CENTRO UNIVERSITÁRIO LUTERANO DE PALMAS

Recredenciado pela Portaria Ministerial nº 1.162, de 13/10/16, D.O.U. nº 198, de 14/10/2016
AELBRA EDUCAÇÃO SUPERIOR - GRADUAÇÃO E PÓS-GRADUAÇÃO S.A.

Stefan Lucas Aquino Silva

LOGI KINGDOM: Implementação das mecânicas e dinâmicas de um jogo para dispositivos móveis baseado em uma HQ de lógica

Palmas – TO

2022

Stefan Lucas Aquino Silva

LOGI KINGDOM: Implementação das mecânicas e dinâmicas de um jogo para dispositivos móveis baseado em uma HQ de lógica

Projeto de Pesquisa elaborado e apresentado como requisito parcial para aprovação na disciplina de Laboratório de Criação do curso de bacharel em Ciência da Computação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Prof.a Dr.a Parcilene Fernandes de Brito.

Palmas – TO

2022

Stefan Lucas Aquino Silva

LOGI KINGDOM: Implementação das mecânicas e dinâmicas de um jogo para dispositivos móveis baseado em uma HQ de lógica

Projeto de Pesquisa elaborado e apresentado como requisito parcial para aprovação na disciplina de Laboratório de Criação do curso de bacharel em Ciência da Computação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Prof.a Dr.a Parcilene Fernandes de Brito.

Aprovado em: _____/_____/_____

BANCA EXAMINADORA

_____

Prof.a Dr.a Parcilene Fernandes de Brito

Orientador

Centro Universitário Luterano de Palmas – CEULP

_____

Prof. M.e Jackson Gomes de Souza

Centro Universitário Luterano de Palmas – CEULP

_____

Prof. M.e Fabiano Fagundes

Centro Universitário Luterano de Palmas – CEULP

Palmas – TO

2022

RESUMO

SILVA, Stefan Lucas Aquino. **Logi Kingdom:** Implementação das mecânicas e dinâmicas de um jogo para dispositivos móveis baseado em uma HQ de lógica. 2022. 25 f. Projeto Tecnológico (Graduação) – Curso de Ciência da Computação, Centro Universitário Luterano de Palmas, Palmas/TO, 2022.

Os jogos fazem parte da cultura das pessoas e cada vez mais, com o avanço da tecnologia, os jogos digitais estão, também, sendo usados para tornar mais lúdicas as atividades de ensino. Considerando estes pontos, esta pesquisa foca no desenvolvimento de um jogo para dispositivos móveis baseado na história em quadrinhos Logi Kingdom, que traz conhecimentos da lógica proposicional para seu mundo distópico onde não existe mais lógica e os personagens enfrentam vários desafios, tendo como base as dez regras do cálculo proposicional. O presente trabalho apresenta o desenvolvimento do jogo concentrando-se na implementação das suas mecânicas e dinâmicas, utilizando a plataforma *Unity*. A partir da construção do *game design document,* que é onde são descritos todos os elementos principais que produzirão o maior impacto na experiência do jogador. Implementando com a *Unity* a interface montando todos os elementos visuais, baseado em camadas para e por fim a implementação das mecânicas utilizando componentes da própria *engine* facilitando a produção, os elementos do jogo são estruturados em *Game objects*, utilizando componentes como *transform* para cuidar da posição e elementos como *rigidbody* para cuidar da física. Assim obtendo um produto em que os jogadores possam aprender e se divertir. Por esta razão, este trabalho dedicou-se ao desenvolvimento do jogo referido, bem como das mecânicas que aplicam o conceito da lógica que desafiam os conhecimentos lógicos de quem se propõe a cumprir as missões e completar os níveis do jogo.

Palavras-chave: Lógica proposicional. Jogos educacionais. Revista em quadrinhos.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

HQ - História em quadrinhos.

GENTE - Grupo de Estudos em Novas Tecnologias para Processos de Ensino e Aprendizagem do Centro Universitário de Palmas CEULP/ULBRA.

CEULP/ULBRA – Centro Universitário Luterano de Palmas.

GDD – *Game Design Document*.

NPC - *non-player character*

APK - *Android Application Pack*

SUMÁRIO

1 INTRODUÇÃO

Segundo Souza (2020, p. 12), a sociedade atual vive um período em que os jogos digitais estão presentes na vida de todos, principalmente crianças e adolescentes, que estão cada vez mais imersos no mundo da tecnologia. Assim, os jogos vão além de sua função de entretenimento e entram no ambiente escolar como parte da abordagem de ensino e aprendizagem em diversas áreas. Algumas vantagens de inserir os jogos no ensino são objetivos e regras claras, uma percepção instantânea de objetivos que foram atingidos, bastante interatividade, realização de desafios e um certo nível de competição, associada a um alto nível de envolvimento e motivação (KICKMEIER-RUST et al., 2007, p. 648).

Hoje em dia os jogos estão cada vez mais presentes nas escolas, acessíveis a partir de computadores ou dispositivos móveis, o que tem contribuindo com o desenvolvimento de novos ambientes onde a interatividade entre artefatos e os utilizadores é direta, proporcionando uma motivação que é fundamental para a aprendizagem (MARQUES; SILVA, 2009, p. 1358). Nesse sentido, "os jogos oferecem a oportunidade de ampliar o potencial do uso de imagens, animações e interatividade, além de resgatar o aspecto lúdico e prazeroso da aprendizagem" (HAGUENAUER et al., 2007, p. 2).

"Atividades potencialmente lúdicas desenvolvidas de maneira contextualizada podem ser uma alternativa para uma renovação das práticas pedagógicas" (SOUZA e SALVADOR, 2019, P. 11). Utilizando metodologias lúdicas e interativas, os jogos vêm conseguindo espaço em meio às salas de aula, estimulando os alunos com desafios dinâmicos. Desta forma os jogos passam a ser uma ótima forma de engajar os alunos em disciplinas que necessitam de um esforço e dedicação maior, transformando aqueles conteúdos considerados difíceis em desafios interativos e divertidos.

O uso de métodos lúdicos para ensinar e aprender Lógica pode promover a capacidade de compreensão e raciocínio dos alunos, pois, "o estudo da lógica é o estudo dos métodos e princípios usados para distinguir o raciocínio correto do incorreto" (COPI, 1978, p. 19). Nolt e Rohatyn (1991, p. 1) explicam que a lógica é o estudo de argumentos que são uma sequência de enunciados onde um é a conclusão e o restante premissas que servem para provar ou fornecer informação para a conclusão. Nesse sentido, o uso de jogos como parte de uma abordagem lúdica para o ensino de Lógica pode motivar os alunos a compreender melhor e abstrair o conteúdo.

Com o intuito de auxiliar e motivar os alunos da disciplina de lógica, o Grupo de Estudos em Novas Tecnologias para Processos de Ensino e Aprendizagem (GENTE) do Centro Universitário Luterano de Palmas CEULP/ULBRA, iniciou o projeto *iLogic* que visa

ensinar com metodologias lúdicas baseadas em histórias fictícias e histórias em quadrinhos, seguindo com a criação de jogos neste contexto.

Segundo Freyermuth (2015, p. 144), os jogos digitais não seguem um roteiro ou *script* pré determinado, pois a estrutura do *game design* depende da equipe envolvida no projeto. O *game design document* apresenta também informações sobre personagens e sobre a interface do jogo. Os personagens são descritos como figuras que podem ser controladas ou não pelo jogador (*non-player character* ou NPC), geralmente eles são responsáveis por desenvolver a história e o enredo do jogo. Na interface é apresentada a descrição clara dos elementos visuais que ficarão presentes na tela e que o usuário irá interagir. Nesse sentido, segundo Lima (2013, p. 23-24) o *game design document* é definido como um projeto textual contendo tudo que será implementado dentro do jogo, desde as mecânicas primárias que são os elementos que o jogador terá contato e que são essenciais para a conclusão do jogo até a dinâmica que são experiências que o jogador terá com a execução das mecânicas.

Com isto, buscando inovar e facilitar o aprendizado da lógica proposicional, o presente trabalho tem como propósito implementar as mecânicas e dinâmicas do jogo utilizando a plataforma *Unity*. Nesse sentido, foram desenvolvidas funcionalidades que permitem que o jogador possa interagir com elementos contidos na história e possa avançar no jogo tendo a melhor experiência.

Neste contexto, o presente trabalho apresenta conceitos relacionados a jogos possibilitando uma melhor compreensão sobre a área e a fase de desenvolvimento do *Design* do jogo. Explica também como as mecânicas nos jogos funcionam e a sua divisão entre mecânicas primárias e secundárias. Além de apresentar as possíveis plataformas em que o jogos podem ser executados e trabalhos relacionados que possuem a mesma proposta.

2 REFERENCIAL TEÓRICO

Esta seção apresenta os conceitos de jogos e jogos digitais e *mobiles,* aponta o que são as mecânicas e como estão divididas e apresenta trabalhos relacionados que tem como foco o ensino.

2.1 JOGOS

Os jogos fazem parte do cotidiano das pessoas desde os tempos mais antigos, presentes na vida não só no período da infância, mas também em outros momentos. Os jogos podem ser instrumentos educacionais bastante eficientes, pois além de motivar e facilitar o aprendizado, divertem, aumentando a capacidade de retenção dos jogadores, exercitando funções mentais e intelectuais (ROLAND, 2004, p. 2).

Para Falkembach (2006, p. 1), os jogos são exercícios que servem para treinar as habilidades cognitivas e a imaginação, assim como as brincadeiras de rua ou os desafios lógicos, são atividades lúdicas que prendem a atenção do público, passando informações e estimulando diversos sentidos sem se tornar cansativo. Salen e Zimmermann (2012, p. 35), complementam que "os jogos refletem os valores da sociedade e da cultura onde são jogados porque fazem parte da estrutura dessa própria sociedade".

De acordo com Grübel e Bez (2006, p. 1), os jogos educacionais, tanto digitais quanto físicos, podem ser utilizados como um recursos riquíssimos para desenvolver conhecimentos e habilidades dos alunos, pois segundo Franco et al. (2018, p. 2) os jogos auxiliam na construção estruturada da leitura e interpretação de textos, contribui para o desenvolvimento social e colabora na interação entre alunos e professores. Os jogos educacionais digitais possuem mais recursos para criar e desenvolver conhecimentos e habilidades do que os métodos educacionais tradicionais, tornando-os um ótimo recurso didático ou estratégico para o ensino de diversas áreas do conhecimento.

Para Murcia (2005, p. 9) os jogos estão sendo uma ótima ferramenta para o ensino e aprendizagem, mas não eram bem vistos pela pedagogia tradicional, que consideravam que a educação e os jogos não poderiam ser aliados na melhoria do desempenho escolar. No entanto, a pedagogia tradicional mudou de opinião quanto aos jogos, reconhecendo-os como uma forma lúdica capaz de auxiliar na formação de pessoas (MURCIA, 2005, p. 9).

Segundo Silva (2019), "existe uma variedade de atividades didático-pedagógicas associadas ao desenvolvimento de raciocínio lógico: jogos de raciocínio, robótica educacional, objetos de aprendizagem e pensamento computacional". Nesse contexto, uma

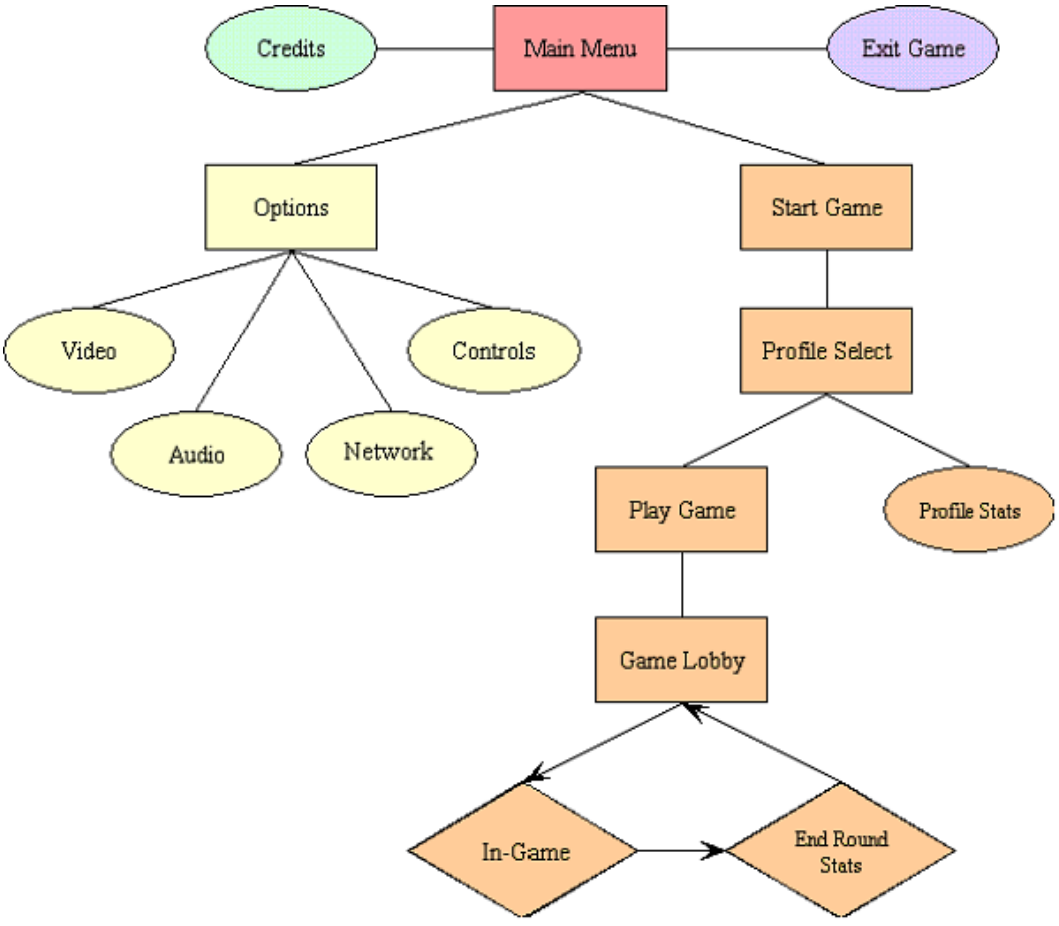
abordagem narrativa pode se tornar eficaz, mantendo a retenção do jogador ao acompanhar uma história atrativa, garantida por um bom roteiro.

2.1.1 Design de Jogos

O processo de *design* do jogo é a fase de planejamento e estruturação, nesse processo os desenvolvedores se concentram nos elementos principais que produzirão o maior impacto na experiência do jogador (ZICHERMANN e CUNNINGHAM 2011, p. 35). Este processo pode ser considerado como uma das principais etapas na elaboração de um jogo, pois é nesta fase que é feito todo o planejamento do projeto por parte do *game designer* que especifica o que a equipe de desenvolvimento deverá implementar.

Durante o processo de elaboração do *design* do jogo é criado o (GDD) *Game Design Document* que, segundo Freyermuth (2015, p. 144), é responsável por manter todos os detalhes do jogo, sejam físicos, funcionais ou estéticos, bem como espaços de narração de histórias, além das regras e condições das interações do usuário.

Figura 1. Diagrama de Fluxo de telas.

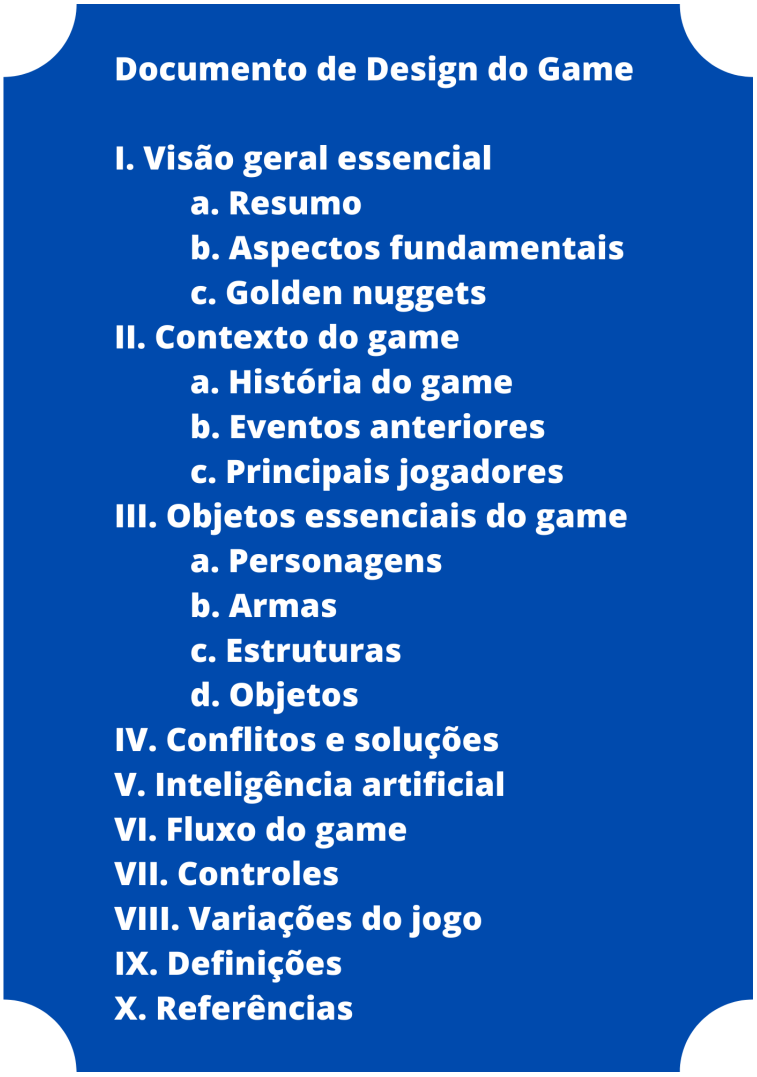


Fonte: GAMUX, (2011).

