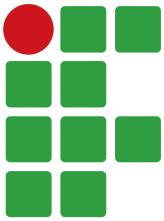

INSTITUTO FEDERAL
Goiás
Câmpus Anápolis

David Lucas Souza Andrade

Jogo Educacional para Conceitos de Estruturação de Algoritmos: Code Blocks

Anápolis-GO

2023

David Lucas Souza Andrade

Jogo Educacional para Conceitos de Estruturação de Algoritmos: Code Blocks

Trabalho de Conclusão de Curso apresentado ao Instituto Federal de Goiás como parte das exigências para obtenção do título de Bacharel em Ciência da Computação.

Instituto Federal de Goiás - Câmpus Anápolis

Orientador: Prof. Alexandre Bellezi José

Coorientador: Diego Santos Leão

Anápolis-GO

2023

Dados Internacionais de Catalogação na Publicação (CIP)

Andrade, David Lucas Souza

A554j Jogo educacional para conceitos de estruturação de algorítimos: *code blocks.* / David Lucas Souza Andrade. – Anápolis: IFG, 2023.
50 f. il. color.

Orientador: Prof. Me. Alexandre Bellezi José.

Trabalho de Conclusão de Curso – Instituto Federal de Educação, Ciência e Tecnologia de Goiás: Campus Anápolis – Bacharelado em Ciência da Computação, 2023.

1. Softwares educacionais. 2. algorítimos. 3. jogos eletrônicos.
I. José, Alexandre Bellezi (orient.).
II. Leão, Diego Santos (coorient.).
III. Título.

CDD 005.1

Ficha catalográfica elaborada pela bibliotecária Claudineia Pereira de Abreu CRB1/1956
IFG Campus Anápolis

ATA DA SESSÃO PÚBLICA DE APRESENTAÇÃO E DEFESA DO TRABALHO DE CONCLUSÃO DE CURSO

No 6º dia do mês de julho do ano de dois mil e vinte e três, às 14 horas e 3 minuto, foi realizada a sessão pública de apresentação e defesa do Trabalho de Conclusão de Curso do Graduando **David Lucas Souza Andrade** (matrícula 20191060140193) do curso de BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO. A banca foi composta pelos seguintes membros: Alexandre Bellezi José, Daniel Xavier de Sousa e Alessandro Rodrigues e Silva sob a presidência do primeiro. O trabalho de conclusão de curso tem como título**"Jogo Educacional para Conceitos de Estruturação de Algoritmos"**, da área de Engenharia de Software, sob orientação da Prof. Alexandre Bellezi José. Após a apresentação, tendo sido o autor arguido pela Banca Examinadora, a nota obtida foi 9,5 (nove e meio) e o Trabalho de Conclusão de Curso foi considerado **aprovado com correções** pela banca examinadora.

Encerra-se a presente sessão às 15 horas e 11 minutos. Eu, Prof. Alexandre Bellezi José, dato e assino a presente ata que segue assinada por todos os membros da Banca e pelo graduando.

(*Assinado eletronicamente*)

Ma. Alexandre Bellezi José

Prof. IFG - Câmpus Anápolis

(*Assinado eletronicamente*)

Dr. Daniel Xavier Sousa

Prof. IFG - Câmpus Anápolis

(*Assinado eletronicamente*)

Dr. Alessandro Rodrigues e Silva

Prof. IFG - Câmpus Anápolis

(*Assinado eletronicamente*)

David Lucas Souza Andrade

Orientando

Documento assinado eletronicamente por:

- **David Lucas Souza Andrade**, **20191060140193 - Discente**, em 04/08/2023 13:00:29.
- **Daniel Xavier de Sousa**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 14/07/2023 09:26:33.
- **Alessandro Rodrigues e Silva**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 14/07/2023 09:18:13.
- **Alexandre Bellezi Jose**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 14/07/2023 09:04:35.

Este documento foi emitido pelo SUAP em 06/07/2023. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse https://suap.ifg.edu.br/autenticar-documento/ e forneça os dados abaixo:

Código Verificador: 428253
Código de Autenticação: b3846e167d

TERMO DE AUTORIZAÇÃO PARA DISPONIBILIZAÇÃO NO REPOSITÓRIO DIGITAL DO IFG - ReDi IFG

Com base no disposto na Lei Federal nº 9.610/98, AUTORIZO o Instituto Federal de Educação, Ciência e Tecnologia de Goiás, a disponibilizar gratuitamente o documento no Repositório Digital (ReDi IFG), sem ressarcimento de direitos autorais, conforme permissão assinada abaixo, em formato digital para fins de leitura, download e impressão, a título de divulgação da produção técnico-científica no IFG.

Identificação da Produção Técnico-Científica

[ ]	Tese	[ ]	Artigo Científico
[ ]	Dissertação	[ ]	Capítulo de Livro
[ ]	Monografia – Especialização	[ ]	Livro
[X]	TCC - Graduação	[ ]	Trabalho Apresentado em Evento
[ ]	Produto Técnico e Educacional - Tipo: _____		

Nome Completo do Autor:
Matrícula:
Título do Trabalho:

Autorização - Marque uma das opções

1. (X) Autorizo disponibilizar meu trabalho no Repositório Digital do IFG (acesso aberto);
2. ( ) Autorizo disponibilizar meu trabalho no Repositório Digital do IFG somente após a data ___/___/_____ (Embargo);
3. ( ) Não autorizo disponibilizar meu trabalho no Repositório Digital do IFG (acesso restrito).

Ao indicar a opção **2 ou 3**, marque a justificativa:

( ) O documento está sujeito a registro de patente.
( ) O documento pode vir a ser publicado como livro, capítulo de livro ou artigo.
( ) Outra justificativa: _____

DECLARAÇÃO DE DISTRIBUIÇÃO NÃO-EXCLUSIVA

O/A referido/a autor/a declara que:

i. o documento é seu trabalho original, detém os direitos autorais da produção técnico-científica e não infringe os direitos de qualquer outra pessoa ou entidade;
ii. obteve autorização de quaisquer materiais inclusos no documento do qual não detém os direitos de autor/a, para conceder ao Instituto Federal de Educação, Ciência e Tecnologia de Goiás os direitos requeridos e que este material cujos direitos autorais são de terceiros, estão claramente identificados e reconhecidos no texto ou conteúdo do documento entregue;
iii. cumpriu quaisquer obrigações exigidas por contrato ou acordo, caso o documento entregue seja baseado em trabalho financiado ou apoiado por outra instituição que não o Instituto Federal de Educação, Ciência e Tecnologia de Goiás.

Anápolis, 16/08/2023.
Local Data

_____
Assinatura do Autor e/ou Detentor dos Direitos Autorais

Página de assinaturas

David Andrade
702.881.151-47
Signatário

HISTÓRICO

16 ago 2023 18:14:14 — **David Lucas Souza Andrade** criou este documento. (E-mail: laykabuss@gmail.com, CPF: 702.881.151-47)

16 ago 2023 18:14:15 — **David Lucas Souza Andrade** *(E-mail: laykabuss@gmail.com, CPF: 702.881.151-47)* visualizou este documento por meio do IP 201.41.91.60 localizado em Goiânia - Goias - Brazil

16 ago 2023 18:14:18 — **David Lucas Souza Andrade** *(E-mail: laykabuss@gmail.com, CPF: 702.881.151-47)* assinou este documento por meio do IP 201.41.91.60 localizado em Goiânia - Goias - Brazil

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

MIT	Massachusetts Institute of Technology
GUI	Graphical User Interface
IDE	Integrated Development Environment
PC	Personal Computer
PIBITI	Programa Institucional de Bolsas de Iniciação em Desenvolvimento Tecnológico e Inovação
IFG	Instituto Federal de Educação, Ciência e Tecnologia De Goiás Instituto Federal

Sumário

1 Introdução

Os alunos que ingressam em cursos de graduação relacionados a Computação frequentemente enfrentam desafios significativos, especialmente nas disciplinas que envolvem o ensino de algoritmos e cálculos matemáticos (HOED, 2016). De acordo com Paula (PAULA; JÚNIOR; FREITAS, 2009), para alunos que estudam Computação, "é uma tarefa árdua construir representações mentais que realmente abstraiam completamente um problema, o que exige a adoção de abordagens que estimulem o desenvolvimento dessa capacidade".

Diante dessas dificuldades enfrentadas pelos alunos, torna-se fundamental compreender os motivos que causam evasão nos estágios iniciais dos cursos. A evasão é influenciada por fatores institucionais, socioeconômicos e vocacionais (HOED, 2016). No contexto institucional, em particular, podem surgir desafios relacionados às disciplinas de algoritmos e programação, à compreensão abstrata e às estratégias didáticas em sala de aula (HOED, 2016).

Vendo que a lógica é um dos pilares para a construção do entendimento dos algoritmos e que é de difícil compreensão dos alunos, uma possível abordagem é de que o professor possa prover ferramentas que auxiliem o aluno nesse processo de aprendizado. Sites como o "Desafio do Código" (CÓDIGO, 2022). e "*Blockly Games*" (GAMES, 2022a), além de *software* educativos como *Scratch* (SCRATCH, 2022), *Robocode* (ROBOCODE, 2022) e *Visualg* (VISUALG, 2022), são exemplos de meios lúdicos e interativos que complementam os estudos a fim de favorecer a metodologia aplicada em sala de aula. Esses *software* educativos podem ser ótimos para o aprendizado introdutório de Lógica de Programação (VIANA; PORTELA, 2019), pois apresentam objetivos pedagógicos bem estabelecidos para que sejam adequadamente inseridos no contexto do ensino e aprendizagem (MOTA et al., 2014).

Para Piaget, (FERRACIOLI, 1999) o desenvolvimento do pensamento simbólico e pré-conceitual é de suma importância para a estruturação futura do conhecimento, pois, partindo de tais símbolos, o indivíduo pode realizar operações que colaboram para a organização do pensamento. Deste modo, podemos explorar no jogo aqui proposto um conjunto de símbolos, para permitir que o usuário seja conduzido a explorar as possibilidades de operações nos desafios propostos pelo jogo. Essa exploração pode ser utilizada para iniciar seu processo de aprendizagem, pois de acordo com Piaget a aprendizagem não é espontânea e deve ser instigada por um professor; tendo isso em vista, Seymour Papert (SILVEIRA, 2016) propôs trabalhos que servem de ferramenta intermediária na educação, como a linguagem de programação *LOGO* (SOLOMON et al., 2020).

Nesse contexto, o presente trabalho teve como propósito a elaboração e implementação de um *software* do tipo jogo digital, com o objetivo de auxiliar os alunos que estão estudando algoritmos, priorizando a estruturação dos comandos utilizando a linguagem Portugol. Essa linguagem também é utilizada como ferramenta de apoio ao ensino de lógica de programação, visto que, por ser uma pseudo-linguagem com comandos em português, facilita o aprendizado, tornando-se algo mais natural para os iniciantes (MANSO; OLIVEIRA; MARQUES, 2009). A ideia central foi desenvolver um jogo interativo, o *Code Blocks*(ANDRADE, 2023a), que faça uso da pseudo-linguagem, baseada em Portugol, e que também incluísse blocos adaptados ao cenário de lógica de programação. Acreditamos que essa ferramenta possa contribuir para minimizar a evasão e a reprovação nas disciplinas de programação de computadores, além de proporcionar um primeiro acesso ao mundo da lógica de programação, despertando maior interesse dos alunos nessa área de estudo.

1.1 Objetivos

1.1.1 Objetivos gerais

1. Desenvolver uma aplicação que sirva de metodologia suplementar ao ensino comum de estruturação de algoritmos;
2. Disponibilizar aos alunos iniciantes em disciplinas de algoritmos um incentivo lúdico para assimilar o conceito da estrutura de um algoritmo;
3. Projetar com base nas ideias de Seymour Papert (HAREL; PAPERT, 1991), uma estrutura que permitirá ao usuário testar e experimentar a integração de ações possíveis com resolução de problemas.

1.1.2 Objetivos específicos

1. Descrever os conceitos de aprendizado de Seymour Papert para aplicação no *software*;
2. Fazer pesquisas e análises de trabalhos semelhantes;
3. Descrever a estruturação de um algoritmo e sua importância;
4. Criar um modelo conceitual do sistema;
5. Desenvolver o *software*, utilizando os conceitos de aprendizado;
6. Projetar diferentes fases com puzzles que incentivem o uso da lógica junto aos conceitos da construção de algoritmos que serão apresentados;
7. Publicar a aplicação pelo *itch.io* (CORP, 2023).

2 Referencial Teórico

2.1 Algoritmos

Um dos pontos fundamentais na base do ensino de computação é o funcionamento de um algoritmo. Um algoritmo é definido como uma sequência de operações ou comandos que possui um determinado objetivo, é executada uma quantidade finita de vezes, e contém entrada e saída de dados (MEDINA; FERTING, 2006). Com base nessa definição, podemos entender um algoritmo como as regras a se seguir para cumprir uma determinada tarefa; um bom exemplo é uma receita de bolo, que contém ingredientes (dados de entrada) e um passo-a-passo (instruções), e tem como objetivo o bolo pronto. Sendo que os algoritmos serão utilizados para apresentar os conceitos de programação nas fases do jogo, como será descrito na seção 3.3.

2.2 Software

O objetivo principal desse artigo é apresentar um *software* que permite facilitar o aprendizado da estruturação de um algoritmo. *Software* é um conjunto de instruções executadas em uma sequência que serão interpretadas pelo computador a fim de executar determinadas tarefas, podendo ser um programa, instruções ou dados que vão comandar os mais diversos dispositivos, como computadores, *smartphones*, dentre outros eletrônicos (CARVALHO; LORENA, 2017).

Alguns desses *software* são aplicativos que executam diversas tarefas nos dispositivos, como navegar na internet, realizar cálculos em uma calculadora e acessar meios de comunicação, como *Gmail* e *WhatsApp*. Por outro lado, *software* de jogos são, em geral, desenvolvidos com propósito recreativo; no entanto, também podem ser utilizados como ferramentas intermediárias para fins educacionais (CARVALHO; LORENA, 2017), que é a finalidade deste trabalho.

2.3 Teorias do aprendizado

O campo dos estudos é uma das áreas em que é possível a ação da tecnologia. Tendo em vista que o aprendizado é de suma importância para a evolução da sociedade, diversos pesquisadores já investiram seus estudos na exploração e criação de suas próprias teorias de aprendizagem, e buscam fundamentar a dinâmica de ensinar e aprender com base na capacidade cognitiva do ser humano (PRÄSS, 2012). É com base nos trabalhos de Jean Piaget e Seymour Papert que esse trabalho desenvolve sua argumentação teórica.

