CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

EDUARDA DA CONCEIÇÃO LOPES TIBURCIO

IGOR VINICIUS SANTANA DE MENEZES

VALLEY OF LOGIC: APRENDENDO LÓGICA DE PROGRAMAÇÃO ATRAVÉS DE JOGOS E INTELIGÊNCIA ARTIFICIAL

GUARULHOS

2024

EDUARDA DA CONCEIÇÃO LOPES TIBURCIO

IGOR VINICIUS SANTANA DE MENEZES

VALLEY OF LOGIC: APRENDENDO LÓGICA DE PROGRAMAÇÃO ATRAVÉS DE JOGOS E INTELIGÊNCIA ARTIFICIAL

Trabalho de graduação apresentado ao Curso de Análise e Desenvolvimento de Sistemas como requisito parcial para obtenção do Título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador (a): Profa.Esp. Jane Maria dos Santos Eberson

GUARULHOS

2024

EDUARDA DA CONCEIÇÃO LOPES TIBURCIO
IGOR VINICIUS SANTANA DE MENEZES

VALLEY OF LOGIC: APRENDENDO LÓGICA DE PROGRAMAÇÃO ATRAVÉS DE JOGOS E INTELIGÊNCIA ARTIFICIAL

Trabalho de graduação apresentado (a) ao Curso de Análise e Desenvolvimento de Sistemas como requisito parcial para obtenção do **Título de Tecnólogo** em Análise e Desenvolvimento de Sistemas.

Banca Examinadora

Orientador:_____

Profª: Esp. Jane Maria Dos Santos Eberson.

Fatec Guarulhos

Banca**:**_____

Titulação + nome completo

IES de origem

Banca:_____

Titulação + nome completo

IES de origem

Guarulhos, 28/06/2024

RESUMO

TBURCIO, Eduarda da Conceição Lopes; MENEZES, Igor Vinicius Santana de. **VALLEY OF LOGIC: APRENDENDO LÓGICA DE PROGRAMAÇÃO ATRAVÉS DE JOGOS E INTELIGÊNCIA ARTIFICIAL.** 2024. 76 p. Trabalho de Conclusão de Curso – Faculdade de Tecnologia de Guarulhos, Guarulhos.

Este Trabalho de graduação concentra-se no desenvolvimento de um jogo para ensinar os conceitos básicos de programação para crianças de 4 a 9 anos e utilizando os conceitos existentes de inteligência artificial. O jogo foi projetado para criar um ambiente de ensino compatível com a faixa etária do público-alvo, incorporando elementos interativos e aspectos visuais atrativos. Foram desenvolvidos desafios e atividades práticas que estimulam o pensamento lógico e a resolução de problemas, essenciais para a programação e para o desenvolvimento cognitivo do público-alvo. Através de pesquisas bibliográficas e observações diretas, constatou-se que um jogo consegue engajar as crianças de maneira positiva, promovendo o desenvolvimento de habilidades cognitivas importantes. Este projeto visa não apenas entreter, mas também proporcionar uma experiência de aprendizado divertida e educativa, contribuindo para a inclusão de crianças no mundo da programação de forma lúdica e eficaz. *Valley of Logic* tem como objetivo, auxiliar no aprendizado com o engajamento do público-alvo.

Palavras-chave: Programação, Crianças, Engajamento, Jogo, Educação.

ABSTRACT

This Course Completion Work focuses on developing a game to teach the basics of programming to children ages 4 to 9 and utilizing existing concepts of artificial intelligence. The game was designed to create a teaching environment compatible with the age group of the target audience, incorporating interactive elements and attractive visual aspects. Challenges and practical activities were developed that stimulate logical thinking and problem solving, essential for the programming and cognitive development of the target audience. Through bibliographical research and direct observations, it was found that a game can engage children in a positive way, promoting the development of important cognitive skills. This project aims not only to entertain, but also to provide a fun and educational learning experience, contributing to the inclusion of children in the world of programming in a playful and effective platform "Valley of Logic" aims to assist in learning with the engagement of the target audience.

Keywords: *Programming, Children, Engagement, Game, Education.*

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

Os jogos eletrônicos ultrapassaram a simples função de entretenimento e alívio do estresse, desempenhando um papel significativo no aprimoramento de habilidades como raciocínio lógico, estratégia, coordenação motora e até mesmo na educação, como uma ferramenta para adquirir conhecimento e auxiliar nos processos educacionais.

No mercado profissional atual, os conceitos de inteligência artificial estão ganhando cada vez mais relevância, com esse crescimento, é natural que esses conceitos sejam incorporados em ambientes educacionais, já que o uso de tecnologias para auxiliar o aprendizado tem conquistado uma popularização nos tempos atuais.

Este trabalho tem como objetivo principal, desenvolver uma aplicação lúdica envolvendo inteligência artificial que auxilie o aprendizado de programação. Os objetivos específicos incluem estudar os conceitos de IA existentes para a implantação dentro do projeto, a criação de um ambiente de ensino que seja compatível com a idade do público-alvo e desenvolver desafios e atividades práticas que estimulem o pensamento lógico e a resolução de problemas, fundamentais para o aprendizado de programação.

Diante desse panorama, o presente estudo propõe a criação de uma plataforma lúdica que auxilie no processo educacional e que tem o objetivo de proporcionar aprendizado em lógica de programação para crianças de 4 a 9 anos. O principal objetivo da plataforma é fornecer conhecimentos básicos em seus primeiros anos de formação para que possam desenvolver projetos simples e, gradualmente, despertar o interesse pela programação, estimulando-os a buscar uma aprendizagem contínua.

2. FUNDAMENTAÇÃO TEÓRICA

Este capítulo visa tratar os aspectos da literatura e fontes bibliográficas utilizadas como base para a confecção deste trabalho, analisando todos os conceitos utilizados para o desenvolvimento da aplicação.

2.1 Revisão Sistemática da Literatura

Com o objetivo de revisão da literatura, as questões abaixo foram utilizadas como norteadoras:

1. Quais são os principais benefícios dos jogos no processo de aprendizado?
2. Em que contextos educacionais os jogos são mais eficazes como ferramentas de ensino?
3. Quais são as características dos jogos que melhor promovem a aprendizagem dos alunos?
4. O conceito de jogos apoiando o aprendizado é uma tendência no mundo atual?

Para destacar a busca de informações, foi criada uma *String* de busca adaptada para o contexto educacional aplicado a este trabalho de graduação. A *String* mencionada consiste em: ("Jogos educacionais" OU "Aprendizado baseado em jogos" OU "Gamificação" OU "Educação lúdica") E ("Aprendizado" OU "Ensino") E ("Benefícios" OU "Efeitos") E ("Contextos" OU "Características").

Foram consideradas as seguintes bases de dados para a revisão da literatura:

- Google acadêmico: https://scholar.google.com.br/?hl=pt
- Bdtd: https://bdtd.ibict.br/vufind/
- ScienceDirect: https://www.sciencedirect.com/

Utilizou-se a *String* criada dentro das bases de dados e com base nos artigos apresentados fazendo uso dos critérios de inclusão e exclusão, conforme o quadro 1:

Quadro 1: Critérios de inclusão e exclusão

Critério	Id	Descrição
Inclusão	CI1	Trabalhos que apresentem aplicações de tecnologias para auxílio do ensino.

Inclusão	CI2	Trabalhos que apresentem conceitos de inteligência artificial voltadas para o meio educacional.
Inclusão	CI3	Trabalhos que apresentem jogos como uma maneira de aprender.
Inclusão	CI4	Trabalhos que apresentem propostas e conceitos do ensino de programação.
Exclusão	CE1	Trabalhos que apresentem conceitos de inteligência artificial para o meio organizacional
Exclusão	CE2	Trabalhos que apresentem outras maneiras intuitivas de aprender, que não sejam jogos

Fonte: Dos Autores, 2024

2.2 Resultados Utilizando os Critérios de Inclusão e Exclusão.

Após definir os critérios de inclusão e exclusão, houve o refinamento da pesquisa apontando os artigos abaixo como resultados, os quais serviram para complementar a pesquisa juntamente aos demais livros sobre o assunto.

- MEEGA+: Um Modelo para a Avaliação de Jogos Educacionais para o ensino de Computação. (2019)
- A utilização de jogos educacionais digitais como proposta de metodologia ativa de ensino para uma aprendizagem significativa na educação básica. (2020)
- Uso de laboratórios remotos integrados a jogos digitais para o ensino de física. (2019)
- Jogos didáticos nas aulas práticas de Anatomia Humana como recurso complementar para estudo no ensino superior. (2022)
- *Comparing effectiveness of educational video games of different genres in computer science education*. (2023)
- *Timing of learning supports in educational games can impact students' outcomes*. (2022)

Ao analisar a revisão sistemática de literatura, conclui-se que, ao utilizar os critérios de inclusão e exclusão corretos, foram encontradas as fontes bibliográficas

mais adequadas ao escopo do trabalho, facilitando a busca por materiais que sustentem a fundamentação teórica do projeto.

2.3 Os Jogos

Durante os últimos anos, houve uma evolução do universo dos videogames e como essa cultura ocupa um espaço cada vez maior na sociedade. Além de ser uma fonte de diversão e entretenimento para muitas pessoas, de acordo com a *E-commerce* Brasil “indústria de jogos eletrônicos já emprega mais de quatro mil pessoas e movimenta R$ 900 milhões de reais por ano no país” (*E-commerce* Brasil, 2016). Segundo dados da Unifebe “A indústria de jogos eletrônicos já emprega mais de 4 mil pessoas no Brasil e chegou a faturar em 2015, cerca de US$ 1,28 bilhão no mundo.” (Unifebe, 2018).

Em 1923, Ricciotto Canudo criou o **Manifesto das Sete Artes**, que serviu como métrica para a categorização de certas atividades como arte (CANUDO, 1923). Após a estipulação das sete artes, que são: pintura, escultura, arquitetura, música, poesia, eloquência e dança, outras atividades foram reconhecidas como arte, sendo categorizadas após conclusões de um senso geral de determinadas massas em nossa sociedade, incluindo assim os videogames como a décima arte.

Já foi provado por meio de diversos estudos que os videogames melhoram a criatividade, interpretação, leitura, busca por soluções alternativas, sociabilidade e estimulam o bem-estar, o que melhora problemas relacionados a baixa autoestima e depressão. De acordo Olímpia e Abreu (2022), em um artigo publicado pelo Instituto Federal Goiano (IFG), "considerando todos os apontamentos, é notório que no processo de alfabetização faz-se necessário usar os jogos digitais como uma ferramenta lúdica para o desenvolvimento infantil" (OLÍMPIA; ABREU, 2022). Algumas instituições de ensino dizem que os jogos podem ajudar a treinar a agilidade no raciocínio e nos reflexos. publicado por Karine e Murilo (2016), que teve o intuito de mostrar como os jogos podem auxiliar no desenvolvimento cognitivo das crianças, “utilização de jogos digitais durante os atendimentos focais de estimulação cognitiva no ambiente escolar pode trazer benefícios significativos” (KARINE; MURILO, 2016).

Baseado nos estudos citados, pode-se afirmar que os *videogames* podem exercitar áreas do cérebro mais do que algumas atividades de lazer. O ato de jogar *videogames* traz diversos benefícios, fazendo com que o jogador possa conhecer

tradições e culturas de outros países, aprender mitologias, desenvolver o gosto pela leitura, traduzir diálogos, assim melhorando e treinando outros idiomas como o inglês, porém, os *videogames* devem ser uma fonte de lazer, que por sua vez precisa ser utilizada de maneira moderada por seus consumidores, Nolan Bushnell, criador do Atari, afirma que "as pesquisas mostram que as crianças que jogam com moderação têm melhor desempenho se comparadas às que não jogam" (BUSHNELL, 2006).

Segundo um levantamento feito pela NewZoo, uma das principais condutoras de pesquisas sobre a indústria dos games no mundo, "o mercado global de jogos gerará receitas de US$ 183,9 bilhões, com crescimento anual de +0,5%" (NEWZOO, 2024).

Estes dados mostram o quão popular está se tornando a indústria de jogos ao longo dos anos. Porém, é quase dubitável olhar para este crescimento constante e não nos indagarmos em que ponto esta indústria atingiu tal relevância.

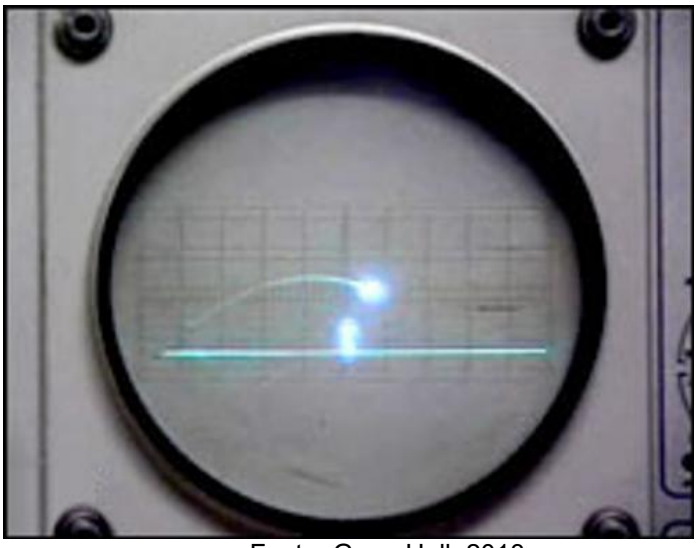
2.3.1 História dos Jogos

Os jogos tiveram origem no século XVI na Roma e na Grécia com o propósito de ensinar letras. Contudo, com o início do Cristianismo, esse interesse decaiu, pois como eles tinham um ideal de educação disciplinadora, os jogos começaram a ser vistos como ofensivos e imorais. De acordo com Brougère (2004): “Antigamente, a brincadeira era considerada, quase sempre como fútil, ou melhor, tendo como única utilidade a distração, o recreio, e na pior das hipóteses, julgavam-na nefasta” (BROUGÈRE, 2004, p. 24).

Logo após a fase do Renascimento, o jogo foi dissociado dessa visão de censura e passou a fazer parte da vida de crianças, jovens e adultos, como meio de diversão, distração, passatempo e até como meio de educação.

Já os videogames têm uma história intrigante que remonta a meados do século XX, quando os primeiros experimentos com computação gráfica e interatividade começaram a ganhar forma. Um dos primeiros datados e registrados foi na época da guerra fria, feito por Willian Higinbotham, que recebeu o nome de *Tennis for Two*, um *videogame* exposto em uma tela de 15 pixels projetado para ser processado em um computador analógico, Figura 1.

Figura 1: Tennis for Two em Computador Analógico



Fonte: GameHall, 2013

A partir da criação desse jogo, outros desenvolvedores puderam usá-lo como inspiração para criação de diversos, como é o exemplo do *Pong* (Figura 2) que se tornou anos depois o primeiro jogo a gerar lucro em toda a história, dando início a indústria de produção de jogos.

Figura 2: Pong

Fonte: BBC News Brasil, 2022

Em 1966, Ralph Baer, um alemão refugiado da Segunda Guerra Mundial, considerado hoje como o pai dos *Consoles*, desenvolveu uma forma onde pudesse ser processado os jogos eletrônicos, tendo a televisão como um meio de vinculação.