Um dos elementos do GDD é um diagrama do fluxo de telas como é ilustrado na Figura 1, que estabelece ordem e como o jogador transita por elas. Ainda na construção do *game design* é definida a forma estrutural do jogo através das regras, que se fazem essenciais para a qualidade do mesmo (SALEN; ZIMMERMAN, 2012). Essas regras definem os tipos de jogabilidade que o jogador terá e, com isso, são estabelecidas quais mecânicas serão utilizadas para executar essas regras.

O Game Design Document não segue uma estrutura rígida, ou seja, a equipe pode optar por elaborar um documento mais complexo ou mais simples dependendo do tipo de jogo que desenvolvem. A figura abaixo apresenta alguns elementos que podem ser utilizados para compor um GDD. Como a descrição das mecânicas que devem ser implementadas no presente trabalho.

Figura 2. Elementos que compõem o GDD.



Fonte: Schuytema, 2008.

A figura acima apresenta alguns elementos de um GDD, dispondo de uma visão geral das características de um jogo, uma breve síntese de como será, uma descrição do “universo” onde se passará à história que é descrita de forma resumida, partindo para o detalhamento de objetos essenciais como os personagens armas estruturas e seguindo para elementos técnicos como como fluxo, controles e definições seja gráfica ou implementação.

Segundo Azevedo (2017, p. 62) sem o desenvolvimento de um GDD que aborde todos os elementos que o jogo deverá ter, o roteiro por si só, não seria capaz de apresentar todas as características e mecânicas pertencentes ao jogo. Um bom GDD deve conter todas as características do jogo, mecânicas, dinâmicas, elementos visuais ou qualquer outro componente que seja importante para a criação do jogo. “Ou seja, o GDD – *Game Design Document* consiste de um documento complexo, concebido antes de qualquer outra etapa ou ação que possa estar relacionada à produção de um *game*” (AZEVEDO, 2017, p. 60).

2.1.2 Mecânicas do Jogos

As mecânicas descrevem como o jogador irá interagir com o jogo, definem como o controle do jogo irá funcionar, quais tipos de provocações estarão presentes no ambiente e estratégias que o jogador deverá utilizar para que seja possível terminar desafios, (CHANDLER, 2009, p. 401). Segundo Werbach e Hunter (2012, p. 79), as mecânicas são os processos básicos que proporcionam as ações que geram engajamento do jogador. Um exemplo são as recompensas que podem ser itens, pontos ou moedas como benefícios por uma determinada ação.

Segundo Freyermuth (2018, p. 168) os jogos digitais que desejam contar uma história ao longo do processo de imersão do jogador devem apresentar mecânicas que possibilitem a narrativa do jogo. A relação entre mecânicas e dinâmicas constituem um papel essencial para que o jogador tenha uma experiência agradável ao jogar. Dentre os componentes dos jogos, as mecânicas são as partes mais importantes, pois proporcionam o engajamento do jogador e delimitam as regras do jogo.

A indústria de *design* de jogos classifica alguns elementos importantíssimos que constituem a mecânicas de jogos, Schell (2019) evidencia algumas:

- O espaço funcional do jogo: dimensões, características, limites e cenários;
- Os objetos que podem ser encontrados no espaço do jogo: seus atributos, seus estados e suas dinâmicas;
- As ações que são possíveis no espaço do jogo: as possibilidades de navegação, atuação, comunicação e manipulação de objetos;

- As regras do jogo que controlam e coordenam as sub-regras do espaço do jogo, objetos e ação: por exemplo, as relações temporais das ações ou a relação entre habilidade e acaso no decorrer e resultado do jogo;
- Os objetivos do jogo: sua concretude, seu alcance, as recompensas oferecidas e assim por diante.

As mecânicas podem ser divididas em dois tipos principais, sendo elas mecânicas primárias e secundárias, cada uma aborda um tipo de interação com o jogador. No presente trabalho serão abordadas as mecânicas primárias e secundárias no desenvolvimento de um jogo *mobile* baseado em uma história em quadrinho.

2.1.3.1 Mecânicas Primárias

Mecânicas primárias são aquelas que definem ações básicas do jogador, porém são cruciais para que o jogador consiga progredir e terminar o jogo. “A mecânica primária está prontamente disponível, explicada nos estágios iniciais do jogo e consistente ao longo da experiência de jogo” (SICART, 2008). Mörshbächer (2015, p. 35) complementa dizendo que esse tipo de mecânica é a que pode ser aplicada para a resolução de problemas que levam ao resultado final de um estado. Para exemplificar, tem-se a popular franquia *Grand Theft Auto*, em que as mecânicas primárias são o movimento do personagem, ação de atirar e controlar veículos.

Essas mecânicas são o que define qual a estrutura de jogo, podendo inclusive definir o gênero, ao contrário de outros tipos de mídias, como a literatura ou cinema onde o gênero define as demais estruturas da obra (FREYERMUTH, 2015, p. 218). As mecânicas são as estruturas descritas no *game design* dentro da jogabilidade geralmente descritas como verbos, por exemplo escolher, mover, mirar, atirar, coletar, chutar ou socar (JÄRVINEN, 2008, p. 263). A Figura 2, apresenta um exemplo de mecânica primária presente no jogo *One Piece Bounty Rush*.

Figura 3. Exemplo visual de uma mecânica primária.

Fonte: ***One Piece Bounty Rush - Bandai Namco*****.**

A figura acima apresenta parte do jogo *One Piece Bounty Rush* baseado na franquia *One Piece* do autor Eiichiro Oda. O jogo possui quatro personagens que devem ser controlados alternadamente pelo jogador, competindo com outras equipes que também possuem quatro jogadores, além disso, cada batalha é baseada em um local do anime que jogo foi baseado. A trama do jogo consiste em vencer o time oponente conquistando mais moedas ao final de cada batalha. A mecânica apresentada na figura acima é do tipo primária, pois para vencer os oponentes o jogador precisa utilizar golpes que eliminam o oponente, como socos e chutes.

Complementando, Järvinen (2008, p. 398) apresenta que, quando uma habilidade cognitiva ou psicomotora é encontrada nos jogadores, essa habilidade, mesmo que não esteja ligada diretamente à mecânica, pode ser desenvolvida de acordo com o decorrer do jogo devido a repetição de mecânicas, tornando-se um possível subproduto. Nesse sentido, é possível encontrar diversos trabalhos relacionados com intuito de ensinar através de metodologias recreativas, abrangendo o ensino e exemplificando a produção de um jogo, como o trabalho de Mioto et al. (2017, p. 2) que propõe ensinar raciocínio lógico e o trabalho de Oliveira et al. (2018, p. 02) que se propõe a ensinar lógica de programação.

2.1.3.2 Mecânicas Secundarias

As mecânicas secundárias são aquelas que ajudam na conclusão de desafios e objetivos do jogo, mas não são obrigatórias, Sicart (2008, n.p). Estas mecânicas geralmente estão sempre presentes no jogo, mas não são necessárias para que se conclua o jogo, porém são funcionais. A figura 4 abaixo ilustra uma mecânica secundária do jogo Ben 10: Protector of Earth.

Figura 4. Exemplo visual de uma mecânica secundária.

Fonte: Ben 10: Protector of Earth - *D3 Publisher*

O jogo Ben 10: Protector of Earth é um jogo de ação e aventura desenvolvido pela High Voltage Software e publicado pela D3 Publisher e é baseado na série animada de televisão Ben 10. No game há uma mecânica de planagem ao jogar com o personagem Chama. Ao dar um segundo pulo e ficar bem alto ao segurar a tecla de pulo permite que o personagem caia devagar planando podendo alcançar lugares ainda mais altos ou escondidos como mostra a figura 4.

Com base nisto uma outra forma de descrever as mecânicas secundárias segundo Zaffari e Battaiola (2021, p. 277), são as regras que facilitam o alcance do objetivo geral apresentado em um jogo, mas não são obrigatórias que o jogador as use para alcançar o objetivo final. Estas mecânicas criam geralmente cria uma experiência agradável para o jogador, tornando o jogo mais divertido e imersivo.

Ainda assim, para Sicart (2008, n.p), as mecânicas devem ser voláteis onde uma mecânica primária pode se tornar secundária e o contrário, durante o progresso do jogo. Com isto as mecânicas utilizadas devem ser categorizadas e bem estruturadas devido seu impacto na experiência. É comum essa mudança durante o jogo em *games* produzidos para

dispositivos móveis, que possuem uma estrutura simplificada, porém bastante imersiva e atrativa.

2.2 Jogos Digitais e *Mobiles*

Segundo Lucena (2014, p. 17) os jogos digitais fazem parte de um fenômeno cultural que envolve diferentes gerações da sociedade atual, devido ao seu caráter interativo, imersivo e interconectado tornou-se uma das maiores indústrias de entretenimento nos últimos anos. Com a facilidade de acesso a equipamentos eletrônicos portáteis como tablets e smartphones, os jogos digitais ganharam espaço no mercado de entretenimento e passaram a ter grande importância para as indústrias dos jogos.

Os telefones celulares a princípio não foram projetados para jogos, no entanto, à medida que foram avançando e se tornando populares entre as pessoas, a indústria dos jogos começou a aproveitar todos os recursos que os dispositivos móveis poderiam oferecer para o lucrar e contribuir com o entretenimento do público (BYL, 2017, p. 384). Interação com telas sensíveis ao toque, sensores de movimento, serviços de localização e praticidade foram alguns dos motivos que contribuíram para o crescimento do mercado de jogos mobiles.

Os jogos mobiles possuem uma tendência de crescimento para os próximos anos, segundo a companhia de pesquisa e análise de jogos Newzoo, os jogos para dispositivos móveis movimentaram em 2021 aproximadamente US$ 90,7 bilhões e com uma taxa de crescimento de 4,4% ao ano. Com um mercado tão lucrativo, desenvolver jogos que atendam aos mais variados públicos pode ser uma jogada bastante lucrativa para empresas que desejam entrar nesta área, devido ao potencial vasto que os jogos possuem, não só no campo do entretenimento mas também como ferramenta de ensino.

Dentre os diversos jogos para dispositivos móveis estão os jogos educativos que têm encontrado espaço nas salas de aula para complementar e trazer para o ambiente escolar recursos digitais que contribuem para o desenvolvimento racional e cognitivo dos alunos. Silva et al (2017) apresenta em seu trabalho o “Desafios das Diagonais”, um jogo mobile para ensinar conceitos simples da matemática e do raciocínio lógico, a Figura 5, apresenta algumas telas do jogo.

Figura 5. Telas do jogo “Desafios das Diagonais”.

Fonte: Silva et al (2017).

A figura acima apresenta algumas telas do trabalho desenvolvido por Silva et al (2017). O jogo é composto por 10 fases, sendo que em cada uma das fases o jogador é desafiado a resolver problemas matemáticos de raciocínio lógico, o que envolve operações básicas e figuras geométricas, e à medida que o jogador avança nas fases os desafios vão ficando cada vez mais complexos, fazendo com que o jogador aprenda conceitos novos de matemática enquanto se diverte jogando.

Os jogos podem potencializar o ensino e aprendizagem de diversas disciplinas nos mais variados segmentos, pois possuem um efeito motivador e desafios que em sua maioria estão relacionados com o engajamento e a interatividade dos jogadores. As mecânicas e dinâmicas dos jogos, quando bem trabalhadas, podem auxiliar alunos e professores no processo de ensino de disciplinas que são consideradas mais complexas e que possuem uma taxa de reprovação muito alta.

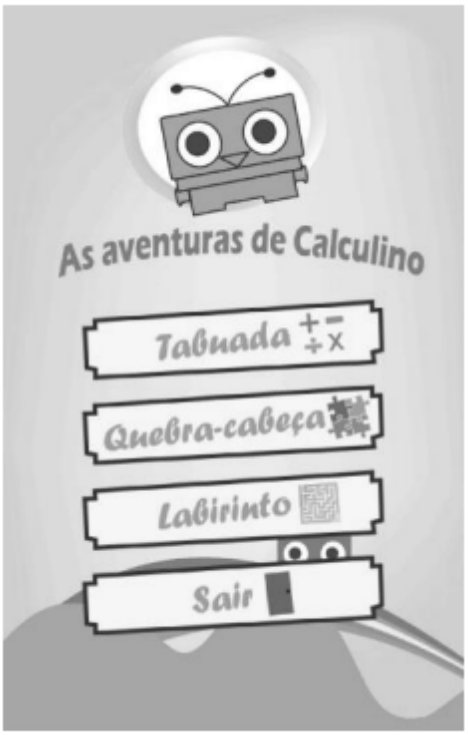
2.3 Trabalhos relacionados

Em Mioto et al. (2017, p. 2) é citado que diversas tecnologias podem ser utilizadas no contexto da educação e que é necessário ter um embasamento teórico metodológico bem claro. As “Aventuras de Calculino” é um jogo educacional para o ensino de raciocínio lógico matemático com um enredo sobre um robô chamado Calculino, dividido em três módulos, tabuada, labirinto e quebra-cabeças.

Esse jogo foi desenvolvido utilizando o motor de jogos 2D *Jeasy Game* e as ilustrações foram feitas no software *Corel Draw*. Trata-se de um jogo educacional para dispositivos móveis voltado para o público infantil, que tem por objetivo o estímulo de raciocínio lógico. O objetivo é buscar o equilíbrio entre educação e ludicidade, favorecendo o

desenvolvimento do raciocínio lógico por meio de um jogo que motive enquanto a criança aprende e se diverte.

Figura 6. Tela inicial do jogo Aventuras de Calculino.



Fonte: Mioto et al., (2017).

Aventuras de Calculino (Figura 6) foi avaliado por nove crianças de 8 a 12 anos, aplicando testes de usabilidade e divertimento. No teste de usabilidade foram utilizadas as heurísticas simplificadas e adaptadas para crianças, e as heurísticas de Malone que permitem avaliar o divertimento proporcionado por uma interface com o usuário.

Continuando nesse contexto educacional, encontra-se o *MazeLogic*: Jogo Educacional para Ensino de Lógica de Programação, apresentado na Figura 7. Oliveira et al. (2018, p. 02) destacam o uso da tecnologia para auxiliar o aprendizado, que está sempre em transformação. A proposta dos autores é fazer um jogo educativo projetado para apresentar os principais conceitos de lógica de programação e raciocínio lógico computacional.

Figura 7. *MazeLogic* - modo de jogo nivelamento.

Fonte: Oliveira et al., (2018).

O jogo possui dois modos de jogos: nivelamento, que é uma espécie de quiz; e o modo labirinto. Oliveira et al. (2018, p. 02) destacam que no modo nivelamento, apresentado na figura 3, o jogador conta com uma pergunta com quatro opções de resposta, com apenas uma das alternativas correta. O modo labirinto é um método mais fácil de jogo voltado para o público mais jovem, um método mais divertido de jogo, que conta com um mapa onde as perguntas são desafios a serem alcançados para o avanço do jogo: um personagem é colocado no seu mapa em forma de labirinto e, a cada pergunta respondida de forma correta, o personagem avança para mais próximo do final.

Os trabalhos apresentados na seção destacam o uso da tecnologia para auxiliar o aprendizado, utilizando o contexto de jogos, esses trabalhos se tornam relevantes para o entendimento da produção de jogos educacionais. Dessa forma, busca-se encontrar uma maneira de abordar o ensino e aprendizagem dos conteúdos da computação, como a disciplina de lógica de programação.