A teoria de Piaget não é sobre a aprendizagem em si e sim um trabalho epistemológico quanto ao desenvolvimento cognitivo, tendo como principais pilares dessa teoria a assimilação, acomodação e equilíbrio do conhecimento (FERRACIOLI, 1999). Levando em conta esses conceitos, quando há assimilação a mente não se modifica, mas quando o ser não consegue assimilar certa situação, há dois possíveis caminhos: em um, a mente desiste, e em outro ela se modifica e com isso ela se acomoda, conduzindo assim à construção de novos esquemas de assimilação, que resultam no desenvolvimento cognitivo (MOREIRA, 1999).

Deve-se observar que o mecanismo de aprender é a sua capacidade de se reestruturar mentalmente em busca de um novo equilíbrio. Logo, o ensino deve ativar esse mecanismo por meio de atividades desafiadoras, que provoquem consecutivas quebras de equilíbrio para ser possível haver reequilíbrio. O papel do professor, então, é de propor situações que sejam equivalentes com o nível de desenvolvimento cognitivo do aluno, com atividades que sejam adequadamente desafiadoras, e que não forcem um grande desequilíbrio para o aluno (MOREIRA, 1999).

O início de projetos que envolvem a computação e a educação se dá por Seymour Papert, que foi um educador, cientista da computação e matemático. Seu trabalho de pesquisa foi realizado no *Massachusetts Institute of Technology* (MIT) com um grupo de sua criação "*Epistemology and Learning Research Group*" e teve como base a epistemologia do construtivismo de Jean Piaget para desenvolver sua própria teoria de aprendizagem, conhecida como construcionismo (SILVEIRA, 2016).

O Construcionismo é centrado no aluno, no qual este adquire novos conhecimentos por meio da prática, aplicando o que já sabe em projetos relacionados diretamente com o mundo real. Isso possibilita que suas experiências passadas sirvam como base para a construção de novas ideias. Com base nesse princípio, o conjunto de fases é elaborado, levando em consideração os conhecimentos previamente adquiridos nas etapas propostas.

2.4 Software educacionais

Com base no Construcionismo, Papert aplicou sua teoria voltada para a computação na criação da linguagem de programação *LOGO* (SOLOMON et al., 2020). Esta linguagem tem como intuito ser uma ferramenta que ajude as crianças a pensar e resolver problemas. No programa os alunos utilizam de comandos que controlam o cursor, uma "tartaruga", que ajuda as crianças a se colocarem no lugar e raciocinar o que fariam se fossem ela, para criar desenhos gráficos de linha ou vetoriais, como pode ser observado na Figura 1.

Assim se deu o desenvolvimento de *software* que fossem capazes de auxiliar o professor a aplicar novas metodologias para facilitar o aprendizado dos alunos. Dentre eles estão os jogos educacionais, que são capazes de potencializar como se experimenta e

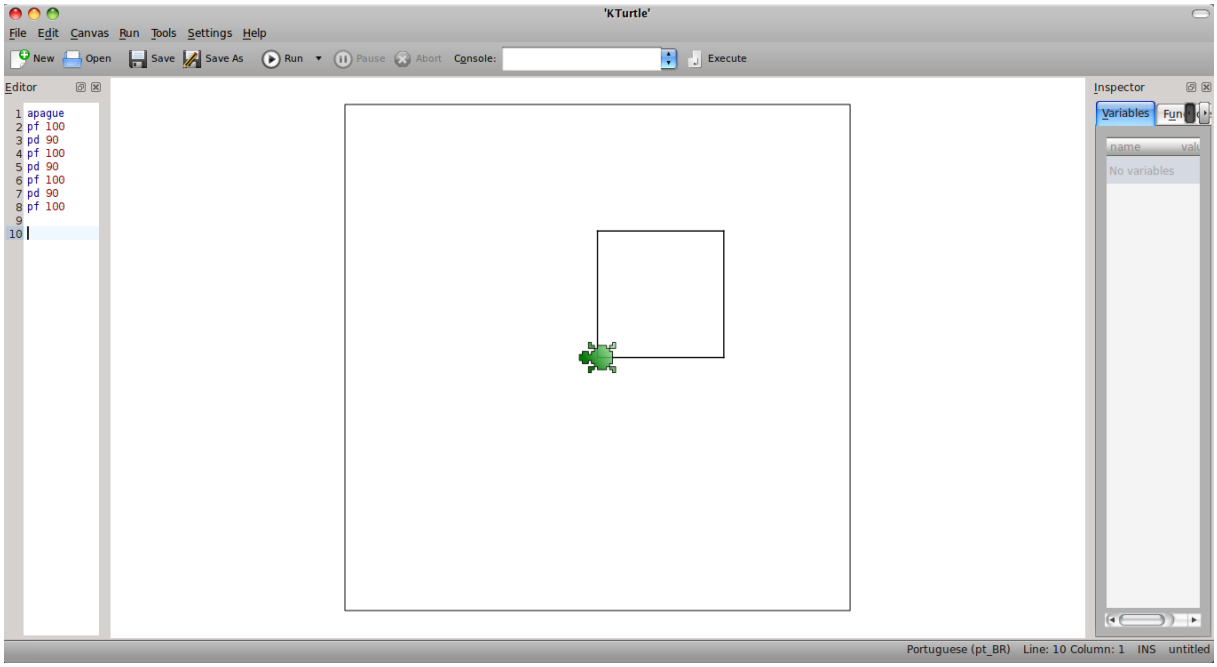

Figura 1 – Interface da linguagem de programação Logo
Fonte: (KLEINSORGE, 2022)

visualiza os conceitos, além de serem um ambiente mais propenso a estimular a criatividade e o interesse dos alunos (GRAMIGNA, 2022).

2.5 Portugol

A seleção de uma linguagem de programação que facilite e se aproxime do aluno revelou-se de extrema importância para este trabalho, pois buscou-se integrar o ensino de programação com a abordagem lúdica característica dos jogos.

A introdução ao aprendizado de programação é de grande importância, pois representa a base dos cursos na área de computação. Para auxiliar nesse processo de aprendizagem, foram desenvolvidas diversas ferramentas educacionais, incluindo o *Portugol Studio* (MANSO; OLIVEIRA; MARQUES, 2009). Trata-se de uma *Integrated Development Environment* (IDE) voltada para uso didático, baseada no idioma português, o que possibilita a inclusão de estudantes com pouca fluência no inglês, visto que muitas linguagens de programação utilizam termos em língua estrangeira. Essa ferramenta oferece vários recursos comuns a IDEs profissionais, mas sempre com o foco nos elementos didáticos para a aprendizagem em programação.

2.6 Puzzle

Nesta seção, será abordado o gênero de quebra-cabeça, denominado *puzzle*, sendo o gênero no qual o jogo proposto neste trabalho está inserido. Esse gênero incita problemas de execução mental que exigem raciocínio logico para alcançar a resolução de certo problema, por meio de artifícios lúdicos. Os jogos de *puzzle* são um ambiente para o desenvolvimento de problemas de forma lúdica, e neste trabalho foram utilizados para fins didáticos.

Esses problemas focam em desafios mentais que podem assumir diferentes formas, como o desenvolvimento de raciocínio com complexidade progressiva com base em *puzzles* anteriores; resolução de *puzzles* com restrição de espaço e tempo; aplicação de conhecimentos anteriores conhecidos sobre o mundo; reconhecimento de objetos que possam ser categorizados por padrões, formatos, cores, etc. Os jogos desse gênero têm o objetivo em comum de estimular o raciocínio nas mais diversas situações, assim como o trabalho aqui proposto.

Além dessas formas de desafios há também aqueles problemas com um ou mais caminhos para a solução. Essa diferenciação pode ser nomeada como linearidade; assim, *puzzles* lineares são aqueles que têm apenas uma solução possível: no jogo *Gris* (STUDIO, 2022a), por exemplo, o jogador obtém com o passar do jogo um conjunto de habilidades para usar, mas é função do jogador descobrir como irá alcançar seu objetivo em um caminho projetado para incentivar o uso dessas habilidades. Por outro lado, há também os *puzzles* não-lineares, que contém mais de uma solução possível: no jogo *Outer Wilds* (MOBIUS, 2022), o jogador inicia com um conjunto de ferramentas para explorar o mundo aberto, com diversos *puzzles* espalhados, mas sem necessidade de seguir determinada ordem para concluir. Sendo que, o presente trabalho se enquadra na categoria de *puzzles* lineares, como será descrito na seção 4.1.

2.7 Subgênero de puzzle

Há uma grande variedade de jogos do tipo *puzzle*, sendo que dentre eles diversos subgêneros são explorados. Podemos citar o de exploração ou indução por tentativa-e-erro, em que, no geral, os desafios de raciocínio apresentados têm como base a interação com o mundo ao redor do *puzzle*, seja por mecânica própria ou através de objetos, sendo necessário verificar e explorar o ambiente proposto para ter um melhor entendimento do que é para ser feito, como o próprio *Outer Wilds* comentado na seção 2.6.

Existe também jogos de *puzzle* com subgênero híbrido, contendo elementos de outros gêneros, como, por exemplo, Ação e Aventura (*SuperHot* (SUPERHOT, 2022)), plataforma (*Braid* (THEKLA, 2022)), ou jogos de cartas (*Dicey Dungeons* (CAVANAGH, 2022)). A exemplo disso, temos um híbrido de combinação de peças com estilo arcade

(*Tile-matching arcade-style game*), que foca na habilidade de raciocínio para combinar um conjunto de peças, de acordo com cor, forma ou outros fatores, podendo considerar variáveis temporais como limite de tempo ou velocidade das peças, sendo o caso do jogo *Tetris* (COSTA, 2022). No contexto deste trabalho, o subgênero de combinação de peças com estilo arcade é utilizado como base para a concepção do jogo proposto.

2.8 Tetris

O jogo *Tetris* surgiu em 1984 na antiga União Soviética, criado pelo cientista russo Alexey Pajitnov. Esse cientista tinha como objetivo criar um jogo baseado em *Pentominoes*, um jogo de tabuleiro de origem grega; suas peças eram compostas por cinco quadrados, e isso teve influência no nome *Tetris*, visto que suas peças contêm quatro quadrados cada (PLANK-BLASKO, 2015).

A ideia deste jogo é ser dinâmico, contendo uma corrida contra o tempo sem final definido. Para isso é necessário que o jogador fosse enfrentando um desafio cada vez maior, gerado pelo incremento da velocidade que as peças caem, o que causa uma maior dificuldade em completar as linhas, como pode ser visto na Figura 2. Hoje o *Tetris* é o jogo mais vendido de todos os tempos (MUNDO, 2023) com mais 520 milhões copias distribuídas, influenciando assim a criação de diversos jogos.

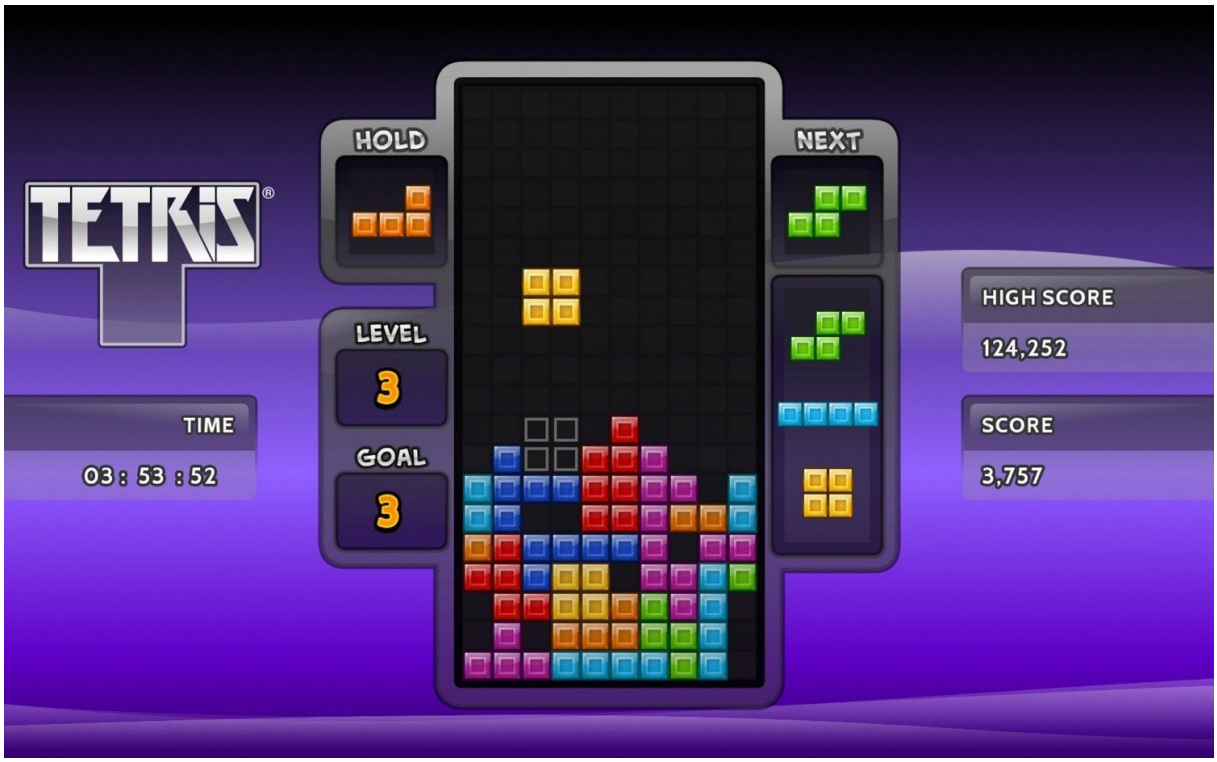

Figura 2 – Captura de tela do jogo Tetris Online
Fonte: (TETRIS, 2022)

No Tetris não há condição específica de vitória, apenas condição de derrota. A principal lógica de controle está relacionada aos blocos, pois a cada partida de Tetris há o cálculo de pontuação conforme o progresso que o jogador obteve ao sobreviver. Para ganhar pontos é necessário que o jogador complete uma linha de blocos para que esta linha seja desfeita, liberando assim espaço para as próximas peças. Por outro lado, há uma condição de derrota fixa, em que os blocos não podem atingir o topo do espaço disponível; logo, quanto mais tempo o jogador permanecer vivo sem chegar ao topo, limpando as linhas de blocos abaixo, mais pontuação ele irá alcançar.

Exemplos são *Tetris 99* (NINTENDO, 2022a) como uma competição entre 99 jogadores simultaneamente e sua mecânica de enviar fileiras fixas de blocos que o jogador completar para o campo dos outros jogadores como é possível ver na Figura 3 e *Tricky Towers* (GAMES, 2022b) em que até quatro jogadores podem disputar pela vitória em que sua torre de blocos não pode cair, como pode ser visto na figura 4, contendo mecânicas únicas como peças enormes para dificultar o campo do adversário. É importante ressaltar que o jogo proposto neste trabalho também tem o *Tetris* como base, porém busca inovar ao incorporar elementos educacionais e desafios específicos para auxiliar os alunos no desenvolvimento de habilidades de estruturação de algoritmos.