Desde então os jogos eletrônicos estão evoluindo de simples jogos para experiências imersivas que capturam a imaginação de bilhões de pessoas em todo o mundo.

Em 1972 foi lançado *Odyssey* (Figura 3), o primeiro *console* doméstico que foi reconhecido mundialmente pela indústria, ele também pode ser reconhecido como *Magnavox Odyssey*.

Figura 3: Magnavox Odyssey

Fonte: BBC News Brasil, 2023

Anos após a criação do *Tennis for Two*, mais precisamente na década de 1970, a humanidade presenciou a popularização dos arcades em estabelecimentos, esta era foi responsável por introduzir jogos bem conhecidos como é o caso do *Pong* (Atari) e o *Space Invaders* (Taito), Figura 4 cativando milhares de pessoas a se aventurarem pelos arcades da época. Naquela época os jogos eram mais simples, porém voltados para o cenário competitivo, graças a isto, muitas competições de videogames que foram criadas na época, ajudaram na popularização dos videogames na sociedade.

Figura 4: Space Invaders

Fonte: Daniel Gularte, 2020

Após a criação dos arcades obter um estrondoso sucesso, as empresas decidiram desenvolver consoles para utilização caseira, atraindo assim um número ainda maior de consumidores para este nicho do entretenimento, pois diversas pessoas na época não queriam abandonar o conforto dos seus lares para jogarem arcades em estabelecimentos.

O lançamento do Atari 2600 na década de 1980 ficou marcado como início dessa era de consoles domésticos, mas os verdadeiros responsáveis por popularizar a ideia foram os lançamentos do NES (1985), Figura 5, mais conhecido popularmente no Brasil pelo nome de nintendinho e o lançamento do Sega Genesis (1989) mais conhecido como mega drive.

Figura 5: Nintendo NES

Fonte: Murillo Benevides, 2022

Na década de 1990, os videogames avançaram tecnologicamente com a introdução de gráficos em 3D e o lançamento do CD-ROM, permitindo jogos maiores e mais imersivos (BURKE, 2023).

A Era Digital dos jogos destaca-se pela digitalização e a diversidade de plataformas de desenvolvimento, permitindo que desenvolvedores indies competissem com grandes empresas, democratizando o mercado (BURKE, 2023).

A trajetória da indústria de jogos mostra avanços tecnológicos sincronizados, gerando expectativas para o futuro, como jogos totalmente baseados em realidade virtual ou criados por inteligência artificial (BURKE, 2023).

2.4 Jogos Educativos

Com o crescimento dos jogos dentro da indústria do entretenimento e o seu sucesso entre consumidores dessa indústria, os desenvolvedores pensaram em maneiras de aproveitar a atenção das crianças aos *videogames* e idealizaram a criação de jogos com abordagem lúdica e que aflorasse cada vez mais à vontade das crianças de buscar novos conhecimentos, de acordo com Mattar (2010), as pessoas que estão em contato com a tecnologia em seu cotidiano possuem uma expectativa de aprendizado maior (MATTAR, 2010).

Jogos educativos são ferramentas poderosas que combinam diversão e aprendizado eficaz. Projetados para estimular o pensamento crítico e resolver problemas, também desenvolvem habilidades específicas, proporcionando uma experiência envolvente (KIRRIEMUIR; MCFARLANE, 2004).

Jogos educativos dinamizam a aprendizagem, especialmente para jovens, ao usar gamificação com desafios, recompensas e progressão de níveis, incentivando a persistência e a motivação intrínseca (GEE, 2003).

Jogos educativos são adaptáveis para diversas idades e habilidades, oferecendo aprendizado personalizado em temas variados como matemática, ciências e habilidades socioemocionais (PRENSKY, 2001).

Jogos educativos são acessíveis em plataformas digitais como computadores, tablets e smartphones, facilitando o aprendizado interativo em diversos contextos (VAN ECK, 2006).

Jogos educativos estimulam competências essenciais como pensamento crítico, colaboração, criatividade e habilidades tecnológicas, sendo ferramentas valiosas para educadores integrarem ao currículo escolar (SHAFFER, 2006).

Em resumo, os jogos educativos representam uma abordagem inovadora e eficaz para promover a educação de forma envolvente e significativa. Ao combinarem diversão e aprendizado, esses jogos têm o potencial de transformar a maneira como as pessoas adquirem conhecimento e desenvolvem habilidades importantes ao longo da vida.

Um exemplo de jogo educativo é o jogo *Mickey's Safari in Letterland*, Figura 6, publicado pela *hi tech expressions* em 1993. O jogo era um *plataformer* que tinha como intuito o ensino de alfabetização para crianças de uma maneira intuitiva.

Figura 6: Mickey's Safari in Letterland

Fonte: MobyGames, 2007

2.5 Inteligência Artificial

Analisando a área de estudo dedicada à inteligência artificial, encontramos uma vasta quantidade de informações. Atualmente, diversas aplicações utilizam conceitos de inteligência artificial (IA) para auxiliar em várias tarefas, tornando seu uso cada vez mais rotineiro e comum no cotidiano humano. Isso leva à seguinte questão: quando surgiram, de fato, os conceitos que permitiram à inteligência artificial se tornar um campo amplamente desenvolvido?

De acordo com McCarthy (1956), uma IA pode ser definida como "a ciência e a engenharia de produzir máquinas inteligentes" (McCARTHY, 1956). Esta foi a

primeira vez que foi mencionado o conceito de IA de uma maneira próxima a que é utilizada atualmente, esta definição foi dita pelo professor em 1956 durante uma conferência, e mesmo com o passar dos anos e o surgimento de novas tecnologias e a fundamentação de novos conceitos. Esta é a definição mais utilizada nos dias de hoje devido a sua alta representação do que seria uma Inteligência artificial em sua mais pura essência.

2.5.1 História da IA

Segundo AUGUSTO (2022), a história da IA remonta a séculos passados, desde os conceitos iniciais até o século XIX com a Máquina Analítica de Charles Babbage e o Teste de Turing de Alan Turing no século XX.

Nos anos 50 e 60, surgiram as primeiras pesquisas formais em IA, com a criação de programas que podiam jogar xadrez e resolver problemas matemáticos (Instituto de engenharia, 2018).

Segundo Awari (2023), a década de 70 trouxe avanços como o sistema especialista MYCIN. Nos anos 80, o boom da IA foi seguido pelo "inverno da IA" devido a expectativas não cumpridas.

Nos anos 90 e 2000, o foco da IA mudou para algoritmos de aprendizado de máquina, como redes neurais, levando a avanços em reconhecimento de fala, visão computacional e IA geral (Instituto de Engenharia, 2018).

2.5.2 Expectativas para IA

Atualmente, a IA permeia muitos aspectos de nossa vida, desde assistentes virtuais até carros autônomos. A pesquisa continua avançando em direção a IA mais robusta, ética e responsável, abordando desafios como viés algorítmico e transparência.

As expectativas para a inteligência artificial no futuro são imensas e abrangentes. Espera-se que a IA continue a transformar profundamente diversos setores, como saúde, educação, transporte e entretenimento. Segundo Boden (2020), "a IA impulsionará a próxima onda de inovação e crescimento econômico, criando novas indústrias e transformando as existentes" (BODEN, 2020, p. 59).

O desenvolvimento de IA robusta, capaz de raciocinar, aprender e se adaptar de forma autônoma, é uma meta a ser alcançada. Além disso, a integração ética e responsável da IA na sociedade será fundamental para garantir seus benefícios enquanto se mitigam riscos potenciais, como viés algorítmico e questões de privacidade. O futuro da inteligência artificial promete avanços fascinantes e desafios instigantes à medida que exploramos suas capacidades e limitações.

2.6 Lógica de Programação

Segundo PIUBELLO (2023), a lógica de programação é essencial para desenvolver *software*, permitindo a criação de algoritmos estruturados e eficientes através de diagramas de fluxo, pseudocódigo e linguagens de programação.

2.6.1 Origem da Lógica

A lógica se estabeleceu desde os primeiros conceitos do pensamento lógico, essencial tanto profissionalmente quanto em tarefas cotidianas como trocar um pneu ou fritar um ovo. O constante aprimoramento do pensamento lógico é fundamental para resolver questões simples do dia a dia (COPI; COHEN; McMAHON, 2014).

A origem do pensamento lógico remonta ao século IV a.C. com Aristóteles, que criou a lógica formal, base para o pensamento lógico atual (D'OTTAVIANO e FEITOSA, 2003).

A lógica, como ciência autônoma, estuda as leis do pensamento humano de forma estruturada e abstrata, sem considerar aspectos materiais, sendo denominada lógica formal. Essa disciplina foi fundamental para áreas como filosofia e matemática (CHURCH, 1996).

Na Figura 7, podemos ver um exemplo da lógica formal idealizada por Aristóteles.

Figura 7: Exemplo de Lógica Formal

Fonte: Victor França, 2014

De acordo com MARIA e ARAUJO (2003) a partir da criação da lógica formal, diversos tipos de lógicas foram desenvolvidos utilizando o pensamento estruturado, como a lógica matemática a lógica computacional e entre outras.

2.6.2 Lógica Computacional e Programação

A evolução das linhas de pensamento lógico aplicou-se em várias áreas profissionais, resultando na lógica de programação com conceitos computacionais (TASSINARI, 2012).

2.6.2.1 Linguagens de Programação

A lógica de programação, originada com Ada Lovelace e o algoritmo para a máquina analítica de Babbage em 1843, sequência pensamentos de forma estruturada para alcançar objetivos, sendo fundamental para todas as linguagens de programação atuais (FAGUNDES, 2023).

Segundo FACUNDES (2023), Konrad Zuse criou a linguagem Plankalkül na década de 40, para criar procedimentos e armazenar código em tarefas de rotina. Nessa mesma época, surgiu a linguagem Assembly, utilizada em calculadoras automáticas para operações básicas de computação.

No mesmo ano da criação da linguagem Assembly, surge a primeira linguagem de alto nível, batizada pelo nome de *Shortcode* pelo Militar John

McCauley, que implantou a mesma nos computadores Binac e Univac (FACUNDES, 2023).

Conforme FACUNDES (2023), em 1952, Alick Glennie desenvolveu o *Autocode*, a primeira linguagem compilada para o computador Mark 1, utilizando um compilador para linguagem de máquina.

Cinco anos após o *Autocode*, John Backus criou o Fortran para cálculos científicos avançados, tornando-se uma das linguagens mais antigas ainda em uso. Na mesma década, Lisp foi desenvolvida para manipulação de listas e Cobol para processamento de negócios (FACUNDES, 2023).

Segundo FACUNDES (2023), nas décadas de 60 e 70 surgiram paradigmas como programação imperativa (C), programação estruturada (Pascal), programação lógica (Prolog), e programação declarativa (SQL), este último utilizado em bancos de dados.

Nos anos 80, C++ se destacou nas universidades com programação orientada a objetos. Na década de 90, *Python* ganhou popularidade na internet, enquanto Java, JavaScript e PHP dinamizaram o desenvolvimento de aplicativos e páginas *web*, com C# focado na plataforma *Windows* (FACUNDES, 2023).

3. METODOLOGIA

O presente texto descreve a metodologia de pesquisa bibliográfica proposta para um Trabalho de Graduação (TG) que tem como objetivo investigar o uso de jogos como ferramenta de aprendizado.

Segundo Marconi e Lakatos (2018), a metodologia cientifica é: "uma investigação que tem por objetivo a obtenção de conhecimento a partir de material já publicado, tais como livros, artigos científicos, teses, dissertações, documentos oficiais e outros" (MARCONI e LAKATOS 2018, p. 18). A pesquisa bibliográfica é uma técnica de pesquisa essencial para o desenvolvimento de trabalhos científicos, pois permite ao pesquisador obter acesso a informações referentes ao seu respectivo tema.

A pesquisa bibliográfica que serviu como base para a implementação dos conceitos referentes a gamificação e aos métodos de ensino mais atualizados que utilizem jogos como ferramentas ativas dentro do desenvolvimento cognitivo das pessoas inseridas no contexto acadêmico.

Após a coleta dos dados, é necessário realizar uma análise rigorosa e sistemática. A análise pode envolver a organização dos dados coletados, a identificação de padrões e temas emergentes, e a busca por relações entre as informações obtidas.

Os resultados serão analisados e discutidos para que se criem uma relação com os objetivos da pesquisa e da revisão bibliográfica. As descobertas são comparadas com estudos anteriores, e as implicações e limitações da pesquisa são analisadas.

4. DESENVOLVIMENTO

Neste capítulo serão abordadas todas as questões referentes a todo o desenvolvimento do projeto, desde seu escopo até seus processos de desenvolvimento.

4.1 *Valley of Logic*

Valley of Logic é um jogo educativo desenvolvido com o intuito de auxiliar o aprendizado de lógica de programação para crianças de 04 a 09 anos de idade. Este jogo pretende proporcionar uma experiencia interativa e divertida onde os jogadores iniciarão uma jornada fascinante com Luke, uma raposa que representa sabedoria, coragem e astucia.

No início do jogo, os jogadores irão enfrentar problemas de lógica simples, tais como identificar padrões, completar sequências e resolver quebra-cabeças. Essas atividades têm como objetivo introduzir conceitos básicos fundamentais da lógica de forma descontraída e divertida. À medida que a criança vai avançando, há um aumento gradual na dificuldade dos desafios, envolvendo estrutura de decisões e lógica condicional.

O jogo se perfaz em um combate com o inimigo central, que representa os desafios mais complexos da lógica da programação. Nessa fase a criança será incentivada a aplicar as habilidades e conhecimentos adquiridos ao longo do jogo para superar mais um obstáculo, dessa forma, a última fase apresenta um nível de dificuldade elevado, preparando a criança para uma pensar de forma estruturada e criativa.

4.1.1 Identidade Visual

A identidade visual de um jogo refere-se a um conjunto de elementos visuais que definem a aparência e estilo do jogo. Isso inclui o design dos personagens, ambientes, paleta de cores, tipografia ou qualquer outro elemento que faça parte da experiencia visual.

4.1.1.1 Logo e Tipografia

O logo da aplicação (Figura 8) é uma representação vibrante de um cenário de uma floresta, com a raposa Luke, o personagem principal, posicionado no centro. O design remete à ideia principal da tradução do nome e da história do jogo **Vale da Lógica,** reunindo a essência da aventura e da procura pela sabedoria que Luke enfrentará. Através da jornada, o ***Valley of Logic*** pretende não apenas auxiliar na lógica de programação, mas também busca incentivar um amor pelo aprendizado e pela resolução de problemas.

Figura 8: Logo do Valley of Logic

Fonte: Dos Autores, 2024

A tipografia (Figura 9) do nome do jogo é um conjunto de imagens de letras que formam ***Valley of Logic***.