3 METODOLOGIA

3.1 DOMÍNIO

3.1.1 *Lógica Proposicional*

A lógica Proposicional é uma versão simplificada da lógica formal, que estuda como raciocinar corretamente a partir de afirmações verdadeiras ou falsas, a partir de hipóteses também chamadas de proposições, que determinam se a conclusão é verdadeira ou falsa no mesmo contexto em que foi inserida (BEDREGAL e ACIÓLY, 2007, p. 40). A lógica proposicional utiliza enunciados declarativos e é capaz de lidar com diversos problemas e formalizar precisamente a busca de suas soluções (NICOLETTI, 2017, p. 2).

Nolt e Rohatyn (1991, p. 95) afirmam que na criação das fórmulas dos enunciados são utilizados três conjuntos de símbolos que compõem o vocabulário da lógica proposicional, divididos em lógicos e não-lógicos. São definidos como letras sentenciais, que são as letras maiúsculas que podem ser utilizadas para representar sentenças declarativas e conclusões de argumentos, conectivos lógicos, que são os símbolos utilizados para representar a conexão entre enunciados e argumentos (ex.: negação, conjunção, disjunção, condicional e bicondicional). E para definir a ordem de resolução dos conectivos são usados parênteses, assim como nos cálculos matemáticos.

Para a validação das formas de argumento, segundo Nolt e Rohatyn (1991), o método chamado de tabela verdade é um teste minucioso e completo. Preenchidas com todas as possibilidades de combinações de verdade associadas aos conectivos lógicos e aos símbolos proposicionais que formam o argumento. Durante o jogo Logi Kingdom, o jogador deve utilizar os conceitos e regras das tabelas-verdade para solucionar desafios que fazem parte das mecânicas e regras do jogo, assim como a revista em que o jogo se baseia utiliza as regras de inferência.

3.1.2 *Logi Kingdom*

A revista em quadrinhos Logi Kingdom utilizada como base na criação do jogo foi planejada e construída tendo as dez regras de inferência do cálculo proposicional como pauta e tema central da história (MORENO et al., 2021, p. 82). A história é apresentada em um mundo distópico e totalmente decadente onde não existe mais lógica e os personagens enfrentam vários desafios para conseguir trazer a lógica de volta.

As regras são explicadas e executadas no decorrer da história por meio dos diálogos e ações dos personagens principais, a guerreira e o fantasma. Essas regras são abordadas de acordo com o nível de dificuldade dos cálculos de cada operador lógico, que possui uma regra

de eliminação e outra de introdução, começando com as mais simples como eliminação da negação, até chegar nas regras mais complexas como a redução ao absurdo.

3.2 MATERIAIS

Para desenvolvimento do projeto foram utilizadas as seguintes tecnologias e ferramentas.

3.2.1 *Adobe Photoshop*

O *Photoshop* é um software de edição de imagens, é considerado atualmente o líder no mercado dos editores profissionais, assim como de trabalhos de pré-impressão. Criado por Thomas Knoll em 1987 durante sua tese de doutorado, o programa exibia imagens em preto e branco. Hoje no software é possível combinar diversos tipos de elementos visuais em uma única imagem, criando algo novo (PHOTOSHOP, 2021).

Photoshop será usado para produção de imagens que compõem a interface gráfica do jogo, como itens, botões e até textos personalizados, sendo processadas de forma que garanta uma ótima qualidade e em um formato compatível com a plataforma *Unity*, utilizada no desenvolvimento.

3.2.2 Plataforma *Unity*

A *Unity* é uma plataforma de desenvolvimento de jogos e aplicativos 2D e 3D, com bastante recursos, o que permite que designers, artistas e desenvolvedores colaborem e compartilhem experiências tornando seus projetos interativos. A plataforma disponibiliza recursos que ajudam seus usuários a publicarem seus projetos em diversas plataformas como PlayStation, Xbox, Nintendo Switch, Android e IOS. *Unity* conta com versões gratuitas para estudantes e desenvolvedores individuais, além das versões pagas para empresas de desenvolvimento (*UNITY*, 2021).

Aguiar (2017, p. 24) diz que é possível processar diversos recursos de jogos que são utilizados no desenvolvimento moderno através da *Unity*, o que possibilita aos desenvolvedores adotar a ferramenta em várias etapas da produção de um jogo. A *Unity* utiliza componentes chamados de cenas que trazem uma estruturação através de objetos, assim a utilização de linguagens orientadas a objetos facilita o desenvolvimento. Atualmente C# é a sua principal linguagem de programação.

3.2.3 Linguagem de programação C#

O C# é uma linguagem segura e bem atual criada pela Microsoft, que fornece estruturas de linguagem com suporte direto a conceitos de orientação à objeto, tornando-se uma linguagem natural para criar e usar componentes de software. Possui ainda recursos que ajudam a criar aplicativos robustos e duráveis, como a coleta de lixo que recupera

automaticamente a memória ocupada por objetos não utilizados inacessíveis (MICROSOFT, 2021).

A linguagem é considerada simples devido sua forma expressiva, utilizando chaves como marcação, que apontam onde inicia e onde finaliza um bloco de código, outras linguagens consideradas mais complexas como Java, C e C++ seguem essa estrutura, porém o C# simplifica muitas dessas complexidades possuindo uma biblioteca de funcionalidades maior, (MICROSOFT, 2010). Sendo assim é possível construir algoritmos, com uma estrutura mais simples, facilitando a leitura e possuindo o mesmo poder de ação que em outras linguagens. No presente trabalho em específico a linguagem é utilizada para controlar os componentes do jogo, como interface gráfica, mecânicas dos personagens e cenários, e sua narrativa, estabelecida na prototipação do desenvolvimento.

3.3 MÉTODOS

Para o desenvolvimento do presente projeto, a produção foi dividida em etapas, como apresentado na figura 8.

Figura 8. Metodologia aplicada no desenvolvimento do trabalho.

Na Figura 8 é apresentada a metodologia utilizada no desenvolvimento, tendo como consequência a conclusão do trabalho a partir dos objetivos propostos. Na primeira etapa deste processo foi feita a análise do protótipo construído através do *game design document,* levando os requisitos do que seria implementado e quais componentes seriam necessários.

Após esta análise, na etapa 2 foi realizada a configuração do ambiente da *Unity* para que execute o projeto no formato *mobile,* importando pacotes que são necessários como por exemplo a identificação de toque na tela.

Na etapa 3 onde foi realizada a construção dos cenários, utilizando vários elementos criados como *game objects* do tipo *sprite 2D* como paisagem, colunas, portas, cercas, blocos e etc. Utilizando camadas diferentes para eles, passando a sensação de que um objeto está na frente do outro, e foi aplicado colisões nestes objetos, dando ainda mais sensação de realidade. Na etapa quatro a fase de implementação das mecânicas dos personagens seguindo uma estrutura que a *engine* estabelece, utilizando componentes como *transform* e *position* para movimentação e *Rigidbody2D* para física, foram criadas funções que executam as chamadas para as mecânicas a partir dos botões da interface.

A quinta etapa é onde se encontra a implementação dos desafios, estabelecendo as regras do jogo, utilizando listas de sentenças lógicas e temporizadores, vinculando com todos os outros sistemas de pontuação e de vida da personagem. E por fim na sexta etapa após a finalização de toda implementação, é gerado o *build* do jogo, criando um arquivo APK que pode ser executado em qualquer dispositivo *Android* podendo assim ser feito o teste e a validação pela especialista do domínio.

4 DESENVOLVIMENTO

Esta seção tem por objetivo apresentar e descrever as etapas que foram realizadas no presente trabalho e os resultados obtidos.

4.1. Projeto do Jogo

O jogo foi projetado juntamente com a equipe do Logi Kingdom que ajudaram a entender o universo da revista. Resultando na criação do *game design document*, onde foi definido todos os componentes, as estruturas e as mecânicas. Com isso, seguiu-se a realização dos requisitos definidos, como manter as características da revista, está disponível para dispositivos *mobile* e dar um *feedback* ao jogador em relação às suas respostas.

4.2. Concepção do Jogo

Para o desenvolvimento se fez necessário o uso de três *scenes*, que são objetos que guardam informações de outros, como uma cena de um filme que guarda todos os elementos ali presentes. Para a construção das cenas foram utilizados `GameObjects` com componentes do tipo *sprite* compondo os cenários e personagens. Para construir os *scripts* foi utilizada a linguagem de programação C#. Esses *scripts* controlam as regras do jogo, atribuídas nos botões, cenários, interface e personagens, criando um sistema completo e funcional.

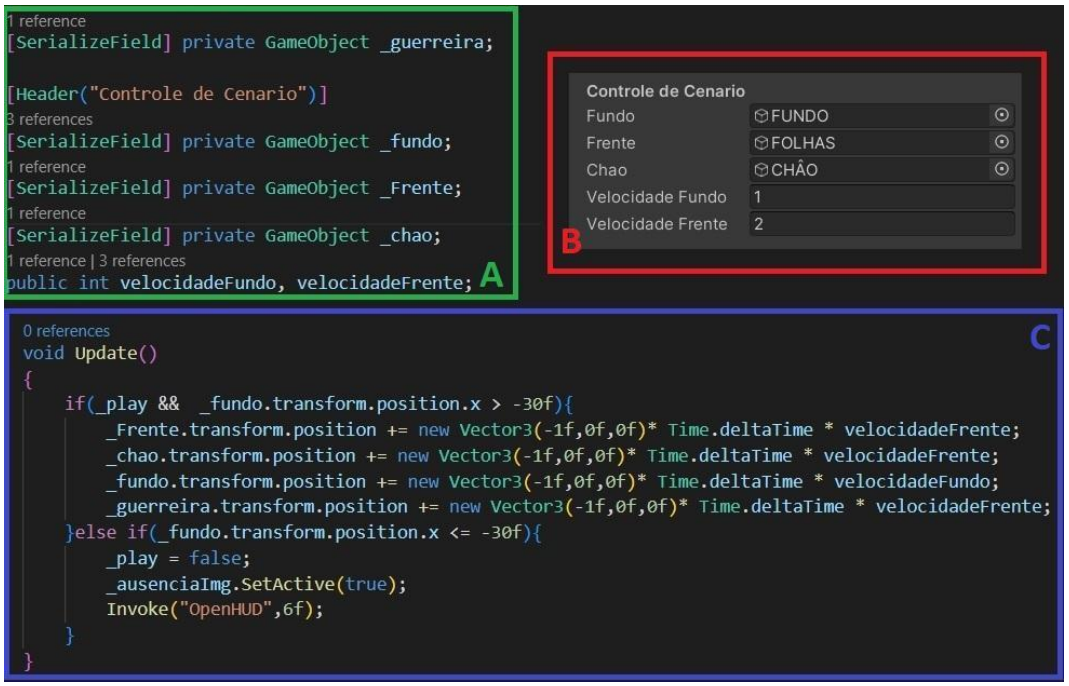
4.2.1. Narrativa

A história presente no Logi Kingdom seguirá de forma linear em cenários que lembram o universo da revista. Com a ausência da lógica, a personagem Guerreira terá que explorar esse mundo e encontrar dicas de lógica para resolver desafios e derrotar inimigos. Para contextualizar o jogador, ao iniciar o jogo uma *cutscene* aponta como funciona aquele mundo, em um cenário que se move da direita para a esquerda e frases que contam um pouco da história sobem alternadamente como mostra a Figura 9.

Figura 9. *Cutscene* inicial de contextualização do jogador.

Na figura acima é possível observar o texto no centro da tela com uma frase de contexto. Para a implementação do movimento e dos textos, foram serializadas algumas variáveis para que fosse possível editar e ajustar pela *engine*, como mostra a Figura 10, a seguir. As frases apresentadas no início do jogo servem para dar ao jogador uma introdução da história e do contexto do jogo, para que ele não fique completamente perdido.

Figura 10. Código e configuração da *cutscene* de entrada.



Na Figura 10, o bloco de código A mostra as variáveis serializadas. Os objetos são do tipo `GameObject` que é a classe padrão de objetos da *Unity*, que são indicados através da *engine* como mostra Figura 10-B, os elementos que compõem o cenário e a velocidade que irão se mover. Estes elementos que da paisagem são movidos pelo código da Figura 10-C dentro da função *update*, controlando o componente `transform` dentro de todos os `GameObjects`. A função *update* é chamada a cada `frame`, portanto nesse bloco de código verifica-se a posição do fundo observando o atributo do *x* dentro do atributo *position* do componente `transform`, representando a posição no eixo horizontal. O atributo *x* é decrementado até chegar em um valor definido que ajusta o cenário e o personagem na posição da figura 9 dentro da câmera do jogo. Para finalizar é chamada a função `openHud()`, que torna o `Joystick` visível para o jogador começar a controlar a personagem, levando-a para o módulo um.

Os elementos visuais complementam a narrativa, como os textos da *cutscene*, as corujas que representam a vida da Guerreira, as sentenças lógicas e até mesmo os controles e botões de ações da personagem, que são elementos da interface e todas as ações relacionada a eles são controladas pelo jogador.

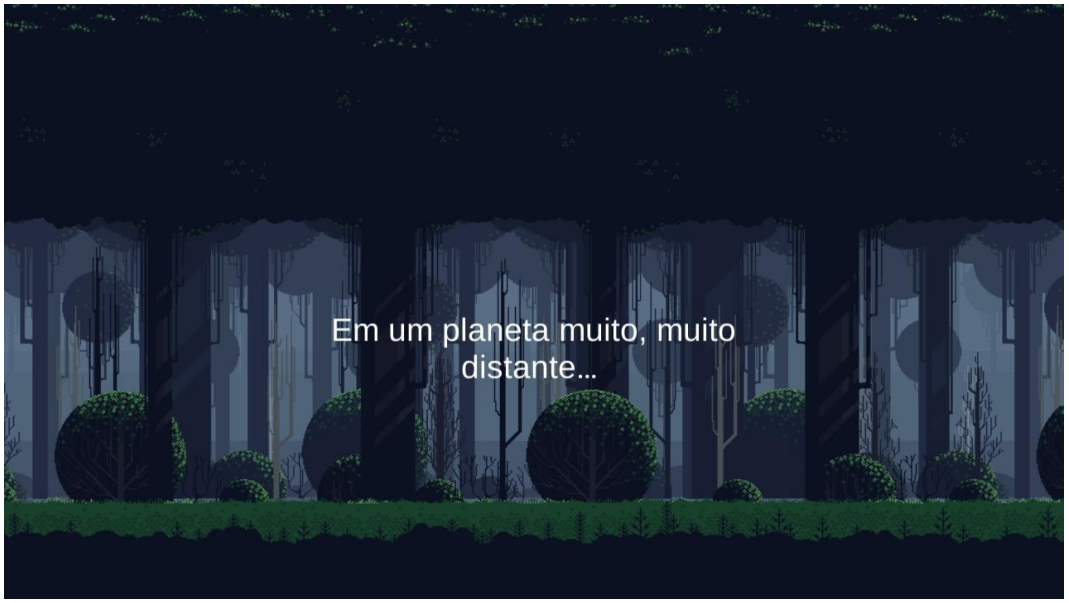
4.2.2. Cenários

Os cenários do jogo também foram baseados na revista em quadrinhos, possui três cenários fazendo referência a ambientes da HQ. Primeiro a vegetação seca do planeta, segundo a Biblioteca do conhecimento e a toca da coruja onde acontece o combate entre a

guerreira e a falaciosa. Ambos os cenários possuem características visuais próprias e com isso foram criados *scripts* para cada um se adequando às suas necessidades e mecânicas.

No primeiro cenário do jogo, a vegetação faz referência à presente na revista em quadrinhos. A história se passa em uma floresta sombria devido à escuridão que cobriu parte do céu do planeta, que resultou na incapacidade de evoluir da vegetação com árvores de baixa estatura coberta com musgo e lodo. Essa estética foi abordada no cenário um, na introdução do universo, como mostra a figura 11, controlado por *script* a *cutscene* torna o cenário vivo como descrito na narrativa a subseção 4.2.1.