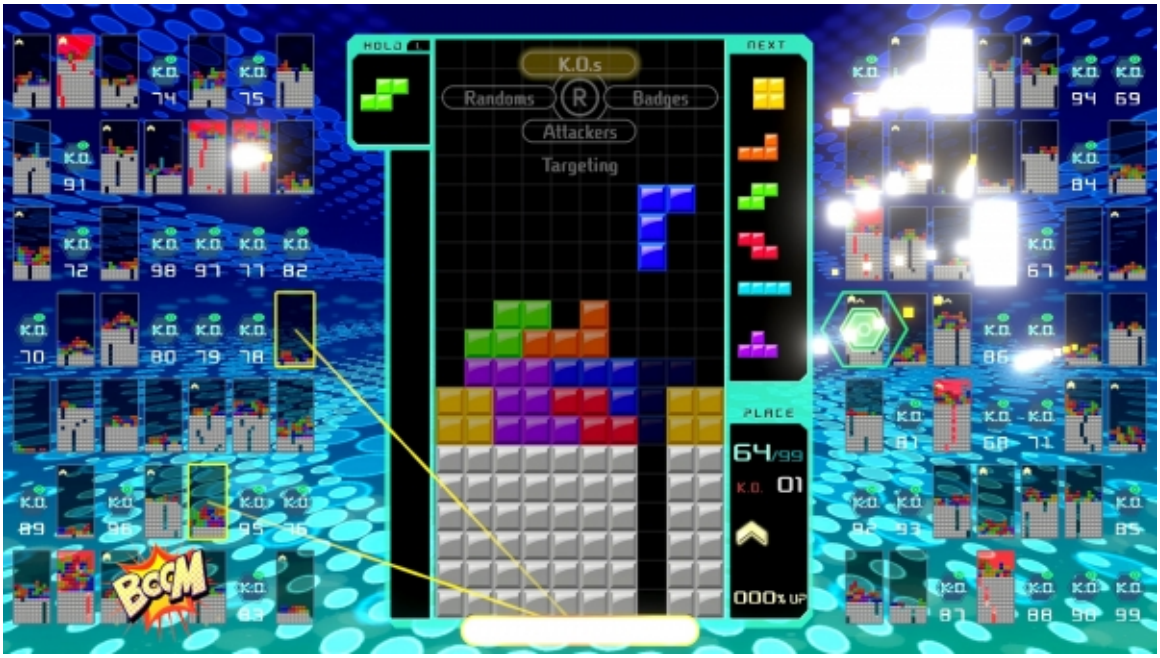

Figura 3 – Captura de tela do jogo Tetris 99 (2019)
Fonte: (SPACE, 2022)

Figura 4 – Captura de tela do jogo Tricky Towers (2016)
Fonte: (NINTENDO, 2022b)

2.9 Trabalhos Relacionados

Muitas outras iniciativas que utilizavam a teoria do Construcionismo surgiram com o passar dos anos. Exemplos são *SmallTalk* (KAY, 2022), *AgentSheets* (AGENTSHEETS, 2022), *Etoys* (STORYMILL, 2022), *Scratch* (SCRATCH, 2022), *StarLogo* (TECHNOLOGY, 2022) e *NetLogo* (WILENSKY, 2022), dentre outras que se aproximam mais com o aprendizado na área da robótica temos o *Physical Etoys* (STORYMILL, 2022), *Lego Mindstorms ev3* (LEGO, 2022b), *Lego WeDo* (LEGO, 2022a) e *Robot Emil* (STUDIO, 2022b). A seguir, serão apresentadas algumas dessas iniciativas em particular, que desempenharam papéis importantes ao auxiliar no desenvolvimento e inspiração do presente trabalho.

2.9.1 Scratch

O *Scratch* é uma linguagem de programação de alto nível, com foco na educação infantil. Foi desenvolvida pelo MIT *Media Lab*, e tinha como fundamento incentivar a combinação e reutilização de códigos. Seu visual é baseado na construção de algoritmos por meio do uso de blocos, como pode ser visto na figura 5, e tem como código base o *JavaScript*. Neste trabalho, o *Scratch* serviu de inspiração para a criação de uma funcionalidade de alto nível, mas com uma abordagem diferente de construção por blocos, como será detalhado na seção 3.2.1.

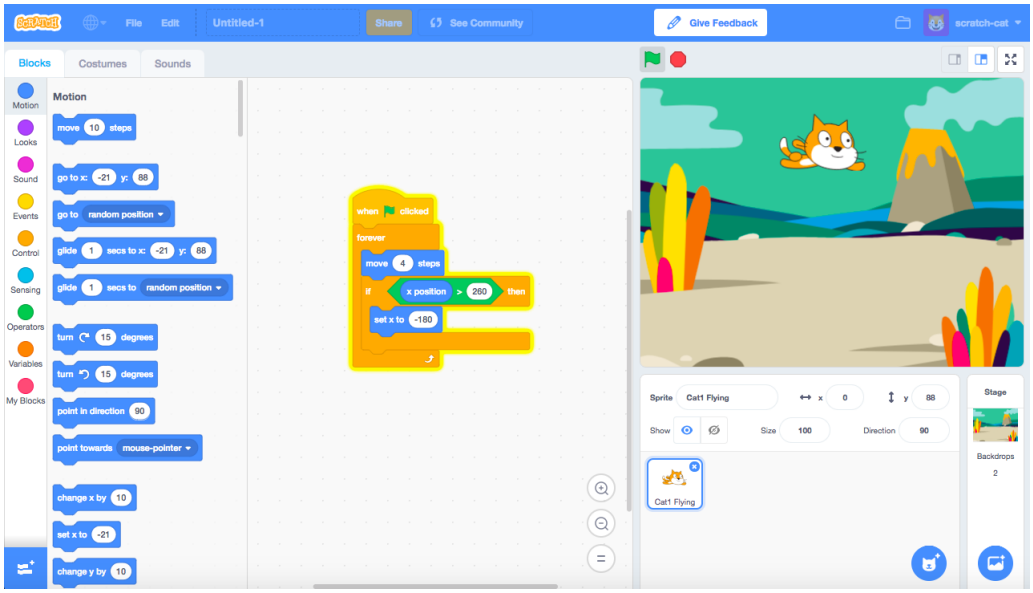

Figura 5 – Interface da linguagem de programação Scratch
Fonte: (COMPUTAçÃO, 2022)

2.9.2 AgentCubes

O *AgentCubes*, uma evolução do seu precursor *AgentSheets*, é uma ferramenta que se baseia na metodologia de programação por blocos de arrastar e soltar. Seu objetivo principal é fomentar o pensamento computacional entre os alunos, sem necessariamente formar programadores, conceito este adotado pelo presente trabalho. Assim, para tornar o ensino da computação mais prático, uniram-se ferramentas de apoio à programação e de apoio à criatividade. Ferramentas que dão suporte à programação permitem a abordagem de desafios semânticos e pragmáticos por meio do processo de depuração, a fim de analisar a especificidade do programa em determinadas situações. Já as ferramentas de apoio à criatividade dão a possibilidade dos alunos criarem seus próprios objetos por meio da modelagem 3D, para que assim a programação e a parte criativa possam caminhar em conjunto, motivando assim o aluno.

Figura 6 – Interface da linguagem de programação AgentCubes
Fonte: (WIKIPEDIA, 2022)

2.9.3 Fabrica Inorgânica

Além dessas iniciativas que tiveram sua base na teoria do construcionismo, há também uma aplicação de autoria própria desenvolvido em colaboração com Filipe Maia e Luiz Loja, o jogo Fabrica Inorgânica (ANDRADE FILIPE MAIA, 2022). Esse aplicativo foi desenvolvido em projeto do Programa Institucional de Bolsas de Iniciação em Desenvolvimento Tecnológico e Inovação (PIBITI), em uma colaboração do Instituto Federal de Educação, Ciência e Tecnologia De Goiás (IFG) Campus Anápolis, cujos alunos desenvolveram a aplicação, e o IFG Campus Luziânia, que aplicou e avaliou os resultados do uso da ferramenta, validando a aplicação como um jogo educacional (MARTINS, 2022). O objetivo do projeto foi a criação de uma ferramenta que auxiliasse os alunos no ensino de química inorgânica.

No jogo, o usuário deve separar os frascos de compostos inorgânicos em suas respectivas classificações, podendo cometer até três erros em um total de cem compostos, como pode ser visto na figura 7. Ao fim, caso o jogador tenha obtido algum erro, uma tela irá apresentar qual foi o erro e apresentar, qual seria a resposta correta, assim como na figura 8. Assim, quanto mais o jogador usasse a ferramenta, mais ele iria se aperfeiçoando na identificação dos compostos desse conteúdo de química.

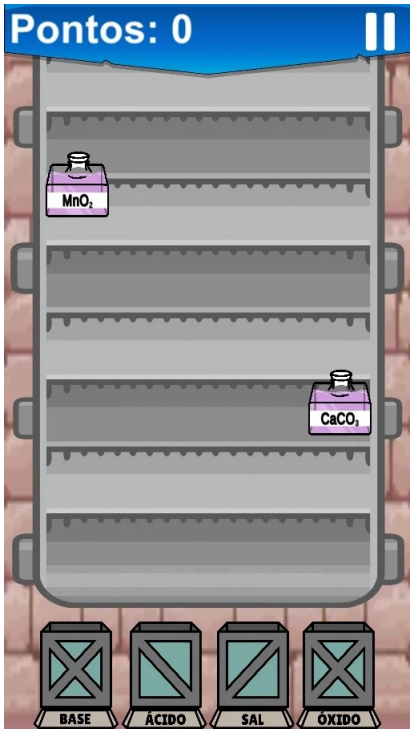

Figura 7 – Tela do jogo Fabrica Inorgânica
Fonte: (ANDRADE FILIPE MAIA, 2022)

Figura 8 – Tela final do jogo Fabrica Inorgânica após cometer três erros
Fonte: (ANDRADE FILIPE MAIA, 2022)

3 Metodologia

Esta seção apresenta como foi desenvolvido o trabalho de acordo com os objetivos propostos na seção 1.1. Inicialmente serão descritas as ferramentas que integraram o desenvolvimento deste projeto. Em seguida, será feita a descrição do *design* do jogo, incluindo suas mecânicas, condições de vitória e derrota e estética visual. Além disso, será discutido o design das fases, abordando como foi aplicado o conteúdo de algoritmos nas fases e como elas utilizam o conceito de construtivismo de Papert para a fundamentação crescente do conhecimento, bem como da dificuldade que os usuários enfrentarão.

A proposta deste trabalho é de desenvolver um *software* do tipo jogo digital, que poderá ser usado como ferramenta de auxílio para os alunos que estiverem estudando algoritmos, tendo como principal foco a estruturação dos comandos utilizando a linguagem Portugol. Essa linguagem é usada também como ferramenta de apoio ao ensino de lógica de programação, por ela ser uma pseudo-linguagem que utiliza comandos em português facilita o aprendizado, sendo algo mais natural ao usuário iniciante (MANSO; OLIVEIRA; MARQUES, 2009). O trabalho aqui proposto desenvolveu um jogo que faça uso da pseudo-linguagem, baseada em portugol, assim como os blocos adaptados para o cenário de lógica de programação.

3.1 Ferramentas

As ferramentas que foram utilizadas no desenvolvimento deste projeto são enquadradas em quatro diferentes grupos: as *game engines*, ferramentas que fornecem um ambiente para a construção de um jogo, plataformas de distribuição de *software*, onde foi hospedado o *download* do programa executável, plataforma de gerenciamento de projetos, utilizada para facilitar o planejamento, acompanhamento e entrega de projetos e plataforma de hospedagem de repositórios de código-fonte.

3.1.1 Game engines

Uma *game engine* é um *software* que fornece um ambiente para construção dos mais diversos elementos dos jogos, contando com um conjunto de bibliotecas para prestar suporte ao desenvolvimento. Dentre as capacidades dessas *engines* está a programação. Há também disponibilidade de *engines* gráficas que serão responsáveis por renderizar tanto gráficos 2D e 3D, *engine* capaz de simulações da física dos objetos no ambiente, como de colisões e de gravidade, além disso, é possível fazer animações, gerenciamento de sons e arquivos.

A *Unity* é a *engine* que foi utilizada para o desenvolvimento deste projeto, pois ela conta com uma versão gratuita, que possibilita sua utilização por desenvolvedores independentes, além de contar com diversos recursos para desenvolvimento de jogos tanto 2D quanto 3D, podendo distribuir seus jogos para PC(*Personal Computer*), *Mac*, *Xbox*, *Playstation*, entre outros. Exemplos de jogos que usaram a plataforma são: *Cuphead*(MDHR, 2022); *Pokemon GO*(COMPANY, 2022); *Inside*(PLAYDEAD, 2022); *Hearthstone*(ENTERTAINMENT, 2022); *Ori and the Blind Forest*(STUDIOS, 2022).

O uso eficaz desta ferramenta já foi constatado pelo desenvolvimento de outro *software* educacional de autoria própria, como citado na seção 2.9.3, mas é também justificado, em geral, devido à abundância de recursos disponíveis graças a contribuições da comunidade: está facilmente disponível tanto material teórico para o estudo no uso dessa *engine*, quanto *assets*, *scripts*[1] e outros tipos de suplementos, gratuitos ou pagos, para uso dentro do desenvolvimento do jogo.

3.1.2 Plataformas de distribuição de software

O lugar onde é possível disponibilizar um *software* que foi desenvolvido é em uma plataforma especializada. Para este trabalho, foram consideradas as plataformas em que é possível publicar jogos, e no mercado há uma grande variedade entre elas. Cada uma tem suas especificidades: a *Google Play* (GOOGLE, 2022), por exemplo, tem como plataforma de destino dispositivos *Android*, enquanto a *Steam* (CORPORATION, 2022) possibilita o acesso e *download* de jogos nos sistemas operacionais *Windows*, *macOS* e *Linux*.

A publicação deste trabalho foi feita pela plataforma *itch.io* (CORP, 2023). O *itch.io* é uma plataforma de distribuição de jogos e conteúdo interativo, que oferece um ambiente dedicado à publicação e compartilhamento de criações independentes. O *itch.io* proporciona uma vitrine acessível para projetos de diferentes tamanhos e estilos. Sua flexibilidade permite aos desenvolvedores disponibilizarem seus jogos gratuitamente ou com opções de pagamento, além de oferecer ferramentas para a personalização e divulgação dos projetos. A escolha do *itch.io* para este projeto foi motivada por permitir o lançamento do jogo de forma gratuita, além de sua interface amigável e a possibilidade de alcançar uma audiência específica, interessada em jogos independentes.

