ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
DEPARTAMENTO DE ENGENHARIA DA COMPUTAÇÃO E SISTEMAS DIGITAIS

Beogival Wagner Lucas Santos Junior
Gabriel Moreira Minossi
Victor de Mendonça Carbonezi

Jogo digital para desenvolver a habilidade de colaboração em programação

São Paulo
2021

Beogival Wagner Lucas Santos Junior
Gabriel Moreira Minossi
Victor de Mendonça Carbonezi

Jogo digital para desenvolver a habilidade de colaboração em programação

Trabalho de conclusão do curso apresentado a Escola Politécnica da Universidade de São Paulo para a obtenção do título de Engenheiro de Computação.

São Paulo

2021

Beogival Wagner Lucas Santos Junior
Gabriel Moreira Minossi
Victor de Mendonça Carbonezi

Jogo digital para desenvolver a habilidade de colaboração em programação

Trabalho de conclusão do curso apresentado a Escola Politécnica da Universidade de São Paulo para a obtenção do título de Engenheiro de Computação.

Área de Concentração: Engenharia de Computação e Sistemas Digitais

Orientador: Prof. Dr. Ricardo Nakamura

São Paulo

2021

AGRADECIMENTOS

"Agradeço ao professor Doutor Ricardo Nakamura, pela orientação, paciência e por todo o tempo dedicado a me ajudar no desenvolvimento deste trabalho e nas três disciplinas que tive o privilégio de participar com ele. Agradeço também ao professor Doutor Romero Tori pelo conteúdo ministrado na disciplina de Tecnologia de Computação Gráfica, e aos meus colegas Daniel Piassi, Mayara Gomes e Paulo Hirami, que contribuíram com os elementos gráficos do projeto. Sou grato à professora Doutora Lúcia Vilela Leite Filgueiras pelo conteúdo da disciplina de Interação Humano-Computador que, juntamente com meus colegas Maria Barceló, Pedro Oliveira e Renan Nunes, enriqueceram também o projeto. Agradeço à Luiza Minossi, pela ajuda indispensável nos últimos momentos da redação desta monografia. E por fim, sou muito grato à minha namorada e futura esposa Débora, aos meus irmãos Su, Lu, Audrinha, Tutu, Benja e Cathe, à minha mãe Rosangela, ao meu pai Beogival, à minha madrasta Letícia e à minha falecida avó Concheta, por me sustentarem emocionalmente e financeiramente, garantido todos os privilégios necessários para que eu pudesse realizar este trabalho." - *Beogival Wagner Lucas Santos Junior*

“Em primeiro lugar, agradeço ao professor doutor Ricardo Nakamura pela orientação, pelo tempo e pelas disciplinas que ministrou. Pelas oportunidades e pelo suporte, gostaria de agradecer a meus companheiros: Luke e Léia, meus cachorros, Firelord, meu querido laptop e Luiza, minha irmã. Agradeço também às Mímicas e Mímicos desta e de outras semanas, sem os quais ainda estaria cursando Cálculo 3. Finalmente agradeço a Zeus pela força e motivação.” - *Gabriel Moreira Minossi*

“Agradeço primeiramente a orientação do professor Doutor Ricardo Nakamura, sem ele o resultado final deste trabalho não teria sido tão satisfatório quanto foi. Sou grato a toda a minha família, principalmente minha avó Elide, que me sustentaram em todos os sentidos durante esses tempos mais difíceis do que o norma,” - *Victor de Mendonça Carbonezi*

"Agradecemos também a Kay Lousberg, que disponibilizou os recursos gráficos utilizados.”

RESUMO

Ensinar a programar é uma tarefa que envolve vários aspectos, como, por exemplo, o raciocínio lógico, os símbolos específicos para cada linguagem, as regras de nomenclatura e definição de identificadores, entre outros. Dentre as diversas formas de ensino, existe uma que se dá por meio de jogos digitais voltados para a educação. Esses jogos, em geral, abordam particularmente o aspecto do raciocínio lógico, por ser um ponto em comum entre todas as diversas linguagens de programação. Este trabalho contribui para o gênero trazendo um jogo que aborda como nomear, de forma adequada, alguns tipos de identificadores de linguagens de programação, particularmente identificadores de funções e de parâmetros.

O propósito deste trabalho foi o de desenvolver uma plataforma onde se possa praticar, de forma lúdica e descontraída, algumas diretrizes de nomenclatura de funções e variáveis. Para dar vida ao protótipo, algumas técnicas de projeto foram necessárias, entre elas destaca-se a de subdividir o processo de desenvolvimento em quatro etapas: a concepção, a pré-produção, a produção e a manutenção. Considerando as limitações e responsabilidades deste trabalho, foram enfatizadas principalmente as etapas de concepção e de produção.

O resultado obtido foi um protótipo de jogo cooperativo que pode ser considerado como uma prova de conceito para o que foi planejado como propósito desse trabalho. O jogo cumpriu todos os requisitos mínimos esperados e testes informais demonstraram um forte estímulo ao cumprimento das regras de nomeação planejadas para cada situação. O uso da cooperação como forma de validação para a adequação dos nomes também demonstrou um resultado satisfatório. Afinal, se a dupla de jogadores foi capaz de resolver os desafios a partir de um determinado conjunto de nomes, é plausível concluir que esses são nomes suficientemente adequados.

Palavras-chave: Jogos sérios. Jogos educativos. Ensino de programação. Nomeação de funções e parâmetros. Jogo cooperativo. Comunicação efetiva.

ABSTRACT

Teaching to program is a task that involves several aspects, such as, for example, logical reasoning, specific symbols for each language, naming rules and definition of identifiers, among others. Among the various forms of teaching, there is one that takes place through digital games aimed at education. These games, in general, particularly address the aspect of logical reasoning, as it is a common point among all the different programming languages. This work contributes to the genre by bringing a game that addresses how to properly name some types of programming language identifiers, particularly function and parameter identifiers.

The purpose of this work was to develop a platform where one can practice, in a playful and relaxed way, some nomenclature guidelines for functions and variables. To bring the prototype to life, some design techniques were necessary, amongst them the one to subdivide the development process into four stages: conception, pre-production, production and maintenance. Considering the limitations and responsibilities of this work, the conception and production stages were mainly emphasized.

The result obtained was a cooperative game prototype that can be considered a proof of concept for what was planned as its purpose. The game fulfilled all the expected minimum requirements and its informal tests demonstrated a strong incentive to comply with the naming rules planned for each situation. The use of cooperation as a form of validation for the adequacy of names also demonstrated a satisfactory result. After all, if both players were able to solve the challenges from a given set of names, it is plausible to conclude that these are adequate enough names.

Keywords: Serious games. Educational games. Programming teaching. Function and parameter naming. Cooperative play. Effective communication.

SUMÁRIO

1. INTRODUÇÃO 1
1.1 Motivação 1
1.2 Objetivo 2
1.3 Justificativa 3
1.4 Organização do Trabalho 4
2 ASPECTOS CONCEITUAIS 5
2.1 Nomeação de funções e parâmetros 5
2.2 Jogos digitais 5
2.3 Jogos aplicados à educação 6
2.4 Game Engine 3
2.5 Cliente/Servidor 7
2.6 Open source 7
3. TECNOLOGIAS UTILIZADAS 8
3.1 GitHub 8
3.2 Godot 8
3.3 Nakama 9
3.4 Blender 9
4. MÉTODO DO TRABALHO 11
5. ESPECIFICAÇÃO DE REQUISITOS DO SISTEMA 13
6. IMPLEMENTAÇÃO 15
6.1 Servidor / Cliente 15
6.2 Godot 15
6.3 Ordem de priorização e implementação das funcionalidades 18
7. RESULTADOS 20
7.1 Concepção 20
7.1.1 Modo de Criação 21
7.1.2 Modo de Labirinto 22
7.2 Produção 23
7.2.1 Modo de Criação 23
7.2.2 Peça A 26
7.2.3 Peça B 29
7.2.4 Peças C 30

7.2.5 Labirintos 32
7.2.6 Labirinto 1 34
7.2.7 Labirinto 2 35
7.2.8 Labirinto 3 35
7.2.9 Labirinto 4 36
7.3 Recursos gráficos 37
7.3.1 Personagens principais 37
7.3.2 Recursos adquiridos de Kay Lousberg 40
8. PLANO DE TESTES 42
8.1 Plano de testes 42
8.1.1 Entrevista 42
8.1.2 Teste de usabilidade: 42
8.1.3 Termo de Consentimento Livre e Esclarecido: 43
8.2 Testes informais 43
9 CONCLUSÃO 45
REFERÊNCIAS 46
REFERÊNCIAS CONSULTADAS 47
APÊNDICE A - Teste de Usabilidade 48

1. INTRODUÇÃO

Nesta monografia é apresentado o desenvolvimento de um jogo aplicado ao aprendizado de habilidades e práticas associadas à colaboração em atividades de programação. Mais especificamente, a colaboração que se dá ao nomear funções e parâmetros de uma interface que será utilizada por outros membros de um determinado time de desenvolvimento de software. O trabalho foi motivado pela importância dessa etapa do desenvolvimento, que é muitas vezes subestimada, apesar de algumas pesquisas apontarem uma importância equiparável à própria documentação. O objetivo é desenvolver um jogo digital que seja capaz de fornecer um meio pelo qual se pode treinar e desenvolver tais habilidades, usando-se de mecanismos lúdicos associados a metáforas que tenham amparo em regras de nomeação bem estabelecidas por livros como o Code Complete (MCCONNELL; 1993) e o Clean Code (MARTIN; 2009). São comuns os jogos que procuram ensinar a lógica de programação e o desenvolvimento do raciocínio lógico de forma geral. A contribuição do presente trabalho é trazer um jogo que foque na nomeação como forma de documentação e comunicação de um software.

1.1 Motivação

A documentação é muito importante no processo de desenvolvimento de software, mas existe algo que é tão importante quanto e ainda assim menos lecionado nos cursos e ferramentas de ensino de programação: a nomeação de parâmetros, funções e classes. A nomeação pode ser inclusive mais importante que a documentação, do ponto de vista da compreensão ágil de um código fonte (BLINMAN; COCKBURN, 2005). Blinman e Cockburn (2005) compararam quatro cenários de compreensão de código: um no qual o código deveria ser compreendido contendo documentação e nomes descritivos em suas funções e parâmetros; outro no qual o código possuía documentação mas nomes não descritivos; um terceiro no qual o código não possuía documentação mas nomes descritivos; e um último no qual o código não possuía nem documentação nem nomes descritivos. Eles concluem nesse artigo que o segundo melhor cenário para um framework ser bem compreendido de forma ágil é o cenário onde esse framework tem nomes descritivos em suas funções e não tem documentação, e não onde ele tem nomes não descritivos para suas funções mas tem documentação. Ou seja, nesse teste, mostraram-se mais importantes os nomes

descritivos do que uma documentação. O primeiro melhor cenário é, evidentemente, quando se tem nomes descritivos e documentação para as funções. No mundo corporativo, onde os protótipos precisam ser rápidos e baratos, pode-se considerar muito importante esse resultado apresentado por Blinman e Cockburn (2005). Afinal, produzir documentação costuma ser mais oneroso do que o simples ato de nomear de forma descritiva as classes, as funções, os parâmetros e as variáveis. Porém, ser menos oneroso não significa, necessariamente, ser mais fácil.

Um dos maiores desafios enfrentados em projetos de desenvolvimento de *software* é a disseminação e a definição de boas práticas de nomenclatura. Não foi por acaso que Phil Karton cunhou a famosa frase: "Existem apenas duas coisas difíceis em ciência da computação: invalidação de cache e nomear coisas."[1]. (HILTON; HERMANS, 2017, tradução nossa). Alguns programadores cometem o erro de focar muito na executabilidade do código e não no valor dele como algo para humanos lerem, esquecendo uma outra citação famosa de Knuth: "Os programas são feitos para serem lidos por humanos e apenas incidentalmente para serem executados por computadores."[2] (HILTON; HERMANS, 2017, tradução nossa). Hilton e Hermans (2017) compilaram em seu artigo diretrizes que forneçam ajuda prática, na hora de nomear classes, funções e parâmetros, pois, segundo eles, é difícil de se encontrar diretrizes que sejam razoáveis e práticas ao mesmo tempo. Por exemplo, a diretriz: "Os nomes das variáveis devem ser curtos, mas significativos" é razoável, mas não oferece uma ajuda muito prática.

1.2 Objetivo

As diretrizes apresentadas por Hilton e Hermans (2017), trazem de fato mais praticidade para o assunto. Elas são todas acompanhadas por exemplos e por boas justificativas, e algumas também por boas referências. O que se pretende então acrescentar ao assunto, com o presente trabalho de conclusão de curso, é um meio no qual se possa colocar em prática e treinar essas e outras diretrizes.

Este trabalho pretende, então, desenvolver um jogo digital que explore diretrizes de nomeação de funções e parâmetros, de forma a incentivar que os jogadores

[1] Frase original: "There are only two hard things in Computer Science: cache invalidation and naming things."
[2] Frase original: "Programs are meant to be read by humans and only incidentally for computers to execute."

treinem e aprendam essas regras. A intenção não é, necessariamente, desenvolver uma ferramenta de ensino que deva ser usada por professores ou em cursos de programação. O plano é tentar manter o lado lúdico e despretensioso do mundo dos jogos, mas trazer também esse conteúdo de valor para quem possa apreciar.

O público alvo, a princípio, se restringiria a jogadores casuais com inclinação para jogos multijogador cooperativos com um foco na comunicação e na resolução de *puzzles*[3].

1.3 Justificativa

Hilton e Hermans (2017) comentam em seu artigo que nomear não é um tópico novo, é antigo e já muito explorado por diversos livros como por exemplo o Code Complete (MCCONNELL; 1993) e o Clean Code (MARTIN; 2009). Esses autores comentam também que livros de programação mais recentes infelizmente tendem a dedicar menos páginas ao assunto, mas talvez por terem pouco a acrescentar sobre ele. Por que então a frase[4] de Karton ainda hoje contempla a frustração de tantos programadores? Se nomenclatura já é um problema resolvido, do ponto de vista conceitual, o que ainda falta para que ela atinja de vez o mundo da programação? Para Hilton e Hermans (2017), o que faltava eram diretrizes mais práticas.

Na CppCon 2019, Kate Gregory apresentou algumas regras e conselhos a respeito de nomenclatura e defendeu fortemente a importância de bons nomes (informação verbal)[5]. Segundo Gregory, nomear é subjetivo, ou seja, não é algo que se possa colocar em uma "máquina" de correção de código e integrar a uma IDE[6]; logo, a melhor maneira de se desenvolver tal habilidade é praticando. É difícil decorar diretrizes sem colocá-las em prática, mas é também difícil colocar essas diretrizes em prática em um ambiente profissional, onde refatorar custa caro e os prazos são apertados.

Um jogo digital seria então o meio ideal para que se pudesse praticar diretrizes de nomenclatura sem o ônus associado ao ambiente profissional. Jogos Sérios (LAMB, et al., 2018) vêm sendo usados como métodos efetivos de ensino já há algum tempo, e se tornaram uma abordagem comum para um número crescente de organizações (ȘTEFAN, et al., 2019). O apelo lúdico dos jogos, associado à proximidade

[3] Puzzle é um termo em inglês que se traduz literalmente para quebra-cabeças, muito usado para descrever o gênero de jogos voltados para a resolução de desafios mentais e labirintos.
[4] "There are only two hard things in Computer Science: cache invalidation and naming things."
[5] CppCon 2019: Kate Gregory “Naming is Hard: Let's Do Better”
[6] Do inglês Integrated Development Environment ou Ambiente de Desenvolvimento Integrado.

que eles costumam ter com o mundo da programação, são pontos que sustentam a existência e o sucesso desse nicho. É muito comum encontrar jogos que ensinam a programar focados na lógica de programação e nas operações mais comuns desse meio, como desvios condicionais, operações booleanas, laços etc.

A lacuna que esse trabalho pretende preencher, então, é a de um jogo digital voltado para programação que foque mais nos aspectos de nomenclatura e comunicação entre jogadores do que nos aspectos de lógica computacional. Um jogo que possa apresentar as diferentes diretrizes do assunto, demonstrando a importância de cada uma delas e fornecendo um meio onde se possa praticar a nomeação de forma lúdica. Será utilizado, inclusive, uma mecânica cooperativa de multijogadores como forma de validação orgânica da qualidade das habilidades de nomeação desenvolvidas pelos jogadores. Afinal, uma boa maneira de se verificar se um nome comunica claramente o comportamento de uma dada função é usando-o como único ponto de comunicação entre dois jogadores distintos.