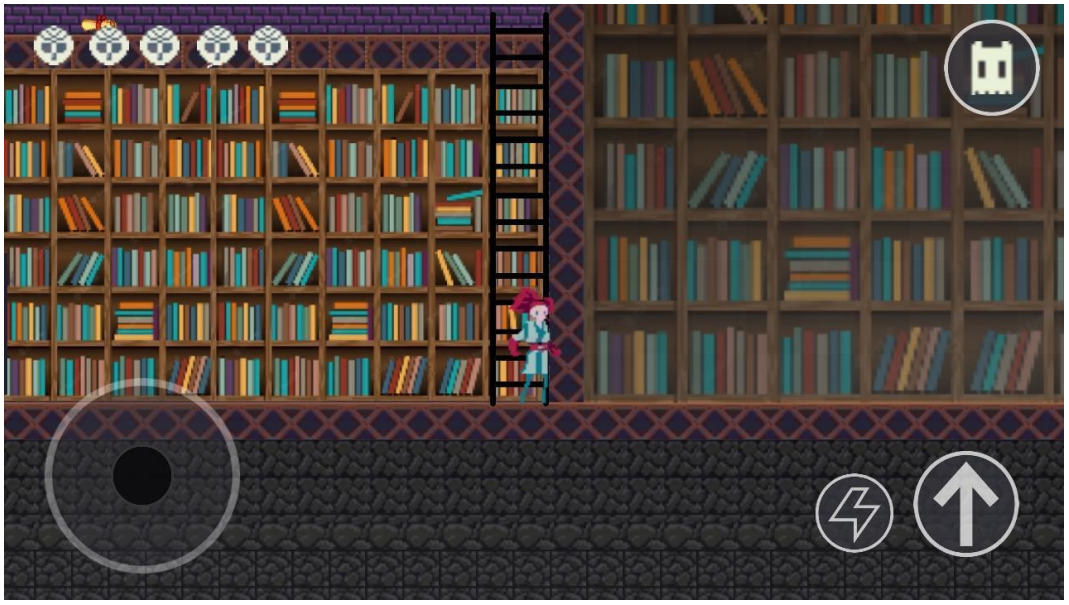
Figura 11. Primeiro cenário do jogo.



Fonte: Eder Muniz, 2022.

Assim como o cenário um da figura 11, o cenário dois, a biblioteca, possui alguns elementos visuais que fazem referência ao estilo e características da revista em quadrinhos. A biblioteca está presente em algumas páginas da revista e, por ter grande relevância para história, ela foi trazida para dentro do jogo. Neste módulo o jogador se depara com plataformas que deve pular e andar, bem como escada que deve subir, possui um *script* de controle, ajustando a câmera dependendo de onde esteja o jogador, dando a sensação de grandeza, figura 31 representa a biblioteca no jogo.

Figura 12. Segundo cenário do jogo, Biblioteca.



Após concluir o desafio presente no módulo da biblioteca representado na figura 12, o jogador vai para o terceiro cenário do jogo. O módulo dois representa a toca da coruja que é um cenário característico da HQ. Neste cenário ocorre a batalha entre a Guerreira e a Falaciosa, o jogador deve responder desafios identificando a validade das fórmulas, utilizando os botões como resposta, sendo “V” ou “F”, representando o Ataque da Guerreira como mostra a Figura 13, a seguir.

Figura 13. Terceiro cenário do jogo, Toca da Coruja.

Na toca da coruja, apresentada na figura 13, o jogador irá finalmente enfrentar os desafios respondendo proposições lógicas e enfrentar a Falaciosa que quer impedi-la de trazer a lógica para o pequeno planeta. Neste módulo, algumas mecânicas são descartadas para que a jogabilidade seja facilitada, focando apenas no ataque para que derrote a falaciosa com a

resposta correta. Os desafios são pontuados de acordo com o nível de dificuldade, possuindo tempos diferentes e complexidade maior.

4.2.3. Interface

A interface é um `GameObject` da *Unity*, definido como `prefab`, é um objeto que pode ser usado em várias cenas do jogo, porém ele possui o componente *canvas* que tem essa propriedade de agir como telas, segurando os outros `GameObjects` com outras propriedades, como imagem, botões ou textos. No seu *script* de controle existe uma função chamada ajustar, que é chamada inicialmente, dependendo do nome da cena, passado por uma variável pública ele ativa e desativa os objetos que devem estar aparentes no momento, como mostra a Figura 14.

Figura 14. Função Ajustar da interface.

```
1 reference
void Ajustar(){
    if(_cena == "Biblioteca"){
        _Joystick.SetActive(true);
        _vidas.SetActive(true);
        _btnJump.SetActive(true);
        _btnDash.SetActive(true);
        _btnFantasma.SetActive(true);
        _btnAtaqueF.SetActive(false);
        _btnAtaqueV.SetActive(false);
        _btnFantasma.GetComponent<Button>().interactable = false;
    }else if(_cena == "Inicio"){
        _Joystick.SetActive(true);
    }else if(_cena == "modulo2"){
        _Joystick.SetActive(false);
        _vidas.SetActive(true);
        _proposicao.gameObject.SetActive(true);
        _pontosText.gameObject.SetActive(true);
        _pontosText.text = "Pontos: 0000";
        _btnFantasma.SetActive(true);
        _btnAtaqueF.SetActive(true);
        _btnAtaqueV.SetActive(true);
        _btnFantasma.GetComponent<Button>().interactable = true;
        _btnAtaqueF.GetComponent<Button>().interactable = true;
        _btnAtaqueV.GetComponent<Button>().interactable = true;
    }
}
```

Os `GameObjects` possuem um atributo chamado `active`, e de acordo com o nome da cena recebe verdadeiro para que fiquem visíveis na interface através da função `SetActive`. Com a função `GetComponent()` é possível pegar o componente `Button` de alguns objetos e bloquear ou liberar o clique nestes, assim como mostra a Figura 14. A partir disso, são exibidas as telas onde o jogador pode interagir com os botões, controlando a personagem ou transitando entre as telas. Na cena inicial, a tela menu fica disponível se tornando ponto de partida para as demais, como mostra a Figura 15.

Figura 15. Tela Menu.

A partir do menu é possível seguir para as demais telas ao pressionar um determinado botão, ao pressionar o botão "começar" desativa o menu e inicia o código da *cutscene* inicial mostrada na seção anterior 4.2.2. Ao pressionar o botão "continuar" o jogador é enviado para a cena que estiver salva como último *checkpoint* através da função `LoadScene()`, como mostra o código da Figura 16.

Figura 16. O código do botão "Continuar".

```
void Awake(){
    if(SAVEGAME.instance.SaveExists()){
        jogoSalvo = SAVEGAME.instance.GetSave();
    }

0 references
public void Continuar(){
    if(jogoSalvo != null){
        SceneManager.LoadScene(jogoSalvo.cena);
    }
}
```

Como mostra a figura acima, ao verificar se existe um jogo salvo no dispositivo, a função executa o `LoadScene()` recebendo como parâmetro o nome da cena que deve ser carregada através do atributo cena do objeto carregado pela função `GetSave()`. Outra tela disponível através do menu é a tela de *ranking* onde mostra os nomes e as pontuação dos melhores jogadores, como mostra a Figura 17, abaixo.

Figura 17. Tela de *ranking*.

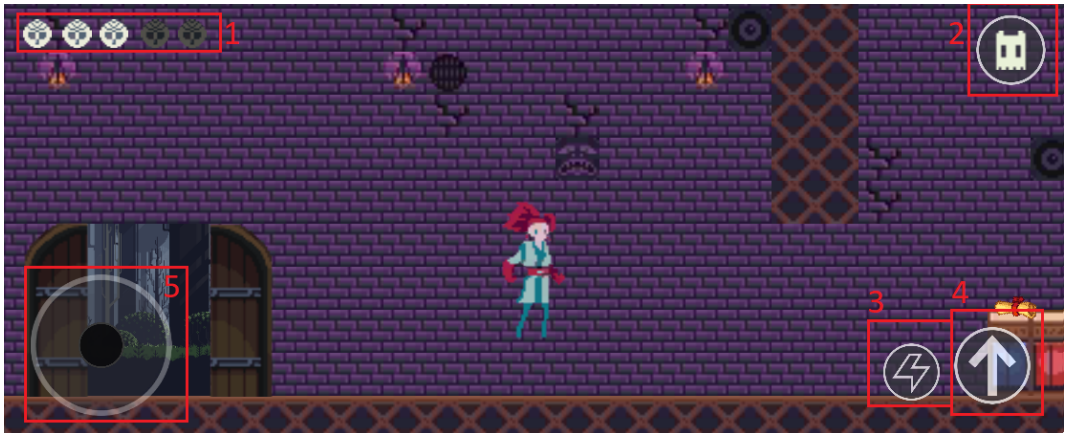
A tela de *ranking* é ativada após o toque no botão *ranking*, nela é listada os dez melhores jogadores, buscando em uma API Json que está na internet trazendo o nome e a pontuação dos jogadores, depois são ordenados de forma decrescente e, por fim, são geradas na tela. O código da Figura 18 insere em cada cartão da sua respectiva posição no *ranking*.

Figura 18. Código montagem do *ranking*.

```
1 reference
void montarRanking(List<Ranking> lista){
    List<Ranking> top10_lista = Ordenar(lista);
    for(int i=0; i<10;i++){
        if(i < top10_lista.Count){
            _top10[i].text = top10_lista[i].nome+"\nPontos: "+top10_lista[i].pontos;
        }else{
            _top10[i].transform.parent.gameObject.SetActive(false);
        }
    }
    _ranking.SetActive(true);
}
```

No primeiro módulo, a interface se ajusta novamente, ativando os controles como os botões, joystick e corujas que são os pontos de vida, ilustrado na figura 19 abaixo. Devido o primeiro módulo possuir apenas uma proposição ele foca mais nas mecânicas ensinando o jogador a como controlar a personagem e suas habilidades.

Figura 19. Componentes do HUD no módulo biblioteca.



Os controles do HUD mostra todas as mecânicas disponíveis naquele módulo, neste módulo há duas mecânicas primárias, sendo elas a de movimentar a Guerreira para os lados usando o *joystick* virtual, item 5 da figura 16, e de pulo através do botão 4, possui também a mecânica secundária representada por um raio o *dash,* o movimento realizado ao pressionar o botão 3. As corujas representam os pontos de vida da guerreira, que são verificados em cada momento através da função `update()`, como mostra a figura 20.

Figura 20. Controle das corujas na interface.

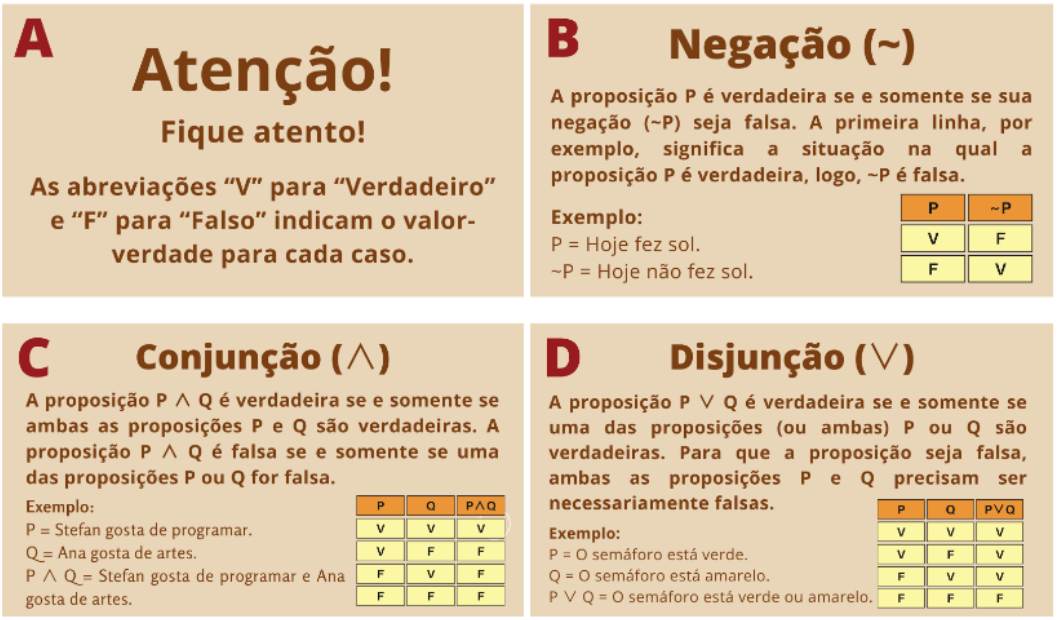
```
for(int x = 0; x < 5; x++){
    if(x<_vidas){
        _vidasHud.transform.GetChild(x).GetComponent<Image>().color = new Vector4(1f,1f,1f,1f);
    }else{
        _vidasHud.transform.GetChild(x).GetComponent<Image>().color = new Vector4(1f,1f,1f,0.5f);
    }
}
```

A cada *frame* do jogo é feita a verificação da quantidade de vidas da guerreira e monta o visual das corujas, devido as cores da *unity* poderem ser representadas como vetores de 4 pontos, é possível indicar o nível da opacidade para aquelas que estão em uma posição na lista maior que a quantidade de vidas da personagem. Outro elemento gráfico disponível no módulo um é o botão 2 da figura 16, representado pelo fantasma, que guia e dá dicas para a guerreira. Ao pressionar o botão, são exibidas as dicas coletadas pelo jogador, como mostra a figura 21.

Figura 21. Tela pergaminho de dicas.

Ao coletar as dicas espalhadas no cenário é possível visualizar na tela de pause, assim ao pressionar o botão do fantasma surge com um pergaminho na tela. Para apresentar as regras aplicadas a esse módulo, foram criadas dicas com as regras de inferência, sendo elas negação, conjunção e disjunção apresentadas de forma educativa como mostra a figura 22.

Figura 22. Todas as dicas da tela do pergaminho.



As dicas podem ser utilizadas no módulo dois que possui desafios mais complexos. Ao conseguir completar o módulo, o jogador avança no jogo, salva sua pontuação atual e, ao completar todos os desafios, restabelece a lógica no universo e atinge seu objetivo final. Este módulo possui características diferentes na interface, ao mudar a cena a interface esconde os botões de controle, pulo e *dash*, e ativa os botões de ataque V e F, aciona também um campo onde serão exibidas as proposições que serão os desafios que o jogador deverá passar, como é mostrado na figura 23.

Figura 23. Componentes do HUD no módulo dois.

Como ilustra a figura 23, a tela de componentes no módulo dois exibe apenas os elementos que são de combate. Os botões de ataque passam a ser visíveis para que seja possível responder o desafio com seus respectivos valores. As vidas continuam visíveis devido ao combate com as falaciosas. Caso consiga sobreviver até o fim do nível três

concluindo o módulo, o jogador vence revelando a tela de conclusão do jogo, como mostra a figura 24.

Figura 24. Tela de conclusão do jogo.

Ao completar todos os desafios, a tela de conclusão é carregada como mostra a figura 24, apresentando a pontuação final do jogador, o número de acertos e de erros. A tela permite que o jogador salve seus pontos e, caso esteja entre os dez melhores, seu perfil aparece na tela do *ranking*. Ao completar todas as etapas, o jogador ajuda os personagens a restabelecer a lógica nesse mundo.

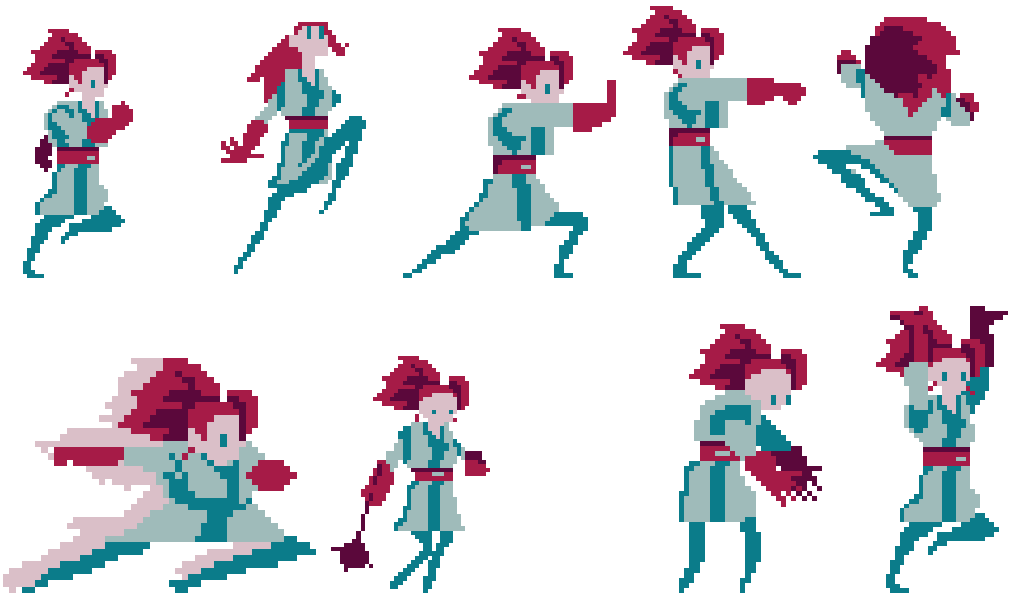
4.2.4. PERSONAGENS

Os personagens do jogo foram baseados na HQ, porém adaptados para o estilo de *pixel art*, que é comum em jogos de plataforma 2D. As mecânicas relacionadas à Guerreira e a Falaciosa foram implementadas utilizando colisões de objetos, um sistema oferecido pela *Unity* que é explicado a seguir.