3.1.3 Plataformas de gerenciamento de projetos

A plataforma de gerenciamento de projetos é uma ferramenta essencial que proporciona um ambiente favorável para a gestão eficaz de uma variedade de atividades relacionadas a projetos. Com uma ampla gama de recursos e funcionalidades, essa plataforma auxilia no

[1] Um script, no contexto de desenvolvimento de software, refere-se a um conjunto de instruções ou comandos executados por um programa ou sistema.

planejamento, acompanhamento e controle das tarefas, além de promover a comunicação e colaboração entre os membros da equipe.

No contexto deste projeto em específico, foi selecionada a plataforma *Jira* (ATLASSIAN, 2023) como a solução ideal de gerenciamento de projetos. Essa escolha foi embasada em vários fatores, incluindo a disponibilidade de uma versão gratuita do *Jira*, que possibilita seu uso por equipes de desenvolvimento independentes. Além disso, a familiaridade com a plataforma também foi considerada, o que facilita sua utilização e integração com o fluxo de trabalho existente.

O *Jira* oferece recursos abrangentes, tais como a capacidade de criar e atribuir tarefas, estabelecer prazos, acompanhar o progresso, gerar relatórios e integrar-se a outras ferramentas relevantes. Essas funcionalidades contribuem para aumentar a eficiência e a organização no gerenciamento de projetos. Sendo que essas tarefas com identificações únicas auxiliavam no processo de gerenciar as atualizações feitas e salvas no *GitHub*, que será descrito na próxima seção.

3.1.4 Plataforma de hospedagem de repositórios de código fonte

Uma plataforma de hospedagem de código-fonte é um serviço *online* que oferece um ambiente seguro para armazenar e gerenciar projetos de *software*. Essas plataformas permitem que os desenvolvedores armazenem seu código-fonte em repositórios e forneçam recursos de controle de versão para acompanhar as alterações feitas ao longo do tempo. Além disso, essas plataformas facilitam a colaboração entre os membros da equipe, permitindo que trabalhem em conjunto e compartilhem seu trabalho de forma eficiente.

O *GitHub* (GITHUB, 2023) é uma das plataformas de hospedagem de código-fonte mais populares e amplamente utilizadas, contando com mais de 100 milhões de desenvolvedores (GOMES, 2023). Ele oferece um ambiente robusto para hospedar projetos de *software* e colaborar com outros desenvolvedores. No *GitHub*, os desenvolvedores podem criar repositórios públicos ou privados para armazenar seu código-fonte. Além disso, o *GitHub* fornece recursos poderosos de controle de versão, como branches para desenvolvimento paralelo, solicitações de pull para revisão de código e problemas para rastrear e resolver questões relacionadas ao código.

Ao utilizar o *GitHub* no projeto, foram obtidas vantagens significativas. A integração com o *Jira* permitiu um melhor gerenciamento do projeto, facilitando o acompanhamento das tarefas e prazos. Além disso, a combinação do *GitHub* com o *Jira* possibilitou uma visão clara do progresso do desenvolvimento por meio dos *commits* no repositório, proporcionando um registro detalhado das alterações feitas ao longo do tempo. Essa integração entre o *GitHub* e o *Jira* contribuiu para uma gestão eficiente do projeto.

3.2 Design do jogo

O programa em questão teve como inspiração o jogo já existente *Tetris*, que foi explicado na seção 2.8. O fator principal para essa decisão foi a familiaridade com o conceito de *Tetris* pelo público geral: conforme os dados apresentados na seção 2.8, mais de 520 milhões de cópias foram distribuídas. Isto facilita no reconhecimento das mecânicas básicas que serão usadas no jogo proposto por este trabalho. Possuindo uma jogabilidade simples e intuitiva de conectar as peças, o *Tetris* possibilita que os usuários que não conhecem o tipo de jogo se adaptem facilmente.

3.2.1 Mecânicas

Uma das principais mudanças é o movimento das peças que começa na parte inferior do quadro e termina na parte superior. Neste jogo, ao contrário do *Tetris*, não há uma grande aleatoriedade, pois o objetivo é construir um algoritmo de forma natural, seguindo a escrita de cima para baixo e da esquerda para a direita, o que proporciona uma melhor compreensão do código que está sendo construído.

Cada peça no jogo contém um trecho de código em Portugol, conforme ilustrado na figura 9. O bloco possui uma altura de dois quadrados e uma largura que varia de um a doze quadrados, uma cor específica determinada pelo esquema de cores, descrito em detalhes na tabela 1, um texto que representa o trecho de código e uma posição relativa correta no plano X e Y.

Figura 9 – Exemplo de peça com trecho de códgio
Fonte: Elaborado pelo autor (2023)

O jogador é responsável por conduzir os blocos de código utilizando as teclas de seta do teclado. Existem três possíveis movimentos: pressionar a seta para cima acelera o movimento de subida; pressionar a seta para a direita movimenta a peça para a direita; e pressionar a seta para a esquerda movimenta o bloco para a esquerda. Todos os movimentos são limitados pelo limite da grade para evitar ultrapassagens. Por outro lado, para interagir com os botões disponíveis na tela, é necessário utilizar o mouse. Sobre as peças é importante destacar que, diferentemente do *Tetris*, não é possível rotacionar o bloco de código.

Implementou-se também a mecânica de limpar as linhas quando ocorre um erro. Os erros acontecem quando o jogador não posiciona a peça corretamente segundo o gabarito da fase, seja errando a linha em que o bloco foi colocado ou sua posição dentro da linha, mesmo que esteja na linha correta. Quando uma dessas condições é atendida, tanto a linha em que o bloco foi posicionado incorretamente quanto a linha à qual o bloco pertencia são eliminadas. Isso permite que o jogador reescreva a linha com os mesmos blocos que foram apagados, posicionando-os corretamente. O jogador tem a chance de corrigir o erro até três vezes, caso atinja esse limite, a fase é encerrada e é necessário resetar para jogar a fase novamente desde o início.

O conjunto de elementos possíveis a serem criados é estabelecido seguindo critérios específicos. No início do jogo, o primeiro requisito determina que o bloco inicial do algoritmo será o primeiro a aparecer. Após fixá-lo na grade, outros critérios devem ser considerados. O segundo requisito consiste em adicionar à lista de alternativas o bloco que será conectado ao bloco anteriormente fixado, desde que ainda haja blocos a serem colocados na mesma linha. Além disso, é aplicada uma condição adicional, incluir na lista de possibilidades o bloco da próxima linha, caso exista uma próxima linha de algoritmo. Se ainda não houver blocos nessa linha, o primeiro bloco é inserido na lista de alternativas. Caso já existam blocos naquela linha, verifica-se a partir do último bloco colocado se ainda há blocos disponíveis para serem usados. Se houver, esses blocos também são adicionados à lista de alternativas. É importante ressaltar a verificação da largura do bloco na próxima linha para assegurar que ela não impeça a colocação de blocos na linha anterior. Ao final, essa lista de alternativas é acessada e uma delas é escolhida aleatoriamente para criar a peça, permitindo ao jogador posicioná-la na grade do jogo. Em seguida, essa lista continua sendo atualizada com novas opções, seguindo todos esses critérios, e o processo de criação prossegue até não haver mais elementos para serem colocados.

Podemos perceber então que todo jogo tem mecânicas que definem o jogo, sendo possível definir um jogo com uma palavra-chave: por exemplo, *Counter-Strike* (VALVE, 2022) é “atirar”, e *Forza Horizon* (MICROSOFT, 2022) é “dirigir”. Para o jogo proposto neste trabalho, a mecânica central que guia a ideia do jogo é “conectar”, seja conectar os blocos posicionando-os na grade, como conectar os trechos de código para concluir a fase.

3.2.2 Condição de vitória e derrota

Comparado à versão original do jogo *Tetris*, conforme visto na seção 2.8, as condições de vitória e derrota foram modificadas. Ao contrário da contagem de pontos, nesta versão, o jogador será obrigado a concluir a montagem do algoritmo proposto pela fase como condição de vitória, resultando na exibição do painel de Vitória. Por outro lado, a condição de derrota está relacionada à contagem de vidas. Durante o processo de montagem do algoritmo, se o jogador cometer erros no posicionamento dos blocos, uma

vida será subtraída e ocorrerá a eliminação das linhas para que o jogador possa tentar novamente, conforme mencionado na seção 3.2.1. Se o jogador acumular três erros, o painel de "Tentar Novamente" será exibido na tela.

3.2.3 Visual e sonoro

No desenvolvimento de um jogo, otimizar o tempo é de extrema importância. Nesse sentido, a utilização de recursos visuais e sonoros disponíveis na comunidade desempenhou um papel fundamental. Foram empregados pacotes de recursos como o *GUI PRO Kit - Casual Game* (LAB, 2023), que proporcionou elementos visuais de qualidade, e o pacote *FREE Casual Game SFX Pack* (DUSTYROOM, 2023), que contribuiu com efeitos sonoros adequados. Ambos os pacotes estão disponíveis na *Unity Asset Store* (TECHNOLOGIES, 2023e). Além disso, foram utilizados efeitos de música obtidos no *OpenGameArt* (KELSEY, 2023). Por fim, foi desenvolvido pelo autor outros componentes visuais para complementar o jogo. Essa abordagem não apenas facilitou o desenvolvimento das telas, mas também contribuiu para a padronização visual do jogo.

3.3 Design de fase

As fases foram projetadas a fim de explorar diversos algoritmos, sendo um por fase. Esses algoritmos começam com conceitos mais básicos e aumentam gradualmente para propor um desafio cada vez maior, mas sempre usando conceitos já apresentados anteriormente, como no construtivismo de Papert citado na sessão 2.4.

Considerando que todo algoritmo proposto em uma fase deve ser possível; logo, os blocos que irão aparecer devem conter partes de código que possibilitem concluir o objetivo da fase. Sendo que os critérios para a sua criação sejam conforme o apresentado na seção 3.2.1.

3.3.1 Tabela de cores

Na área da computação, muitas interfaces de programação oferecem o recurso de syntax highlight, que consiste em realçar visualmente as diferentes partes de um código, com base em sua função gramatical ou sintática (HANNEBAUER; HESENIUS; GRUHN, 2018). Esse recurso é utilizado para destacar elementos como palavras-chave, variáveis, comentários e estruturas de controle, exibindo-os com cores diferentes. Neste projeto, foi implementada uma classificação de cores visando agrupar os elementos em cinco categorias distintas, facilitando assim a identificação e compreensão do código.

A primeira categoria, representada pela cor roxa, abrange as instruções gerais, como escrever, ler, limpar e parar. Essas instruções são utilizadas para executar ações básicas

dentro do programa. A segunda categoria, destacada pela cor azul, refere-se ao controle de fluxo. Ela inclui palavras-chave como “se”, “senão”, “enquanto”, “para” e “início()”, utilizadas para controlar a execução do código, estabelecendo condições e repetições.

A terceira categoria, representada pela cor ciano, trata das definições de funções. Nessa categoria, estão presentes as palavras-chave relacionadas às funções, como funções no geral, “retorno”, “construtor”, “inclua” e “programa”. Essas palavras são usadas para definir e criar funções personalizadas no código.

A quarta categoria, destacada pela cor vermelha, diz respeito à declaração de variáveis. Nela são incluídos os tipos de variáveis, como “inteiro”, “real” e “caractere”, além dos identificadores usados para nomear as variáveis declaradas, como “nome” e “índice”. Essa categoria também abrange outros elementos relacionados à declaração de variáveis.

A quinta categoria, temos a categoria verde, que engloba os comentários e as *strings*, ou seja, sequências de caracteres utilizadas para representar textos no código. Por fim, a categoria de Indicadores de Bloco de Código, representadas pela cor laranja, que contém “{”, “}”, “(” e “)”. É possível visualizar essa classificação de cores e conforme classificados por meio da Tabela 1.

Através dessa classificação de cores, é possível melhorar a legibilidade do código, facilitar a identificação de erros de digitação e sintaxe, e proporcionar uma experiência mais intuitiva e amigável ao desenvolvedor durante a escrita e edição do código.

Nome	Cor	Hexadecimal da Cor	Componentes
Instruções Gerais	Roxo	691CFE	- escreva - leia - limpa - para
Controle de Fluxo	Azul	0086FC	- se - senao - enquanto - para - inicio()
Definição de Funções	Ciano	02CFD0	- retorno - construtor - inclua - programa
Declaração de variável	Vermelho	B70B19	- inteiro - real - caracter
Strings	Verde	2BC40A	- "Olá Mundo!" - "O resultado e: " - "Você tem mais de 18 anos"" - //Comentário
Indicadores de Bloco de Código	Laranja	F36800	- { - } - (-)

Tabela 1 – Tabela de classificação de código por cores
Fonte: Elaborado pelo autor (2023)

3.3.2 Exemplo de construção de fase

Neste tópico, será apresentado um exemplo de como realizar a construção de uma fase, abordando os elementos necessários, as configurações envolvidas e a elaboração do algoritmo que representará a solução final da fase. Serão explorados os passos essenciais para a criação de um desafio, sendo o algoritmo proposto pela fase.

Para desenvolver o algoritmo de resposta da fase, é essencial considerar as etapas anteriores, uma vez que o objetivo é aplicar os conhecimentos adquiridos nessas etapas, conforme a teoria construcionista de aprendizagem mencionada na seção 2.3. Neste exemplo específico, o tema abordado é a entrada e saída de dados, e o desafio consiste em criar o algoritmo “Olá Mundo!” em Portugol, tendo como referência o website *Portugol WBStudio* (WEBSTUDIO, 2023).

Primeiramente, o código é analisado e dividido em linhas, levando em consideração que cada linha deve ter uma largura de 12 na grade do jogo, evitando linhas com uma sequência extensa de comandos. Após a formatação do código, obtemos um número reduzido de linhas, garantindo que nenhuma linha contenha apenas um elemento, como

era o caso anteriormente com as chaves, conforme ilustrado na Figura 10 o antes e o depois das alterações.

Figura 10 – Exemplo de código em Portugol antes e depois das alterações
Fonte: Elaborado pelo autor (2023)

Após essa divisão é necessário separar quantos blocos serão colocados por linha, segundo a Figura 11, e em seguida atribuir uma largura que permitam o texto caber corretamente, para manter com pouca variação o tamanho do texto que será apresentado, sempre observando para completar os doze de largura necessário para preencher uma linha. Além disso, já podemos aplicar a tabela de cores aos blocos desse algoritmo, conforme a Figura 12.