1.4 Organização do Trabalho

O trabalho começa, no capítulo II, apresentando os principais aspectos conceituais que irão guiar o desenvolvimento do jogo, para que os conceitos passados por ele sejam válidos e para que o projeto como um todo seja de fato concluído dentro do prazo. Em seguida, no capítulo III, serão listadas e justificadas as ferramentas que devem ser utilizadas no desenvolvimento do jogo. No capítulo IV, é detalhado o planejamento do processo de desenvolvimento, descrevendo as suas etapas e os resultados esperados em cada uma delas. Para garantir a qualidade e a coerência do jogo desenvolvido, o capítulo V detalha o que é indispensável para o resultado final do projeto. No capítulo VI, são apresentados os resultados obtidos ao final do desenvolvimento do jogo, elencando novamente os objetivos iniciais com os resultados finais. Um possível plano de testes será discutido no capítulo VII, embora no presente trabalho não houve tempo hábil para realizar tais testes. Os últimos capítulos, VIII e IX, concluem o trabalho e apresentam as referências, respectivamente.

2. ASPECTOS CONCEITUAIS

2.1 Nomeação de funções e parâmetros

O trabalho foi amparado conceitualmente no artigo de Hilton e Hermans (2017) e nos livros Code Complete (MCCONNEL; 1993) e Clean Code (MARTIN; 2009), para definir e delimitar quais diretrizes de nomeação foram exploradas no jogo, garantindo assim que os *puzzles* criados estimulem regras coerentes e bem embasadas.

Uma função, neste contexto, é um conjunto de operações computacionais executadas sequencialmente com um objetivo final. Em programação, definem-se funções de acordo com as necessidades de um *software*, e o conjunto dessas funções, quando executadas em uma ordem específica, formam um *software*. Os parâmetros são as entradas que essa função pode receber para operar de forma mais genérica. O nome que uma função pode receber, assim como os de seus parâmetros, é criado pela pessoa que a desenvolveu com o objetivo de deixar claro para que serve a tal função. Por exemplo, se uma função calcula o comprimento de uma circunferência de raio dois, concatenando sequencialmente um conjunto de operações matemáticas, ela poderia ser nomeada "comprimento_circunferência_raio_dois". Porém, adicionando um parâmetro genérico, essa função poderia calcular o comprimento de uma circunferência de qualquer raio, com as devidas modificações internas. Essa função hipotética poderia então ter um novo nome "calcular_comprimento", com um parâmetro chamado "raio", que pode ser um número natural qualquer. Em programação, é comum que as funções não tenham nomes muito grandes, com mais de 32 caracteres, por exemplo, e que esses nomes venham seguidos de seus parâmetros entre parênteses, por exemplo "calcular_comprimento(raio)".

A escolha de quais os parâmetros e qual seria o conteúdo dessas funções não foi o tema principal desse trabalho, mas sim, dada uma certa função de nome N e parâmetros de nome X, Y e Z, quais seriam os melhores nomes a serem atribuídos a essa função.

2.2 Jogos digitais

A definição de jogo digital, de acordo com Miranda e Stadzisz (2017, p. 4), é

> Atividade voluntária, com ou sem interesse material, com propósitos sérios ou não, composta por regras bem definidas e objetivos claros, capazes de

> envolver os(as) jogadores(as) na resolução de conflitos e que possui resultados variáveis e quantificáveis. Esta atividade deve ser gerenciada por software e executada em hardware.

No presente trabalho, o jogo desenvolvido faz parte de uma categoria específica de jogos digitais conhecida como *puzzle*, que pode ser definida de acordo com Tonéis e Frant (2015, p. 98)

> Os *puzzles* constituem um tipo de enigma ou problema cuja finalidade está em desenvolver o raciocínio lógico e matemático em seus níveis ontológicos, cognitivos, como força motriz para as mais diversas formas de produção de conhecimentos. Dessa maneira, os puzzles podem estar presentes em games com diferentes temas e objetivos, seja na matemática ou qualquer outra ciência.

Com o objetivo de tornar os desafios atraentes e instigantes, foram usados conceitos apresentados por Jolie Menzel em sua palestra "Level Design Workshop: Solving Puzzle Design"[7].

2.3 Jogos aplicados à educação

Os jogos aplicados à educação são conhecidos como Jogos Sérios e eles são, de acordo com Aldrich (2005), jogos que têm o propósito de ensino, aprendizagem ou treinamento e não apenas entretenimento. Essa categoria de jogos associada ao gênero de *puzzle* é o que guiará o desenvolvimento desse projeto e, para garantir o caráter educacional do jogo, é usado o livro Serious Educational Games (2008) e o artigo Concepts Of Serious Game In Education de Darwesh (2015). Para aproveitar corretamente a interação multijogador, do ponto de vista do aprendizado, foi estudado o artigo de Paraskeva; Mysirlaki e Papagianni (2010).

2.4 Game Engine

De acordo com Valencia-Garcia et al (2016), uma *game engine* é um conjunto de ferramentas de *software* concebido para o desenvolvimento de jogos digitais, que geralmente inclui bibliotecas de código relevantes. O termo "engine" vem do inglês e significa motor em português, fazendo uma analogia ao motor de um carro, como se esse conjunto de ferramentas fosse o que impulsiona o jogo, principalmente do ponto de vista gráfico e de simulação física. O *site*[8] da documentação da Godot, *engine*[9] do jogo, foi de extrema importância para o seu desenvolvimento.

---

[7] GDC 2016 Level Design Workshop: Solving Puzzle Design, Jolie Menzel.
[8] *Site* da documentação da Godot disponível em: <https://docs.godotengine.org/>. Acesso em: 16 nov. 2021.
[9] *Engine* é um programa de computador e/ou conjunto de bibliotecas, para simplificar o desenvolvimento de jogos eletrônicos.

2.5 Cliente/Servidor

A arquitetura cliente/servidor, de acordo com Tanenbaum (apud Picoli, 2011, p. 32),

> [...] possibilita que dois processos executados em máquinas distintas enviem mensagens um para o outro, e para isso, um processo deve ser o servidor, que estará escutando em uma determinada porta e o outro processo, o cliente, o qual enviará uma mensagem ao servidor através de algum protocolo (TANENBAUM, 1995). Para que o cliente envie com sucesso uma mensagem ao servidor, este deve necessariamente estar escutando em alguma porta.

2.6 Open source

Open source é um termo usado no mundo da computação para se referir ao software que atende a uma série de critérios que especificam os modelos de distribuição e licenciamento do código. De acordo com KAVANAGH (2007), esses critérios são: redistribuição gratuita, código-fonte, trabalhos derivados, integridade do código-fonte do autor, não discriminação a pessoas ou grupos, não discriminação a campos de aplicação, distribuição de licença, licenças não devem ser específicas para um produto, licenças não devem restringir outros softwares, e a licença deve ser neutra a tecnologias.

De acordo com a distribuição livre, a licença não impede que nenhuma das partes venda ou distribua gratuitamente o software como um componente de uma distribuição de softwares agregados que contenha programas de diversas fontes. O programa deve incluir o código fonte e deve permitir a distribuição desse código de forma compilada. A licença deve permitir modificações e trabalhos dela derivados, assim como permitir que sejam distribuídos sob os mesmos termos da licença do software original. A licença pode impedir que o código-fonte seja distribuído em sua forma modificada. Os direitos ligados ao programa devem se aplicar a todos a quem o programa seja redistribuído, sem necessidade de execução de uma licença adicional para essas partes.

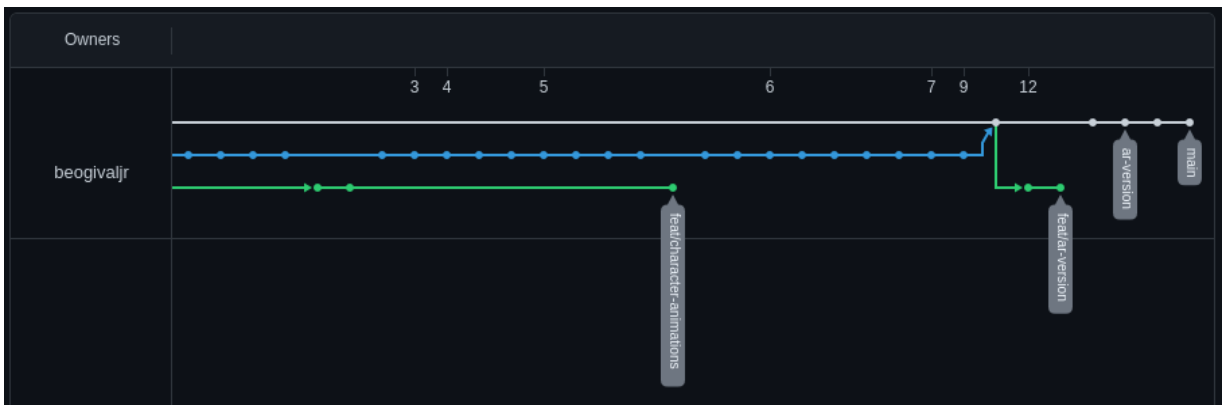
3 TECNOLOGIAS UTILIZADAS

Para a execução do projeto foram necessárias diversas ferramentas; entre elas as principais são a de controle de versão e distribuição do código fonte, a de gerenciamento de projetos, a de desenvolvimento do lado cliente e servidor do jogo, a de modelagem gráfica e, por fim, a ferramenta de hospedagem online.

3.1 GitHub

O GitHub[10] foi escolhido para o controle de versão e distribuição do código fonte, pois é uma ferramenta que usa o Git como tecnologia de versionamento e é também uma ferramenta já muito conhecida e utilizada no mundo *open source*[11], já que um dos principais requisitos do projeto é que ele seja de código aberto e livre para que futuros desenvolvedores e alunos possam se beneficiar deste trabalho. Um outro motivo para a escolha dessa ferramenta foi o fato de ela ser gratuita e possuir um sistema de gerenciamento de projetos integrado ao controle de versão, logo também foi usado o GitHub para o gerenciamento do projeto.

Figura 1 - Árvore de commits no GitHub



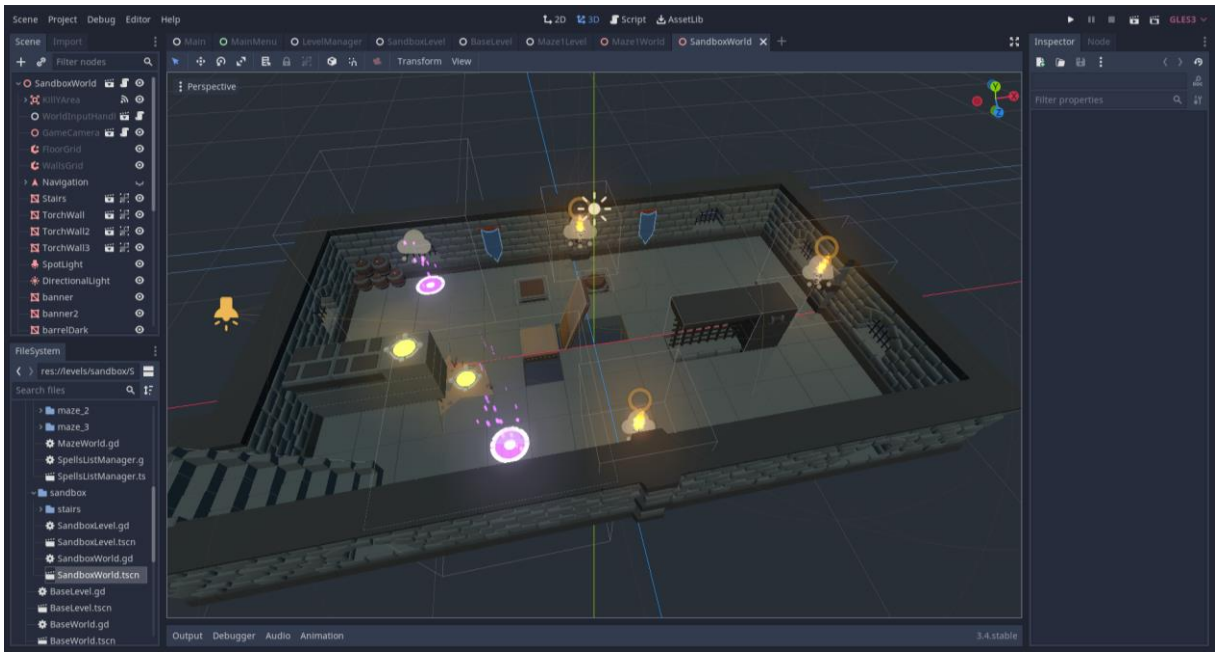
Fonte: GitHub disponível em: <https://github.com/beogivaljr/blorgs/network>. Acesso em: 6 dez. 2021

[10] Página principal do site disponível em: <https://github.com/>. Acesso em: 6 dez. 2021.
[11] Open Source é o nome dado ao modelo de desenvolvimento de software que promove a colaboração de produção intelectual.

3.2 Godot

A Godot[12] foi escolhida como a *engine* de desenvolvimento do lado cliente e servidor do jogo. A decisão não foi fácil, pois se trata da principal ferramenta do projeto, uma que tomou a maior parte do tempo de desenvolvimento, porém a escolha pode ser justificada pelos seguintes pontos: é uma ferramenta completamente *open source*, ou seja, usando-a, pode-se desenvolver e publicar o código fonte seguindo poucas restrições[13]; é muito mais leve do que as outras competidoras, tornando o projeto resultante ainda mais acessível para futuros desenvolvedores.

Figura 2 - Ambiente de composição de “cenas” da ferramenta Godot



Fonte: Elaborado pelos autores.

3.3 Nakama

O Nakama[14] foi escolhido como a ferramenta para implementação do servidor e da conexão com o cliente, visto que ele possui uma boa integração com a Godot e funções que facilitam para os desenvolvedores a parte de conexão, captura de pacotes e abstração dos protocolos de comunicação em rede. O Nakama permite a implementação de funções que são chamadas em momentos específicos da comunicação do jogo com o servidor, como na criação de uma partida e na conexão de um novo

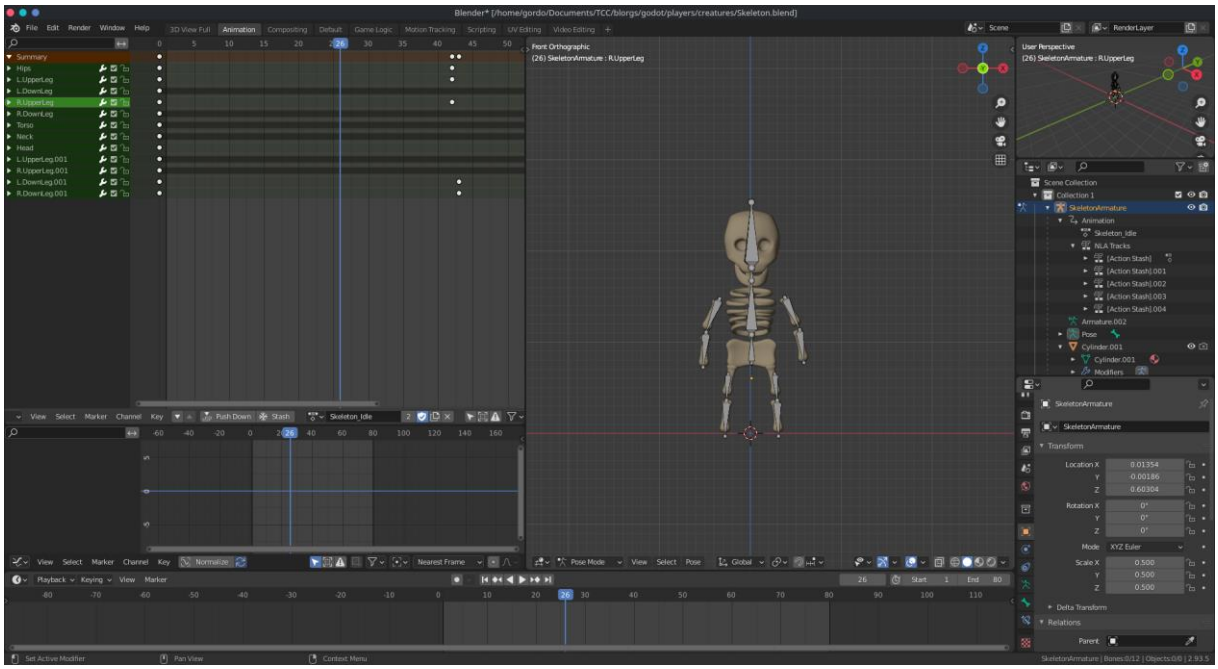
[12] Encontra-se disponível em: <https://godotengine.org/>. Acesso em: 6 dez. 2021.
[13] O conjunto completo de regras da Godot pode ser encontrado em sua página inicial do GitHub. Disponível em:<https://github.com/godotengine/godot-docs>. Acesso 22 nov. 2021.
[14] Disponível em: <https://heroiclabs.com/>. Acesso em 18 dez. 2021.

jogador. Também é disponibilizada uma interface para troca de mensagens arbitrárias com o servidor, que é utilizada para enviar informações relevantes do estado do jogo.