Figura 9: Tipografia do Jogo

Fonte: Dos Autores, 2024

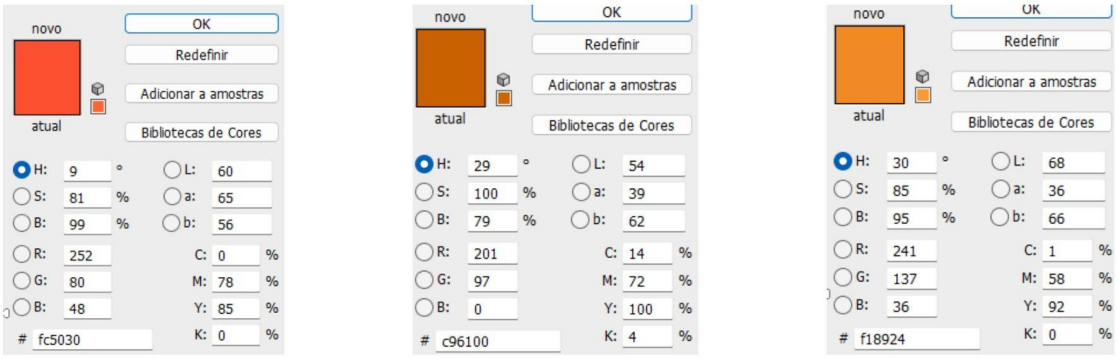
4.1.1.2 Paleta de Cores

A paleta de cores para o nome, logo e personagem principal foi escolhida para criar uma atmosfera envolvente e para complementar o estilo *pixel art.*

O logo é composta de três elementos: o personagem principal, a floresta e o nome do jogo. Cada elemento tem suas cores e variações predominantes. A seguir serão detalhados cada elemento e cor.

A cor laranja foi escolhida por sua capacidade de transmitir energia e aventura, refletindo a vivacidade de Luke. Além disso, o laranja busca remeter também a cor dos pelos da raposa vermelha. Na Figura 10 são apresentadas as variações de laranja que foram utilizadas.

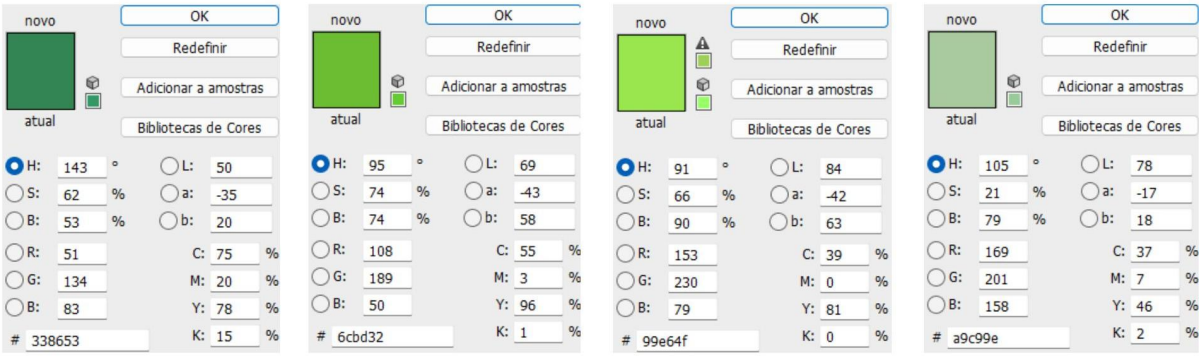
Figura 10: Tons do Laranja



Fonte: Dos Autores, 2024

A cor verde foi escolhida por ser característica da vegetação predominante em uma floresta. Na Figura 11 são apresentadas as variações utilizadas.

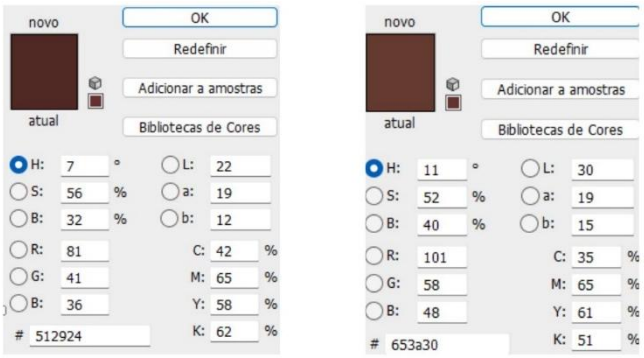
Figura 11: Tons de Verde



Fonte: Dos Autores, 2024

A cor marrom foi escolhida para simbolizar a terra presente na natureza. Na Figura 12 são apresentados os tons utilizados.

Figura 12: Tons de Marrom



Fonte: Dos Autores, 2024

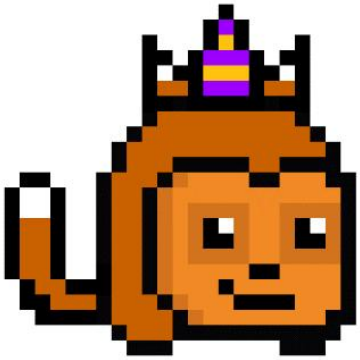
4.1.1.3 Inspirações e Referências

O ***Valley of Logic*** é composto por 4 personagens, sendo um o personagem principal e 3 inimigos. Para cada personagem foi utilizado uma referência diferente com tabelas de cores diversas que melhor combinasse com o motivo da escolha do inimigo e do cenário.

Para a criação do personagem principal, Luke (Figura 13), utilizou-se como base, uma raposa devido a todas as características que ela representa e que seriam

vitais para o desenvolvimento do jogo e simboliza caraterísticas que esse jogo espera desenvolver nas crianças, tais como sabedoria, coragem e inteligência.

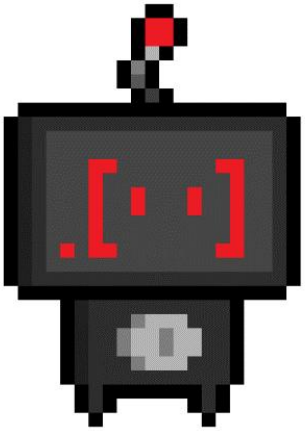
Figura 13: Luke



Fonte: Dos autores, 2024

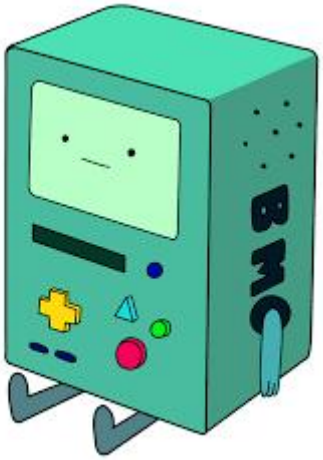
Para o inimigo Vetor (Figura 14) foi pensado em deixar claro a ideia do colchete como parte do rosto do personagem, após uma pesquisa realizada, foram utilizados conceitos do personagem BMO (Figura 15) do desenho da *Cartoon Network* chamado Hora de Aventura como inspiração.

Figura 14: Inimigo Vetor



Fonte: Dos Autores, 2024

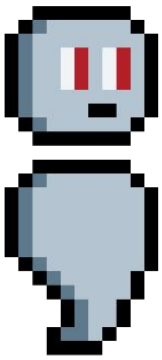
Figura 15: BMO



Fonte: Fandom, 2024

Para o inimigo Ponto e Vírgula (Figura 16) foi idealizado a ideia do ponto e vírgula utilizados para terminar linhas de código, sua estrutura clara é a do próprio sinal.

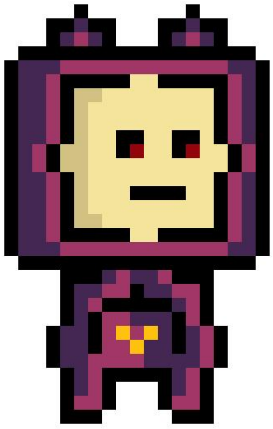
Figura 16: Ponto e Vírgula



Fonte: Dos Autores, 2024

Por último, para a criação do terceiro inimigo O Chave (Figura 17) foi elaborado um personagem que contasse o elemento chave em seu rosto. A confecção de seu corpo e cores foi baseada no personagem Sentinela (Figura 18) da Marvel.

Figura 17: O Chave



Fonte: Dos Autores, 2024

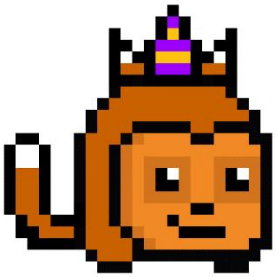
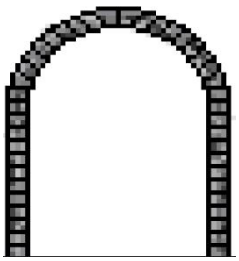
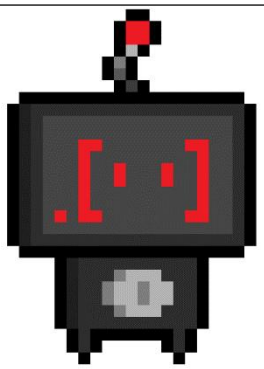
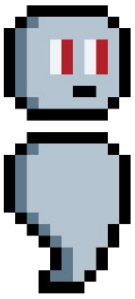
Figura 18: Sentinela

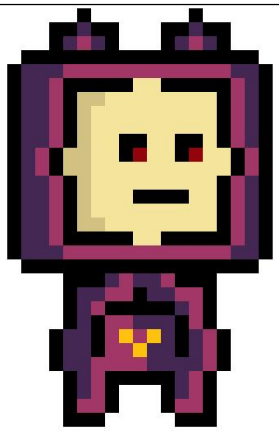
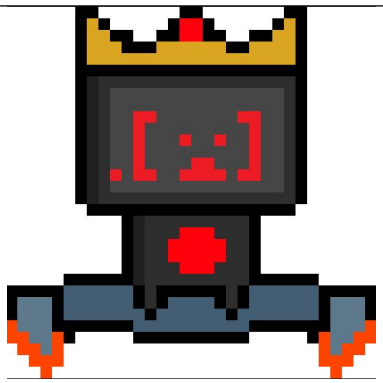
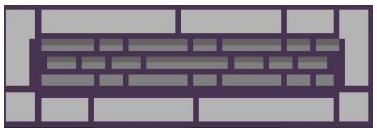
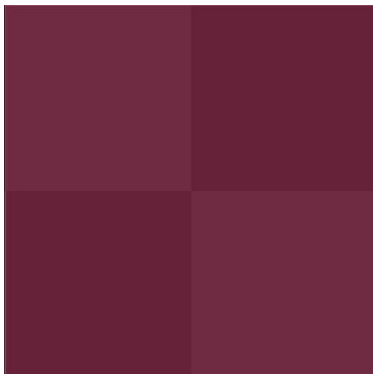
Fonte: Guiadosquadrinhos, 2007

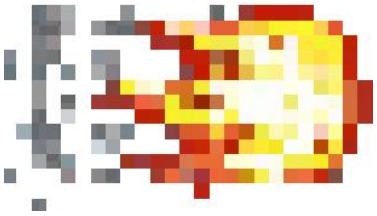
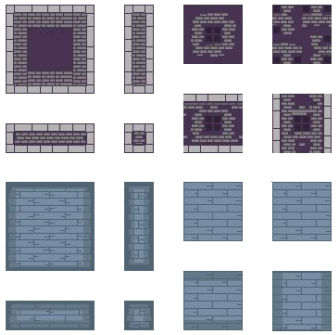
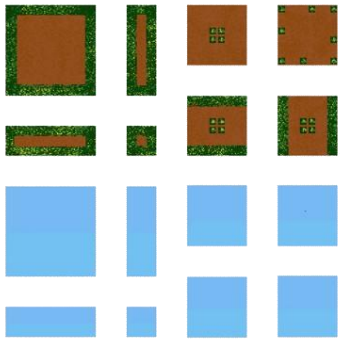
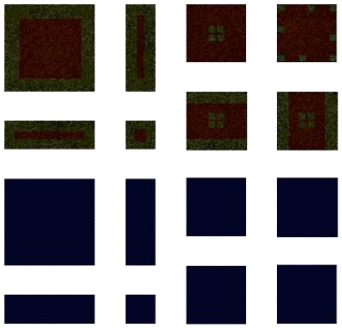
4.2 Os Sprites

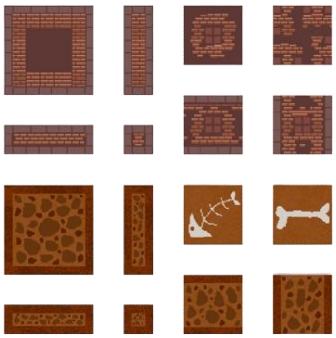
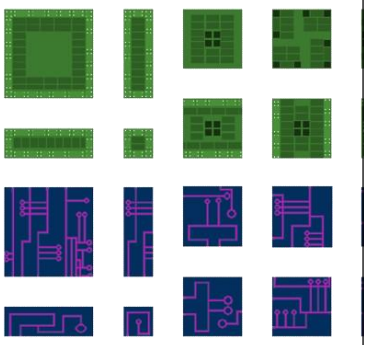
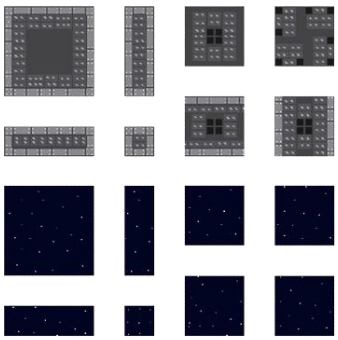
Os *sprites* são elementos gráficos no mundo do jogo que podem representar personagens, objetos, efeitos visuais e outros elementos. São normalmente usados em jogos 2D, mas em alguns casos podem ser utilizados em elementos específicos em jogos 3D. Geralmente os *sprites* precisam ser animados para darem vida aos elementos do jogo, para realizar esse feito é necessário dividir os *sprites* em frames diferentes e criar uma sequência de animação, temos os *sprites* do jogo indicados no Quadro 2.

Quadro 2: Sprites

ID	Sprite	Descrição
Player		*Sprite* padrão do personagem principal, o personagem variações de sprites para cada ação realizada pelo personagem principal, as ações são: atirar, andar, sair, pular, entrar, tomando dano, cair, morrer, impulso e parado.
Portal		*Sprite* do estado estático, ao todo, o portal tem 3 variações de sprites, eles são: estático, abrindo e fechando.
Inimigo Vetor		*Sprite* padrão do inimigo vetor, ao todo o inimigo vetor possui 4 variações de sprites: parado, andando, levando dano e morrendo.
Inimigo Ponto e Vírgula		*Sprite* padrão do inimigo ponto e vírgula, ao todo esse inimigo possui 4 variações de sprites: parado, andando, levando dano e morrendo.

Inimigo Chave		*Sprite* padrão do inimigo chave, ao todo esse inimigo possui 4 variações de sprites: parado, andando, levando dano e morrendo.
Chefe Vetor		*Sprite* padrão do chefe da primeira vila.
Plataforma móvel		*Sprite* das plataformas móveis durante o jogo.
Sprite chão		*Sprite* do objeto do chão que é responsável por representar o chão as paredes e teto do cenário, esta sprite é invisível para o jogador.

Coração		*Sprite* utilizado para representação da vida do personagem principal durante o jogo.
Shoot		*Sprite* utilizado para a funcionalidade shoot.
Caverna Moderna		*Sprite* do cenário da caverna moderna.
Grama Dia		*Sprite* do cenário externo de dia.
Grama Noite		*Sprite* do cenário externo de noite.

Caverna de Terra		*Sprite* do cenário da caverna pré-histórica.
Tecnologia		*Sprite* do cenário dentro de um computador.
Espaço		*Sprite* do cenário no espaço.

Fonte: Dos Autores, 2024

4.3 Engine e Ferramentas

Para o processo de desenvolvimentos do jogo, foram utilizadas algumas ferramentas para auxiliar nos aspectos de criação da aplicação.

