PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA POLITÉCNICA E DE ARTES
GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

DESENVOLVIMENTO DE UM JOGO DIGITAL: ADAPTAÇÃO DO JOGO DARK SOULS PARA O GÊNERO *METROIDVANIA*

LUÍS FERNANDO FREITAS PAGOTTI

Goiânia
2025

LUÍS FERNANDO FREITAS PAGOTTI

DESENVOLVIMENTO DE UM JOGO DIGITAL: ADAPTAÇÃO DO JOGO DARK SOULS PARA O GÊNERO *METROIDVANIA*

Trabalho de Conclusão de Curso apresentado à Escola Politécnica e de Artes, da Pontifícia Universidade Católica de Goiás, como parte dos requisitos para obtenção do título de Bacharel em Engenharia de Computação.

Orientador(a):
Prof(a). Ma. Angélica da Silva Nunes.

Banca examinadora:
Profº Me Fernando Gonçalves Abadia
Profº Esp. Anibal Vicente Vieira

Goiânia
2025

LUÍS FERNANDO FREITAS PAGOTTI

DESENVOLVIMENTO DE UM JOGO DIGITAL: ADAPTAÇÃO DO JOGO DARK SOULS PARA O GÊNERO *METROIDVANIA*

Trabalho de Conclusão de Curso aprovado em sua forma final pela Escola Politécnica e de Artes, da Pontifícia Universidade Católica de Goiás, para a obtenção do título de Bacharel em Engenharia de Computação em ___/___/_____.

_____

Orientador(a): Profª. Ma. Angélica da Silva Nunes

_____

Prof. Me. Fernando Gonçalves Abadia

_____

Prof. Esp. Anibal Vicente Vieira

GOIÂNIA

2025

DEDICATÓRIA

Dedico este trabalho à minha amada família, pilares essenciais que moldaram minha jornada e desempenharam um papel crucial na concretização dos meus sonhos. Agradeço a Deus pela sua graça e orientação que estiveram sempre comigo, sou profundamente grato pela sua guia e força em cada etapa deste projeto

AGRADECIMENTOS

Agradeço aos meus amigos que me acompanharam nesta jornada. A Pontifícia Universidade Católica de Goiás por me fornecer a estrutura necessária para o meu aprendizado. A minha família em especial ao meu pai que me ajudou ao decorrer de todo o período, aos meus avos que no céu estão com muito orgulho de mim e sempre presentes no meu coração, a minha namorada. E em especial a minha Orientadora Professora Angélica da Silva Nunes, pelos conselhos e orientações neste trabalho.

RESUMO

Este trabalho tem como objetivo o desenvolvimento de um protótipo que propõe a adaptação dos elementos centrais do jogo *Dark Souls* para o gênero *Metroidvania*. A proposta busca transpor mecânicas de combate desafiadoras, ambientação sombria e narrativa fragmentada características marcantes de *Dark Souls* para uma estrutura de progressão não linear típica dos jogos *Metroidvania*. Além da mudança de gênero, foi realizada a conversão do estilo visual tridimensional original para gráficos em 2D com estética *pixel art*, preservando a atmosfera imersiva da obra original. O projeto se concentra em manter aspectos essenciais da experiência do jogador, como o senso de desafio, descoberta e exploração. A pesquisa envolveu a análise dos dois gêneros e a viabilidade de integrar suas principais características, resultando em um protótipo funcional que contribui para o debate sobre *design* de jogos híbridos.

Palavras-chave: *Dark Souls*, *Metroidvania*, *pixel art*, desenvolvimento de jogos, experiência do jogador.

ABSTRACT

This work aims to develop a prototype that adapts the core elements of *Dark Souls* into the *Metroidvania* genre. The project focuses on translating challenging combat mechanics, dark environments, and fragmented *storytelling* key features of *Dark Souls* into the non-linear progression structure typical of *Metroidvania game*s. In addition to the genre transition, the original 3D visuals were reimagined in 2D *pixel art*, preserving the immersive atmosphere of the original work. The project emphasizes maintaining core aspects of player experience, such as the sense of challenge, discovery, and exploration. The research included analysis of both genres and the feasibility of integrating their key features, resulting in a functional prototype that contributes to the discussion on hybrid *game design*.

Keywords: Dark Souls, *Metroidvania*, *pixel* art, *game* development, player experience.

LISTA DE SIGLAS E ABREVIATURAS

API	*Application Programming Interface*, Interface de Programação de Aplicações
CPU	*Central Processing Unit*, Unidade Central de Processamento
FPS	*First Person Shooter*, Tiro em Primeira Pessoa
GHz	*Gigahertz*
GIF	*Graphics Interchange Format*, Formato de Intercâmbio Gráfico
GML	*GameMaker Language, Linguagem de Programação do GameMaker*
GLSL	*OpenGL Shading Language*, Linguagem de Sombreamento do OpenGL
IDE	*Integrated Development Environment*, Ambiente de Desenvolvimento Integrado
ISO	*International Organization for Standardization,* Organização Internacional de Padrões
MB	*Megabyte*
MMORPG	*Massively Multiplayer Online Role-Playing Game*, RPG Multijogador Massivo Online
MUD	*Multi-User Dungeons/Dimensions*, Masmorras/Dimensões Multiusuário
NPC	*Non-playable Character,* Personagem Não Jogável
PC	*Personal Computer,* Computador Pessoal
RAM	*Random Access Memory,* Memória de Acesso Aleatório
RPG	*Role-Playing Game,* Jogo de Interpretação de Papéis
RTS	*Real-Time Strategy,* Estratégia em Tempo Real
UI	*User Interface,* Interface de Usuário
UX	*User Experience,* Experiência de Usuário

LISTA DE FIGURAS E ILUSTRAÇÕES

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

O presente trabalho propõe-se a analisar a *User Experience*, Experiência de Usuário (UX) no contexto dos jogos digitais, com especial atenção aos elementos de *design* e sua influência sobre a percepção e interação dos jogadores. A investigação concentra-se nos fatores que promovem a imersão, a usabilidade e a satisfação dos usuários, considerando aspectos técnicos, como *interface* e navegabilidade, bem como dimensões emocionais, tais como engajamento e prazer. A delimitação da pesquisa restringe-se a jogos digitais desenvolvidos a partir de 2010, e permite uma abordagem direcionada às transformações tecnológicas e às práticas de UX que impactam o *design* e a experiência do usuário na indústria contemporânea de jogos digitais (Teixeira s.d).

A UX em jogos digitais refere-se ao conjunto de sensações, percepções e interações vivenciadas pelos jogadores durante o uso de um jogo. Esse conceito abrange desde aspectos funcionais, como a facilidade de navegação e a clareza das *interface*s, até elementos subjetivos, incluindo o envolvimento emocional e a sensação de desafio ou recompensa. Em jogos digitais, a UX desempenha um papel essencial ao integrar *design*, narrativa e mecânicas de jogo de maneira coesa, de tal forma a proporcionar que os jogadores não apenas compreendam o sistema, mas também se sintam engajados e imersos no ambiente virtual. Assim, a UX ultrapassa o caráter técnico, transformando-se em um fator determinante para a satisfação e a retenção do público (Stati e Sarmento 2021).

A escolha da temática acerca da experiência do usuário em jogos digitais se justifica pela crescente relevância do *design* de experiências interativas na indústria de jogos, especialmente diante da evolução tecnológica e da exigência por produtos cada vez mais imersivos e agradáveis. Com o aumento do número de jogadores e a diversificação do mercado, compreender como os elementos de *design* impactam a satisfação e o engajamento do público torna-se relevante para o sucesso comercial e criativo dos jogos.

A objetivo central deste trabalho é: Desenvolver um protótipo de jogo, baseado no jogo original *Dark Souls*, mudando seu gênero se *Soulslike* para *Metroidvania*, fazendo algumas adaptações. E mantendo aspectos essenciais da experiência do jogador, que estão presentes no jogo original.

1.1 Objetivos

Este trabalho objetiva contribuir para proporcionar uma melhor compreensão sobre a abrangente relação entre *game design* e a UX em jogos digitais.

1.1.1 Objetivo Geral

O objetivo geral deste trabalho é desenvolver um protótipo de jogo, inspirado na plataforma do jogo *Dark Souls*, adaptando para outro gênero e plataforma gráfica, na qual algumas interações entre o *game design* e a UX possam ser avaliadas.

1.1.2 Objetivos específicos

Os objetivos específicos deste trabalho são:

- Analisar os elementos essenciais do *game design* que contribuem para uma UX positiva, com foco nas mecânicas de jogo, *interface*, narrativa e ambientação, e como cada um desses aspectos influenciam a imersão no ambiente pelo jogador;
- Examinar o jogo *Dark Souls* para identificar práticas de *game design* que favoreçam a experiência do usuário, avaliando como o *design* contribui para a criação de uma experiência intuitiva, acessível e envolvente;
- Apresentar o protótipo de jogo desenvolvido neste trabalho.

1.2 Metodologia

A metodologia adotada neste trabalho é qualitativa e descritiva, objetivando analisar e compreender objetivando analisar e compreender a relação entre o *game design* e a experiência do usuário em jogos digitais. A pesquisa será realizada através de duas abordagens principais: análise de jogos e revisão bibliográfica.

Análise de Jogos: A pesquisa envolveu o estudo de casos específicos de jogos digitais reconhecidos pela sua qualidade de *design* e pela experiência positiva proporcionada aos jogadores. Foram analisados jogos que exemplifiquem boas práticas de *game design*. A análise considerou aspectos como mecânicas de jogo, *interface*, narrativa, ambientação, equilíbrio de dificuldade e como esses elementos colaboram para a criação de uma experiência imersiva e satisfatória. A análise foi feita por meio de uma abordagem exploratória e comparativa, com foco na identificação de padrões de *design* que contribuem para uma UX positiva.

Para embasar teoricamente a pesquisa, será realizada uma revisão bibliográfica sobre *game design*, experiência do usuário e *design* centrado no usuário. Foram consultados livros, artigos acadêmicos, e estudos de caso de autores na área.

A revisão teórica permitiu um entendimento aprofundado dos conceitos de *game design*, das melhores práticas e das metodologias centradas no usuário, além de fornecer uma base sólida para a análise dos jogos.

1.3 Resultados Alcançados

Foi desenvolvido um protótipo de jogo, baseado no jogo *Dark Souls*, em ambiente gráfico 2D do tipo *Pixel Art*. Foi aplicado um gênero *Metroidvania* em contraste com o gênero *Soulslike* empregado no original. Com as contribuições destas modificações implementadas por esta proposta de desenvolvimento, procurara-se evidenciar algumas interações positivas entre o *game design*.

1.4 Estrutura da monografia

Este trabalho foi dividido em 6 capítulos.

O capítulo 1 é apresentado a introdução do tema do trabalho, com secções de objetivos gerais e específicos, procedimentos metodológicos.

O capítulo 2 apresenta a Experiência do Usuário, explicando seus fatores e aprofundando nas Leis da Psicologia usadas nela.

No capítulo 3 descreve a Experiência do Usuário em Jogos Digitais, apresentando seus aspectos e apresentando o jogo que foi feito o estudo de caso *Dark Souls*.

O capítulo 4 apresenta os procedimentos metodológicos utilizados no desenvolvimento deste projeto.

O capítulo 5 descreve os testes e resultados feitos no desenvolvimento do protótipo do jogo.

O capítulo 6 apresenta a conclusão do trabalho, com a importância deste trabalho para a comunidade de jogos digitais e sugestões para trabalhos futuros.

2 ASPECTOS GERAIS DA EXPERIÊNCIA DE USUÁRIO

A UX constitui uma área de estudo e desenvolvimento que abrange múltiplas dimensões, com o objetivo de conceber sistemas, *interfaces* e interações, que possam proporcionar valor, satisfação e engajamento aos usuários. Pereira ([s. d.]) destaca que a UX transcende os aspectos puramente estéticos ou funcionais de um produto, englobando toda a jornada de interação do usuário. Esse processo inclui percepções emocionais, significados atribuídos e memórias construídas durante a utilização de produtos e serviços.

Nesse mesmo sentido, Stati e Sarmento (2021) enfatizam que a essência da UX reside na centralidade do usuário e na compreensão de suas necessidades. Segundo os autores, a adoção de práticas fundamentadas em usabilidade, acessibilidade e empatia torna-se indispensável para o desenvolvimento de experiências mais eficazes. Essa abordagem humanizada e estratégica pretende atender aos comportamentos, desejos e limitações dos usuários, promovendo interação significativa e satisfatória.

Este capítulo, portanto, tem como propósito examinar os conceitos e fundamentos essenciais da UX, abordando suas principais diretrizes e aplicações no contexto dos jogos digitais. Busca-se, assim, evidenciar a relevância da UX como um fator determinante para o *design* e para o impacto direto na satisfação e no engajamento dos usuários.

2.1 O que é UX

O termo UX é subjetivo. Para Teixeira ([s. d.]), é complexo descrever como se dá a UX de maneira objetiva e direta, mas é possível aprender como desenvolver um produto, serviço ou ambiente, que proporcione uma experiência satisfatória para alguém que o use e identificar todos os aspectos da interação do usuário com esse produto. A UX é como uma pessoa se sente ao usar um determinado produto. Dentro desta grande área de desenvolvimento da UX, destaca-se neste trabalho as definições das: Norma ISO9241 – 210, Leis da Psicologia e os Fatores do UX *design*.

2.2 A Norma ISO 9241-210

No ano de 2011 a Organização Internacional para Padronização, *International Organization for Standardization* (ISO) criou a norma ISO 9241-210. Está série (9241) aborda a ergonomia e a interação homem-máquina, e a nova parte (210) o *design* centrado em pessoas

e em sistemas interativos. Esta norma define a UX com o resultado alcançado nas respostas e percepções do usuário em determinado produto (Alura, [s. d]).

A ISO 9241-210 possui um total de seis princípios chaves, sendo eles:

> 1. O projeto é baseado no entendimento explícito de usuários, tarefas e ambientes;
> 2. O processo é iterativo;
> 3. O projeto aborda toda a experiência do usuário;
> 4. O projeto aborda toda a experiência do usuário. (ISO, 2011 apud Alura, [s. d].)

2.3 As Leis da Psicologia Aplicadas em UX

Yablonski (2020), propõe a aplicação de algumas Leis da Psicologia nos desenvolvimentos de UX e *game design*, contribuindo para ampliar as interações e percepções positivas dos usuários. Possibilitando uma melhor observação e entendimento dos comportamentos emocionais que promovem a satisfação das pessoas. São elas:

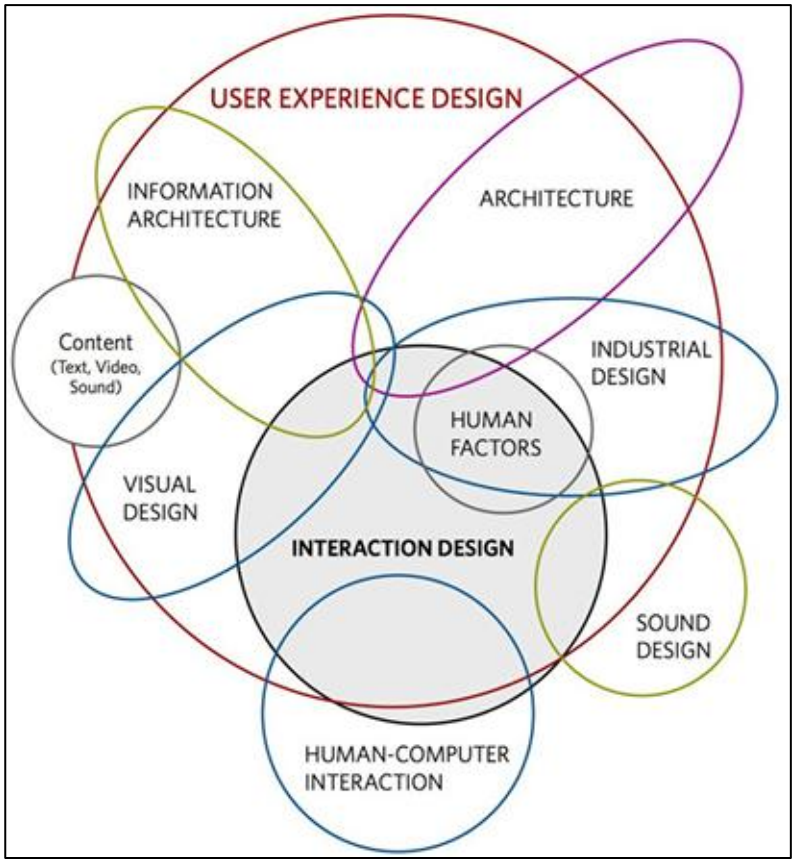
1. Lei de Jakob: os usuários passam a maior parte do tempo em outros *sites* e preferem que seu *site* funcione da mesma maneira que todos os outros *sites* que já conhecem;
2. Lei de Fitts: o tempo para atingir um alvo é uma função de distância até o alvo e seu tamanho;
3. Lei de Hick: o tempo necessário para tomar uma decisão aumenta com o número e a complexidade das opções disponíveis;
4. Lei de Miller: a pessoa média consegue manter apenas 7 itens em sua memória de trabalho;
5. Lei de Postel: as pessoas são conservadoras no que fazem, mas são liberais no que aceitam dos outros;
6. Regra do Pico Final: as pessoas julgam uma experiência em grande parte com base em como se sentiram no auge e no final, em vez da soma total ou média de cada momento da experiência;
7. Efeito estética-usabilidade: os usuários em geral percebem o *design* esteticamente agradável como um *design* mais utilizável;
8. Efeito von Restor: quando vários objetos semelhantes estão presentes, é mais provável que aquele que difere dos demais seja lembrado;

9. Lei de Tesler: a lei de Tesler, também conhecida como lei de conservação e complexidade diz que, para qualquer sistema, existe uma certa quantidade de complexidade que não pode ser reduzida;
10. Limiar de Doherty: a produtividade aumenta quando um computador e seus usuários interagem em um ritmo (<400 ms) que garante que nenhum deles precise esperar pelo outro.

2.4 Fatores do UX *Design*

Há vários fatores que formam o UX *design*. A Figura 1 apresenta um diagrama das disciplinas que compõem o UX, na qual o grande círculo de UX *Design* engloba uma série de outras intersecções com disciplinas tais como: a Arquitetura de Informação, o *Design* Industrial, o *Sound Design* e como todo o elemento de UX foi projetado. A partir deste diagrama, pode-se elaborar um projeto de UX *design* muito bem fundamentado para proporcionar uma excelente interação e satisfação do usuário.

Figura 1: Diagrama das disciplinas que compõem User Experience



Fonte: Pereira – User Experience *Design*: : Como criar produtos digitais com foco nas pessoas

2.4.1 História

Para Iglesias (2018) a história é uma parte muito importante na questão da construção da UX em um jogo. O conhecimento da história (ou *storytelling*) de um jogo e seu universo temporal são fundamentais para a construção do ambiente da UX.

Adicionalmente, a história desempenha um papel importante na imersão do usuário. Possibilitando elaborar um ambiente de UX, onde o jogador se sinta parte da história e acabe se envolvendo cada vez mais na narrativa. A emoção e o apego ao personagem, explicitando ao usuário todas as informações necessárias ao bom divertimento e sem frustrações (Iglesias 2018).

2.4.2 Design

Um dos pontos mais importantes da UX é o *design*, pois é um dos pilares da construção da UX não só em jogos como em qualquer tipo de produto. O *design* é o que possivelmente mais chama a atenção do usuário a testar ou adquirir o produto. (Schell 2008).

O *design* em si tem principais aspectos para a sua relevância na UX, como o *design* visual, que é responsável pela *User Interface*, *Interface* de Usuário (UI), que precisa ser clara e intuitiva para o jogador para que este não fique perdido acerca dos comandos do jogo, por exemplo. A estética e o estilo artístico definem a identidade do jogo, como é a arte do jogo, se é voltado para algo mais realista ou não. O *level design* influencia como o mundo virtual funciona, fazendo com que a UX não pareça artificial, deixando-o mais vivo e imersivo para o jogador (Tekinbas 2003).

2.4.3 Texto

A importância do Texto no desenvolvimento da UX é de esclarecer várias questões nos jogos. A exemplo, desde instruções de jogabilidade, controles do personagem, descrições de itens e suas capacidades, destaques da história e diálogos dos personagens e interações com o ambiente (Rabin s. d.).

