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.

Henrique Ferreira Nogueira

JOGO EDUCATIVO PARA AUXILIAR NO APRENDIZADO DE CONCEITOS DE LÓGICA DE PROGRAMAÇÃO

Palmas – TO

2022

Henrique Ferreira Nogueira

JOGO EDUCATIVO PARA AUXILIAR NO APRENDIZADO DE CONCEITOS DE LÓGICA DE PROGRAMAÇÃO

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

Orientador: Prof. Esp. Fernanda Pereira Gomes.

Palmas – TO

2022

Henrique Ferreira Nogueira

JOGO EDUCATIVO PARA AUXILIAR NO APRENDIZADO DE CONCEITOS DE LÓGICA DE PROGRAMAÇÃO

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

Orientador: Prof. Esp. Fernanda Pereira Gomes.

Aprovado em: _____/_____/_____

BANCA EXAMINADORA

_____

Prof. Esp. Fernanda Pereira Gomes

Orientador

Centro Universitário Luterano de Palmas – CEULP

_____

Prof. Esp. Fábio Castro Araújo

Centro Universitário Luterano de Palmas – CEULP

_____

Prof. M.e Madianita Bogo Marioti

Centro Universitário Luterano de Palmas – CEULP

Palmas – TO

2022

RESUMO

NOGUEIRA, Henrique F.. **Jogo Educativo Para Auxiliar No Aprendizado De Conceitos De Lógica De Programação**. 2021. 00 f. Projeto Tecnológico (Graduação) – Curso de Sistemas de Informação, Centro Universitário Luterano de Palmas, Palmas/TO, 2021[1].

O presente trabalho discute o desenvolvimento de um jogo educativo para o auxílio na aprendizagem dos conceitos de lógica de programação. O processo de aprendizagem de lógica de programação é marcante dentro de cursos da área da computação, porém, existem muitos casos de dificuldades, reprovação e desistência em meio a este processo. Dada a barreira no ensino de novos estudantes da área, ferramentas e métodos diferentes buscam superá-las em inúmeras situações. Portanto, o atual projeto busca apresentar uma ferramenta de apoio para os seus usuários no processo de aprendizagem de lógica de programação. O jogo Aprendizado de Makina foi desenvolvido após uma entrevista com uma especialista no domínio que atua na área de ensino de lógica de programação, onde foram levantados pontos de dificuldade e de atenção para o desenvolvimento. Com base nisso, o jogo foi definido, os requisitos foram levantados e o desenvolvimento ocorreu utilizando um motor de jogo com scripts em Lua, o *Solarus* e desenhado utilizando a ferramenta *Aseprite*. O jogo é do gênero *Puzzle* e também contém aspectos do gênero RPG, nele o jogador deve controlar a personagem principal Makina para conseguir concluir os cenários presentes no jogo. Cada cenário possui desafios relacionados à conceitos básicos da lógica de programação, com o intuito de tornar a experiência de jogá-los um aprendizado.

Palavras-chave: Educação, Lógica de programação, Jogos, Apoio, Aprendizado, Ensino.

[1] Elemento incluído com a finalidade de posterior publicação do resumo na internet. Sua formatação segue a norma ABNT NBR 6023, por isto o alinhamento e o espaçamento diferem do padrão do texto.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CEULP/ULBRA - Centro Universitário Luterano de Palmas

NPC - *non-playable character,* que se traduz como personagem não-jogável

SUMÁRIO

1 INTRODUÇÃO

Segundo Mortari (2001, p. 2), “lógica é a ciência que estuda princípios e métodos de inferência, tendo o objetivo principal de determinar em que condições certas coisas se seguem (são consequências), ou não, de outras”. Em seu dia a dia, o ser humano toma decisões baseando-se em lógica a todo momento, portanto, é possível afirmar que esta é então uma habilidade inerente de uma pessoa.

Ao levar a lógica do dia a dia para um contexto mais aplicado ou diferente do que o indivíduo está familiarizado, é possível que haja certa confusão para acordar os próprios conceitos e ideias ao novo ambiente. No ambiente computacional é comum que isso ocorra com novas pessoas sendo introduzidas na área.

De acordo com Xavier (2018), para inserir a lógica no contexto das máquinas é preciso que uma melhor sequência de ações seja almejada para que um objetivo seja atingido, essas sequências são nomeadas algoritmos. Forbellone e Eberspächer (2005, p. 3) definem algoritmos como uma sequência de ações claras e precisas, que produzem um estado final esperado, solucionando o problema para o qual foi proposto. A visão de um algoritmo para alguém que acabou de ser inserida no contexto de programação, por muitas vezes, é visto como algo abstrato e de difícil compreensão.

“O processo de ensino e aprendizagem de lógica de programação é considerado um desafio para estudantes de Computação, apesar dos numerosos esforços de pesquisa para melhorar esse processo [...]” (FERREIRA et al., 2010, p. 981). Em alguns casos, segundo Santos et al. (2008), há uma relação de falta de motivação às altas taxas de reprovação e evasão dos cursos com disciplinas de algoritmos e programação.

Visto que é gerada uma barreira no aprendizado e que por falta de recursos para contorná-la, o aluno pode vir a se sentir desmotivado pela quantidade de esforço despendido na tentativa de aprender algo novo, sem resultado. E com esta dificuldade em mente, diversas ferramentas e metodologias são propostas e desenvolvidas para tornar o aprendizado tangível a um número maior de pessoas.

Estas ferramentas são desenvolvidas para apoiar tanto quem ensina quanto quem aprende. Ambientes especiais de programação, ferramentas de depuração de pseudocódigo e simuladores de código com representação visual, são exemplos destas ferramentas e, mesmo com um número considerável de ambientes, ainda são criadas propostas de materiais de auxílio.

Estes meios alternativos de ensino de lógica de programação, podem ser ambientes especiais e simuladores visuais de código, no entanto, estes ainda podem falhar no seu propósito. Logo, outras formas de facilitar o processo de ensino são estudadas e avaliadas sendo os jogos eletrônicos uma delas .

Os jogos podem ser utilizados como ferramentas de ensino em diversos contextos, incluindo o contexto de lógica de programação. De acordo com Battaiola (2000, p. 7), um jogo classificado como educativo, pode possuir características de qualquer outro tipo de jogo, porém, leva em consideração critérios didáticos e pedagógicos com a visão de transmitir um conceito no qual está associado.

Quanto à utilização de jogos com o objetivo de educar, Grübel e Bez (2006, p.6) afirmam que

> "Os jogos educativos tanto computacionais como outros são, com certeza, recursos riquíssimos para desenvolver o conhecimento e habilidades se bem elaborados e explorados. São uma estratégia de ensino podendo atingir diferentes objetivos e áreas do conhecimento."

A existência de ferramentas como os jogos educativos pode auxiliar no ensino e aprendizado, pois apresentam uma abordagem diferente do tema e trazem meios alternativos de explorá-lo. Jogos, em geral, possuem elementos que desafiam o seu jogador de forma justa, o que desperta suas características competitivas, aumentando o interesse e a atenção de seu usuário, devido ao caráter lúdico da atividade. Portanto, um jogo que utilize estes elementos com o intuito de prestar apoio ao docente possibilitando a ele um acompanhamento dos pontos de dificuldade, mostra-se interessante por possuir meios atrativos ao usuário-aluno de alcançar suas metas em aprender e assimilar os conteúdos ensinados.

Dessa maneira, utilizar um motor de jogo, que é um kit de desenvolvimento de *software* reutilizável para a criação de jogos (GREGORY, 2009, p.3), com o intuito de desenvolver um jogo educativo para prestar auxílio no ensino de lógica, é uma possível alternativa para a resolução de algumas dificuldades de aprendizado na programação.

Portanto, este trabalho se propôs a desenvolver um jogo educativo para auxiliar no ensino e na aprendizagem de lógica de programação de alunos ingressantes nos cursos de Ciência da computação, Engenharia de Software e Sistemas de Informação do CEULP/ULBRA.

Consequentemente, o objetivo de desenvolver um jogo foi destrinchado em passos menores, como: definir gênero e motor do jogo a ser desenvolvido, definir os aspectos de design do jogo tais como níveis, personagens, enredo, regras, interfaces e cenário, criar

protótipos de telas do jogo, modelar níveis de acordo com os conceitos apresentados em lógica de programação e implementar o jogo de acordo com os protótipos e modelos criados.

Isso foi feito com a hipótese de que com um motor de jogo e um planejamento adequado, seria possível o desenvolvimento do jogo educativo, apresentando uma abordagem diferente da tradicional aos alunos, pois esta etapa da formação do aluno é fundamental, e em alguns casos, apresenta-se certa dificuldade do mesmo na assimilação do conteúdo, visto que, é um contato com algo novo, que não lhe fora apresentado na grade do ensino médio tradicional.

2 REFERENCIAL TEÓRICO

Esta seção apresenta definições da Lógica de Programação (subseção 2.1), os seus conceitos básicos (subseção 2.1.1), proporciona o entendimento do cenário das dificuldades no aprendizado de lógica de programação (subseção 2.1.2), dispõe uma visão geral das definições de jogos (subseção 2.2), apresenta a vertente de jogos digitais (subseção 2.2.1) e jogos educativos (subseção 2.2.2) e apresenta trabalhos relacionados (subseção 2.3).

2.1 LÓGICA DE PROGRAMAÇÃO

O dicionário Houaiss (LÓGICA, 2021) define lógica como uma "maneira por que necessariamente se encadeiam os acontecimentos, as coisas ou os elementos de natureza efetiva" e, de acordo com o dicionário Michaelis (LÓGICA, 2021), lógica inserida no contexto de informática é a "maneira pela qual instruções, assertivas e pressupostos são organizados num algoritmo para viabilizar a implantação de um programa". Logo, a lógica pode ser entendida como um meio para certo fim, e aplicá-la à informática é apenas utilizá-la de forma específica para um contexto específico.

Segundo Almeida (2008), a lógica de programação "(...) tem como objetivo mostrar métodos para solucionar problemas, além de determinar a sequência lógica para o desenvolvimento de um programa", e esta é composta pelos seguintes elementos: organização, perseverança, padronização, criatividade e otimização.E com a necessidade de resolução de inúmeros problemas pela área de programação, a lógica é vista como um dos principais recursos necessários para tal feito.

Dentro do contexto de lógica de programação, existem alguns conceitos a serem tratados para o seu entendimento, a próxima subseção tratará dos conceitos básicos referentes a lógica de programação.

2.1.1 Conceitos básicos

Referente aos conceitos básicos da lógica de programação, temos os seguintes tópicos a serem abordados:

- **Algoritmo:** conforme Forbellone e Eberspächer (2005) , "Um algoritmo pode ser definido como uma sequência de passos que visam a atingir um objetivo bem definido.". Podendo ser facilmente percebido no cotidiano de qualquer pessoa, pois para realizar qualquer tipo de tarefa, são necessárias algumas ações (passos) e sabe-se qual o resultado esperado desta sequência. Um exemplo comumente utilizado é o de que o algoritmo é como uma receita de bolo, que descreve os ingredientes e o passo a

passo do que deve ser feito para obter o bolo ao fim da série. Quando algoritmos são escritos em uma linguagem de computador, para serem interpretados e executados por uma máquina, eles se tornam programas, diz Moraes (2000).

- **Operadores:** Almeida (2008) define operadores como meios de trabalhar com dados no computador, sendo comparado, incrementando ou decrementando. E os separa em aritméticos, relacionais e lógicos, que estão a seguir.

  Os operadores aritméticos trabalham com expressões aritméticas e se dividem em dois tipos, os binários, onde agem dois operandos, e unários, onde opera-se a inversão de um valor. Exemplos de operadores: + (adição), - (subtração), * (multiplicação), / (divisão). Com os operadores + e - estando presentes em ambos os tipos, porém, com funções diferentes, onde nos operadores unários eles representam, respectivamente, manutenção do sinal e inversão do sinal, e nos operadores binários, adição e subtração.

  Esses operadores podem ser vistos nos primeiros ensinamentos de programação, onde são feitos programas de mini-calculadoras somando dois números dados pelo usuário, ou até em programas mais complexos e aplicados a contextos diferentes, como sistemas de lojas ou cálculos matemáticos diversos.

  Já os operadores relacionais, comparam valores e retornam valores lógicos, como verdadeiro (*true*) e falso (*false*). Exemplos: > (maior que), < (menor que), = (igual) e <> (diferente). Estes operadores e os citados anteriormente, são de maior conhecimento prévio, pois são utilizados em níveis inferiores de ensino.

  Por fim, os operadores lógicos unem expressões relacionais, formando novas expressões, que também resultam em valores lógicos, verdadeiro ou falso. E são eles: E (*AND*), OU (*OR*) e NÃO (*NOT*). Onde os termos são comparados de acordo com os operadores lógicos, que na ordem, todos devem ser verdadeiros para um resultado verdadeiro, algum dos termos deve ser verdadeiro para que se retorne verdadeiro e por fim, o resultado será o inverso do valor lógico do termo.

- **Variável:** Em concordância com Almeida (2008),

  > "Variável é a representação simbólica dos elementos de certo conjunto. Cada variável ocupa uma posição de memória, cujo conteúdo pode se modificar durante a execução de um programa. Mesmo que uma variável assuma diferentes valores, ela só poderá armazenar um valor por vez.".

Um exemplo de definição de variável pode ser visto na Figura 1, a seguir.

Figura 1 - Variável e conteúdo da variável

Fonte: Livro Curso Essencial de Lógica de Programação

- **Tipos de dados: s**egundo Medina e Ferting (2006), um tipo de dados é "como um conjunto de objetos que têm em comum o mesmo comportamento diante de um conjunto definido de operações". O autor define como os tipos mais frequentes nas linguagens de programação: os números inteiros, os números reais, as letras e os booleanos. Sendo este último derivado da lógica de Boole, onde existe um conjunto de apenas dois valores significativamente opostos, representados por 1 e 0 ou Verdadeiro (V) e Falso (F).

- **Condicionais:** quando há a necessidade de alterar o curso de execução do algoritmo a depender de certas condições, utiliza-se os comandos de condição, conforme diz Medina e Ferting (2006). Com isso, alguns comandos são utilizados para referenciar de que forma será conduzido o fluxo a depender do resultado da condição. Geralmente, esses comandos são: **se** (*if),* **senão** *(else)* e **senão se** (*elif)*. Tais comandos são detalhados a seguir:
    - o comando **se** (*if*) pode ser interpretado assim como na vida real, na qual caso uma condição seja verdadeira, condição que pode ser representada por uma variável, constante ou expressão, um bloco de comandos será executado;
    - o comando **senão** (*else*) é opcional e acompanha o comando se, onde, caso a condição do comando **se** não seja verdadeira, o bloco de comandos do comando senão será executado;
    - já o comando **senão se** (*elif*), funciona de forma semelhante ao comando **senão**, porém nele é feita outra verificação de condição, para então, caso verdadeira, ocorra a execução do bloco.

- **Laços de repetição:** de acordo com Almeida (2008), eles são estruturas que aceitam a repetição de uma sequência de comandos. Essa repetição pode ser contada, na qual sabe-se quantas vezes o comando será executado, ou condicional, onde a iteração depende de uma alteração dentro das instruções repetidas. Representantes comuns destas estruturas são os comandos: ***while*** (enquanto) e ***for*** (para). A seguir, explicações sobre os comandos citados:
    - o comando ***while***, está dentro dos laços condicionais e testa uma condição em seu início, e caso seja verdadeira, executa os comandos dentro do laço e volta a testar se a condição continua verdadeira, até o momento em que ela se torne falsa e a execução continue para após o bloco do laço;
    - o ***for***, representa os laços de repetição contada, onde a partir de um valor inicial, os comandos são executados e ao fim destes esse valor inicial é incrementado até atingir um valor final, encerrando a repetição de comandos. o valor inicial, valor final e incremento são todos conhecidos antecipadamente.

Com esses conceitos básicos bem estabelecidos na formação, o aluno consegue por si só começar a pensar da forma lógica que é necessária para a programação. Porém, nem sempre é possível que esse aprendizado seja feito de forma amena e sem grandes obstáculos, o que será abordado no tópico a seguir.

2.1.2 Dificuldades no aprendizado

A dificuldade no aprendizado de lógica de programação gera numerosas discussões por diferentes autores, demonstrando a relevância do tema no meio educacional. Diferentes autores destacam alguns pontos observados que revelam certas semelhanças entre si.

O nível de abstração requerido pelo estudante é visto como um dos obstáculos no estudo de programação. De acordo com Júnior e Rapkiewicz (2004), é notável a dificuldade apresentada por grande parte dos alunos em assimilar as abstrações envolvidas na programação, onde o mesmo por vezes não consegue compreender o contexto de um problema apresentado e o que é necessário para resolvê-lo, levando ao fato de que um dos maiores índices de reprovação em parte das instituições de ensino brasileiras se dá por esta disciplina.

Além de frisar o mesmo ponto, Ribeiro et al. (2011), acrescenta a falta de motivação do aluno referente ao despreparo e desânimo, a falta de clareza no objetivo da disciplina e as dificuldades no desenvolvimento do raciocínio lógico, decorrente ao fato do aluno estar

acostumado a decorar os conteúdos apresentados a ele, como possíveis causas da dificuldade apresentada.

Mattos (2001), similarmente, salienta que em alguns casos há a falta de experiência dos alunos com os aspectos apresentados nos contextos expostos aos estudantes, pois, a visão do ambiente passada nos contextos dos exercícios propostos, em sua maioria, são abstratos para o aluno, tornando difícil a identificação das informações necessárias para a resolução dos problemas descritos nos exercícios.

Moretto (2000), também cita o modelo pedagógico utilizado, o modelo instrucionista, que constitui-se em o professor somente como um transmissor de conhecimento, como o centro das relações entre o aluno e o conhecimento. O que pode ser problemático em alguns casos, tanto em questão de formas de pensar quanto em questões sociais, pois é um relacionamento entre dois seres humanos e está fadado a fatores humanos, mesmo tendo um objetivo claro e de ambas as partes.

Portanto, vê-se a necessidade de meios alternativos de abordar os assuntos relacionados à lógica de programação, tornando mais fácil a abstração e buscando motivar o acadêmico na busca de aprender os conceitos apresentados. Haguenauer et al. (2008) afirma que os jogos têm a capacidade de ser uma ferramenta eficiente, pois além de facilitarem o aprendizado, apoiando a retenção do que foi ensinado, eles motivam e divertem.

Macedo et al. (2008)**,** traz o jogo e o lúdico como um meio alternativo pois o mesmo acredita que essa também é uma forma de comunicar e expressar ideias, enquanto estudos feitos por Calderaro et al. (2003) indicam que os jogos

> "favorecem a apreensão de estratégias de aprendizagem específicas, como estratégia de processamento, a metacognição e as estratégias afetivas, o desenvolvimento das quais poderá contribuir para a prevenção das dificuldades de aprendizagem".

2.2 JOGOS

Diversos autores buscam uma definição para "jogo", porém, devido à ampla quantidade de formas e características que o mesmo pode tomar, alguns desafios podem surgir na tentativa de conceituar jogos. Fatores sociais, culturais e até temporais podem afetar a visão de um estudioso quanto à conceituação de jogo. A seguir, serão apresentadas algumas definições e categorizações de diferentes autores para o jogo.

Juul (2005, p. 6), afirma que para um jogo ser considerado como tal, ele necessita ser:

1) um sistema formal baseado em regras;
2) com resultados quantificáveis e variáveis;

3) onde resultados diferentes são atribuídos a valores diferentes;
4) onde o jogador exerce esforço para influenciar o resultado;
5) o jogador se sente emocionalmente ligado ao resultado; e,
6) as consequências da atividade são opcionais e negociáveis.