4.3.1 Engine

Uma *engine* de jogo é um dos componentes vitais presentes durante o desenvolvimento de jogos digitais, servindo como o núcleo que impulsiona a experiência interativa para os jogadores. Essa poderosa ferramenta oferece um

conjunto abrangente de recursos e funcionalidades que permitem aos desenvolvedores a possibilidade de criar, projetar e implementar jogos de forma eficiente e eficaz.

Uma *engine* é composta por uma variedade de sistemas interconectados que abrangem desde gráficos e física até aspectos de inteligência artificial e gerenciamento de recursos. Estes sistemas trabalham em conjunto para criar um ambiente coeso no qual a jogabilidade pode florescer.

Uma *engine* de jogo geralmente oferece suporte a renderização de jogos em 2D e/ou 3D, permitindo aos desenvolvedores criarem mundos visuais deslumbrantes.

Ao escolher em qual *engine* que seria utilizada no projeto, foi levado em consideração, a escolha de uma engine que utilizasse conceitos simples para a confecção de jogos e que tivesse recursos para lidar com jogos de plataforma 2D que foi o gênero de jogo escolhido durante o planejamento da aplicação e que também pudesse dar suporte para outras plataformas como dispositivos móveis.

Levando todos estes requisitos em consideração optou-se pela utilização da *engine GameMaker* (Figura 19), devido disposição vasta de recursos para jogos de plataforma e da sua possibilidade de portabilidade para outras plataformas.

Figura 19: Logo GameMaker

Fonte: Make Indie Games, 2023

Um grande exemplo de um jogo desenvolvido com a *engine* da *GameMaker* é o famoso jogo de RPG (*Role-playing game*) chamado *Undertale* (Figura 20).

Figura 20: Logo do Undertale

Fonte: Undertale, 2015

4.3.2 Recursos Visuais

Pensando também nas opções para os recursos visuais do jogo, optou-se por desenvolver um jogo com traços no estilo de *pixel art,* este estilo foi escolhido, devido a popularidade de jogos de plataforma 2D com estes traços e por conta de sua simplicidade e fácil identificação visual, tendo em mente que o público-alvo desta aplicação serão crianças que estão passando pelo processo de aprendizagem.

Após resolver todas as questões referentes a identidade visual que o jogo carregaria, foi necessário realizar a procura de ferramentas para desenvolver todos os componentes visuais do jogo que são denominadas de ***Sprites***. Ao realizar esta busca encontrou-se a ferramenta *Pixilart.*

O *Pixilart* (Figura 21) é um site que disponibiliza ferramentas para todo o processo de confecção de *sprites*, além de oferecer uma vasta quantidade de *templates* para auxiliar durante a criação, também escolhemos o *Pixilart* devido a sua facilidade e praticidade no momento da sua utilização.

Todos os recursos visuais utilizados durante a criação deste projeto foram desenvolvidos dentro do *Pixilart.*

Figura 21: Logo do Pixilart

Fonte: Pixilart, 2023

4.3.3 Recursos de Documentação

Para a parte de diagramação utilizou-se a plataforma *Lucidchart* (Figura 22), que tem o intuito de ser uma ferramenta para auxiliar no processo de diagramação de projetos na criação de mapas mentais e entre outras funções essenciais para o planejamento de uma proposta.

Figura 22: Logo do Lucidchart

Fonte: Googleplay, 2024

4.4 Análise do Sistema

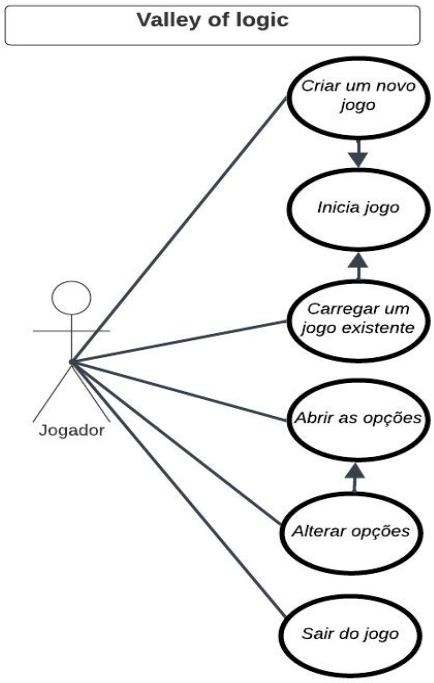
Neste tópico serão apresentadas todas as partes referentes a engenharia da aplicação e os diagramas que servem como base para realizar toda a documentação do software.

4.4.1 Diagrama de Caso de Uso

Um diagrama de caso de uso é um tipo de diagrama de modelagem utilizado em engenharia de software e sistemas para representar as funcionalidades e os requisitos de um sistema do ponto de vista do usuário. Ele é uma das ferramentas fornecidas pela UML (*Unified Modeling Language*) e é particularmente útil para visualizar as interações entre os usuários (atores) e o sistema, bem como as principais funcionalidades oferecidas pelo sistema.

Após uma análise minuciosa da aplicação, foram visualizadas as seguintes interações (Figura 23):

Figura 23: Diagrama de Contexto

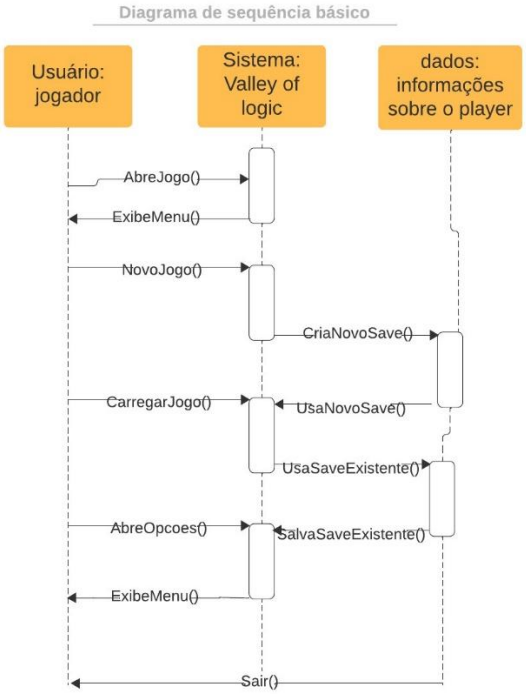


Fonte: Dos Autores, 2024

4.4.2 Diagrama de Sequência

Um diagrama de sequência é um tipo de diagrama de modelagem usado em engenharia de software e sistemas para representar a interação entre diferentes elementos de um sistema ao longo do tempo. Ele é uma das ferramentas fornecidas pela UML (*Unified Modeling Language)* e é particularmente útil para visualizar como os processos e as operações se desenrolam passo a passo em uma aplicação ou sistema. (Figura 24).

Figura 24: Diagrama de Sequência



Fonte: Dos Autores, 2024

4.4.3 Mecânicas do Jogo

O jogo de plataforma desenvolvido como parte deste trabalho tem como objetivo principal auxiliar crianças no desenvolvimento do raciocínio lógico. Para alcançar tal objetivo, foram implementadas diversas mecânicas de jogo que incentivam a resolução de problemas, o pensamento crítico e a tomada de decisões. A seguir, descrevemos as principais mecânicas utilizadas no jogo.

4.4.3.1 Movimentação do Personagem

O personagem principal do jogo pode se mover para a esquerda e direita, além de pular para superar obstáculos. A movimentação é controlada por teclas direcionais, permitindo que a criança desenvolva habilidades de coordenação motora e noção espacial.

4.4.3.2 Resolução de Puzzles

O jogo apresenta uma série de *puzzles* que precisam ser resolvidos para garantir o avanço dentro do jogo. Esses *puzzles* variam em dificuldade e exigem que a criança utilize raciocínio lógico para encontrar soluções. Além dos desafios impostos nas fases normais do jogo, haverá um desafio único para cada chefe do jogo. Esses desafios de chefes são especialmente projetados para exigir que a criança desenvolva soluções diferentes e criativas para cada problema proposto, incentivando o pensamento fora da caixa e a aplicação de habilidades adquiridas ao longo do jogo em novos contextos. Essa variedade de *puzzles* e desafios contribui para um aprendizado dinâmico e adaptativo, mantendo o interesse e a motivação da criança.

4.4.3.3 Interação com o Ambiente

O jogo inclui elementos interativos no ambiente, como elementos de decoração (tochas, armaduras e entre outros) e plataformas móveis. A criança precisa entender como esses elementos funcionam e como interagir com eles de maneira apropriada para progredir nos níveis. Essa mecânica promove o aprendizado de causa e efeito.

4.4.3.4 Feedback Imediato

O jogo oferece *feedback* imediato sobre as ações do jogador, informando se uma ação foi correta ou incorreta. Este *feedback* ajuda a criança a aprender com seus erros e ajustar suas estratégias de forma eficiente.

4.4.3.5 Progresso e Recompensas

Conforme a criança avança pelos níveis, ela ganha recompensas e desbloqueia novos conteúdos. Esse sistema de progressão e recompensas mantém a motivação e o engajamento, além de proporcionar um senso de realização e conquista para a criança. Essa sensação de conquista é fundamental, pois motiva a

criança a continuar progredindo no jogo. Além do aprendizado, o jogo oferece outras motivações, como a curiosidade para descobrir novos desafios e a satisfação de alcançar objetivos e desbloquear novos níveis e itens. Esse equilíbrio entre diversão e aprendizado garante que a criança permaneça interessada e comprometida com o jogo, promovendo um ambiente educativo eficaz e envolvente.

4.4.3.6 Tutorial e Dicas

No início do jogo e durante as fases iniciais, um tutorial guiado apresenta as mecânicas básicas e oferece dicas sobre como resolver os primeiros *puzzles*. Isso garante que a criança compreenda as regras do jogo e como utilizar as mecânicas para resolver os desafios.

Essas mecânicas foram cuidadosamente projetadas para serem educativas e divertidas de maneira simultânea, proporcionando uma experiência de aprendizado interativa e engajante. O uso dessas mecânicas visa não apenas entreter, mas também desenvolver habilidades cognitivas importantes para o desenvolvimento do raciocínio lógico nas crianças. A integração de elementos lúdicos com objetivos educacionais garante que as crianças se divirtam enquanto aprendem, incentivando a exploração, a resolução de problemas e a tomada de decisões de forma natural e envolvente. Assim, o jogo se torna uma ferramenta eficaz para o desenvolvimento cognitivo, ao mesmo tempo em que mantém o interesse e a motivação dos jovens jogadores.

4.5 IA

A Inteligência Artificial (IA) desempenha um papel fundamental no desenvolvimento de jogos eletrônicos modernos, proporcionando desafios dinâmicos e interações realistas que enriquecem a experiência do jogador. No jogo desenvolvido como parte deste trabalho, a IA foi implementada com o objetivo de criar um ambiente de aprendizado adaptativo e envolvente para as crianças. Foram utilizados algoritmos de IA para controlar a movimentação dos inimigos do jogo, fazendo com que eles tomem atitudes aleatórias durante todo o jogo. Além disso, foi implantada uma função que permite que os inimigos detectem buracos e alterem sua

direção antes de cair, evitando mortes desnecessárias e tornando o jogo mais desafiador para o jogador.

4.6 Público-alvo

Este jogo foi desenvolvido especificamente para crianças de 4 a 9 anos, uma faixa etária crucial para o desenvolvimento do raciocínio lógico e outras habilidades cognitivas. Durante esses anos, as crianças têm uma curiosidade natural e uma alta capacidade de aprendizado. De acordo com Piaget “A infância é o tempo de maior criatividade na vida de um ser humano” (PIAGET, 2001, p. 20).

4.7 Elementos do Jogo

O jogo desenvolvido neste trabalho inclui diversos elementos cuidadosamente projetados com o intuito de criar uma experiência educativa e envolvente. Esses elementos foram selecionados e implementados para estimular o desenvolvimento cognitivo, promover a resolução de problemas e manter o interesse e a motivação dos jogadores. A seguir, descrevemos os principais elementos do jogo.

4.7.1 Personagens

O jogador controla um personagem principal que se move pelo ambiente do jogo. Este personagem pode andar, pular e interagir com diversos elementos do cenário. A simplicidade dos controles garante que crianças pequenas possam jogar sem dificuldades, desenvolvendo habilidades motoras e de coordenação.

4.7.2 Inimigos

Os inimigos são controlados por algoritmos de IA e apresentam comportamentos diversos, como patrulhar áreas específicas ou evitar buracos. Esses inimigos adicionam um nível de desafio ao jogo, incentivando as crianças a pensarem estrategicamente e a planejarem suas ações, encontrando as melhores

soluções para determinadas situações fazendo com que desenvolvam sua coordenação motora e pensamento rápido.

4.7.3 Plataformas e Obstáculos

O ambiente do jogo inclui plataformas móveis, obstáculos e armadilhas que o jogador deve superar. A interação com esses elementos promove o desenvolvimento de habilidades de coordenação motora e ajuda no desenvolvimento cognitivo da criança, além de servir como um aspecto que gera dificuldade para o jogador, trazendo uma sensação de conquista durante o seu progresso no jogo.

4.7.4 Desafios de Chefes

Durante o decorrer do jogo, existem fases que culminam em um desafio único contra um chefe, este desafio exige que a criança aplique as habilidades e conhecimentos adquiridos ao longo do nível. Esses desafios são projetados para serem mais complexos, incentivando soluções criativas e estratégias avançadas.

4.7.5 Sistema de Recompensas

O jogo possui um sistema de recompensas que premia o jogador por completar níveis, resolver *puzzles* e entre outros feitos. As recompensas incluem pontos, desbloqueio de novos níveis e acesso a conteúdo exclusivo como novos personagens jogáveis. Esse sistema mantém o jogador engajado e motivado a continuar jogando.

4.7.6 Interface Intuitiva

A interface do jogo é simples e intuitiva, com menus claros e controles fáceis de usar. Isso garante que as crianças possam navegar pelo jogo e entender suas mecânicas sem frustrações, proporcionando uma experiência agradável e acessível.

4.8 Progressão do Jogo

A progressão no jogo é um aspecto fundamental no design de jogos, pois determina como os jogadores avançam através do conteúdo e como seu desenvolvimento é recompensado ao longo do tempo. A progressão pode assumir várias formas, desde níveis lineares até sistemas complexos de habilidades e desbloqueios, todos projetados para manter o interesse dos jogadores.

O método de progressão utilizado pelo ***Valley of Logic*** acontece por meio de níveis e estágios, no qual cada nível geralmente aumenta em dificuldade e complexidade, desafiando o jogador a dominar novas habilidades ou superar obstáculos mais difíceis. O sucesso em um nível leva à abertura do próximo, criando um caminho claro de avanço.

4.8.1 História do Jogo

Luke, a sábia raposa, era conhecido por toda a sua família e aldeia por conta da sua grande sabedoria e coragem. Não havia um habitante que não o via como um símbolo de bravura e inteligência. No dia do seu aniversário, alguns de seus amigos de infância, animados com a ocasião, o convidaram para uma aventura com muitas explorações e cheia de mistérios, pois agora, Luke tinha idade para adentrar a floresta.

A pequena raposa aceita o convite cheio de disposição. O grupo de amigos se aprofundou entre as árvores, correndo, rindo e compartilhando aventuras anteriores. Contudo, à medida que ia escurecendo, Luke percebeu que tinha algo estranho. Ele tinha se afastado dos amigos, distraído com os encantos da floresta.