2.4.4 Fatores Humanos

Os fatores humanos em jogos referem-se à compreensão das capacidades, limitações e características dos jogadores e como esses aspectos influenciam sua experiência com os jogos. Eles englobam aspectos físicos, cognitivos, emocionais e sociais, e seu bom gerenciamento garante que a UX seja intuitiva, acessível e envolvente. Fatores humanos na UX de jogos dizem respeito à adaptação do *design* do jogo às necessidades e comportamentos dos jogadores. Isso inclui a ergonomia, a percepção sensorial, a memória, o tempo de reação, a motivação e as emoções.

O objetivo é minimizar frustrações e maximizar a satisfação e envolvimento (Borges et al, 2019).

2.4.5 Trilha Sonora

A trilha sonora é um elemento que proporciona imersão e impacto emocional nos jogos. Ela vai além de ser apenas um de fundo sonoro, pois pode comunicar mensagens, reforçar emoções, orientar o jogador e criar uma identidade única para o jogo. A trilha sonora permite ao usuário ter a imersão que está acontecendo naquele momento. Um exemplo seria em uma luta de chefe. Para que o usuário sinta que está enfrentando alguém, o jogo reproduz um som com mais ação para que o usuário perceba a luta (Tekinbas 2003).

A trilha sonora não se limita à música, pode conter orientações, como no jogo *Candy Crush* que, após o usuário realizar uma certa ação, é reproduzido um som que dá início ao evento específico que está acontecendo no nível que o jogador se encontra. Essa trilha pode provocar diversas emoções, como tensão, caso haja algum som que indique que algo perigoso está prestes a acontecer, até para tranquilidade, como o som reproduzido na conclusão de alguma fase.

A trilha sonora é um elemento muito importante pois, com ela, o usuário pode criar memória e identidade com o jogo e que contribui para a sua identidade cultural (Tekinbas 2003).

2.4.6 Interações

As interações do jogador com o sistema abrangem todas as formas de interação entre o jogador e o jogo inclui comandos diretos como, clicar no *Menu* do jogo, até as mais avançadas, como interações sociais em jogos *multiplayer* (multijogador), no qual o jogador interage com

outras pessoas que estão no mesmo jogo que o usuário se encontra. A qualidade dessas interações é essencial para criar um estado de imersão e manter o jogador engajado (Tekinbas 2003).

3 EXPERIÊNCIA DO USUÁRIO EM JOGOS DIGITAIS

A UX em jogos digitais refere-se à interação do jogador com o jogo de forma a gerar uma vivência imersiva, na qual elementos como *design*, mecânicas e narrativa se integram para proporcionar satisfação e envolvimento (Borges et al, 2019).

De acordo com Iglesias (2018), a classificação dos jogos por gênero estabelece uma lógica específica de interação, orientando as expectativas e comportamentos dos jogadores em cada categoria, como ação ou aventura.

Pereira (s. d.) salienta a relevância de um *design* centrado no usuário, que visa atender às suas necessidades e proporcionar uma experiência intuitiva e fluida, com *interfaces* claras e uma progressão de desafios equilibrada.

Para Rabin (s. d.), o desenvolvimento de jogos requer a harmonização entre as mecânicas de jogo e a experiência emocional do jogador, assegurando que a interação seja contínua e que o *feedback* fornecido favoreça a imersão e a satisfação. Assim, o *design* de UX nos jogos digitais vai além da usabilidade e busca criar uma experiência profunda que estabeleça uma conexão significativa entre o jogador e o jogo, incentivando a continuidade da interação.

3.1 *Design*

Um dos principais fatores sobre a UX em jogos é o *design* do jogo, ele é a parte que é a primeira impressão que o usuário tem com o jogo. O *design* é dividido em subtópicos, que é um grande aglomerado que faz parte do *design*, são eles: o ambiente, os personagens e o *storytelling* (Schell 2008).

3.1.1 Ambiente

De acordo com Tekinbas e Zimmerman (2003) o ambiente de um jogo é algo que faz com que o usuário entenda onde ele está e o local no qual está se passando a narrativa. O ambiente pode impactar positivamente um jogo, sendo até seu ponto forte.

Um exemplo desse impacto positivo do Ambiente, é a série de jogos *Batman Arkham*, tratado na maioria das vezes, como um herói sombrio na vigilância da cidade de *Gotham City*. Um ambiente noturno, monopolista, poluído e sujo, cheio de criminosos de diversos tipos. Este tipo de ambientação faz o jogador sentir como é estar em um ambiente parecido como é citado nas narrativas do herói (Tekinbas e Zimmerman 2003).

3.1.2 Personagens

Tekinbas e Zimmerman (2003) afirmam que, na maioria dos jogos, principalmente nos jogos de modo história ou como mais conhecidos *single players*, é comum ter personagens que fazem parte da história e narrativa, do enredo do jogo. Isso é importante para o usuário que está consumindo este tipo de mídia, crie algum tipo de vínculo com os personagens, introduzidos no jogo, tanto para os personagens secundários (conhecidos por *Non-playable Character, Personagem Não Jogável* (NPC)), mas também o personagem principal.

Existem dois tipos de personagens principais neste estilo de jogo, o personagem com a história já estabelecida, na qual o jogador observa a sua jornada e suas ações. Com isso, cria-se um tipo de conexão com o personagem. Existe o segundo tipo, que é o criador de personagem, no qual, o jogador cria o seu próprio personagem, escolhe suas ações, negativas ou positivas, e com quem e como o personagem interage no jogo (Tekinbas e Zimmerman 2003).

3.2 *Storytelling*

Para Adifa (2019), o termo *storytelling* pode ser desmembrado em *story* que quer dizer história, e refere-se a uma narrativa ou construção de enredo em que se dispõe a contar algo para alguém e *telling* que é contar. É, portanto, a forma como se conta uma história a um público ou nicho que se quer atingir. Não basta apresentar uma narrativa qualquer, mas utilizar técnicas na qual existam a perpetuação do conteúdo, a sua manutenção no espaço empresarial, social ou de comunicação que se deseja alcançar.

Em alguns jogos, o *storytelling* não tem muita relevância. Um exemplo é o jogo já citado no item 2.4.5 desse trabalho, o *Candy Crush*, no qual a história não tem grande importância para o jogador, não precisa de muito contexto sobre o que está ocorrendo no jogo. Mas há jogos em que o *storytelling* é importante para o contexto daquele jogo em si, uma narração que introduz o jogador ao universo no qual o jogo se passa e que explica o porquê daquela situação estar ocorrendo, e como o jogador deve agir para prosseguir na narrativa (Adifa 2019).

3.3 Gêneros dos Jogos Digitais

Para Borges et al (2019), os jogos digitais por si possuem suas diferenças, e entre essas diferenças existem os tipos de gêneros que separam um jogo de outro. De acordo com Rabin (s. d.), os gêneros da maioria dos *videogames* podem ser relacionados a um gênero particular ou classificados como híbridos de dois ou mais gêneros. Esses gêneros foram surgindo durante os anos, em geral, como resultado de tentativas e erros, mas frequentemente, também como uma evolução.

A seguir há uma descrição de alguns gêneros importantes: aventura, ação, ação-aventura, plataforma, *Metroidvania*, luta, *First Person Shooter*, Tiro em Primeira Pessoa (FPS), *Real-time strategy*, Estratégia em Tempo Real (RTS), estratégia baseada em turno, *Role-Playing Game*, Jogo de Interpretação de Papéis (RPG), *Massive Multiplayer Online* RPG, RPG Multijogador Massivo *Online* (MMORPG), *Soulslike*, espionagem, horror-sobrevivência, simulação, corrida, esportes, ritmo, *puzzle*, minijogos, tradicional, educacional, sério.

O gênero aventura, há dois tipos importantes de subgêneros: aventura baseada em texto e aventura gráfica (Borges et al, 2019).

O jogo de ação é um combinado de muitos outros gêneros. Tiro em primeira pessoa, ação-aventura, simuladores de combate, jogos de luta. até mesmo jogos de plataforma, são todos parte dos gêneros de ação. Tais jogos são conhecidos por combate em ritmo rápido e movimentação (Vargas 2018).

Para Iglesias (2018) os jogos de ação-aventura são similares aos de aventura, mas incorporam elementos de ação.

Para Borges et al (2019), os jogos originais de plataforma envolviam a personagem correndo e pulando em um campo de jogo com visão lateral. Embora a definição tenha sido expandida para campos de jogo 3D, o gênero ainda e bem fiel as suas raízes.

De acordo com Goldenboy (2022), estilo de jogo *Metroidvania* é marcado por ser *sidescrolling* (ou seja, quando os personagens andam horizontalmente) e possuir grande mapa com áreas interconectadas. Essas áreas são bloqueadas no início do jogo, mas podem ser liberadas, após o jogador obter uma arma, um item ou uma habilidade específica. Assim, é necessário solucionar um quebra-cabeças que, por vezes, mantem o jogador em um vaivém não-linear para que este possa progredir.

Em jogos de luta, o usuário disputa contra outros jogadores ou contra o computador com artes marciais ou espadas. Esse gênero se originou nos fliperamas, nos quais os jogadores podiam desafiar uns aos outros inserindo fichas no gabinete (Borges et al, 2019).

De acordo com Borges et al (2019) o jogo de FPS é de ação e coloca o usuário "atrás dos olhos", da personagem. O jogador é capaz de utilizar uma variedade de armas e acabar com os inimigos atirando neles.

Para Vargas (2018) em um jogo RTS típico, a meta é coletar recursos, construir um exército e controlar suas unidades para atacar o inimigo. A ação tem ritmo rápido e devido ao jogo ser contínuo, decisões estratégicas devem ser feitas rapidamente.

Para Renatito (2021), os jogos de estratégia baseada em turno são similares aos RTS (na verdade, foram os precursores). Neles, os usuários levam turnos para tornar as decisões. Por exemplo, a maioria dos jogos de tabuleiro (como xadrez e damas) utiliza turnos.

Vargas (2018) descreve os RPGs como a versão em *videogame* dos jogos de mesa, como o *Dungeons & Dragons*. Esse tipo jogo difere de sua contraparte na maioria devido à habilidade de criar um mundo que não necessita de imaginação. A maior parte das diferenças da fórmula é híbrida de outros gêneros.

Para Renatito (2021), o MMORPG ou MMO é um jogo de RPG estabelecido em um mundo virtual povoado por centenas de jogadores de forma simultânea, conectados através da *Internet*. O MMO foi precedido por jogos baseados em texto chamados *MultiUser Dungeons/Dimensions*, Dimensão/Masmorra Multiusuário (MUDs). Nos jogos, o usuário é representado por um personagem na figura do avatar.

De acordo com Emboava (2024), jogos *Souls-like* são basicamente RPGs de ação marcados pela alta dificuldade, combates intensos e sistemas de progressão. Esse gênero é usado para marcar que um jogo é desafiador e recompensador.

De acordo com Renatito (2021) jogos de espionagem (também chamados de *stealth games* ou *sneakers*) são caracterizados por seu foco em subterfugio e sua jogabilidade deliberada e planejada. São, em geral, similares a jogos de tiro em primeira e terceira pessoa, mas menos orientados a ação e mais metódicos.

De acordo com Renatito (2021), horror-sobrevivência é um subgênero de ação, aventura e jogos de tiro em primeira pessoa. Normalmente, envolvem explorar construções e cidades abandonadas por onde vários monstros e zumbis circulam. Os elementos de sobrevivência são destacados pelo fato de o jogador nunca receber balas ou vidas suficientes, aumentando assim a tensão. O aspecto de horror que define o tema é o ritmo, que comumente levam o jogador a explorar corredores silenciosos, desertos cheios de sangue, sustos provocados pela chegada repentina de monstros ou um corpo aparentemente sem vida começar a se mexer. Os jogadores

em geral se assustam e podem se tornar visivelmente abalados com a experiência, tal como acontece em um bom filme de horror.

Para Renatito (2021), os jogos de simulação são baseados na simulação de um sistema. Esse sistema pode ser qualquer coisa, desde o funcionamento e a economia de ferrovias, até um cenário de combate no qual o jogador controla grandes movimentos de tropas, ou até mesmo uma simples aeronave.

Renatito (2021) define os jogos de corrida envolvendo alguma competição com veículos desde carros até motocicletas e *karts*. Esse gênero é um pouco diferente dos demais e tentam recriar da melhor maneira possível uma atividade no mundo real.

O gênero de esportes**,** segundo Renatito (2021), abrange uma grande quantidade de jogos que simulam experiencias esportivas, como nos de corrida, os jogos de esporte são, em geral, uma tentativa de recriar interações complexas de um esporte real.

Para Renatito (2021), os jogos de ritmo medem o sucesso do jogador baseando-se em sua habilidade de ativar os controles no tempo da batida de uma música.

Para Vargas (2018), jogos de *puzzle* unem elementos de combinação de padrões, lógica e sorte - geralmente com um elemento de tempo.

Na visão de Renatito (2021), os minijogos são tipicamente curtos e simples, e fazem parte do conteúdo de um jogo tradicional maior. São muitas vezes usados como uma recompensa por completar um desafio ou são destravados pela descoberta de um segredo.

Para Renatito (2021), jogos tradicionais incluem versões de computador de jogos de cartas e tabuleiro.

De acordo com Renatito (2021), os jogos educacionais são planejados para ensinar conceitos de escola para crianças e jovens adultos de maneira lúdica.

Para Renatito (2021), o gênero sério emergiu como uma forma barata e divertida de oferecer treinamentos ao público adulto. Esses jogos, em geral, são financiados para usos específicos, como, por exemplo, pelo governo norte-americana profissionais da área médica. Desenvolvedores de jogos podem criar simuladores de treinamento relativamente baratos, enquanto combinam simulação com valor de entretenimento. A diversão e importante para que os usuários fiquem motivados a jogar novamente e assim tornar-se mais bem treinados.

3.4 Estudo de Caso: Dark Souls

O jogo apresentado neste trabalho é o *Dark Souls* que, segundo Losada (2012) é um jogo singular, para poucos, oferece uma experiência única, que ainda não havia ocorrido em outros jogos, desde a maneira de se jogar até o seu alto nível de dificuldade, o que pode afastar a boa parte dos jogadores.

3.4.1 Design do Jogo

O *design* de *Dark Souls* é uma obra que combina maestria estética e mecânica, conforme descrito por From*software* (2014), autor do jogo, com uma abordagem profundamente centrada na criação de um ambiente imersivo e desafiador. A interconexão das áreas, cuidadosamente projetadas para promover a exploração e recompensar a curiosidade do jogador, é uma das principais características destacadas. Além disso, o estilo visual, inspirado pela arquitetura gótica e pelo simbolismo da decadência, reflete o tom sombrio e melancólico do jogo.

De acordo com Baggins (2016), o equilíbrio entre dificuldade e aprendizado é um pilar fundamental do *design*, criando uma experiência que respeita a inteligência do jogador e valoriza sua perseverança. As mecânicas minimalistas e precisas de combate, aliadas a uma narrativa implícita e fragmentada, estimulam o engajamento e incentivam a interpretação pessoal do jogador. Esse *design*, que integra ambiente, jogabilidade e história de forma coesa, estabelece *Dark Souls* como uma referência notável na indústria de jogos.

3.4.2 Ambiente do Jogo

O ambiente de *Dark Souls* foi meticulosamente projetado para transmitir uma atmosfera de decadência, mistério e desolação, elementos amplamente explorados em *Dark Souls: Design Works* por From*software* (2014), autor do jogo.

Cada área do jogo apresenta uma estética única, inspirada na arquitetura gótica e em paisagens medievais, repletas de detalhes que narram visualmente a história do mundo. O uso de cores escuras, iluminação cuidadosamente posicionada e cenários imponentes cria um senso constante de opressão e perigo, enquanto os elementos ambientais sugerem a presença de uma civilização outrora grandiosa, agora em ruínas (From*software*, 2014).

A ambientação não é apenas decorativa, mas desempenha um papel crucial na narrativa implícita do jogo, convidando o jogador a interpretar os símbolos e pistas presentes no cenário.

A interconectividade do mundo reforça a imersão, permitindo uma exploração fluida e revelando segredos que recompensam a atenção aos detalhes, consolidando o ambiente como um dos elementos mais memoráveis e impactantes do jogo (From*software*, 2014).

3.4.3 Personagens do Jogo

De acordo com a entrevista feita por Otsuka (2018) os personagens de *Dark Souls* são projetados para serem uma extensão da narrativa profunda e misteriosa do jogo. Eles variam de figuras heroicas a antagonistas trágicos, cada um representando temas como sacrifício, decadência e resistência em um mundo à beira do colapso. Exemplos marcantes incluem Gwyn, o Lorde da Luz Solar, que sacrificou sua humanidade para prolongar a Era do Fogo; e Artorias, o Cavaleiro do Abismo, cuja corrupção demonstra o peso de lutar contra o próprio destino. Além disso, o jogo possui NPCs como Solaire de Astora oferecem um raro alívio e otimismo, incentivando os jogadores em sua jornada solitária. Essas personalidades são intimamente conectadas à mecânica do jogo, influenciando diretamente a progressão e o desenrolar da história enquanto os jogadores exploram o reino de Lordran. A complexidade dessas interações é parte do que torna a experiência tão imersiva e intrigante para os fãs da série.

3.4.4 Storytelling do Jogo

Segundo Losada (2012) sua história se passa nos tempos antigos onde habitavam os anciões, o mundo ainda não possuía uma forma definida. Era uma terra fria e coberta de uma extensa e densa névoa, com imensos rochedos cinzentos que se estendiam sem fim e desfiladeiros repletos de gloriosas Árvores antigas e colossais, com troncos que alcançavam os céus escuros e sombrios. E nesta terra, reinavam os grandiosos dragões imortais. E logo surgiu o fogo no mundo, mas junto com o fogo surgiram também diferenças. Calor e frio, vida e morte. E o mais importante, a luz e a escuridão.

Com a chegada desses fatores surgiram os primeiros Lordes que foram responsáveis por dar um fim ao reinado dos Dragões Imortais, eles eram Nito, o primeiro dos mortos, a bruxa Izarith e suas filhas do caos. Gwyn, o senhor da luz solar e seus fiéis cavaleiros e a furtiva Pygmy, tão facilmente esquecida, eles tiveram também a ajuda de um dos Dragões Imortais aquele que traiu a sua própria raça por inveja Seath o Dragão sem Escamas. Juntos eles deram início a idade do fogo, mas logo as chamas irão se extinguir e apenas a escuridão irá prevalecer Losada (2012).

Mesmo nos tempos atuais, existe apenas o calor do fogo queimando entre a madeira e o carvão, e o homem não enxerga mais a luz, e sim noites sem fim. Este é o prólogo do jogo, a jornada do jogador começa depois de todos esses acontecimentos e ele acorda em um Asilo de Mortos Vivos onde é resgatado por outro cavaleiro que lhe dá a chave para se libertar de sua cela. E assim começa a aventura do jogador neste mundo mágico e cheio de perigos Losada (2012).

3.4.5 Gênero do Jogo

O gênero de *Dark Souls* é frequentemente categorizado como *action* RPG, mas seu impacto na indústria de jogos deu origem a uma subcategoria conhecida como *"Soulslike"* citado no item 3.3 deste trabalho. Este gênero se distingue por combinar combate meticuloso e estratégico, progressão baseada na coleta de recursos e exploração não linear em um mundo interconectado. Em *Dark Souls*, a dificuldade elevada e a curva de aprendizado desafiadora são características marcantes, enfatizando a necessidade de paciência, habilidade e adaptação do jogador. O jogo incorpora elementos tradicionais de RPG, como personalização de personagens, escolha de classes, habilidades e equipamentos, mas eleva essas mecânicas com a integração de um sistema de combate altamente técnico e uma narrativa implícita. Através dessas características, *Dark Souls* transcende os limites do RPG de ação convencional, estabelecendo um modelo que influenciou profundamente o *design* de jogos em todo o mundo.

4 PROCEDIMENTOS METODOLÓGICOS

Esse capítulo detalha os elementos utilizados na proposta de desenvolvimento deste trabalho. São apresentados o seu objetivo, os elementos que compõe a sua construção e a interação desses elementos para a elaboração de um UX no ambiente de simulação.

4.1 Ambiente de Testes

O ambiente de testes é composto por um computador pessoal e um software de criação de jogos digitais *Game Maker*.

4.1.1 Computador

A simulação é executada localmente em um computador com processador AMD *Ryzen* 5 5600 6 *Core Processor*, *Central* Processing Unit, Unidade Central de Processamento (CPU) @ 3.7 *Gigahertz* (GHz) 4.6 GHz, L2 *Cache* 3 *Megabytes* (MB) L3 *Cache* 32 MB *Random Access Memory*, Memória de Acesso Aleatório (RAM), placa de vídeo NVIDIA 4060-TI, Sistema operacional de 64 *bits*, processador baseado em x64 e utiliza o Sistema Operacional Windows 11 Pro na versão 10.0.26100.