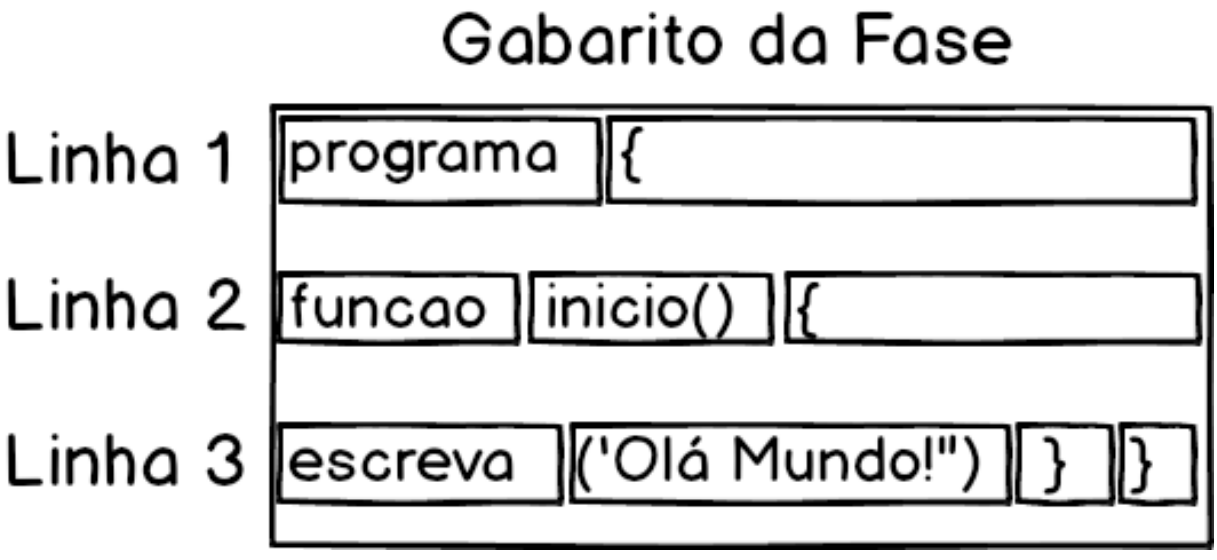

Figura 11 – Primeira parte da construção das linhas
Fonte: Elaborado pelo autor (2023)

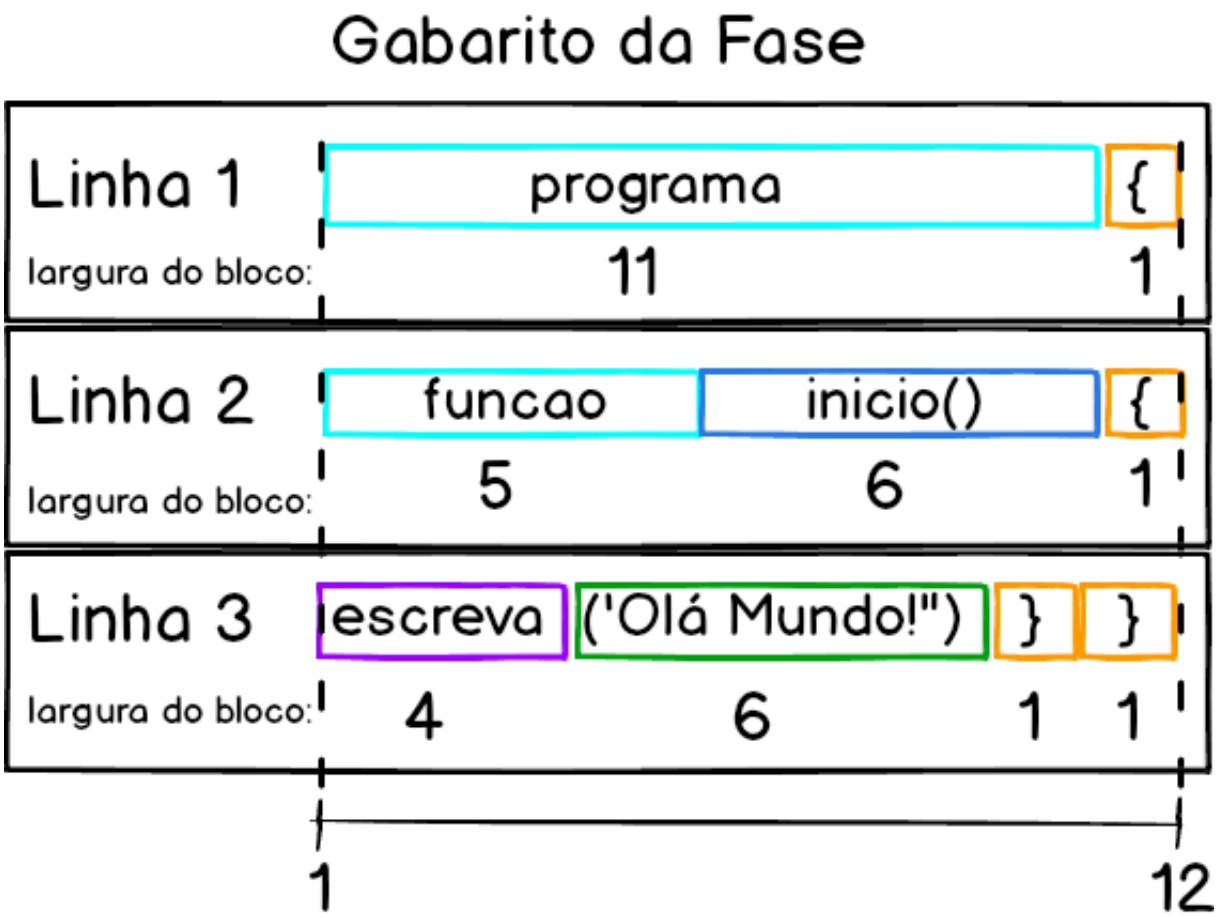

Figura 12 – Segunda parte da construção das linhas
Fonte: Elaborado pelo autor (2023)

Com as atribuições mencionadas, podemos avançar para a interface do *Unity*. No contexto do *Unity*, um *gameObject* representa a unidade fundamental de todos os elementos presentes em um cenário ou cena de jogo. Ele desempenha o papel de objeto interativo, manipulável e posicionável dentro do ambiente virtual. Um *gameObject* pode assumir diferentes formas, como personagens, objetos de cenário, luzes, partículas, elementos de interface do usuário (GUI), entre outros (TECHNOLOGIES, 2023b). No contexto deste projeto em particular, as linhas e blocos são exemplos de gameObjects, cada um com suas próprias características distintas.

Considerando que várias linhas e blocos serão criados em uma mesma fase, o conceito de *prefab* do *Unity* foi utilizado. Um prefab é um recurso que permite a criação de *gameObjects* pré-fabricados e reutilizáveis. Ele possibilita armazenar um objeto completo, incluindo sua hierarquia, componentes, propriedades e configurações, para ser utilizado em diferentes partes do projeto. O *prefab* é criado a partir de um objeto existente no cenário, podendo ser um personagem, objeto de cenário, GUI ou qualquer outro elemento. Uma vez criado o *prefab*, é possível instanciá-lo várias vezes no jogo, mantendo todas as suas

características e comportamentos (TECHNOLOGIES, 2023c).

Assim, cada bloco exibe características distintas, como nome, largura, altura, texto e cor, enquanto as linhas podem variar em termos de quantidade de elementos, conforme estabelecido pelo *prefab* utilizado. Essa abordagem assegura maior flexibilidade e reutilização dos elementos do jogo, mantendo suas propriedades específicas. Nesse contexto específico, serão criados três *prefabs* de linhas denominadas "Linha 1", "Linha 2" e "Linha 3", bem como nove *prefabs* de blocos com os seguintes textos: "programa", "{", "funcao", "inicio()", "{", "escreva", "('Olá Mundo!')", "}" e "}", totalizando assim doze *prefabs* para a montagem do algoritmo. É importante notar que os textos podem ser iguais, mas o nome do prefab é ajustado ao ser instanciado para receber uma identificação única, como pode ser visto na Tabela 2. Além disso, a representação visual no jogo das linhas da Figura 12 é apresentado no exemplo da Figura 13, em que os blocos estão com suas cores conforme as definições da Tabela 1, e em suas respectivas posições corretas.

Nome	Texto	Largura	Cor
programa(clone)1	programa	11	Ciano
{(clone)2	{	1	Laranja
funcao(clone)3	funcao	5	Ciano
inicio()(clone)4	inicio()	6	Azul
{(clone)5	{	1	Laranja
escreva(clone)6	escreva	4	Roxo
("Olá Mundo!")(clone)	("Olá Mundo!")	6	Verde
}(clone)8	}	1	Laranja
}(clone)9	}	1	Laranja

Tabela 2 – Tabela contendo os prefabs da fase "Olá Mundo!"
Fonte: Elaborado pelo autor (2023)

Conforme ilustrado na Figura 14, o *prefab* do bloco programa é constituído por quatro componentes distintos. O componente *Transform*, indicado pela cor vermelha na figura, é responsável pelo controle da posição, rotação e escala do objeto no espaço 3D (TECHNOLOGIES, 2023d). O componente *Script* Tetro Mov, representado pela cor amarela na figura, desempenha a função de manipular a movimentação do bloco na grade. Já o componente *Script* Constroi Bloco, destacado em rosa na figura, é utilizado durante a instanciação do prefab para acessar seus valores e criar um bloco com base nos parâmetros definidos. Por fim, o componente *AudioSource*, identificado pela cor verde na figura, é responsável pela reprodução e controle de sons, permitindo a adição de áudio ao objeto do jogo (TECHNOLOGIES, 2023a). Já em relação à linha, é necessário considerar apenas a representação lógica, composta pelos seguintes componentes: *Transform*, que define as

Figura 13 – Exemplo do jogo com a construção de blocos coloridos
Fonte: Elaborado pelo autor (2023)

propriedades de posicionamento, rotação e escala da linha; e *Script* Linha Gabarito, cuja função é armazenar os blocos presentes na linha e sua ordem correspondente. Na Figura 15, é possível observar essas informações, sendo o componente *Transform* representado em vermelho e o componente *Script* Linha Gabarito em amarelo.

Por fim, O objeto 'Gabarito' desempenha um papel fundamental no gerenciamento dos blocos, abrangendo sua criação, exclusão e verificação para confirmar que o jogador está respondendo corretamente. Essa funcionalidade é implementada por meio de um *gameObject* no *Unity*, conforme ilustrado na Figura 16. O 'Gabarito' é composto por diversos componentes, cada um representado por uma cor distintiva na figura: o componente *Transform* (vermelho), responsável pelo controle da posição, rotação e escala do objeto no espaço 3D; o *Script* Gabarito (amarelo), responsável por controlar os índices das linhas e dos blocos, definir o título da fase e executar animações; o *Script* Vida (rosa), encarregado de controlar e atualizar visualmente a vida atual na fase; e o *AudioManager* (verde), responsável pela gestão dos aspectos sonoros. Esses componentes trabalham em conjunto para fornecer a maioria da inteligência relacionada à manipulação dos blocos do algoritmo, concentrada no objeto 'Gabarito'.

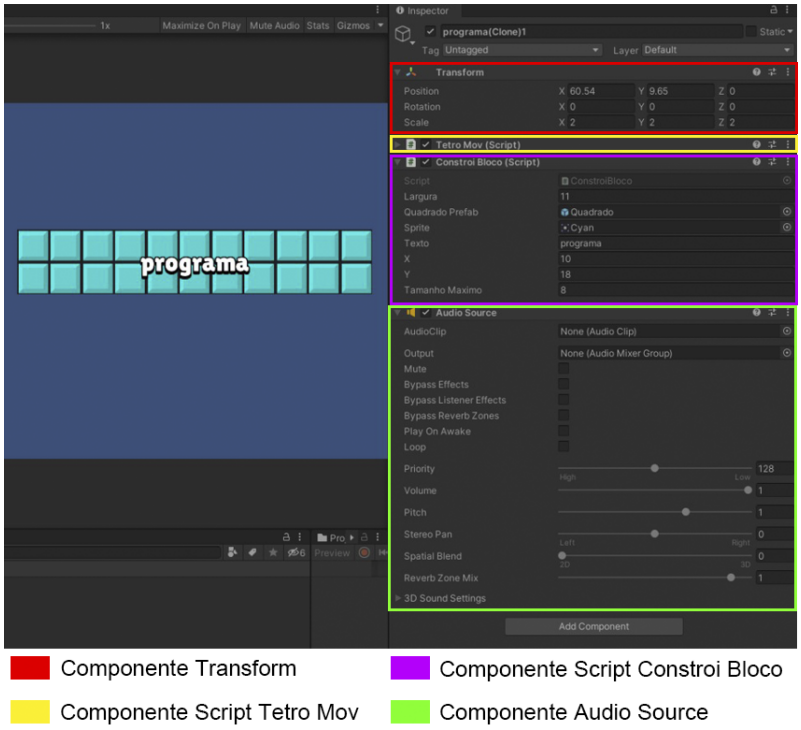

Figura 14 – Trecho retirado no Unity sobre os dados do prefab do bloco
Fonte: Elaborado pelo autor (2023)

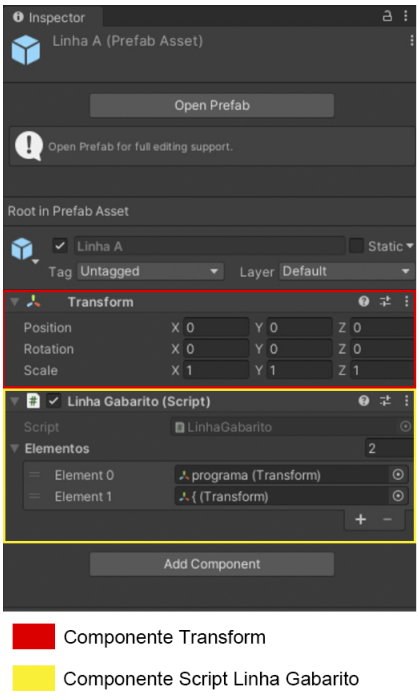

Figura 15 – Trecho retirado no Unity sobre os dados do prefab da linha
Fonte: Elaborado pelo autor (2023)

Além da construção do algoritmo, há outros parâmetros que devem ser ajustados na interface do *Unity*. Esses parâmetros incluem o título da fase, o texto exibido na tela de objetivo da fase e o texto de resumo do objetivo da fase no menu lateral esquerdo da

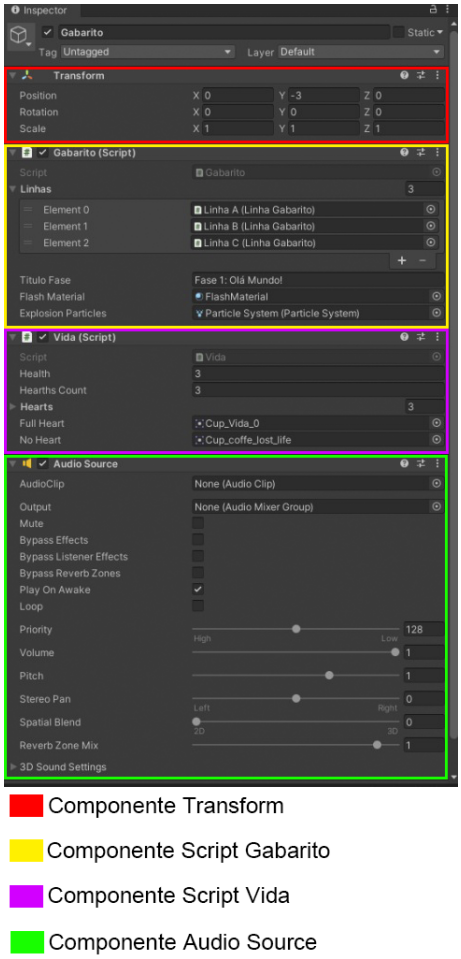

Figura 16 – Trecho retirado no Unity sobre os dados do gameObject gabarito
Fonte: Elaborado pelo autor (2023)

tela do jogo. Além disso, existem outros *gameObjects* que não precisam ser adaptados e se repetem em todas as fases, como os menus de Pausa, Vitória e Tentar Novamente.