3.4 Blender

O Blender[15] também foi eleito pelo fato de ser gratuito e open source, algo que simplifica o processo de abertura do código do projeto. Além disso, é uma ferramenta de modelagem e animação muito poderosa e inclusive considerada um dos padrões industriais.

Figura 3 - Leiaute padrão do fluxo de trabalho de animação no Blender



Fonte: Elaborado pelos autores.

[15] Site da ferramenta disponível em: <https://www.blender.org/>. Acesso em: 22 nov. 2021.

4. MÉTODO DO TRABALHO

O trabalho se divide em duas grandes responsabilidades; a primeira é a presente documentação e a segunda é o desenvolvimento do jogo efetivamente, tanto a aplicação do cliente quanto a aplicação do servidor. A documentação não é incluída explicitamente no planejamento, pois deve se desenvolver organicamente durante o projeto. O desenvolvimento do jogo é subdividido em quatro etapas (Fullerton, 2014).

A primeira é conhecida como a etapa da concepção. Nela, o time de desenvolvimento cria a proposta do jogo, tenta definir a suas regras e os objetivos, lembrando sempre de garantir que essas características estejam de acordo com os objetivos do trabalho. Isso é feito por meio de mecanismos de geração de ideias, como o *brainstorming*[16] por exemplo, e a prototipação em papel ou softwares de representação gráfica não programáveis. Essa etapa deve consumir em média um mês da equipe e os principais entregáveis no final são a primeira versão do documento de *design* do jogo e uma lista de principais riscos do projeto.

A próxima etapa costuma ser a pré-produção. Nela avaliam-se os riscos do projeto e aplicam-se eventuais ajustes ao *design* do jogo. Em geral, esses riscos envolvem questões de investimento e retorno do projeto, algo que dentro do contexto deste trabalho não faz muito sentido, já que não se espera nenhum retorno financeiro. O único aspecto dessa etapa que será aproveitado é a validação do presente documento de *design* do jogo frente aos requisitos e objetivos do projeto. Por esse motivo, essa etapa será muito sucinta, algo em torno de uma semana, mas podendo ainda gerar no final um novo documento de *design* do jogo.

A terceira etapa é a de produção. Nela ocorre o desenvolvimento do jogo propriamente dito. Será usado um método iterativo de desenvolvimento de *software* conhecido como Agile[17]. O tempo reservado para essa etapa é de aproximadamente dois meses, divididos em 8 Sprints[18]. Ao final dessa etapa, é esperado que se tenha o lançamento de uma primeira versão do jogo, além de sua documentação técnica.

[16] *Brainstorming* é uma técnica de discussão em grupo que se vale da contribuição espontânea de ideias por parte de todos os participantes, no intuito de resolver algum problema ou de conceber um trabalho criativo. Definição de Oxford Languages.
[17] Conjunto de regras de planejamento de projetos muito utilizadas em desenvolvimento de *software*.
[18] Nome dado ao período de um ciclo de um projeto ágil.

A última etapa prevista é a de manutenção. Nela, a primeira versão do jogo será reavaliada, corrigida e ampliada. Essa etapa não entra no escopo deste trabalho, por ser mais voltada para o crescimento do jogo e para a definição de novos requisitos.

5. ESPECIFICAÇÃO DE REQUISITOS DO SISTEMA

Os requisitos funcionais para o jogo são:

- Permitir a entrada, pelo jogador, de nomes de funções e variáveis;
- Gerar nível;
- Incluir jogadores ao nível;
- Salvar progresso do nível para que possa ser usado em outro;
- Executar interação entre jogador e objeto do nível, conforme definido pelas regras do jogo;
- Fornecer funções com nomes estranhos aos jogadores;
- Solicitar seleção de função;
- Executar função;
- Executar troca de controle de objetos entre jogadores;
- Fornecer interface de troca de controle;
- Fornecer interfaces para que exatamente duas pessoas joguem em tempos diferentes;
- Garantir que a comunicação entre os jogadores seja indireta, por meio de nomes de funções e nomes de parâmetros;
- Proporcionar um ambiente de treino aos jogadores onde se possa desenvolver agilidade e capacidade de síntese ao se comunicar;
- Deve ter um sistema de convidar outros jogadores de maneira descomplicada.

Os requisitos não funcionais são:

- Deve ter um servidor para facilitar e centralizar a comunicação entre os clientes;
- Deve ser possível de ser executado em dispositivos móveis com Android versão 8.0 ou superior, ou iOS 12 ou superior;
- Deve ser de código aberto;
- Deve possuir um banco de dados relativo aos usuários;
- Deve armazenar temporariamente durante a partida o progresso de cada par de jogadores;
- Os níveis devem ser jogáveis em um período de aproximadamente 5 minutos;

- Deve possuir uma temática atrativa;
- Permitir que jogadores em diferentes plataformas joguem na mesma seção.

6. IMPLEMENTAÇÃO

6.1 Servidor/Cliente

O Nakama define um conjunto de funções que são chamadas em momentos específicos da comunicação servidor/cliente. Para o cliente, o Nakama abstrai funções de autenticação, conexão e desconexão com o servidor, e de se juntar à partida. Para o servidor, o Nakama oferece o controle fino da função que é executada ciclicamente até o fim da partida, e de funções que são chamadas quando ocorrem determinados eventos, tais quais a criação da partida e a entrada ou a saída de um jogador da partida.

Além disso, o Nakama permite o envio de mensagens arbitrárias pertinentes ao estado do jogo. Cada uma das mensagens tem um significado diferente e é utilizada em um momento específico do ciclo de vida do jogo. Quando essas mensagens são recebidas pelo servidor, são armazenadas em uma fila e processadas na função que roda ciclicamente (match_control.match_loop). Quando são recebidas pelo cliente, causam execuções de funções específicas.

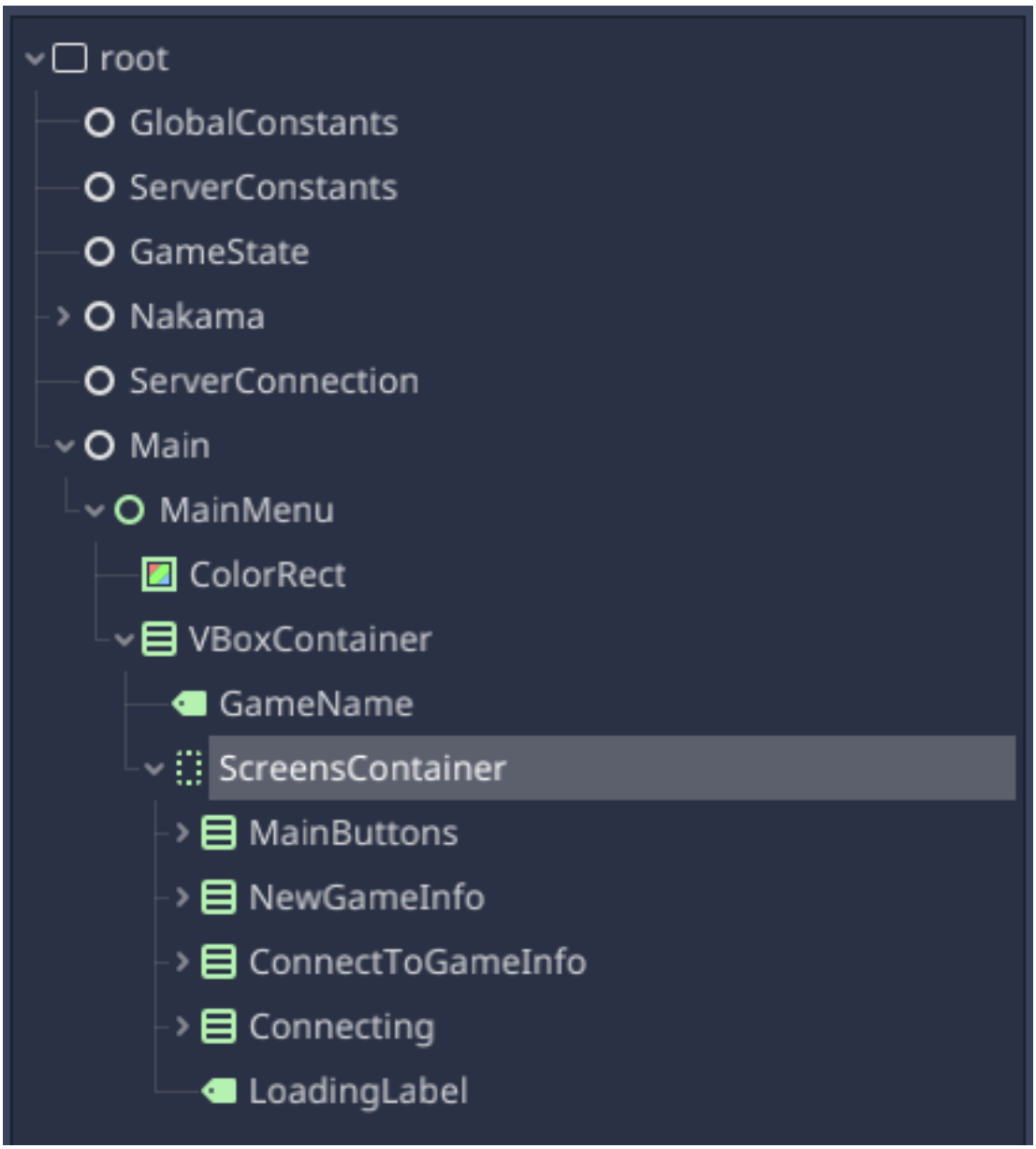
6.2 Godot

Um projeto na Godot é composto por uma árvore de nós. Esses nós podem ter *scripts* atrelados a eles, que estendem as funcionalidades de um nó e fornecem interfaces que podem ser acessadas automaticamente pela *game engine* ou manualmente, através de outros *scripts*. Para alcançar dinamismo no jogo, são adicionados e removidos nós da árvore, e seus atributos são modificados através de *scripts* antes ou durante a execução do jogo.

O jogo em questão tem dois nós principais: o Menu Principal (MainMenu) e o Gerenciador de Níveis (LevelManager).

O Menu Principal é responsável por apresentar os primeiros botões ao Jogador, antes de iniciar a partida, e por requisitar a conexão com o servidor.

Figura 4 - Lista de nós da Godot referente ao menu principal



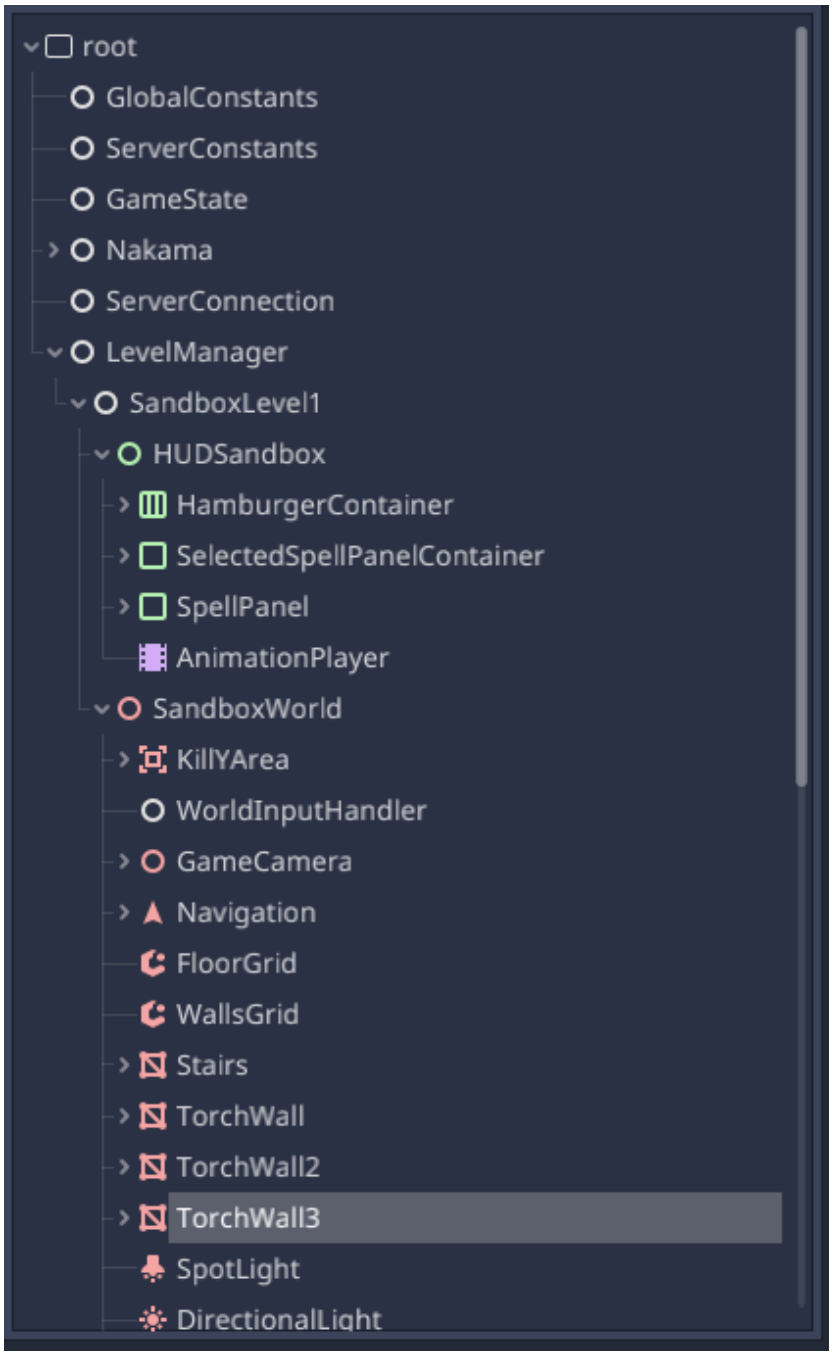
Fonte: Elaborado pelos autores

Por meio do *script* “ServerConnection.gd”, se estabelece a comunicação com o servidor. É ele que envia e recebe as mensagens e as interpreta e transforma para os formatos apropriados.

Depois que a conexão é estabelecida, o nó do Menu é substituído pelo nó do Gerenciador de Níveis, responsável por alternar entre os níveis do jogo na ordem correta e nos momentos apropriados. Cada Nível (Level) contém um Mundo (World), que tem nós aqui separados em quatro categorias: decorações, interativos, funcionais e de personagens. Os nós interativos são os objetos com os quais o Jogador pode interagir, tais quais os Portões e Botões. Os funcionais têm papel vital na operação do

jogo. São o Chão onde o personagem pisa (FloorGrid), as Paredes (WallsGrid), a Câmera do Jogo (GameCamera), e o Gerenciador de Interações do Usuário (WorldInputHandler). Por último, os personagens são compostos por uma forma de colisão, um objeto visível que representa o Jogador em si, um reprodutor de animações e um nó que contém a lógica dos movimentos e possíveis ações que o personagem pode tomar.