4.1.2 GameMaker

O *software GameMaker* foi escolhido por ser gratuito, possuir diversos recursos que apoiam o desenvolvimento do jogo e de possuir uma linguagem própria parecida com C++. Para o projeto utilizou-se a versão 2024.13.1.242.

4.2 Descrição do Protótipo de Jogo

Esta proposta de desenvolvimento de um protótipo de jogo digital, foi baseado no conhecido jogo *Dark Souls*, modificando alguns elementos presentes no jogo como gênero, *gameplay* e gráficos e mantendo a essência do jogo como ambientação, personagens e efeitos sonoros.

4.2.1 Características Gerais do Jogo

Dark Souls é um *Soulslike* desenvolvido pela *FromSoftware*, lançado originalmente em 2011. Reconhecido por sua alta dificuldade, ambientação sombria e narrativa indireta, o jogo desafia os jogadores a explorarem um mundo interconectado repleto de inimigos letais, armadilhas e chefes imponentes (Baggins, 2016). A estrutura de progressão é marcada pela experimentação, paciência e repetição, o que incentiva o aprendizado com os erros e promove uma intensa sensação de conquista.

A obra se destaca pelo *design* de mundo cuidadosamente planejado, onde cada área se conecta de forma coesa, promovendo uma exploração fluida e recompensadora (Otsuka, 2018). A ausência de tutoriais explícitos e a narrativa fragmentada incentivam o jogador a montar sua própria compreensão do universo, por meio de descrições de itens, diálogos enigmáticos e observação ambiental (Software, 2014).

Além disso, o jogo introduz um sistema online inovador, que permite interações indiretas entre jogadores, como mensagens deixadas no chão e invasões entre mundos, fomentando uma comunidade ativa e colaborativa mesmo diante da natureza desafiadora do título (Baggins, 2016).

Segundo Losada (2012), *Dark Souls* redefine a noção de dificuldade ao tratar o fracasso não como punição, mas como parte essencial da jornada. Essa filosofia de *design* reforça a imersão do jogador e amplia a sensação de recompensa ao superar obstáculos aparentemente intransponíveis.

4.2.2 Aspectos do Desenvolvimento do protótipo do jogo denominado "Echoes of Dust"

Esta proposta de desenvolvimento emprega um gênero *Metroidvania* utilizando a ferramenta de criação de jogos *GameMaker*, possuindo gráficos 2D *pixelart*, com um estilo de arte sombria e melancólica inspirada em estéticas do gênero *Dark Fantasy*.

A história do jogo se passa em um castelo abandonado onde o personagem principal se encontra em uma cela e tem como objetivo, fugir do local, enfrentando desafios que encontra pelo caminho.

O ambiente do jogo é composto por 3 áreas para exploração denominadas de: Castelo, Torre e Esgoto.

A sessão do Castelo é um caminho linear onde o jogador pode apenas seguir em frente e enfrentar o chefe principal do jogo, o jogador tem a opção de apenas derrotar o chefe sem

precisar explorar a área inteira e terminar o jogo, o chefe será extremamente desafiador. É, no entanto, recomendado o jogador explore a área em questão, enfrentando oponentes menores, e adquirindo o item especial dispersado pela sessão, para tornar-se mais poderoso, e então combater o chefe principal com menor dificuldade.

O Esgoto é a parte inferior do castelo, onde seu acesso inicial somente é permitido pela passagem por um buraco, sem condições de retornar.

Uma vez dentro do buraco, o jogador irá enfrentar diferentes desafios na forma de inimigos menores e encontrará também um chefe secreto. Caso o jogador opte por enfrentar e vencer o chefe secreto, irá receber um item valioso para ajudá-lo em sua jornada. Caso o jogador opte por não enfrentar esse desafio, basta explorar o ambiente na direção oposta, onde encontrará uma escada, liberando sua saída da sessão esgoto.

A terceira e última sessão será a Torre, onde após interagir com um elevador presente na sessão do castelo, o jogador irá para o nível superior do ambiente do jogo. Nesta sessão, haverá dois caminhos para exploração à esquerda e á direta. No caminho da esquerda encontrará um item de presente. No caminho a direita, encontrará um item interativo. Após o jogador interagir com esse item, receberá uma recompensa que reduzirá o poder do seu inimigo principal, na sessão do Castelo.

De volta à sessão do Castelo, o jogador estará mais equipado para derrotar o chefe principal. Após derrotar o inimigo, um portão se abrirá, liberando o caminho para o final do jogo.

4.2.3 Adaptações realizadas

Para a realização deste jogo, os ambientes, as áreas, músicas e personagens foram adaptados constituindo características de semelhança e de diferenças entre o jogo original e o protótipo aqui em desenvolvimento. O Quadro 1 apresenta as semelhanças utilizadas, e o Quadro 2 as diferenças incluídas no desenvolvimento proposto.

Quadro 1: Semelhanças com *Dark Souls*

Semelhanças com Dark Souls
Estética Dark Fantasy
Começa na primeira fase do jogo original
Áreas Únicas Interligadas
Músicas dos Chefes foram ultilizadas
Inimigo do Dark Souls está no Jogo
Continua...

Semelhanças com Dark Souls
Itens semelhantes
Sistema de Combos
Alguns inimigos do Dark Souls estão no jogo

Fonte: Elaborado pelo autor do trabalho a partir de Microsoft Excel (2025)

Quadro 2: Diferenças com *Dark Souls*

Diferenças com Dark Souls
Progressão de Personagem por Exploração
Mudança de 3D para 2D
Um storytelling mais direto e simples de entender
Quando o jogador sofre dano ele fica imune á dano por um tempo
Alguns inimigos não estão presentes em Dark Souls
Há apenas uma magia
Há inimigos diferentes
Chefes / Bosses diferentes

Fonte: Elaborado pelo autor do trabalho a partir de Microsoft Excel (2025)

4.3 Descrição do *GameMaker*

A ferramenta utilizada neste projeto é o *GameMaker*, possui *Integrated Development Environment*, Ambiente de Desenvolvimento Integrado (IDE) voltado à criação de jogos eletrônicos 2D. Desenvolvido pela *YoYo Games*, o *software* oferece uma combinação entre *interface* visual baseada em eventos e uma linguagem de programação própria, a *GameMaker Language*, Linguagem de Programação do *Game*Maker (GML), que permite maior controle sobre a lógica do jogo, estrutura de dados e manipulação gráfica.

A *engine* opera com um sistema orientado a objetos, onde entidades chamadas de *objects* podem ser associadas a sprites, comportamentos e eventos específicos, como colisões, entrada do usuário e atualizações de frame. A GML possui sintaxe própria, similar a linguagens como *JavaScript* e C, permitindo o uso de estruturas condicionais, laços, *arrays*, scripts personalizados e funções nativas de alto desempenho.

O *pipeline* de desenvolvimento do *GameMaker* permite a compilação e exportação para múltiplas plataformas a partir do mesmo projeto-base, com suporte a renderização por *Central Processing Unit*, Unidade Central de Processamento (GPU), efeitos de partículas, gerenciamento de recursos em tempo de execução e integração com *Application Programming Interface*, Interface de Programação de Aplicações (API) externas.

Além disso, o *GameMaker* oferece ferramentas integradas como editor de *sprites*, gerador de *tilesets*, sistema de física 2D, suporte a *shaders* via *OpenGL Shading Language*, Linguagem de Sombreamento do *OpenGL* (GLSL) e gerenciamento de som e música. Essa combinação torna o *GameMaker* uma solução robusta para o desenvolvimento de jogos 2D com alto controle técnico e possibilidade de otimização.

4.3.1 Terminologia

A seguir, são apresentados os principais termos técnicos utilizados ao longo deste trabalho, com o objetivo de padronizar o entendimento dos conceitos abordados:

- ***Assets*:** Conjunto de elementos gráficos, sonoros e visuais utilizados na construção do jogo digital, como sprites, sons e animações;
- ***Checkpoints*:** Pontos de salvamento distribuídos ao longo do jogo que permitem ao jogador reiniciar a partir deles após falhas, sem a necessidade de repetir todo o percurso anterior;
- ***Engine*:** Ambiente de desenvolvimento de jogos que reúne ferramentas e funcionalidades para construção, execução e exportação de projetos;
- ***Level Design:*** Planejamento e construção das fases do jogo, considerando ritmo, progressão de dificuldade, posicionamento de obstáculos e estímulos à exploração;
- ***Pixel Art*:** Estilo visual que utiliza *pixel*s como unidade básica de composição gráfica, resultando em uma estética retrô com baixa resolução;
- ***Sprite**:* É uma imagem ou uma sequência de imagens (animação) usadas para representar visivelmente um objeto no jogo;
- ***Tileset**:* Uma imagem que contém várias tiles organizadas em uma grade.
- **Evento *Create**:*** Evento executado uma única vez quando o objeto é criado na *room* (sala);
- **Evento *Begin Step*:** Executado antes do *Step* normal;
- **Evento *Step*:** Evento principal onde acontece a lógica do jogo;
- **Evento *End Step**:*** Executado depois do *Step* normal.

4.3.2 Telas

Na sequência são apresentadas as telas de programação utilizadas para o desenvolvimento do protótipo do jogo digital proposto.

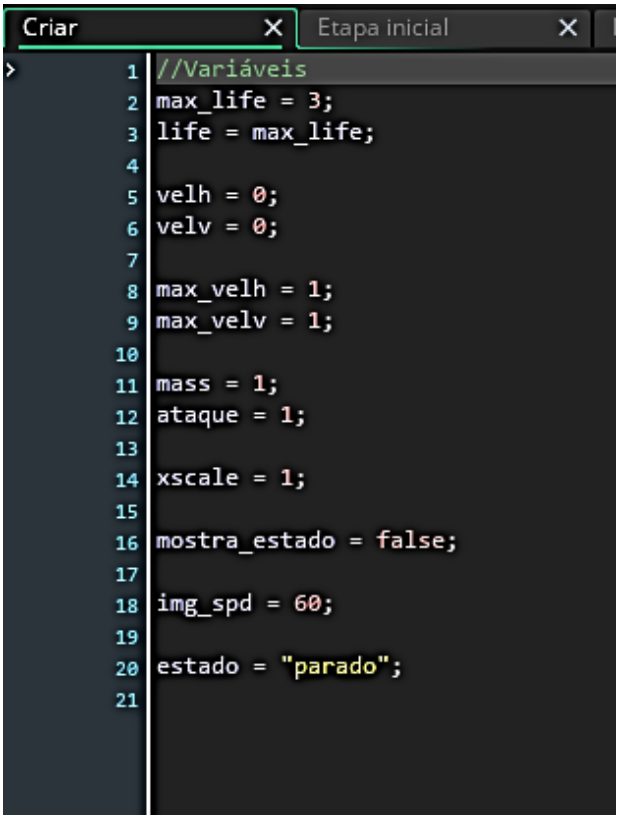
A Figura 2 apresenta a tela de criação de sessões do jogo, definindo os ambientes, as artes do cenário, e as dimensões da tela.

Figura 2: Tela de Criação de Sala

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

Esta tabela de códigos define as variáveis das entidades do jogo, tais como sua vida, velocidade, massa, estado e massa.

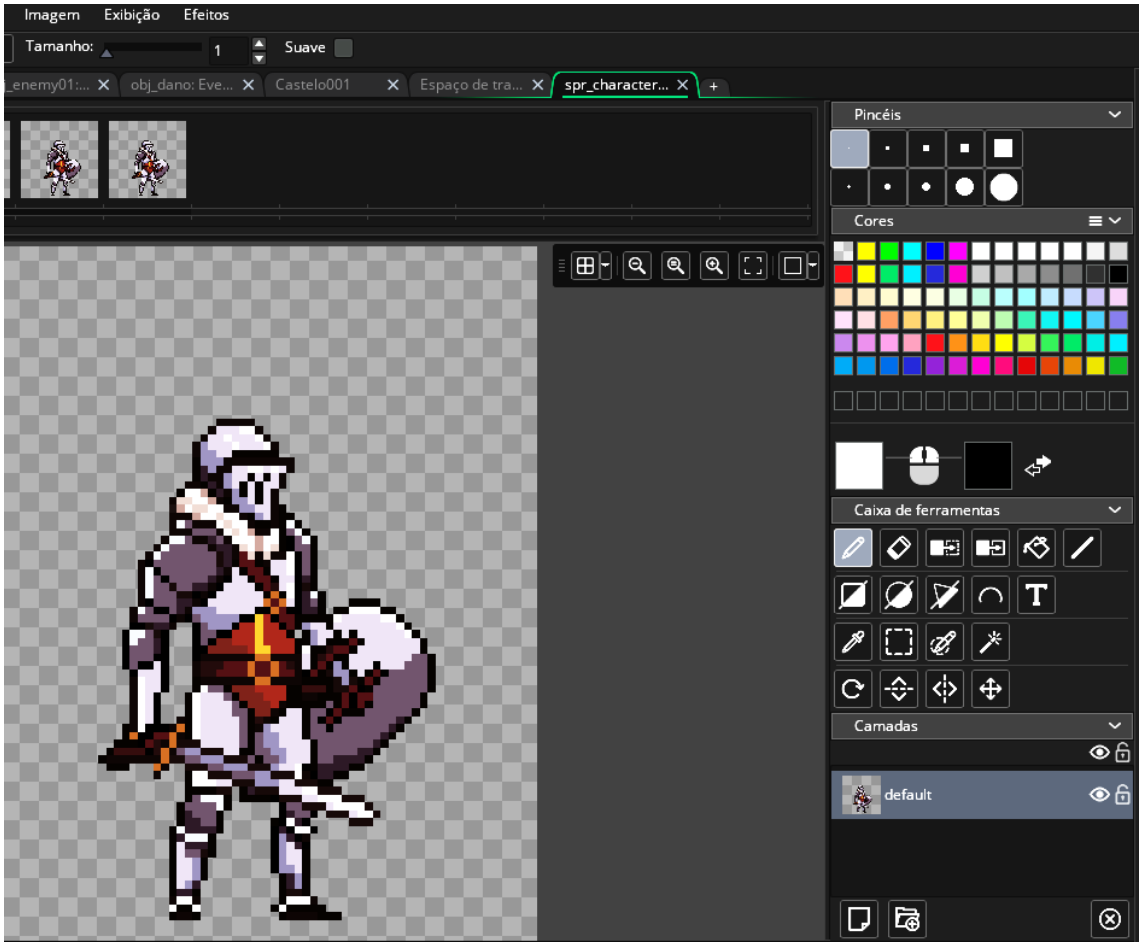
Figura 3: Tela do Código do Objeto



Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

A Figura 4 apresenta uma amostra das telas de Editor de Imagens. Nela são criados e editados os *sprites* do jogo. Com essa configuração, é possível fornecer a animação aos personagens criados com seus movimentos.

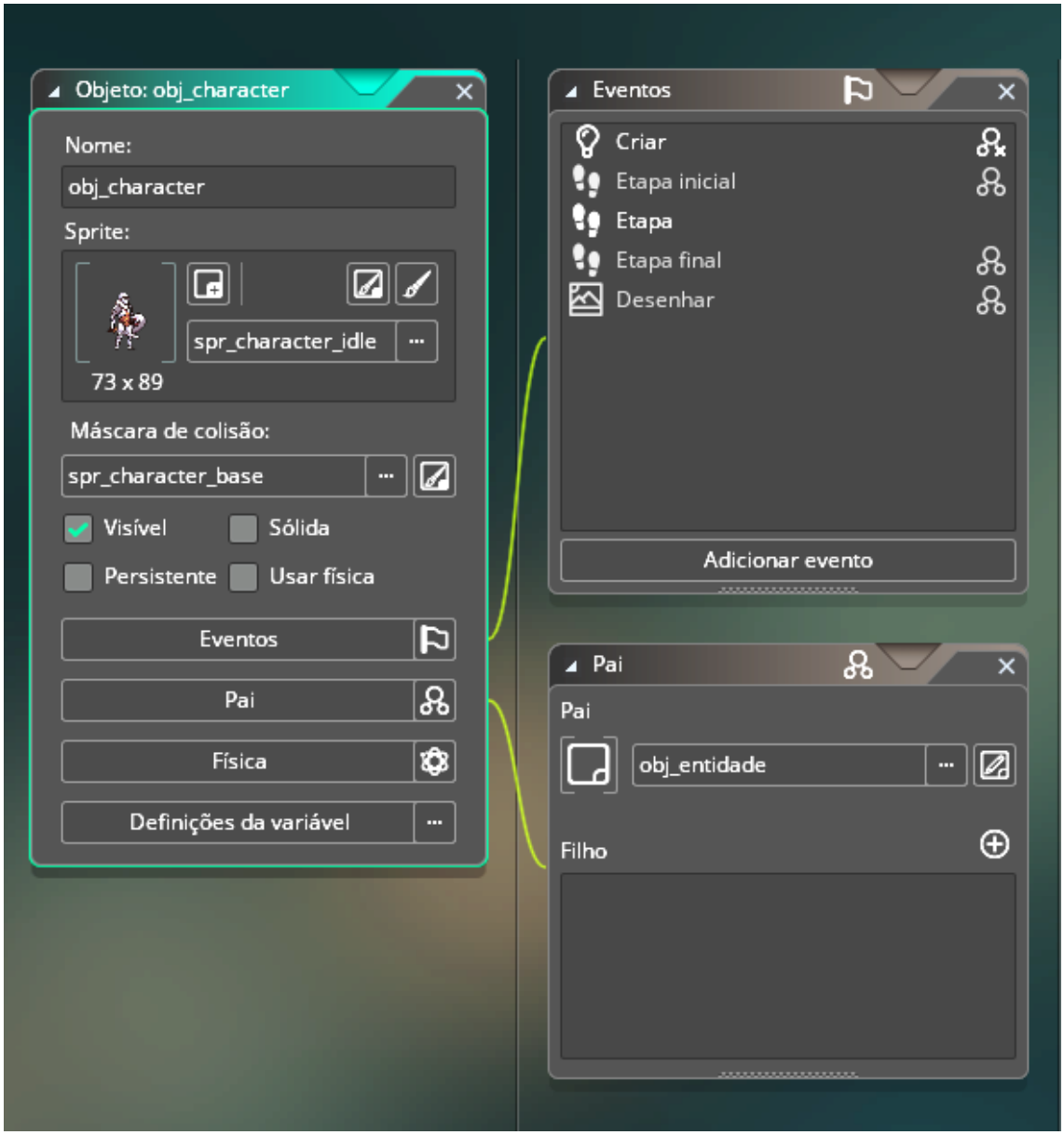
Figura 4: Tela de Editor de Imagem



Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

A tela de Configuração dos Objetos do jogo é apresentada na Figura 5. Nela é possível nomear e identificar os personagens e outras entidades presentes no jogo. Define propriedades de movimento e de visualização, pode-se editar o objeto selecionado, criando eventos, aplicando *sprites*, herdando outros objetos e adicionando máscaras de colisão.