3.4 Fluxo de aplicação

Para esta aplicação, foram desenvolvidas treze telas, incluindo o menu principal, o menu de fases, a tela do jogo e nove painéis flutuantes. Os painéis flutuantes são: Opções, Instruções, Sobre, Carregamento, Pausar, Vitória, Tente Novamente e dois painéis de Confirmação. A seguir, serão apresentadas mais detalhadamente as funcionalidades de cada tela.

No menu principal, é exibido o título do jogo, juntamente com as opções de acesso ao menu de fases, ao painel de instruções de como jogar e, por fim, o botão que direciona para o menu de opções, conforme apresentado na Figura 17. O menu de fases contém todas as fases bloqueadas e desbloqueadas do jogo. Cada botão fornece informações resumidas sobre o objetivo da fase. Além disso, a cada seis fases, uma nova seção de conteúdo é

desbloqueada, permitindo a opção de avançar para a próxima página ou voltar. Por fim, há a opção de retornar para o menu principal, conforme apresentado na Figura 18.

Figura 17 – Tela do menu principal
Fonte: Elaborado pelo autor (2023)

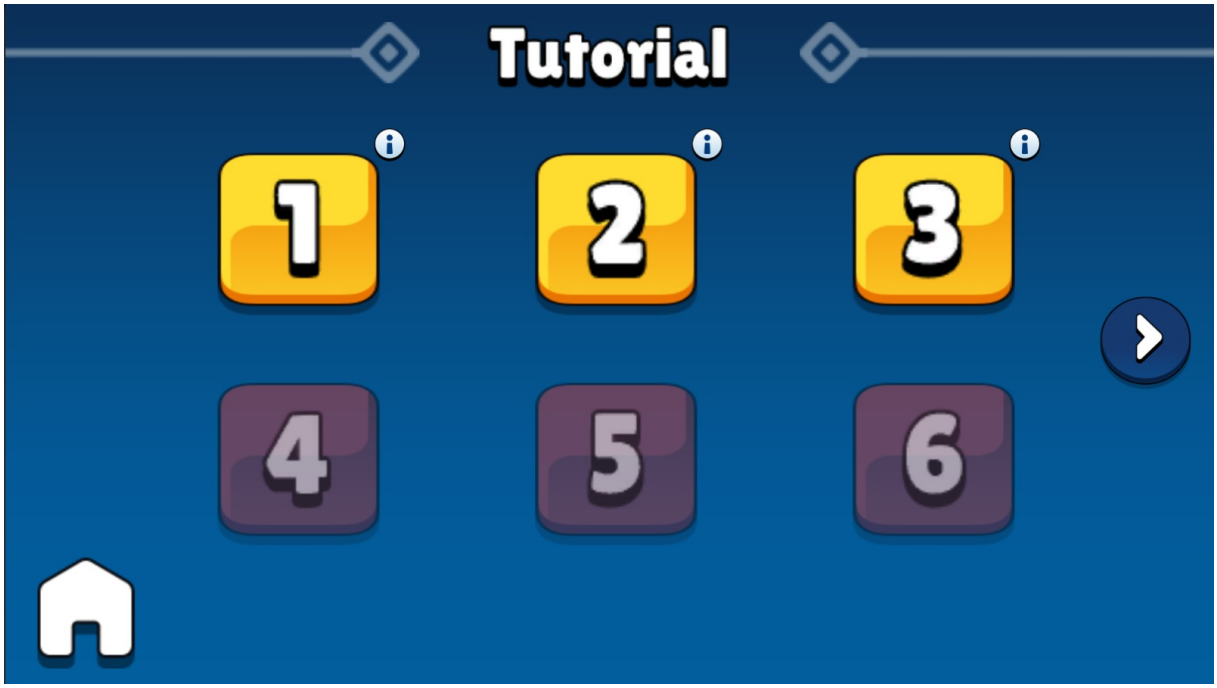

Figura 18 – Tela do menu de fases
Fonte: Elaborado pelo autor (2023)

No menu de opções, é possível ajustar o volume da música e dos efeitos sonoros por meio de controles deslizantes. Além disso, o menu oferece um botão "Sobre" que abre o painel com informações adicionais sobre o desenvolvedor e o propósito do jogo. Existe também um botão para fechar a tela atual e voltar para a tela anterior. Por fim, contém um botão "Sair" que encerra completamente o jogo, conforme apresentado na Figura 19. Já o painel de instruções há texto com algumas orientações e um botão para fechar o painel e retornar para a página principal, como pode ser visto na Figura 20

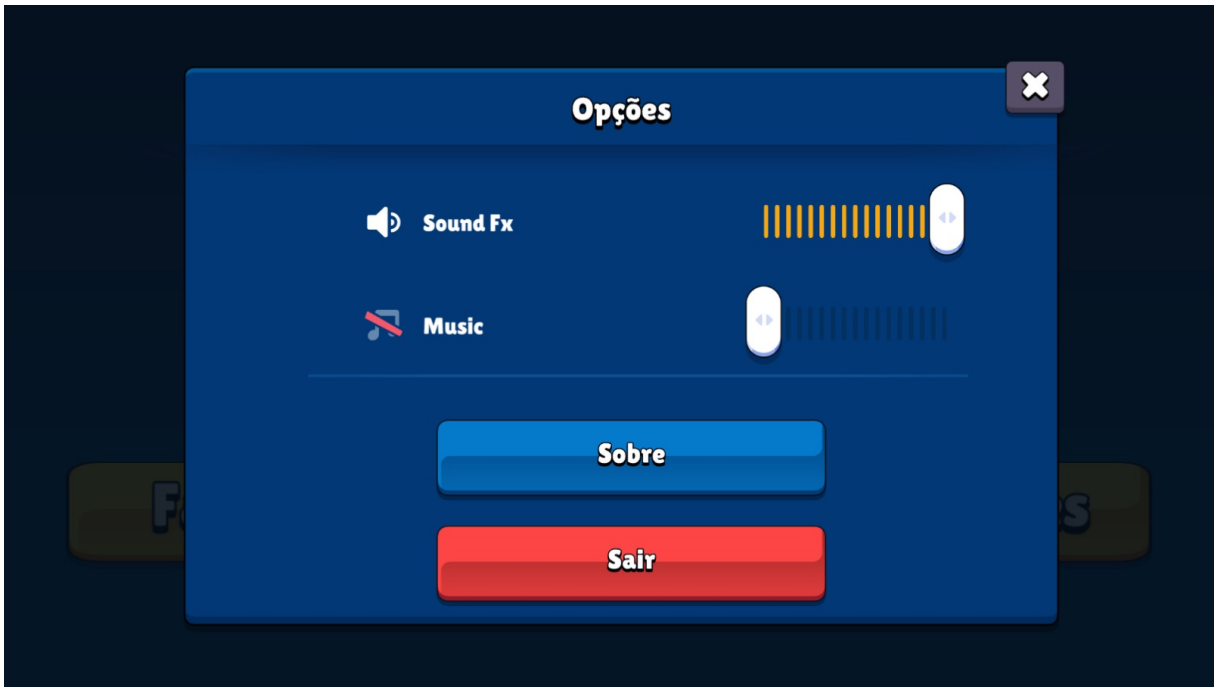

Figura 19 – Tela do painel de opções
Fonte: Elaborado pelo autor (2023)

Na tela de jogo, o painel inicial exibe o objetivo específico da fase, acompanhado por um botão que permite ao jogador iniciar quando estiver familiarizado com o que precisa fazer e outro botão que permite ele retornar para o menu principal, conforme apresentado na Figura 21. Após o início, a visão normal da fase é liberada, com um menu lateral esquerdo contendo um breve resumo da fase, a quantidade de vidas disponíveis e a opção de pausar o jogo. No quadro central, o jogador deve conectar os blocos de código, enquanto o quadro lateral direito exibe o título da fase e apresenta o resultado das ações corretas realizadas pelo jogador, conforme apresentado na Figura 22.

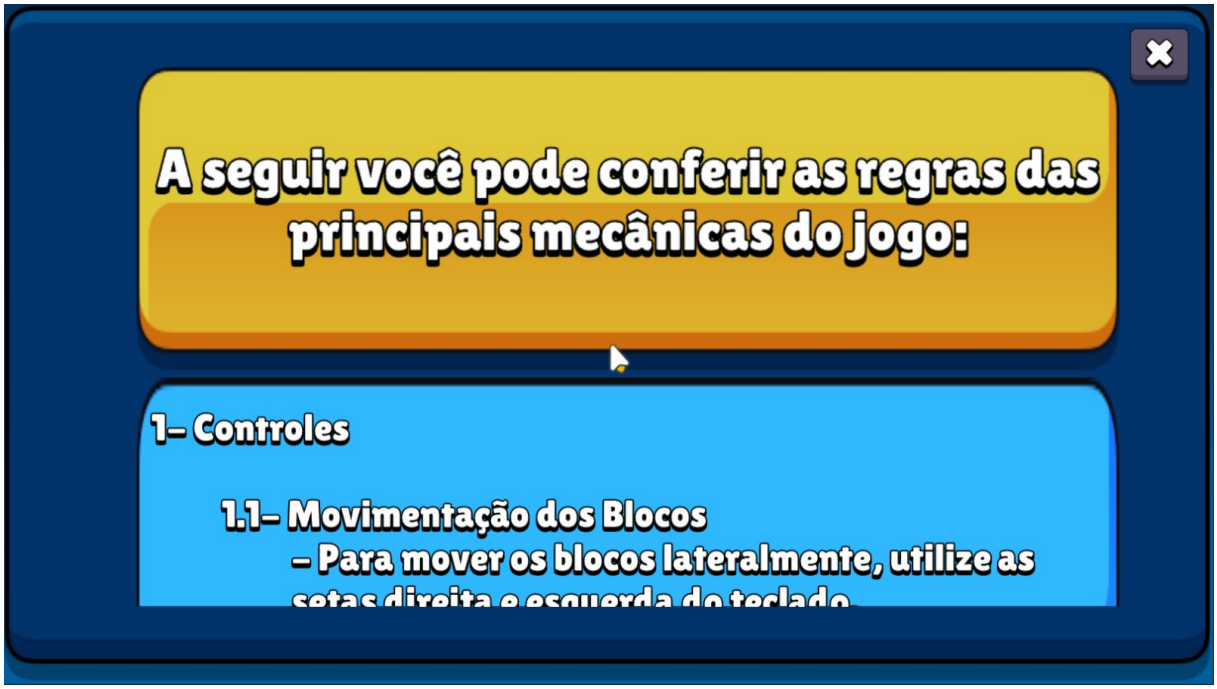

Figura 20 – Tela do painel de instruções
Fonte: Elaborado pelo autor (2023)

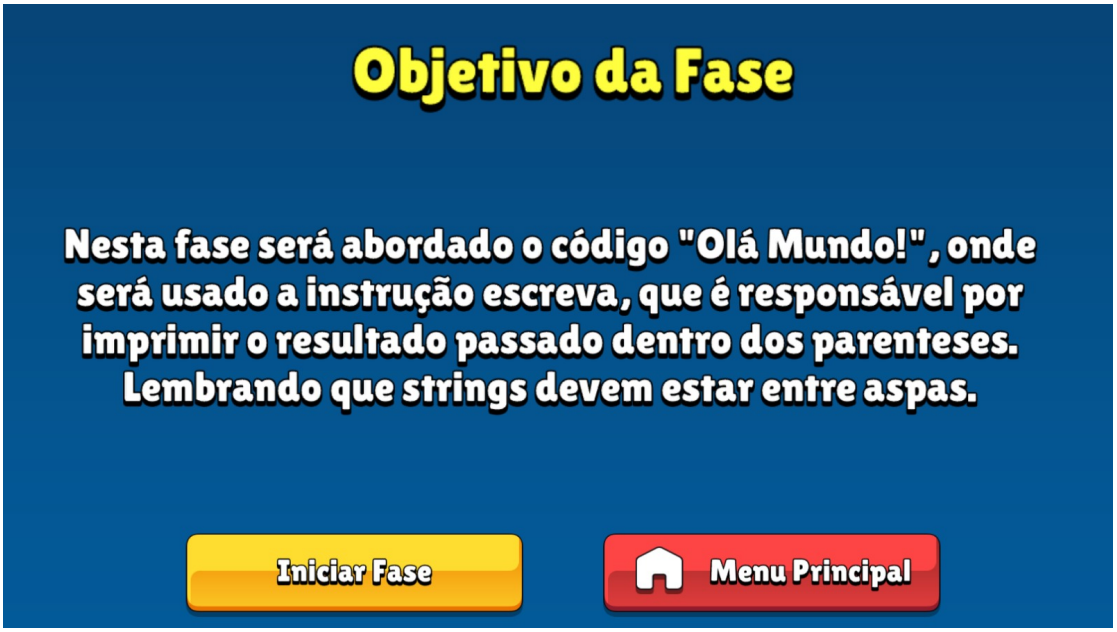

Figura 21 – Tela do painel de objetivo da fase
Fonte: Elaborado pelo autor (2023)

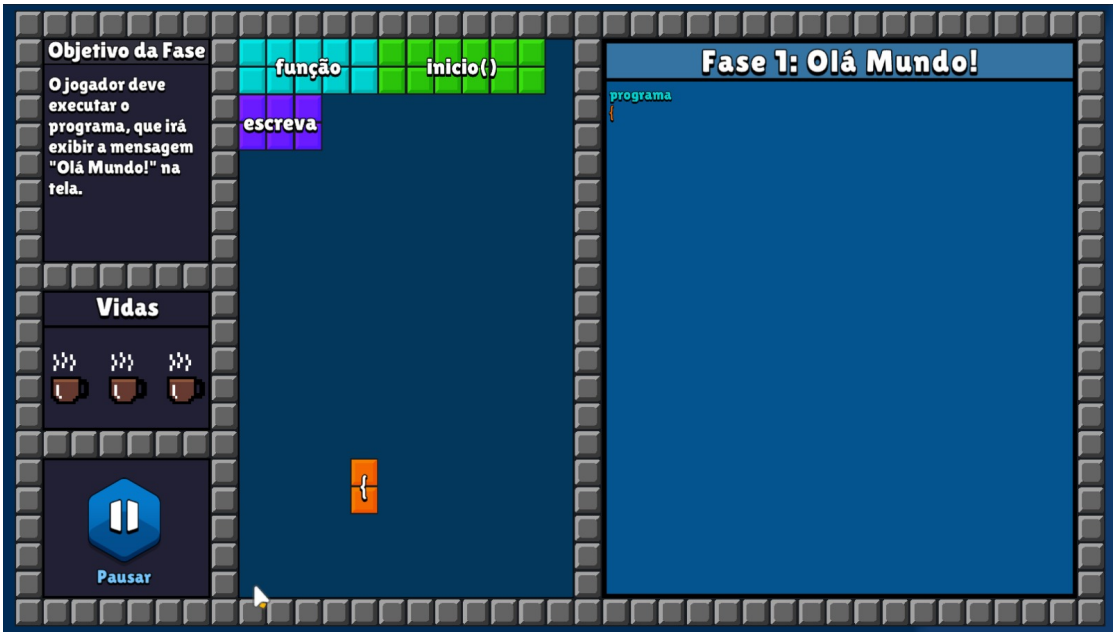

Figura 22 – Tela do jogo
Fonte: Elaborado pelo autor (2023)

O painel de pausa possui três botões: “Continuar”, que permite retomar a fase normalmente; “Resetar”, que reinicia a fase; e “Menu principal” que volta para a tela inicial, como apresentado na Figura 23. No entanto, os dois últimos botões possuem um painel de confirmação para cada um, solicitando ao jogador que confirme sua decisão antes de prosseguir com a ação, conforme apresentado na Figura 24. Há também o painel de Carregamento, onde é possível ver uma barra preenchida conforme o tempo de carregamento da próxima tela, conforme pode ser visto na Figura 25.