Huizinga (2007) caracteriza o jogo como uma atividade voluntária, inserida dentro de uma realidade que extrapola a esfera da vida. Dentro desta atividade o jogador deve reconhecer o jogo e todo o seu conjunto de regras. Sensações de tensão, alegria e incerteza dos resultados, são fundamentais dentro do sistema e sempre há algo "em jogo" que ultrapassa as necessidades da vida e dão sentido à ação. Em termos de tempo, o jogo sempre tem um fim dentro dele mesmo, já quanto a espaço, o jogo ocorre e se encontra dentro de uma área delimitada antecipadamente, seja de forma física ou imaginária.

De acordo com Crawford (1984), em sua busca para compreender quais são os elementos fundamentais comuns aos jogos, ele percebeu quatro fatores:

- **Representação:** o jogo é um sistema fechado formal que subjetivamente representa um subconjunto da realidade. Ele é completo e autossuficiente como uma estrutura. Suas regras são explícitas, suas partes interagem entre si formando o sistema, representando uma realidade subjetiva e fantasiosa que significa algo na mente do jogador;
- **Interação:** algumas mídias representam a realidade de forma estática e outras de forma dinâmica, porém o que mais fascina não é o que ela é nem que ela muda, mas como ela muda. A rede de causa e efeito onde todas as coisas estão ligadas, só pode ser representada permitindo o espectador da mídia, a explorar cada parte da obra para gerar causas e observar efeitos. Dito isto, a maior forma e mais completa forma de representação é a representação interativa;
- **Conflito:** conflitos surgem naturalmente da interação em um jogo. O jogador está sempre tentando atingir o seu objetivo de forma ativa. Obstáculos são postos para impedir que o jogador alcance seu objetivo facilmente e estes necessitam de um agente inteligente, que por sua vez geram o conflito com o jogador;
- **Segurança:** Jogos são formas seguras de experimentar a realidade. Por mais que, consequências sejam geradas pelos feitos do jogador no jogo, estas são menores do que seriam caso o mesmo ocorresse na situação que o jogo representa.

Desta forma é possível destacar alguns pontos comuns entre os autores, como a representação de uma realidade em que o jogador vai existir, as regras do sistema em que ele está participando e a causa e consequência, em que o ator deve tomar suas decisões para então receber um feedback do sistema em forma da tal consequência.

Caillois (2017) por sua vez classifica os jogos em quatro categorias: *Agôn*, *Alea*, *Mimicry* e *Ilinx*.

> "As quatro pertencem ao campo dos jogos: *jogamos* futebol ou bolinhas de gude ou xadrez (*agôn*) ou *jogamos* na roleta ou na loteria (*alea*), *jogamos de brincar* de pirata ou *jogamos de representar* Nero ou Hamlet (*mimicry*), ou o jogo é provocar em nós, por um movimento rápido de rotação ou de queda, um estado orgânico de confusão e de desordem (*ilinx*)."

Enquanto Crawford (1984) observa cinco grandes áreas dos jogos: jogos de tabuleiro, jogos de cartas, jogos atléticos, jogos de crianças e jogos de computador. Dentro da última área ele a separa em dois grupos, os jogos de habilidade-e-ação e os jogos de estratégia. No grupo dos jogos de habilidade-a-ação, têm-se: jogos de combate, jogos de labirinto, jogos de esportes, jogos de remo, jogos de corrida e jogos diversos. Ao mesmo tempo que outro grupo temos os jogos de aventura, Jogos *D&D*, jogos de guerra, jogos de possibilidade, jogos infantis e educacionais e jogos interpessoais.

Consequentemente, a literatura disponível sobre o tema traz diferentes visões e definições, com pontos comuns, como também apresenta meios variados de categorizar os jogos. O estudo e análise de sua essência e como ela afeta o ser humano são realizados por diversos autores, pois como Huizinga (2007) apresenta, "O jogo é fato mais antigo que a cultura, pois esta, mesmo em suas definições menos rigorosas pressupõe sempre a sociedade humana", logo, ele está na natureza humana desde o seu princípio.

Com a evolução tecnológica presente na vida de todos, ferramentas de trabalho, comunicação e entretenimento sofreram algumas mudanças. Com isso, os jogos também possuem sua participação na área tecnológica, com os jogos digitais que serão apresentados na seção a seguir.

2.2.1 Jogos digitais

O conceito de jogos digitais se iniciou com o jogo *Tennis for Two*, desenvolvido por William Higinbotham, em 1958. Durante um evento do *Brookhaven National Laboratory*, Higinbotham decidiu deixar a atração mais engajadora para o público. De acordo com Donovan (2010), ele utilizou a tela de um osciloscópio construído com a ajuda do engenheiro

Robert Dvorak, para recriar um jogo de tênis, visto lateralmente, com uma rede ao meio, uma bola sendo rebatida para ambos os lados e pequenas linhas indicando as raquetes dos jogadores, como ilustrado na figura 2.

Figura 2 - Recriação do jogo *Tennis for Two* apresentada no 50º aniversário do jogo

Fonte: Website do Brookhaven National Laboratory

A partir deste ponto, diversos outros jogos digitais começaram a surgir, como o jogo *Spacewar!*, finalizado em 1962, e o *Pong*, lançado oficialmente em 1972, sendo este um dos primeiros jogos lucrativos da história (LOWOOD, 2009). Com isso, diversos pesquisadores começaram a buscar uma definição clara para o que seria então um jogo digital.

Como definição para um jogo digital, Kirriemuir e McFarlane (2004) propõem as seguintes características:

- proporcionam alguma informação visual digital ou matéria para um ou mais jogadores;
- recebem alguma entrada dos jogadores;
- processam a entrada de acordo com um conjunto de regras programadas do jogo;
- alteram as informações digitais fornecidas aos jogadores.

Assim sendo, o maior fator para caracterizar um jogo digital seria a interação do jogador com o jogo, diante uma interface que apresente as informações digitais diretamente ao usuário. Para Battaiola (2000), três partes básicas compõem um jogo por computador:

- Enredo: trata sobre a temática e a trama do universo criado, apresentando o objetivo do jogo, no qual o jogador despende esforços para atingir;
- Interface interativa: aborda a representação gráfica de alteração de estados. Na interface há valores artísticos, cognitivos e técnicos, envolvendo desde a apresentação do jogo e interpretação da informação gráfica a performance e portabilidade. Controla a interação usuário-motor;
- Motor: sistema de controle do jogo. Mecanismo onde ocorre a coordenação de reações do jogo quanto a alguma ação do jogador. Envolve aspectos computacionais, como linguagem de programação e técnicas para estruturar dados.

Portanto, é possível perceber a relação entre jogos e computadores, trazendo as características de um jogo e implementando-as por meios computacionais, utilizando de interfaces digitais, linguagens de programação e outros mecanismos da área.

Herz (1997), propôs um sistema para categorizar os jogos digitais nas seguintes categorias: jogos de ação, aventura, luta, quebra-cabeças, interpretação de papéis, esportes, estratégia e simulações. No entanto, alguns autores trazem mais categorias para os jogos, ou até subdividem as apresentadas por Herz (1997). Battaiola (2000) e Crawford (1984), identificando também os jogos educacionais, que serão apresentados a seguir.

2.2.2 Jogos Educativos

O autor Crawford (1984) defende a ideia de que todos os jogos são de alguma maneira educacionais, e que os mesmos são o meio mais antigo e consagrado de educação. Porém, os jogos nesta categoria tem em seu planejamento, o objetivo explícito de educar o jogador.

Kapp (2013) apresenta algumas razões para implementar uma experiência de aprendizado interativa, como criar interatividade na entrega do aprendizado, superar o desinteresse, proporcionar oportunidades para reflexão profunda, mudar positivamente o comportamento e a prática autêntica. Adicionalmente, o escritor afirma que o uso de jogos encoraja o aprendiz, pois desafios, objetivos e progressão são peculiaridades que engajam e encorajam o ser humano.

McGonigal (2011) enuncia que os jogos educacionais são uma indústria enorme e crescente, e que eles estão sendo desenvolvidos para ajudar a ensinar praticamente qualquer tópico ou habilidade imaginável. Quando este tipo de jogo atende seu requisito, ele promove um alívio gratificante ao aluno que, de outra maneira, se sente pouco engajado em sua vida escolar.

Com isso em mente, diversas empresas e instituições passaram a ver esse tipo de jogo como um método para apoiar nas dificuldades encontradas e/ou na passagem de conhecimento para alunos e funcionários. Alguns desses jogos já tem seu acesso disponível e abordam diferentes áreas do conhecimento.

O jogo "**Jogos de Matemática - Adição e subtração, contagem**" é um bom exemplo, ele é um aplicativo para *smartphone*, disponível na *Google Play*, desenvolvido pela *RV AppStudios*, que busca ensinar matemática para crianças de 05 a 12 anos. Em seu ambiente são apresentadas diferentes atividades para o jogador exercitar suas habilidades em contagem, soma e subtração. A figura 3 representa a tela onde o jogador pode selecionar o modo de jogo que ele jogará.

Figura 3 - Tela do jogo Jogos de Matemática - Adição e subtração, contagem

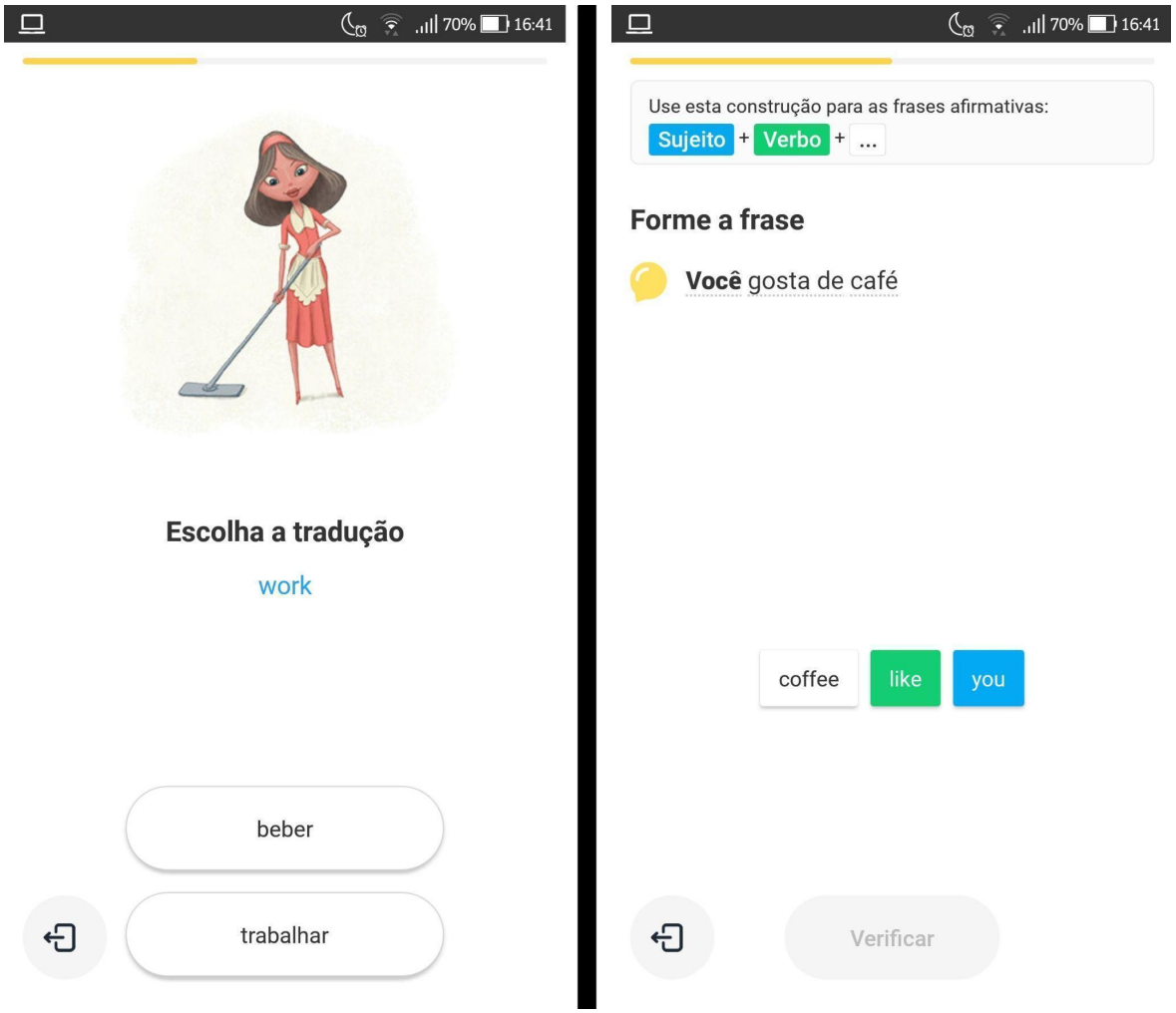
Fonte: Google Play

O jogo se baseia em uma interface onde o jogador deve escolher as opções corretas para realizar as operações matemáticas e passar de fase. Alguns modos de jogo possuem níveis de dificuldade a serem escolhidos. Alguns indicadores de ajuda surgem caso o usuário esteja com alguma dificuldade e tome muito tempo para responder. Também há um incentivo audiovisual por parte do jogo quando o jogador passa pelos desafios.

Já o jogo "**Simpler**", explora outra área de conhecimento, ele é um aplicativo para *smartphone*, desenvolvido pela *Goodvas*, que ensina inglês dentro de sua plataforma. Com

uma sensação de progresso e recompensas a partir de pontuações, a plataforma incentiva seu jogador a manter uma rotina de jogo para continuar praticando os seus estudos de inglês. A figura 4 demonstra algumas telas de jogo, onde o jogador recebe opções para responder o que lhe foi pedido na fase.

Figura 4 - Telas do jogo Simpler



Fonte: Google Play

O jogo conta com um feedback audiovisual após as respostas do jogador e recompensas em forma de pontuação. É possível visualizar o progresso do jogador durante a fase e em cada tema proposto pelo jogo. O jogo conta com uma versão *Premium* onde mais fases são liberadas, como também outras opções para o jogador acessar.

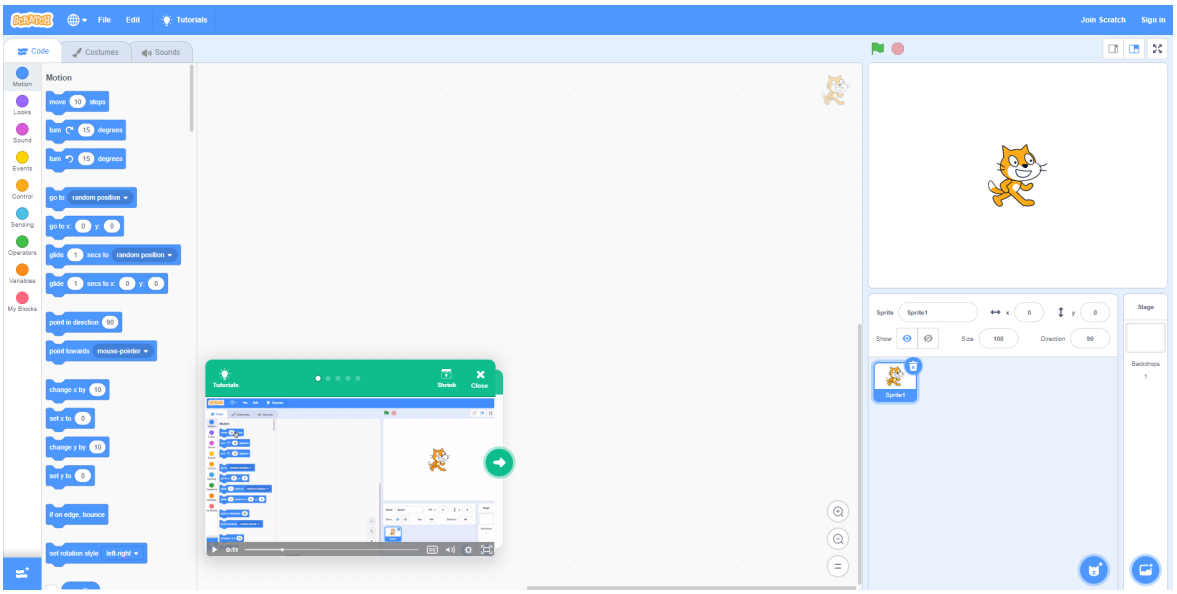
2.3 Trabalhos Relacionados

Acerca de trabalhos relacionados à jogos educativos no contexto da lógica de

programação, alguns projetos e aplicativos foram desenvolvidos. Como por exemplo o Scratch, que tem uma interface gameficada para programar e criar e o Code.org, onde similar ao Scratch o usuário possui um espaço criativo dentro de um ambiente de programação e um *feedback* visual.

O *Scratch* é uma linguagem de programação utilizada para criar histórias, jogos e animações de forma simples, com uma interface própria para o desenvolvimento de seus projetos. Desenvolvido e mantido pela equipe *Scratch* do grupo *Lifelong Kindergarten* do *Media Lab* do *MIT*. Ele possui foco em jovens de 8 a 16 anos, porém possui uma versão simplificada, o *ScratchJr*, para crianças dos 5 aos 7 anos. O *Scratch* possui entre seus valores a colaboração entre sua comunidade, levando em conta como uma competência essencial à vida no século XXI.

Figura 5 - Tela Inicial de Tutorial do Scratch



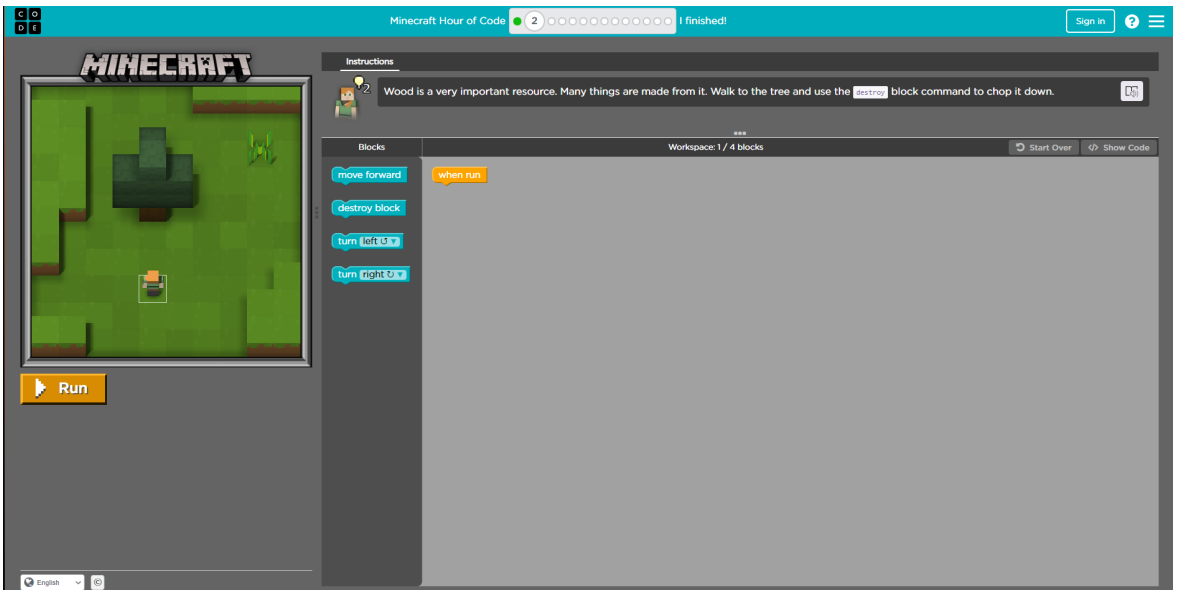
Fonte: Website Scratch

Como mostra a figura 5, o scratch possui uma interface para o desenvolvimento do projeto do usuário, possuindo ferramentas, preview, linhas de código de forma visual e alguns outros detalhes. Dentro da plataforma, existem alguns tutoriais para auxiliar o pensamento criativo do usuário e como programar para realizar o que foi imaginado.