Figura 5 - Lista de nós do Godot referente ao nível do Sandbox

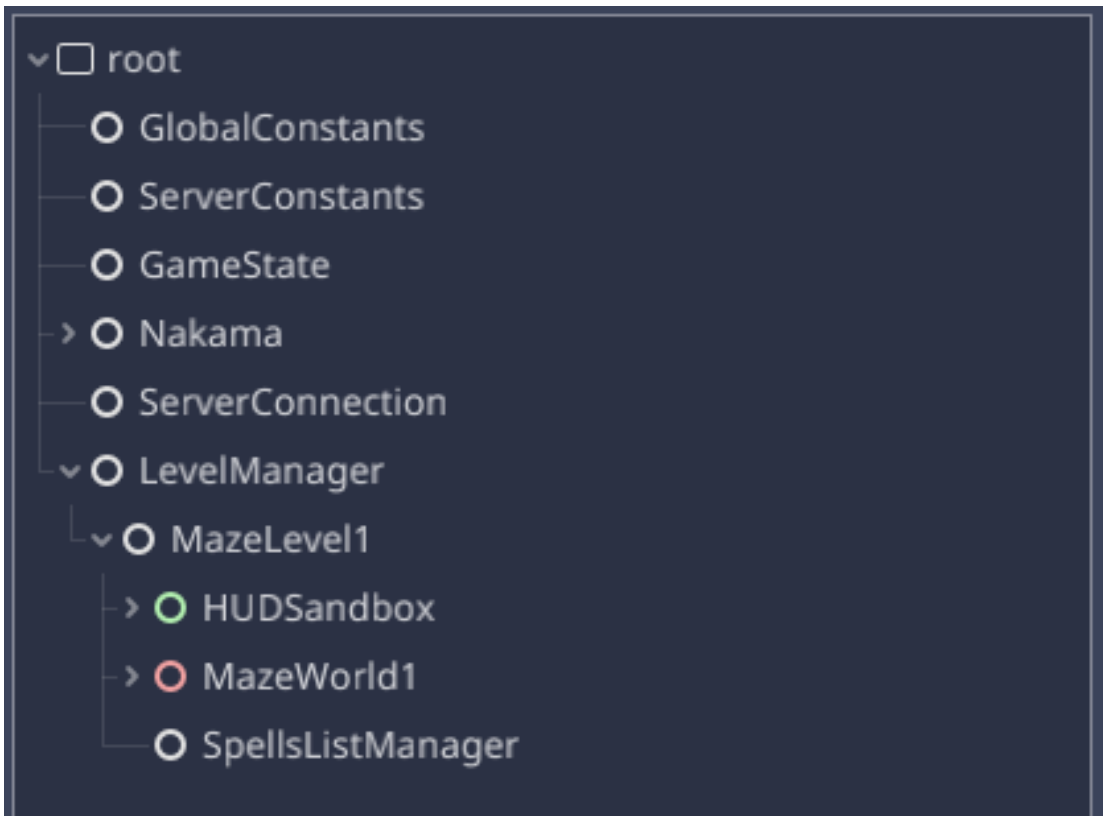


Fonte: Elaborado pelos autores

O Estado do Jogo, que contém, por exemplo, a cor do Jogador e os nomes dos Feitiços, é guardado de forma persistente no *script* “GameState.gd”.

Para que o Jogador interaja com o jogo, deve haver uma interface entre os dois. No projeto, foi criado um nó chamado HUD (do inglês *Heads Up Display*), que contém botões e caixas de texto estáticas e editáveis. O HUD, em ambos os Modos do jogo, é composto pelos painéis esquerdo e direito e pelo *pop-up* central. O *pop-up* central serve de informação de retorno ao usuário do que está acontecendo no jogo e de dicas do que se pode fazer. O painel esquerdo contém a Lista de Feitiços disponíveis – que no Modo de Criação podem ser renomeados e executados, e no Modo Labirinto podem apenas ser executados – e o botão Pronto e os de Desfazer e Passar a Vez, que só aparecem no Modo Labirinto. O painel direito possui a Fila de Feitiços a serem executados e só aparece no Modo Labirinto.

Figura 6 - Lista de nós da Godot referente ao primeiro labirinto



Fonte: Elaborado pelos autores

6.3 Ordem de priorização e implementação das funcionalidades

Durante a implementação do projeto, algumas funcionalidades receberam maior prioridade que outras, por motivos diferentes.

O primeiro a ser implementado foi o Menu Principal do jogo, por ser uma tarefa relativamente simples, e por ser um estágio de aprendizado das ferramentas. Em seguida, criamos um personagem base com a capacidade de andar quando se clicava em algum lugar do Chão. Desenvolveu-se então a primeira versão do Modo de Criação, um nível para que fosse possível ver o personagem em movimento.

Foi acrescentado o Esqueleto, a música do jogo e vários recursos gráficos tais quais o Piso e os Muros.

Posteriormente adicionou-se ao protótipo o Botão Quadrado e o Portão, para que houvesse algo mais complexo com que interagir.

Na sequência construiu-se uma interface com que o jogador pudesse interagir para comandar os personagens, depois desenvolveu-se um Gerenciador de Níveis (LevelManager); um novo tipo de botão (Botão Redondo); as Plataformas e as Pontes; e a funcionalidade de enviar a Lista de Feitiços ao servidor e, por fim, foram criados os Portais onde os Esqueletos nascem.

Com os elementos básicos implementados, foram desenvolvidos os quatro níveis e a integração do cliente ao servidor. Ao final de tudo, implementou-se novamente as malhas de navegação utilizadas para calcular os caminhos que o personagem pode percorrer ao se movimentar.

7 RESULTADOS

Este capítulo descreve em detalhes o resultado da aplicação do método do trabalho, portanto está dividido em "Concepção" e "Produção", conforme o processo indicado por Fullerton (2014), e é finalizado com uma descrição da autoria e origem dos artefatos gráficos utilizados, em "Recursos gráficos".

7.1 Concepção

A wiki do GitHub foi o ambiente utilizado para registrar o documento de *design* do jogo contendo suas regras de funcionamento, as cenas, bem como os elementos que compõem cada nível. Como consequência disso, essa wiki serviu como guia para a implementação.

Relembrando resumidamente os principais objetivos deste, trabalho tem-se:

- Treinar a nomeação de funções de uma forma a transmitir com precisão o seu comportamento e finalidade;
- Treinar a nomeação de parâmetros de funções, de forma a descrever corretamente o que representa esse parâmetro e qual é a relação dele com a sua função;
- Abstrair os conceitos de funções e parâmetros e envolvê-los em um contexto que não intimide jogadores não familiarizados com programação.

Para garantir a validade e a utilidade desse jogo eletrônico, é necessário fazer um paralelo entre o seu funcionamento básico e os objetivos deste trabalho.

Ao iniciar uma nova partida, os Jogadores se deparam com um nível em que devem selecionar algum dos Feitiços disponíveis, que têm nomes sem sentido, e selecionar um alvo para tentar conjurá-lo. Por meio de experimentação, cada Jogador descobre o significado de cada um dos Feitiços, e deve renomeá-los para que o outro Jogador possa entender os seus efeitos e utilizá-los corretamente no Modo Labirinto.

A avaliação da eficácia dos nomes escolhidos ocorre no Modo Labirinto, quando o outro Jogador se mostra capaz ou não de resolver o desafio sabendo apenas o nome das funções disponíveis e de seus parâmetros. Essa ideia valida os dois primeiros principais objetivos do trabalho citados acima.

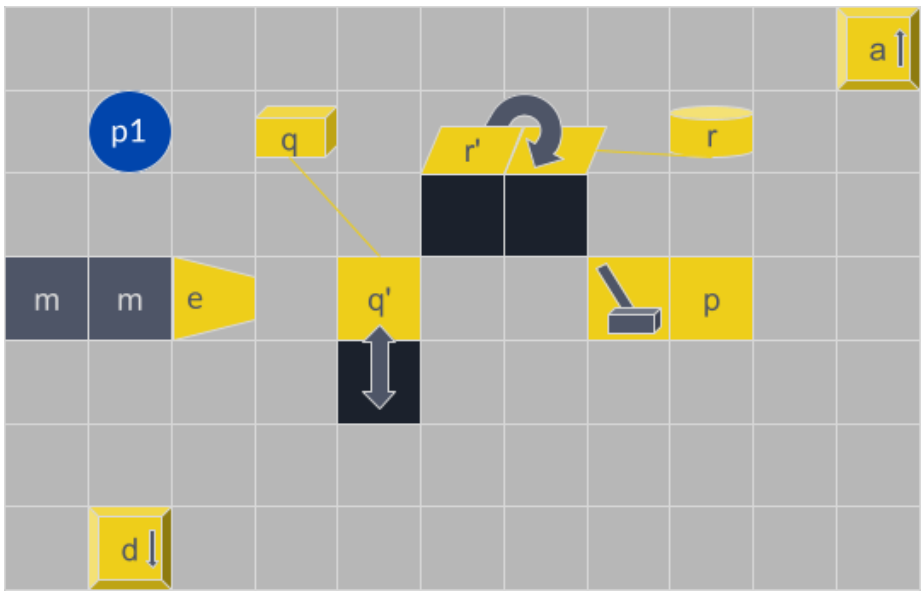
A contextualização temática do jogo fará o papel de abstrair os conceitos de função e parâmetros, tratando as funções, por exemplo, como magias criadas e proferidas pelas Magas Convocadoras, algo que acaba validando o terceiro ponto citado.

O jogo se passa alternando entre dois modos: o Modo de Criação, onde o Jogador poderá experimentar e renomear funções e o Modo Labirinto, onde o Jogador deverá solucionar *puzzles* para prosseguir na sua jornada e passar de nível.

7.1.1 Modo de Criação

O Modo de Criação é acessado no início das partidas ou no meio delas, se os dois Jogadores julgarem necessário. Nesse modo não existem *puzzles* para serem resolvidos: o mapa consiste apenas em elementos interativos espalhados aleatoriamente pelo espaço, de forma tal que o jogador pode experimentar e renomear as funções que lhe foram fornecidas. O processo de experimentação é essencial, pois essas funções não poderão ter nomes que façam sentido, para não enviesar o processo de nomeação.

Figura 7 - Esquema do mapa no modo de criação.



Fonte: Elaborado pelos autores.

O Mapa do Modo de Criação contém os seguintes elementos:

- **Muros:** indicados pela letra "m" na imagem, são elementos que bloqueiam a passagem do personagem, indicado por "p1" na imagem. Além disso, podem também servir de passarela, caso o Jogador encontre-se em cima dele.

- **Elevadores:** indicados pela letra "e" na imagem, são elementos interativos que levam o personagem que interage com elas a um nível mais elevado de chão, possibilitando, por exemplo, que esse Personagem possa andar em cima do muro.
- **Botões quadrados:** indicados pela letra "q", são elementos que podem ser pressionados gerando o deslocamento das plataformas "q' ", conforme indica a seta da figura. Um ponto importante sobre o botão é que quando o Jogador sai de cima dele, ele volta ao seu estado inicial, trazendo também consigo a plataforma que ele deslocou.
- **Botões redondos:** indicados pela letra "r", são elementos que podem ser pressionados gerando a rotação das pontes "r' ", conforme indica a seta da figura. Um ponto importante sobre o botão é que quando o Jogador sai de cima dele, ele volta ao seu estado inicial, trazendo também consigo a ponte que ele deslocou.
- **Portões:** indicados pela letra "p", são elementos que abaixam e levantam, bloqueando e liberando a passagem dos Jogadores em determinados caminhos. Eles também podem ser usados como uma forma de subir ao nível superior do chão, caso um Jogador esteja em cima do portão enquanto o outro o aciona.
- **Portais ascendentes e descendentes:** indicados respectivamente por "a" e "d", são elementos que, ao serem acionados, geram uma criatura que passará a ser controlada pelo Jogador até que esse explicitamente a desative. Os diferentes portais são meramente para estimular a capacidade do Jogador de nomear elementos sutilmente diferentes, o comportamento dos portais em si é praticamente idêntico.

7.1.2 Modo Labirinto

O principal Modo do jogo, o Modo Labirinto é onde um Jogador interage com o outro e precisa utilizar as funções nomeadas por si mesmo e pelo outro Jogador para interagir com o mapa e resolver os *puzzles* que encontrar de maneira criativa.

Os Jogadores precisam lançar Feitiços e comandar Criaturas para encontrar o caminho até o outro lado do mapa, onde se concluirá a fase. Para isso, ambos os Jogadores receberão a lista completa com todos os Feitiços nomeados por eles.

A concepção dos níveis do protótipo se deu pela definição de Peças, que quando combinadas formam o Labirinto completo. Uma Peça é um trecho de Labirinto com a finalidade de versar sobre alguma diretriz de nomeação de funções e parâmetros específica. As Peças são também subdivididas em dificuldades sequenciais. Logo a Peça "A" é dividida em "A1", "A2" e "A3", e cada um desses fragmentos será combinado com outro de outra Peça, formando assim um Nível. Foram concebidas três Peças para esse protótipo, todas elas embasadas nas diretrizes de Hilton e Hermans (2017).

A Peça "A" trata da diretriz intitulada "Describe meaning" [descrever significado]. Nessa diretriz, o autor reforça que os nomes devem descrever um conceito que possa ser reconhecido dentro daquele contexto. E que se deve evitar nomes que deliberadamente não signifiquem nada em específico, por exemplo "uma_variável". Uma segunda diretriz é a intitulada "Use a large vocabulary" [usar um léxico amplo], que aconselha usar uma única palavra mais rica de significado ao invés de usar muitas palavras com menos significado para descrever um conceito conhecido.

A Peça "B" trata da diretriz intitulada "Choose concrete words" [escolher palavras concretas], que explica que se deve usar palavras que têm um significado único e claro, dentro daquele contexto.

A Peça "C" trata novamente da diretriz "Choose concrete words", mas também trata da intitulada "Be precise" [ser específico], que aconselha a identificação de um tipo de informação específica e o seu propósito para que se possa identificar corretamente o objetivo daquela função.

7.2 Produção

A etapa de produção se apropriou da documentação obtida na concepção, gerando como resultado um primeiro protótipo do jogo, capaz de provar alguns conceitos levantados nos objetivos deste trabalho. As principais características do protótipo foram o "Modo de Criação", as "Peças": "A", "B" e "C" e os "Labirintos".

7.2.1 Modo de Criação

Os níveis do Modo de Criação apresentam os Feitiços que serão utilizados nos desafios que eles precedem, assim como os objetos-alvo desses Feitiços. Esses níveis são o espaço que os Jogadores têm para experimentar as funções a fim de descobrir os seus efeitos e renomeá-las de maneira adequada.

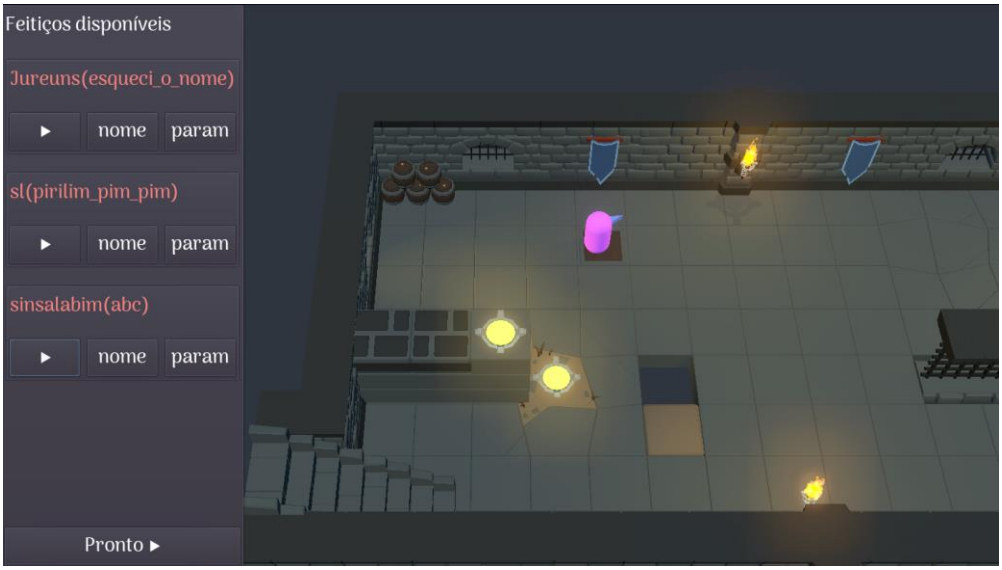
O nível é acessado antes de cada Labirinto ou quando os Jogadores falham em completá-los, oferecendo-lhes uma oportunidade para renomear as funções e seus parâmetros.

Figura 8 - Captura de tela da ferramenta de desenvolvimento Godot do mapa no Modo de Criação.

Fonte: Elaborado pelos autores.

A interface gráfica do Modo de Criação conta com um botão no canto inferior esquerdo chamado "Pronto", que deve ser pressionado quando o Jogador já houver terminado de experimentar e renomear os Feitiços em questão; e uma lista de Feitiços a serem experimentados e renomeados.

Figura 9 – Captura de tela do jogo no Modo de Criação.



Fonte: Elaborado pelos autores

No processo de renomeação do Modo de Criação, o Jogador recebe uma lista de funções, mostrada no lado esquerdo da tela, com nomes não adequados, e deverá testar cada uma para descobrir o que ela faz e assim renomeá-la de forma correta. Vemos que abaixo de cada função há três botões: o primeiro à esquerda é o que executa a função, o segundo irá pedir para o Jogador digitar o novo nome da função, e o terceiro, de forma semelhante ao segundo, pede para o Jogador digitar o novo nome do parâmetro da função.

Figura 10 – Captura de tela de um exemplo de botões de renomear função e parâmetro e de executar função.