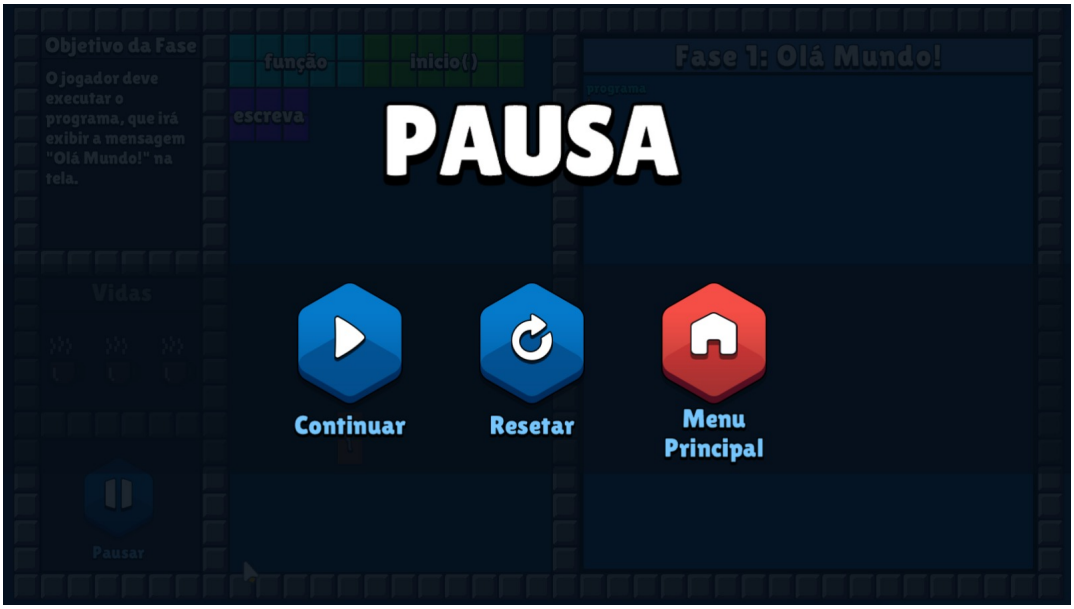

Figura 23 – Tela do painel de pausa
Fonte: Elaborado pelo autor (2023)

Figura 24 – Tela do painel de confirmação
Fonte: Elaborado pelo autor (2023)

Além disso, existem os painéis de vitória e Tente Novamente. O painel de vitória apresenta um título, uma imagem com animação de fundo, um texto indicando para conferir o resultado da fase e uma área onde o jogador pode visualizar o que foi concluído corretamente na fase. Por fim, há três botões: “Próxima” para avançar para a próxima fase, “Reiniciar” com um painel de confirmação e “Menu Principal” que retorna para a página principal, conforme apresentado na Figura 26. Já o painel de Tente Novamente contém um campo de título e texto com animação de fundo, além de dois botões: “Tentar Novamente”

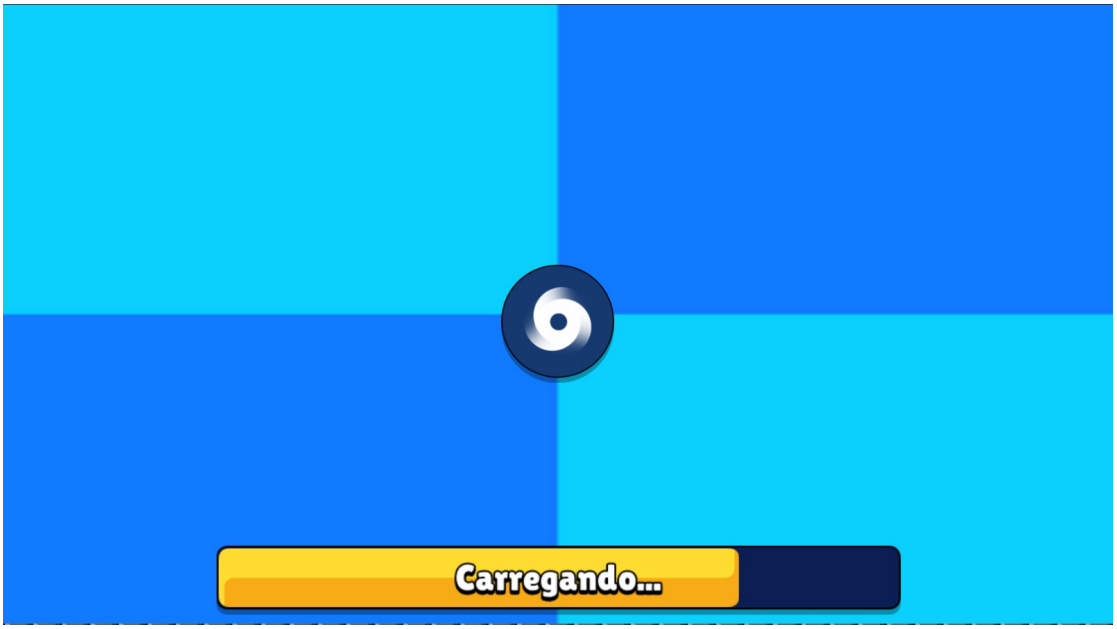

Figura 25 – Tela do painel de carregando
Fonte: Elaborado pelo autor (2023)

para resetar a fase e "Menu Principal" que redireciona para a página inicial, conforme apresentado na figura 27. Este fluxo também está descrito no fluxograma funcional da figura 28.

Figura 26 – Tela do painel de vitória
Fonte: Elaborado pelo autor (2023)

Figura 27 – Tela do painel de tente novamente
Fonte: Elaborado pelo autor (2023)

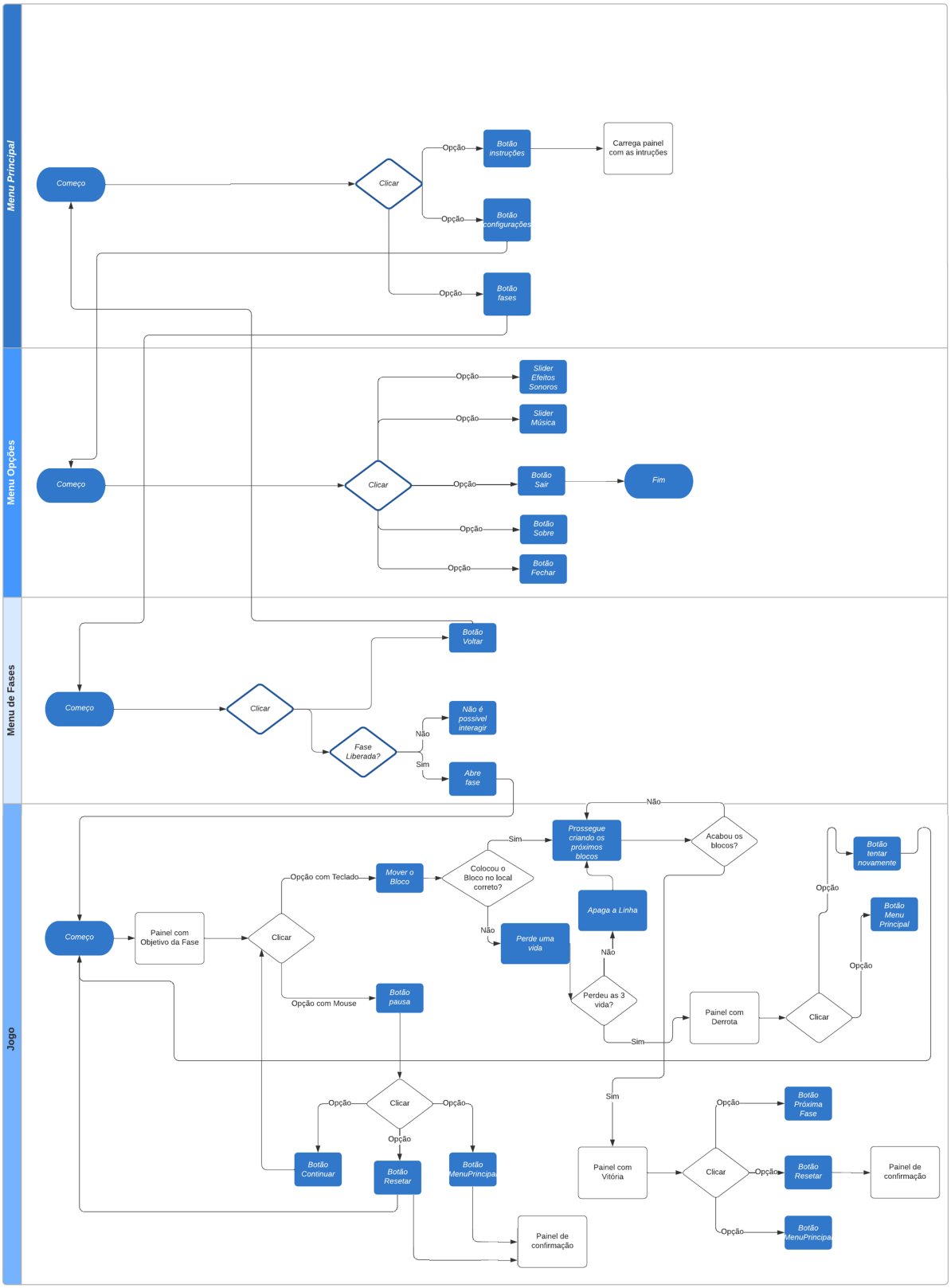

Figura 28 – Fluxograma funcional da aplicação
Fonte: Elaborado pelo autor (2023)

4 Resultados Alcançados

Como observado, os diversos conceitos apresentados na seção 2 desempenharam papéis importantes em várias etapas do processo da produção do jogo proposto neste trabalho, abrangendo desde a compreensão do que é um *software* até a fundamentação de teorias de aprendizagem que visam explorar a capacidade humana de desenvolvimento. Além disso, definições relacionadas ao jogo *Tetris* foram utilizadas como inspiração, porém com o desenvolvimento de novas mecânicas e abordagens para a conclusão do jogo. Também foram necessárias definições sobre o gênero de jogo ao qual ele se enquadra, a fim de orientar o processo de desenvolvimento. A combinação desses conceitos foi de extrema importância para a criação do jogo proposto neste trabalho, cujo objetivo é ser uma ferramenta útil no desenvolvimento de conceitos de estruturação de algoritmos.

Os objetivos propostos neste trabalho foram alcançados com sucesso. Desenvolveu-se uma aplicação que pode servir como uma metodologia complementar ao ensino tradicional de estruturação de algoritmos, proporcionando aos alunos iniciantes uma abordagem lúdica para assimilar os conceitos fundamentais. Inspirado pelas ideias de Seymour Papert descrito na seção 2.3, o *software* foi projetado para permitir aos usuários experimentar a integração de ações na resolução dos problemas propostos.

Para atingir esses objetivos, foram projetadas diferentes fases, como será descrito na seção 4.1, que incentivam o uso da lógica em conjunto com os conceitos da programação de computadores. Esses desafios foram elaborados para promover a aplicação prática dos conhecimentos adquiridos e oferecer uma boa experiência de aprendizado aos usuários.

Em resumo, todos os objetivos gerais e específicos propostos foram alcançados, resultando em um software que poderá ser eficaz e educativo na sua proposta de auxiliar no ensino de estruturação de algoritmos, oferecendo uma abordagem lúdica e interativa para os alunos. A aplicação foi publicada exclusivamente por meio da plataforma *itch.io* (ANDRADE, 2023a), garantindo sua disponibilidade e acessibilidade, e o código fonte está disponivível no *GitHub* (ANDRADE, 2023b).

4.1 Fases disponíveis

Foram desenvolvidas um total de doze fases, alinhadas com a teoria do aprendizado apresentada na seção 2.3. O jogo é iniciado com um conjunto de seis fases de tutorial, que fornecem uma introdução abrangente às mecânicas principais, como a conexão de peças para formar sequências lógicas definidas pelos objetivos das fases. Durante essas fases, os jogadores também têm a oportunidade de se familiarizar com o tratamento de erros, que

inclui a limpeza das linhas e a perda de vida como consequência, conforme explicado na seção 3.2.1.

À medida que os jogadores progridem, são gradualmente introduzidos novos conceitos e desafios nas fases subsequentes. O segundo conjunto de seis fases aborda especificamente o uso de entrada e saída no contexto do Portugol. Essas fases cobrem aspectos essenciais, como a declaração de variáveis, a utilização dos comandos "escreva()" e "leia()", bem como a concatenação de informações no comando "escreva()". Além disso, são apresentados exemplos práticos de como criar e utilizar funções no Portugol. O conjunto de fases é concluído com uma fase que sintetiza todos os conceitos anteriores, proporcionando aos jogadores a oportunidade de aplicar seus conhecimentos em um desafio mais abrangente. É importante destacar que as fases subsequentes estão bloqueadas e só serão liberadas quando as fases anteriores forem concluídas, garantindo uma progressão gradual do conhecimento proposto em cada seção.