O Code.org é uma organização sem fins lucrativos que busca expandir o acesso à ciência da computação em escolas e aumentar a diversidade de membros da área, como jovens mulheres e estudantes de outros grupos pouco representados. Foi lançado em 2013 pelos irmãos gêmeos Hadi e Ali Partovi.

Dentro da plataforma é possível perceber diversos tutoriais, onde o usuário possui recursos relacionados aos conceitos básicos da programação para realizar tarefas, criar seus próprios mini-jogos, contar histórias e jogar alguns jogos. Como pode ser visto na Figura 6, onde se demonstra um jogo de puzzle onde o jogador deve usar linhas de comando para realizar as instruções passadas a ele.

Figura 6 - Tela de tarefa do Code.org



Fonte: Website do Code.org

O puzzle demonstrado na Figura 6 mostra a criação de um algoritmo, onde o jogador deve organizar suas ações para atingir um objetivo claro dado a ele. Há uma representação visual à esquerda, onde o personagem realiza as ações instruídas pelo jogador na área à direita, o *workspace*.

3 METODOLOGIA

A seção a seguir apresenta os materiais e métodos a serem utilizados no desenvolvimento do trabalho.

3.1 MATERIAIS

Para o desenvolvimento do jogo educativo foram utilizadas as ferramentas *Solarus* como motor de jogo, para criar telas, mecânicas, executar o jogo, entre outros , e o software da *Igara Studio S.A.,* o *Aseprite,* para desenhar as imagens que serão inseridas no jogo, seja de interface, personagens, cenário ou ícone do executável do jogo.

O motor de jogo *Solarus* é uma plataforma gratuita e *open-source*, para desenvolver jogos 2D no estilo ação-RPG. Fundada por Christopho em 2002 para promover um jogo feito por fã, que daria sequência ao jogo *The Legend of Zelda: A Link to the Past (*SOLARUS, 2021).

Programado na linguagem C++, o motor faz toda a computação tradicional (como checagem de colisões) e as operações de baixo nível como desenhar a tela, tocar sons e animar *sprites*, que são imagens de um conjunto de animações, com seus estados e quadros. No Solarus, os jogos criados são chamados de *quest*, e são programados em Lua. Uma *API* chamada *Solarus Lua API* faz a comunicação entre o motor em C++ e os roteiros em Lua, que ditam os objetos e eventos do jogo (SOLARUS, 2021).

O *Aseprite* é uma ferramenta utilizada para criar animações em 2D para jogos. De *sprites* a *pixel-art,* ele permite a criação de qualquer tipo de imagens de jogos de eras passadas dos jogos. Distribuído e atualizado na plataforma *Steam*, ele possibilita desenho e animações 2D com as funções necessárias para tal, em sua interface simples e inspirada em sua própria área artística, os jogos retrôs (ASEPRITE, 2022).

3.2 MÉTODOS

Na Figura 7 é apresentada a metodologia elaborada para o desenvolvimento deste trabalho.

Figura 7 - Fluxograma dos procedimentos metodológicos do trabalho

Fonte: Autor

Como demonstra a figura 7, o desenvolvimento do projeto foi dividido em 6 grandes etapas para o cumprimento dos objetivos propostos, que estão descritas a seguir:

1. **Coletar dados com especialista de domínio:** nesta primeira etapa, foi feita uma entrevista com uma especialista do domínio, para definir alguns parâmetros e métricas para o desenvolvimento do projeto. Nessa entrevista foram estabelecidos pontos de maior dificuldade percebidos pela especialista, passada a percepção de profundidade para cada assunto a ser abordado e discutidas algumas ideias para o jogo. Definidas as métricas e parâmetros, o próximo passo foi fazer uma análise dos dados coletados do domínio de ensino de lógica de programação e jogos, para a sua compreensão para a realização das próximas etapas.

2. **Definição do jogo:** ao realizar o estudo do domínio, foi possível realizar um planejamento adequado da prototipagem e desenvolvimento do projeto. Para isso foi gerado um documento com o escopo do projeto e a descrição de seus elementos. Estabelecendo os critérios necessários para a criação do que foi proposto nos objetivos apresentados.

3. **Prototipagem:** foi realizada a prototipagem do projeto, apresentando telas e mecânicas que seriam desenvolvidas posteriormente. Esta etapa serviu de apoio direto

à etapa seguinte, validando os requisitos a serem atendidos na produção do jogo educativo e auxiliando no entendimento do processo de criação e testes.

4. **Desenvolvimento:** esta etapa consistiu na implementação do jogo proposto, seguindo o que foi prototipado e utilizando os materiais apresentados na subseção 3.1 como recursos para a execução da etapa.

5. **Testes:** esta etapa propôs testes a serem realizados para a validação do projeto e desenvolvimento. Ao serem realizados os testes de unidade, foi possível certificar o correto funcionamento de mecânicas, telas e interfaces, como também aspectos de enredo do jogo.

4 RESULTADOS

Esta seção tem por objetivo apresentar e descrever as etapas que foram realizadas no presente trabalho e os resultados obtidos. As etapas foram divididas em: subseção 4.1 Entrevista com especialista, 4.2 Definição do jogo, onde foram definidas Narrativa e Personagens (subseção 4.2.1), Sistemas do Jogo (subseção 4.2.2), Cenários (subseção 4.2.3), Desafios (subseção 4.2.4), Regras (subseção 4.2.5) e o *Design* do Jogo (subseção 4.2.6), 4.3 Prototipagem e 4.4 Desenvolvimento com as subseções de Cenários (subseções de 4.4.1 a 4.4.7).

4.1 Entrevista com especialista

Nesta etapa do processo foi feita uma entrevista com a especialista no domínio Profª. M.e Madianita Bogo Marioti, que é professora na disciplina de lógica de programação, entre outras disciplinas na área de programação há mais de 20 anos. A entrevista foi realizada com o intuito de levantar os conceitos em que os alunos têm mais dificuldades na parte inicial do processo de formação em lógica de programação e o nível de relevância para cada assunto a ser abordado no jogo.

Foi discutido que existe certa dificuldade por parte dos alunos na interpretação do contexto dos problemas passados em sala de aula e do que é necessário para resolver tais problemas. Problemas como identificar as variáveis de entrada, e o que se esperar de resultado dentro do contexto ou problema apresentado, nesse caso as variáveis de saída.

Já para os tipos de dados, foi visto que uma abordagem em relação à diferenciação entre os tipos, como os valores que podem ser recebidos por cada um e como eles podem ser trabalhados. Essas são questões a serem tratadas e que podem ser inseridas no meio do contexto do jogo. Para laços de repetição, foi recomendado que fossem apresentadas as condições de parada para um laço de forma a ser perceptível e simples.

Por fim, dos operadores, os operadores relacionais e lógicos foram os selecionados para um enfoque, principalmente nos casos em que os dois são utilizados juntos e demonstrar quais os possíveis resultados obtidos e como chegar ao resultado certo. Porém não isentando a presença dos demais operadores no progresso do jogador, para um entendimento geral do funcionamento das funcionalidades básicas disponíveis na lógica de programação.

Esses pontos foram levantados com o objetivo do jogador se familiarizar com o funcionamento das estruturas e conseguir abstrair para fora do jogo, deixando de enxergar como uma simples mecânica de jogo e utilizar como ferramenta em seus programas.

Cumprindo o objetivo do jogo de auxiliar no aprendizado e abrindo espaço para o jogador se aprofundar mais em questões da área da computação.

4.2 Definição do jogo

Tendo em vista o objetivo do trabalho desenvolvido, foram dispostos como abordagem de construção do jogo os conceitos básicos de lógica de programação. Conceitos como algoritmos, operadores, variáveis, tipos de dados, condicionais e laços de repetições, foram previamente discutidos com a especialista no domínio e estabelecidas prioridades sobre cada conceito a ser apresentado.

O jogo desenvolvido tem como gênero RPG e *Puzzle*, onde o jogador possui elementos do RPG como ferramentas para passar por desafios de lógica voltados aos conceitos anteriormente definidos. A escolha do gênero RPG vem da ideia de aproximar o jogador do personagem, dando a ele a tarefa de controlar as ações do personagem e receber os desafios do personagem como seus. Já a escolha do gênero *puzzle* vem do intuito de desafiar o jogador a pensar em meios de superar obstáculos entendendo os artifícios que possui dentro de uma situação específica.

Para melhor compreensão do necessário para o desenvolvimento do jogo, foram estabelecidos os requisitos que foram tomados em conta, conforme na tabela 1 abaixo.

Tabela 1 - Requisitos levantados

REQUISITOS	**RELEVÂNCIA**		
	Desejável	Importante	Essencial
O primeiro cenário do jogo deve abordar conceitos de algoritmo			**X**
O segundo cenário do jogo deve abordar conceitos de variáveis			**X**
O terceiro cenário do jogo deve abordar conceitos de operadores			**X**

O quarto cenário do jogo deve abordar conceitos de tipos de dados			**X**
O quinto cenário do jogo deve abordar conceitos de condicionais			**X**
O sexto cenário do jogo deve abordar conceitos de laços de repetição			**X**
O jogo deve apresentar as melhores soluções possíveis para o jogador	**X**		
O jogo deve possuir um cenário de perguntas para validar o conhecimento do jogador		**X**	
O jogo deve estar disponível para sistemas operacionais Windows			**X**
O jogo deve possuir componentes de licença gratuita			**X**
O jogo deve ser executável sem conexão com internet		**X**	

Conforme apresentado na tabela 1, os requisitos tratam das necessidades do jogo quanto a sua jogabilidade, do que é visto dentro do jogo para que o mesmo cumpra seu objetivo. E também os requisitos que abordaram questões fora do ambiente de jogo, como o produto que ele é e o que é necessário para o seu funcionamento esperado.

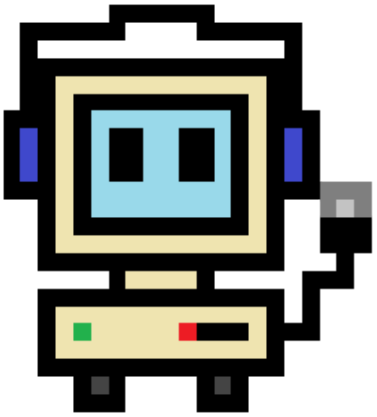
Foi desenvolvido também um *Game Development Design,* que é um documento descrevendo o jogo. O mesmo serve de apoio para o *design* e desenvolvimento do jogo, contendo questões sobre cenário, personagens, história, controles e sistemas. Este documento está presente na seção de apêndices, como apêndice 1.

4.2.1 Narrativa e Personagens

O cenário em que o jogo se passa é um mundo onde a sociedade é formada por humanos e máquinas trabalhando conjuntamente. No caminho de seu emprego, Makina sofre um acidente e sofre um curto-circuito, corrompendo todos os seus dados de configuração. Makina então é levada para um centro de recuperação onde é instruída a passar por uma bateria de desafios para aos poucos ir recuperando suas funcionalidades antigas.

Makina é uma máquina ainda operante no mercado de trabalho que tem grande apreço por suas conquistas e por seu lugar conquistado na sociedade. Após sofrer o acidente, Makina percebe que havia perdido grande parte do que tinha conquistado, mas após um choque inicial, ela logo se dispõe a recuperar o que perdeu e a não se abalar e seguir em frente. A aparência da personagem pode ser vista na figura 8, a seguir.

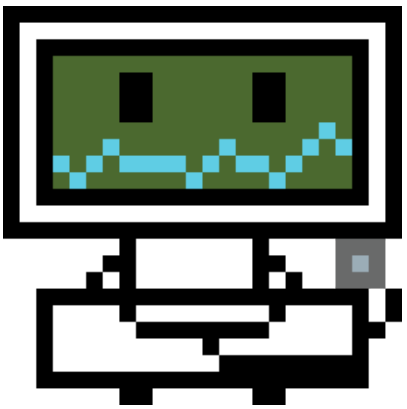
Figura 8 - *Sprite* da personagem Makina



A figura 8 demonstra como é a aparência da personagem Makina dentro de jogo, as animações para esta personagem foram criadas com essa figura como base e conta com vinte e quatro imagens diferentes para contemplar as direções e movimentos da personagem.

Dentro do centro de recuperação, Makina é apoiada por uma máquina chamada Doc, que trabalha no centro de recuperação e instrui Makina em meio aos seus desafios dentro do centro. Doc, funcionário do centro de recuperação, é uma máquina experiente no seu ramo, tendo passado por um grande número de situações e lidado com muitas outras máquinas em situação de reparo e até humanos. Doc repara os esforços de Makina e decide que irá com ela até o fim de sua recuperação, tendo um papel de guia na jornada do protagonista, ensinando e testando o que foi aprendido. A imagem do personagem Doc pode ser vista a seguir na figura 9.

Figura 9 - *Sprite* da personagem Doc



A figura 9 representa a imagem do Doc em jogo, o segundo personagem conta com dezoito imagens para criação de sua animação e teve como base para seu *design* e nome um médico. Ele está em todos os cenários para apoiar a personagem principal e explicar o que ela deve fazer.

O jogo dispõe de um personagem principal, Makina, sendo controlado pelo jogador, o permitindo protagonizar os acontecimentos dentro da história do jogo. E um NPC (*non-playable character,* que se traduz como personagem não-jogável) Doc, que é um personagem que não pode ser controlado pelo jogador mas interage com o mesmo, que tem um papel coadjuvante na narrativa, apoiando o personagem principal a alcançar seus objetivos.

4.2.2 Sistemas do Jogo

Com o objetivo de permitir que o jogador interaja com o personagem e atue dentro do cenário do jogo, foram criados alguns sistemas para permitir que isso seja possível. Em seguida estão presentes os sistemas projetados:

- **Sistema de variáveis:** esse sistema cria uma mecânica no jogo que permite ao jogador interagir com valores presentes dentro de cada cenário, armazená-los dentro das duas variáveis que ele possui em sua interface e/ou usá-los como entrada;
- **Sistema de restauração:** o jogador pode em meio a algum desafio optar por restaurar o cenário para o ponto inicial, permitindo partir do ponto zero e rejogar o desafio;
- **Sistema de retorno:** o jogador pode em meio a qualquer desafio optar por voltar para o cenário inicial, permitindo ele acessar o desafio atual ou os que ele já concluiu;

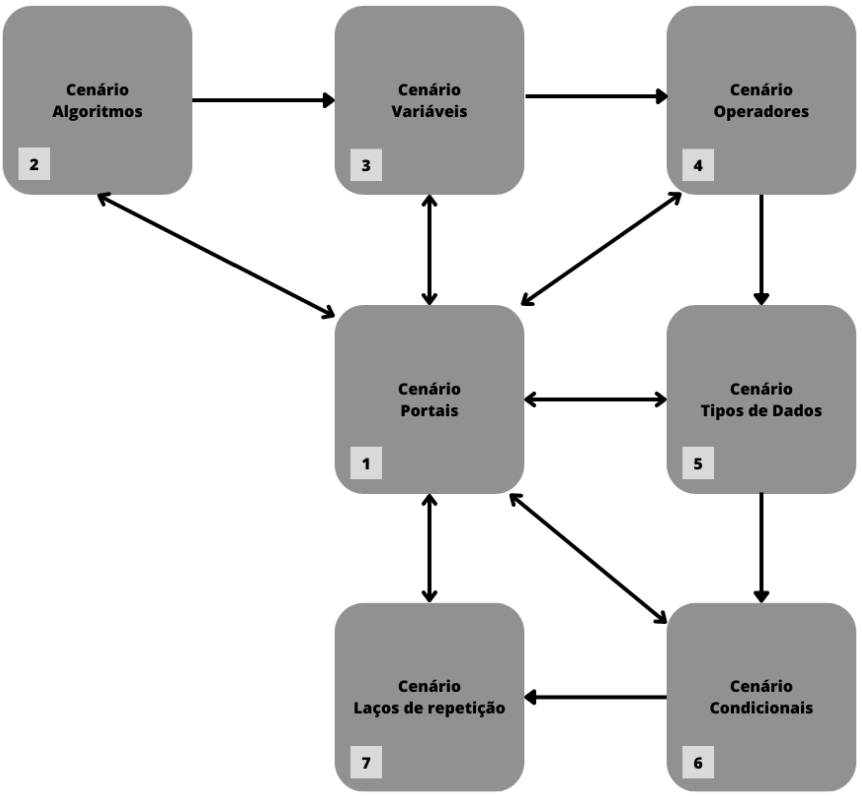
- **Sistema de desafios:** sistema responsável por apresentar os obstáculos a serem superados pelo jogador e por testar os conhecimentos adquiridos quanto aos conceitos abordados;
- **Sistema de chaves:** sistema responsável por, ao completar um desafio, o jogador receber uma chave que abrirá o portal na sala de portais, para a sala do próximo desafio;
- **Sistema de ações:** é o sistema responsável pelas ações do jogador, seja ela movimentar-se pelo cenário, interagir com objetos ou NPC's;
- **Sistema de apoio:** em alguns cenários o jogador contará com esse sistema que irá escrever instruções em sua tela para a conclusão do desafio do cenário.

Os sistemas serão apresentados dentro dos cenários como mecânicas e artifícios do usuário para completar os desafios propostos. O jogador receberá tutoriais do NPC Doc do funcionamento e aplicação de cada sistema, a partir do momento que eles puderem ser utilizados.

4.2.3 Cenários

O jogo possui sete cenários diferentes, ligados por portais que somente podem ser acessados após o jogador conseguir cumprir o desafio do cenário em que está. Progressivamente, em cada cenário um novo conceito será abordado, aumentando a complexidade dos desafios e o número de conceitos aprendidos necessários para completá-los. O jogador pode se mover pelos cenários progressivamente ou voltar para o primeiro cenário e acessar os cenários anteriores, conforme apresenta a Figura 10 a seguir.

Figura 10 - Fluxo de cenários e conceitos abordados



Conforme mostra a Figura 10, cada cenário do jogo terá um conceito abordado, com exceção do primeiro que servirá como um tutorial inicial, onde o jogador irá aprender os comandos básicos do jogo e um ponto de acesso entre os cenários, porém um cenário só poderá ser acessado caso ele seja o desafio atual do jogador ou o jogador já tenha cumprido o desafio dele.

4.2.4 Desafios

O jogo conta com sete cenários com desafios que devem ser cumpridos para avançar de um cenário para outro. Cada cenário possuirá explicações sobre os conceitos abordados e como eles serão tratados dentro do desafio. A seguir, a tabela 2 apresenta o funcionamento de cada desafio por cenário.