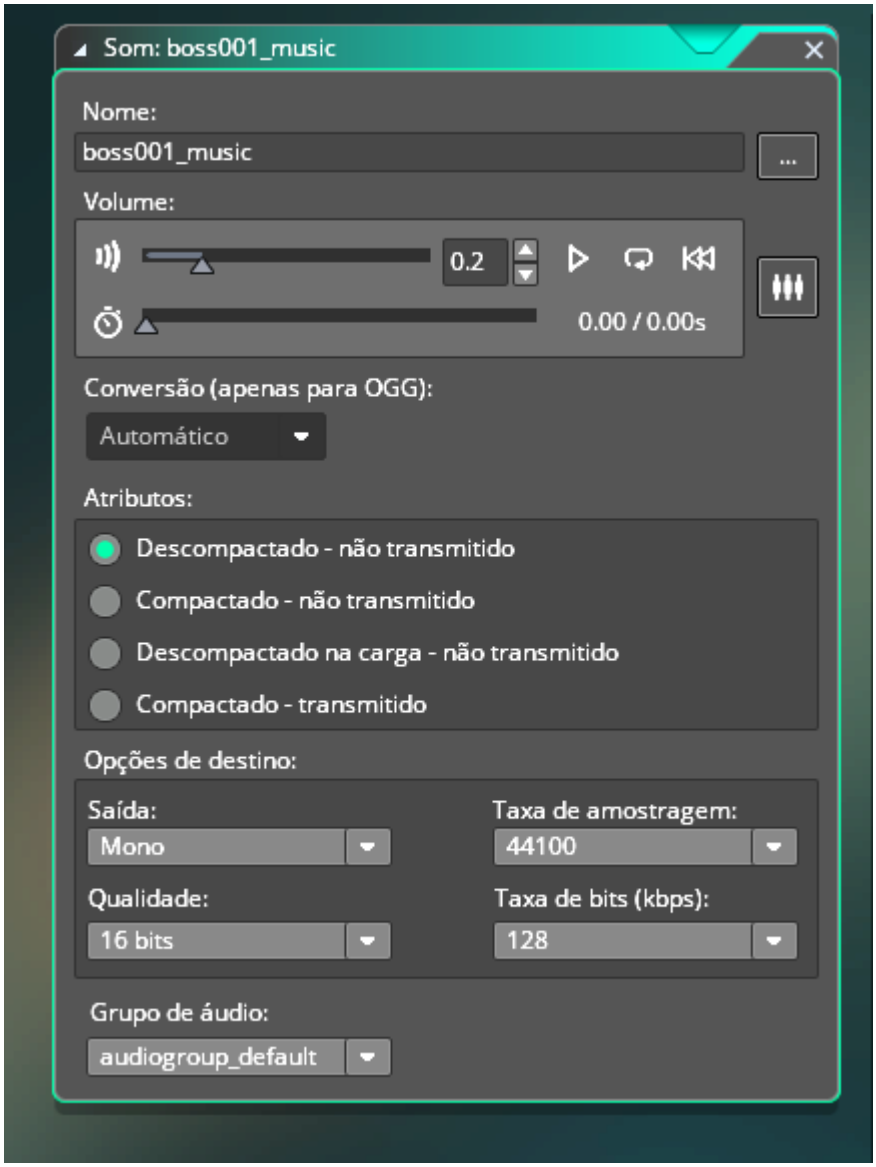
Figura 5: Tela do Objeto



Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

A Figura 6 apresenta a Tela de configuração de Som e suas propriedades de áudio. Nesta pode-se editar a trilha sonora que está presente no jogo, aumentando seu áudio, duração, e configurando as opções de destino do som.

Figura 6: Tela de Som



Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

4.3.3 Linguagem

A linguagem utilizada neste projeto citada no Capítulo 4.3 é a GML, é uma linguagem de script proprietária baseada no paradigma imperativo parecida com a linguagem C++, utilizada para definir a lógica de jogos desenvolvidos na *engine GameMaker*. Sua estrutura é orientada a eventos, permitindo a manipulação direta de objetos, instâncias, variáveis, *sprites*, colisões e outros elementos do ambiente de jogo por meio de comandos específicos, com suporte a estruturas de controle, funções definidas pelo usuário, *arrays, structs* e recursos gráficos.

4.4 Etapas de Desenvolvimento deste Protótipo de Jogo Digital

Na sequência, são listadas as etapas de desenvolvimento, que nortearam o planejamento da execução deste trabalho.

4.4.1 Planejamento

O planejamento deste trabalho foi feito com base no cronograma, apresentado no Quadro 3. Sua utilidade foi importante para planejar as partes do projeto colocando a data de início e a data de término de cada tarefa individualmente.

Quadro 3: Cronograma de desenvolvimento do jogo

ID	Tarefa	Início	Duração / Dias	Témino
1	Fazer Salas	04/fev	22	26/fev
2	Fazer *Design* das Salas	07/fev	20	27/fev
3	Fazer *Sprite* das Entidades	07/fev	32	11/mar
4	Programar Itens	14/mar	3	17/mar
5	Programar Jogador	07/fev	44	23/mar
6	Programar Inimigos	23/mar	10	02/abr
7	Programar Chefe	03/abr	3	06/abr
8	Programar Transição de Salas	07/fev	59	07/abr
9	Fazer Trilha Sonora	11/mar	1	12/mar
10	Fazer Final do Jogo	07/abr	1	08/abr
11	Fazer *Menu* do Jogo	07/abr	1	08/abr
12	Arrumar Detalhes Mínimos	07/abr	6	13/abr

Fonte: Elaborado pelo autor do trabalho a partir de Microsoft Excel (2025)

O cronograma gráfico apresentado no Quadro 1 foi fundamental para direcionar e organizar as atividades de desenvolvimento para possibilitar o cumprimento dos prazos do projeto. Foi elaborado no início das atividades em 04 de fevereiro de 2025.

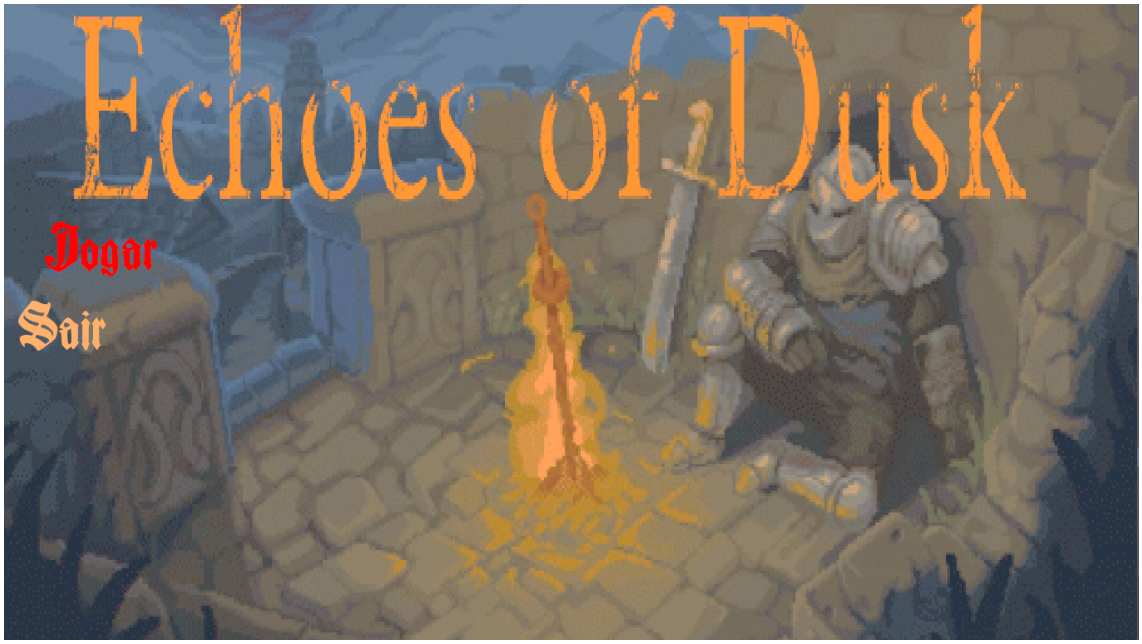
4.4.2 Menu

O *Menu* desenvolvido para o jogo é simplificado em duas opções diretas: Jogar e Sair. Está ilustrado pela Figura 7.

O *Menu* possui um *Graphics Interchange Format*, Formato de Intercâmbio Gráfico (GIF) de um personagem do jogo *Dark Souls* descansando sobre uma fogueira em *pixelart*, para o título do jogo foi usado a fonte *Nightmae Pills* por lembrar a estética *Dark Fantasy*, e apenas

duas opções para o jogador interagir. Foi empregada uma cor laranja no texto do *Menu* como base em *Dark Souls*, onde ela está presente no jogo representando as chamas que é um elemento importante do jogo original.

Figura 7: *Design* do *Menu*



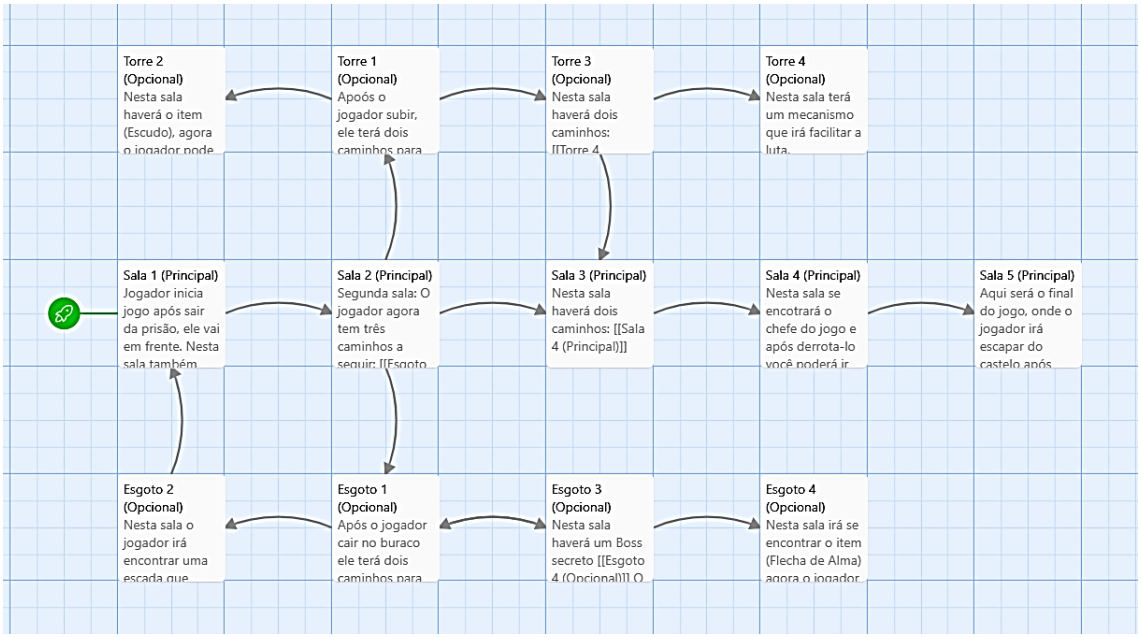
Fonte: Elaborado pelo autor do trabalho a partir de Twinery (2025)

4.4.3 Salas

Os ambientes do jogo estão divididos em três segmentos interligados pelo *level desing* do jogo.

A Figura 8 mostra o planejamento que as salas se encontram, dentro dos três seguimentos principais. Algumas salas podem se conectar com a outra de forma aberta, dando a sensação de todas serem apenas uma, mesmo estando em salas diferentes. Temos 12 salas diferentes que se interligam com o jogo.

Figura 8: Planejamento de Salas



Fonte: Elaborado pelo autor do trabalho a partir de Twinery (2025)

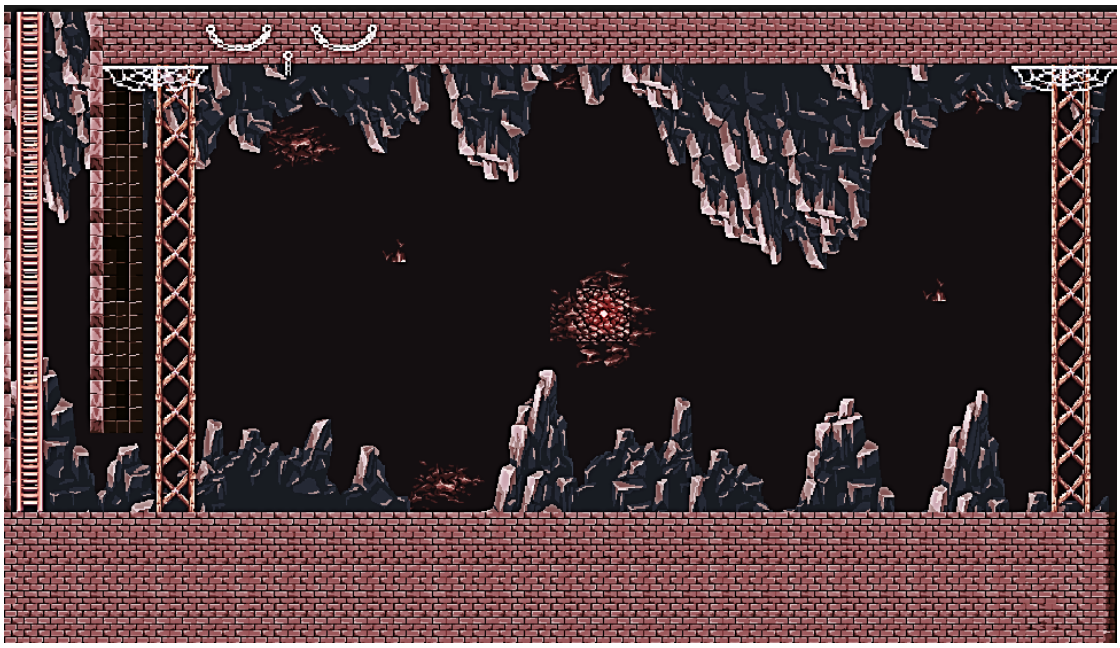
Na Figura 9, tem-se o primeiro seguimento de salas, que constitui a parte central do castelo. Nesta área o jogador se encontra em um ambiente imenso, quase vazio e sombrio, o caminho do castelo é linear, mas tem acesso a áreas que levam o jogador para as outros seguimentos do jogo.

Figura 9: Primeiro Seguimento de Salas

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

A Figura 10, ilustra o segundo seguimento de salas que se passa na parte inferior do castelo, o esgoto. O jogador será transportado para esta área após cair em um buraco no castelo, sem sofrer dano. Nesta área encontrará inimigos diferentes e exclusivos. Neste seguimento há um chefe secreto que, opcionalmente, após o jogador derrotá-lo, receberá um item de recompensa.

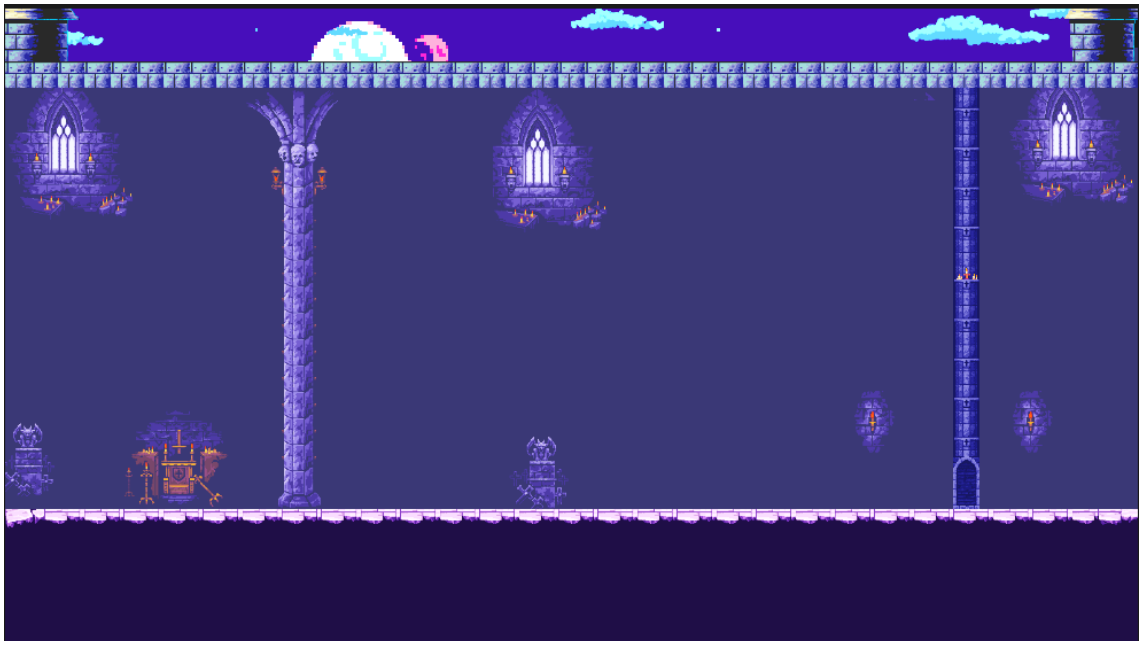
Figura 10: Segundo Seguimento de Salas



Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

O terceiro e último seguimento de Salas é a Torre e é ilustrada pela Figura 11. Localizada na parte superior do Castelo, o jogador terá seu acesso garantido após interagir com um elevador presente no primeiro seguimento. Neste local, o jogador deve seguir principalmente para a direita, onde terá acesso a um item interativo, que reduzirá o poder chefe principal do primeiro seguimento.

Figura 11: Terceiro Seguimento de Salas



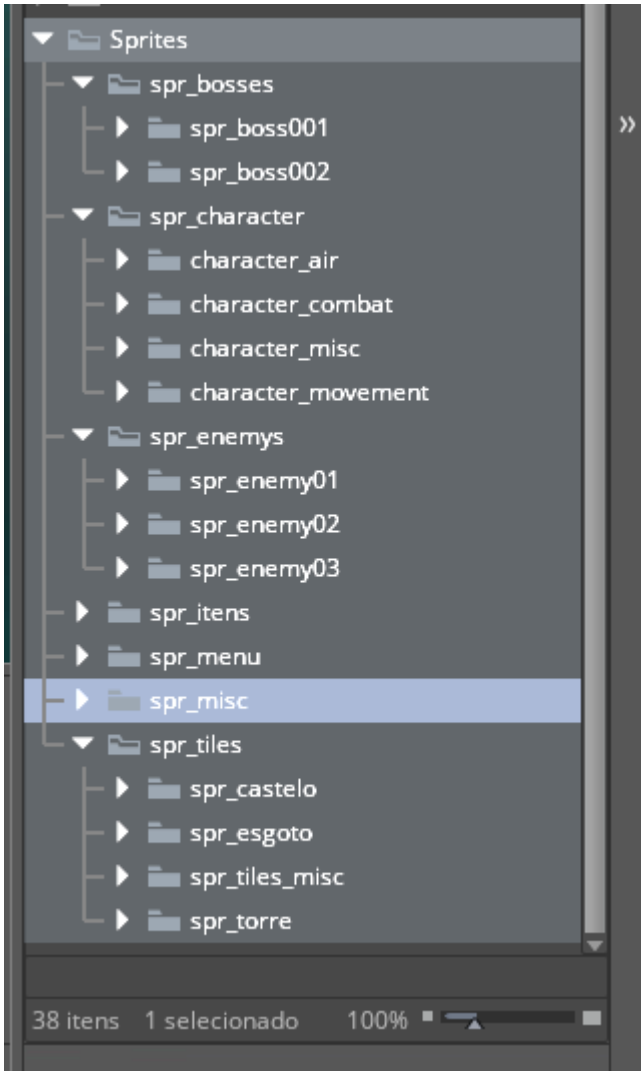
Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

4.4.4 Sprites e tiles

Os *sprites* caracterizam-se por ser a arte dos personagens e itens do jogo. Define as características e propriedades do visual aplicado aos personagens. As *tiles* são a arte do cenário e do ambiente, proporcionando uma estética configurável em cada um dos níveis do jogo.

A Figura 12 apresenta a lista contendo todos os *sprites* presentes no projeto, incluindo também as *tiles*. Cada pasta presente na Figura é organizada para separar cada sprite de um objeto de outro, cada uma possui um modelo de personagem e seus movimentos de animação.

Figura 12: Lista de *Sprites*

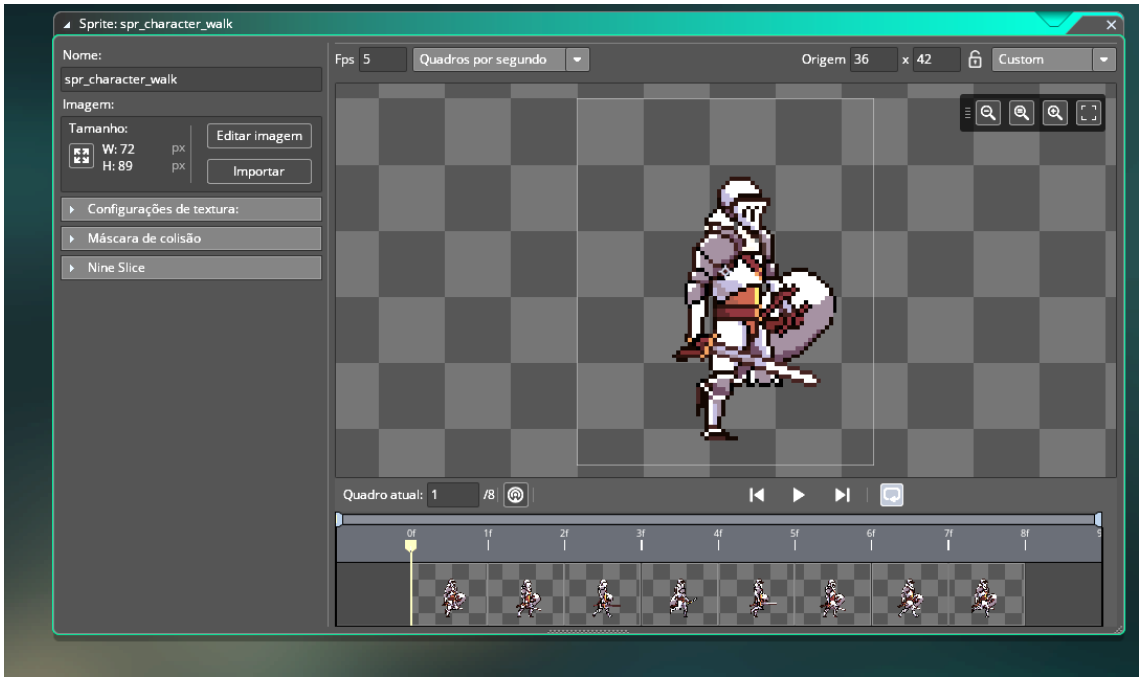


Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

Na Figura 13 é apresentado a tela de edição de *sprite*. Nesta tela, pode-se avaliar quadro a quadro, o sprite escolhido. Também é possível editar o tamanho da figura e sua resolução, configurando a textura, a direção de observação da figura, a máscara de colisão do *sprite* e a opção de importar.