4.2.4.1. Guerreira

Na *unity* todos os objetos do jogo são `GameObjects` e esses objetos possuem diferentes funções. A Guerreira possui os componentes `Animator`, `BoxCollider2D` e `RigidBody2D`, da própria *engine,* e o *Script* `Player`, que foi criado especificamente para a personagem. O `BoxCollider2D` foi utilizado para identificar colisões entre os objetos, `RigidBody2D` aplica física ao objeto e o `Animator` é responsável por controlar as animações da personagem, ilustrada na figura 25.

Figura 25. Ilustração das mecânicas da Guerreira.

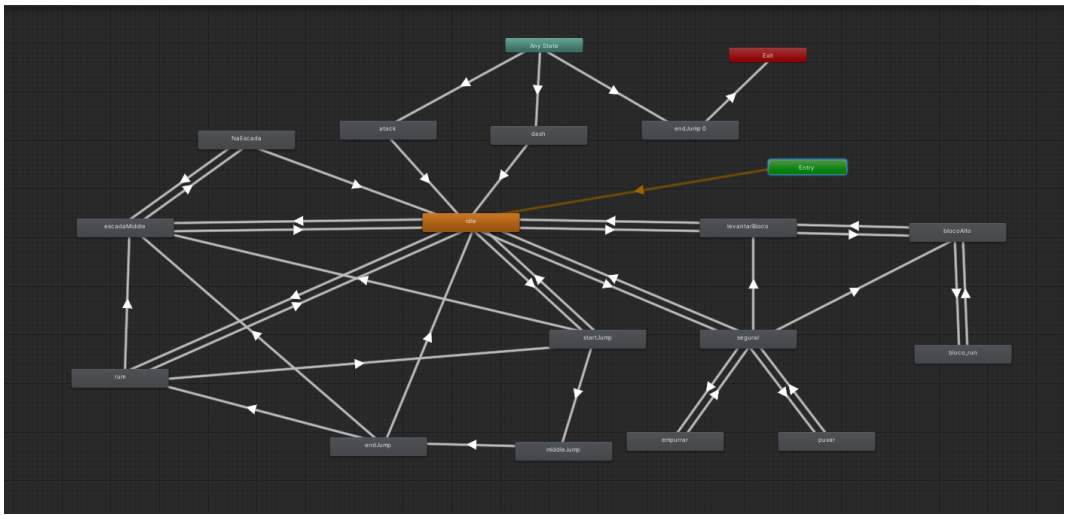


Fonte: Eler, 2022.

4.2.4.1.1. implementação das mecânicas primárias

As mecânicas utilizadas na criação da Guerreira são encontradas dentro do *Script Player*, que se utiliza dos componentes `Animator` e `RigididBody2D`, e são executadas na função *move* que é chamada a cada *frame* através da função *update* padrão da própria *Unity*.

Figura 26. Diagrama de animações da Guerreira



O diagrama apresentado na figura 26 mostra as relações e disposição entre as animações que são as representações visuais das mecânicas. Ao centro, tem-se a animação inicial e principal *idle* onde a guerreira fica parada e a partir dela com variáveis de controle do próprio componente `animator` da personagem. As setas representam a ligação e nelas estão as definições que fazem com que o `animator` identifique qual animação deve ser executada, alternando entre elas.

Andar ou Correr é a mecânica responsável por fazer a personagem correr ou andar pelos cenários, explorando e dando seguimento à trama do jogo.

Figura 27. Código da mecânica correr.

```
1 reference
void Move(){
    Vector2 movimento = _joystick.joystickInput;
    Vector3 direcao = new Vector3(0f,0f,0f);

 if(movimento.x > 0f && !_estaNaEscada){
     direcao.x = 1f;
     AnimControle.SetInteger("controle".1):
}else if(movimento.x < 0f && !_estaNaEscada){
    direcao.x = -1f;
    AnimControle.SetInteger("controle",1);
    transform.eulerAngles = new Vector3(0f,180f,0f);
transform.position += direcao * Time.deltaTime * velocidadeCorrida;
```

Na figura 27 é possível verificar que os movimentos do *joystick* virtual são captados através do atributo `JoystickInput`, assim é possível identificar para qual lado o jogador está se movimentando. Se este valor for maior que zero, o movimento é para a direita, se menor que zero, para a esquerda, com isso o componente transform do `GameObject` da Guerreira recebe a cada frame uma nova posição no atributo *position* multiplicado pela velocidade da personagem e pela escala de tempo.

Pular é a função de fazer com que a personagem chegue em um lugar mais alto, subindo em plataformas altas ou subindo em blocos.

Figura 28. Código da mecânica de pulo

```
1 reference
public void Pular(){
    if(!_segurando && _estaNoChao && !_estaNaEscada){
        AnimControle.SetInteger("controle",2);
        _estaNoChao=false;
        Rig.AddForce(new Vector2(0f,1f)*_forcaPulo, ForceMode2D.Impulse);
    }
}
```

A figura 28 mostra que ao pressionar o botão de pulo é feita uma verificação que analisa se a Guerreira não está segurando nenhum bloco ou escada e se está com os pés no chão. Se sim, ela poderá pular, portanto é indicado qual animação deve ser executada através da função `SetInteger()` com Componente `Animator`, em seguida utilizando o componente `Rigibody2D` na variável `Rig` aplica uma força com a função `AddForce()` que impulsiona o objeto da Guerreira para cima, controlando gravidade e peso automaticamente.

Empurrar ou Puxar são mecânicas que representam a grande força da Guerreira, podendo arrastar as caixas grandes que estão em seu caminho. Em alguns momentos ela deverá utilizar esta mecânica para conseguir progredir no jogo subindo em lugares mais altos.

Figura 29. Código da mecânica Empurrar e Puxar

```
public void ColisaoObjetos(Transform obj, int tipo){
    if(tipo == 1){
        if(_segurando && !_ocupada){
            Controller.GetComponent<bibliotecaControle>().setAcao("bloco");
            _btnAcao.SetActive(true);
            obj.SetParent(transform);
             ocupada = true;
void Move(){
  if(_segurando && _estaNoChao){
     if(transform.eulerAngles.y == 0f){
         AnimControle.SetInteger("controle",5);
     }else{
         AnimControle.SetInteger("controle",4);
     }
```

A mecânica se baseia na colisão com os blocos grandes. Ao utilizar a função `ColisaoObjetos()` é possível analisar o toque com outros objetos. Com a identificação de que são blocos grandes, o `GameObject` do bloco recebe o da Guerreira com a função *`SetParent()`*, tornando-o um objeto filho da personagem, o que permite que ele se mova na mesma direção que ela. Estas condições alteraram as propriedades da guerreira mudado sua animação.

Subir é a escalada em escadas para alcançar algumas plataformas que estão dispostas nos cenários, além de ter que utilizar esta mecânica para alcançar lugares que são essenciais para chegar ao módulo dois.

Figura 30. Código da mecânica Subir escada

```
if(_estaNaEscada){
    if(!AnimControle.GetBool("NaEscada")){
        AnimControle.SetBool("NaEscada",true);
    }
    if(movimento.y > 0){
        direcao.y = 1f;
        AnimControle.SetInteger("controle",7);
        transform.position += direcao * Time.deltaTime * velocidadeCorrida;
    }else if(movimento.y < 0){
        direcao.y = -1f;
        AnimControle.SetInteger("controle",7);
        transform.position += direcao * Time.deltaTime * velocidadeCorrida;
    }
    _checkBox.SetActive(false);
}else{
    _checkBox.SetActive(true);
}
```

A figura 30 mostra a mecânica de subida das escadas, que apenas verifica se a personagem colidiu com uma escada. Se sim, ele ativa a animação no componente `animator`. E assim como a de correr, ao mover o *joystick* virtual para cima ou para baixo é alterada a posição da personagem através do atributo *position* do *transform* da Guerreira, bloqueando outras mecânicas e ativando a variável `_estaNaEscada`.

Levar e Carregar Blocos são as mecânicas que consistem em colocar blocos menores espalhados pelo cenário em cima da cabeça da Guerreira, e carregá-los para uma nova posição. Essas mecânicas são necessárias para a conclusão do desafio do módulo um, que necessita que o jogador coloque um bloco específico no pedestal para completar uma sentença.

Figura 31. Código da mecânica Levantar e Carregar Blocos

```
if(child.gameObject.tag == "BlocoPequeno"){
    child.GetComponent<BoxCollider2D>().isTrigger = false;
    AnimControle.SetInteger("controle",15);
    if(_pedestal){
        Controller.GetComponent<bibliotecaControle>().Verificar(child.gameObject);
        _posicaoBloco.transform.position = child.transform.position;
        child.transform.position = _blocoPedestal.transform.position;
        child.transform.localScale = new Vector3(7.220971f,7.220971f,7.220971f);
    }else{
        Vector3 posTemp = child.transform.position;
        child.transform.position = _posicaoBloco.transform.position;
        _posicaoBloco.transform.position = posTemp;
        child.transform.GetComponent<BoxCollider2D>().isTrigger = false;
    }
}
```

A mecânica apresentada na figura 31 é chamada ao pressionar o botão de ação, se o bloco que está sendo carregado possuir a *tag* requerida, o bloco passa a ser um objeto filho, assim como os blocos grandes. Ao se tornar filho, o bloco troca de lugar com um Objeto temporário que apenas cede sua posição em cima da cabeça da personagem, ativando as variáveis de controle e mudando a animação da personagem. Quando a guerreira vai colocar o bloco no chão novamente é feita uma verificação se está próxima do pedestal, caso esteja, o bloco troca de lugar com um objeto do cenário, caso contrário o bloco deve ir para o chão trocando de lugar com o bloco temporário.

Atacar é a mecânica utilizada para derrotar os inimigos. A Guerreira deverá utilizar dois tipos de ataques, o que determina o tipo de ataque será a resposta dos desafios propostos no jogo.

Figura 32. Código da mecânica Ataque

```
    public void Ataque(string resposta){
        if(resposta == _sentencaAtual.resposta){
            if(_sentencaAtual.nivel == 3){
                pontos += 5;
            }else{
                pontos += _sentencaAtual.nivel + 1;
            }
            acertos++;
        }else{
            erros++;
            _guerreira.GetComponent<Player>().Dano(1);
        }
        if(usadas.Count != _listaSentencas.Count){
            indice = NovaSentenca();
            _sentencaAtual = _listaSentencas[indice];
        }else{
            SubirNivel();
        }
}
```

A figura 32 mostra a função de ataque da Guerreira, que recebe como parâmetro uma letra como resposta (V ou F). Se uma destas letras são recebidas, o sistema guarda a proposição que está sendo apresentada na interface e verifica se a resposta do jogador está correta. Caso a proposição esteja correta, o jogador recebe a quantidade de pontos pelo nível da sentença e aumenta o número de acertos. Caso o jogador erre, aumenta o número de erros e a Guerreira perde uma coruja de vida. Independente da resposta do jogador, uma nova sentença é gerada, buscando na lista e evitando que se repita.

4.2.4.1.2. implementação das mecânicas secundárias

Buscando trazer a forma lúdica da revista e visando facilitar o movimento da personagem no cenário, foi desenvolvido uma mecânica secundária. A mecânica não é obrigatória, permitindo que o jogador conclua o jogo sem usá-la. Porém a mecânica facilita a locomoção e até mesmo diminui o tempo em que o jogador poderia levar para completar o jogo.

O ***Dash*** é uma mecânica de investida curta para se movimentar mais rápido pelo jogo, não é uma mecânica obrigatória e pode ser usada em qualquer momento.

Figura 33. Código da mecânica *Dash*

```
0 references
public void Dash(){
    Vector2 direcao;
    if(_estaNoChao && !_estaNaEscada && _timeDash < 15){
        _timeDash ++;
        _segurando=false;
        if(transform.eulerAngles.y == 0f){
            direcao = new Vector2(1f,0f);
        }else{
            direcao = new Vector2(-1f,0f);
        }
        AnimControle.SetTrigger("dash");
        AnimControle.SetInteger("controle",0);
        Rig.AddForce( direcao * _dashForce,ForceMode2D.Impulse);
    }
}
```

Ao pressionar o botão de *dash,* a função da figura 33 é chamada, se a personagem estiver no chão e não estiver em uma escada, ela recebe uma força na direção em que estiver se movendo. Isso muda também a animação utilizando a função `SetTrigger()` do componente `Animator` que ativa a animação de mesmo nome da palavra passada como parâmetro.

4.2.4.1.3. Sistema de vidas

O jogo desenvolvido neste projeto possui um sistema de vidas inspirado no gênero RPG, a Guerreira possui cinco vidas durante o jogo, o jogador deve se preocupar em manter o maior número de vidas possíveis durante a jogabilidade, ou irá perder e terá que começar novamente. A vida da personagem é representada por ícones de corujas como mostra a figura abaixo.

Figura 34. Representação visual das vidas da Guerreira

A figura 34 acima revela as cinco vidas que a Guerreira possui durante o jogo, impedindo a conquista de vidas adicionais. O jogador deve acertar os desafios propostos no jogo (subseção 4.2.5), caso perca todas as vidas será apresentada a tela de *Game Over* com a

pontuação final que o jogador conquistou, sendo possível salvar ou reiniciar o jogo para tentar alcançar uma pontuação maior.

Figura 35. Código Controle de Vidas

```
1 reference
void ControleVidas(){
    if(_vidasHud != null){
        if(_vidas <= 0){
            Controller.GetComponent<Modulo2Controle>().GameOver();
        }else{
            for(int x = 0; x < 5; x++){
                if(x<_vidas){
                    _vidasHud.transform.GetChild(x).GetComponent<Image>().color = new Vector4(1f,1f,1f,1f);
                }else{
                    _vidasHud.transform.GetChild(x).GetComponent<Image>().color = new Vector4(1f,1f,1f,0.5f);
                }
            }
        }
    }
}
```

Como mostra a figura 35, a vida está dentro de uma variável e a cada frame, dentro da função `ControleVidas()`, é verificado se seu valor é maior que zero e, caso seja, é feita a exibição das corujas como descrito na seção 4.2.2. Caso seja igual a zero, é feita a chamada da função *`GameOver()`* no *script* do módulo dois, que trava a sequência do jogo e exibe a tela de fim de jogo, como mostra a figura 36 a seguir.

Figura 36. Tela Game Over

Ao zerar os pontos de vida da Guerreira, a tela de *game over* é revelada, onde o jogador pode salvar sua pontuação, assim como na tela de conclusão do jogo descrita na seção 4.2.2. O módulo dois permite que as personagens se enfrentem em uma batalha usando a lógica e, nesse contexto, o jogador ajuda a Guerreira a vencer. Caso o jogador erre todos os desafios, a vitória é de sua inimiga, a Falaciosa, que vencerá a batalha e impedirá que a lógica seja restabelecida no mundo.

4.2.3.2 Falaciosa

A Falaciosa é um personagem mais simples possuindo apenas três animações, e duas mecânicas controladas pelo mesmo *script*. A personagem é inimiga da Guerreira e deve correr e atacar quando estiver próxima de sua rival, assim como mostra a figura 37 abaixo.

Figura 37. Ilustração das Falaciosas

Fonte: Eler, 2022.

A Falaciosa é o inimigo do segundo módulo e para derrotá-la a Guerreira deverá acertar os desafios que aparecerem no jogo, pressionando os botões de verdadeiro ou falso para cada uma das proposições que surgirem. As mecânicas da personagem são controladas automaticamente como mostra a figura 38 a seguir.

Figura 38. *Script* da Falaciosa.

```
void Update()
{
    if(running){
        anim.SetInteger("controle",1);
        transform.position += new Vector3(-1f,0f,0f) * velocidade * Time.deltaTime;
    }else{
        anim.SetInteger("controle",0);
        transform.position += new Vector3(0f,0f,0f) * 0 * Time.deltaTime;
    }
}

0 references
void OnCollisionEnter2D(Collision2D obj){
    if(obj.gameObject.tag == "Player"){
        player = obj.transform;
        running = false;
        Invoke("Atacar",0.5f);
    }
}

0 references
void Atacar(){
    anim.SetTrigger("ataque");
    player.GetComponent<Rigidbody2D>().AddForce(new Vector2(-1f,0f)* 0.3f, ForceMode2D.Impulse);
    transform.parent.GetComponent<Modulo2Controle>().Ataquefalaciosa();
}
```

Ao ser iniciada no jogo, a personagem se move para a esquerda, como mostra a figura 38. Dentro da função `update()`, caso a variável `running` seja positiva, ela se move para esquerda adicionando a cada *frame* a multiplicação de um vetor com valor negativo no eixo x.