Tabela 2 - Cenários e desafios

Cenário	Conceito	Desafio
2	**Algoritmo**	O jogador terá que seguir as orientações de ações do NPC Doc para prosseguir em uma sala com paredes invisíveis, tendo que executar cada comando que Doc passar para ele para chegar ao fim da sala e concluir o desafio.
3	**Variável**	O jogador terá que interagir com consoles diferentes para receber variáveis e usá-las como entradas em outros consoles que pedem entradas específicas.
4	**Operadores**	O jogador terá que interagir com consoles pelo cenário para escolher os operadores corretos para usar como entrada em consoles com expressões para prosseguir pela sala e conseguir concluir o desafio.
5	**Tipos de dados**	O jogador irá acessar consoles para receber números inteiros, os quais deverá somar e transformar em strings, para então concatená-las e usar como entrada para concluir o

		desafio.
6	**Condicionais**	O jogador terá que avançar por uma sala de tapetes com portais que levam a direções diferentes de acordo com a condição da cor do tapete e conseguir atravessar a sala para concluir o desafio.
7	**Laços de repetição**	O jogador irá trabalhar com condições de parada para que os elementos do cenário que estão o impedindo de atravessar a sala, desapareçam. Inicialmente ele terá que trabalhar com valores em dois consoles e posteriormente executar uma repetição manualmente para terminar um loop e conseguir atravessar.
1	**-**	Sala com portais para os diferentes desafios, inicialmente utilizada como tutorial e acesso para o segundo cenário. Posteriormente o jogador pode voltar a ela e acessar as salas que ele já completou os desafios.

A tabela 2 demonstra como os cenários lidam com cada conceito a ser apresentado, descrevendo o desafio que o jogador terá que cumprir para concluir o cenário e avançar no jogo. Ao concluir o desafio o jogador terá acesso ao próximo cenário, e ao mesmo cenário novamente caso volte para o primeiro cenário utilizando o sistema de retorno.

A conclusão do jogo não segue uma linha de pontuação, e sim um caminhar pela narrativa do personagem principal. Ao deixar de concluir os desafios do jogo, o jogador fica privado de prosseguir na história e aprender novos conceitos.

4.2.5 Regras

Para o funcionamento e progressão corretos do jogo, o jogador terá que seguir as seguintes regras dentro de jogo:

1 - O avanço de cenários só é permitido quando o jogador cumpre o desafio do cenário em que está.

2 - Ao finalizar um cenário, o jogador poderá ter acesso a ele novamente via cenário 1.

3 - O acesso a um cenário só é permitido se o jogador já tiver completado o cenário de número anterior ao que deseja acessar, com exceção do cenário 1.

4 - O jogador não tem tempo limite por desafio e pode reiniciar o cenário quantas vezes ele quiser.

Ao seguir essas regras, o jogo terá uma progressão regular entre os cenários e delimita as ações e comportamentos do jogador e do jogo.

4.2.6 *Design* do Jogo

Com a definição dos elementos do jogo, foram prototipadas as interfaces do jogador para que sejam apresentadas informações visuais necessárias para o progresso do jogo. A interface em que o jogador possivelmente terá mais contato é a interface em jogo a seguir.

Figura 11 - Interface do jogador em jogo

A figura 11 representa os itens que estarão presentes na tela do jogador. Os elementos estão definidos como:

1. **Botão *Pause*:** representado pelo círculo amarelo com "*Pause"* escrito no meio dele no canto superior esquerdo da tela. Esse botão serve para suspender as ações do jogo, caso o jogador julgue necessário.
2. **Caixas de variáveis:** é uma interface onde o jogador irá armazenar valores ao decorrer do jogo, funcionando como itens obtidos e utilizados no cenário. São representadas por duas bolas cinzas com X e V no meio e caso o jogador tenha algum valor armazenado, itens com os dados são exibidos dentro desses espaços, visíveis no canto superior esquerdo da tela.
3. **Botão de interação:** esse botão aparece quando o jogador pode interagir com algum elemento do jogo. Representado por um círculo azul no canto superior esquerdo da tela.
4. **Caixa de diálogo:** Os diálogos do jogo são representados por textos dentro deste elemento. Representado por um retângulo de cor clara na parte inferior da tela, podendo aparecer no topo da tela, caso o personagem do jogador esteja ocupando a parte inferior.

Essas interfaces foram projetadas para remeter a jogos de RPG clássicos, como *Legend ot Zelda: A Link to the Past*, com caixas de itens na parte superior da tela e *Chrono*

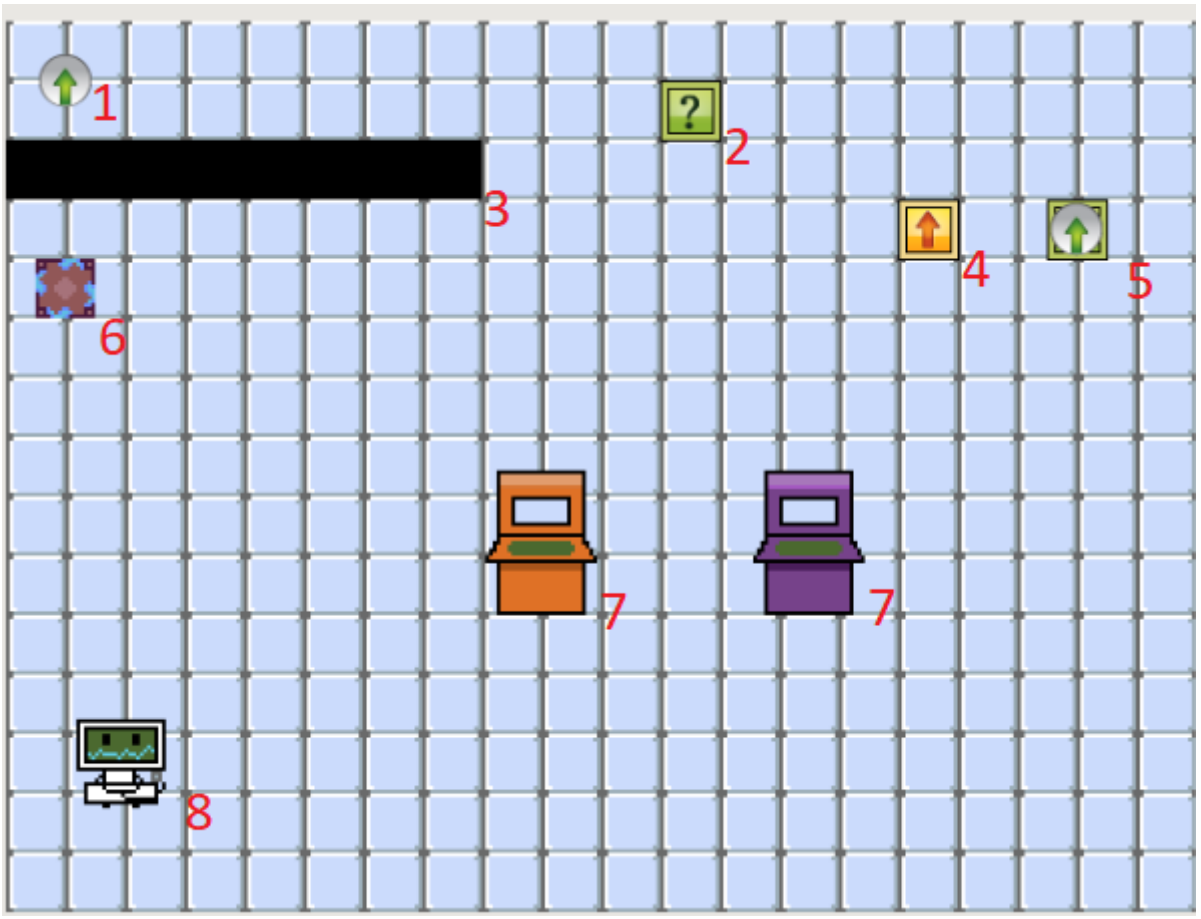
Cross, com caixas de diálogo na parte inferior. Isso foi feito para remeter a sistemas mais simples e, possivelmente familiares, para a aproximação do jogador com o jogo.

4.3 Prototipagem

A partir da definição do jogo foram criados protótipos para um cenário de jogo para executar as funções, carregar *sprites* e diálogos, servindo como protótipo para o desenvolvimento dos cenários que iriam fazer parte do jogo de fato. Esta etapa teve como objetivo validar os requisitos, telas e mecânicas definidos na etapa anterior.

Foi criado um mapa de testes para acessar com facilidade as funções desenvolvidas, desenhar o *layout* de mapas, verificar os funcionamentos de entidades dentro do jogo, como sensores, teleportes, encadeamento de diálogos, etc. Em seguida, o que foi testado e verificado como de acordo com os requisitos do jogo foi levado para dentro dos mapas que necessitavam desses recursos.

Figura 12 - Mapa de protótipos



Fonte: Autor

Na figura 12, é possível ver o mapa criado para os protótipos criados para o jogo, com as *sprites* e entidades que mais adiante seriam utilizados para a implementação do jogo. E na tabela 3 abaixo estão representadas as entidades enumeradas na figura 12, com o nome e função de cada uma delas. Elas foram criadas no cenário de protótipo para validação dos seus respectivos funcionamentos e interações com o jogador, para então serem aplicadas nos cenários nos quais eram necessárias.

Tabela 3 - Entidades do protótipo

Número da entidade	Nome da entidade	Função
1	Ponto de *Spawn*	Essa entidade serve como ponto de surgimento para o jogador, seja ao chegar no cenário ou seja ao entrar em um portal.
2	Sensor	Essa entidade é um sensor que é acionado quando há uma colisão do personagem do jogador com ela, acionando a função que foi estabelecida para o sensor.
3	Buraco	Essa entidade é um buraco onde ao ocorrer a colisão do personagem do jogador com ele, o jogador é enviado de volta para o ponto de *spawn* inicial do cenário.
4	Portal	Essa entidade é acionada quando há colisão do personagem do jogador com ela. Ela envia o personagem para um ponto de *spawn*, seja no mesmo cenário ou não.
5	Sensor e ponto de *spawn*	Essa entidade funciona como as entidades 1 e 2, porém com sobreposição. Quando o jogador entra no portalcom destino ao ponto de *spawn* dela, ele é transportado para cima de um sensor que já ativa uma função.

6	Portal com *sprite*	Essa entidade tem funcionamento igual a entidade 4, porém nesta foi carregada uma sprite com animação visível para o jogador.
7	Console	Essa entidade possui duas variações de cor e funciona com a interação do jogador com ela. Ela abre uma caixa de texto pré-definida e apresenta opções de escolha para o jogador. Essa entidade pode entregar valores para variáveis do jogador.
8	NPC	Essa entidade é um personagem não jogável que interage com o jogador de certa forma, podendo ter caixas de diálogo, validar informações e ativar funções.

A próxima etapa ficou responsável pelo desenvolvimento de todos os cenários, *sprites*, funções necessárias para cumprir com os apresentados na subseção 4.2.

4.4 Desenvolvimento

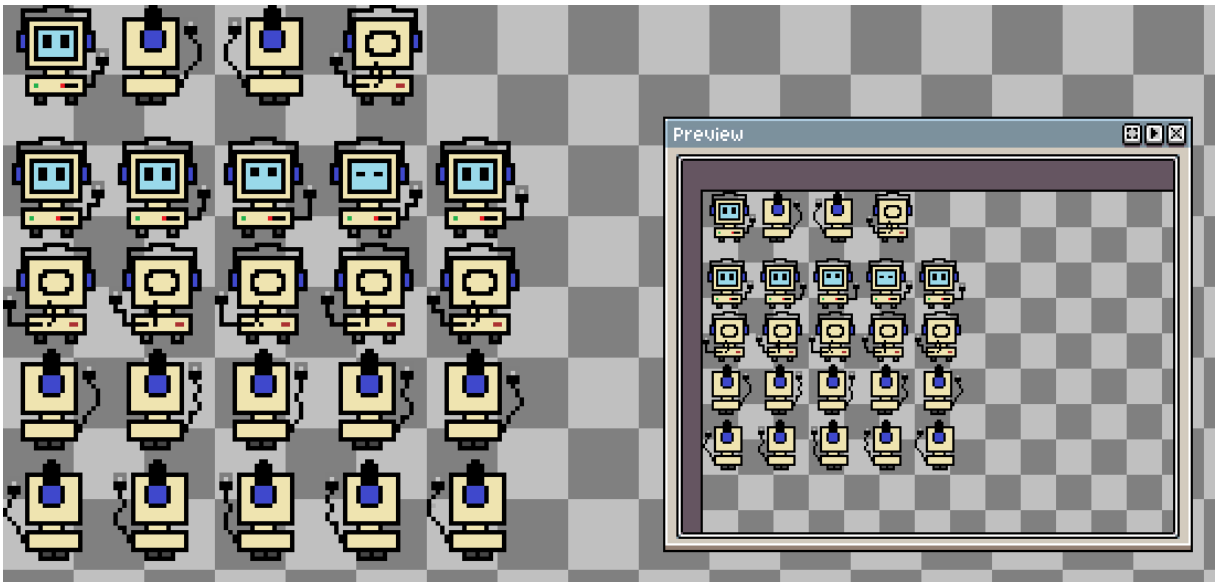
O desenvolvimento do jogo se iniciou com a criação das imagens do cenário na ferramenta *Aseprite* e carregados no motor de jogo *Solarus,* imagens como o *tileset* (pacote de texturas do cenário) das salas, *sprites* dos personagens e objetos que iriam compor o cenário.

Figura 13 - Tela de criação dos *tilesets*

Fonte: Autor

A figura 13 apresenta o *tileset* utilizado nos cenários do jogo, com os desenhos do piso, paredes, obstáculos, buraco e os tapetes utilizados no cenário 6. As texturas utilizadas nas salas dos cenários foram tiradas desse *tileset*. Na figura 14, está representada a *sprite* de animação da personagem principal do jogo, Makina.

Figura 14 - *Sprite* Makina



Fonte: Autor

Na figura 14 são apresentadas as *sprites* da Makina, com todas as direções possíveis dentro do ambiente do jogo e suas respectivas animações. Essa *sprite* foi carregada no jogo dentro da função `map:on_started()` dentro de cada mapa do jogo. As funções do jogo

são chamadas em função de eventos, no caso da função citada anteriormente, quando o mapa é iniciado e carregado para o jogador, o bloco de código dentro dela é executado.

Figura 15 - Trecho de código do script do cenário 4

```
16 function map:on_started()
17   local item = game:get_item("var_nula")
18   item:get_game():set_value(item:get_savegame_variable(), 1)
19   game:set_item_assigned(1, item)
20   game:set_item_assigned(2, item)
21   hero:set_tunic_sprite_id("hero/makina")
22   game:start_dialog("map4.initial")
23 end
```

Fonte: Autor

Na figura 15, é exibido um trecho de código do script do cenário 4 de tipos de dados, nele há um exemplo da função citada anteriormente. Dentro dela é iniciado um item na linha (17) com valor e *sprite* nulos, para ser designado para as caixas de variáveis do jogador nas linhas (19) e (20). Na linha (21) é onde a *sprite* do personagem Makina é carregada e na linha (22), o diálogo inicial que aparece quando o jogador entra no cenário 4, é chamado.

Figura 16 - Trecho de código do script para sistemas de retorno e restauração

```
31 function map:on_key_pressed()
32   if sol.input.is_key_pressed("r") then
33     if game:is_dialog_enabled() then
34       game:stop_dialog()
35     end
36
37     game:start()
38   end
39   if sol.input.is_key_pressed("p") then
40     if game:is_dialog_enabled() then
41       game:stop_dialog()
42     end
43     hero:teleport("map_portais")
44   end
45 end
```

Fonte: Autor

Na figura 16 é apresentada a função que é chamada quando o jogador pressiona certas teclas para utilizar os sistemas de retorno e restauração apresentados na subseção 4.2.2. Essa função chama a função `game:start()` quando o jogador quer restaurar a sala e iniciá-la novamente, e a função `hero:teleport("map_portais")` para enviar o jogador para o cenário de portais, onde terá acesso a outras salas.

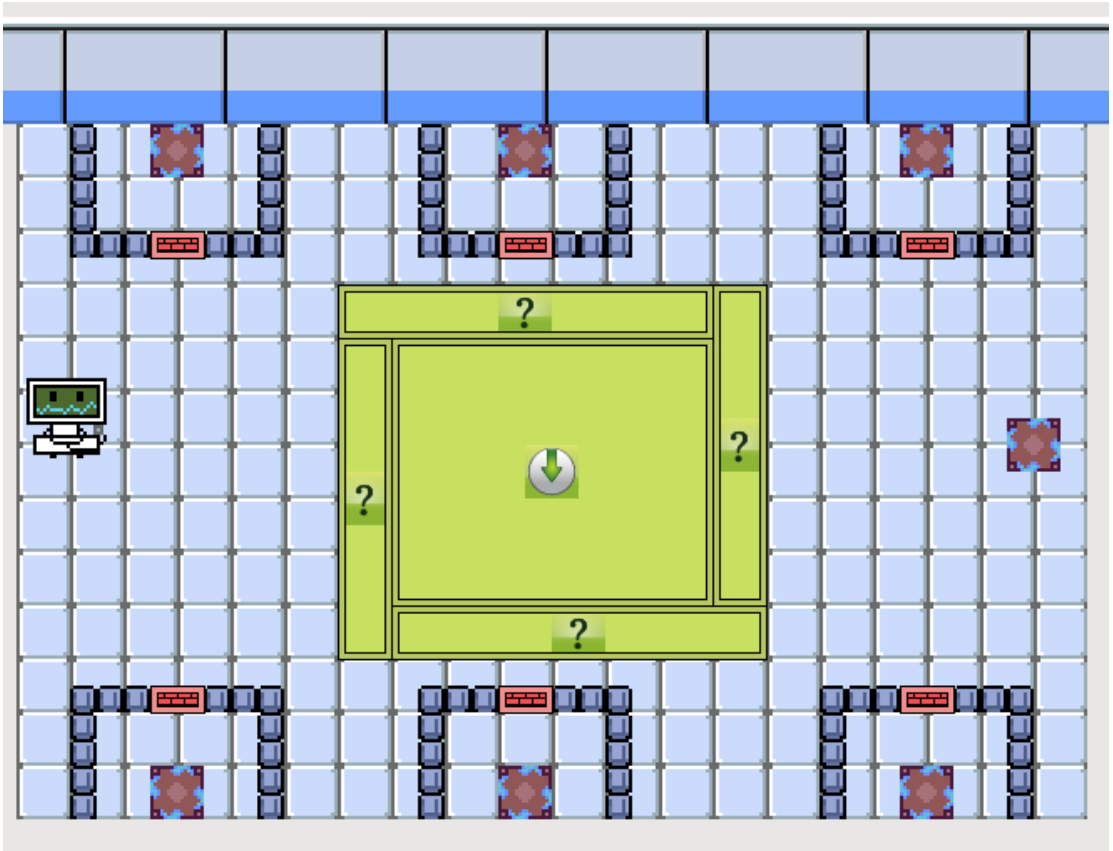
Figura 17 - Trecho de código do script para sistema de chaves

```
251 function sensor_chave5:on_activated()
252   local item = game:get_item("chave5")
253   item:get_game():set_value(item:get_savegame_variable(), 1)
254 end
```

Fonte: Autor

A figura 17 mostra um exemplo de função responsável pelo *script* para o sistema de chaves. Ao fim de cada sala, o jogador recebe um item chave para abrir, na sala de portais, o portal para a sala do próximo desafio. Ao chegar na sala de portais, uma entidade do tipo sensor, vista anteriormente na prototipagem, verifica se o jogador possui as chaves para cada portal e, caso tenha, o portal para a respectiva chave é liberado.