Nesta importação, pode-se carregar uma imagem do banco de dados do computador. Esta imagem importada pode ser editada pelo editor de imagens, conforme ilustra a Figura 14.

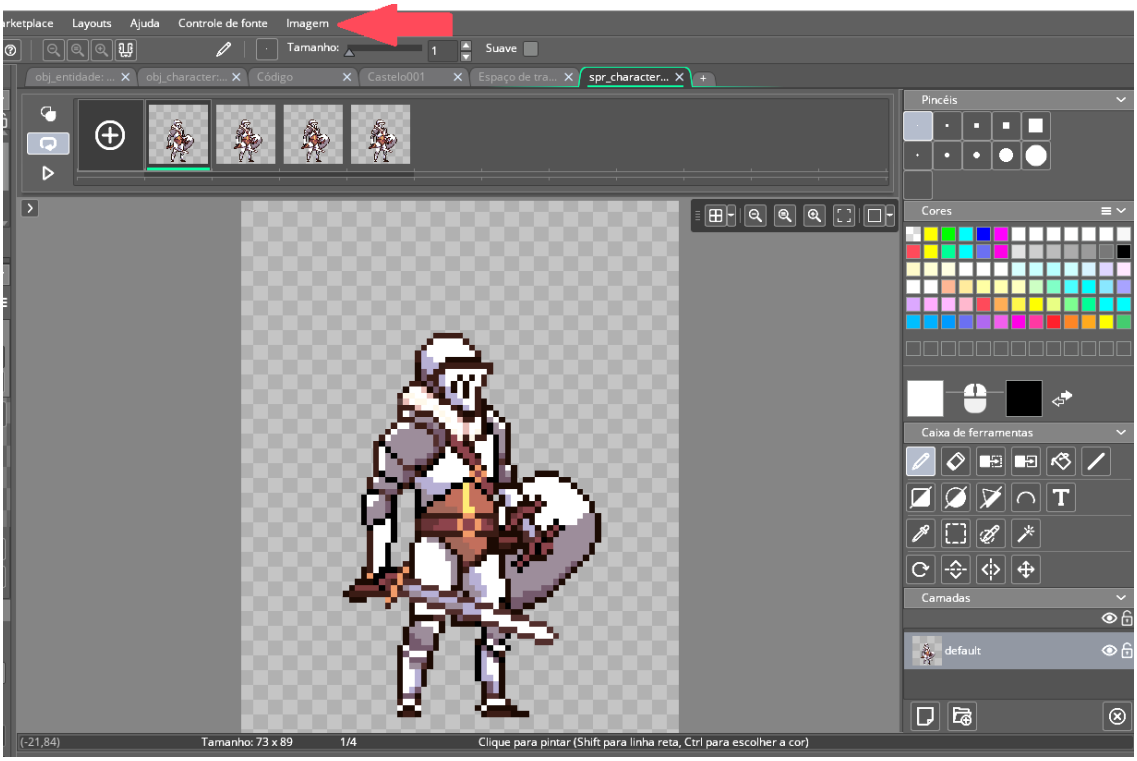
Figura 13: Tela de Edição de Sprite



Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

O editor de imagem é usado para a criação de arte de *sprite* usando ferramentas como lápis, borracha, balde de tinta e ferramenta de configuração de ângulo e posição. Na parte superior da ferramenta há uma opção escrita Imagem indicada pela seta acima, nesta opção pode-se editar, separar e converter os quadros da imagem.

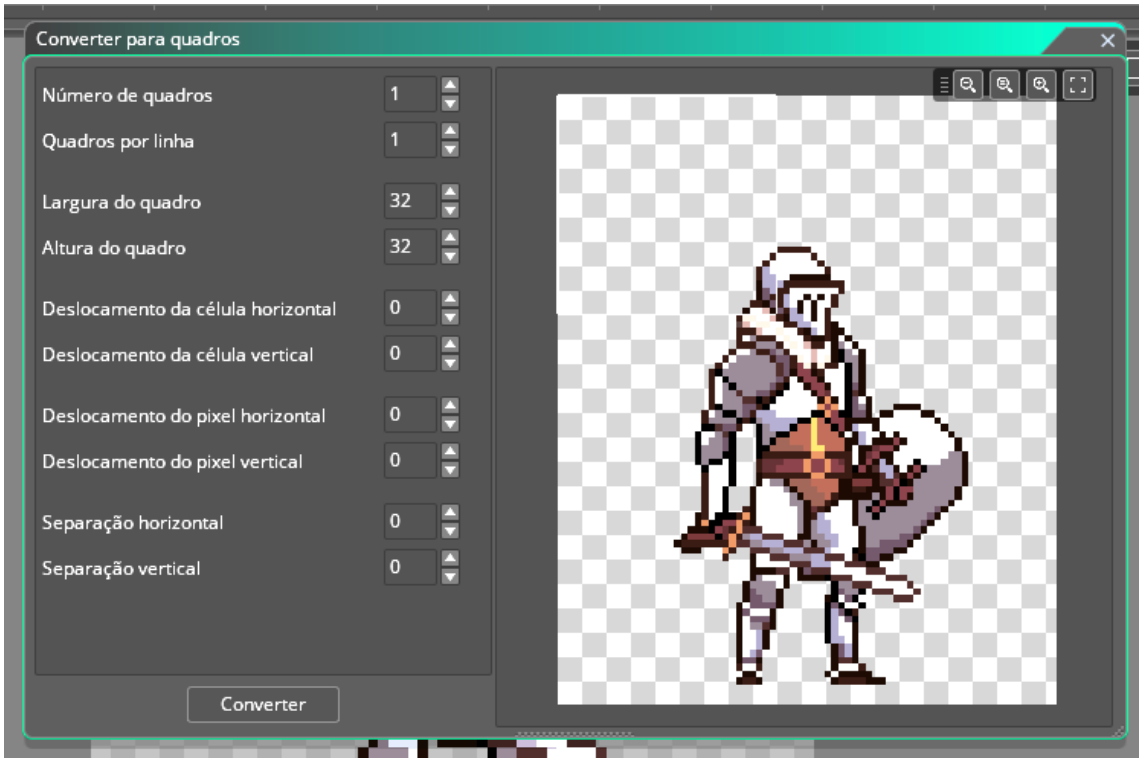
Figura 14: Tela de Editor de Imagem



Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

Na Figura 15 é apresentado a ferramenta de conversão de quadros, onde configuram-se o número de quadros e a quantidade de quadros por linha, que se pode separar da imagem. Criando assim, um seguimento de quadros, que resulta em uma animação para o *sprite*.

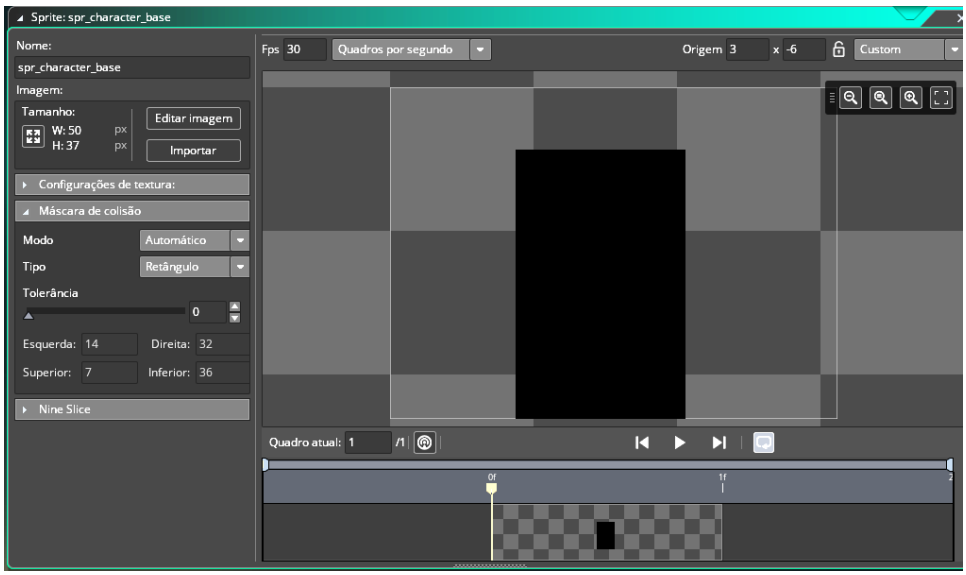
Figura 15: Tela de Conversão de Quadros



Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

Mesmo a ferramenta de edição de sprite permitir criar sua própria máscara de colisão, foi criada uma dedicada para o personagem. Dessa forma, unifica a configuração das máscaras de colisão em cada sprite do personagem. A Figura 16 ilustra esta tela de configuração.

Figura 16: Máscara de Colisão do Personagem



Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

4.4.5 Personagens

Os personagens do jogo foram elaborados com semelhanças aos do jogo original *Dark Souls*, são personagens com estética medieval e fantasiosa. A Figura 17 ilustra a imagem do personagem principal.

O personagem principal é um cavaleiro que possui uma espada e um escudo, no início do jogo o escudo do personagem estará quebrado e não conseguirá defender de ataques inimigos. O personagem é inspirado no personagem principal de *Dark Souls*, e a sua armadura é a da capa do jogo original.

Figura 17: Cavaleiro (Jogador)

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

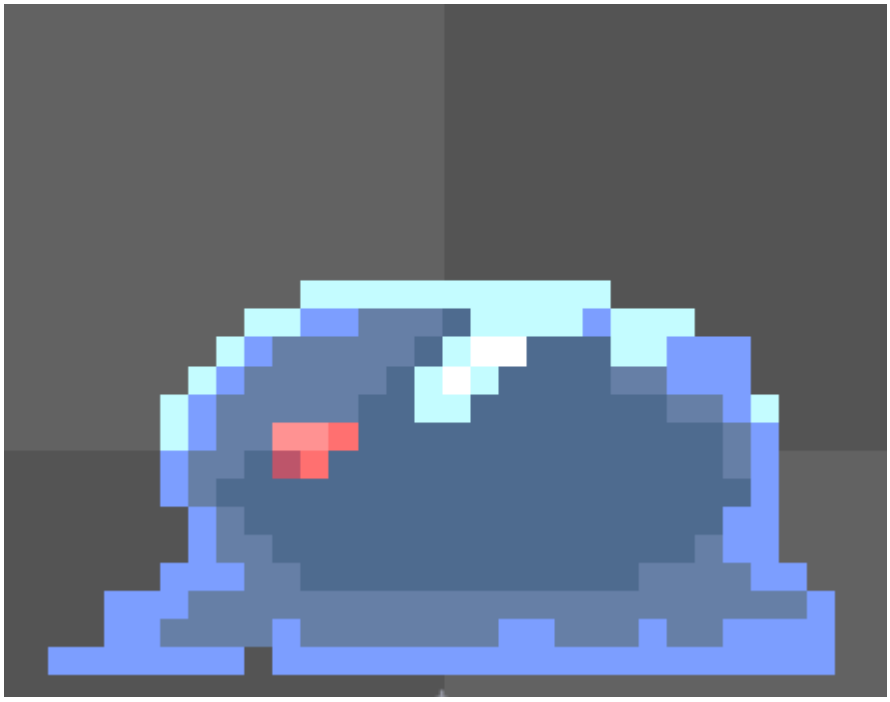
O primeiro o inimigo do jogo é um morto vivo, é um zumbi que anda pelo castelo abandonado. Na história pode ter sido alguém que viva no castelo há muito tempo. É um inimigo de baixa periculosidade e é semelhante ao do jogo original. Há presente diversos mortos vivos em várias áreas do jogo. A Figura 18, ilustra a imagem característica do personagem.

Figura 18: Morto Vivo

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

A Figura 19 ilustra outro personagem inimigo. É um personagem novo caracterizado por uma espécie de gosma conhecida como *Slime*. Vive no esgoto do castelo e anda em grupos, apresenta baixo poder de dano.

Figura 19: *Slime*

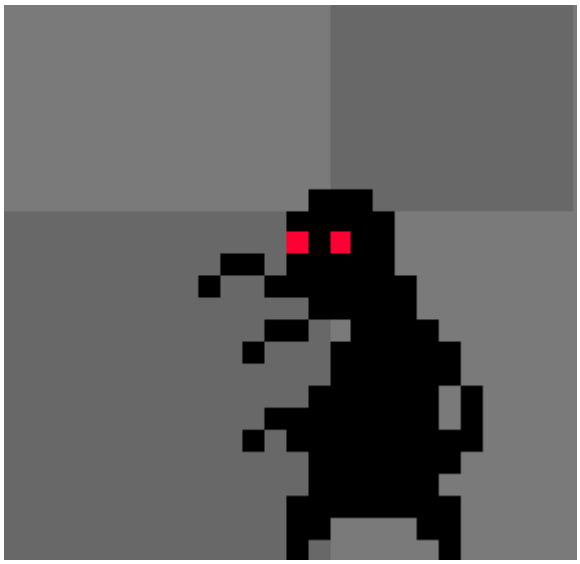


Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

A Figura 20 ilustra outro personagem original do jogo, mantendo uma base estética sombria fantasiosa no estilo *Dark Fantasy*. Na história, O Humanoide Sombrio, anteriormente era humano, mas após ter sido contagiado por uma praga desconhecida, transformou-se em uma criatura humanoide, sombria e de olhos vermelhos. Seu poder de ataque é diferenciado dos

demais por possuir maior alcance para provocar danos. Tornando-se um oponente mais desafiador para o jogador.

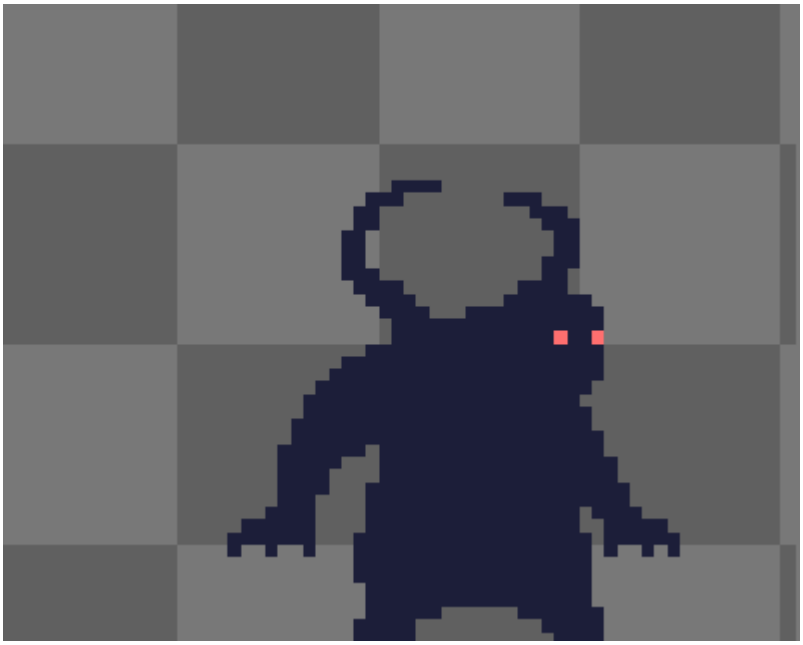
Figura 20: Humanoide Sombrio



Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

Um dos chefes do jogo é da mesma origem do inimigo apresentado na Figura 20. Difere-se do anterior por ser mais poderoso, e é denominado de Humanoide Sombrio Ancestral. Na história, é um humanoide sombrio evoluído, de elevada idade, contendo mutações que lhe conferem maior poder e tamanho. Trata-se de um chefe secreto de elevado nível de desafio, localizado no esgoto, e tem seu enfrentamento pelo jogador como opção adicional. O personagem é ilustrado pela Figura 21.

Figura 21: Humanoide Sombrio Ancestral (Chefe 01)



Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

O principal chefe denominado de Minotauro. Também é um personagem característico deste jogo, mas inspirado em dois personagens do jogo original. Na história ele é responsável por guardar a saída do Castelo, impedindo qualquer fuga do ambiente. O personagem é ilustrado na Figura 22.

O personagem Minotauro têm como arma uma enorme e poderosa espada, desferindo golpes de ataque de elevadíssimo dano. Sendo o maior desafio do jogador. Para vencê-lo, é importante reduzir seu dano de ataque, conseguindo esse objetivo na fase da Torre do jogo.

Figura 22: Minotauro (Chefe 02)

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

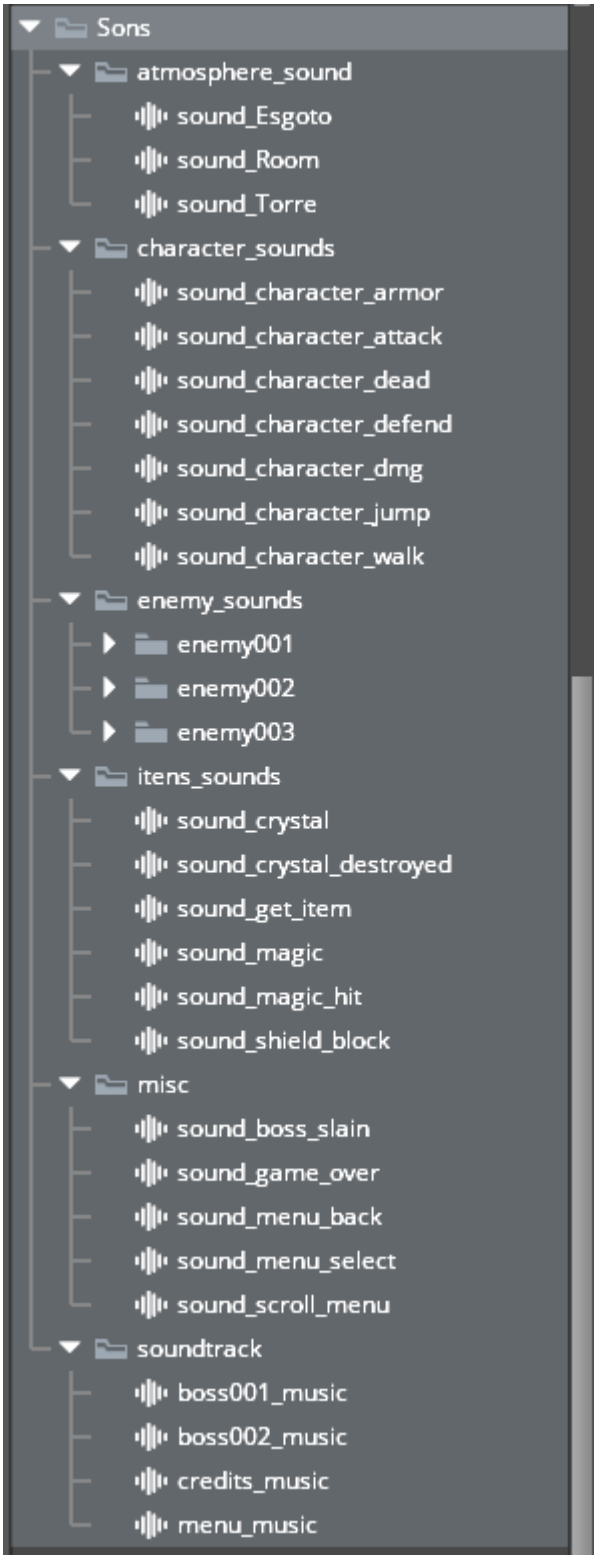
4.4.6 Trilha Sonora

A trilha sonora presente no jogo: músicas e sons ambientes, foram obtidas de áudios disponíveis na plataforma do *Youtub*e sem *copyright*. Já as músicas dos chefes inimigos foram retiradas do próprio jogo *Dark Souls* e possuem direitos autorais. Por este motivo não devem ser divulgadas neste jogo por redes sociais. Serviram apenas para contribuir com este desenvolvimento neste momento em caráter de aprendizado.