Caso seja negativo, o vetor é zerado fazendo com que a personagem fique parada, e as animações são controladas passando valores inteiros para o componente Animator.

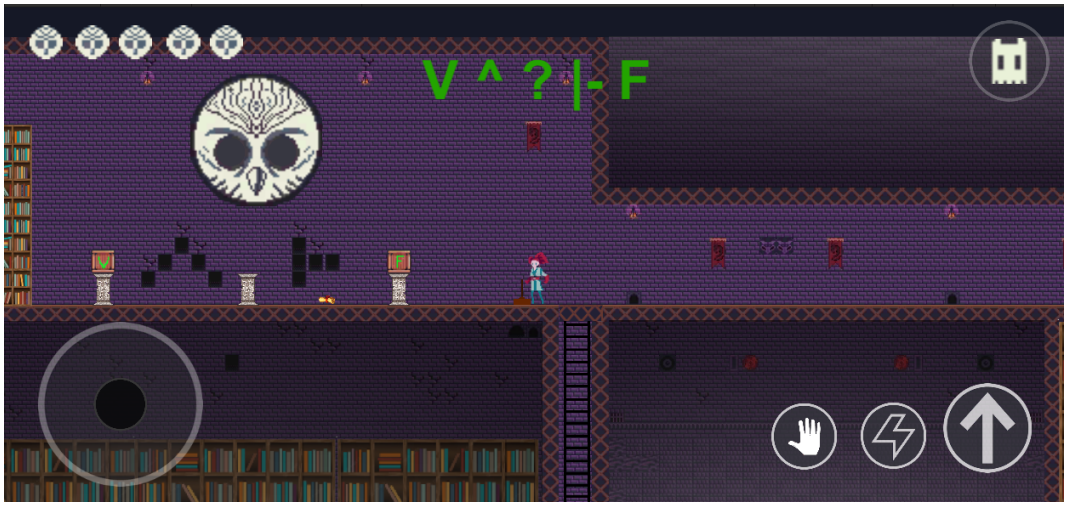
A função `OnCollisionEnter2D()`, herdada da classe `MonoBehaviour`, detecta automaticamente a colisão do `GameObject` com outros, e ao colidir com o `Player` a falaciosa para de correr e após 0.5 segundos executa a função de ataque no intervalo do tempo que a Guerreira tem para contra atacar. Para atacar, o componente `Animator` recebe um gatilho para o ataque mudando a animação e é então aplicado um impulso representando a força do ataque na Guerreira, e por fim aplica um dano diminuindo a vida da Guerreira.

4.2.5 Desafios e Pontuação

O Logi Kingdom possui desafios baseados nas regras de inferência, utilizando as sentenças básicas. No primeiro módulo o desafio é completar uma sentença, onde a Guerreira deve utilizar sua mecânica de carregar blocos para colocar o bloco com símbolo correto no pedestal. No segundo módulo, o desafio fica mais complexo, pois a Guerreira enfrenta a Falaciosa, que traz como desafio uma sentença formada pelos operadores inicialmente a negação, conjunção e disjunção.

Quando a Falácia é derrotada, esta retorna com uma sentença nova e mais difícil. Porém quando a Guerreira erra um desafio ela perde pontos de vida, sendo nocauteada pela Falaciosa. Já no primeiro módulo, os olhos da Coruja no cenário piscam vermelho indicando que o bloco colocado está incorreto.

Figura 39. Desafio da biblioteca.



O primeiro desafio apresentado na figura 39 não possui pontuação, porém se faz necessário para que o jogador consiga avançar para o segundo módulo, além de servir como um pequeno desafio para que o jogador entenda como o jogo funciona. A implementação do

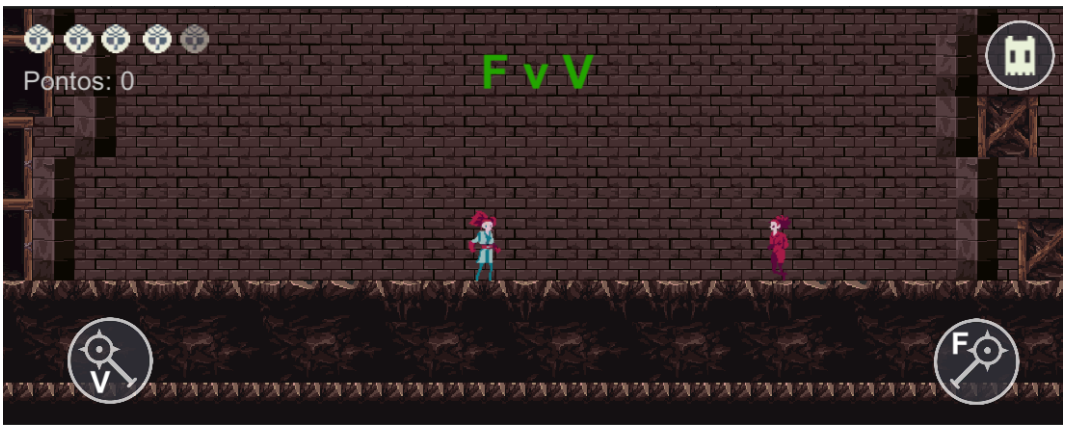
desafio foi feita através de um identificador no bloco, caso seja o bloco correto efetua o código abaixo na figura 40.

Figura 40. Código do desafio da biblioteca.

```
1 reference
public void Verificar(GameObject bloco){
    if(bloco == _BlocoPequenoV){
        _corujaEfigy.GetComponent<Animator>().SetTrigger("certo");
        Invoke("Transicao",2f);
    }else{
        _corujaEfigy.GetComponent<Animator>().SetTrigger("errado");
    }
}
```

Após pressionar o botão de ação, colocando o bloco no pedestal a função verificar é chamada, ela identifica se o bloco é igual ao que já está determinado como correto. Caso o jogador acerte, o sistema ativa a animação do objeto coruja *Effigy* que fica na parede atrás dos pedestais, logo após a Guerreira é levada para a toca da coruja. No módulo dois, a toca da coruja, o jogo fica mais complexo, o jogador deve solucionar a proposição que aparece na tela antes que a falaciosa o alcance e o ataque, como mostra a figura 41.

Figura 41. Exemplo de desafio do módulo dois.



Todo o controle de pontos é definido a partir do nível que o jogador está. Como foi definido no *game design* que cada sentença do nível um vale dois pontos, do nível dois três pontos e do nível três vale cinco pontos. Ao acertar um desafio, o código de controle aumenta os pontos do jogador e busca uma nova sentença, repetindo o processo até o fim do nível, concluindo assim o jogo, como mostra a figura 42 abaixo.

Figura 42. Código do desafio do módulo dois.

```
if(resposta == _sentencaAtual.resposta){
    if(_sentencaAtual.nivel == 3){
        pontos += 5;
    }else{
        pontos += _sentencaAtual.nivel + 1;
    }
    acertos++;
}else{
    erros++;
    _guerreira.GetComponent<Player>().Dano(1);
}
```

O sistema que controla a pontuação foi desenvolvido através da função `Ataque()`, que ao acertar o sistema identifica através do atributo nível da sentença atual do desafio. Com este valor é possível acrescentar a pontuação equivalente ao nível para a Guerreira. Para o nível três é acrescentado cinco pontos, caso seja diferente os pontos recebem o nível mais um, chegando ao valor definido no *Game Design Document*.

Concluindo o sistema de pontos, o desenvolvimento do jogo foi finalizado. Seguindo para o processo de *build* que é o processo de compilação do projeto, utilizando a compilação de lançamento, que inclui apenas o necessário para executar o aplicativo. Com este processo finalizado é gerado um arquivo no formato APK, sendo assim possível teste e validação da especialista, obtendo ótimos resultados.

5 CONSIDERAÇÕES FINAIS

Este trabalho teve como objetivo desenvolver as mecânicas e dinâmicas de um jogo para dispositivos móveis que possa auxiliar os alunos com o entendimento dos conteúdos apresentados na sala de aula e exercitando os mesmos com vários desafios lógicos. Seguindo a estrutura apresentada na seção 3.2 da metodologia foi possível alcançar os resultados esperados, e se utilizando dos elementos da revista em quadrinhos, obteve-se um jogo divertido e que apresenta conceitos de lógica em sua dinâmica. Através dos trabalhos relacionados, com desenvolvimento de jogos para o ensino, foi possível ter um entendimento da estrutura de interface que pode motivar e facilitar a aprendizagem dos alunos.

Durante a implementação foi encontrado dificuldades ao detectar as colisões onde a guerreira precisava empurrar os blocos, onde foi necessário criar um novo objeto dentro da personagem para executar a ação. Usando a função `OnCollisionEnter2D` para detectar as colisões ativa um gatilho, mudando os estados do bloco. No processo de desenvolvimento houve alguns desafios, como a configuração de uma interface que funcionasse em mais de uma cena diferente, mecânicas que não funcionavam como esperado.

Buscando soluções que deixassem o jogo mais bonito, em fóruns foi encontrado a utilização da interface como *prefab* deixando-a disponível em qualquer cena do jogo, e para que os botões funcionasse foi necessário gerar um *Event System,* objeto com configuração da própria *Unity*. Para a solução das mecânicas como a de levantar os blocos, foi utilizado um outro *GameObject* porém sem nenhum componente, além do *Transform* que é também é padrão da *Unity.* Com este objeto foi possível definir uma posição para que o bloco ficasse ao ser carregado, trocando os valores da propriedade de *posição* do *transform.*

O jogo foi assim então disponibilizado para o público de dispositivos móveis com sistema operacional *Android*, o *download* pode ser feito no site https://logikingdom.herokuapp.com/. E para os trabalhos futuros, seria interessante produzir um sistema que a partir de uma banco de questões fosse alterado o desafio do módulo um sempre que o jogador entrasse nessa fase, para que o desafio da biblioteca seja dinâmico assim como no módulo dois. Pensando ainda em melhorias futuras do jogo, os personagens devem possuir mais objetos guia para uma melhor colisão com outros, com objetivo de melhorar os movimentos da personagem.

Como trabalhos futuros, pretende-se implementar um *dashboard* adicionando informações úteis como desafios em que errou e acertou, quantidade de tentativas, além de toda a questão da pontuação que já é salva no *ranking.* Com estes dados é possível o desenvolvimento de um painel que verifica o desempenho do jogador e repassa para ele um

feedback dos erros e acertos, destacando as regras ele possui mais dificuldade e apresentando dicas de como melhorar seu desempenho. Além da criação de novos módulos utilizando as demais regras e operadores lógicos, e até mesmo criando mais níveis para o módulo dois, criando novas listas. Portanto, o jogo não ficará limitado ao ensino da negação, disjunção ou conjunção, trabalhando em novos conteúdos da lógica e evoluindo o jogo se tornando um grande produto.

REFERÊNCIAS

AGUIAR, Leandro Henrique Freitas de. **Uma Ferramenta para Definição de Níveis para Jogos 2D por meio de Curvas de Terreno no Unity**. 2017. 42 f. TCC (Graduação) - Curso de Ciência da Computação, Centro de Informática, Universidade Federal de Pernambuco, Recife, 2017.

AZEVEDO, Janaina Leite de. **"Na trilha de Macunaíma", game didático GDD & roteiro interativo de jogo digital para literatura no ensino médio**. 2017. 301 f. Dissertação (Mestrado) - Curso de Mídia e Tecnologia, Arquitetura, Artes e Comunicação, Universidade Estadual Paulista, Bauru, 2017. Disponível em: http://hdl.handle.net/11449/152646. Acesso em: 01 jul. 2022.

BEDREGAL, Benjamin René Callejas; ACIÓLY, Benedito Melo. **Introdução à Lógica Clássica para a Ciência da Computação**. Natal, Bra: [s.n.], 2007. v. 3. Disponível em: <http://www.dimap.ufrn.br/~jmarcos/books/BA_Jul07.pdf>. Acesso em: 07 mar. 2022.

BYL, Penny de. **Holistic Game Development with Unity**: an all-in-one guide to implementing game mechanics, art, design and programming. 2. ed. New York: Crc Press, 2017. 469 p.

CHANDLER, Heather M. **Manual de produção de jogos digitais**. Bookman Editora, 2009.

COPI, Irving M. **Introdução a lógica**. 2. ed. São Paulo: Mestre Jou, 1978.

CUNHA, Marisa.Ortegoza. D.; MACHADO, Nílson. J. **Lógica e linguagem cotidiana**. Belo Horizonte: Grupo Autêntica, 2019.

FALKEMBACH, Gilse A. Morgental. O lúdico e os jogos educacionais. **CINTED-Centro Interdisciplinar de Novas Tecnologias na Educação**, UFRGS, 2006.

FRANCO, Magda Aparecida de Oliveira; ZAMPIERI, Margarete Fátima de Oliveira; MACIEL, Reive Guedes; SILVA, Charles René Sousa; OLIVEIRA, Lucimara de. Jogos como ferramenta para favorecer a aprendizagem. In: CONGRESSO NACIONAL DE EDUCAÇÃO, 5., 2018, Olinda. **Anais Eletrônicos**. Olinda: CONEDU, 2018. p. 1-13.

FREYERMUTH, Gundolf S. **Games | Game Design | Game Studies**: an introduction. 20. ed. [S. L.]: Transcript Verlag, 2015. 296 p.

GAMUX. **Como organizar o desenvolvimento do seu jogo de forma eficiente com um Game Design Document**. 2011. Traduzido por Jonathan Ramos. Disponível em: https://code.tutsplus.com/pt/articles/effectively-organize-your-games-development-with-a-game-design-document--active-10140. Acesso em: 04 jul. 2022.

GOMES, Fernanda Pereira. **Desenvolvimento de um framework para gamificação baseada na tríplice contingência e aplicação no módulo de tabela verdade do Logic Live**. 2019. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação). Centro Universitário Luterano de Palmas, Palmas, Tocantins, 2019. Disponível em: <http://ulbra-to.br/bibliotecadigital/publico/home/documento/689>. Acesso em: 16 set. 2021.

GRÜBEL, Joceline Mausolff; BEZ, Marta Rosecler. Jogos Educativos. **Renote**, [S.L.], v. 4, n. 2, p. 1-7, 22 dez. 2006. Universidade Federal do Rio Grande do Sul. http://dx.doi.org/10.22456/1679-1916.14270.

HAGUENAUER, Cristina Jasbinscheck; CARVALHO, Fabricia Silva de; VICTORINO, Ana Lúcia Quental; LOPES, Marise Castello Branco Altro; CORDEIRO FILHO, Francisco.. Uso de jogos na educação online: a experiência do LATEC/UFRJ. **Revista Educação On Line**, Rio de Janeiro, v. 1, n. 1, jan./abr. 2007. Disponível em: http://www.latec.ufrj.br/revistaeducaonline/vol1_1/2_jogos.pdf. Acesso em: 13 out. 2021.

JÄRVINEN, Aki. **Games without Frontiers: theories and methods for game studies and design**. 2007. 416 f. Tese (Doutorado em estudos de Media Culture). University of Tampere, Finland, 2007.

KICKMEIER-RUST, Michael D. et al. Immersive digital games: the interfaces for next-generation e-learning?. In: **International Conference on Universal Access in Human-Computer Interaction**. Springer, Berlin, Heidelberg, 2007. p. 647-656.

KREIMEIER, Bernd. **Game Design Methods: A 2003 Survey**. 2003. Game Developer. Disponível em: https://www.gamasutra.com/view/feature/2892/game_design_methods_a_2003_survey.php. Acesso em: 01 jul. 2022.

LEITE, Patricia da Silva; MENDONÇA, Vinícius Godoy de. Diretrizes para Game Design de Jogos Educacionais. In: Simpósio Brasileiro de Jogos e Entretenimento Digital, 12., 2013, São Paulo. **Anais do XII SBGames**. São Paulo: Sbgames, 2013. p. 132-141. Disponível em: https://www.sbgames.org/sbgames2013/proceedings/artedesign/17-dt-paper.pdf. Acesso em: 04 jul. 2022.

LIMA, Arthur Siqueira de. **Modelo de desenvolvimento de jogos com propósito baseado em mecânicas tradicionais de jogo**s. 2013. 105 f. Dissertação (Mestrado) - Curso de Engenharia de Sistemas e Computação, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2013.

MARQUES, Natália; SILVA, Bento. Cenários de aprendizagem com recurso à ferramenta The Sims Carnival game creator. In: VI CONFERêNCIA INTERNACIONAL DE TIC NA EDUCAÇÃO, 6., 2009, Porto. **Actas do IX Colóquio Sobre Questões Curriculares**. Porto: Faculdade de Psicologia e Ciências da Educação da Universidade do Porto, 2009. p. 1357-1365.