Figura 18 - Cenário da sala de portais no modo edição



Fonte: Autor

A figura 18 apresenta a sala de portais no modo de desenvolvimento, o modo utilizado para criar, posicionar e configurar as entidades do jogo, nele todos os elementos são visíveis, inclusive os elementos que são invisíveis enquanto em jogo. Em verde estão os sensores anteriormente citados. O sensor no centro verifica se é a primeira vez que o jogador está na

sala e se o jogador possui as chaves para liberar os portais. Os sensores externos, identificam quando o jogador se move sobre ele e, caso seja a primeira vez do jogador na sala, inicia um diálogo de tutorial e entrega a chave do primeiro portal.

Figura 19 - Trecho de código do script de um console que atribui valor

```
137 function console_apple_g:on_interaction()
138   game:set_item_assigned(1, nil)
139   local variavel =  game:get_item_assigned(1)
140   if variavel ~= nil then
141     variavel:get_game():set_value(variavel:get_savegame_variable(), 0)
142   end
143   game:start_dialog("map2.strings", function(answer)
144     if answer == 1 then
145       local item = game:get_item("string_correta")
146       item:get_game():set_value(item:get_savegame_variable(), 1)
147       game:set_item_assigned(1, item)
148     elseif answer == 2 then
149       local item = game:get_item("string_incorreta")
150       item:get_game():set_value(item:get_savegame_variable(), 1)
151       game:set_item_assigned(1, item)
152     end
153   end)
154 end
```

Fonte: Autor

A figura 19 corresponde a um exemplo de função para consoles que atribuem valor para as variáveis do jogador ao interagir com ele. Essa função inicia diálogos com opções de escolha e atribui valores à variável de acordo com o selecionado.

Figura 20 - Trecho de código do script do sistema de apoio

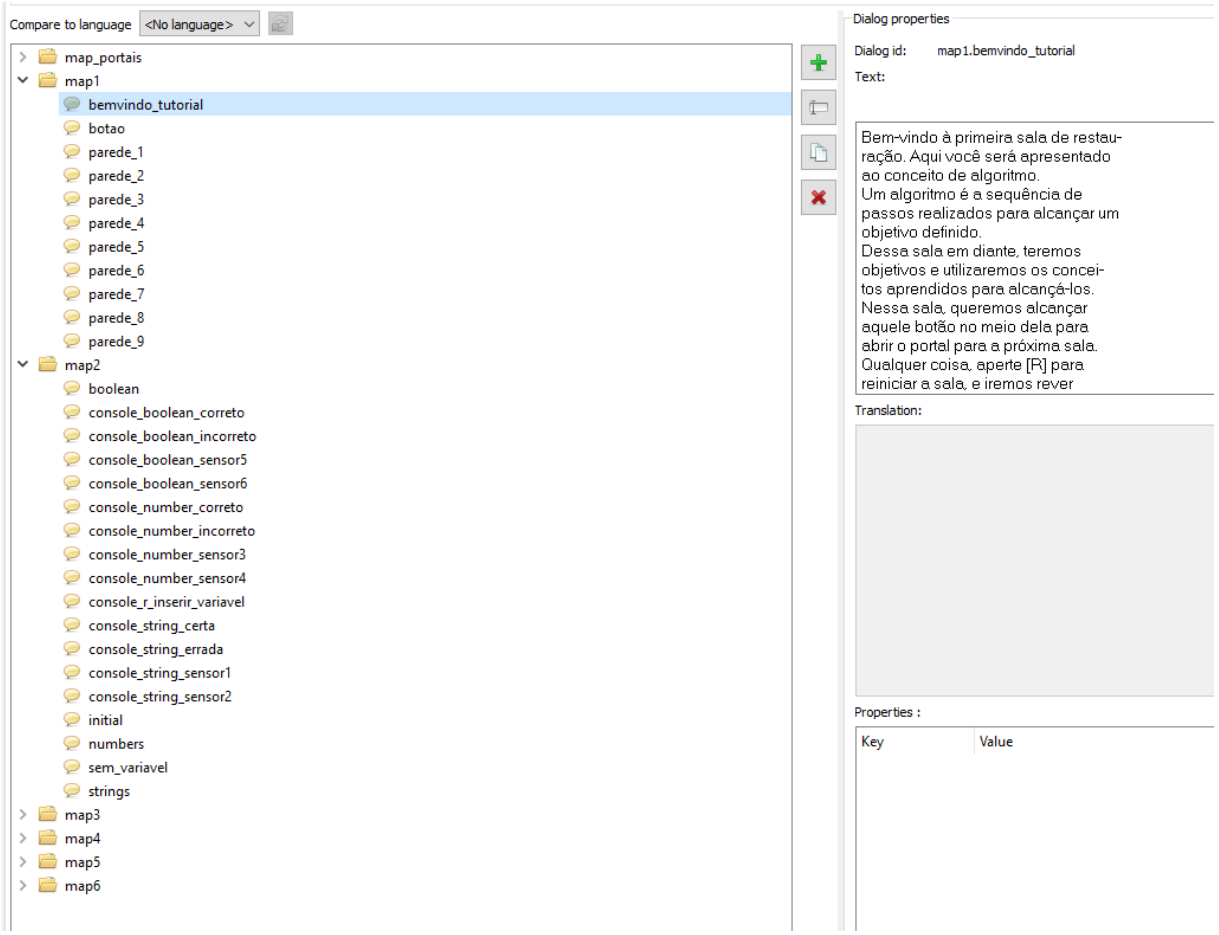
```
36 function map:on_key_pressed()
37   if sol.input.is_key_pressed("h") then
38     orientacoes_a = sol.text_surface.create({text = text1})
39     orientacoes_b = sol.text_surface.create({text = text2})
40     orientacoes_c = sol.text_surface.create({text = text3})
41     orientacoes_d = sol.text_surface.create({text = text4})
42     orientacoes_e = sol.text_surface.create({text = text5})
43     function map:on_draw(screen)
44       orientacoes_a:draw(screen, 5, 70)
45       orientacoes_b:draw(screen, 5, 80)
46       orientacoes_c:draw(screen, 5, 90)
47       orientacoes_d:draw(screen, 5, 100)
48       orientacoes_e:draw(screen, 5, 110)
49       orientacoes_a:set_opacity(255)
50       orientacoes_b:set_opacity(255)
51       orientacoes_c:set_opacity(255)
52       orientacoes_d:set_opacity(255)
53       orientacoes_e:set_opacity(255)
54     end
55   end
```

Fonte: Autor

A figura 20 retrata a função para o sistema de apoio, que assim como na figura 16, está dentro da função `map:on_key_pressed()` que atua com o pressionar de uma tecla específica. Nesse caso, quando o jogador pressiona a tecla “h” são criados e desenhados textos na tela.

Os diálogos em meio ao jogo são chamados pela função `game:start_dialog("nomeDialogo")` e são escritos em outra área da ferramenta que é demonstrada na figura 21 a seguir.

Figura 21 - Tela diálogos



Fonte: Autor

A figura 21 exibe a tela de diálogos, onde os diálogos do jogo estão presentes e organizados por cenário. Cada diálogo apresentado no jogo está nessa tela e seus *id's* são utilizados quando chamada a função citada anteriormente que serve para iniciar um diálogo. Os diálogos podem ter *keys* e traduções, e no jogo deste projeto, quando necessário um diálogo com opções, a *key question* foi utilizada para indicar perguntas e disponibilizar opções para o jogador escolher.

4.4.1 Cenário 1

Para o primeiro cenário, o cenário dos portais, foram utilizadas funções para verificações utilizando sensores, seja para averiguar se é a primeira vez do jogador nessa tela, seja para verificar se o jogador possui chaves para abrir os portais. Isso foi feito com a função `game:has_item()`, caso o jogador já tivesse passado pela sala uma primeira vez, ele já teria a chave para o primeiro portal, nesse caso o sensor verifica se o jogador possui a primeira chave e não exibe diálogos iniciais, nem diálogos de tutorial.

Figura 22 - Trecho de código do script para verificar chave

```
83  function sensor_ver_portais:on_activated()
84
85    if game:has_item("chave1") then
86      map:get_entity("parede_1"):set_enabled(false)
87      sensor_initial2_1:set_enabled(false)
88      sensor_initial2_2:set_enabled(false)
89      sensor_initial2_3:set_enabled(false)
90      sensor_initial2_4:set_enabled(false)
91
```

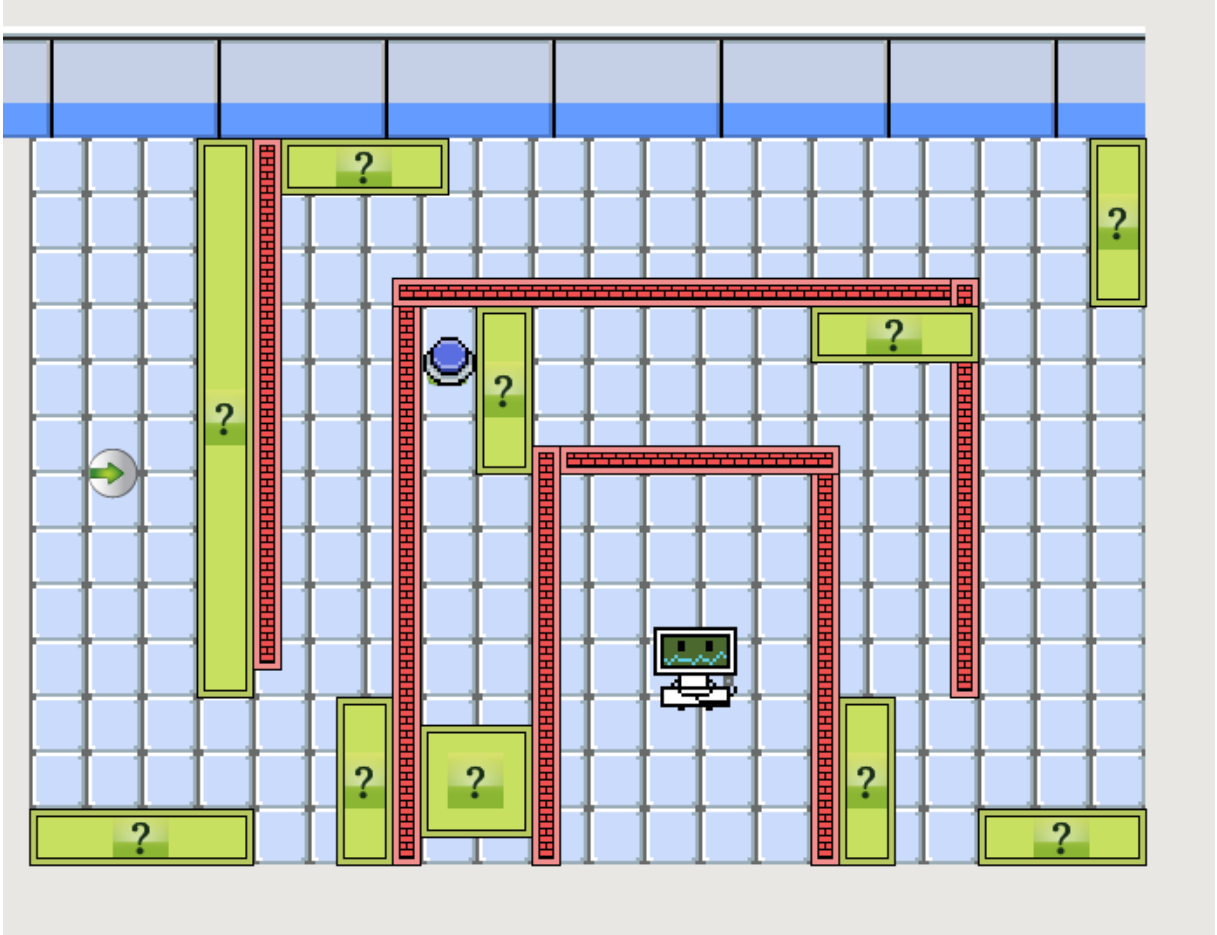
Fonte: Autor

O script desse cenário se baseia basicamente nos sensores checando se o jogador possui chaves e utilizando funções `set_enabled()` para desativar as portas e liberar os portais.

4.4.2 Cenário 2

O cenário dois trabalha com algoritmos e funciona com base em sensores que dão instruções para o jogador conseguir passar por um labirinto invisível. Ao atravessar o labirinto o jogador pode interagir com um botão e abrir o portal para o próximo desafio.

Figura 23 - Cenário 2 no modo edição



Fonte: Autor

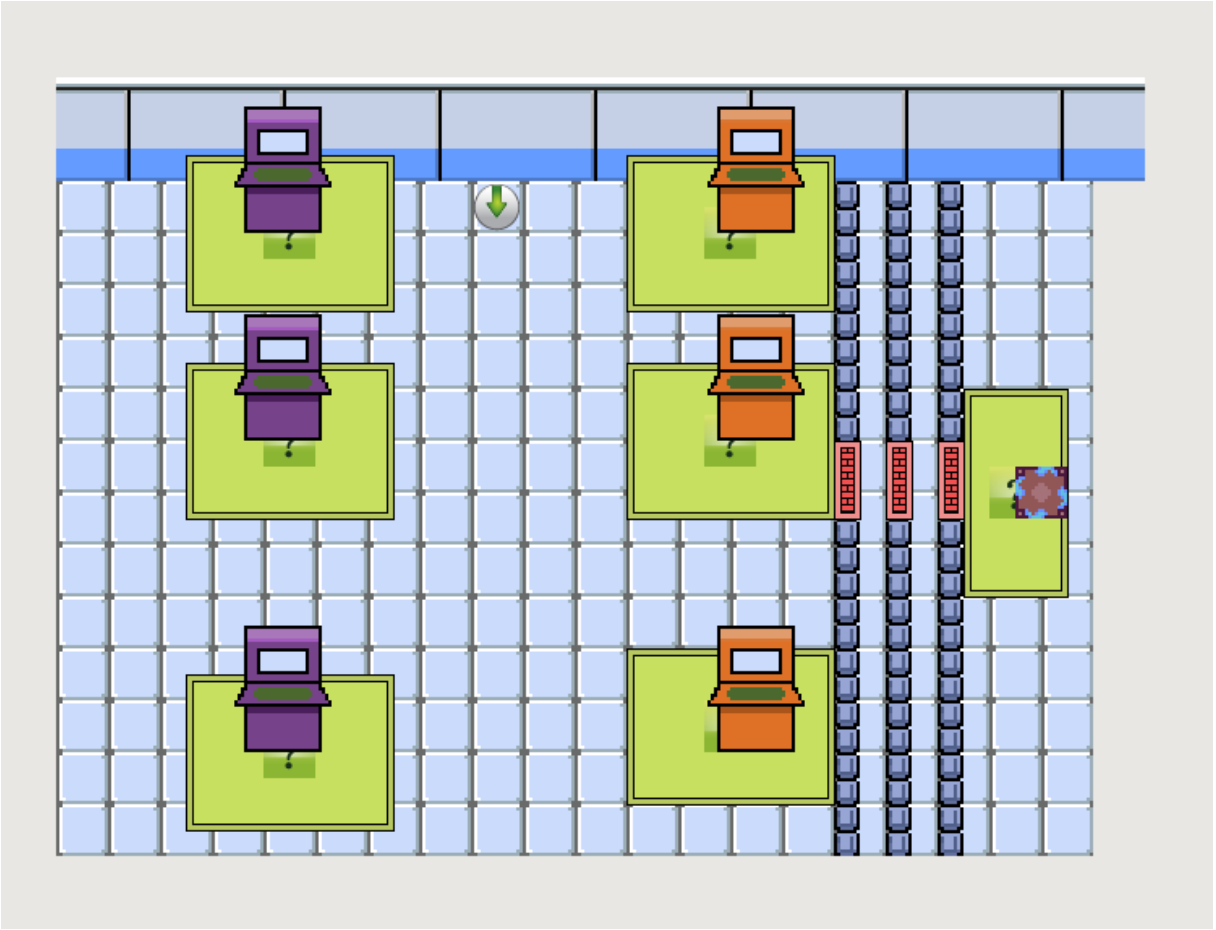
Na figura 23, estão visíveis as paredes que irão criar o labirinto invisível quando executado o jogo. Também, é possível ver a disposição dos sensores para que o jogador consiga andar em linhas retas e conseguir concluir o desafio do cenário.

Este cenário foi construído para que o jogador não consiga o concluir sem seguir uma sequência de passos correta, que são entregues a ele conforme avança pelo Doc. Remetendo à sequência de passos para alcançar um objetivo que são os algoritmos.

4.4.3 Cenário 3

O cenário 3, trata de variáveis e como é possível observar na figura 24 a seguir, trabalha com consoles que são acionados pela função `on:interaction()` que por sua vez são acionadas quando o jogador interage com a entidade responsável pela função.

Figura 24 - Cenário 3 no modo edição



Fonte: Autor

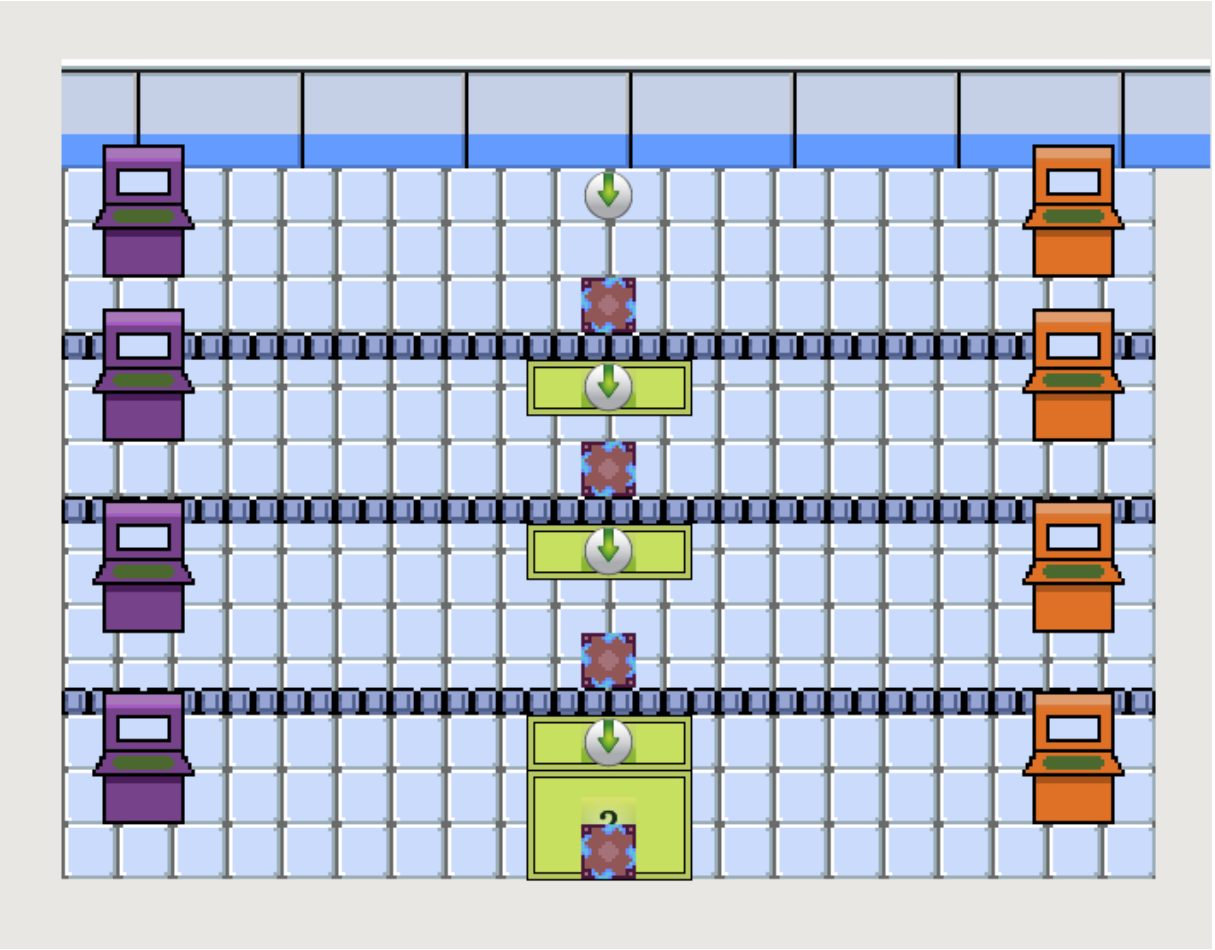
Na figura 24 é possível perceber que a sala possui sensores (em verde), que servem para instruções do funcionamento dos consoles, que por sua vez podem sofrer interações por parte do jogador e acionar diálogos, atribuir valores para variáveis ou checar valores para abrir as paredes para o portal para a próxima sala.

