i == j	i > j	i < j
115	i <= 10	i < 10	i == j	i < j	i == j
116	i <= 10	i < 10	i == j	i < j	i > j
117	i <= 10	i < 10	i == j	i < j	i < j
118	i <= 10	i < 10	i > j	i == j	i == j
119	i <= 10	i < 10	i > j	i == j	i > j
120	i <= 10	i < 10	i > j	i == j	i < j
121	i <= 10	i < 10	i > j	i > j	i == j
122	i <= 10	i < 10	i > j	i > j	i > j
123	i <= 10	i < 10	i > j	i > j	i < j
124	i <= 10	i < 10	i > j	i < j	i == j
125	i <= 10	i < 10	i > j	i < j	i > j
126	i <= 10	i < 10	i > j	i < j	i < j
127	i <= 10	i < 10	i < j	i == j	i == j
128	i <= 10	i < 10	i < j	i == j	i > j
129	i <= 10	i < 10	i < j	i == j	i < j
130	i <= 10	i < 10	i < j	i > j	i == j
131	i <= 10	i < 10	i < j	i > j	i > j
132	i <= 10	i < 10	i < j	i > j	i < j
133	i <= 10	i < 10	i < j	i < j	i == j
134	i <= 10	i < 10	i < j	i < j	i > j
135	i <= 10	i < 10	i < j	i < j	i < j

Continua...

Opção	Condição do primeiro laço *for*	Condição do segundo laço *for*	Condição do primeiro comando *if*	Condição do segundo comando *if*	Condição do terceiro comando *if*
136	i <= 10	j < 10	i == j	i == j	i == j
137	i <= 10	j < 10	i == j	i == j	i > j
138	i <= 10	j < 10	i == j	i == j	i < j
139	i <= 10	j < 10	i == j	i > j	i == j
140	i <= 10	j < 10	i == j	i > j	i > j
141	i <= 10	j < 10	i == j	i > j	i < j
142	i <= 10	j < 10	i == j	i < j	i == j
143	i <= 10	j < 10	i == j	i < j	i > j
144	i <= 10	j < 10	i == j	i < j	i < j
145	i <= 10	j < 10	i > j	i == j	i == j
146	i <= 10	j < 10	i > j	i == j	i > j
147	i <= 10	j < 10	i > j	i == j	i < j
148	i <= 10	j < 10	i > j	i > j	i == j
149	i <= 10	j < 10	i > j	i > j	i > j
150	i <= 10	j < 10	i > j	i > j	i < j
151	i <= 10	j < 10	i > j	i < j	i == j
152	i <= 10	j < 10	i > j	i < j	i > j
153	i <= 10	j < 10	i > j	i < j	i < j
154	i <= 10	j < 10	i < j	i == j	i == j
155	i <= 10	j < 10	i < j	i == j	i > j
156	i <= 10	j < 10	i < j	i == j	i < j
157	i <= 10	j < 10	i < j	i > j	i == j
158	i <= 10	j < 10	i < j	i > j	i > j
159	i <= 10	j < 10	i < j	i > j	i < j
160	i <= 10	j < 10	i < j	i < j	i == j
161	i <= 10	j < 10	i < j	i < j	i > j
162	i <= 10	j < 10	i < j	i < j	i < j

Feedback da Opção 60: Parabéns! Você conseguiu!
Feedback das opções 28 a 54 e 109 a 135: Loop infinito!
Feedback das opções 2 a 27 e 82 a 108 : Posições inválidas da matriz!
Feedback das demais opções: Uso incorreto de operadores relacionais!