A Figura 23, apresenta a janela de configuração das trilhas sonoras. Uma boa trilha sonora é importante para manter o jogador imerso no universo apresentado, prestando atenção e mantendo o foco na dinâmica do jogo.

Nesta lista configuram-se vários grupos de sons separados por sons de atmosfera, sons de personagem, sons de inimigos, músicas, sons de itens e diversos "*misc*" que são mecânicas e outros sons que não possui categoria própria.

Figura 23: Lista de Sons



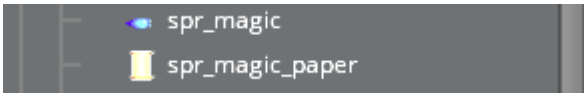
Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

4.4.7 Itens

O desenvolvimento deste jogo procurou empregar poucos itens. Apresentado apenas um item apelidado de magia. Das fermentas, a Figura 24 ilustra a lista de itens de magia. A qual permite ataques a distância maiores.

Diferente do jogo original, um item é capturado é automaticamente equipado ao personagem. Constituindo uma vantagem na experiência do usuário com um novo poder. Está técnica é conhecida como *Power Up*, em jogos do gênero *Metroidvania*, e foi incorporada neste desenvolvimento.

Figura 24: Lista de Itens



Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

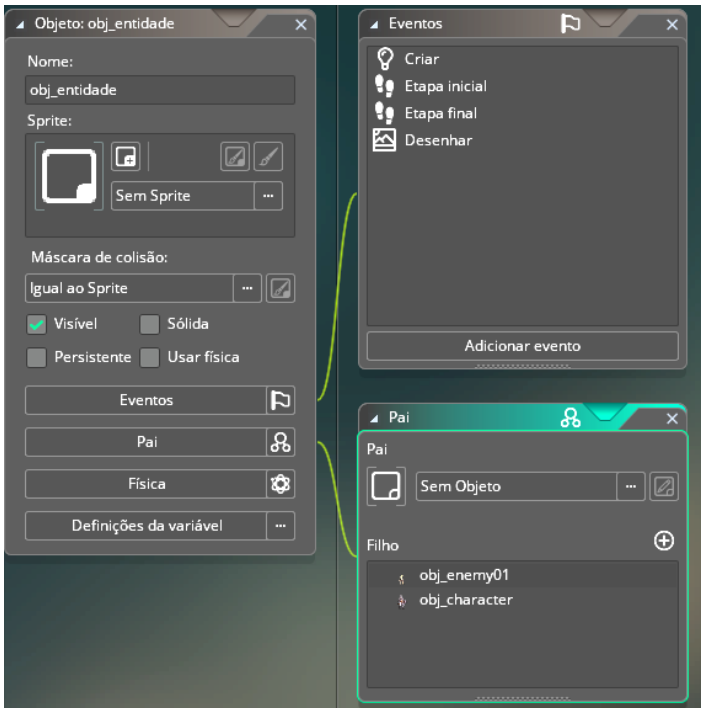
5 TESTES E RESULTADOS

Neste capítulo, são descritos os testes efetuados sobre a produção do jogo e como foi cada etapa do projeto.

5.1 Teste 1: Criando Entidades Pai

A Figura 25 ilustra a configuração de entidades pai, cada entidade no jogo é filho dessa entidade conhecida como obj_entidade, as entidades presentes no jogo inimigos, chefes e o próprio jogador herdam características desta entidade pai.

Figura 25: Tela do Obj_Entidade

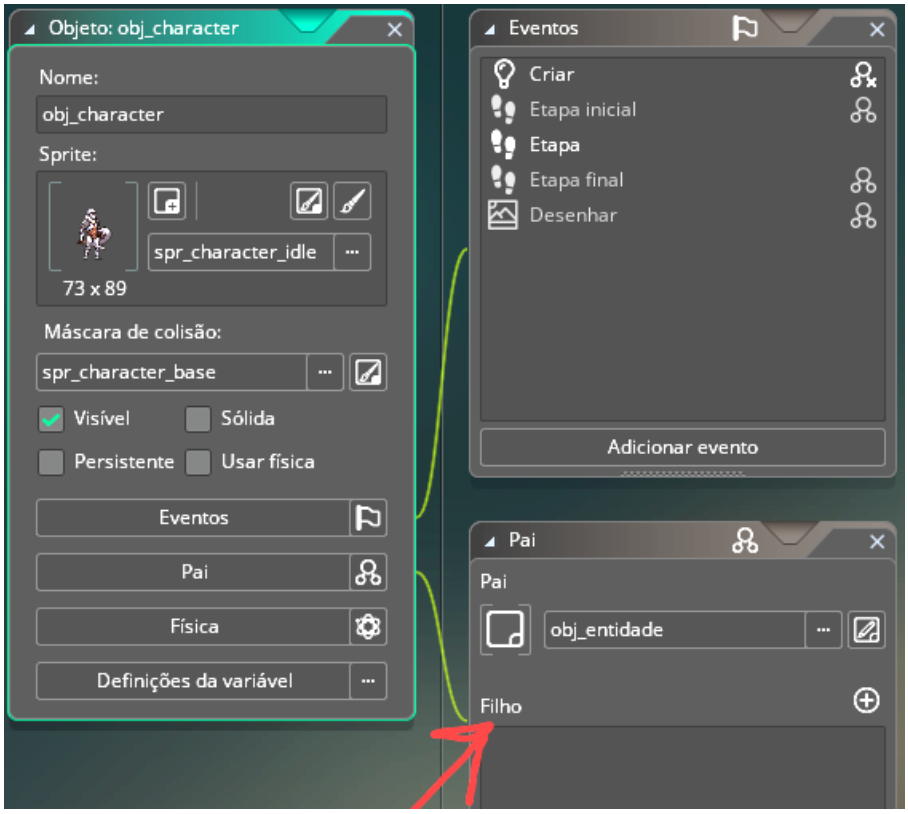


Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

O objeto criado possui eventos que serão herdados para todas as entidades presentes no jogo. Conforme ilustra a Figura 26.

Cada entidade do jogo também possui seu próprio código de acordo com a Figura 26, o jogador herdou alguns eventos do obj_entidade como Etapa Inicial, Etapa Final e Desenhar. Mas também possui seu próprio evento que é o Etapa. Ele também pode subscrever alguns eventos herdados do pai como está no Criar, utilizando as variáveis do pai, mas também tendo suas próprias variáveis.

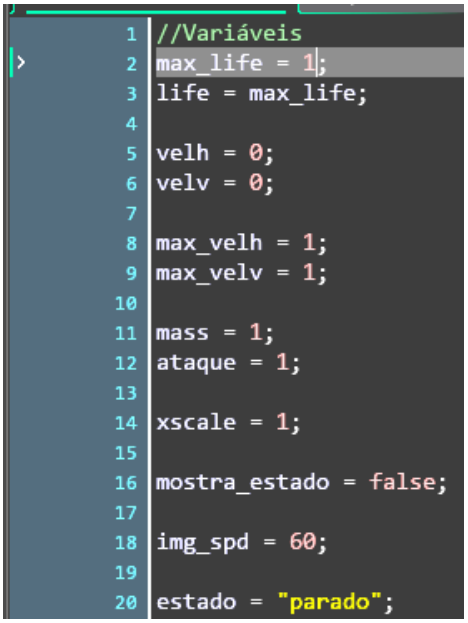
Figura 26: Outros Jogador tendo Obj_Entidade como Pai



Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

A Figura 27 mostra as variáveis que o obj_entidade possui. Essas variáveis são passadas para todos os personagens do jogo, sendo eles inimigos ou o próprio jogador. Os parâmetros de valores máximos que as entidades podem assumir, são os itens: *max_life* (vida máxima da entidade), *life* = *max_life* (vida atual da entidade), velh (velocidade horizontal) e velv (velocidade vertical), max_velh e max_velv. O item *mass* é a massa do objeto. O item ataque é o dano que a entidade irá passar. O item *xsclae* é a escalo do objeto influenciando nos *sprites*. O estado atual em que a entidade se encontra é definido pela mostra_estado. O item *igm-spd* é a velocidade que o *sprite* se comporta, e estado = “parado” é qual estado inicial o objeto irá iniciar.

Figura 27: Evento Criar do Obj_Entidade

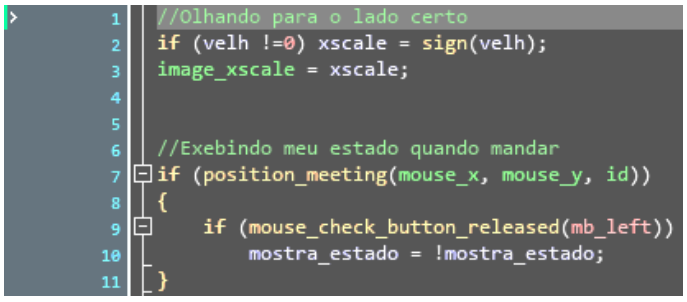


```
1  //Variáveis
2  max_life = 1;
3  life = max_life;
4
5  velh = 0;
6  velv = 0;
7
8  max_velh = 1;
9  max_velv = 1;
10
11 mass = 1;
12 ataque = 1;
13
14 xscale = 1;
15
16 mostra_estado = false;
17
18 img_spd = 60;
19
20 estado = "parado";
```

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

Na Figura 28 está o evento Etapa Inicial presente no obj_entidade, este evento é onde se iniciam as variáveis, configura estados iniciais, define propriedades padrão e prepara o objeto para o que vai acontecer no restante do jogo. Como está apresentado na linha 2 e 3, quando a entidade muda de posição para a esquerda e para direita a imagem irá se virar para a posição que a entidade está se movendo.

Figura 28: Evento Etapa Inicial do Obj_Entidade

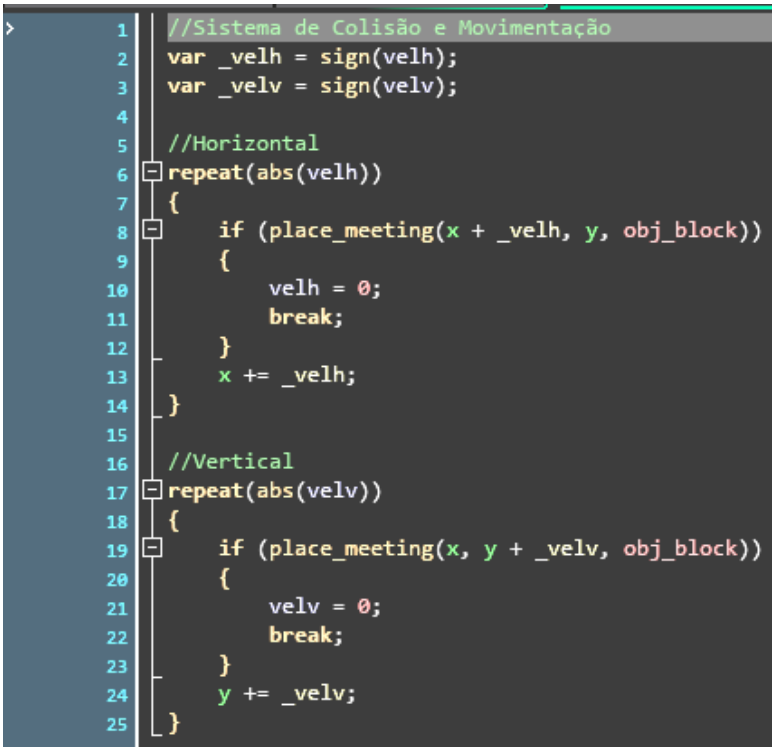


```
1  //Olhando para o lado certo
2  if (velh !=0) xscale = sign(velh);
3  image_xscale = xscale;
4
5
6  //Exebindo meu estado quando mandar
7  if (position_meeting(mouse_x, mouse_y, id))
8  {
9      if (mouse_check_button_released(mb_left))
10         mostra_estado = !mostra_estado;
11 }
```

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

Na Figura 29 está presente a tela de configuração evento Etapa Final, esse evento é executado após todos os outros eventos de Etapa e Colisão terem sido processados. Neste evento do obj_entidade é onde a entidade possui sua colisão com o obj_block que é o chão e paredes do jogo onde a sua movimentação é aplicada nestes objetos.

Figura 29: Evento Etapa Final do Obj_Entidade

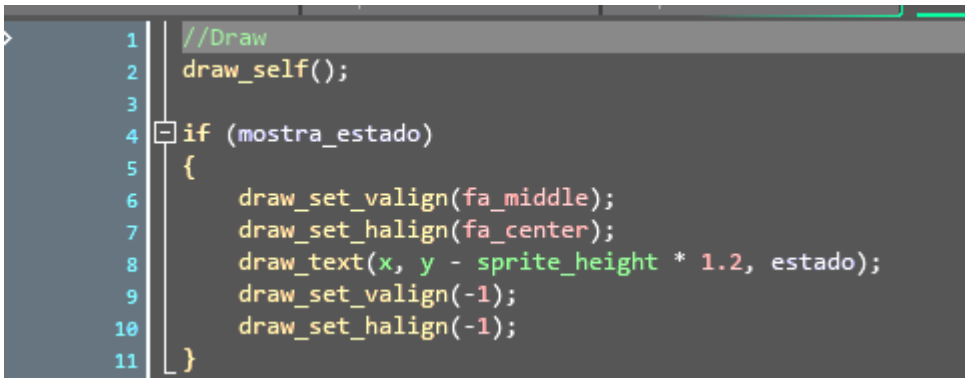


```
//Sistema de Colisão e Movimentação
var _velh = sign(velh);
var _velv = sign(velv);

//Horizontal
repeat(abs(velh))
{
    if (place_meeting(x + _velh, y, obj_block))
    {
        velh = 0;
        break;
    }
    x += _velh;
}

//Vertical
repeat(abs(velv))
{
    if (place_meeting(x, y + _velv, obj_block))
    {
        velv = 0;
        break;
    }
    y += _velv;
}
```

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

Finalmente o evento Desenhar apresentado na Figura 30, foi idealizado para fazer teste de qual estado a entidade se encontra, sendo os estados parado, movendo, dano e morto.

Figura 30: Evento Desenhar do Obj_Entidade



```
//Draw
draw_self();

if (mostra_estado)
{
    draw_set_valign(fa_middle);
    draw_set_halign(fa_center);
    draw_text(x, y - sprite_height * 1.2, estado);
    draw_set_valign(-1);
    draw_set_halign(-1);
}
```

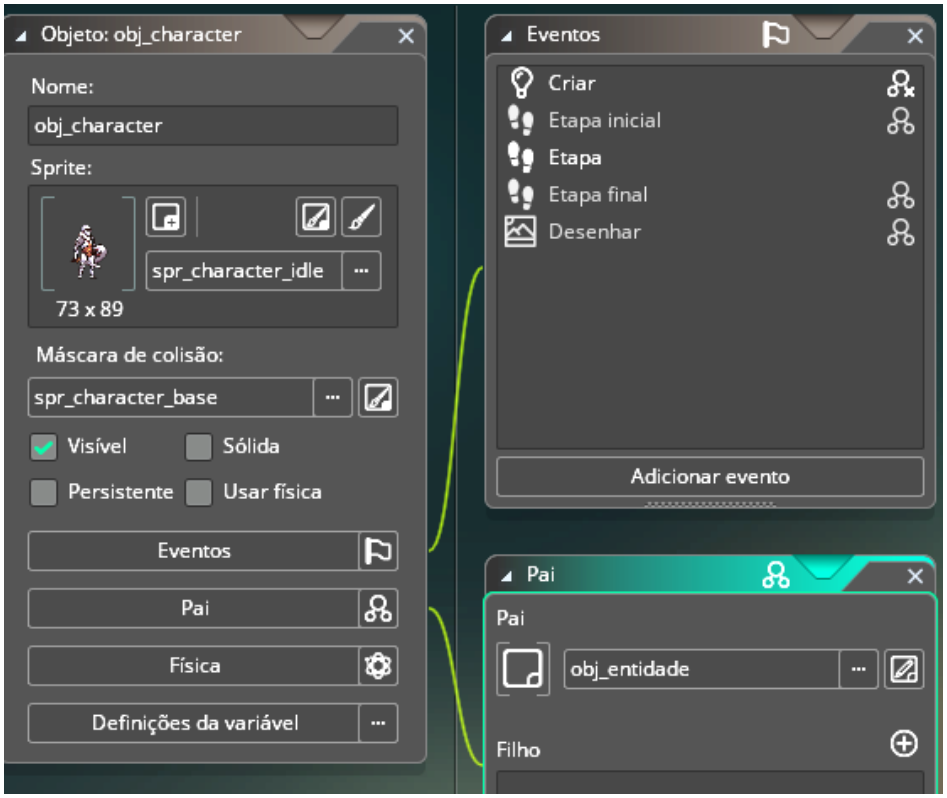
Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

5.2 Teste 2: Movimentação do Personagem

O segundo teste abordou a movimentação do personagem. A movimentação personagem consiste em mover para direita e esquerda, pular, cair e atacar. Sua tela de configuração é ilustrada na Figura 31.

Inicialmente foi criado o objeto que será o personagem do jogador, ele recebeu a herança da programação da entidade pai, e possui seus próprios eventos, que estão ilustrados na tela da Figura 32.

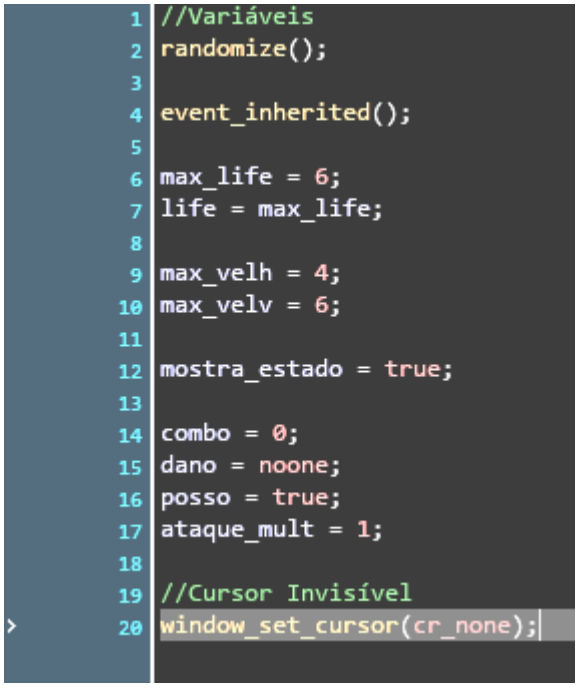
Figura 31: Tela do Objeto do Jogador



Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

Neste evento da Figura 32, inicializam-se as variáveis do personagem do jogador, sendo elas sua vida, sua velocidade, seu estado, suas variáveis de dano, e a adição do cursor invisível durante o jogo.

Figura 32: Evento Criar do Jogador



```
1  //Variáveis
2  randomize();
3
4  event_inherited();
5
6  max_life = 6;
7  life = max_life;
8
9  max_velh = 4;
10 max_velv = 6;
11
12 mostra_estado = true;
13
14 combo = 0;
15 dano = noone;
16 posso = true;
17 ataque_mult = 1;
18
19 //Cursor Invisível
20 window_set_cursor(cr_none);
```

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

Na Figura 33 é apresentado o evento Etapa do jogador onde se inicializam suas variáveis e seu código de movimentação, usando as setas do computador para esquerda e direita, a tecla espaço para pular e o ataque é usado o botão direito do *mouse*.

Figura 33: Evento Etapa do Jogador

```
1  //Iniciando Variáveis
2  var right, left, jump, attack;
3  var chao = place_meeting(x, y + 1, obj_block);
4
5  right = keyboard_check(vk_right);
6  left = keyboard_check(vk_left);
7  jump = keyboard_check(vk_space);
8  attack = mouse_check_button_pressed(mb_right);
9
10
11
12 //Código de Movimentação
13 velh = (right - left) * max_velh * global.vel_mult;
14
15
16 //Aplicando Gravidade
17 if (!chao)
18 {
19     if (velv < max_velv * 4)
20     {
21         velv += GRAVIDADE * mass * global.vel_mult;
22     }
23 }
```

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

A Figura 34, apresenta a tela de configuração do script da gravidade onde seta o valor da variável da gravidade multiplicando por 0.3.