Este cenário apresenta ao jogador propriedades das variáveis, como valores podem ser recebidos e substituídos numa mesma variável, como podem ser apresentados como entrada em algumas situações e que situações diferentes esperam valores de variáveis diferentes.

4.4.4 Cenário 4

O cenário 4 exibe o mesmo tipo de mecânicas que os consoles do cenário anterior possui, atribuindo valores e checando valores. Este é o cenário para operadores.

Figura 25 - Cenário 4 no modo edição



Fonte: Autor

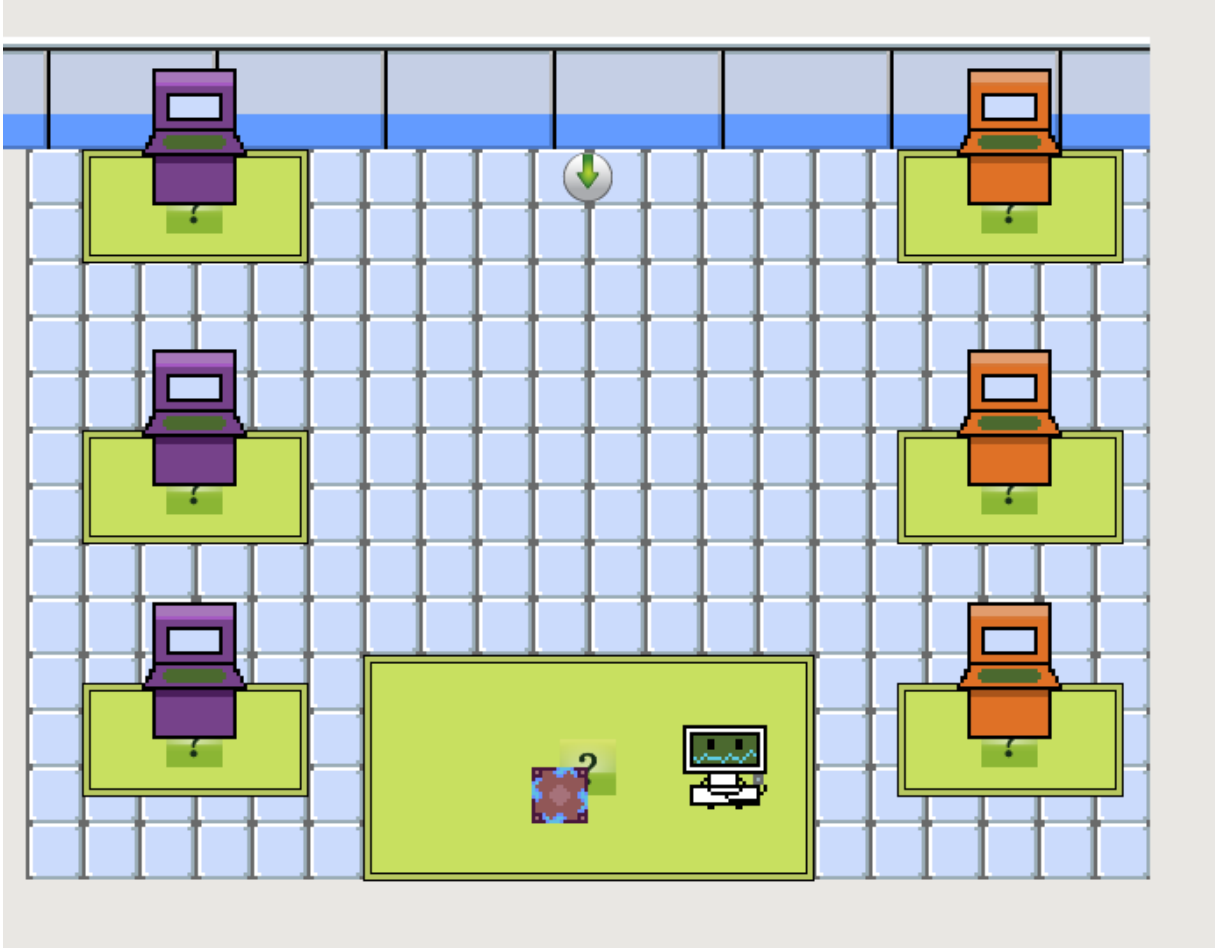
Os portais exibidos na figura 25. só são ativados quando os consoles que checam valores passam na verificação de de valores, enviando o jogador para o outro lado do obstáculo e iniciando um novo diálogo via sensor.

Este cenário apresenta ao jogador os diferentes tipos de operadores, qual é o funcionamento deles e como são aplicados em diferentes expressões. O jogador também trabalha com tipos de entradas e resultados esperados neste cenário, ao interagir com operadores corretos ou incorretos para as expressões.

4.4.5 Cenário 5

O cenário 5, ou cenário dos tipos de dados, possui mecânicas de soma e concatenação. Ele também possui um console que funciona como uma variável externa.

Figura 26 - Cenário 5 no modo edição



Fonte: Autor

Como exibe a figura 26, a mecânica do cenário 5 também se baseia em sensores e consoles para a realização do desafio.

Este cenário trabalha com dois tipos de dados, inteiros e *strings*, para que o jogador compreenda que para cada tipo de dado há diferentes tipos de operações e tratamentos. O jogador também compreende que valores podem ser parecidos visualmente mas pertencer a tipos de dados diferentes.

4.4.6 Cenário 6

O cenário 6, ou cenário dos condicionais, possui uma mecânica de portais, que funcionam diferente para cada cor de tapete. A configuração dos portais foi feita na própria entidade e não via *script*.

Figura 27 - Cenário 6 no modo edição

Fonte: Autor

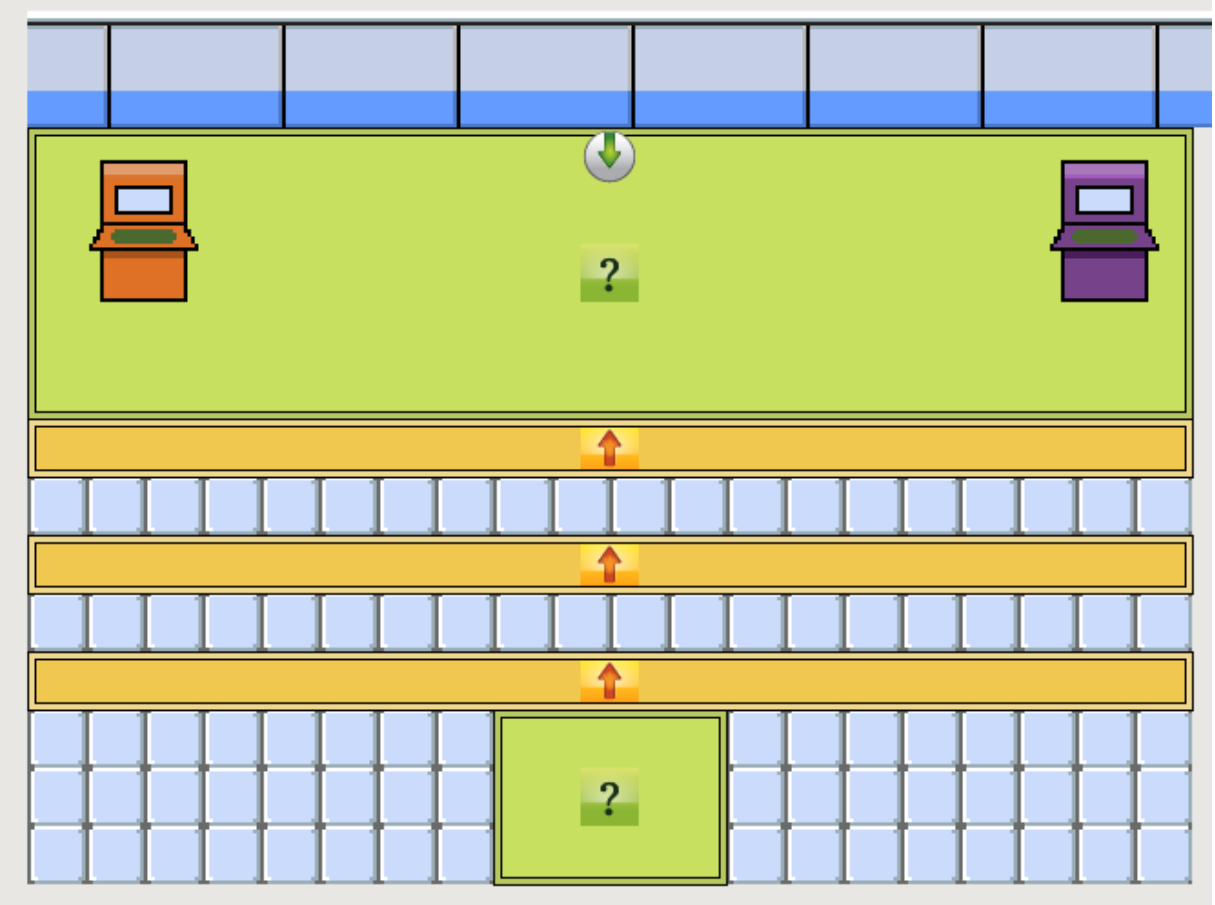
A figura 27 permite a observação dos pontos de destino dos portais, visíveis somente no modo edição do cenário. O destino de cada portal foi inserido na entidade, e foram postas paredes invisíveis entre os tapetes de portais para que o jogador não consiga cair no buraco ao redor dos tapetes para não gerar um reinício indesejado de cenário.

Este cenário opera com uma abordagem visual de condicionais e seus resultados, tendo como condicional a cor de cada tapete em que o jogador puder se encontrar e como resultado o destino de cada portal dentro da área do tapete.

4.4.7 Cenário 7

O cenário 7, responsável pelos laços de condição, se baseia em sensores e consoles, porém contém uma mecânica única do cenário, que é um contador de quedas, necessário para a finalização do desafio deste cenário.

Figura 28 - Cenário 7 no modo edição



Fonte: Autor

A figura 28 ilustra o cenário e como os portais (em amarelo) foram utilizados como “buracos” para que o jogador pudesse cair e voltar para que a mecânica do cenário fosse possível de ser executada.

Este cenário trabalha com condições de parada para os laços, apresentando ao jogador formas de parar laços ao alterar valores de expressões, ou a completar os laços, realizando todas as iterações em que ele foi programado.

5 CONSIDERAÇÕES FINAIS

Jogos são formas alternativas de ensinar algo, pois de acordo com Crawford (1984) todos os jogos são de alguma maneira educacionais, e que os mesmos são o meio mais antigo e consagrado de educação. E quando se trata de aprender algo novo, o engajamento do aluno é importante para que haja uma aquisição de conhecimento mais suave e prazerosa. E conforme Kapp (2013) o uso de jogos encoraja o aprendiz, pois desafios, objetivos e progressão são peculiaridades que engajam e encorajam o ser humano.

Portanto, esse projeto teve como objetivo o desenvolvimento de um jogo educacional para o apoio do aprendizado de lógica de programação. Para que o resultado final fosse alcançado, os passos da seção 3.2 foram realizados. Primeiramente, ocorreu uma coleta de dados com uma especialista do domínio que atua na área de ensino de lógica de programação para esclarecimento de pontos de dificuldade e pontos relevantes a serem abordados, para então ser feita a definição do jogo e enfim o desenvolvimento do resultado final.

Para a produção do jogo foram utilizadas as ferramentas *Solarus* e *Aseprite*, sendo a primeira um motor de jogo, que serviu de kit de desenvolvimento para o jogo, trazendo as ferramentas necessárias em seu pacote. Já a segunda foi utilizada para a produção das imagens que seriam utilizadas no jogo.

Como resultado final, obteve-se o jogo Aprendizado de Makina, um jogo educativo de gênero RPG e *Puzzle*, no qual o jogador controla o personagem Makina para cumprir desafios passados entre cenários diferentes, que abordam conceitos de lógica de programação como algoritmos, variáveis, operadores, tipos de dados, condicionais e laços de repetição.

O objeto principal do jogo é o apoio no aprendizado dos conceitos básicos de lógica de programação, em razão disso, foram desenvolvidos cenários, diálogos e desafios para abordar esses conceitos e instigar o jogador a aprendê-los.

O jogo conta com jogabilidade para apenas um jogador e foi desenvolvido para ser executado em computadores pessoais e não necessita de acesso à internet para sua execução, e ainda não possui meios de distribuição acessíveis a todos.

Para trabalhos futuros, seria pertinente a criação de uma função para salvar o progresso do jogador, para que ele possa voltar a qualquer momento para o ponto de salvamento. A criação de música de fundo e efeitos sonoros para uma melhor experiência de usuário. Também o desenvolvimento de um menu inicial para o jogo, com opções de configuração de tamanho de tela, volume de música e efeitos, carregar jogos salvos e finalizar o jogo.

Posteriormente, novos desafios e mecânicas abordando os mesmos temas ou temas diferentes para enriquecer a esfera de conhecimento que o jogo disponibiliza, envolveria pessoas de diferentes públicos, expandindo o número de jogadores e aplicações do jogo. Bem como um estudo sobre a eficácia do uso do jogo no aprendizado em diferentes níveis de conhecimento e de graduação para melhor direcionamento do uso da ferramenta desenvolvida.

Acredita-se que com essas propostas a experiência do jogador seja aprimorada, o alcance de conhecimento dentro da área de lógica de programação aumente e que a ferramenta seja indicada para usuários de nível e conhecimento compatíveis com ela.

REFERÊNCIAS

ALMEIDA, Marilane. **Curso essencial de lógica de programação**. Universo dos Livros Editora, 2008.

ASEPRITE. Disponível em: https://https://www.aseprite.org. Acesso em: 08/12/2022.

BATTAIOLA, André L. Jogos por computador–histórico, relevância tecnológica e mercadológica, tendências e técnicas de implementação. In: **Anais da XIV JORNADA DE ATUALIZAÇÃO EM INFORMÁTICA**, SBC, 2000, Florianópolis. Florianópolis: Csbc, 2000. p. 83-122.

BROOKHAVEN NATIONAL LABORATORY. Disponível em: https://www.bnl.gov/about/history/firstvideo.php. Acesso em: 05 dez. 2021 .

CAILLOIS, Roger. **Os jogos e os homens: a máscara e a vertigem**. Editora Vozes Limitada, 2017.

CALDERARO, Marilene Munguba et al. Jogos eletrônicos: apreensão de estratégias de aprendizagem. **Revista Brasileira em Promoção da Saúde**, v. 16, n. 2, p. 39-48, 2003.

CODE.ORG - LEARN TODAY, BUILD A BRIGHTER TOMORROW. Disponível em: https://code.org. Acesso em: 05 dez. 2021.

COMPUTAÇÃO - CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 2010, Belo Horizonte. Belo Horizonte: 2010. p. 981-990.

CRAWFORD, Chris. **The art of computer game design**. 1984.

DE ALMEIDA, Eliana S. et al. AMBAP: um ambiente de apoio ao aprendizado de programação. In: **Anais do XXII Congresso da Sociedade Brasileira de Computação.** 2002. p. 79-88.

DE MACEDO, Lino; PETTY, Ana Lúcia Sícoli; PASSOS, Norimar Christe. **Os jogos e o lúdico na aprendizagem escolar**. Artmed Editora, 2009.

DONOVAN, Tristan. **Replay: The history of video games**. Yellow Ant, 2010.

FERREIRA, Cláudia; GONZAGA, Flávio; SANTOS, Rodrigo. Um estudo sobre a aprendizagem de lógica de programação utilizando programação por demonstração. In: **XVIII Anais do Workshop sobre Educação em Computação**. 2010.

FORBELLONE, André Luiz Villar; EBERSPACHER, Henri Frederico. **Lógica de Programação: A Construção de Algoritmos e Estruturas de Dados**. 3. ed. Belo Horizonte: Pearson Universidades, 2005.

GRÜBEL, Joceline Mausolff; BEZ, Marta Rosecler. Jogos Educativos. **Renote - Revista Novas Tecnologias na Educação**, Porto Alegre, v. 4, n. 2, p. 2-6, dez. 2006.

GREGORY, Jason. **Game Engine Architecture**. A K Peters/Crc Press, 2009.

HAGUENAUER, Cristina Jasbinscheck et al. Uso de jogos na educação online: a experiência do LATEC/UFRJ. **Revista Educaonline**, v. 2, p. 1-11, 2008.

HERZ, Jessie Cameron. **Joystick nation: how videogames gobbled our money, won our hearts, and rewired our minds**. Abacus, 1997.

HUIZINGA, Johan. **Homo ludens: o jogo como elemento da cultura**. Editora da Universidade de S. Paulo, Editora Perspectiva, 2007.

JOGOS DE MATEMÁTICA - ADIÇÃO E SUBTRAÇÃO, CONTAGEM. Google Play. Disponível em: https://play.google.com/store/apps/details?id=com.rvappstudios.math.kids.counting&hl=pt_BR&gl=US. Acesso em: 05 dez. 2021.

JUUL, Jesper. Half-real. **Video games between real rules and fictional worlds**, 2005.

KAPP, Karl M. **The gamification of learning and instruction fieldbook: Ideas into practice**. John Wiley & Sons, 2013.

KIRRIEMUIR, John; MCFARLANE, Angela. **Literature review in games and learning**. 2004.

LOWOOD, Henry. Videogames in Computer Space: The Complex History of Pong.

MATTOS, Mauro M. Construção de abstrações em Lógica de Programação. In: **Anais do XXI Congresso da Sociedade Brasileira de Computação, Fortaleza**. 2001.

MCGONIGAL, Jane. **Reality is broken: Why games make us better and how they can change the world**. Penguin, 2011.

MEDINA, Marco; FERTING, Cristina. **Algoritmos e programação: teoria e prática**. Novatec Editora, 2006.

MORAES, Paulo Sérgio. Curso básico de lógica de programação. **Campinas: Centro de Computação**, 2000.

MORETTO, Vasco Pedro. **Construtivismo: a produção do conhecimento em aula**. DP & A, 2000.

MORTARI, Cezar A.. **Introdução à lógica**. São Paulo: Unesp, 2001.

PEREIRA JÚNIOR, José Carlos Rocha; RAPKIEWICZ, Clevi Elena. O processo de ensino-aprendizagem de fundamentos de Programação: uma visão crítica da pesquisa no Brasil. In: **Anais do XII Workshop sobre Educação em Computação (SBC)**. 2004. p. 4.