Fonte: Elaborado pelos autores

Quando o jogador seleciona algum Feitiço para testá-lo, dois mecanismos de *feedback* entram em ação: O primeiro é um *pop-up* no topo central da interface pedindo para o Jogador selecionar o parâmetro adequado para aquele Feitiço, de acordo com o nome atual do parâmetro.

Figura 11 – Pop-up no topo central pedindo para selecionar o parâmetro.

Fonte: Elaborado pelos autores

O segundo mecanismo é um destaque verde e semi-transparente que aparece sobrepondo os objetos passíveis de interação, como por exemplo o Chão, caso o Feitiço selecionado seja o que move o personagem.

Figura 12 – Captura de tela do jogo, ressaltando o destaque em verde para auxiliar o jogador a interpretar o parâmetro adequado.

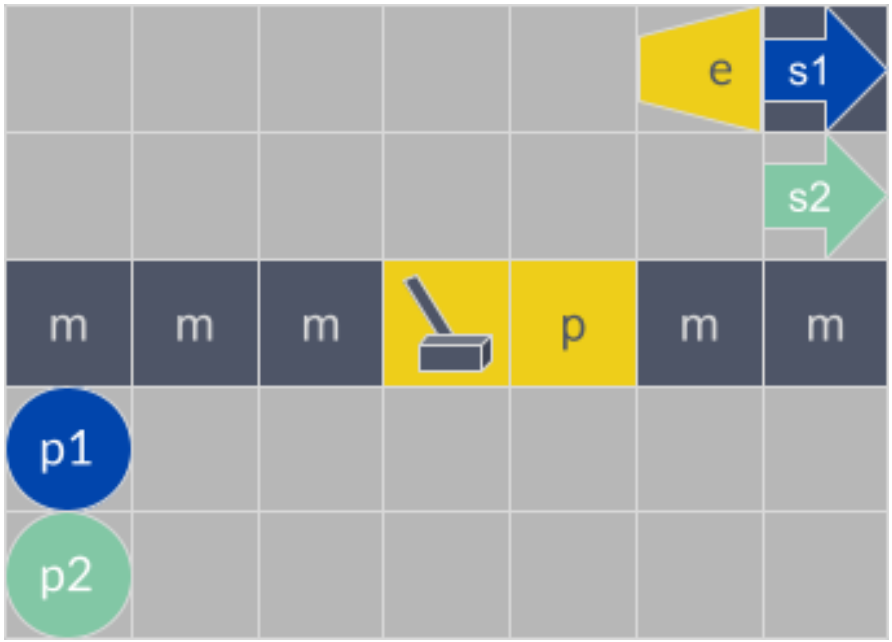
Fonte: Elaborado pelos autores

7.2.2 Peça A

Esta primeira Peça adiciona ao jogo dois tipos de objetos com os quais se pode interagir, o Portão e o Elevador. O Elevador tem a mesma função em todas as Peças, enquanto o Portão tem funcionalidades múltiplas e apresenta desafios diferentes aos jogadores, pois quebra a expectativa de como utilizá-lo.

No fragmento "A1", o Portão serve apenas como uma barreira ao jogador, que deve ativar a alavanca para transpor o obstáculo. Esse fragmento introduz o objeto do Portão, que é o foco dessa Peça. A expectativa nesse fragmento é de que o Jogador dê um nome à função que interage com o Portão sem considerar que essa função fecha o Portão além de abri-lo, pois a única utilidade apresentada nesse fragmento é a de abrir. Logo, espera-se algum nome como "abrir_portão(portão)", ou então "abrir(portão)". Essa imprecisão será explorada com mais detalhes no fragmento "A2".

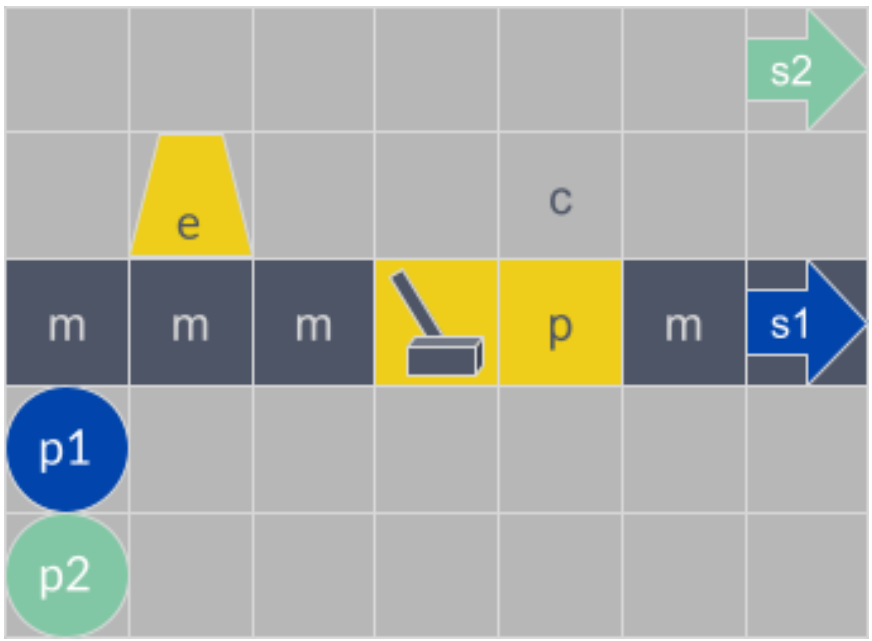
Figura 13 – Esquema do fragmento A1.



Fonte: Elaborado pelos autores

No fragmento "A2", os Jogadores precisam caminhar por cima do Portão para completar o nível. Sendo assim, é necessário que se feche o Portão após atravessá-lo. Para isso, eles devem perceber que o Portão pode ser aberto e fechado também, e devem nomear o Feitiço correspondente de acordo com esse comportamento. Ou seja, espera-se algum nome como, por exemplo, "abrir_fechar(portão)".

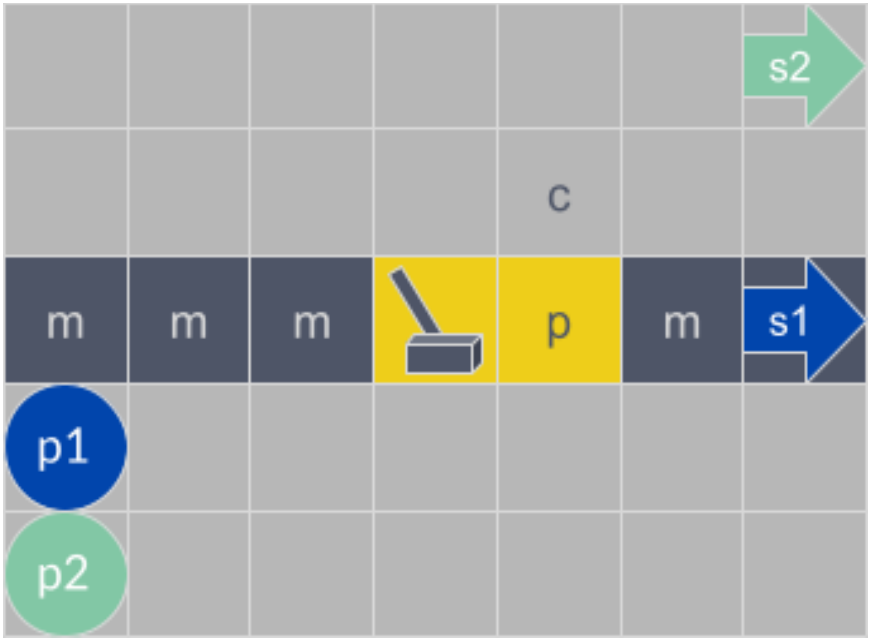
Figura 14 – Esquema do fragmento A2.



Fonte: Elaborado pelos autores

O fragmento A3 requer dos Jogadores mais uma extrapolação do funcionamento do Portão. Para resolver o desafio proposto, os Jogadores devem abaixar o Portão e um deles deve se movimentar para cima do Portão e então o outro deve ativar o Portão de novo. Isso faz com que o Jogador que estava parado seja elevado enquanto o Portão sobe e permite a ele acessar o caminho do nível superior.

Figura 15 – Esquema do fragmento A2.



Fonte: Elaborado pelos autores

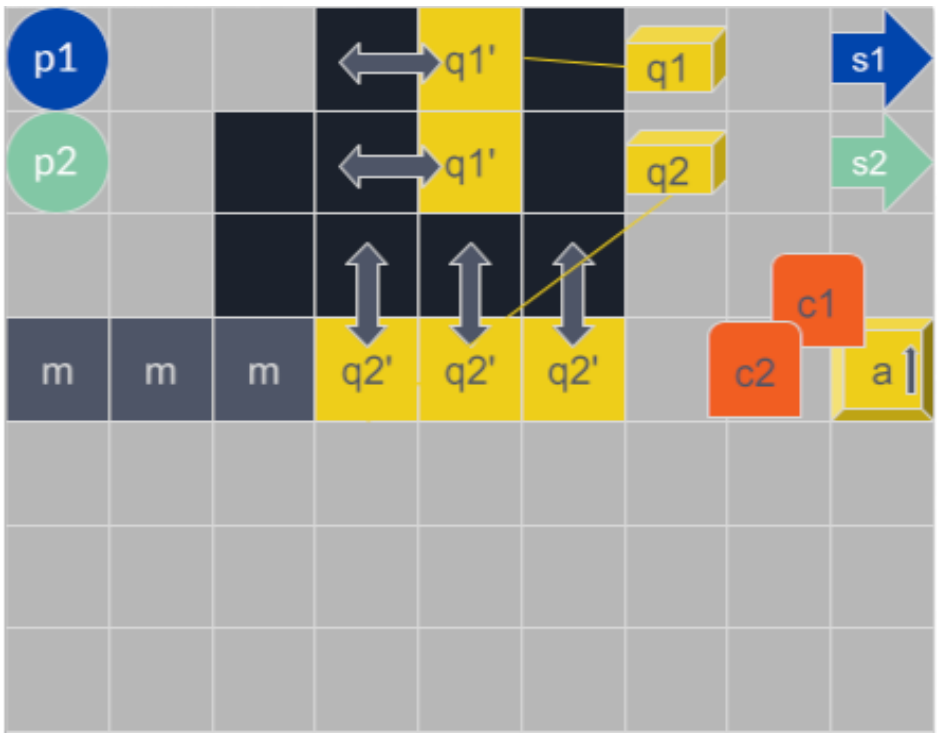
Caso os Jogadores tenham realmente dado o nome "abrir(portão)", ou algo próximo disso, para o Feitiço de interação com o Portão no primeiro fragmento eles teriam, de certa forma, violado a diretriz "Describe meaning", pois não está corretamente descrito neste nome o significado da interação, algo que fica claro no fragmento seguinte, quando surge a necessidade de fechar o Portão além de abri-lo. Uma outra diretriz que poderia ser aplicada pelo Jogador nessa Peça é a "Use a large vocabulary", pois o nome do Feitiço poderia incluir mais vocabulário significativo, como por exemplo a palavra "alternar" para indicar que a ação de abrir e fechar ocorre de forma alternada. Um exemplo de nome interessante seria "alternar_abrir_ou_fechar(portão)", ou mais precisamente ainda, contemplando a funcionalidade do terceiro fragmento, "alternar_levantar_abaixar(portão)".

7.2.3 Peça B

A Peça B adiciona ao jogo mais dois tipos de objeto com os quais se pode interagir, os Portais com Partículas Ascendentes e Descendentes. Há um Feitiço específico para interagir com cada um dos dois, e os Jogadores devem nomeá-los de maneira descritiva o suficiente para que possam diferenciá-los. Os Portais marcam o local onde um personagem novo, o Esqueleto, irá aparecer. Quando um Portal é ativado, a recém-invocada Criatura passa a ser a agente de todos os outros Feitiços, até que um Feitiço específico seja conjurado para desativá-la.

No fragmento "B1", o Portal Ascendente é apresentado e as Criaturas, indicadas por "C1" e "C2", passam a ser indispensáveis para a conclusão do Labirinto. A esta altura, apenas o Portal com Partículas Ascendentes foi apresentado, induzindo o Jogador a nomear esta interação "gerar_esqueleto(portal)", ou algo similar.

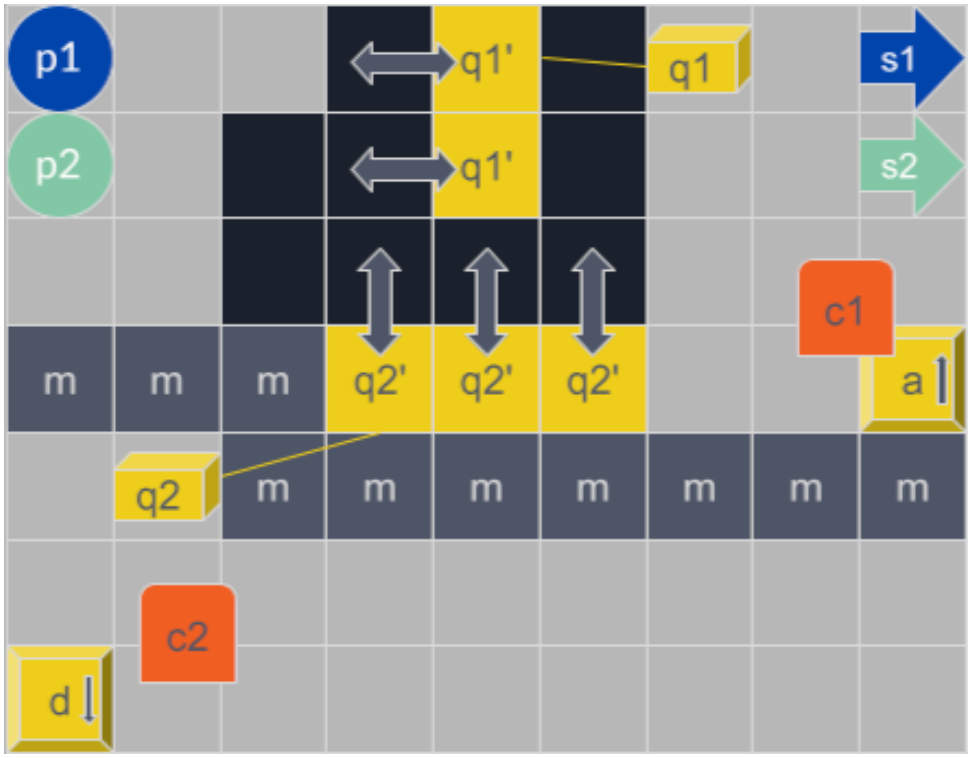
Figura 16 – Esquema do fragmento B1.



Fonte: Elaborado pelos autores

No segundo e último fragmento dessa Peça, é introduzido o Portal com Partículas Descendentes, gerando assim a necessidade de nomear dois Feitiços muito similares de forma claramente diferente. Nesse ponto, o jogo procura estimular a criatividade do Jogador, pois a diferença física entre os dois Portais é muito sutil – um tem Partículas que sobem e o outro tem Partículas que descem.

Figura 17 – Portal com partículas descendentes.



Fonte: Elaborado pelos autores

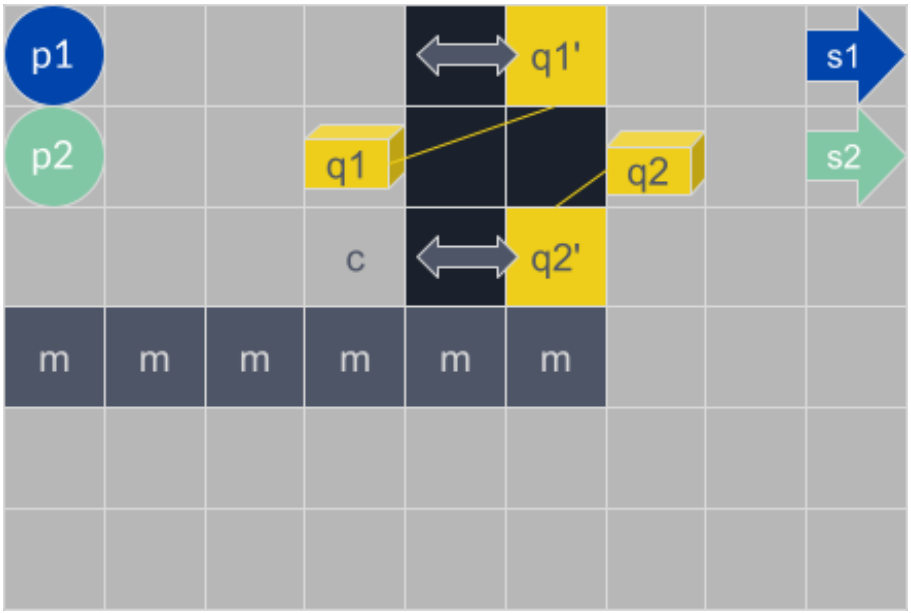
Essa situação trata da diretriz "Choose concrete words", pois a melhor maneira de deixar claro qual Feitiço interage com qual Portal é encontrar uma palavra concreta para diferenciá-los. O ideal para essa situação seria algo que descrevesse o comportamento das Partículas, que é a única diferença concreta entre os Portais.