Figura 34: Script da Gravidade

```
1  function scr_ini(){
2      #macro GRAVIDADE .3
3
4  }
```

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

A gravidade está sendo aplicada no jogador utilizando os valores da velocidade vertical do próprio jogador multiplicando-a por 4, e multiplicando novamente pela variável global.vel_mult que apresentada na Figura 35.

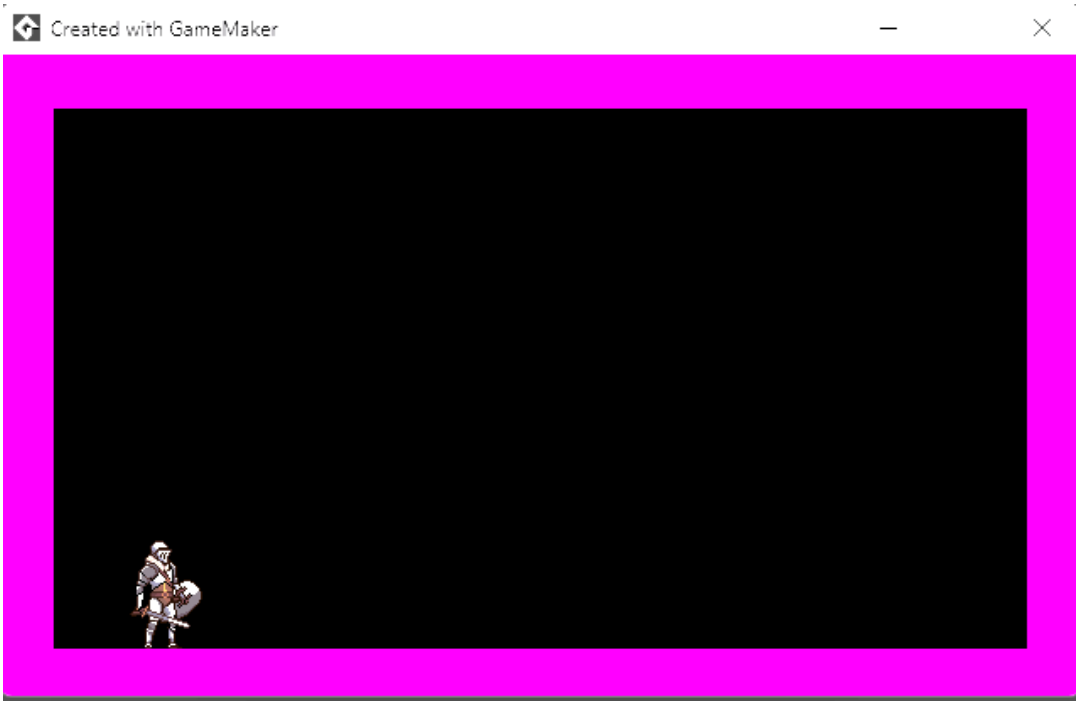
Figura 35: Aplicando Gravidade ao Jogador

```
16  //Aplicando Gravidade
17  if (!chao)
18  {
19      if (velv < max_velv * 4)
20      {
21          velv += GRAVIDADE * mass * global.vel_mult;
22      }
23  }
```

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

Foi criado uma sala fechada onde foi testado a movimentação do jogador para depois alocá-lo para uma área maior. A Figura 36 ilustra este ambiente de teste.

Figura 36: Ambiente de teste de movimentação do Jogador



Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

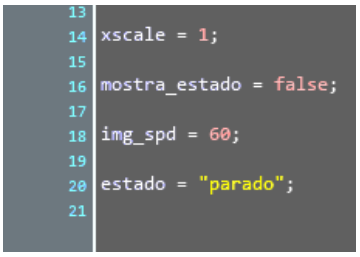
5.3 Teste 3: Máquina de Estados

A Máquina de Estados é utilizada para alternar o comportamento das entidades presentes no jogo, alternando suas ações conforme um evento no jogo é ativado.

Na Figura 37 é mostrada as variáveis presentes para a máquina realizar as trocas de estado, *xscale* é a posição dos sprites ao se trocar de estado, mostra_estado é a visualização para o código de qual estado a entidade se encontra no momento, *img_spd* é a velocidade que a

imagem se comporta quando troca de estado e a variável estado = “parado” é o estado que todas as entidades do jogo irão se iniciar.

Figura 37: Criação da Variável Estado

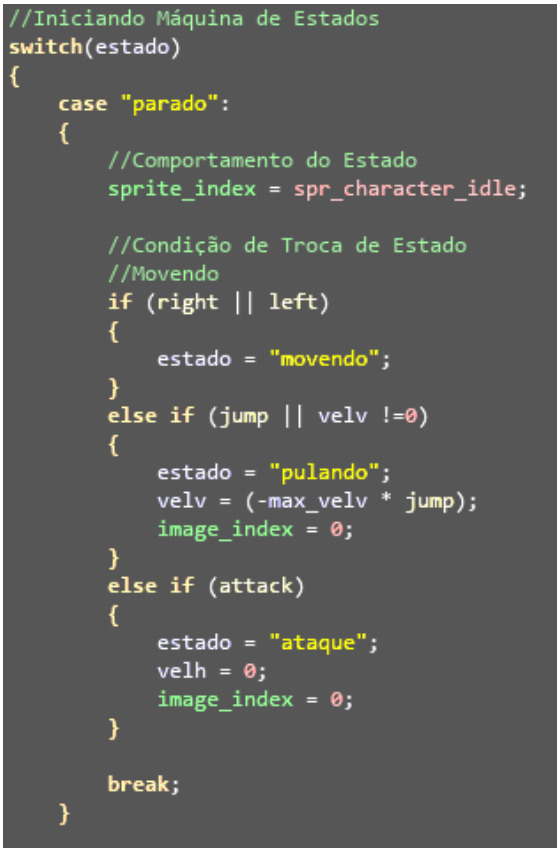


```
13
14 xscale = 1;
15
16 mostra_estado = false;
17
18 img_spd = 60;
19
20 estado = "parado";
21
```

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

Ao iniciar o jogo, a máquina irá entender que o jogador está no estado parado, pois quando se inicia o jogo o personagem é gerado e não faz nenhuma ação, e continuará nesse estado enquanto o jogador não interagir com o personagem. A tela de início da máquina de estado é ilustrado pela Figura 38.

Figura 38: Iniciando a Máquina de Estado (Parado)

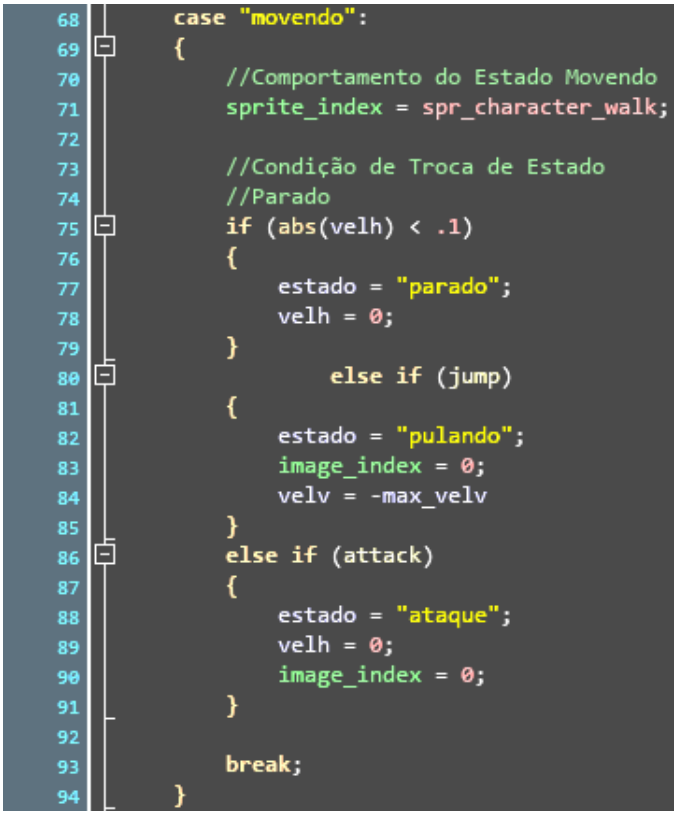


```
//Iniciando Máquina de Estados
switch(estado)
{
    case "parado":
    {
        //Comportamento do Estado
        sprite_index = spr_character_idle;

        //Condição de Troca de Estado
        //Movendo
        if (right || left)
        {
            estado = "movendo";
        }
        else if (jump || velv !=0)
        {
            estado = "pulando";
            velv = (-max_velv * jump);
            image_index = 0;
        }
        else if (attack)
        {
            estado = "ataque";
            velh = 0;
            image_index = 0;
        }

        break;
    }
```

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

A Figura 39 ilustra a primeira troca de estado criada pela movimentação do jogador, quando o jogador se move a máquina percebe que o jogador está em uma velocidade maior que zero, logo irá alternar o *sprite* de parado para o *sprite* de movimentação.

Figura 39: Troca de Estado (Movendo)



```
68      case "movendo":
69      {
70          //Comportamento do Estado Movendo
71          sprite_index = spr_character_walk;
72
73          //Condição de Troca de Estado
74          //Parado
75          if (abs(velh) < .1)
76          {
77              estado = "parado";
78              velh = 0;
79          }
80                  else if (jump)
81          {
82              estado = "pulando";
83              image_index = 0;
84              velv = -max_velv
85          }
86          else if (attack)
87          {
88              estado = "ataque";
89              velh = 0;
90              image_index = 0;
91          }
92
93          break;
94      }
```

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

O próximo estado executado foi o pulo, que quando o jogador apertar a tecla espaço e a máquina entende que o jogador está com sua velocidade vertical acima do valor da variável seu *sprite* é trocado para o de pulo, e quando o jogador está caindo por causa da gravidade o *sprite* é alternado paro o de queda até o jogador chegar ao chão, voltando ao estado parado. O jogador pode se mover enquanto pula e cai, mas não pode usar seu ataque. A Figura 40 ilustra estatela de configuração de pulo.

Figura 40: Troca de Estado (Pulando)

```
97      case "pulando":
98      {
99          //Caindo
100         if (velv > 0)
101         {
102             sprite_index = spr_character_fall;
103         }
104         else
105         {
106             sprite_index = spr_character_jump;
107             //Condição para a animação não se repetir
108             if (image_index >= image_number -1)
109             {
110                 image_index = image_number -1;
111             }
112         }
113
114         //Condição de Troca de Estado
115     if (position_meeting(mouse_x, mouse_y, self))
116         {
117             if (mouse_check_button_pressed(mb_left))
118             {
119                 estado = "Dano"
120             }
121         }
122         if (chao)
123         {
124             estado = "parado";
125         }
126
127         break;
```

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

A troca de estado apresentada foi a de ataque e sequência de combo, quando o jogador iniciar o ataque ao clicar no botão direito do *mouse*, o *sprite* de ataque será iniciado, e caso o jogador ataque mais de uma vez se iniciará o *combo*, que irá alternar entre três *sprites* de ataques diferentes, o jogador enquanto ataca não pode se movimentar ou pular.

Figura 41: Troca de Estado (Ataque e Combo)

```
130     case "ataque":
131     {
132
133         image_speed = 1;
134         velh = 0;
135
136         if (combo ==0)
137         {
138             sprite_index = spr_character_attack01;
139         }
140         else if (combo ==1)
141         {
142             sprite_index = spr_character_attack02
143         }
144         else if (combo ==2)
145         {
146             sprite_index = spr_character_attack03;
147         }
148         if (attack &&  combo < 2 && image_index >= image_number-2)
149         {
150             combo++;
151             image_index = 0;
152         }
153         if (image_index > image_number-1)
154         {
155             estado = "parado";
156             velh = 0;
157             combo = 0;
158         }
159         break;
160     }
```

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

Na Figura 42 é apresentado a troca de estado para o Dano, essa troca ocorre após o jogador sofrer dano, quando o jogador está neste estado ele não pode se mover e após entrar no estado, é inicializada uma variável para ficar invencível por um curto tempo.

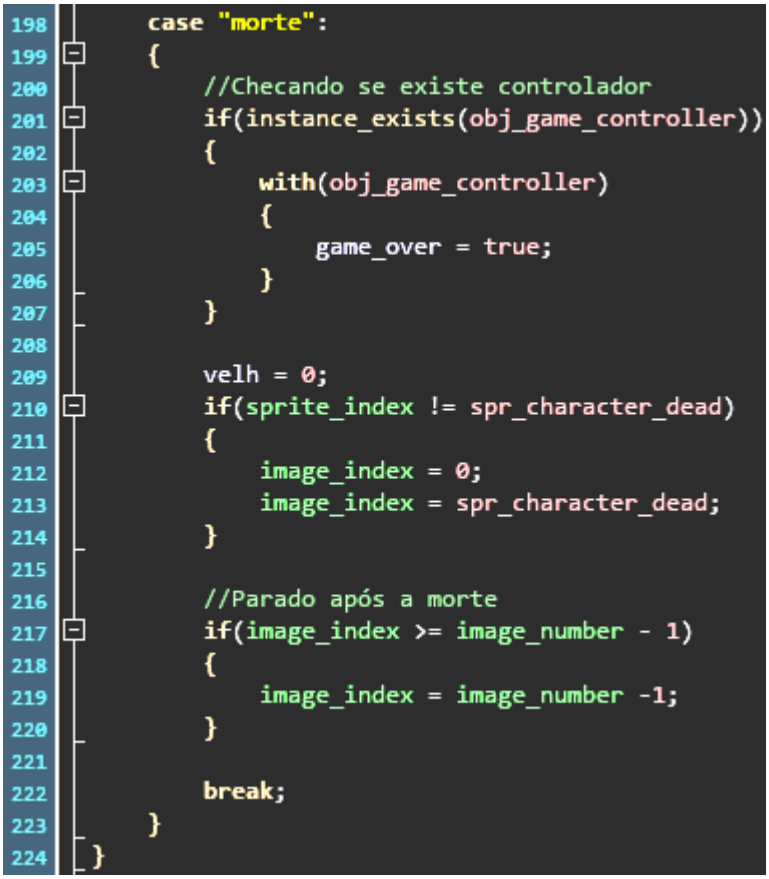
Figura 42: Troca de Estado (Dano)

```
161     case "Dano":
162     {
163         if(sprite_index != spr_character_dmg)
164         {
165             sprite_index = spr_character_dmg;
166             image_index = 0;
167
168             //Deixando Invencivel
169             invencivel = true;
170             tempo_invencivel = invencivel_timer;
171         }
172
173         //Ficar parado após o Dano
174         velh = 0;
175
176         //Saindo do Estado
177
178         //Checando possível Morte
179
180         if (life > 0)
181         {
182         if (image_index >= image_number - 1)
183             {
184                 estado = "parado";
185             }
186         }
187         else
188         {
189             if (image_index >= image_number - 1)
190             {
191                 estado = "morte";
192             }
193         }
194
195         break;
196     }
```

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

O último estado presente na máquina é o estado de morte, o jogador entra neste estado após a sua vida máxima chegar a zero, após isso o jogador irá morrer e a tela de *game* over será acionada na tela do jogo. Esta configuração está ilustrada na Figura 43.

Figura 43: Troca de Estado (Morte)



```
198     case "morte":
199     {
200         //Checando se existe controlador
201         if(instance_exists(obj_game_controller))
202         {
203             with(obj_game_controller)
204             {
205                 game_over = true;
206             }
207         }
208
209         velh = 0;
210         if(sprite_index != spr_character_dead)
211         {
212             image_index = 0;
213             image_index = spr_character_dead;
214         }
215
216         //Parado após a morte
217         if(image_index >= image_number - 1)
218         {
219             image_index = image_number -1;
220         }
221
222         break;
223     }
224 }
```

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

5.4 Teste 4: Inimigos

No Teste 4 foi realizado a criação dos inimigos do jogo, todos os inimigos possuem os mesmos eventos, códigos e o possuem outro pai.

Na Figura é 44 foi criado primeiramente uma entidade pai para os inimigos, que é pai de todos os inimigos do jogo, e possui como pai o obj_entidade, com isso todos os inimigos do jogo possuem uma programação semelhante.

Figura 44: Criando Obj_Entidade_Inimigo

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

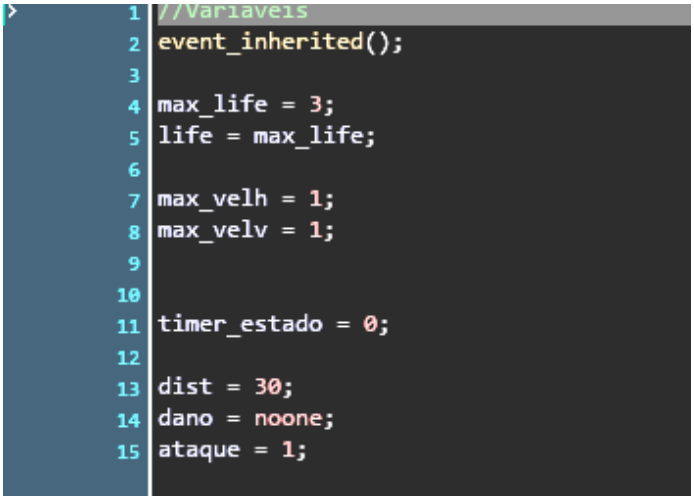
São três objetos de entidade inimigos presentes no jogo que possuem o mesmo código tendo pequenas diferenças mínimas, ambos possuem eventos herdados da entidade pai obj_entidade_inimigo, que tem eventos herdados da sua entidade avô obj_entidade e seus eventos próprios. Estas características de configuração estão ilustradas na Figura 45.

Figura 45: Tela do Obj_Enemy01

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

Na Figura 46 é apresentado as variáveis do inimigo 01 do jogo, ambos possuem as mesmas variáveis, porém com os valores diferentes, *max_life* do inimigo 01 é de valor três, a velocidade vertical e horizontal são a mesma do jogador, *timer*_estado é o valor inicial que ele iniciará seu estado pois sua máquina de estado funciona de maneira diferente do jogador, dist é a distância que o inimigo percebe o jogador e seu ataque tira uma unidade de valor de vida do jogador.

Figura 46: Variáveis do Inimigo

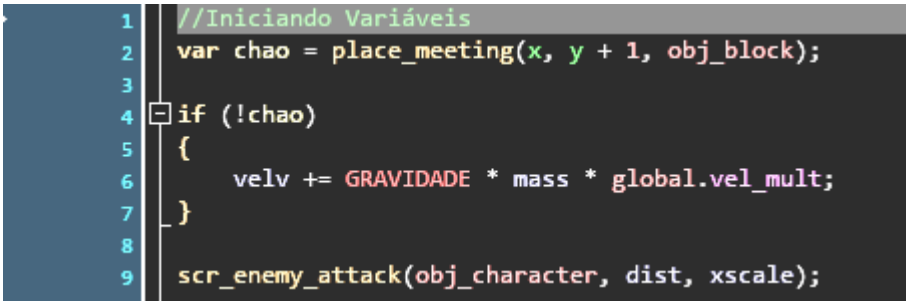


```
1  //Variáveis
2  event_inherited();
3
4  max_life = 3;
5  life = max_life;
6
7  max_velh = 1;
8  max_velv = 1;
9
10
11 timer_estado = 0;
12
13 dist = 30;
14 dano = noone;
15 ataque = 1;
```

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

No começo do evento Etapa do inimigo serão inicializadas suas variáveis assim como ocorre com o jogador apresentado na Figura 46 deste trabalho. Após ser gerado e implementar sua gravidade o inimigo também irá iniciar seu campo de visão apresentado na linha 9 do código, o campo de visão é o que o inimigo enxerga, caso detecte o jogador ele irá iniciar o ataque. A troca de estado do inimigo está ilustrada na Figura 48.

Figura 47: Iniciando Variáveis do Inimigo



```
1  //Iniciando Variáveis
2  var chao = place_meeting(x, y + 1, obj_block);
3
4  if (!chao)
5  {
6      velv += GRAVIDADE * mass * global.vel_mult;
7  }
8
9  scr_enemy_attack(obj_character, dist, xscale);
```

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

A troca de estado do inimigo funciona de forma semelhante ao do jogador, porém sua troca de estado de movimentação acontece de forma aleatória sem precisar de uma ação para iniciá-la, onde o inimigo pode se locomover para qualquer lado da fase, já as outras trocas de estado acontecem após uma ação, como o ataque citado na Figura 44, o estado de dano que irá iniciar após receber um ataque do jogador e a morte da entidade.