RIBEIRO, Paula Ceccon; MARTINS, Carlos Bazílio; BERNARDINI, Flávia Cristina. A Robótica como Ferramenta de Apoio ao Ensino de Disciplinas de Programação em Cursos de Computação e Engenharia. In: **Anais do Workshop de Informática na Escola**. 2011. p. 1108-1117.

SANTOS, Rodrigo P. et al. Uma Proposta de Cenário para Ensino de Algoritmos e Programação com Contribuições de Cooperação, Colaboração e Coordenação. In: **Anais do XVI Workshop sobre Educação em Computação–WEI**. 2008. p. 218-227.

SCRATCH - IMAGINE, PROGRAM, SHARE. Disponível em: https://scratch.mit.edu. Acesso em: 05 dez. 2021.

SIMPLER - APRENDER INGLÊS É MAMÃO COM AÇÚCAR. Google Play. Disponível em: https://play.google.com/store/apps/details?id=ru.zengalt.simpler&hl=pt_BR&gl=US. Acesso em: 05 dez. 2021.

SOLARUS - AN ARPG GAME ENGINE. Disponível em: https://www.solarus-games.org. Acesso em: 05 dez. 2021.

XAVIER, Gley Fabiano Cardoso. **Lógica de programação**. Senac, 2018.

APÊNDICE 1 - GAME DEVELOPMENT DOCUMENT

Aprendizado de Makina

Game Design Document

Versão: 1.0

Autores:

Henrique Ferreira Nogueira

Palmas,
Dezembro de 2022

Índice

1. História

O cenário em que o jogo se passa é um mundo onde a sociedade é formada por humanos e máquinas trabalhando conjuntamente. No caminho de seu emprego, Makina sofre um acidente e sofre um curto-circuito, corrompendo todos os seus dados de configuração. Makina então é levada para um centro de recuperação onde é instruída a passar por uma bateria de desafios para aos poucos ir recuperando suas funcionalidades antigas.

Makina é uma máquina ainda operante no mercado de trabalho que tem grande apreço por suas conquistas e por seu lugar conquistado na sociedade. Após sofrer o acidente, Makina percebe que havia perdido grande parte do que tinha conquistado, mas após um choque inicial, ela logo se dispõe a recuperar o que perdeu e a não se abalar e seguir em frente.

Dentro do centro de recuperação, Makina é apoiada por uma máquina chamada Doc, que trabalha no centro de recuperação e instrui Makina em meio aos seus desafios dentro do centro. Doc, funcionário do centro de recuperação, é uma máquina experiente no seu ramo, tendo passado por um grande número de situações e lidado com muitas outras máquinas em situação de reparo e até humanos. Doc repara os esforços de Makina e decide que irá com ela até o fim de sua recuperação, tendo um papel de guia na jornada do protagonista, ensinando e testando o que foi aprendido.

2. Sistemas do jogo

- **Sistema de variáveis:** esse sistema cria uma mecânica no jogo que permite ao jogador interagir com valores presentes dentro de cada cenário, armazená-los dentro das duas variáveis que ele possui em sua interface e/ou usá-los como entrada;

- **Sistema de restauração:** o jogador pode em meio a algum desafio optar por restaurar o cenário para o ponto inicial, permitindo partir do ponto zero e rejogar o desafio;

- **Sistema de retorno:** o jogador pode em meio a qualquer desafio optar por voltar para o cenário inicial, permitindo ele acessar o desafio atual ou os que ele já concluiu;

- **Sistema de desafios:** sistema responsável por apresentar os obstáculos a serem superados pelo jogador e por testar os conhecimentos adquiridos quanto aos conceitos abordados;

- **Sistema de chaves:** sistema responsável por, ao completar um desafio, o jogador receber uma chave que abrirá o portal na sala de portais, para a sala do próximo desafio;

- **Sistema de ações:** é o sistema responsável pelas ações do jogador, seja ela movimentar-se pelo cenário, interagir com objetos ou NPC's;

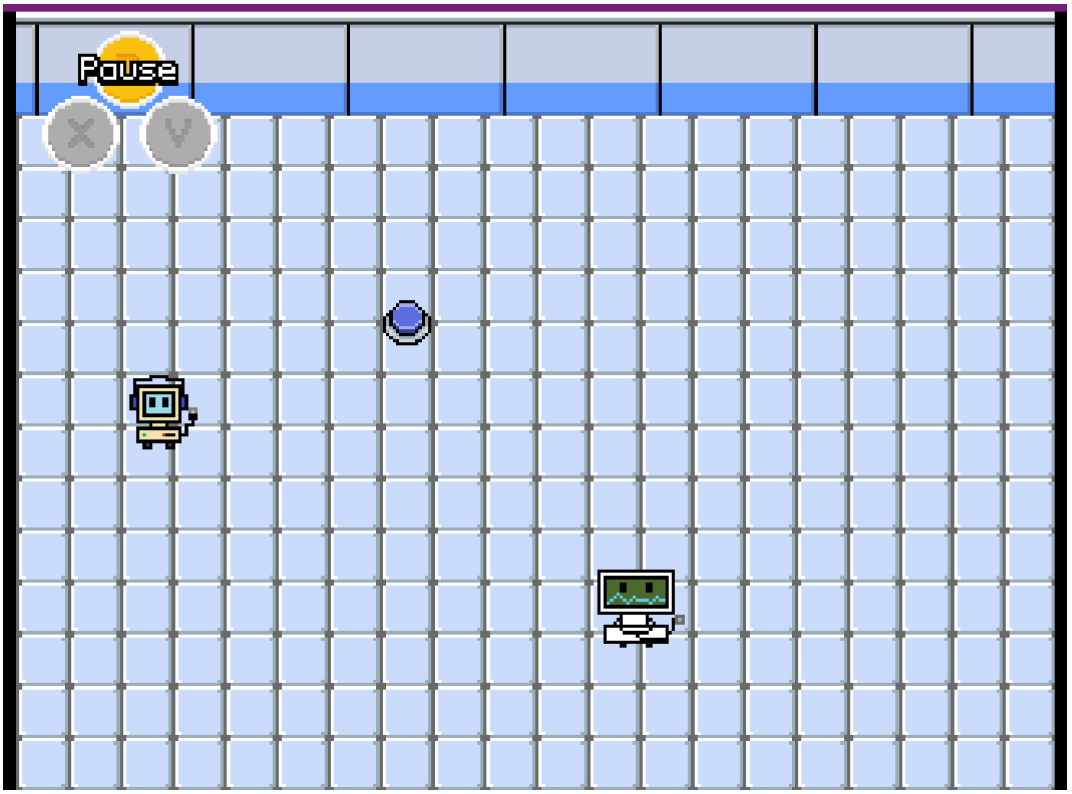
- **Sistema de apoio:** em alguns cenários o jogador contará com esse sistema que irá escrever instruções em sua tela para a conclusão do desafio do cenário.

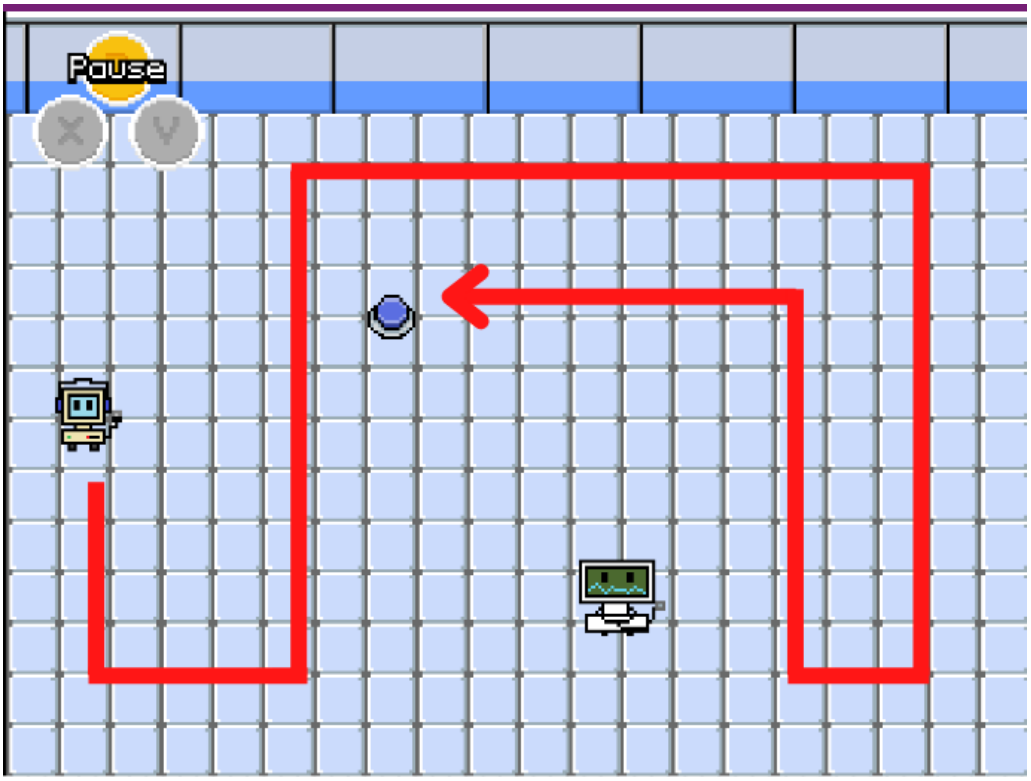
3. Gameplay

O jogo se inicia no cenário 1, o cenário dos portais, e nele o jogador é introduzido no contexto da história do jogo e os primeiros comandos que serão utilizados durante o jogo. A figura abaixo, demonstra o diálogo inicial do jogador com o NPC Doc e o cenário 1 durante o jogo.

Ao se mover pelo cenário, o jogador recebe mais instruções do NPC, desta vez relacionadas aos portais presentes na sala e como eles o levam para os desafios do jogo. Neste cenário também é apresentado o sistema de retorno.

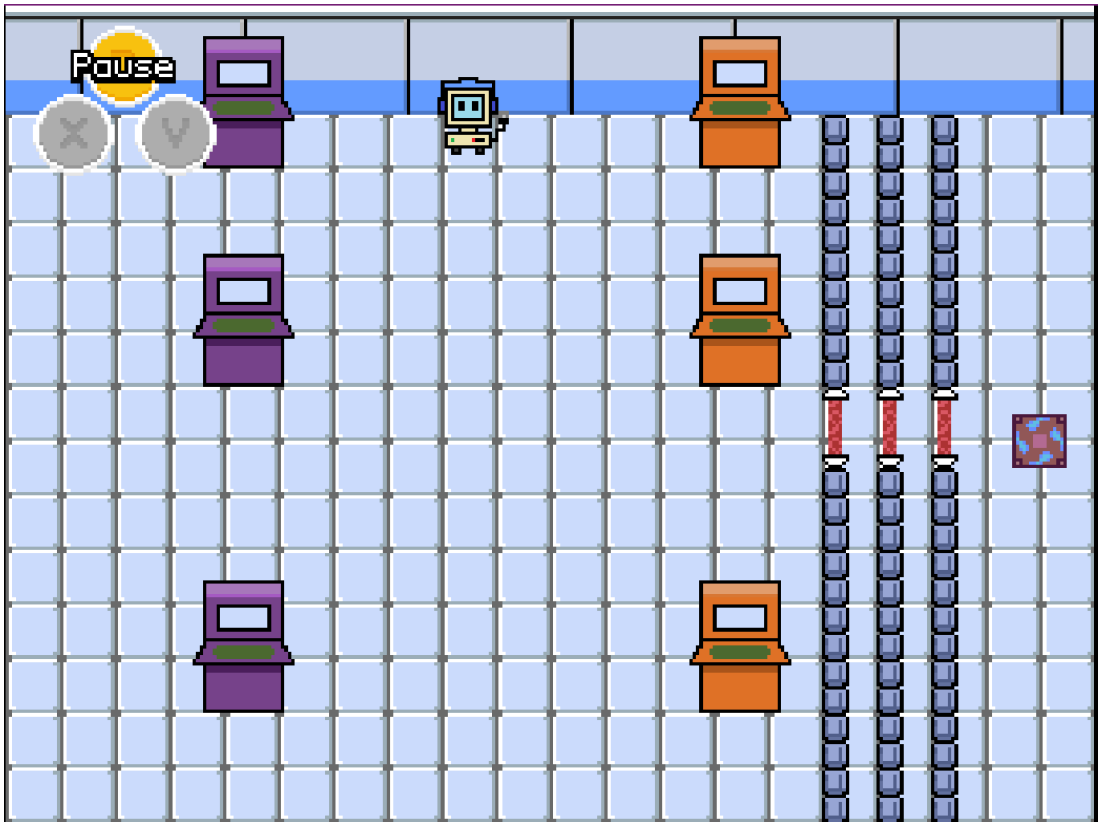
Ao seguir as orientações do NPC o jogador consegue acessar o primeiro desafio, relacionado aos algoritmos. Na figura abaixo está representado o cenário 2 do jogo. Neste cenário, o jogador deve andar pela sala de acordo com as instruções dadas pelo NPC Doc e apertar o botão azul presente na mesma. A sala possui paredes invisíveis, para que o jogador só consiga completá-la ao seguir as palavras do NPC. Na imagem seguinte está a solução do cenário.



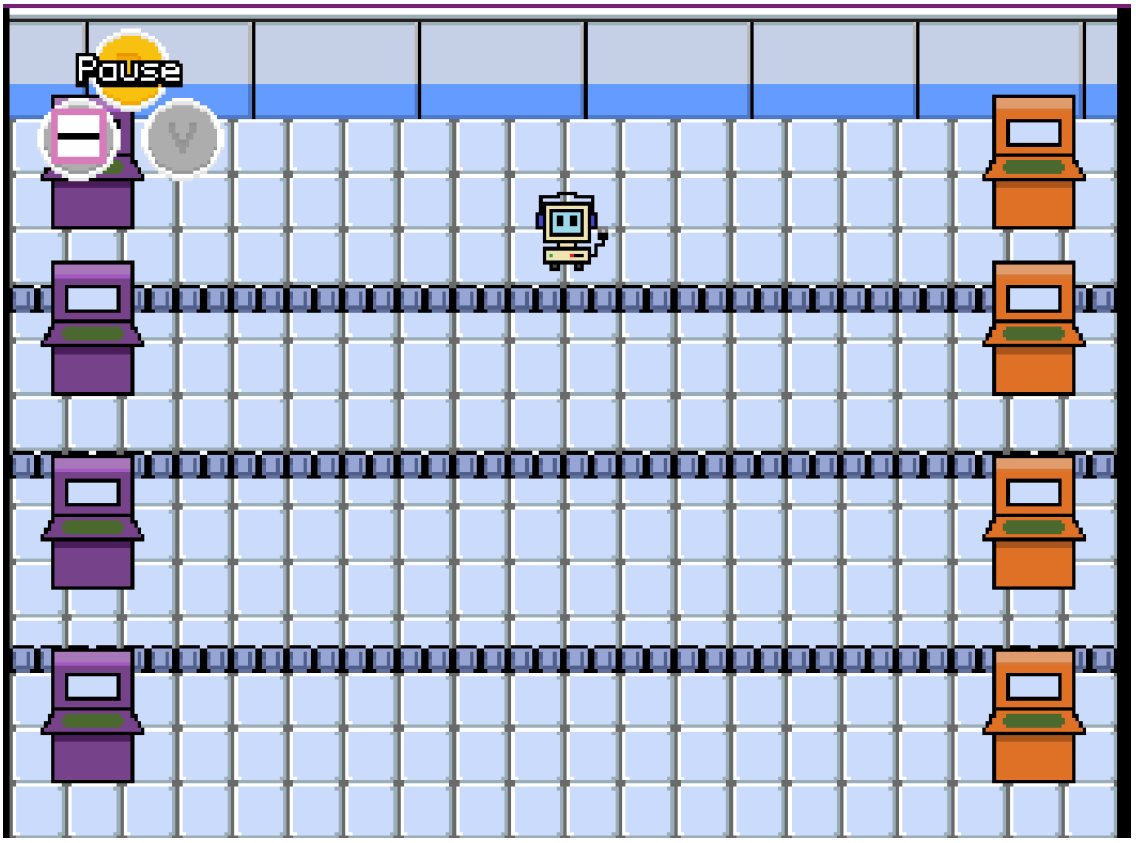


Ao concluir o primeiro cenário um portal irá aparecer para que o jogador possa acessar o próximo cenário, o cenário das variáveis. Neste cenário o jogador terá que interagir com consoles de diferentes cores, nos roxos ele receberá valores e nos laranjas irá usar os valores

como entrada. Os consoles laranjas possuem dicas sobre as entradas necessárias para abrir passagem para o portal para o próximo cenário. As respostas corretas, do console mais acima para o mais abaixo são: ‘Maçã’, ‘38’, ‘Falso’.

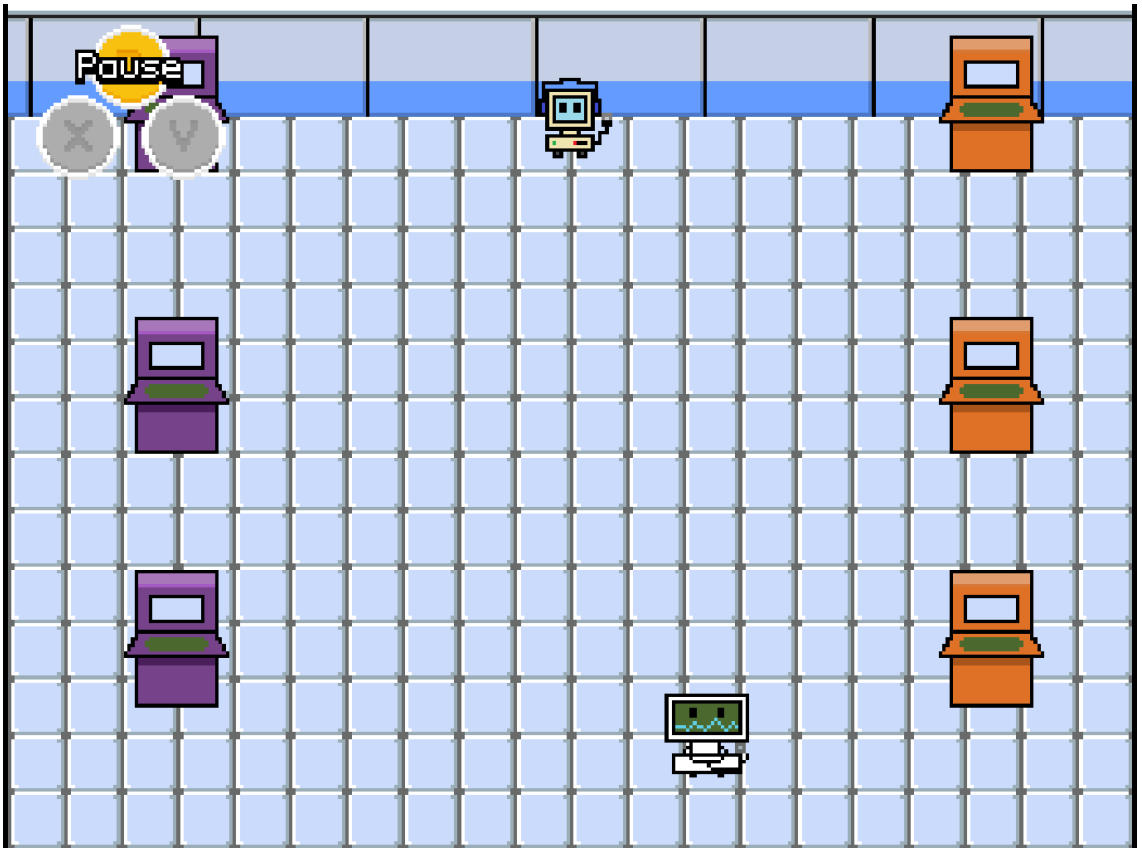


Ao concluir o cenário das variáveis, o jogador acessa o cenário 3, o cenário dos operadores. Neste cenário, assim como no anterior, o jogador irá interagir com consoles de diferentes cores, nos roxos ele recebe valores e nos laranjas, os usam como entrada. Ao concluir cada par de consoles, o jogador avança no cenário. As respostas corretas são: ‘-’, ‘>’, ‘AND’, ‘OR’.