MICROSOFT. **Introdução à linguagem C# e ao Framework .NET**. 2010. Disponível em: https://docs.microsoft.com/pt-br/previous-versions/visualstudio/visual-studio-2008/z1zx9t92(v=vs.90)?redirectedfrom=MSDN. Acesso em: 17 nov. 2021.

MICROSOFT. **Um tour pela linguagem C#**. 2021. Disponível em: https://docs.microsoft.com/pt-br/dotnet/csharp/tour-of-csharp/. Acesso em: 17 nov. 2021.

MIOTO, Jessica A.; BATISTA, Esteic Janaina Santos; SANTOS, Quésia de Araújo; BOGARIM, Cintia A. Canteiro; LIMA, Anderson Corrêa de. As Aventuras de Calculino: jogo para ensino de raciocínio lógico. In: Congresso Brasileiro de Informática na Educação, 1.,

2017, [S. L.]. **Anais dos Workshops do VI Congresso Brasileiro de Informática na Educação**. [S. L.]: CBIE, 2017. p. 451-455.

MORENO, Douglas Aquino; SILVA, Stefan Lucas Aquino; SANTOS, Natanna Rocha; SOUZA, Calebe Loures Sampaio Eler de; BRITO, Parcilene Fernandes de. Logi Kingdom: Revista em Quadrinho como Tema para um Ambiente de Aprendizagem de Lógica. In: Jornada de Iniciação Científica, 21., 2021, Palmas. **Anais [...]** . Palmas: Ceulp/Ulbra, 2021. p. 80-86.

MÖRSHBÄCHER, Lucas Gelásio. **Esfinge caduca:** criação de jogo físico de estratégia. 2015. 141 f. TCC (Graduação) - Curso de Arquitetura, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2015.

MURCIA, Juan Antonio M. **Aprendizagem através do jogo**. Artmed Editora, 2005.

MUSBURGER, Robert B.. **Roteiro Para Mídia Eletrônica**. Rio de Janeiro: Elsevier, 2008. 300 p. Tradução de Natelie Gerhardt.

NEWZOO. **Global Games Market Report**. Disponível em: https://newzoo.com/products/reports/global-games-market-report/. Acesso em: 04 jul. 2022.

NICOLETTI, Maria do Carmo. **A cartilha da lógica**. Rio de Janeiro, RJ: LTC, 2017. 3ª Edição.

NOLT, John; ROHATYN, Dennis. **Lógica**. São Paulo: Makron Books, 1991.

OLIVEIRA, Arthur V.; RIBEIRO, Felipe S.; SCHALSK, Luana; SILVA, Júlio César F.; CAMARGO, Jhonatan O.; FARION, Victor; SILVA, Cassiana F. da; MUELLER, Rafael R.. MazeLogic: jogo educacional para ensino de lógica de programação. In: CONGRESSO SUL BRASILEIRO DE COMPUTAÇÃO, 9., 2018, Santa Catarina. **Anais SULCOMP**. Santa Catarina: Sulcomp, 2018. v. 9, p. 1-4. Disponível em: https://periodicos.unesc.net/ojs/index.php/sulcomp/article/view/4779. Acesso em: 04 jul. 2022.

PHOTOSHOP. **Guia do Usuário do Photoshop**. 2021. Disponível em: https://helpx.adobe.com/br/photoshop/user-guide.html. Acesso em: 10 nov. 2021.

P. Schuytema. **Design de games**: uma abordagem prática. São Paulo: Cengage Learning, 2008.

ROLAND, Letícia Coelho; FABRE, Marie-Christine Julie Mascarenhas; KONRATH,Mary Lúcia Pedroso; TAROUCO, Liane Margarida Rockenbach. Jogos educacionais. **RENOTE**, Porto Alegre, v. 2, n. 1, 2004. DOI: 10.22456/1679-1916.13719. Disponível em: https://seer.ufrgs.br/index.php/renote/article/view/13719. Acesso em: 4 jul. 2022.

SALEN, Katie; ZIMMERMAN, Eric. **Regras do jogo**: fundamentos do design de jogos. 4. ed. São Paulo: Editora Blucher, 2012. 155 p.

SANTANA, Camila; SENA, Gildeon; MOURA, Juliana de; ALVES, Lynn. Tríade: delineando o processo de construção de um roteiro de um. In: Simpósio Brasileiro de Jogos e Entretenimento Digital, 7., 2006, São Leopoldo. **Anais do VI SBGames**. São Leopoldo: Sbgames, 2006. v. 7, p. 1-8. Disponível em: http://www.sbgames.org/papers/sbgames07/gameandculture/full/gc6.pdf.. Acesso em: 04 jul. 2022.

SANTOS, Wilk Oliveira dos; ROSALINO, Gustavo; SILVA, Célia. Desafios das Diagonais: Um Jogo Casual para o Aprimoramento do Raciocínio Lógico. **Anais dos Workshops do Congresso Brasileiro de Informática na Educação**, [S.l.], p. 168, out. 2017. ISSN 2316-8889. Disponível em: <http://ojs.sector3.com.br/index.php/wcbie/article/view/7388/5184>. Acesso em: 04 jul. 2022. doi:http://dx.doi.org/10.5753/cbie.wcbie.2017.168.

SCOLARI, Angélica Taschetto; BERNARDI, Giliane; CORDENONSI, Andre Zanki. **O Desenvolvimento do Raciocínio Lógico através de Objetos de Aprendizagem**. **Renote**: Revista Novas Tecnologias na Educação, Porto Alegre, v. 5, n. 2, p. 1-10, dez. 2007. Disponível em: <https://seer.ufrgs.br/renote/article/view/14253/8169>. Acesso em: 04 dez. 2021.

SCHELL, Jesse. **The Art of Game Design**: A book of lenses. Morgan Kaufmann. 2019.

SICART, Miguel. Defining game mechanics. **Game Studies**, v. 8, n. 2, p. 1-14, 2008.

SILVA, Firmiano Alexandre Reis; RUFINO, Hugo Leonardo pereira. Cartas à Mesa: Uma Proposta Lúdico-Didática para o Ensino de Lógica. **Revista Inova Ciência & Tecnologia / Innovative Science & Technology Journal**, [S. l.], v. 6, n. 2, p. 43–51, 2020. Disponível em: https://periodicos.iftm.edu.br/index.php/inova/article/view/1043. Acesso em: 4 jul. 2022.

SOUZA, Juliana Marques Paiva de; SALVADOR, Marco Antonio Santoro. **O Lúdico e as Metodologias Ativas**: possibilidades e limites nas ações pedagógicas. Rio de Janeiro: Imperial, 2019. 43 p.

SOUZA, Susan Martins de. **O uso de jogos eletrônicos como entretenimento para crianças e adolescentes na sociedade pós-moderna**. 2020. 74 f. TCC (Graduação) - Curso de Serviço Social, Universidade Federal Rural do Rio de Janeiro, Seropédica, 2020.

S. Lucena. **Cultura digital, jogos eletrônicos e educação**. In:Repositório Institucional da Universidade federal da Bahia, Salvador – BA, 2014, 1º ed, 246p.

UNITY. **Dê as boas-vindas à plataforma Serviços de jogos da Unity**. 2021. Disponível em: https://unity.com/pt. Acesso em: 17 nov. 2021.

VELASCO, Patrícia Del Nero. **Educando para a argumentação:** Contribuições do ensino da lógica. Belo Horizonte: Grupo Autêntica, 2010..

WERBACH, K.; HUNTER, D. **For the win:** how game thinking can revolutionize your business. Philadelphia: Wharton Digital Press, 2012.

ZAFFARI, G.; BATTAIOLA, A. L. Princípios para o Design de Jogos Digitais com base em Erro Humano. **InfoDesign - Revista Brasileira de Design da Informação**, [S. l.], v. 12, n. 3, p. 267–301, 2015. DOI: 10.51358/id.v12i3.382. Disponível em: https://infodesign.emnuvens.com.br/infodesign/article/view/382. Acesso em: 24 out. 2022.

ZICHERMANN, Gabe; CUNNINGHAM, Christopher. **Gamification by Design**: implementing game mechanics in web and mobile apps. Canada: O'Reilly Media, Inc, 2011. 208 p.

APÊNDICES

LOGI KINGDOM

Game Design Document

Versão: 1.0

Autores:

Calebe Eler

Douglas Aquino Moreno

Parcilene Fernandes de Brito

Stefan Lucas Aquino Silva

Palmas Tocantins, 2022

Índice

1. História

O jogo Logi Kingdom se passa em um universo similar ao nosso planeta Terra, mas bem menor e localizado em uma outra galáxia, neste planeta vivem personagens que estão em uma guerra diária para trazer a lógica ao seu mundo. Há algum tempo que a escuridão cobriu parte do céu do pequeno planeta, a vegetação se desenvolve com dificuldade, e todas as árvores têm baixa estatura e são cobertas por uma espécie de lodo infinito. É nesse ambiente devastado pela ausência da lógica que uma Guerreira e um Fantasma buscam, através do uso das regras de inferência do cálculo proposicional, trazer vida e lógica para o planeta. Nesse contexto inóspito, eles vivem uma série de aventuras, explorando bibliotecas e lutando contra Falaciosas (inimigos) em busca da lógica perdida.

2. Gameplay

A história do jogo se passa em um mundo diferente do nosso, com a ausência da lógica a Personagem Guerreira terá que explorar esse mundo e encontrar dicas de lógica proposicional para resolver desafios e derrotar inimigos falaciosos. As dicas do jogo vão estar espalhadas pelo ambiente e a Guerreira deverá coletá-las, com a ajuda do Fantasma ela poderá revisar essas dicas quantas vezes forem necessárias para conseguir derrotar seus inimigos e prosseguir no jogo. Para derrotar as falaciosas a Guerreira deverá usar um ataque para cada uma das proposições, caso a resposta do desafio seja verdadeiro ela terá que clicar no ataque correspondente a letra “V”, caso contrário deverá clicar no botão “F” que corresponde a Falso.

A experiência do usuário é controlada pelas seguintes mecânicas:

- **Andar:** Ao mover o controle para esquerda ou para a direita o jogador faz com que a guerreia seja deslocada na direção desejada.
- **Pular:** movimento realizado ao clicar no botão de pulo, onde a guerreira é impulsionada para cima podendo pular de objetos e caixas que estarão espalhadas no ambiente, além disso o usuário deverá utilizar esta mecanica para alcançar as plataformas que estarão no jogo e chegar no estado final desejado.
- **Empurrar ou puxar:** ao encostar em blocos grandes será possível empurrá-los ou puxá-los pressionando o botão de ação a partir daí ao mover a guerreira o bloco se move junto.
- **Levantar:** consiste em colocar blocos menores espalhados pelo cenário em cima de sua cabeça, e carregá-los para uma nova posição.
- **Dash:** impulso rápido, para frente, podendo ser executado pela Guerreira enquanto estiver no chão.
- **Atacar**: para derrotar as Falaciosas a Guerreira deverá utilizar dois tipos de ataques, o que determinará o tipo de ataque que ela poderá utilizar será a solução de desafios, por exemplo, caso o resultado do desafio seja falso a Guerreira deverá atacar a Falaciosa com o botão de ataque representado pelo botão “F”, caso contrário deverá atacar com o clicando no botão “V”.

No primeiro módulo a Guerreira deve solucionar uma proposição lógica, e para isso deverá passar por desafios menores, como pular blocos, subir escada, alcançar plataformas

móveis e por fim colocar o bloco específico que resolve a proposição. Já no segundo módulo a Guerreira deve enfrentar as Falaciosas que surgem a cada desafio mostrado na tela, a Guerreira deverá atacar a Falaciosa com o botão de ataque representado por “V” caso a resposta da proposição seja verdadeira ou “F” caso seja falsa. Caso o jogador erre o desafio, a Falaciosa consegue acertar um golpe fazendo com que a Guerreira perca uma coruja de seus pontos de vida podendo perder o jogo caso perca todas as cinco corujas.

No primeiro módulo o jogador avança assim que aprende a controlar a Guerreira utilizando as mecânicas resolvendo o desafio. No segundo os desafios possuem níveis ao conseguir acertar os desafios do nível um passa para o nível dois e assim por diante, e ao finalizar todos avança para um próximo módulo, salvando sua pontuação atual, ao finalizar todos os módulos ou seja completar todos os desafios a Guerreira restabelece a lógica no universo atingindo seu objetivo final.

2.1. Tabela de Pontuação

O jogador ganha certa quantidade de pontos a cada desafio que acertar, com o passar dos níveis essa pontuação será maior e consequentemente quanto maior o nível maior será a pontuação e maior será a complexidade das proposições lógicas. A descrição da pontuação de acordo com o nível é apresentada na tabela a seguir.

Nível	Pontuação	Tempo do Nível	Tempo de entrada das Proposições
Nível 1	2 pontos	1 minuto	1 segundos
Nível 2	3 pontos	2 minutos	2 segundos
Nível 3	5 pontos	3 minutos	2 segundos

2.2. Pontuação e Erros

A seguir é apresentado o cálculo da pontuação final que o jogador poderá conquistar ao concluir qualquer um dos níveis.

$$Pf = acertos - (acertos\ x\ (\frac{erros}{100}))$$

Ao final de cada nível ou quando o jogador perder todas as corujas (vidas) é apresentado a pontuação final que ele conquistou. Os pontos são calculados pela quantidade de acertos menos o percentual de erros.

3. Personagens

3.1 Guerreira

A Guerreira é a personagem principal do jogo, uma jovem destemida que busca a qualquer custo devolver a lógica para seu mundo. Através do conhecimento dos conceitos da lógica que irá adquirir ao longo do jogo ela poderá conquistar aos poucos o seu objetivo, derrotando inimigos e explorando adquirindo conhecimento através das dicas sobre as regras da lógica.

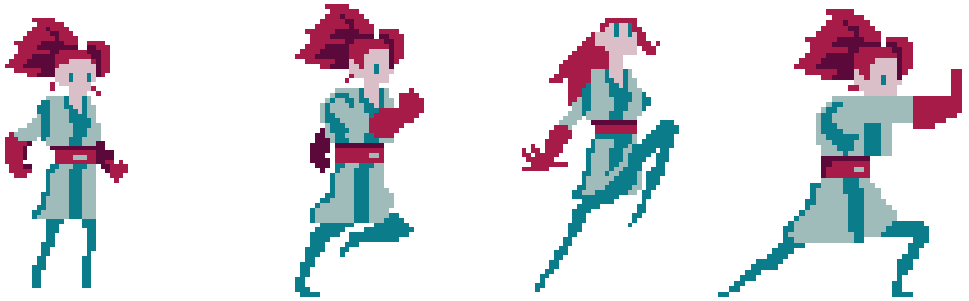


Figura 1. Ações e movimentos que a Guerreira pode executar.

Ficha técnica da personagem:

- **Nome da personagem**: Guerreira.
- **Idade**: entre 20 e 25 anos.
- **Tipo**: personagem principal do jogo.
- **Personalidade**: dinâmica, rápida, corajosa e impulsiva.
- **Habilidades**: a Guerreira possui uma clava que usa como arma para derrotar seus inimigos.
- **História**: uma jovem que perdeu a mãe muito cedo e teve que aprender a enfrentar os desafios da vida sozinha. Durante sua infância, teve um melhor amigo, mas acabou perdendo-o assim como perdeu sua mãe. Como uma Guerreira com um instinto impulsivo e corajoso, muitas vezes enfrenta desafios que a maioria das pessoas não conseguiria resolver e, um de seus desafios é salvar a lógica do mundo em que o jogo se baseia.
- **Ações:** andar, pular, correr, subir escadas, levantar objetos, empurrar objetos, puxar objetos e atacar inimigos.
- **Métricas de gameplay**:

3.2 Fantasma

O fantasma é um personagem secundário que estará relacionado com as dicas do jogo, sobre os conceitos da lógica e das regras de inferência do cálculo proposicional. Para chamar o Fantasma o jogo deverá ter um botão para que o jogador chame ele. O personagem apresenta dicas sobre as regras da lógica em um tipo de inventário de regras, para isso o jogador deverá ter coletado os pergaminhos que contêm as dicas e que estão espalhados pelos ambientes do primeiro módulo do jogo - Biblioteca.

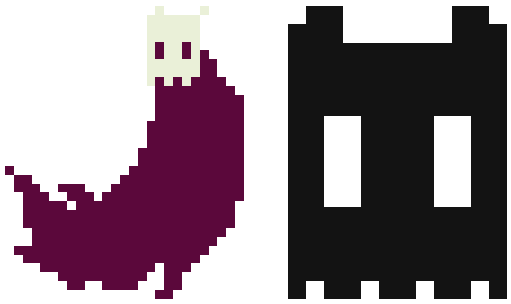
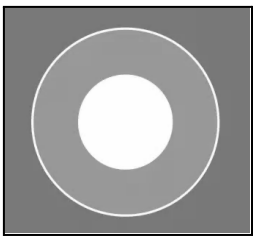


Figura 2. Personagem - Fantasma.