7.2.4 Peça C

A Peça C trata dos dois tipos de botões com os quais se pode interagir, os Botões Quadrado e Redondo. Há um Feitiço específico para interagir com cada um dos dois e os Jogadores devem nomeá-los de maneira descritiva o suficiente para que possam diferenciá-los. Cada tipo de botão tem um efeito diferente no mapa. O Botão Quadrado move Plataformas Deslizantes e carrega os personagens enquanto elas se movem. O Redondo abaixa as Pontes. Os Botões ficam ativos só enquanto os personagens estão em cima deles.

Novamente, no primeiro fragmento, procura-se induzir que o Jogador dê um nome não muito preciso, apresentando apenas um tipo de Botão. O esperado é um nome similar a "pressionar(botão)".

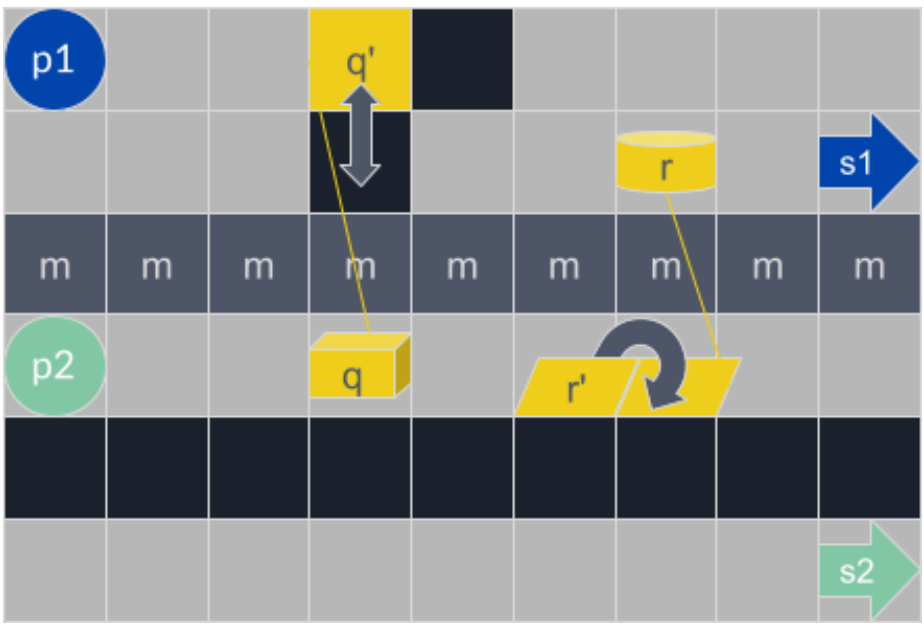
Figura 18 – Botões Quadrados acionam Plataformas Deslizantes. Enquanto deslizam, carregam os personagens.



Fonte: Elaborado pelos autores

No fragmento seguinte, o "C2", é introduzido o próximo tipo de Botão que, diferentemente dos Portais, tem uma diferença muito clara quando comparado ao primeiro tipo.

Figura 19 – Botões Redondos acionam pontes.



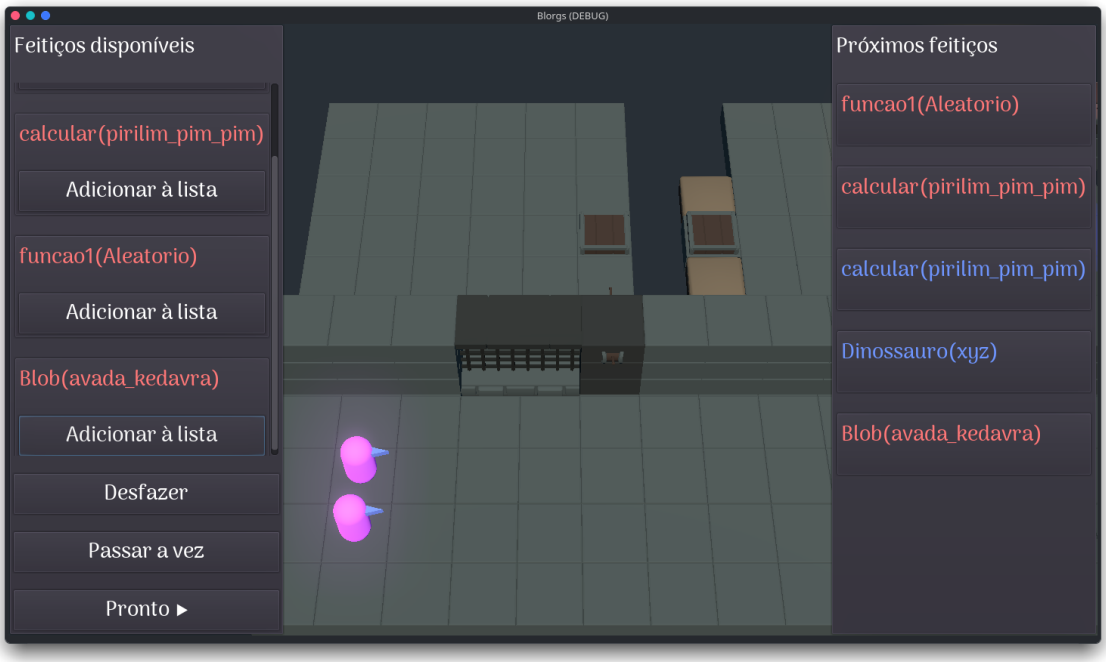
Fonte: Elaborado pelos autores

A resposta para essa Peça é simplesmente nomear cada um dos Feitiços de forma precisa e concreta, reforçando diretamente as diretrizes "Be precise" e "Choose concrete words". Algo como "pressionar(botão_quadrado)" e "pressionar(botão_redondo)", por exemplo.

7.2.5 Labirintos

A interface gráfica dos Labirintos é composta principalmente por dois painéis, um à direita e outro à esquerda.

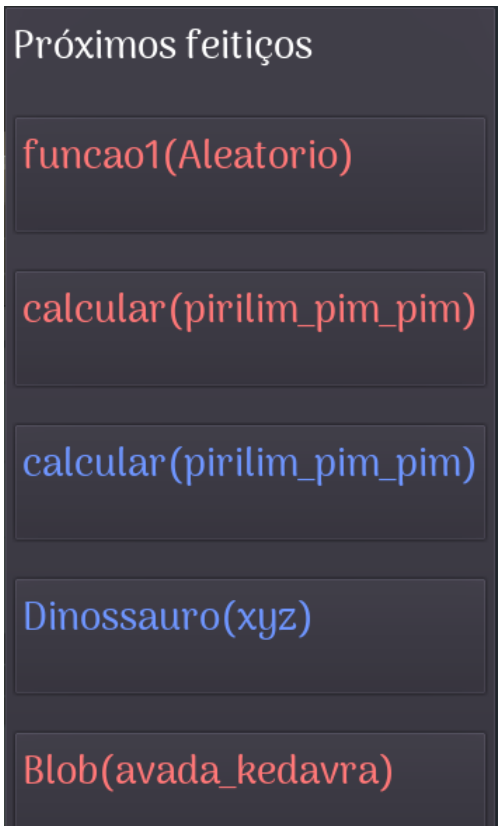
Figura 20 – Interface gráfica do Modo Labirinto.



Fonte: Elaborado pelos autores

O painel direito contém a Lista de Feitiços a serem executados. Os Jogadores adicionam Feitiços ao final da fila durante os seus turnos. Quando ambos indicarem que estão prontos, os Feitiços são executados de acordo com a ordem indicada no painel, enquanto tocam as animações correspondentes.

Figura 21 – Fila dos próximos Feitiços.



Fonte: Elaborado pelos autores

Assim como no Modo de Criação, o painel da esquerda apresenta os Feitiços disponíveis aos Jogadores, com a diferença de que não há botão para renomeá-los, apenas para adicioná-los à lista dos próximos Feitiços. Nesse painel, encontram-se também botões para encerrar o turno do Jogador, para indicar que o Jogador está pronto para a execução da Fila de Feitiços e para desfazer a sua última adição à Fila.

Figura 22 – Feitiços disponíveis.

Fonte: Elaborado pelos autores

Figura 23 – Desfazer, Passar a vez e Pronto.

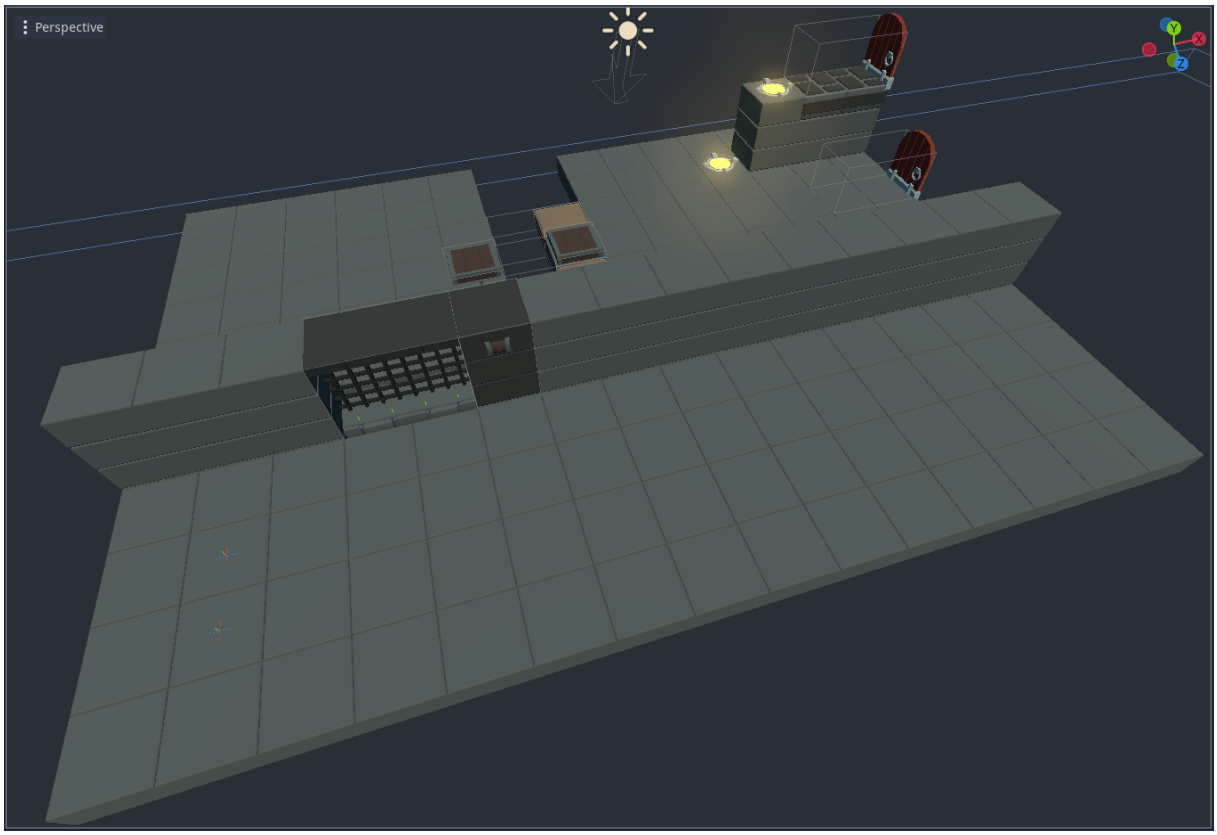
Fonte: Elaborado pelos autores

7.2.6 Labirinto 1

No primeiro nível do protótipo apresentado, os Jogadores devem cooperar para chegar às saídas. Ele é composto pelos fragmentos A1 e C1.

O primeiro Jogador abaixa o Portão e ativa o primeiro Botão Quadrado para que o segundo cruze o abismo e acione o segundo Botão Quadrado para permitir a travessia de seu parceiro.

Figura 24 – Captura de tela do primeiro Labirinto.



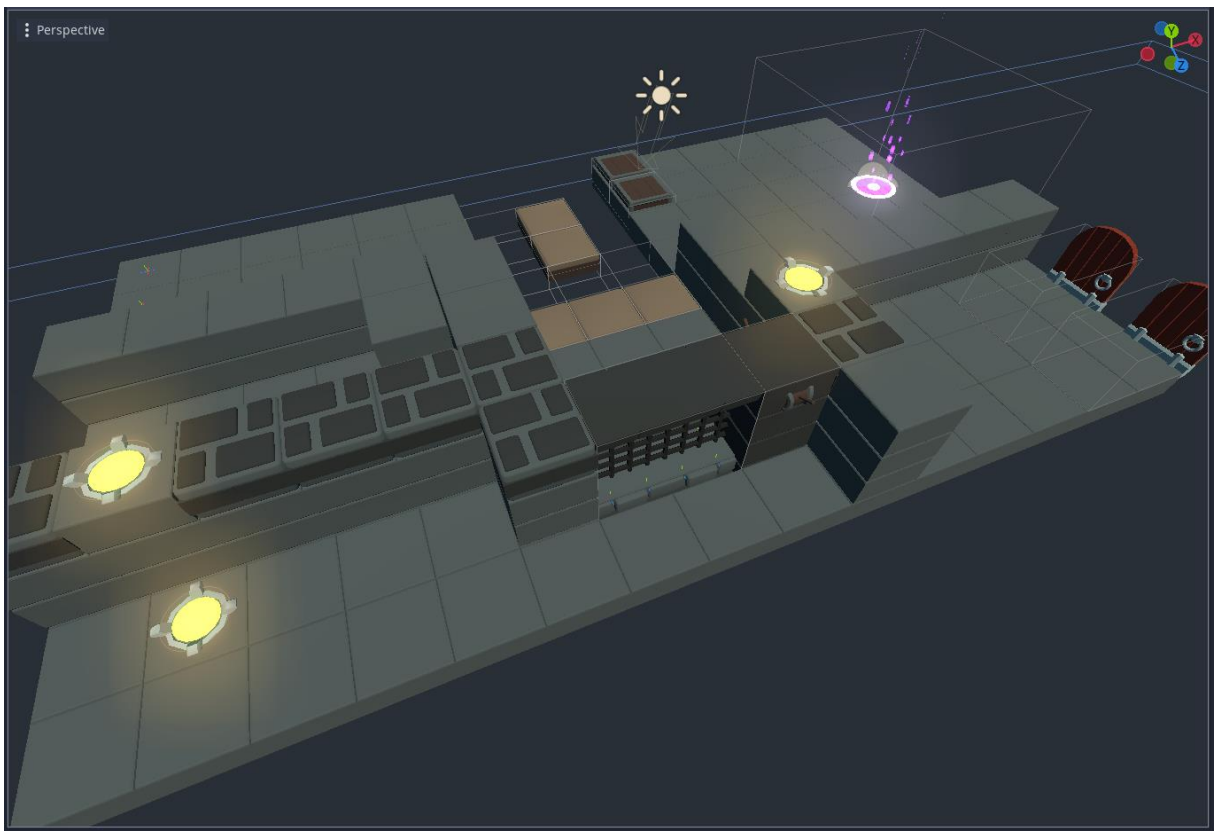
Fonte: Elaborado pelos autores

7.2.7 Labirinto 2

Este segundo Labirinto, composto pelos fragmentos B1 e A2, introduz a necessidade de fechar o Portão novamente. Para completar o desafio, os jogadores devem renomear a função de interagir com o Portão para indicar que ela também fecha o Portão.

Os Jogadores precisam comandar cada um o seu Esqueleto até os dois Botões Quadrados diferentes e desmontá-los em cima dos Botões. Então, precisam recriar os Esqueletos quando estiverem em cima das Plataformas para que os Botões se desativem, fazendo a Plataforma voltar para a posição inicial. Um Jogador abre o Portão e sobe o Elevador enquanto o outro fecha o Portão para que os dois andem por cima dele.

Figura 25 – Captura de tela do segundo Labirinto.