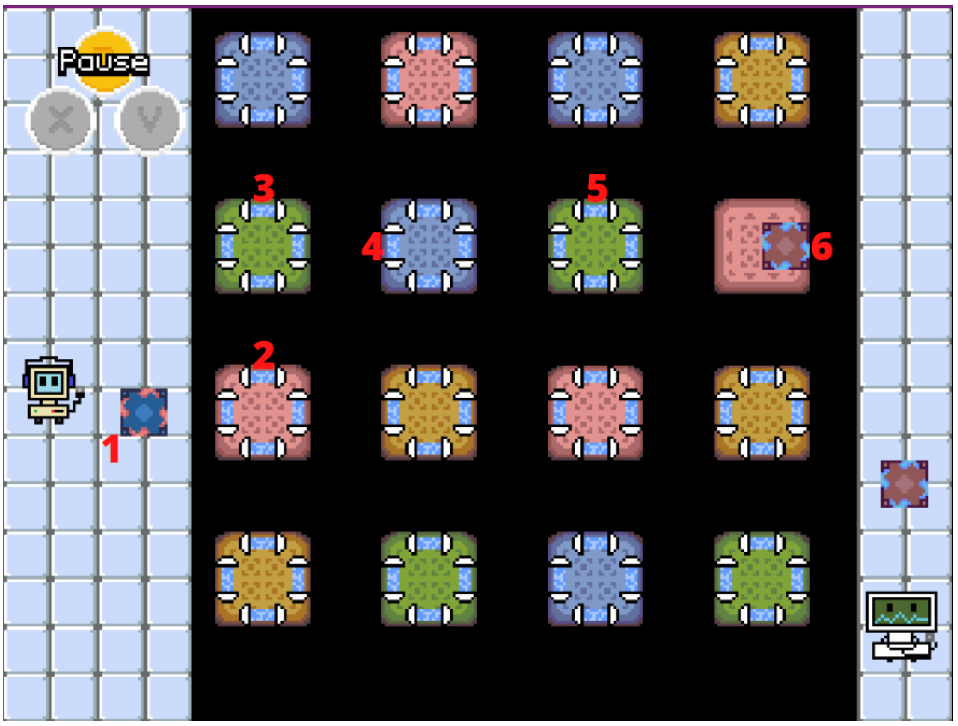
Completando os desafios do cenário 3, o jogador recebe acesso ao cenário 4, o cenário dos tipos de dados. Neste cenário o jogador irá trabalhar com consoles roxos, onde irá receber ou armazenar dados, e consoles laranjas onde irá realizar operações com estes dados.

Para completar o desafio o jogador deve somar os valores ‘60’ e ‘12’ do último console roxo utilizando o último console laranja, depois somar o resultado com o valor ‘5’ do segundo console roxo, converter o resultado no primeiro console laranja e armazená-lo no primeiro console roxo. Após isso, ele deve somar os valores ‘15’ e ‘5’ do segundo roxo e converter o resultado no primeiro laranja. Agora o jogador recebe o valor armazenado no primeiro console roxo na variável V e concatenar os valores. Feito isto, o jogador deve interagir com o NPC para que o mesmo abra o portal para o próximo cenário.

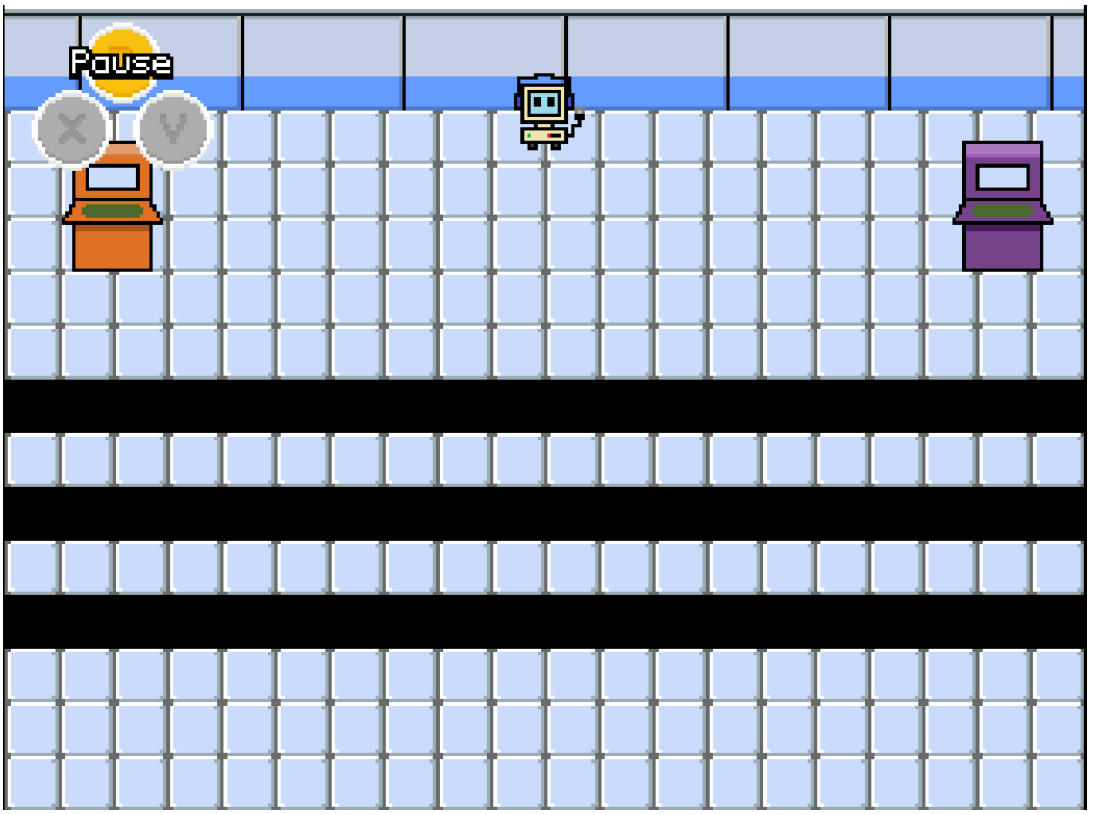


O próximo cenário, o cenário dos condicionais o jogador irá interagir com portais em cima de tapetes para tentar atravessar a sala. Cada cor de tapete tem uma configuração para seus portais. Neste cenário é de grande importância o uso do sistema de apoio, como mostra a imagem a seguir, ela explica as configurações dos portais em cada cor de tapete.

A imagem a seguir mostra o caminho mais curto para a conclusão do cenário dos condicionais.



Ao completar o cenário dos condicionais, o último cenário está acessível, o cenário dos laços de repetição. Neste cenário, o sistema de apoio é uma das ferramentas chave para concluí-lo. O desafio está em conseguir parar os laços de repetição ao alterar valores presentes em suas expressões.



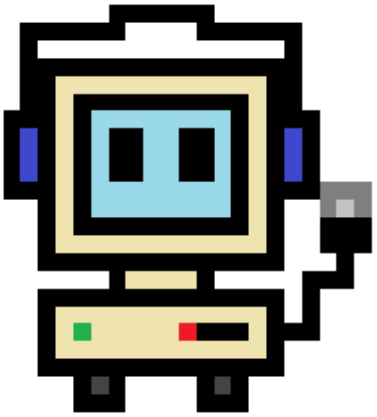
Para concluir o desafio deste cenário o jogador deve:

1. Alterar o valor do console laranja para ‘31’ e o valor do console roxo para ‘38’, assim o primeiro buraco some.
2. Alterar o valor dos dois consoles para ‘15’, assim o segundo buraco desaparece.
3. Cair em algum dos buracos dez vezes, assim o terceiro buraco desaparece.

Ao fazer os três buracos presentes na sala desaparecer e o jogador consegue concluir o jogo e jogar novamente todos os desafios, caso queira.

4. Personagens

- Makina



Personalidade

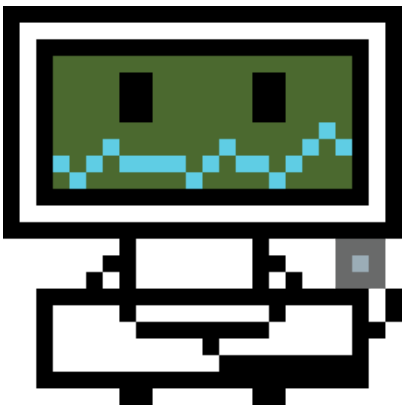
Bem expressiva, mesmo em seu pequeno monitor e perseverante contra os desafios que lhe são impostos. Não se deixou abalar pela perda de dados.

Habilidades

Makina pode:

- se mover pelo cenário;
- interagir com objetos e PNJ's;
- guardar valores dentro de variáveis.

- Doc

- 

Personalidade

Bem humorado e propenso a ajudar, Doc ama sua profissão e a executa muito bem. Professor e guardião de Makina durante a história do jogo, muito experiente na recuperação de pacientes, sejam eles humanos ou máquinas.

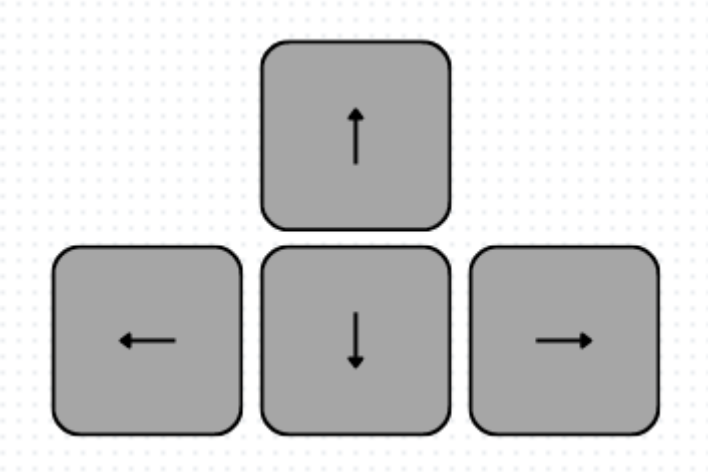
Habilidades

Doc pode:

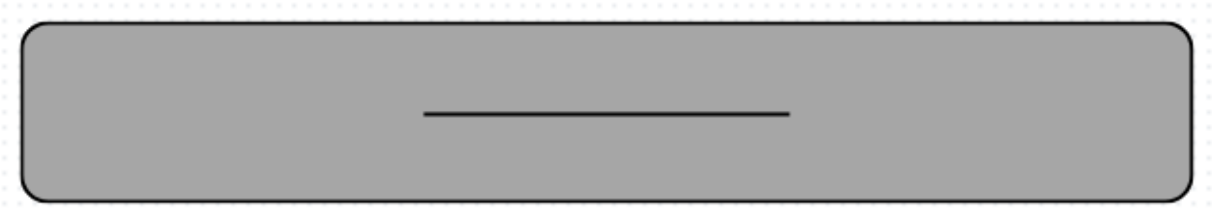
- validar valores entregues pelo jogador.

5. Controles

O jogo possui suporte apenas para teclado.
O jogador utiliza as setas de direção do teclado para mover o personagem.



Utiliza a barra de espaço para interagir com objetos e NPC's.

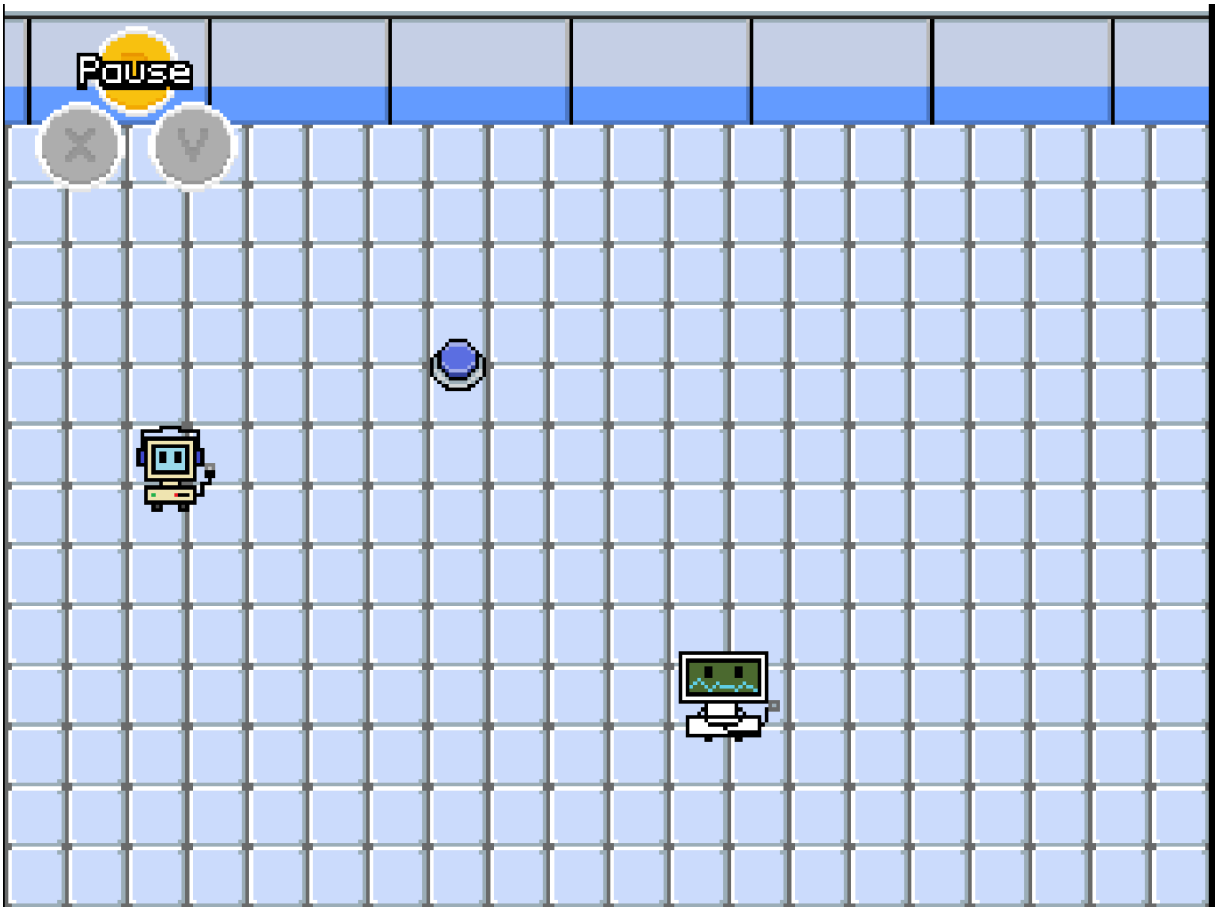


Utiliza a tecla 'P' para utilizar o sistema de retorno e voltar para o primeiro cenário - cenário de portais, a tecla 'R' para utilizar o sistema de restauração e restaurar o cenário atual para o ponto inicial e a tecla 'H' para utilizar o sistema de apoio presente em alguns cenários, onde o jogador recebe instruções via texto em sua tela.

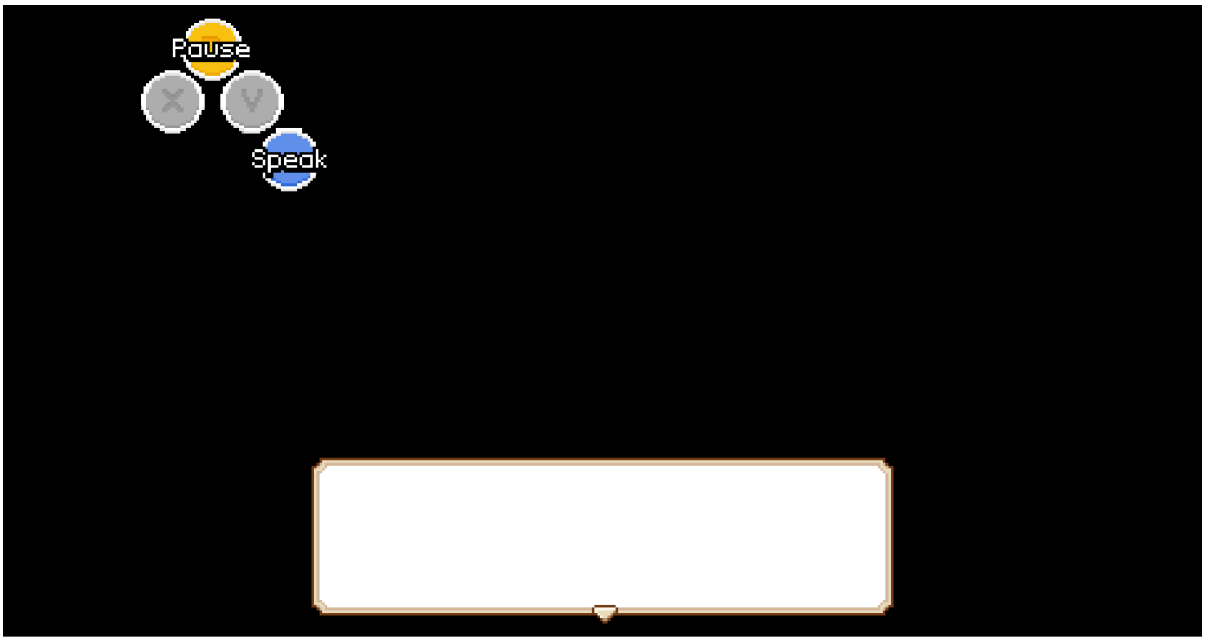
6. Câmera

A câmera do jogo é em uma visão isométrica, simulando um ambiente 3D em um jogo 2D.

A imagem abaixo demonstra a visão do jogo.



7. Interface



A figura acima representa os itens que estarão presentes na tela do jogador. Os elementos estão definidos como:

1. **Botão *Pause*:** representado pelo círculo amarelo com "*Pause"* escrito no meio dele no canto superior esquerdo da tela. Esse botão serve para suspender as ações do jogo, caso o jogador julgue necessário.
2. **Caixas de variáveis:** é uma interface onde o jogador irá armazenar valores ao decorrer do jogo, funcionando como itens obtidos e utilizados no cenário. São representadas por duas bolas cinzas com X e V no meio e caso o jogador tenha algum valor armazenado, itens com os dados são exibidos dentro desses espaços, visíveis no canto superior esquerdo da tela.
3. **Botão de interação:** esse botão aparece quando o jogador pode interagir com algum elemento do jogo. Representado por um círculo azul no canto superior esquerdo da tela.
4. **Caixa de diálogo:** Os diálogos do jogo são representados por textos dentro deste elemento. Representado por um retângulo de cor clara na parte inferior da tela, podendo aparecer no topo da tela, caso o personagem do jogador esteja ocupando a parte inferior.