Com o coração batendo cada vez mais rápido, Luke percebeu que o único meio para voltar para casa era usar suas habilidades de nascença. Com cautela, a raposa explora o local tentando encontrar pistas e pegadas que o levassem de volta. Depois de horas caminhando, Luke escorrega e cai em um buraco que estava escondido sob folhagens secas.

Ao se recompor e tirar a poeira dos pelos, a raposa se dá conta de que estava em um lugar desconhecido: o Vale da Lógica. Esse vale estava das lendas contadas pelos habitantes da aldeia nas noites de fogueira, ele era conhecido por suas

armadilhas e enigmas que levavam a lugares desconhecidos através de um arco de pedras. Sem ter mais opções, Luke decide encarar os desafios que surgiam a sua frente.

A cada avanço, os enigmas iam ficando mais complexos, exigindo cada vez mais que Luke combinasse sua astucia, sabedora e coragem. Agora ele devia enfrentar as charadas que testariam seus conhecimentos e sua lógica e em cada passo, Luke percebia que se aproximava mais da saída e esse fato o deixava mais forte e confiante.

4.8.2 Eventos Anteriores

O **Vale da Lógica** é composto por três vilas formadas por povos antigos que, em tempos remotos, idealizaram o vale com centro de aprimoramento intelectual. Esse vale mágico e misterioso foi planejado para ser uma atração onde viajantes e moradores poderiam desafiar e expandir seus conhecimentos através de enigmas e desafios. Cada vila com suas peculiaridades, se destacavam pela sua arquitetura e pelos habitantes diferentes dedicados aos estudos e à criação de novos desafios.

Porém, algo horrível aconteceu com o vale. Dados corrompidos, vindos de uma origem desconhecida, começaram a afetar as criaturas inocentes das vilas, os transformando em seres hostis. Essa corrupção devastou a harmonia do **Vale da Lógica**, tornando-o um lugar temido e perigoso.

Para proteger o mundo exterior dessa ameaça, o vale foi exilado. Uma imensa floresta cresceu ao seu redor, isolando-o de todo mundo. Com o tempo o **Vale da Lógica** foi lentamente esquecido, suas glórias e mistérios reduzidos a meras histórias contadas ao redor das fogueiras.

As lendas sobre o vale, no entanto, não desapareceram completamente. Gerações após gerações mantiveram viva a memória do lugar, falando sobre as incríveis aventuras de sabedoria que ali ocorreram. O **Vale da Lógica** permanece em enigma em si, uma prova do poder e do perigo do conhecimento, aguardando o dia em que alguém ousado o suficiente tente desvendar seus antigos mistérios mais uma vez.

4.8.3 Principais Jogadores

No contexto dos jogos eletrônicos, os principais jogadores referem-se aos personagens ou entidades mais importantes que participam da narrativa ou mecânica central do jogo. Estes jogadores são cruciais para o desenvolvimento da história e para a dinâmica do jogo, sendo frequentemente os personagens com os quais os jogadores interagem ou assumem o controle.

O protagonista é o personagem central do jogo, através do qual a maior parte da narrativa é experienciada. No caso deste jogo, o protagonista é Luke, a pequena raposa. Luke é o personagem que o jogador controla diretamente, guiando-o através dos desafios e aventuras no **Vale da Lógica**. Sua coragem e astúcia são fundamentais para resolver os enigmas e progredir na história.

Os antagonistas são os principais opositores do protagonista. Eles criam obstáculos e desafios que o jogador deve superar. No jogo, as criaturas das três vilas, que foram corrompidas por dados corrompidos, servem como antagonistas. Estas criaturas hostis fornecem os desafios que Luke deve enfrentar para avançar no jogo e retornar para casa.

5. RESULTADOS E DISCUSSÕES

Neste tópico serão abordadas as questões a fim de explicar trechos importantes para o desenvolvimento do jogo.

5.1 Programação do Jogo

Após a definição das ferramentas que seriam utilizadas para auxiliar no desenvolvimento do projeto, iniciou-se a confecção do jogo em si. Neste ponto foi preciso criar um objeto para servir como chão e parede e um outro objeto que seria responsável por representar o personagem jogável.

O objeto do chão é referenciado dentro do jogo pelo nome obj_chao (Figura 25) e tem a aparência de um quadrado rosa devido a sua facilidade de localização no processo de construir as fases do jogo, porém, este objeto só é visível para os desenvolvedores do jogo durante a criação dos níveis do jogo, o jogador não consegue visualizar este objeto para evitar que isso comprometa a identidade visual do jogo.

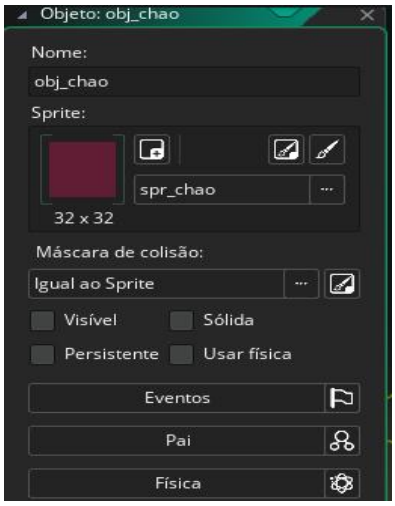


Figura 25: Objeto do chão

Fonte: Dos Autores, 2024

O objeto do jogador é referenciado pelo nome *obj_player* (Figura 26) e diferente do obj_chao, ele é visível para o jogador sendo o personagem utilizado

durante todo o jogo, contendo todos os atributos e variáveis referentes ao personagem jogável.

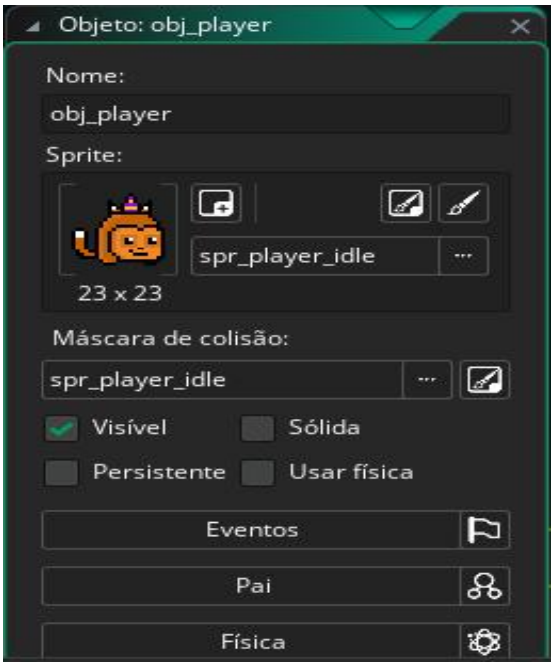


Figura 26: Objeto do Jogador

Fonte: Dos Autores, 2024

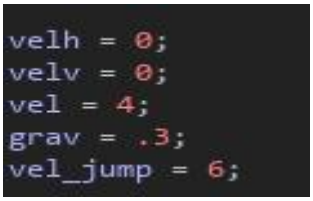
Depois da criação desses objetos, foi preciso pensar em como fazer estes objetos interagirem da maneira correta, ou seja, o objeto que representasse o player pudesse se mover da maneira correta pelo cenário e colidir corretamente com o objeto que representa o chão e as paredes. O primeiro passo para fazer com que o jogo funcionasse da maneira planejada, foi a definição dos controles que o jogador poderia utilizar para locomover o personagem pelo cenário e a criação de variáveis que seriam atribuídas ao player para controlar as suas ações.

Após definir os comandos e toda a movimentação do jogador, o *obj_player* recebe essas variáveis inicialmente, cada uma com sua determinada finalidade, as variáveis denominadas velh e velv servem para atribuir uma velocidade horizontal e vertical ao personagem, ou seja, quando o personagem inicia o jogo, a sua velocidade horizontal e vertical iguala-se ao zero, pois o personagem precisa iniciar o jogo parado.

A variável vel serve para definir a velocidade que o personagem vai se mover, quanto maior for o valor atribuído a ela mais rápido será o personagem, foi

determinado um valor baixo para esta variável devido a velocidade com que o jogo prossegue, um jogo muito rápido pode não ser muito atrativo para crianças. A variável grav serve para definir a gravidade aplicada sobre o personagem ao longo do jogo, ou seja, quanto maior for o valor desta variável, mais o personagem será atraído para baixo enquanto estiver no ar, esta variável precisa estar alinhada com a variável *vel_jump* por meio de testes até achar os valores que se adaptam melhor ao estilo de jogo proposto. Por último, a variável *vel_jump* serve para definir a altura do pulo do personagem principal, o valor atribuído para ela é uma medida equivalente a aproximadamente 2 pixels e meio, porém, pode ser alterada de acordo com a preferência do desenvolvedor (Figura 27).

Figura 27: Variáveis Iniciais do Player



Fonte: Dos Autores, 2024

Com todos os aspectos referentes a movimentação do personagem resolvidas, era necessário desenvolver um método que fizesse com que os objetos colidissem um no outro. A solução para este problema foi criar uma variável com o nome de _col e instanciar os atributos x + velh, y e obj_chao, no qual x representa a posição que o personagem está posicionado na horizontal e o y seria a sua posição na vertical e por último o objeto que com que aconteceria a colisão que neste caso é o obj_chao. Após a definição da variável, ela foi inserida dentro de um condicional *If* que serve para identificar a colisão o obj_chao pelos lados e assim fazer com que o personagem não pudesse atravessar a parede, este conceito é chamado de colisão *pixel perfect* e é usado com muita frequência em jogos de plataforma (Figura 28).

Figura 28: Trecho do Código Responsável pela Colisão.

```
var _col = instance_place(x + velh, y, obj_chao);
//if colisão
if( _col)
{
    //direita
    if (velh > 0)
    {
        //grudando na parte esquerda da parede
        x = _col.bbox_left + (x - bbox_right);
    }
    //esquerda
    if (velh < 0)
    {
        x = _col.bbox_right + (x - bbox_left);
    }
    //colisão
    velh = 0;
    }
```

Fonte: Dos Autores, 2024

Neste ponto o player já pode se movimentar pelo cenário e a sua colisão com o chão e a parede já estava configurada, após isto foi configurada lógica que permite o jogador a pular pelo cenário e aplicar a gravidade nele caso estivesse no ar. O desenvolvimento desta funcionalidade consistiu na criação de uma nova variável com o nome de pulos_extra com o valor de 1, tinha-se como idealização, a implementação de uma funcionalidade aonde o jogador pudesse executar um pulo duplo, ou seja, quando estivesse no ar ele pudesse executar um pulo extra, este conceito é utilizado de maneira frequente em diversos jogos, após definir a variável que possibilita este pulo extra, foi desenvolvida a lógica que permitia o personagem pular e eventualmente realizar o pulo duplo (Figura 29).

Figura 29: Lógica do Pulo

```
    // Pulo
if (_jump) {
    velv = -vel_jump;
    pulos_extra = 1;
}
```

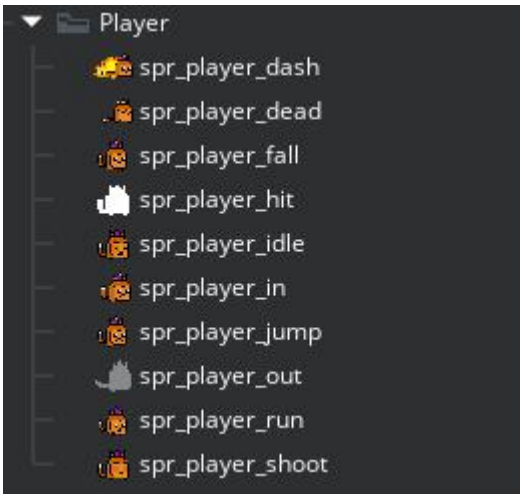
Fonte: Dos Autores, 2024

A lógica para o pulo consiste em algo bem simples, um condicional que recebe a variável *_jump* que tem como função identificar se o personagem executou o pulo, se esta condição for verdadeira, o velv que representa a velocidade vertical se torna o valor negativo de *vel_jump* que representa a distância do pulo do personagem, o sinal de negativo indica que o personagem está indo para cima, pois o eixo y é tratado desta maneira, onde valores negativos estão para cima e os valores positivos estão consequentemente para baixo. Depois de atribuir o valor a

velv fazendo o personagem pular, a variável de pulos_extras tem o seu valor em 1, possibilitando que o personagem execute o segundo pulo.

Com as questões de movimentações básicas do personagem completa, foi implementada as sprites do personagem principal com suas variações de movimentos e funcionalidades (Figura 30).

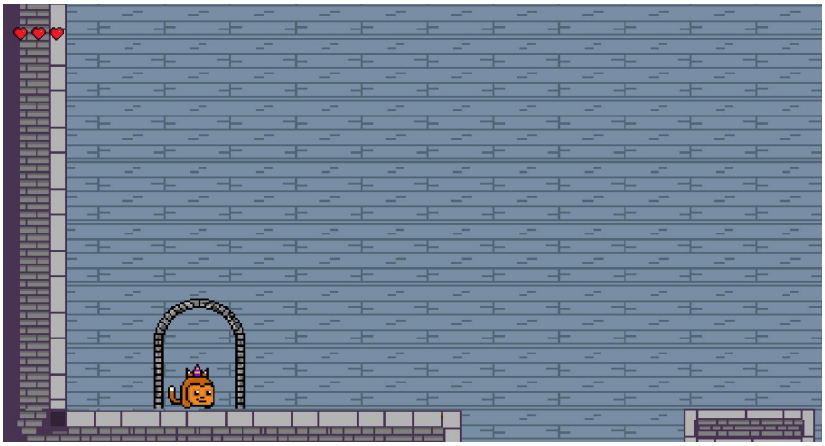
Figura 30: Sprite do Personagem Principal



Fonte: Dos Autores, 2024

Foi necessário também uma definição de qual resolução de tela seria ideal para o jogo, após vários testes e análises entre diferentes escalas de resolução possíveis, foi escolhida a resolução 400x240, devido a sua capacidade de promover uma visualização mais limpa e clara dos pixels do jogo, dessa forma o jogador consegue ter mais contato com o design e a identidade visual da aplicação também oferecendo uma visualização com certa profundidade do cenário, obtendo a capacidade de enxergar os inimigos e o cenário a uma certa distância, mas sem ter uma visualização muito distante dos pixels, ajudando o jogador a ter uma boa experiencia quando estiver jogando (Figura 31).

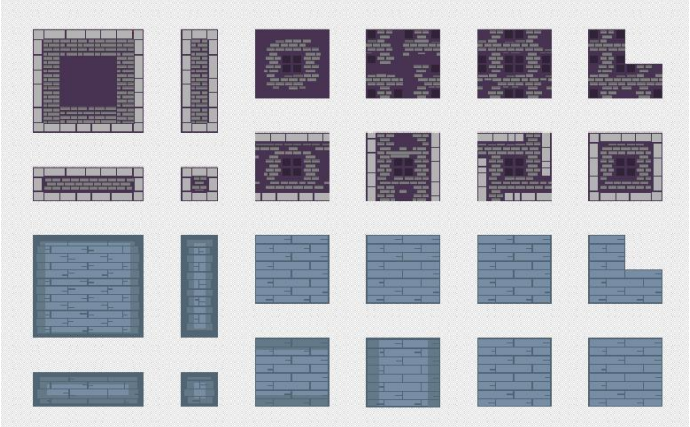
Figura 31: Tela do Jogo



Fonte: Dos Autores, 2024

Após ajustar os detalhes referentes a visualização do jogo, foi realizada a idealização dos cenários. Os *sprites* dos cenários são chamados de *Tilesets*, e elas são compostas pelos *sprites* que juntas formam o chão e o fundo do ambiente (Figura 32).