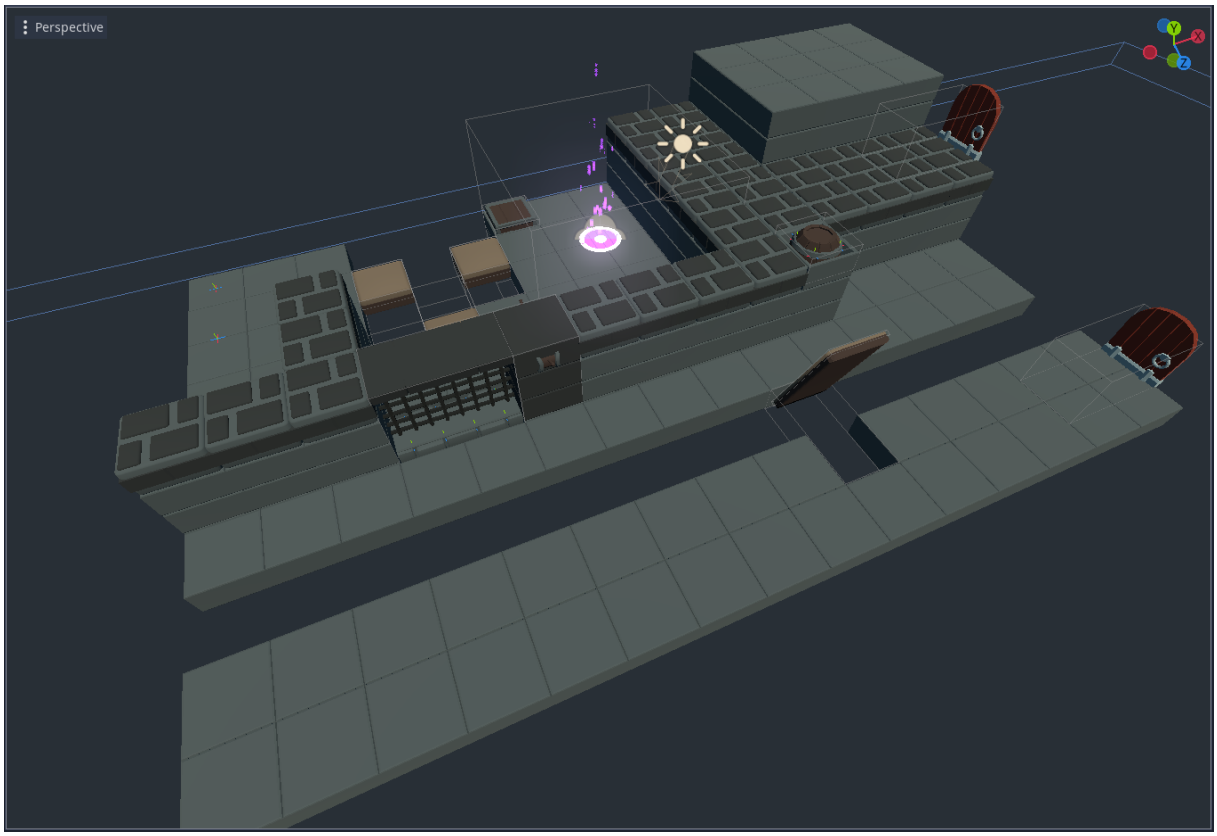
Fonte: Elaborado pelos autores

7.2.8 Labirinto 3

Para completar o terceiro mapa, os Jogadores devem explorar o fato de que o Portão tem uma superfície que pode transportar o Jogador assim como as Plataformas Deslizantes. Ele é composto pelos fragmentos A2 e C2.

Um Jogador precisa ficar em cima do Portão enquanto o outro o ativa e faz com que o primeiro Jogador seja elevado com a subida do Portão e alcance o nível de cima, onde ativa o Botão Redondo, permitindo que o segundo Jogador alcance a sua saída.

Figura 26 – Captura de tela do terceiro Labirinto.



Fonte: Elaborado pelos autores

7.2.9 Labirinto 4

No quarto desafio, que é composto pelos fragmentos B2 e C3, há a introdução dos Portais com Partículas Descendentes, que são diferentes dos Portais com Partículas Ascendentes dos níveis anteriores e têm Feitiços específicos para eles.

Os Jogadores devem nomear duas funções diferentes, específicas para cada um dos dois tipos de Portal, e se organizarem para permitir a passagem das Criaturas através das Plataformas Móveis.

Figura 27 – Captura de tela do quarto Labirinto.

Fonte: Elaborado pelos autores

7.3 Recursos gráficos

O conceito visual do jogo é RPG (Role Playing Game [jogo narrativo]), e a aparência dos elementos do jogo são típicos do gênero. Todos os recursos gráficos para criar esse conceito vieram de duas fontes distintas. Uma delas foi Mayara Cristina dos Santos Gomes, que modelou ambas as personagens principais durante a disciplina PCS3539 – Tecnologia de Computação Gráfica e as disponibilizou para serem usadas no presente trabalho; e a outra fonte foi Kay Lousberg, que disponibilizou os artefatos gráficos sob a licença de *Creative Commons, Public Domain Dedication* (CC0).

7.3.1 Personagens principais

As duas personagens principais são uma Necromante, que é um tipo de maga que ressuscita mortos, e uma Elfa, que é um tipo de maga que dá vida a objetos inanimados como pedras e árvores, por exemplo. O planejamento inicial do projeto previa esses dois tipos de magas, sendo que cada uma delas traria à vida tipos diferentes de criaturas. Por limitações de tempo e decisões de foco do projeto, ambas as personagens acabaram por invocar o mesmo tipo de criatura, um esqueleto.

Necromante: esta personagem enaltece a morte, podendo fazer magias fúnebres. Sua modelagem consiste numa capa que a torna mais misteriosa. A paleta de cores escuras foi utilizada para se adequar à proposta. Abaixo, observa-se a Necromante com acessórios como um colar de rubi e um cajado de madeira rústica com uma caveira.

Figura 28 – Captura de tela da Necromante.

Fonte: Mayara Cristina dos Santos Gomes,

Elfa: esta personagem enaltece a vida, opostamente à Necromante, e tem afinidade com a natureza, as árvores e os animais. Na sua modelagem, características como orelhas pontudas e paleta de cores claras foram exploradas. Abaixo, pode-se observar a Elfa com acessórios como uma tiara de ouro e um cajado de madeira esculpida.

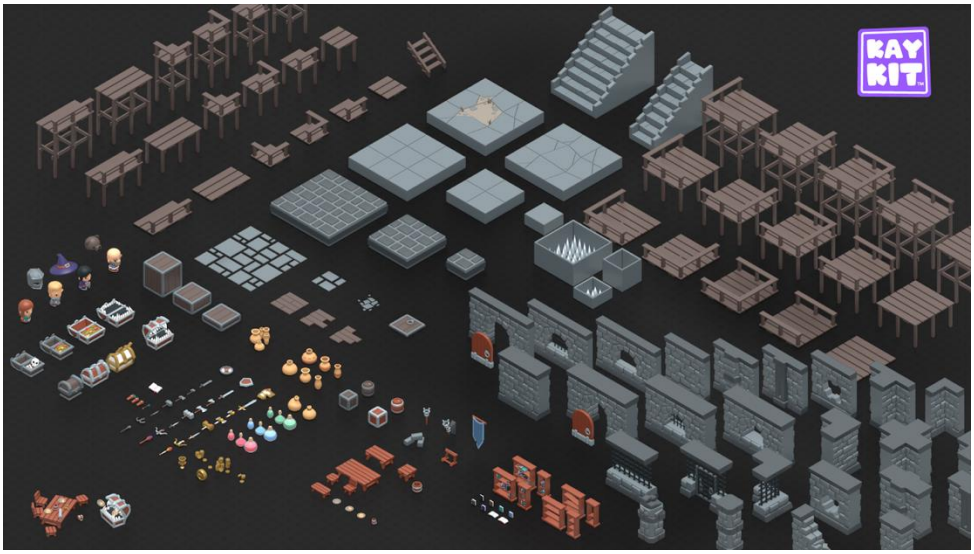
Figura 29 – Captura de tela da Elfa.

Fonte: Mayara Cristina dos Santos Gomes.

7.3.2 Recursos adquiridos de Kay Lousberg

Todos os outros recursos gráficos deste protótipo foram adquiridos por meio do site https://opengameart.org, mais especificamente dois pacotes de recursos disponibilizados livremente por Kay Lousberg –, um chamado "*Dungeon Asset Pack*" e outro chamado "*Spooktober Seasonal Pack*".

Figura 30 – Elementos gráficos presentes no pacote "Dungeon Asset Pack".



Fonte Open Game Arte, disponível em: <https://opengameart.org/sites/default/files/dungeon-pack_overview.png>. Acesso em: 19 dez. 2021.

Figura 31 – Elementos gráficos presentes no pacote "Spooktober Seasonal Pack".

Fonte Open Game Arte, disponível em: <https://opengameart.org/sites/default/files/spooktoberitchcover.png>. Acesso em 19 dez. 2021

Esqueleto: este personagem é a Criatura controlada pelos jogadores.

Figura 32 – Captura de tela do modelo 3D do Esqueleto.

Fonte: Kay Lousberg

8. PLANO DE TESTES

8.1 Plano de testes

Para validar as hipóteses levantadas para o público-alvo e testar a interface desenvolvida, foram planejados testes e entrevistas. No entanto, não foram aplicados, pois o Termo de Consentimento Livre e Esclarecido teria de passar por um comitê de ética e não havia tempo hábil para a avaliação do comitê antes da entrega deste trabalho. A entrevista e os testes foram pensados em outro momento do desenvolvimento do projeto e não necessariamente refletem a realidade atual dele. Ainda assim, possuem aspectos que poderiam ser aproveitados para validar uma versão mais atual do jogo.

8.1.1 Entrevista

A entrevista tem o objetivo de validar algumas hipóteses sobre as personas selecionadas como representantes do nosso público-alvo e verificar a compatibilidade dos perfis com o jogo. Também foi planejada para coletar outras informações sobre os possíveis usuários, como a plataforma que geralmente usam para jogar.

O roteiro traz perguntas sobre os hábitos de jogo da pessoa entrevistada, sobre sua familiaridade e opinião sobre jogos educativos e se ela tem dificuldades em comunicar ideias ou em descrever tarefas e processos em ambiente de trabalho.

8.1.2 Teste de usabilidade

Os **objetivos** do teste de usabilidade, que se encontra no anexo A, são: validação do entendimento da dinâmica do jogo no modo de uso de funções existentes, validação do entendimento da interface apresentada aos participantes e verificação da aceitação/atratividade do mesmo modo de jogo pelos participantes.

As **responsabilidades dos participantes** são: completar um conjunto de tarefas em cenários representativos do uso real da interface, apresentadas a eles de forma ágil e eficiente, e prover feedback com respeito à usabilidade e à aceitação da interface de usuário. Os participantes foram instruídos a fornecer opiniões sinceras sobre a usabilidade da aplicação e a participar dos questionários e entrevistas pós-testes.

O **treinamento** dos participantes reduz-se a explicar as tarefas a serem resolvidas, já que o objetivo é testar o desempenho de um usuário que não possui nenhuma informação sobre a utilização do jogo.

O que se pede no **primeiro cenário** de teste é que o jogador movimente o personagem até a bandeira na esquerda superior do nível e depois mova até o barril na direita inferior.

No **segundo cenário**, o que deve ser feito é: movimentar o personagem até o barril na esquerda, pulando por cima de um obstáculo no meio do caminho. Em seguida, deve-se movimentar o personagem para o ponto de partida, precisando novamente pular sobre o obstáculo.

8.1.3 Termo de Consentimento Livre e Esclarecido:

Os Termos escritos para o teste e para a entrevista informam os participantes do objetivo da atividade, dos procedimentos que ocorreram e garantem privacidade e sigilo e isentam as pessoas de qualquer custo.

8.2 Testes informais

Próximo à fase final de desenvolvimento do projeto, amigos e familiares se voluntariaram para testar o jogo. Os comentários feitos foram sobre a interface, a usabilidade e sobre a dificuldade dos desafios.

Com esses testes informais, vieram à tona inconsistências na interface do jogo. Por exemplo, alguns botões se comportavam de forma diferente de outros botões, o que confundiu os jogadores.

Houve um caso em que o jogador não recebeu instruções suficientes para nomear os feitiços da forma que era esperado e, depois de ter entendido os comportamentos das funções e ter reiniciado o jogo por um problema técnico, renomeou os feitiços para nomes parecidos com os apresentados anteriormente. Isso não gerou confusão para nenhuma das partes, mas gerou descontração durante o teste e teve um resultado positivo e cômico.

Em boa parte dos testes, entendeu-se facilmente o objetivo, mas os jogadores se frustraram com o comportamento contra intuitivo e com a falta de fluidez da interface. Além disso, todos os jogadores tiveram uma certa dificuldade com o *puzzle* em si, o que já era esperado, pois os níveis prototipados são representativos de desafios de um momento intermediário do jogo.

Todas os voluntários deram bastante valor à parte gráfica, às animações, às partículas e às texturas.

No geral, percebeu-se que as pessoas se divertiram e que causou uma boa impressão. O jogo, por ser cooperativo, causou frustração quando os jogadores não conseguiam se comunicar e por ter nomes sem significado aparente para as pessoas trouxe um aspecto cômico ao trabalho.

Apesar das dificuldades, todos resolveram os desafios e notou-se que, durante as iterações, que se alternam entre os modos de criação e de desafio, os voluntários modificaram aos poucos os nomes que deram aos feitiços de forma que ficaram mais descritivos. Sendo assim, pudemos concluir que houve uma melhoria na habilidade que se quer estimular com o jogo.

9. CONCLUSÃO

O presente trabalho se propôs a desenvolver um jogo digital que servisse como um ambiente lúdico e descontraído para treinar habilidades de nomeação de funções e parâmetros. O protótipo resultante cumpriu todos os requisitos mínimos esperados e se mostrou uma boa prova de conceito. Testes informais foram realizados e com eles foi possível observar um forte estímulo ao cumprimento das diretrizes propostas para cada Peça, tal como previsto. O uso da cooperação como forma de validação para a adequação dos nomes também foi satisfatório. Afinal, se a dupla de jogadores foi capaz de resolver os desafios a partir de um determinado conjunto de nomes, é plausível concluir que esses são nomes suficientemente adequados. Complementando as diretrizes formais, o jogo estimula a prática de nomear funções e parâmetros dentro de um contexto genérico.

O projeto, em suas primeiras concepções, previa um protótipo mais voltado para a composição de funções e para a escolha dos tipos de parâmetros adequados para elas. Essa ideia se perdeu com o tempo e foi substituída por uma abordagem mais focada na renomeação de funções já definidas, por questões de redução de complexidade do projeto. No fim, essa substituição trouxe um foco mais interessante para o trabalho, de forma a criar uma contribuição realmente nova para a área. O projeto passou a tratar menos do exercício de raciocínio lógico, muito presente em jogos digitais voltados ao ensino de programação, e mais da prática de nomear adequadamente funções e parâmetros, algo que já não é tão presente.

Um outro aspecto do projeto que se perdeu, mas que nesse caso seria desejável para o futuro, foi a ambientação temática do jogo. A complexidade de implementação do projeto foi subestimada durante a etapa de concepção e muito se planejou com relação ao tema e aos recursos gráficos. Muito disso, infelizmente, se perdeu devido ao tempo e aos recursos limitados.

Trabalhos futuros poderiam contemplar melhor a questão da temática e trazer Peças com diretrizes mais diversas, que também abordem outros tipos de identificadores de elementos da programação, como por exemplo nomes de variáveis e classes. Um jogo, por excelência, precisa ser divertido e, nesse caso em particular, precisaria ser mais capaz de envolver o jogador em sua fantasia. Por isso fica o desejo de uma nova versão que contemple mais essa questão e que traga desafios complexos e diversos.

REFERÊNCIAS

BLINMAN, S.; COCKBURN, A. **Program Comprehension: Investigating the Effects of Naming Style and Documentation**. ACM International Conference Proceeding Series, 2005.

FULLERTON, T. **Game Design Workshop**: A Playcentric Approach to Creating Innovative Games. AK Peters/CRC Press, 2014.

HILTON, P.; HERMANS, F. **Naming Guidelines for Professional Programmers**. PPIG 2017: 28th Annual Workshop, Delft, Netherlands, 2017.

KAVANAGH, P. B. **Open Source Software**: Implementation and Management. Elsevier, pp. 321-322, 2004.

LAMB, R. L.; ANNETTA, L.; FIRESTONE, J.; ETOPIO, E. **A meta-analysis with examination of moderators of student cognition, affect, and learning outcomes while using serious educational games, serious games, and simulations, Computers in Human Behavior**, Volume 80, Pages 158-167, 2018.

MARTIN, R. C. **Clean Code**: A Handbook of Agile Software Craftsmanship. Prentice Hall, 2009.