A sequência de fases abordadas consiste nas seguintes fases: fase 1 - compreensão da construção inicial da Linha (parte 1); fase 2 - compreensão da construção inicial da Linha (parte 2); fase 3 - utilização de múltiplas linhas; fase 4 - montagem de um texto proposto sobre algoritmos; fase 5 - composição de um texto relacionado a uma frase filosófica; fase 6 - criação da tabuada do 2; fase 7 - formulação de uma linha abordando a declaração de variável; fase 8 - desenvolvimento de uma linha que envolva o comando "escreva()"; Fase 9 - construção de uma linha que faça uso do comando "leia()"; Fase 10 - elaboração de uma linha sobre concatenação; fase 11 - criação de uma declaração de função; e Fase 12 - desenvolvimento de um algoritmo para a tarefa de "Escreva seu nome".

É relevante destacar que, embora o projeto tenha como objetivo a construção de algoritmos, não se propõe a ser um compilador. Optou-se por uma abordagem que utiliza um gabarito fixo por fase, o que pode limitar a criatividade na resolução dos desafios, uma vez que as ações são restritas às opções disponíveis no gabarito. No entanto, essa abordagem foi escolhida por ser a forma mais viável de implementação dentro do escopo do projeto. Portanto, para alcançar o objetivo estabelecido, é necessário fornecer respostas de acordo com as opções predefinidas, se enquadrando como *puzzles* lineares como definido na seção 2.6.

5 Trabalhos futuros

Considerando as propostas de futuras implementações, uma sugestão relevante seria a realização de uma reestruturação do código atual. Com base no conhecimento adquirido durante o desenvolvimento, seria possível criar um código mais modular e de fácil manutenção, permitindo uma maior flexibilidade para adicionar novas mecânicas ao jogo. Além disso, seria interessante reformular a abordagem da orientação a objetos, direcionando a inteligência para os blocos em vez do gabarito. É importante, ainda, ampliar a quantidade de linhas em que os blocos são considerados disponíveis para serem instanciados, contudo, mantendo-se os critérios estabelecidos na seção 3.2.1 para garantir uma jogabilidade adequada.

Entre as possíveis melhorias e adições, destaca-se a implementação de recursos como a exibição da próxima peça disponível, a inclusão de peças falsas que não pertencem ao código original, proporcionando a opção de removê-las, e a disponibilização do jogo para outros sistemas operacionais, como o *Linux*. A melhoria da interface do usuário e o aprimoramento dos efeitos visuais e sonoros em cada ação do jogo são essenciais para tornar a experiência de jogo mais responsiva, fluida e imersiva.

Outra ideia interessante seria a criação de um sistema que permita aos usuários construírem suas próprias fases com seus exemplos personalizados, promovendo maior interatividade e engajamento com o jogo. Essa funcionalidade adicionaria um elemento de criatividade e personalização, melhorando a experiência do usuário e ampliando as possibilidades de aprendizado e prática de estruturação de algoritmos.

O software apresentado requer também a realização de experimentos com estudantes e educadores em um ambiente de sala de aula real, a fim de validar sua relevância no aprimoramento da condução das aulas. Para essa avaliação, será necessário coletar dados de desempenho tanto qualitativos quanto quantitativos em projetos futuros.

Referências

AGENTSHEETS. *AgentSheets*. 2022. <https://agentsheets.com/>. Citado na página 16.

ANDRADE, D. *Code Blocks no itch.io*. 2023. Https://laykabuss.itch.io/codeblocks. Citado 2 vezes nas páginas 9 e 42.

ANDRADE, D. *Repositório no GitHub do Code Blocks*. 2023. Https://github.com/david-dlsa/CodeBlock.git. Citado na página 42.

ANDRADE FILIPE MAIA, L. L. D. *Fabrica Inorgânica*. 2022. <https://play.google.com/store/apps/details?id=com.IFGBCC.FabricaInorganica>. Citado 3 vezes nas páginas 2, 18 e 19.

ATLASSIAN. *Jira*. 2023. Https://www.atlassian.com/br/software/jira. Citado na página 22.

CARVALHO, A. C. P. d. L. F. d.; LORENA, A. C. *Introdução à computação: hardware, software e dados*. [S.l.]: GEN/LTC, 2017. Citado na página 10.

CAVANAGH, T. *Dicey Dungeons*. 2022. <https://diceydungeons.com/>. Citado na página 13.

CÓDIGO, D. do. *Desafio do Código*. 2022. <https://www.desafiodocodigo.com.br/>. Citado na página 8.

COMPANY, T. P. *Pokemon GO*. 2022. <https://pokemongolive.com/pt_br/>. Citado na página 21.

COMPUTAçÃO, E. . *3 coisas para saber sobre o Scratch 3.0*. 2022. <https://luisgustavoaraujo.medium.com/3-coisas-para-saber-sobre-o-scratch-3-0-bd69ddb49d88>. Citado 2 vezes nas páginas 2 e 17.

CORP itch. *Itch.io*. 2023. <https://itch.io/>. Citado 2 vezes nas páginas 9 e 21.

CORPORATION, V. *Steam*. 2022. <https://store.steampowered.com/?l=brazilian>. Citado na página 21.

COSTA, V. *Puzzle: definição, subgêneros e hibridismos*. 2022. <https://thegamelogicist.medium.com/puzzle-defini%C3%A7%C3%A3o-subg%C3%AAneros-e-hibridismos-385285385527>,. Citado na página 14.

DUSTYROOM. *FREE Casual Game SFX Pack*. 2023. Https://assetstore.unity.com/packages/2d/gui/gui-pro-kit-casual-game-176695. Citado na página 25.

ENTERTAINMENT, B. *HearthStone*. 2022. <https://hearthstone.blizzard.com/pt-br>. Citado na página 21.

FERRACIOLI, L. Aspectos da construção do conhecimento e da aprendizagem na obra de piaget. *Caderno Brasileiro de Ensino de Física*, Universidade Federal de Santa Catarina (UFSC), v. 16, n. 2, p. 180–194, 1999. Citado 2 vezes nas páginas 8 e 11.

GAMES, B. *Blockly Games*. 2022. <https://www.blockly.games/?lang=pt-br>. Citado na página 8.

GAMES, W. *Tricky Towers*. 2022. <https://www.trickytowers.com/>,. Citado na página 15.

GITHUB. *GitHub*. 2023. Https://github.com/. Citado na página 22.

GOMES, L. *GitHub chega a 100 milhões de desenvolvedores na plataforma*. 2023. Https://www.showmetech.com.br/github-chega-a-100-milhoes-desenvolvedores/: :text=Contribui%C3%A7%C3%A3o%20mundial%20para%20o%20sucesso%20do%20 Citado na página 22.

GOOGLE. *Google Play*. 2022. <https://play.google.com/store/games>. Citado na página 21.

GRAMIGNA, M. R. *JOGOS DE EMPRESA*. 2022. <https://www.travessa.com.br/jogos-de-empresa-2-ed-2007/artigo/ba263fa2-b2af-4c1e-952b-b603069e2b4b>,. Citado na página 12.

HANNEBAUER, C.; HESENIUS, M.; GRUHN, V. Does syntax highlighting help programming novices? *Empirical Software Engineering*, Springer, v. 23, p. 2795–2828, 2018. Citado na página 25.

HAREL, I. E.; PAPERT, S. E. *Constructionism*. [S.l.]: Ablex Publishing, 1991. Citado na página 9.

HOED, R. M. Análise da evasão em cursos superiores: o caso da evasão em cursos superiores da área de computação. 2016. Citado na página 8.

KAY, A. *Smalltalk*. 2022. <http://www.smalltalk.org/>. Citado na página 16.

KELSEY, B. *OpenGameArt.org*. 2023. Https://opengameart.org/. Citado na página 25.

KLEINSORGE, G. *LINGUAGEM LOGO*. 2022. <https://sites.google.com/site/educarparaliberdade/linguagem-logo>. Citado 2 vezes nas páginas 2 e 12.

LAB, L. *GUI PRO Kit - Casual Game*. 2023. Https://assetstore.unity.com/packages/audio/sound-fx/free-casual-game-sfx-pack-54116. Citado na página 25.

LEGO. *Download your WeDo 2.0 software v. 1.9.385*. 2022. <https://education.lego.com/pt-br/downloads/retiredproducts/wedo-2/software>. Citado na página 16.

LEGO. *LEGO® MINDSTORMS® — Invente um robô*. 2022. <https://www.lego.com/pt-br/themes/mindstorms>. Citado na página 16.

MANSO, A.; OLIVEIRA, L.; MARQUES, C. G. Portugol ide–uma ferramenta para o ensino de programação. In: *PAEE'2009—Project Approaches in Engineering Education—Guimaraes*. [S.l.: s.n.], 2009. Citado 3 vezes nas páginas 9, 12 e 20.

MARTINS, C. X. Fábrica inorgânica: Aplicativo educacional para dispositivo móvel. November 2022. Citado na página 18.

MDHR, S. *Cuphead*. 2022. <https://cupheadgame.com/>,. Citado na página 21.

MEDINA, M.; FERTING, C. *Algoritmos e programação: teoria e prática*. [S.l.]: Novatec Editora, 2006. Citado na página 10.

MICROSOFT. *Forza Horizon*. 2022. <https://forza.net/>. Citado na página 24.

MOBIUS. *Outer Wilds*. 2022. <https://www.mobiusdigitalgames.com/outer-wilds.html>. Citado na página 13.

MOREIRA, M. A. *Teorias de aprendizagem*. [S.l.]: Editora pedagógica e universitária São Paulo, 1999. v. 2. Citado na página 11.

MOTA, F. P. et al. Desenvolvendo o raciocínio lógico no ensino médio: uma proposta utilizando a ferramenta scratch. In: *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*. [S.l.: s.n.], 2014. v. 25, n. 1, p. 377. Citado na página 8.

MUNDO, B. N. *Tetris: a dramática história de como 'o maior jogo de todos os tempos' foi criado e deixou a União Soviética*. 2023. <https://www.bbc.com/portuguese/articles/cekdg02gxv0o#:~:text=Mais%20de%20um%20bilh%C3%A3o%20de,c%C3%B3pias%20em%20todo%20o%20mundo.>,. Citado na página 14.

NINTENDO. *Tetris99*. 2022. <https://www.nintendo.com/pt-br/store/products/tetris-99-switch/>,. Citado na página 15.

NINTENDO. *Tricky Towers*. 2022. <https://ec.nintendo.com/NZ/en/titles/70010000003655>. Citado 2 vezes nas páginas 2 e 16.

PAULA, L. Q.; JÚNIOR, D. P.; FREITAS, R. L. A leitura e a abstração do problema no processo de formação do raciocínio lógico-abstrato em alunos de computação. *Reverte-Revista de Estudos e Reflexões Tecnológicas da Faculdade de Indaiatuba*, n. 7, 2009. Citado na página 8.

PLANK-BLASKO, D. 'from russia with fun!': Tetris, korobeiniki and the ludic soviet. *The Soundtrack*, Intellect, v. 8, n. 1-2, p. 7–24, 2015. Citado na página 14.

PLAYDEAD. *Inside*. 2022. <https://playdead.com/games/inside/>. Citado na página 21.

PRÄSS, A. R. Teorias de aprendizagem. *ScriniaLibris. com*, p. 23, 2012. Citado na página 10.

ROBOCODE. *Robocode*. 2022. <https://robocode.sourceforge.io/>. Citado na página 8.

SCRATCH. *Scratch*. 2022. <https://www.scratch.mit.edu/projects/editor/?tutorial=getStarted>. Citado 2 vezes nas páginas 8 e 16.

SILVEIRA, J. de A. Construcionismo e inovação pedagógica: uma visão crítica das concepções de papert sobre o uso da tecnologia computacional na aprendizagem da criança. *THEMIS: Revista da Esmec*, v. 10, p. 119–138, 2016. Citado 2 vezes nas páginas 8 e 11.

SOLOMON, C. et al. History of logo. *Proceedings of the ACM on Programming Languages*, ACM New York, NY, USA, v. 4, n. HOPL, p. 1–66, 2020. Citado 2 vezes nas páginas 8 e 11.

SPACE, O. *Tetris 99 é anunciado e lançado de graça para assinantes do Switch online*. 2022. <https://www.outerspace.com.br/tetris-99-e-anunciado-e-lancado-de-graca-para-assinantes-do-switch-online/>. Citado 2 vezes nas páginas 2 e 15.

STORYMILL. *Squeakland A cadas do Squeak Etoys*. 2022. <http://www.squeakland.org/>. Citado na página 16.

STUDIO, N. *Gris*. 2022. <https://nomada.studio/>. Citado na página 13.

STUDIO, T. *We are developing Computing with Emil for schools and in schools*. 2022. <https://www.robotemil.com/>. Citado na página 16.

STUDIOS, M. *Ori and the Blind Forest*. 2022. <https://www.orithegame.com/>. Citado na página 21.

SUPERHOT. *Super Hot Team*. 2022. <https://superhotgame.com/>. Citado na página 13.

TECHNOLOGIES, U. *AudioSource*. 2023. Https://docs.unity3d.com/Manual/class-AudioSource.html. Citado na página 30.

TECHNOLOGIES, U. *GameObjects*. 2023. Https://docs.unity3d.com/Manual/GameObjects.html. Citado na página 29.

TECHNOLOGIES, U. *Prefabs*. 2023. Https://docs.unity3d.com/Manual/Prefabs.html. Citado na página 30.

TECHNOLOGIES, U. *Transform*. 2023. Https://docs.unity3d.com/Manual/class-Transform.html. Citado na página 30.

TECHNOLOGIES, U. *Unity Asset Store*. 2023. Https://assetstore.unity.com/. Citado na página 25.

TECHNOLOGY, M. institute of. *StarLogo TNG*. 2022. <https://education.mit.edu/project/starlogo-tng/>. Citado na página 16.

TETRIS. *How to Start a Tetris Practice Routine*. 2022. <https://tetris.com/article/90/how-to-start-a-tetris-practice-routine>. Citado 2 vezes nas páginas 2 e 14.

THEKLA. *Braid, Anniversary Edition*. 2022. <http://braid-game.com/>. Citado na página 13.

VALVE. *Counter Strike Global Ofensive*. 2022. <https://blog.counter-strike.net/>. Citado na página 24.

VIANA, G. A.; PORTELA, C. dos S. O uso de softwares educativos para introduçao de lógica de programaçao no ensino de base e superior. *Informática na educação: teoria & prática*, v. 22, n. 1, 2019. Citado na página 8.

VISUALG. *VisuAlg*. 2022. <https://visualg3.com.br/>. Citado na página 8.

WEBSTUDIO, P. *Portugol WebStudio*. 2023. Https://dgadelha.github.io/Portugol-Webstudio/. Citado na página 27.

WIKIPEDIA. *AgentCubes*. 2022. <https://en.wikipedia.org/wiki/AgentCubes>. Citado 2 vezes nas páginas 2 e 18.

WILENSKY, U. *NetLogo*. 2022. <https://ccl.northwestern.edu/netlogo/>. Citado na página 16.