Figura 32: Tileset de um Cenário do Jogo

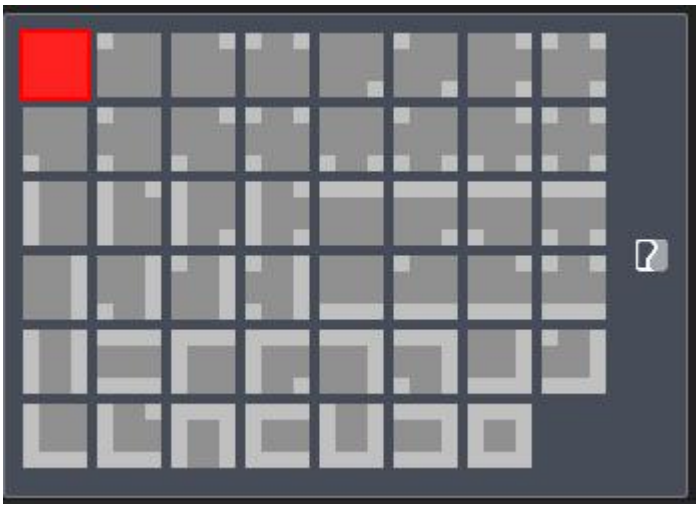


Fonte: Dos Autores, 2024

Os *Tilesets* geralmente são compostos por 24 sprites diferentes divididos entre 12 *sprites* que serão a parte do cenário que irá interagir com o jogador, ou seja, o chão as paredes e o teto, e 12 *sprites* que irão compor o fundo do cenário. Para implementar este *Tileset* dentro da *engine* e realizar os procedimentos de *level design,* o *GameMaker of*erece uma ferramenta chamada *Auto Tile*, esta ferramenta visa auxiliar no processo de montagem das fases. A ferramenta consiste no preenchimento de posições pré-definidas com pixels de um determinado *Tileset*, após o preenchimento destas posições o *GameMaker* compilar estas posições e

consegue identificar os pixels do cenário e realizar um preenchimento automático conforme a fase vai sendo confeccionada. Esta ferramenta auxilia no processo, pois remove a necessidade de preencher pixel por pixel da fase, fazendo com que as fases sejam desenvolvidas de uma maneira mais rápida e prática também trazendo um certo padrão para as fases do jogo (Figura 33).

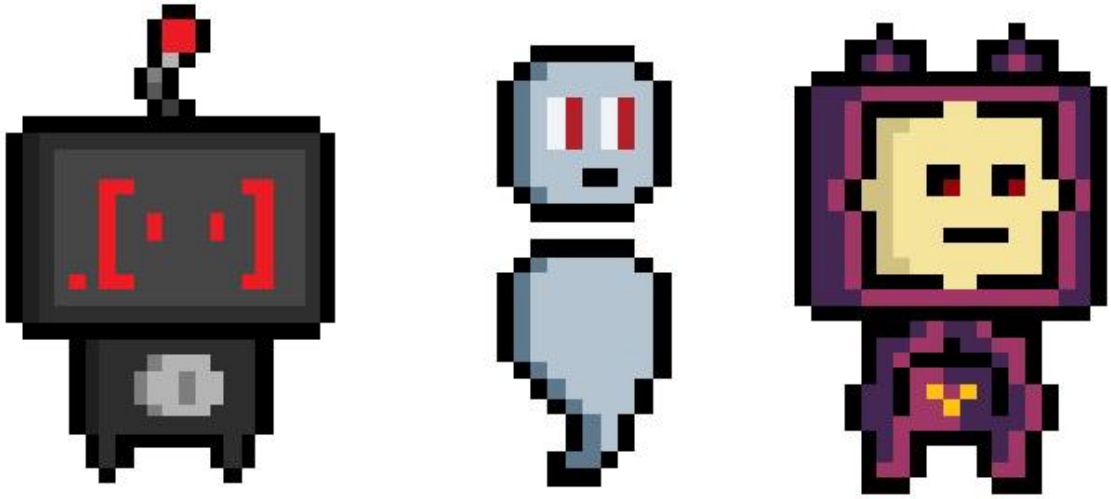
Figura 33: Tela do Preenchimento do Auto Tile



Fonte: Dos Autores, 2024

Com os cenários definidos, começou a idealização de vilões do jogo, foi planejado a utilização de elementos da programação para servir de inspiração para a confecção dos inimigos presentes na aplicação, a partir desta ideia, foram criados 3 tipos de vilões básicos, que compõem diferentes fases do jogo. Foram criados o inimigo inspirado em colchetes (caractere utilizado para o uso de *arrays* em diversas linguagens de programação), o inimigo inspirado em ponto e vírgula (caractere utilizado para separar as instruções de um código uma da outra) e por fim, o inimigo inspirado em uma chave (caractere utilizado para criar blocos de código) (Figura 34).

Figura 34: Inimigos do Jogo



Fonte: Dos Autores, 2024

Assim como o personagem principal, os inimigos precisam ter variações de sua sprite padrão para adaptar cada sprite a uma ação específica.

Todos os inimigos possuem um objeto próprio, porém todos herdam as funcionalidades de um objeto pai chamado obj_inimigo_pai, isso acontece porque todos os inimigos possuem a mesma lógica para receber dano e morrer, por isso utiliza-se a herança dos atributos do objeto pai, para facilitar a implementação de novos inimigos. Como o personagem principal, os inimigos também possuem as variáveis básicas para a sua movimentação como velh, velv e outras variáveis.

A lógica para o dano e a morte dos inimigos consiste na criação de duas variáveis, uma com o nome de dano e outra com o nome de morto. A variável dano servirá para verificar se o inimigo está levando dano do jogador ou não, já a variável morta garante que quando o inimigo receber dano do jogador, ele efetivamente morra.

Para checar a variável dano, ela é inserida dentro de um condicional *If* (Figura 35) e verifica se as variáveis dano e morto são falsas, garantindo que o personagem poderá levar dano, somente enquanto estiver vivo, caso as condições sejam atendidas, o objeto do inimigo irá mudar para sua sprite de dano e o seu velh será igual a zero ou seja, ele não poderá se mexer enquanto estiver recebendo o dano. Um inimigo pode receber dano de 3 formas diferentes, a primeira é quando o jogador pula em sua cabeça, a segunda é quando o jogador atira nele e a terceira é quando o jogador utiliza um *dash* nele (Figura 35).

Figura 35: Lógica de Dano I

```
//checando o dano
if (dano && morto == false)
{
    sprite_index = spr_inimigo_vetor_dano;
    velh = 0;
}
```

Fonte: Dos Autores, 2024

Agora que o inimigo está recebendo dano corretamente, é feita a implementação da lógica de morte do inimigo, dentro de um outro condicional *If,* é feita a verificação no objeto do inimigo, se o objeto estiver com a sprite de dano significa que a variável dano é verdadeira, sendo assim, ao término da animação de dano, o personagem deve morrer, dentro do condicional a variável morto é definida como verdadeira (Figura 36).

Figura 36: Lógica para a Morte do Personagem

```
//se estou usando sprite de dano e a animação acabou eu morro
if (sprite_index == spr_inimigo_vetor_dano)
{
    //morrendo
    morto = true;
}
```

Fonte: Dos Autores, 2024

Com a variável morto sendo definida como verdadeira, o código cai em outro condicional do objeto do inimigo que tem como condição, a variável morto ser verdadeira, após validar esta condição o inimigo assume a sua sprite de morto e após o término da animação, a sua sprite vai ficando invisível por alguns segundo até sumir totalmente da tela do jogador, quando a sprite está totalmente transparente, o objeto do inimigo é destruído para evitar que ele entre em conflito com outros objetos dentro da fase, garantindo que não existam bugs envolvendo o objeto do inimigo (Figura 37).

Figura 37: Lógica de Animação de Morte

```
if (morto)
{
    sprite_index = spr_inimigo_vetor_dead;
    //sumindo se acabou a animação
    if (image_speed <= 0)
    {
        image_alpha -= 0.01;
    }
    //se ele sumir ele se destroi
    if (image_alpha <= 0)
    {
        instance_destroy();
    }
    exit;
}
```

Fonte: Dos Autores, 2024

Para os aspectos da movimentação dos inimigos do jogo, foi desenvolvida uma lógica similar a lógica do player, porém, com a adição de uma variável chamada tempo_decidir_andar que serve para determinar o tempo necessário para o inimigo tomar alguma decisão no jogo, ela é inicializada com um valor equivalente a 3 segundos, é também utilizada uma variável _chao, que verifica que o inimigo deve estar em contanto com o obj_chao (Figura 38).

Figura 38: Lógica para Checar se o Inimigo está no Chão

```
//chcando se o inimigo esta no chão
var _chao = place_meeting(x, y + 1, obj_chao);
```

Fonte: Dos Autores, 2024

Após realizar esta verificação, a variável é adicionada como uma condição dentro de um *If*, aonde é determinado que a variável tempo_decidir_andar é diminuída caso o inimigo esteja no chão, quando este tempo acaba, outro condicional é adicionado para determinar que quando a variável tempo_decidir_andar for menor ou igual a zero, o inimigo irá decidir se continuará parado ou se irá andar, caso ele decida andar, ele cai em outra condição aonde o inimigo irá escolher para qual lado andar, caso ele decida ficar, o seu valor de velh será igual a zero, ou seja, ele permanecerá parado, e quando todas estas condições forem atendidas, o tempo de decisão do inimigo será resetado para que ele possa tomar a próxima escolha e se mover de maneira livre pelo cenário (Figura 39).

Figura 39: Lógica para Movimentação do Inimigo

```
if(_chao)
{
    //se estou no chão diminuir o tempo de decisão
    tempo_decidir_andar -= 1;
    //se o tempo acabar decidir para onde ir
    if (tempo_decidir_andar <= 0){
        andando = choose(true, false);
        //Escolhendo direção se ele decidiu andar
        if(andando){
            velh = choose(vel, -vel);
        }
        else{
            velh = 0;
        }
        //resetando tempo
        tempo_decidir_andar = room_speed * 2;
    }
```

Fonte: Dos Autores, 2024

De acordo com (ALVES, REIS, TENORIO, 2018) ”surgiu uma nova técnica da Inteligência Artificial, a qual é utilizada até hoje, que são as Máquinas de Estado Finito. Elas descrevem os comportamentos dos NPCs.”. Este código é considerado um algoritmo de IA que serve para deixar a movimentação dos inimigos mais imprevisível ao jogador, elevando assim o grau de dificuldade do jogo.

Além da lógica de movimentação, foi implementada uma funcionalidade que permite que os inimigos não caiam em armadilhas feitas para serem obstáculos ao jogador. Para que este comportamento do inimigo aconteça, foi criada uma variável chamada pode_cair, se o valor false for atribuído a esta variável durante a instância do objeto do inimigo na fase do jogo, significa que o inimigo irá fazer uma checagem a uma distância previamente definida, e se ele não identificar o obj_chao dentro dessa distância, ele altera sua direção, fazendo assim ser impossível o inimigo cair em buracos (Figura 40).

Figura 40: Lógica para Evitar Quedas

```
//se não pode cair, mudar direção
if(pode_cair == false)
{
if(place_meeting(x + (velh * 10), y + 1, obj_chao)==false)
{
//se eu não posso cair e não estou tocando no chão na frente, então mudo direção
    velh *= -1;
}
}
```

Fonte: Dos Autores, 2024

Uma lógica um pouco mais simples do inimigo seria a lógica da sua colisão com a parede, se o inimigo estiver andando e colidir com uma parede, o valor do seu velh é multiplicado por -1, fazendo assim o inimigo mudar sua direção, impedindo que ele fique preso andando contra as paredes do cenário (Figura 41).

Figura 41: Lógica para Mudar a Direção

```
//se bater na parede ele muda de direção
if (place_meeting(x + velh, y, obj_chao))
{
    velh *= -1;
}
```

Fonte: Dos Autores, 2024

Definindo todos os aspectos referentes ao comportamento dos inimigos, foi preciso desenvolver uma lógica para conseguir causar dano ao personagem principal quando ele colidisse com um inimigo. Para desenvolver esta funcionalidade, foi planejada a criação de duas variáveis globais (Figura 42) chamadas max_vida e vida dentro de um *Script.* Uma variável global é uma variável que pode ser acessada por qualquer objeto em qualquer parte do jogo, enquanto um *script* é composto por todos os códigos que serão executados antes que todos os outros objetos quando o jogo abrir. Durante a criação destas variáveis, foi atribuído a variável max_vida um valor de 3, pois o personagem principal poderá levar dano 3 vezes antes de morrer, enquanto a variável vida serve para representar a quantidade de vidas atuais do jogador, por isso o valor que é atribuído a ela durante sua inicialização é a variável max_vida.

Figura 42: Variáveis Globais

```
global.max_vida = 3;
global.vida = global.max_vida;
```

Fonte: Dos Autores, 2024

Ao definir estas variáveis, foi criada também, uma variável com o nome posso_perder_vida e inicializada como verdadeira, essa variável serve para

determinar se o jogador pode receber dano. Após a criação dessas variáveis a lógica da morte do personagem principal é definida, quando o jogador recebe dano, a sua sprite muda para representar o dano sofrido, e a variável global vida é decrementada em 1, ou seja, a cada dano que o jogador receber, ele perderá uma vida, quando as vidas do jogador acabarem, ele entra no estado chamado *dead*, se o personagem não perdeu todas as vidas, a variável posso_perder_via recebe o valor falso, isso ocorre para evitar que o jogador possa receber danos consecutivos de um único inimigo (Figura 43).

Figura 43: Lógica de Dano II

```
// Lógica de dano
if (dano) {
    sprite_index = spr_player_hit;
    velh = 0;
    velv = 0;
    if (posso_perder_vida) {
        global.vida--;
        if (global.vida <= 0) {
            estado = "dead";
        } else {
            posso_perder_vida = false;
        }
    }
}
```

Fonte: Dos Autores, 2024

O estado *dead* mencionado no código anterior se trata de um *switch* case (Figura 44) que serve para lidar com diferentes estados do personagem principal. No estado *dead*, ele muda a sprite do personagem para mostrar a animação de morte e utiliza uma variável chamada *timer*_reinicia para reiniciar o jogo após a morte do personagem principal, ela é inicializada com um valor equivalente a 2 segundos, quando o personagem entra no estado *dead*, ela é decrementada, e quando ela for menor ou igual a zero, as variáveis de vida voltam a ter o valor de 3 e a fase é reiniciada, levando o jogador até o início da fase com as suas vidas restauradas.