MCCONNELL, S**. Code Complete**: a practical handbook of software construction. Microsoft Press, 1993.

MIRANDA, F. S.; STADZISZ, P. **Jogo Digital**: definição do termo. SBGames 2017, 2017.

PICOLI, Ivan Luiz. **Arquitetura cliente-servidor em jogos multiplayer**. 2011. Trabalho de Conclusão de Curso. Universidade Tecnológica Federal do Paraná.

ȘTEFAN, I. A.; HAUGE, J. B.; HASSE, F.; ȘTEFAN, A. **Using Serious Games and Simulations for Teaching Co-Using Serious Games and Simulations for Teaching Co-Operative Decision-making**. Procedia Computer Science, Volume 162, Pages 745-753, 2019.

TONÉIS, C.; FRANT, J. B. **Do desenvolvimento e criação de puzzles para a produção de conhecimentos nos jogos digitais**. Teccogs: Revista Digital de Tecnologias Cognitivas. 95-114, 2015.

REFERÊNCIAS CONSULTADAS

ALDRICH, C. **Learning by doing**: a comprehensive guide to simulations, computer games, and pedagogy in e-learning and other educational experiences. San Francisco: Pfeiffer, 2005.

ANNETTA, L. A. **Serious Educational Games**: From Theory to Practice. Sense Publishers, 2008.

DARWESH, A. M. **Concepts Of Serious Game In Education**. International Journal Of Engineering And Computer Science, Volume 4 Issue 12 December, 2015 Page No. 15229-15232, 2015.

PARASKEVA, F.; MYSIRLAKI, S.; PAPAGIANNI, A. **Multiplayer online games as educational tools**: Facing new challenges in learning. Computers & Education, Volume 54, Issue 2, 2010.

VALENCIA-GARCIA, R.; et al. **Technologies and Innovation**: Second International Conference, CITI 2016, Guayaquil, Ecuador, 2016.

APÊNDICE A - Teste de usabilidade

Projeto Programação

PROJETO 2: TESTE DE USABILIDADE

Planejamento da avaliação da experiência de usuário com o projeto e os resultados da sua aplicação a um número de sete participantes

Interação Humano-Computador - PCS3573

Docente: Professora Dra. Lucia Vilela Leite Filgueiras

Beogival Wagner Lucas Santos Junior - 8992836

Gabriel Moreira Minossi - 9349346

Maria Antonia Marzo Barceló - 12258852

Pedro Felipe de Brito Oliveira - 9298750

Renan Yudi Hamada Nunes - 9832999

ÍNDICE

Plano de teste de usabilidade 3
Introdução 4
Objetivo 5
Metodologia 6
Participantes 6
Treinamento 6
Procedimento 6
Papéis 7
Tarefas de usabilidade 8
Cenário 1 8
Cenário 2 9
Métricas de usabilidade 10
Finalização dos cenários 10
Erros críticos 10
Erros não-críticos 10
Avaliações Subjetivas 10
Entrevista pós-teste 11
Resultados 11

Projeto Programação

Plano de teste de usabilidade

Versão 1

Introdução

O documento seguinte descreve um plano de teste para conduzir um teste de usabilidade no desenvolvimento de *Blorgs* (Projeto Programação). Os objetivos de testar a usabilidade da aplicação são vários: estabelecer uma avaliação do desempenho do usuário, estabelecer e validar medidas de desempenho do usuário e identificar possíveis problemas que possam surgir com a utilização do aplicativo. A resposta de todas essas questões permitirá planejar melhorias a serem abordadas a fim de melhorar a eficiência, produtividade e satisfação do usuário final.

Os objetivos do teste de usabilidade:

- Determinar inconsistências no design e problemas de usabilidade relacionados a interface do usuário. Potenciais fontes de problemas são:
    - Erros de navegação, falhas para encontrar funções, uso excessivo de teclas para cumprir uma função, falha em seguir o fluxo de ações esperada.
    - Uso inadequado das ferramentas de interação do aplicativo: uso inadequado da barra de ferramentas, dos campos de entrada...
- Criar entornos controlados de uso que serão utilizados por usuários do público alvo devidamente selecionados para esse fim, e usar os dados de avaliação do desempenho desses usuários para estabelecer medidas quantitativas na usabilidade, desempenho e eficiência do nosso aplicativo.
- Estabelecer níveis básicos de desempenho do usuário e de satisfação do usuário da interface do usuário para futuras avaliações de usabilidade.

O aplicativo está destinado a pessoas que desejem melhorar o seu desempenho no aspecto semiótico da programação, uma ferramenta muito útil para desenvolvedores e profissionais que tenham um contato direto com programação no seu dia a dia. Os usuários selecionados para os testes de usabilidade serão pessoas que cumpram com as características do público alvo definido ao início do planejamento do

projeto: que sintam essa vontade de melhorar o seu desempenho acadêmico/profissional, de faixa etária relativamente jovem e que sintam familiaridade com aplicações de videogames.

O teste de usabilidade será remoto, e se espera que dure em torno de 15 minutos por usuário.

Objetivo

Será utilizado um protótipo simples, feito com a ferramenta Godot, para simular um modo de jogo do projeto.

Os objetivos do teste de usabilidade serão os seguintes: validação do entendimento da dinâmica do jogo no modo de uso de funções existentes, validação do entendimento da interface apresentada aos participantes e verificação da aceitação/atratividade do mesmo modo de jogo pelos participantes.

Metodologia

No seguinte apartado, serão descritos o número de participantes que serão escolhidos, o planejamento das sessões de avaliação, as ferramentas que serão utilizadas para estudar essa interação entre o usuário e o aplicativo

Participantes

Planeja-se recrutar pelo menos 7 participantes para fazer parte do teste de usabilidade. Estes participantes vão ser escolhidos com base no público alvo que foi definido durante o planejamento do projeto e nas *proto-personas* criadas após as pesquisas de usuário feitas.

As responsabilidades dos participantes serão completar um conjunto de tarefas em cenários representativos do uso real da interface, apresentadas a eles de forma ágil e eficiente, e prover *feedback* com respeito à usabilidade e à aceitação da interface de usuário. Os participantes serão instruídos a fornecer opiniões sinceras sobre a usabilidade da aplicação e a participar dos questionários e entrevistas pós-testes.

Treinamento

O treinamento dos participantes vai se reduzir a explicar as tarefas a serem resolvidas, já que o objetivo é testar o desempenho de um usuário que não possui nenhuma informação sobre a utilização do jogo.

Procedimento

O teste vai ter lugar numa sala de conferência de Google Meets. Um link de acesso a dois programas a serem descarregados é posto à disposição dos participantes, e antes de começar a equipe se assegura de que os participantes aceitam o Termo de Consentimento Livre e Esclarecidos (link do termo: https://docs.google.com/document/d/1cMWDA8oIEb3E0nzshf-Kxqfgo9OycDG2aYR7vrlFpZ4Q/edit?usp=sharing), sendo lembrados de que serão gravados. Também é requerido um compartilhamento da tela dos participantes, para fazer a avaliação do seu desempenho possível.

Apôs uma descrição breve do nosso projeto, os participantes são acompanhados no processo até que conseguem abrir a interface satisfatoriamente no computador. Nesse ponto, a tarefa é descrita, solucionam-se possíveis dúvidas e se deixa o jogador sozinho testando o primeiro cenário desenvolvido. Uma vez completado o primeiro cenário, se descreve a tarefa a ser completada no segundo cenário e repete-se o procedimento. Ao finalizar, se realiza uma entrevista com um roteiro breve e se deixa um tempo para outras perguntas ou comentários livres com os participantes.

Um moderador será o encarregado de dar as boas vindas aos participantes, acompanhar eles no processo de descarregar e executar os programas a serem testados e de enunciar as tarefas para eles. Outro moderador será o encarregado de fazer a entrevista final, e os membros da equipe restantes tomarão os dados que sejam mais relevantes para a análise dos resultados. Também será medido o tempo total de realização do teste. Os testes que superem os 20 minutos serão interrompidos.

Papéis

Os papéis definidos são os seguintes:

Jogador

- Usuário que executará os cenários do teste;

Moderador(es)

- responsável/eis pela explicação do teste e do cenário, além de prestar ajuda em eventuais dúvidas ou problemas;
- responsável/eis por dar uma explicação simples sobre o menu de opções;

Observador(es) do teste

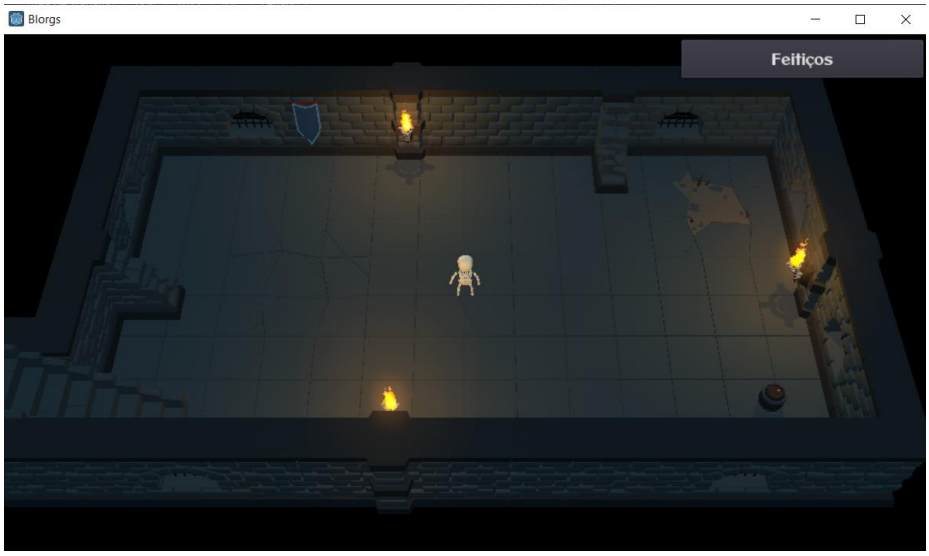
- responsável/eis por observar e anotar pontos relevantes sobre a experiência do jogador nos cenários;
- responsável/eis por identificar eventuais erros, problemas ou *bugs* com o protótipo ao longo da execução dos testes;
- apenas observa o teste, não se comunica com o participante.

Tarefas de usabilidade

Cenário 1

- **Tarefa dada ao usuário:** Movimentar o personagem até a bandeira na esquerda superior e depois mover até o barril na direita inferior, como pode ser visto na imagem abaixo.
- **Interface apresentada:** Botão que apresenta a lista de funções disponíveis (uma função apenas).
- **Caminho esperado:** O usuário deve abrir a lista de funções, selecionar a função de mover e por último selecionar o destino final da movimentação.
- **Link para o Cenário 1 (Windows, é necessário apenas extrair o executável e abrir):** https://drive.google.com/file/d/1VPbK2SxrqfI-4Zr8sCMFazEUeFLYxth4/view?usp=sharing

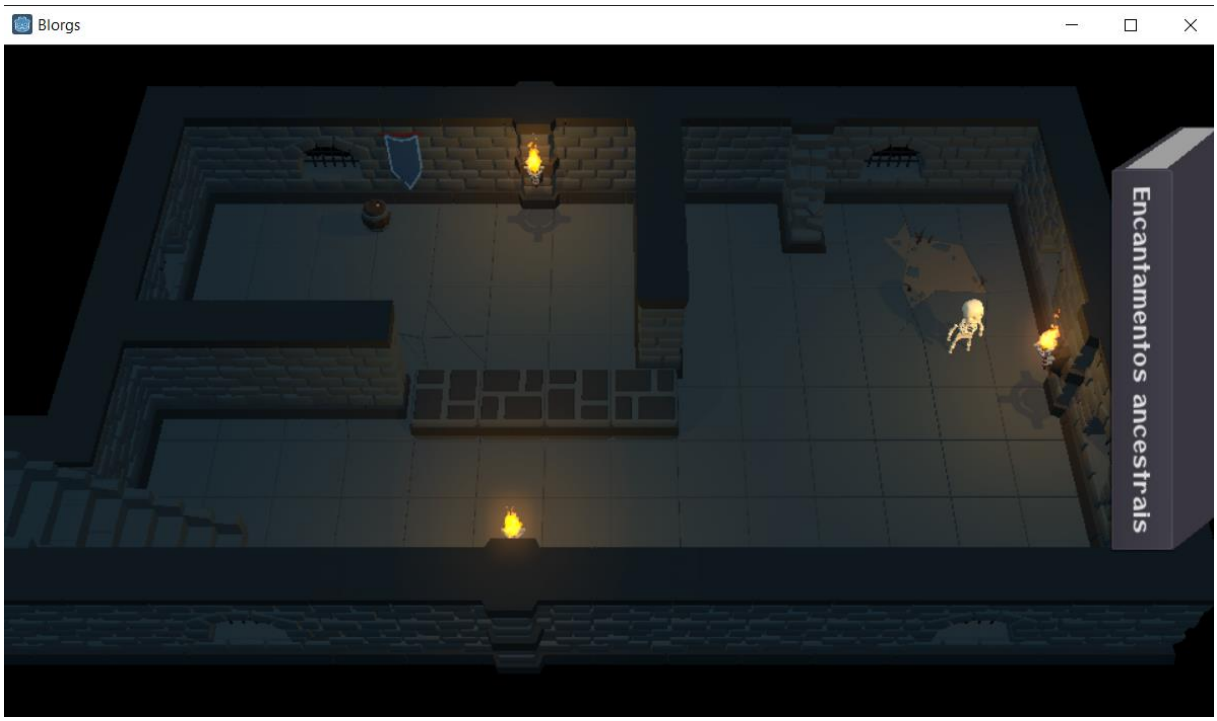
Figura 1 - Protótipo do cenário 1



Cenário 2

- **Tarefa dada ao usuário:** Movimentar o personagem o barril na esquerda, pulando por cima de um obstáculo no meio do caminho, como pode ser visto na imagem abaixo. Em seguida, movimentar o personagem para o ponto de partida, precisando novamente pular sobre o obstáculo.
- **Interface apresentada:** Botão que apresenta a lista de funções disponíveis (uma função de movimentação e uma de pulo).
- **Caminho esperado:** O usuário deve abrir a lista de funções e em seguida selecionar a função de mover, mover-se até o obstáculo, selecionar a função de pulo, ultrapassar o obstáculo, selecionar a função de movimentação de novo e seguir até o objetivo.
- **Link para o Cenário 2 (windows, é necessário apenas extrair o executável e abrir):** https://drive.google.com/file/d/1M2NR-5-Pb7UAVxUUqNqPHfMeuZAGk-IS/view?usp=sharing

Figura 2 - Protótipo do cenário 2



Métricas de usabilidade

Métricas de usabilidade se referem a dados de performance do usuário relativos a execução de tarefas ou objetivos específicos. As métricas que serão coletadas são: finalização dos cenários; taxa de erro; avaliações subjetivas (sendo elas: grau de entendimento das mecânicas, facilidade de navegação pela interface).

Finalização dos cenários

Cada cenário necessita de que o usuário faça determinadas ações/operações no jogo. O cenário é finalizado ao se completar o objetivo que, idealmente, é atingido sem intervenção do moderador, mas, caso o usuário apresenta muita dificuldade, pode ser necessário dar dicas ou até guiá-lo por todo, podendo-se assim considerar como um erro crítico.

Erros críticos

Consideraram-se erros críticos aqueles que impedem o participante de resolver sozinho os dois cenários, ou que causem que ele não seja capaz de resolver a tarefa nos 20 minutos de limite.

Erros não-críticos

Consideraram-se erros não críticos aqueles que impedem uma finalização eficiente dos cenários por parte do jogador. Por exemplo, se ele faz ações de forma aleatória (em nosso caso, cliques do *mouse* ou *trackpad*), ou se ele completa a tarefa com claros indícios de não ter entendido a estrutura do jogo. Se consideraram também erros não-críticos aqueles aspectos da interface que gerem emoções indesejáveis no participante, como mostras de frustração, raiva ou tristeza; ou aqueles que sejam fonte de confusão para o participante.

Avaliações Subjetivas

Algumas métricas subjetivas como referentes a facilidade de uso e satisfação serão coletadas numa entrevista rápida pós teste.

Entrevista pós-teste

Após a finalização das duas fases, o moderador (ou moderadores) faz uma série de perguntas para o participante.

1. O que você achou do fato de ter que clicar na função para usá-la, e então, se quiser usá-la de novo, ter que clicar novamente na função?
2. Achou que o sistema estava autoexplicativo?
3. Achou o sistema complicado de operar?
4. Usaria o produto final?

Além da entrevista, se abre um tempo para sugestões pessoais do participante, de forma que potenciais usuários da interface possam formar também parte do seu processo de design.