Ficha técnica do personagem:

- **Nome do personagem**: Fantasma.
- **Idade**: indeterminada.
- **Tipo**: personagem secundário do jogo.
- **Personalidade**: sistemático, organizado, previsível e misterioso
- **Habilidades**: conhecimento sobre as dez regras de inferência do cálculo proposicional.
- **Ações**: personagem que auxilia a Guerreira a enfrentar desafios e entender a lógica proposicional.
- **Métricas de gameplay**:

4. Controles



O jogador controla a guerreira utilizando um *joystick* virtual, com botões e uma alavanca para o controle direcional, estes controles fazem parte da HUD (*heads-up display*).

Alavanca - São dois Círculos um maior que determina o tamanho máximo de onde a alavanca vai chegar o segundo círculo bem menor que fica ao centro do maior, o jogador interage movimentando com o *touch* este menor ao mover para esquerda, move a guerreira para a esquerda e da mesma forma para a direita, como mostra a figura ao lado.

4.1 Botões

ÍCONE	AÇÃO AO PRESSIONAR O BOTÃO
	● **Botão pulo** - Ao tocar uma vez no botão “pulo” a Guerreira dará um leve pulo.
	● **Botão ação** - Ao ser pressionado executa a ação naquele instante, podendo ativar algum objeto como alavanca, ou fazer com que a guerreira consiga empurrar ou puxar os blocos.
	● **Botão dash** - Quando pressionado faz com que a guerreira tenha um impulso para a frente.
	● **Botão fantasma** - Ao tocar revela o painel de dicas, exibindo dicas que foram coletadas.
	● **Botão V** - Executa o ataque da guerreira, caso a resposta do desafio seja verdadeira.
	● **Botão F** - Executa o ataque da guerreira, caso a resposta do desafio seja verdadeira.

5. Câmera

Em cada módulo a câmera se comporta de uma forma, no primeiro dentro da biblioteca ela é dinâmica, acompanhando o movimento da Guerreira controlada pelo jogador, quando o movimento do personagem se aproxima dos limites ou bordas da tela, o cenário começa a se destacar, permitindo que o jogador se movimente em cenários maiores. Tradicionalmente, o jogador tem uma visão lateral que é conhecida como *side view*, que é comum em jogos 2D de plataforma, Figura x, a seguir.

Figura 3. Representação do primeiro módulo.

A Figura acima apresenta o esboço do primeiro módulo utilizando a ideia de hierarquia. A câmera irá seguir a personagem onde ela for, permitindo que haja uma centralização do objeto selecionado, uma vez que a câmera estará atrelada a Guerreira.

No segundo módulo a câmera se torna estática, consistindo simplesmente em uma tela de visualização que contém todos os elementos da cena e o jogador consegue visualizar tudo que ele precisa para interagir com os elementos presentes no módulo Figura x.

Figura 4. Representação do segundo módulo.

A Figura acima apresenta o esboço do segundo módulo, onde a câmera é estática e não tem qualquer tipo de movimentação, focando apenas em um ponto do cenário e exibindo tudo o que é relevante para a resolução do módulo. A utilização da câmera estática permitirá que o jogador se situe no cenário e tome suas decisões de jogo conforme aquilo que está vendo, nesse caso ele terá que observar o ambiente e o desafio na parte superior da tela.

6. Universo do Jogo

O primeiro módulo do jogo consistirá em uma biblioteca antiga com plataformas para a Guerreira andar, escadas para subir, blocos que ela poderá empurrar, puxar ou levantar. O cenário deve conter elementos que remetem a um ambiente antigo, luminárias e componentes característicos desse ambiente. O segundo módulo é uma caverna escura com pedras e elementos característico de grutas e tocas de animais.

No primeiro módulo o jogador irá aprender as mecânicas básicas que a Guerreira poderá executar e recolher dicas que vai ajudá-lo a resolver as proposições do módulo dois. O módulo dois deve possuir um espaço que vai surgir proposições que o jogador deverá acertar para subir de nível e conquistar pontos, para responder ele terá que clicar nos botões de ataque “V” para verdadeiro e “F” para falso.

A estrutura do jogo tem sua vertente baseada na revista em quadrinhos Logi Kingdom, mas com características distintas por se tratar de um jogo para dispositivos móveis. O mundo do jogo apresentará cenários com cores monocromáticas, personagens que estão presentes na revista, inimigos que vão dificultar a vida da Guerreira no seu objetivo principal que é salvar o mundo da inexistência da lógica.

No primeiro módulo a emoção do jogador estará relacionada com a descoberta e a exploração do ambiente do jogo. Ele poderá recolher dicas que estarão espalhadas em pergaminhos e isso dará uma ideia de o jogador pode “viver” em um mundo onde ele procure dicas que possam ajudá-lo a conquistar seus objetivos. O módulo dois depende das dicas que ele coletou, mas por não ser obrigatórias, caso não colete ele terá uma sensação de que deixou a ajuda passar. O jogo transmite as emoções da Guerreira para quem estiver jogando, é possível notar a força e determinação da personagem para trazer a lógica para seu mundo.

A música do jogo deverá ser misteriosa, mas no primeiro módulo ela poderá ser mais suave e no segundo poderá ser mais grave, visto que o desafio do jogo se concentra no módulo dois. Sugestão: https://opengameart.org/content/exploring-a-cave.

Mapa visual da tela inicial e epígrafe do jogo:

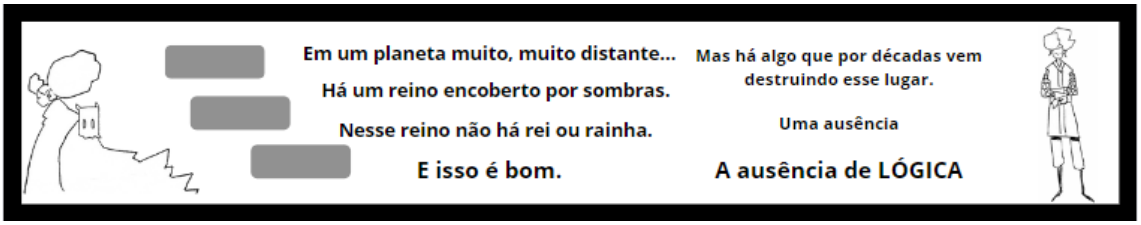


Figura 5: Representação visual da tela inicial do jogo.

Mapa do primeiro Módulo:

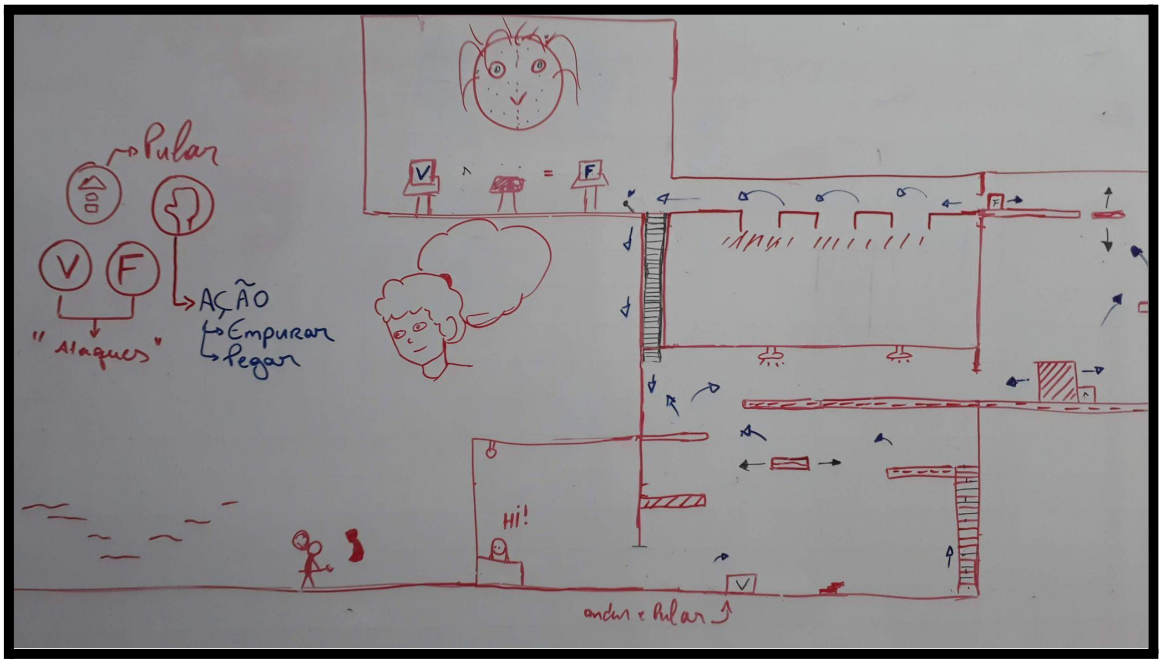


Figura 6: Representação do primeiro módulo do jogo - Biblioteca.

Mapa do segundo módulo:

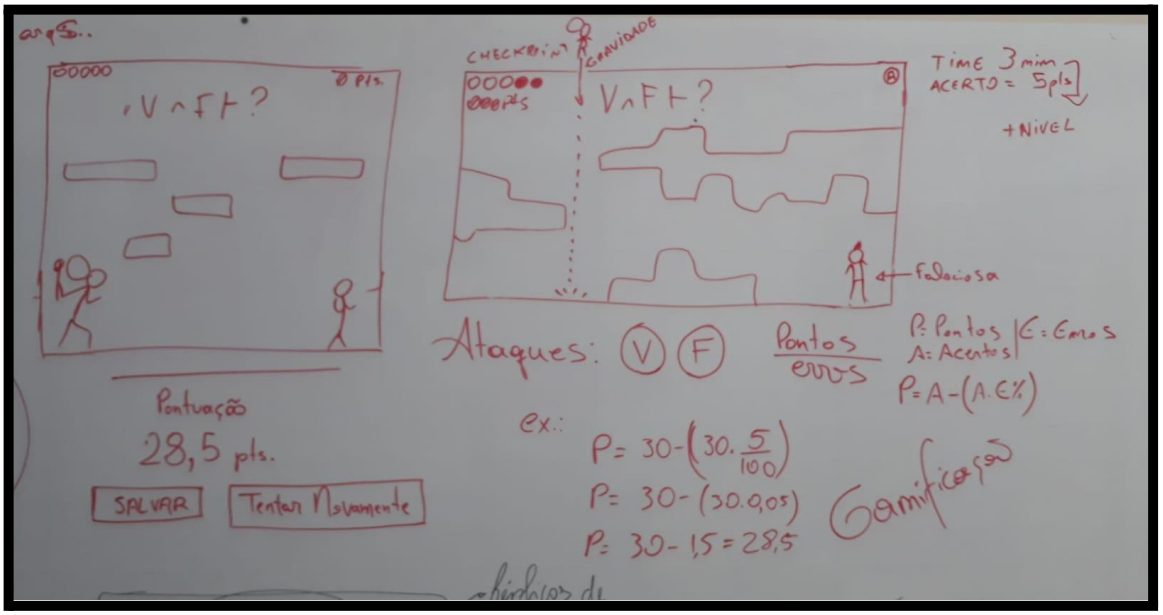


Figura 7: Representação do segundo módulo do jogo - Toca da Coruja.

7. Inimigos

7.1 Falaciosa

As Falaciosas são inimigas que a guerreira deverá enfrentar para restaurar a lógica do seu mundo. Os inimigos presentes no jogo são baseados em uma das vertentes da lógica, as Falácias, onde um raciocínio parece logicamente verdadeiro, mas existem algumas falhas que o fazem falsos. Os inimigos do jogo que estarão presentes no segundo módulo serão baseados nesses raciocínios incorretos.

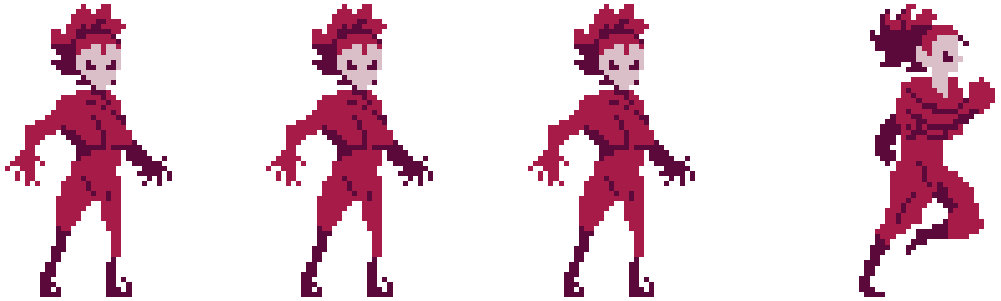


Figura 8. Ações e movimentos da Falaciosa.

Ficha técnica do personagem:

- **Nome da inimiga**: Falaciosa.
- **Idade**: indeterminada.
- **Tipo**: Inimiga da Guerreira.
- **Habilidades**: Ataque com chute.
- **Métricas de gameplay**: A falaciosa aparece no segundo módulo do jogo, após a Guerreira colocar a caixa certa no pedestal, e para derrotá-la a guerreira deverá acertar os desafios que aparecerão, clicando em verdadeiro ou falso para cada uma das proposições que surgirem. Ao derrotar as Falaciosas a Guerreira poderá prosseguir no jogo e retornar para a biblioteca do módulo inicial.

8. Interface

O controle virtual fica no canto inferior esquerdo da tela, já os botões de ação, pulo, dash, verdadeiro e falso, ficam no canto inferior direito. Na parte superior da tela fica as vidas da Guerreira simbolizadas por corujas no canto esquerdo e no canto direito fica o botão do fantasma, como ilustra a Figura 9.

Figura 9. Ilustração do HUD, posição dos botões.

Tela do Menu inicial:

Figura 10. Menu inicial do jogo.

Tela de Pause / dicas:

Figura 11. Tela de pause / dicas.

Tela Game Over:

Figura 12. Tela de fim de jogo.

Tela do Fim do Módulo:

Figura 13. Tela de conclusão do jogo.

Tela de Ranking:

Figura 14. Tela de ranking.

9. Cutscenes

O jogo possui uma *cutscene*, que se inicia assim que o jogador pressionar o botão de começar na tela inicial, a câmera se move para a direita dando a sensação de *parallax* no cenário, movendo lentamente as árvores que estão mais ao fundo e as mais a frente se movem mais rápido, enquanto surgem textos no centro da tela contextualizando o jogador sobre o universo em qual o jogo se passa.

O roteiro do jogo inicia-se com algumas frases que contextualizam a ideia central do *game*, as frases presentes nesta introdução são:

- Em um planeta muito, muito distante…
- Há um reino encoberto por sombras.
- Nesse reino não há rei ou rainha.
- E isso é bom.
- Mas há algo que por décadas vem destruindo esse lugar.
- Uma ausência…
- Ausência de lógica.

Cada frase do roteiro deverá ser trabalhada individualmente, mas que todas façam sentido para o jogador, sendo que a introdução de cada frase não deverá ultrapassar mais que dois segundos para não deixar o contexto solto e sem sentido para o jogador.

Esta *cutscene* será construída com *script* implementando movimento no cenário deixando a câmera parada limitando uma certa distância e assim que o objeto do cenário atingi-la é parado, os textos serão trocados a cada segundo, ao esconder o objeto do texto troca-se a frase que aparecerá de acordo com a lista e por fim se torna visível novamente.

10. Cronograma

A equipe do Jogo Logi Kingdom iniciou seus trabalhos em fevereiro começando com a produção do GDD e do roteiro do jogo que é baseado na revista em quadrinhos de mesmo nome. Ainda em fevereiro foi realizada uma reunião com a equipe do Logi Kingdom para validação do GDD.

Em março, com a aprovação do GDD iniciou-se a produção das artes e desenhos dos personagens, juntamente com o desenvolvimento de controle do jogador e sistemas de colisões que trabalham juntos, no fim de março e começo de abril iniciou-se a produção dos cenários dos módulos.

	Fevereiro				Março				Abril				Maio				Junho				
Tarefa/Semana	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	**Progreso**
Escrever o GDD																					Planejado
Apresentar GDD																					Planejado
Selecionar/desenhar a arte dos personagens e cenários																					Planejado
Desenvolver o sistema de controle do jogador																					Planejado
Desenvolver sistema dos módulos e fases																					Planejado
Implementar a detecção de colisão																					Planejado
Desenvolver sistema de pontuação																					Planejado
Implementar inimigos																					Planejado