Figura 44: Estado Dead

```
case "dead":
    mudando_sprite(spr_player_dead);
    timer_reinicia--;
    if (timer_reinicia <= 0) {
        global.vida = global.max_vida;
        room_restart();
    }
```

Fonte: Dos Autores, 2024

Uma última verificação necessária para o dano causado no personagem seria quando o personagem morre enquanto está no ar, pensando nesta possibilidade, foi feita uma lógica que verifica se o personagem está fora do chão quando entra no estado *dead*, e se esta condição for verdadeira, é aplicada a gravidade no personagem e ele volta para o chão antes de sua morte (Figura 45).

Figura 45: Verificação Sobre onde está o Personagem

```
// Lógica de estado morto
if (estado == "dead") {
    if (!_chao) {
        velv = 3;
        velv += grav;
    }
}
```

Fonte: Dos Autores, 2024

Com todas as funcionalidades básicas do jogador e dos inimigos que irão compor o jogo completas, idealizamos uma forma do jogador conseguir avançar de uma fase para a outra, para solucionar este problema, criamos um objeto chamado obj_portal, que são os portais que existem no começo e no fim de cada fase, estes portais servem para levar o jogador a fase seguinte e conseguir progredir dentro da aplicação. A lógica desses portais baseia-se em uma transição de tela representada pela variável global chamada de transição, e dentro de um script, existe uma função para executar a abertura desses portais, quando o portal abre a variável global com o nome de transição é definida como verdadeira e o personagem é levado para a próxima fase do jogo a partir do código *room_goto_next* que serve para mudar o jogo para a fase que se encontra a frente da fase atual (Figura 46). Os portais somente podem ser abertos pelo jogador.

Figura 46: Lógica dos Portais

```
function sq_transicao_in_Moment()
{
    global.transicao = true;
    //se existe uma proxima room eu vou para ela
    if (room_next(room) != -1)
    {
    room_goto_next();
    }
```

Fonte: Dos Autores, 2024

Caso o portal esteja fechando, ou seja, o jogador esteja saindo do portal na próxima fase, a variável transição é definida como falsa e o portal é desativado.

Com o Intuito de elevar o grau de dificuldade e complexidade do jogo, foram criadas plataformas que se movem, exigindo mais atenção e coordenação motora do jogador. A lógica para manipular essas plataformas consiste em criar dois objetos, um para a plataforma que irá se mover para a vertical e outro para a plataforma na horizontal, após a criação desses objetos, eles serão definidos como filhos do obj_chao para herdar as mesmas características de colisão deste objeto, e por último, o comportamento dessas plataformas vão ser pré-definidos. Quando esta plataforma entrar em contanto com o obj_chao, ela mudará imediatamente a sua direção, obrigando o jogador a prestar atenção em seu movimento (Figura 47).

Figura 47: Lógica da Plataforma Móvel na Horizontal

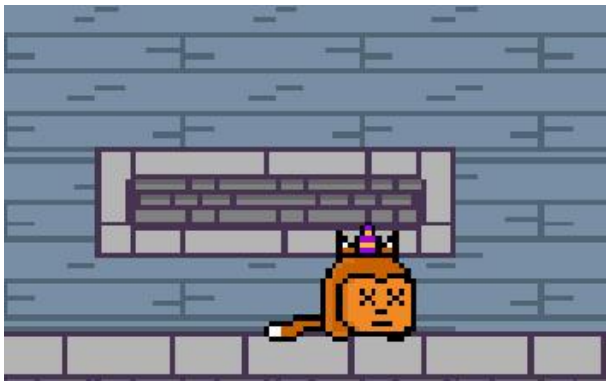
```
hspd = move_dir * move_spd;

if(place_meeting(x+sign(hspd),y,obj_chao))
{
    move_dir*=-1;}
var c = place_meeting(x+sign(hspd),y,obj_player) or place_meeting(x,y-1,obj_player);
if(c)
{
    with(obj_player){
        if(!place_meeting(x+other.hspd,y,obj_chao)){
            x+=other.hspd;
        }
    }
}
x+=hspd;
```

Fonte: Dos Autores, 2024

Para desenvolver a plataforma na vertical a lógica é semelhante a horizontal, porém ao invés de usar a variável hspd que serve para definir a velocidade horizontal da plataforma, usa-se a variável vspd, que tem a mesma função, porém na velocidade vertical.

Analisando o comportamento dessas plataformas, poderia existir a chance do jogador ser prensado por elas causando *bugs* na movimentação do jogador, graças a isto, se fez necessária o desenvolvimento de uma lógica que lidasse com esta colisão do *player*. Para resolver este *bug*, a solução foi criar um evento de colisão dentro do obj_*player* e dentro deste evento, foi adicionado que o personagem entraria no estado *dead* e que a variável vida seria igual a zero, ou seja, ao ser prensado por uma plataforma móvel, o jogador recebe dano e morre (Figura 48).

Figura 48: Personagem Morrendo



Fonte: Dos Autores, 2024

Foi desenvolvida também, alguns golpes para o personagem, visando o engajamento do jogador, foi idealizado a criação de golpes especiais para o personagem principal, estes golpes são chamados de *shoot* e *dash*. O *dash* basicamente se trata de um impulso que o personagem pode utilizar tanto para se movimentar mais rápido como para causar dano a inimigos, enquanto o *shoot* é uma pequena bola de fogo que causa danos instantâneo aos inimigos.

Para implementar estas ideias, foi necessário realizar a criação de sprites específicas para estas ações, logo após foi preciso implementar a lógica necessária

para que estas funcionalidades interagissem corretamente com os outros objetos do jogo.

A lógica do *dash* consiste em iniciar uma variável para o *dash* com o valor de falso e uma variável para o tempo que o *dash* vai durar e fazer a variável se tornar verdadeira quando o jogador ativar o *dash*, quando a variável for verdadeira, o jogador cairá em um condicional que irá definir uma velocidade para o *dash* que irá decrementar a variável que controla o tempo da ação que foi inicializada com 2 segundos, quando este temporizador chegar a um valor menor ou igual a zero, a ação de *dash* ira acabar e o *player* sairá deste estado (Figura 49).

Figura 49: Lógica do Dash

```
// Lógica do dash
if (dash) {
    velh = image_xscale * dash_speed; // Define a velocidade horizontal do dash
    sprite_index = spr_player_dash; // Muda para a sprite de dash durante o dash
    dash_timer--;
    if (dash_timer <= 0) {
        dash = false;
    }
```

Fonte: Dos Autores, 2024

Após esses passos, ainda era necessário desenvolver a lógica de colisão do personagem com inimigos durante o *dash*. Se o personagem está em *dash*, ele verifica colisões com inimigos e marca-os para sofrer dano, iniciando a animação de morte do inimigo. Se o personagem não está em *dash*, ele verifica colisões e, se o temporizador de invulnerabilidade é zero, marca o personagem para sofrer dano e inicia os temporizadores de dano e invulnerabilidade. Além disso, o código gerencia a recarga do *dash*, reduzindo o temporizador a cada atualização (Figura 50).

Figura 50: Lógica do Dano do Dash

```
// Lógica de colisão com inimigo durante o dash
if (dash) {
    var _inimigo_dash = instance_place(x, y, obj_inimigo_pai);
    if (_inimigo_dash) {
        _inimigo_dash.dano = true; // Marca o inimigo para iniciar a animação de morte
        }
} else {
    // Tomando dano
    var _inimigo = instance_place(x, y, obj_inimigo_pai);
    if (_inimigo && inv_timer <= 0) {
        if (!_inimigo.morto && !_inimigo.dano) {
            dano = true;
            timer_dano = tempo_dano;
            inv_timer = inv_tempo;}}}
// Reduzir o temporizador de recarga do dash
if (dash_cooldown_timer > 0) {
    dash_cooldown_timer--;}};
```

Fonte: Dos Autores, 2024

Foi estipulado este tempo de recarga para o *dash*, para assegurar os limites desta função evitando *bugs* e manter um certo grau de dificuldade para o jogador, impossibilitando-o de realizar a mesma ação diversas vezes de maneira consecutiva.

Para a funcionalidade do *shoot*, foi criada uma lógica que detecta se o *shoot* foi ativado pelo jogador e se o *cooldown* (tempo de recarga) de tiro é zero. Se sim, ele calcula a posição do tiro com base na posição do personagem, ajustando conforme necessário se o personagem estiver virado para a esquerda. Em seguida, cria o tiro e define sua velocidade e direção, ativando uma animação de disparo no personagem. O *cooldown* é reiniciado após o tiro. Enquanto o personagem está atirando, a sprite é alterada para a de disparo e um temporizador é decrementado. Quando o temporizador chega a zero, a sprite volta ao normal. O *cooldown* de tiro também é decrementado a cada atualização se for maior que zero.

Foi criada uma funcionalidade para salvar e carregar o jogo de determinados pontos, ao ativar a opção de salvar dentro do jogo, um arquivo da extensão sav, é criado dentro do diretório do jogo, dentro desde arquivo possui a fase que o jogador estava no momento que salvou o jogo, quando o jogador voltar a abrir o jogo e clicar na opção de carregar o jogo no menu, ele poderá voltar na mesma fase em que salvou (Figura 51).

Figura 51: Função para Salvar o Jogo

```
function scr_save_game() {
    var save_file = file_text_open_write("savegame.sav");

    // Salvar a sala atual do jogador
    file_text_write_string(save_file, room_get_name(room));
    file_text_writeln(save_file);

    // Fechar o arquivo
    file_text_close(save_file);
}
```

Fonte: Dos Autores, 2024

Após salvar o jogo, quando o jogo for aberto outra vez, o jogador poderá carregar o jogo salvo e voltar a jogar exatamente da fase em que ele parou, tornando a experiencia do jogo mais leve e menos exaustiva, já que o jogador poderá continuar seu progresso a qualquer momento (Figura 52).

Figura 52: Função para Carregar o Jogo

```
function scr_load_game() {
    if (file_exists("savegame.sav")) {
        var load_file = file_text_open_read("savegame.sav");

        // Carregar o nome da sala do jogador
        var room_name = file_text_read_string(load_file);
        file_text_readln(load_file);


        // Fechar o arquivo
        file_text_close(load_file);
```

Fonte: Dos Autores, 2024

E por fim, foi desenvolvida a criação de um menu para o jogo. O menu será a primeira tela a aparecer após a inicialização do jogo, através do menu seria possível

começar um novo jogo, carregar um jogo existente, alterar as opções do jogo e fechar o jogo (Figura 53).

Figura 53: Menu do Jogo

Fonte: Dos Autores, 2024

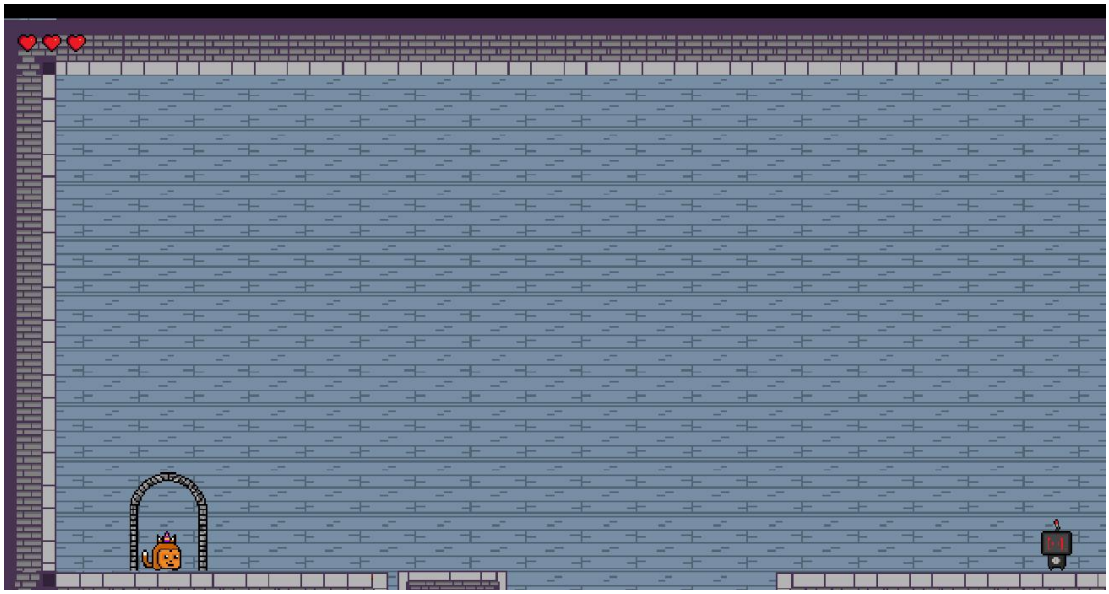
Se a criança optar por escolher entrar em opções, irá aparecer uma tela com todas as funcionalidades que ela pode controlar, sendo elas: volume geral, volume da música, brilho e voltar (Figura 54).

Figura 54: Menu de Opções

Fonte: Dos Autores, 2024

Ao escolher a opção de novo jogo, a criança irá começar o jogo diretamente da primeira fase (Figura 55).

Figura 55: Primeira fase do jogo



Fonte: Dos autores, 2024

Se o jogador optar por selecionar a opção de continuar o jogo, o jogo será carregado a partir da última fase registrada pelo jogo (Figura 56).

Figura 56: Quinta fase do jogo

Fonte: Dos autores, 2024

Um dos exemplos de desafios presentes no jogo, será as fases dos chefes, contendo um desafio único e um jeito diferente de vencer cada chefe, estimulando a lógica e o raciocínio rápido da criança (Figura 57).

Figura 57: Primeiro Chefe

Fonte: Dos autores, 2024

A imagem acima mostra um dos chefes presentes no jogo, o método para vencê-lo consiste na utilização de perguntas que o jogador deve responder para prosseguir dentro do jogo.

Ao entrar na fase do chefe, ele fará uma pergunta lógica com uma resposta simples, e cada botão na esquerda do jogador, irá representar uma alternativa para, enquanto o botão da direita será o botão para confirmar a resposta. O jogador deverá responder à pergunta e confirmar a sua resposta, caso a resposta esteja errada, o jogador perderá uma de suas 3 vidas, caso esteja correta, o chefe perderá uma de suas 3 vidas. Enquanto a criança pensa na resposta o chefe irá lançar ataques no jogador para aumentar o grau de dificuldade e trabalhar o uso do pensamento lógico de maneira mais rápida.

Se todas as vidas do jogador acabarem, a fase é reiniciada, dando uma nova chance a criança, se o chefe perder todas as vidas, uma porta para a próxima fase será aberta e o jogador poderá prosseguir para as fases mais avançadas do jogo.

6. CONSIDERAÇÕES FINAIS

Utilizando o método de pesquisa bibliográfica, foram encontrados alguns artigos que auxiliaram neste trabalho. Esses artigos ajudam a enxergar a viabilidade da implantação de jogos como ferramentas de ensino dentro do contexto acadêmico.

O artigo intitulado **Jogos digitais no processo de ensino e aprendizado do aluno na alfabetização,** escrito por Olímpia e Abreu (2022), tinha como objetivo discutir a implantação de jogos no ambiente escolar atual como ferramentas de ensino. Após uma pesquisa realizada pelas autoras, foi possível identificar que os jogos foram implantados em algumas escolas devido ao contexto de aulas online durante a pandemia. Isso mostra que é possível utilizar jogos como ferramentas de ensino. O artigo conclui que a utilização de jogos no processo de aprendizado é necessária, pois os jogos são uma motivação para manter o aluno engajado, fazendo com que ele aprenda de uma maneira prática.

Ao analisar os resultados desse artigo, é possível concluir que os jogos podem ter um impacto positivo no processo de aprendizagem e podem contribuir para o desenvolvimento de habilidades necessárias para o ser humano. Se forem utilizados de maneira correta, os jogos podem se tornar uma ferramenta que motiva constantemente a criança a progredir no seu desenvolvimento cognitivo de uma maneira simples e leve.

À vista disso, o objetivo deste trabalho foi desenvolver um jogo de plataforma 2D, ***Valley of Logic***, destinado a ajudar crianças de 4 a 9 anos a desenvolver o raciocínio lógico. Ao longo do desenvolvimento, foi essencial considerar tanto a diversão quanto o valor educacional, garantindo que o jogo proporcionasse uma experiência de aprendizado envolvente.

Após pesquisas sobre o impacto dos jogos digitais na educação, identificou-se que os jogos podem ser ferramentas eficazes para o desenvolvimento cognitivo, especialmente quando projetados com objetivos pedagógicos. Foram implementadas mecânicas de jogo que desafiam os jogadores a resolverem *puzzles* e enfrentarem desafios lógicos, incentivando o pensamento crítico e a resolução de problemas.

O primeiro objetivo específico do trabalho foi estudar os conceitos de IA para serem utilizados no projeto. Esse objetivo foi alcançado com sucesso, foram utilizados algoritmos de IA para controlar a movimentação dos inimigos, criando

comportamentos que evitam obstáculos e aumentam a dificuldade de forma equilibrada.

O segundo objetivo específico era criar um ambiente de ensino compatível com a idade do público-alvo. Esse objetivo foi alcançado com êxito, a identidade visual do jogo foi completamente planejada para promover a identificação do público-alvo, com a presença de sprites agradáveis e chamativos para crianças.

Para o terceiro objetivo específico, foram desenvolvidos desafios e atividades práticas para estimular o pensamento lógico e a resolução de problemas, fundamentais para o aprendizado de programação. Esses desafios foram elaborados com o intuito de engajar as crianças e mantê-las motivadas, podemos concluir que este objetivo foi alcançado como o esperado.

Em conclusão, o ***Valley of logic*** atingiu seus objetivos de proporcionar uma experiência educacional divertida e eficaz. O jogo não só mantém as crianças motivadas a continuar aprendendo, mas também desenvolve importantes habilidades cognitivas de uma maneira lúdica e envolvente.

REFERÊNCIAS

A evolução da inteligência artificial desde os anos 1960. **Awari.** 2023. Disponível em: https://awari.com.br/a-evolucao-da-inteligencia-artificial-desde-os-anos-1960/?utm_source=blog&utm_campaign=projeto+blog&utm_medium=A%20evolução%20da%20inteligência%20artificial%20desde%20os%20anos%201960. Acesso em: 04 de julho de 2024.

A história da inteligência artificial. **Instituto de engenharia.** 2018. Disponivel em: https://www.institutodeengenharia.org.br/site/2018/10/29/a-historia-da-inteligencia-artificial/. Acesso em: 25 de maio de 2024.

A indústria de jogos eletrônicos, um setor em ascensão no Brasil. **E-commerce Brasil.** 2024. Disponível em: https://www.ecommercebrasil.com.br/noticias/industria-de-jogos-eletronicos-um-setor-em-ascensao-no-brasil. Acesso em: 25 de maio de 2024.

ABDALA, Vitor. De cada 100 brasileiros, 87 usavam internet em 2022, aponta IBGE. **Agência Brasil.** 2023. Disponível em: https://agenciabrasil.ebc.com.br/geral/noticia/2023-11/de-cada-100-brasileiros-87-usavam-internet-em-2022-aponta-ibge. Acesso em: 25 de maio de 2024.

ANDRÉ, Dagoberto; JOSÉ, Fábio. **A utilização de jogos educacionais digitais como proposta de metodologia ativa de ensino para uma aprendizagem significativa na educação básica.** Universidade Federal de Santa Maria (UFSM), 2020.

AUGUSTO, Heitor. Charles Babbage e a máquina analítica: como o matemático construiu o precursor do computador moderno?. **Velip.** Disponível em: https://velip.com.br/charles-babbage-e-a-maquina-analitica-como-o-matematico-construiu-o-precursor-do-computador-moderno/. Acesso em: 04 de julho 2024.

BENEVIDES, Murillo. Nintendo Entertainment System (NES) ganha novo sistema operacional após 39 anos. **Olhar Digital.** 2022. Disponível em: https://olhardigital.com.br/2022/10/06/reviews/nes-ganha-um-sistema-operacional-apos-39-anos/. Acesso em: 25 de maio de 2024.

BMO | Wiki Hora de Aventura. **Fandom.** 2024. Disponível em: https://horadeaventura.fandom.com/pt-br/wiki/BMO. Acesso em 05 de julho de 2024.

Boden, M. **Inteligência Artificial: Uma Nova Síntese.** São Paulo: Fundação Editora UNESP, 2020.

BRASSCOM. **Estudo da Brasscom aponta demanda de 797 mil profissionais de tecnologia até 2025.** 2021. Disponível em: https://brasscom.org.br/estudo-da-brasscom-aponta-demanda-de-797-mil-profissionais-de-tecnologia-ate-2025/. Acesso em: 16 de março de 2023.

Brougère, G. **Jogo e educação.** São Paulo: Artes Médicas, 2004.

BURKE, Myles. Como foi criado o 1º videogame da história, há 50 anos. **BBC News Brasil**. 2023. Disponível em: https://www.bbc.com/portuguese/articles/c3gyw77e6eko. Acesso em: 25 de maio de 2024.

BUSHNELL, Nolan. **Novas Aventuras**. Revista Oi, 2005.

CANUDO, R. **Manifesto das sete artes.** 1923**.**

CHURCH, Alonzo. **Introduction to Mathematical Logic.** Princeton: Princeton University Press, 1996.

COPI, Irving M.; COHEN, Carl; McMAHON, Kenneth. **Introduction to Logic.** 14. ed. Boston: Pearson, 2014.

D'Ottaviano, I. L; Feitosa, H. Sobre a História da Lógica, a Lógica Classica e o Surgimento das Lógicas Não Clássicas. **Ufop.** 2003. Disponível em: https://cead.ufop.br/professores/haroldo/cursos/matematica/dte026/material/u1b.%20Ottaviano.%20historia%20da%20logica.pdf. Acesso em: 10 de maio 2024.

Denis RICHTER, Flávia Spinelli BRAGA, Mônica FÜRKOTTER. **Informática no processo ensino-aprendizagem: contribuindo para uma nova escola.** Universidade Estadual Paulista (UNESP), 2003.

Elkonin, Daniil B.. **Psicologia do jogo**. 2. ed. São Paulo, WMF Martins Fontes, 2009.

FAGUNDES, Carolina. Evolução das Linguagens de Programação: Um Panorama Histórico até os Dias Atuais. **DIO**. 2023. Disponível em:

https://www.dio.me/articles/evolucao-das-linguagens-de-programacao-um-panorama-historico-ate-os-dias-atuais. Acesso em 20 de maio de 2024.

FERNANDÉZ, Daniel; GORDILLO Aldo; CABRERA, Raúl; ALEGRE, Javier. **Comparing effectiveness of educational video games of different genres in computer science education.** Elsevier, 2023.

FRABASILE, Daniela. A tecnologia está evoluindo mais rápido do que a capacidade humana", diz Friedman. **Época negócios.** 2018. Disponível em**:** https://epocanegocios.globo.com/Tecnologia/noticia/2018/03/tecnologia-esta-evoluindo-mais-rapido-do-que-capacidade-humana-diz-friedman.html.Acesso em: 09 de maio de 2024.

FRANÇA, Victor. Aristoteles. **Slideshare.** 2014. Disponível em: https://pt.slideshare.net/slideshow/aristoteles-40659801/40659801. Acesso em 05 de maio de 2024.

GEE, James Paul. **What Video Games Have to Teach Us About Learning and Literacy.** New York: Palgrave Macmillan, 2003.

Geloneze, F. R.; Arielo, F. S. **Um breve análise sobre a Indústria de Jogos Eletrônicos e os Indie Games.** 8. Ed. São Paulo: Revista Multiplicidades, 2017.

GULARTE, Daniel. Space Invaders (Taito, 1978). **Bojogá.** 2020. Disponivel em: https://bojoga.com.br/artigos/retroplay/arcade-pinball/space-invaders-taito-1978/. Acesso em: 25 de maio de 2024.

Inteligência Artificial: riscos, benefícios e uso responsável. **Usp.** 2021. Disponível em: https://www.revistas.usp.br/eav/article/view/185020#:~:text=A%20Intelig%C3%AAncia%20Artificial%20pode%20tornar,existentes%20e%20trazer%20novos%20riscos. Acesso em: 26 abril 2023.

KARINE, Daniela ; MURILO, Hiago. **Jogos digitais e desenvolvimento cognitivo: um estudo com crianças do Ensino Fundamental.** Revista neuropsicológica latino-americana, 2016.

KIRRIEMUIR, John; MCFARLANE, Angela. **Literature Review in Games and Learning.** Futurelab, 2004. Disponível em:

http://www.futurelab.org.uk/resources/publications-reports-articles/literature-reviews/Literature-Review378. Acesso em: 5 julho de 2024.

Lucidchart. **Google Play.** 2024. Disponível em: https://play.google.com/store/apps/details?id=com.lucidchart.android.chart&hl=pt. Acesso em: 05 de julho de 2024.

MARCONI, Marina de Andrade; LAKATOS, Eva Maria. **Fundamentos da metodologia científica.** 8. ed. São Paulo: Atlas, 2018.

MARIA, Ítala ; ARAUJO, Hércules. **Sobre a história da lógica, a lógica clássica e surgimento das lógicas não-clássicas**. Universidade Estadual Paulista (UNESP), 2003.

Mattar, J. **Games em educação: como os nativos digitais aprendem.** São Paulo: Pearson, 2010.

McCARTHY, J; MINSKY, M; ROCHESTER, N; SHANNON, C. **A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence.** 1955.

McGonigal, Jane. **A realidade em jogo**: Por que os games nos tornam melhores e como eles podem mudar o mundo. Rio de Janeiro, Best Seller, 2012.

Mickey's Safari In Letterland. **MobyGames.** 2007. Disponivel em: https://www.mobygames.com/game/29744/mickeys-safari-in-letterland/. Acesso em: 25 de maio de 2024.

Newzoo's Global Games Market Report 2023 | May 2024 Update. **Newzoo.** 2024. Disponivel em: https://newzoo.com/resources/trend-reports/newzoo-global-games-market-report-2023-free-version. Acesso em: 25 de maio de 2024.

Norvig, P.; Russell, S. **Inteligência artificial: Uma abordagem moderna.** 3. Ed. São Paulo: Gen LTC, 2013.

NUNES, Carinna. **Uso de laboratórios remotos integrados a jogos digitais para o ensino de física.** Universidade estadual de Campinas (UNICAMP), 2019.

O que e GameMaker Engine?. **Make Indie Games.** 2023. Disponível em: https://makeindiegames.com.br/desenvolvimento/o-que-e-gamemaker-engine/. Acesso em: 05 de julho de 2024.

OLÍMPIA, Erica ; ABREU, Raphaella. **Jogos digitais no processo de ensino e aprendizado do aluno na alfabetização.** Instituto Federal Goiano (IFG), 2022.

PETRI, Giani; GRESSE, Christiane; FERRETI Adriano. **MEEGA+: Um Modelo para a Avaliação de Jogos Educacionais para o ensino de Computação.** Revista Brasileira de informática (RBIE), 2019.

PIAGET, Jean. **A Formação do Símbolo na Criança: Imitação, Jogo e Sonho, Imagem e Representação.** São Paulo: Editora LTC, 2001.

PIRES, Diego. **Jogos didáticos nas aulas práticas de Anatomia Humana como recurso complementar para estudo no ensino superior.** Universidade Federal do Rio Grande do Sul (UFRGS), 2022.

PIUBELLO, Henrico. Guia Completo de Lógica de Programação: Conceitos Essenciais e Exemplos Práticos. **CodeCrush.** 2023. Disponível em: https://codecrush.com.br/blog/logica-de-programacao. Acesso em: 04 de julho de 2024.

Pixilart. **Pixilart.** 2023. Disponível em: https://www.pixilart.com/branding. Acesso em: 04 de julho de 2024.

Pong, o jogo que deu origem à indústria de videogames há 5 décadas. **BBC News Brasil.** 2022. Disponível em: https://www.bbc.com/portuguese/geral-60039831. Acesso em: 25 de maio de 2024.

PRENSKY, Marc. **Digital Game-Based Learning.** New York: McGraw-Hill, 2001.

RAHIMI, Seyedahmad; SHUTE, Valerie; FULWIDER, Gary**. Timing of learning supports in educational games can impact students' outcomes.** Elsevier, 2022.

RENATO, Paulo. Da primeira à última geração: a evolução dos jogos eletrônicos. **Ingram.** 2021**.** Disponivel em: https://blog.ingrammicro.com.br/gaming/evolucao-dos-jogos-eletronicos/. Acesso em: 25 de maio de 2024.

SCREPANTI, Mainara. Crescimento do mercado de programação: veja o que especialista explica sobre o assunto. **RNTI.** 2024. Disponível em: https://revistati.com.br/noticias/crescimento-do-mercado-de-programacao-veja-o-que-especialista-explica-sobre-o-

assunto#:~:text=De%20acordo%20com%20dados%20do,cerca%20de%204%2C2%25. Acesso em: 25 de maio de 2024.

SHAFFER, David Williamson. **How Computer Games Help Children Learn.** New York: Palgrave Macmillan, 2006.

TADEU, Vinicius ; TORTELLA, Tiago. Público gamer cresce e 3 em cada 4 brasileiros consomem jogos eletrônicos. **CNN Brasil.** 2022. Disponível em: https://www.cnnbrasil.com.br/tecnologia/publico-gamer-cresce-e-3-em-cada-4-brasileiros-consomem-jogos-eletronicos/. Acesso em: 25 de maio de 2024.

TASSINARI, Ricardo **O que é a Lógica?** Universidade Estadual Paulista (UNESP), 2012.

TASSINARI, Ricardo P. O que é a Lógica? **Unesp de Marília.** 2012. Disponível em: https://www.marilia.unesp.br/Home/Instituicao/Docentes/RicardoTassinari/Intro.pdf. Acesso em: 10 de mai 2024.

Tatiara HELENA, Gladys ROBERTA. **O desafio pedagógico da geração alpha.** Universidade Federal de Rondônia (UNIR), 2021.

Tonéis, Cristiano N. **Os games na sala de aula**: Games na educação ou a gamificação da educação? São Paulo, Bookess, 2017.

Undertale. **Undertale.** 2015. Disponível em: https://undertale.com/. Acesso em 05 de julho de 2024.

VAN ECK, Richard. **Digital Game-Based Learning: It's Not Just the Digital Natives Who Are Restless.** EDUCAUSE Review, vol. 41, n. 2, p. 16-30, 2006.

XAVIER, Lucas. Criando games: conheça o mercado de jogos digitais! **Unifebe.** 2018. Disponível em: https://www.unifebe.edu.br/site/blog/criando-games-conheca-o-mercado-de-jogos-digitais/. Acesso em: 25 de maio de 2024.