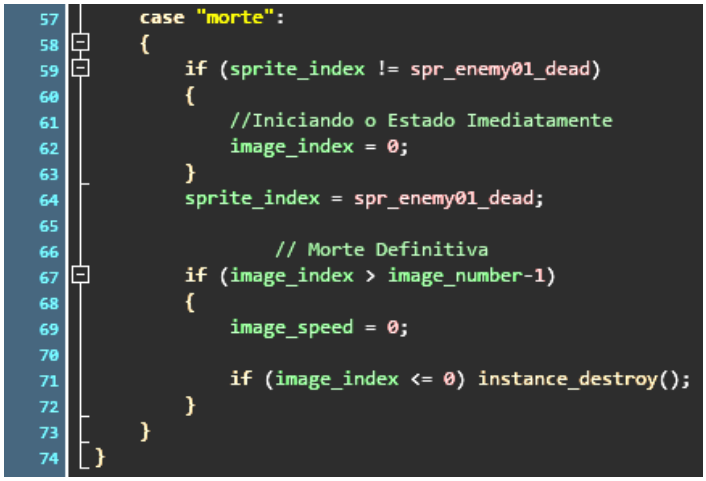
Figura 48: Troca de Estado do Inimigo

```
switch(estado)
{
    case "parado":
    {
        if (sprite_index != spr_enemy01_idle)
        {
            image_index = 0;
        }
        sprite_index = spr_enemy01_idle;

        //Condição de Troca de Estado
        if (position_meeting(mouse_x, mouse_y, self))
        {
            if (mouse_check_button_pressed(mb_right))
            {
                estado = "Dano"
            }
        }

        break;
    }
    case "Dano":
    {
        if (sprite_index != spr_enemy01_dmg)
        {
            //Iniciando o Estado Imediatamente
            image_index = 0;
            life --;
        }
        sprite_index = spr_enemy01_dmg;

        //Condição para sair do Estado
        if (life > 0)
        {
            if (image_index > image_number-1)
```

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

A entidade como apresentado na Figura 49, tem uma quantidade máxima de vida, se esse valor chegar a zero a entidade entrará no estado de morte, onde a entidade irá morrer.

Figura 49: Morte do Inimigo



```
57      case "morte":
58      {
59          if (sprite_index != spr_enemy01_dead)
60          {
61              //Iniciando o Estado Imediatamente
62              image_index = 0;
63          }
64          sprite_index = spr_enemy01_dead;
65
66                  // Morte Definitiva
67          if (image_index > image_number-1)
68          {
69              image_speed = 0;
70
71              if (image_index <= 0) instance_destroy();
72          }
73      }
74  }
```

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

Para fazer o ataque do inimigo foi preciso criar um *script* inteiro apenas para esta mecânica, esse script gera no inimigo um campo de visão, é definido a partir de uma linha reta invisível localizada na parte ocular dos inimigos, onde caso o jogador interaja com esta linha, o inimigo irá ativar o evento de ataque.

Figura 50: Script do Ataque do Inimigo

```
1  function scr_enemy_attack(){
2  var outro = argument0;
3  var dist = argument1;
4  var xscale = argument2;
5
6
7  //Checando se Player está no Campo de Visão
8  var player = collision_line(x, y - sprite_height/-10, x + (dist * xscale), y - sprite_height/-10, outro, 0, 1);
9
10 //Caso veja o Player, ele ataca
11 if (player)
12 {
13     estado = "ataque";
14 }
15
16 }
```

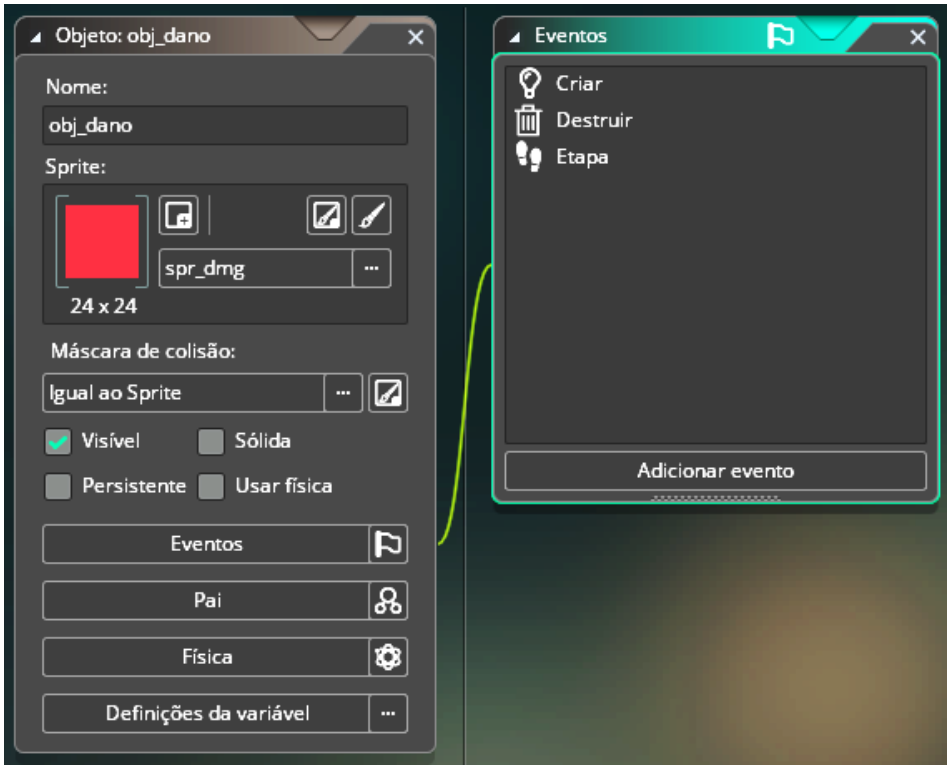
Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

5.5 Teste 5: Dano ao Personagem

Nesta sequência de teste, foi iniciada a análise da dinâmica de danos no personagem principal, com a criação do seu objeto de dano. A Figura 51 ilustra a configuração inicial.

O obj_dano é feito a partir de um quadrado vermelho invisível, sua função é passar o dano de uma entidade para a outra realizando assim a subtração da vida da entidade e uma troca de estado.

Figura 51: Tela do Obj_Dano



Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

O objeto de dano possui as seguintes variáveis e estão ilustradas na Figura 52.

- dano = 0: que o valor zero é baseado no poder que a entidade atacante possui;
- pai = *noone*: sofrerá dano somente entidades que possuem outro tipo de pai;
- image_alpha: é a transparência da imagem que está no valor zero, uma vez que o objeto dano é invisível e não aparece.

Figura 52: Variáveis do Objeto Dano

```
1 //Variáveis
2 dano = 0;
3 pai = noone;
4 image_alpha = 0;
```

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

A inicialização da variável dano é ilustrafa nas Figuras 53 e 54. O objeto dano quando é ativado após uma entidade realizar o ataque, irá aparecer na posição que o ataque foi realizado, assim gerando o dano, após realizar o dano o objeto se destrói para não permanecer na área gerando dano novamente.

Figura 53: Iniciando Variáveis do Objeto Dano

```
var outro = instance_place(x, y, obj_entidade);

if (outro)
{
    if (outro.id !=pai)
    {
        outro.estado = "hit";
        outro.life -= dano;
        instance_destroy();
    }
}
```

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

Caso o objeto dano não encoste em uma entidade que possui a entidade pai, ele irá se limpar e não irá gerar dano.

Figura 54: Limpar Dano caso não encoste no Pai

```
if (pai)
{
    //Limpar
    pai.dano = noone;
}
```

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

As configurações de dano de combo estão apresentadas na Figura 55.

As variáveis apresentadas na entidade do jogador são os valores que o jogador irá aplicar quando gerar o objeto dano de combo.

- dano = *noone*: pois enquanto o jogador não atacar não irá iniciar o dano;
- posso = *true*: é o momento em que o jogador pode iniciar o dano;
- ataque = mult: onde o jogador aplica danos diferentes em ataques diferentes.

Figura 55: Variáveis do Jogador para o Dano de Combo

```
combo = 0;
dano = noone;
posso = true;
ataque_mult = 1;
```

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

A Figura 56, apresenta a configuração final do objeto dano. O objeto dano é criado sendo implementado no evento Etapa das entidades do jogo, após as entidades trocarem seu estado para o estado de ataque será gerado o objeto dano a partir da posição de seu *sprite* de ataque, podendo variar de posição dependendo do tamanho da entidade e tempo de ataque que possui, uma mecânica presente no código do dano que apenas o jogador possui é o sistema de combo,

quanto mais o jogador faz ataques consecutivos, mais seu dano irá aumentar. Caso o objeto dano seja gerado e não detecte nenhuma entidade para receber o dano, ela irá se destruir, e após aplicar o dano na entidade alvo, ela irá se destruir, para não gerar uma área com dano infinito.

Figura 56: Criando Objeto de Dano

```
133         //Criando o Objeto de Dano
134         if (image_index >= 2 && dano == noone && posso)
135         {
136             dano = instance_create_layer(x + sprite_width/3, y - sprite_height/10, layer, obj_dano);
137             dano.dano   = ataque * ataque_mult;
138             dano.pai    = id;
139             posso       = false;
140         }
141
142         if (attack &&  combo < 2 && image_index >= image_number-2)
143         {
144             combo++;
145             image_index = 0;
146             posso = true;
147             ataque_mult += .5;
148             if (dano)
149             {
150                 instance_destroy(dano, false);
151                 dano = noone;
152             }
153         }
154
155         if (image_index > image_number-1)
156         {
157             estado = "parado";
158             velh = 0;
159             combo = 0;
160             posso = true;
161             ataque_mult = 1;
162
163         }
```

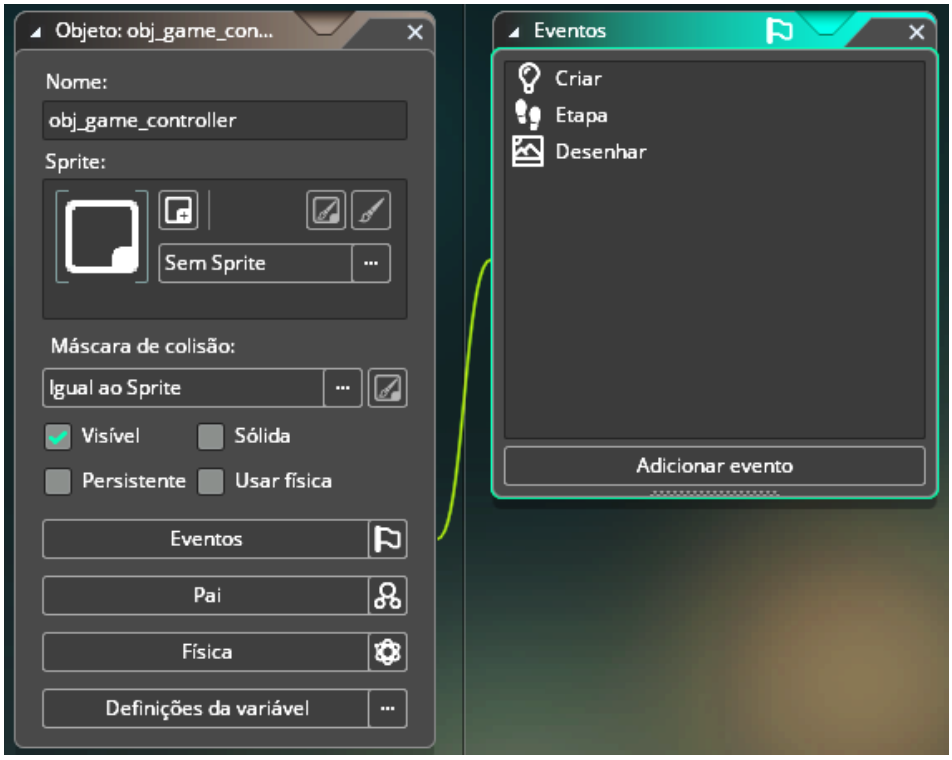
Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

5.6 Teste 6: Tela de Morte

No próximo teste foi elaborado a tela de morte do jogador.

A Figura 57 apresenta a tela de criação. Primeiro é criado um objeto com o nome de *obj_game_controller*, que controla algumas coisas presentes nas salas do jogo e determinados eventos que acontecem após uma ação.

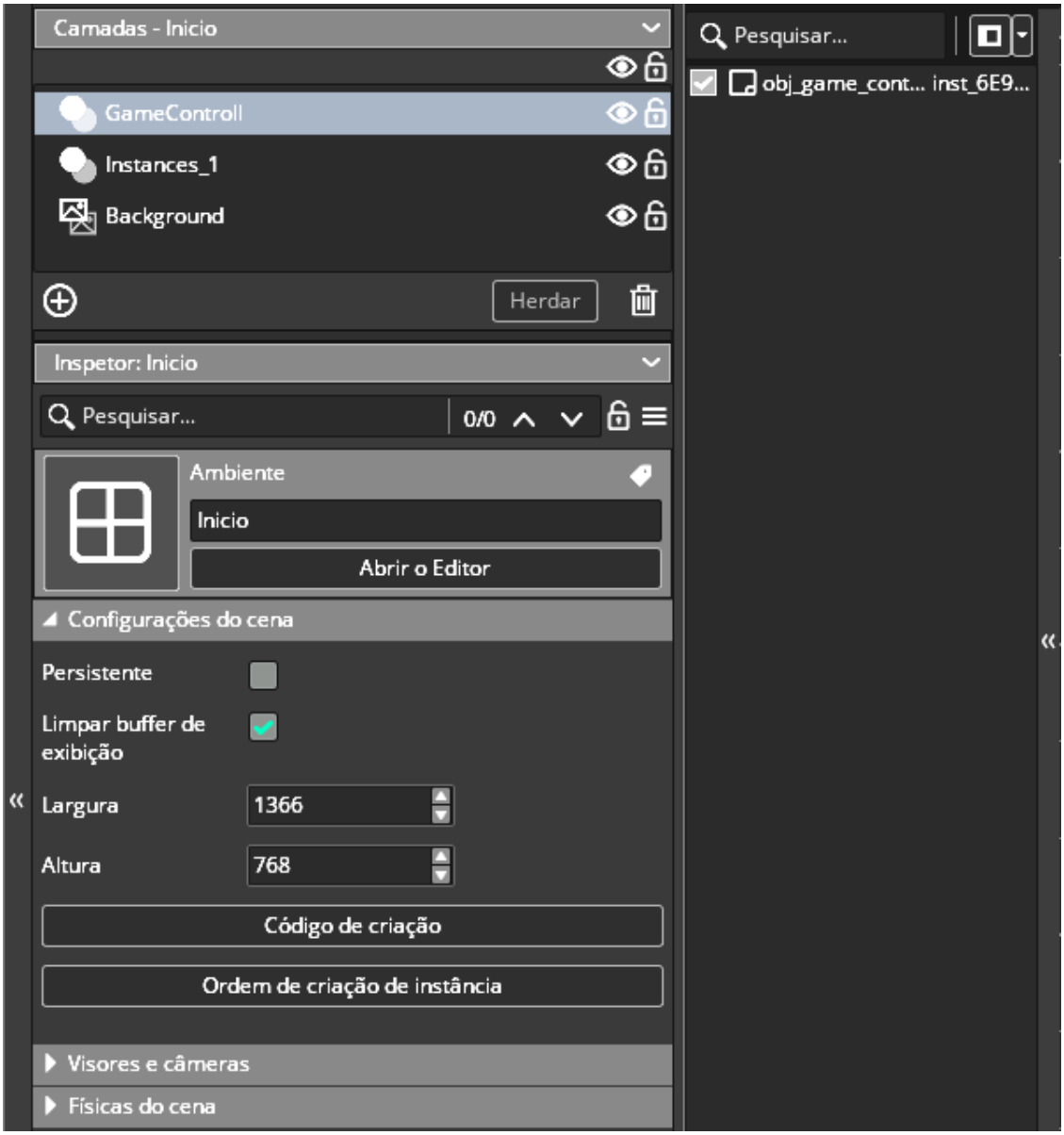
Figura 57: Criando Obj_*Game*_Controller



Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

Para o *obj_game_controller* ser ativado no começo do jogo foi criado uma sala vazia onde o objeto é gerado e após ser ativado passa para a próxima sala, conforme ilustra a Figura 58.

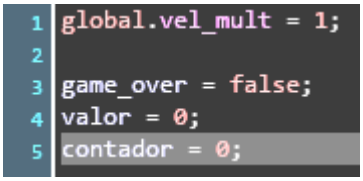
Figura 58: Sala onde o Obj_*Game*_Controller é ativado



Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

É criado dentro deste objeto a variável global.vel_mult, essa variável é a velocidade global do projeto, toda a velocidade do jogo roda com base nessa variável, também possui a variável *game_over*, que irá gerar a tela de morte, o valor inicial que será aplicado neste objeto, e o contador da tela. Estes parâmetros estão indicados na Figura 59.

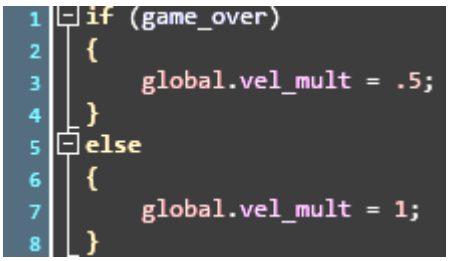
Figura 59: Variáveis do Obj_Controller



Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

A Figura 60, apresenta o evento Etapa do objeto. É iniciado do conceito de quando a tela de *game over*, for acionada a velocidade do jogo será diminuída na metade e caso a tela não apareça o jogo continuará em sua velocidade normal.

Figura 60: Etapa do Obj_Controller



Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

No evento desenhando, ilustrado na Figura 61 é iniciado as bordas pretas serão geradas que quando a tela de morte for acionada, após o jogo ter sua velocidade diminuída pela metade bordas de cor preta serão geradas da parte de cima e da parte de baixo da tela, elas irão se expandir até a posição selecionada e permaneceram lá até o jogador sair da tela.

Figura 61: Iniciando as Bordas

```
//Criando a Tela de Game Over
if (game_over)
{
    //Recebendo Informações
    var x1 = camera_get_view_x(view_camera[0]);
    var w = camera_get_view_width(view_camera[0]);
    var x2 = x1 + w;
    var meio_w = x1 + w/2;
    var y1 = camera_get_view_y(view_camera[0]);
    var h = camera_get_view_height(view_camera[0]);
    var y2 = y1 + h;
    var meio_h = y1 + h/2;



    var qtd = h * .15;

    valor = lerp(valor, 1, .05);

    draw_set_color(c_black);
    //Escurecendo a tela
    draw_set_alpha(valor - 0.3);
    draw_rectangle(x1, y1, x2, y2, false);


    //Desenhando Retângulo de Cima
    draw_set_alpha(1);
    draw_rectangle(x1, y1, x2, y1 + qtd * valor, false);

    //Desenhando o retângulo de baixo
    draw_rectangle(x1, y2, x2, y2 - qtd * valor, false);

    draw_set_alpha(1);
    draw_set_color(-1);
```

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

A Figura 62 apresenta a tela de mensagem final. Após as bordas pretas aparecerem na tela, será exibida uma mensagem na tela escrito "*You – Died*", ela é escrita com cor branca, possui contorno vermelho e tem uma fonte conhecida como Cantara Gotica, que foi escolhida para combinar com a estética do jogo, e abaixo do aviso "*ENTER to restart*" para o jogador poder reiniciar o jogo após morrer.

Figura 62: Iniciando Texto da Tela

```
35
36      //Tempo para aparecer a mensagem
37      if (valor >= .85)
38      {
39          contador = lerp(contador, 1, .01);
40          //Aparecendo Game Over
41          draw_set_alpha(contador);
42          draw_set_font(fnt_gameover);
43          draw_set_valign(1);
44          draw_set_halign(1);
45          //Contorno
46          draw_set_color(c_red);
47          draw_text(meio_w + 1, meio_h + 1, "Y o u - D i e d");
48          //Texto
49          draw_set_color(c_white)
50          draw_text(meio_w, meio_h, "Y o u - D i e d");
51          draw_set_font(-1);
52
53          draw_text(meio_w, meio_h + 100, "ENTER to restart ?");
54
55          draw_set_valign(-1);
56          draw_set_halign(-1);
57
58          draw_set_alpha(-1);
59      }
60  }
61  else
62  {
63      valor = 0;
64  }
```

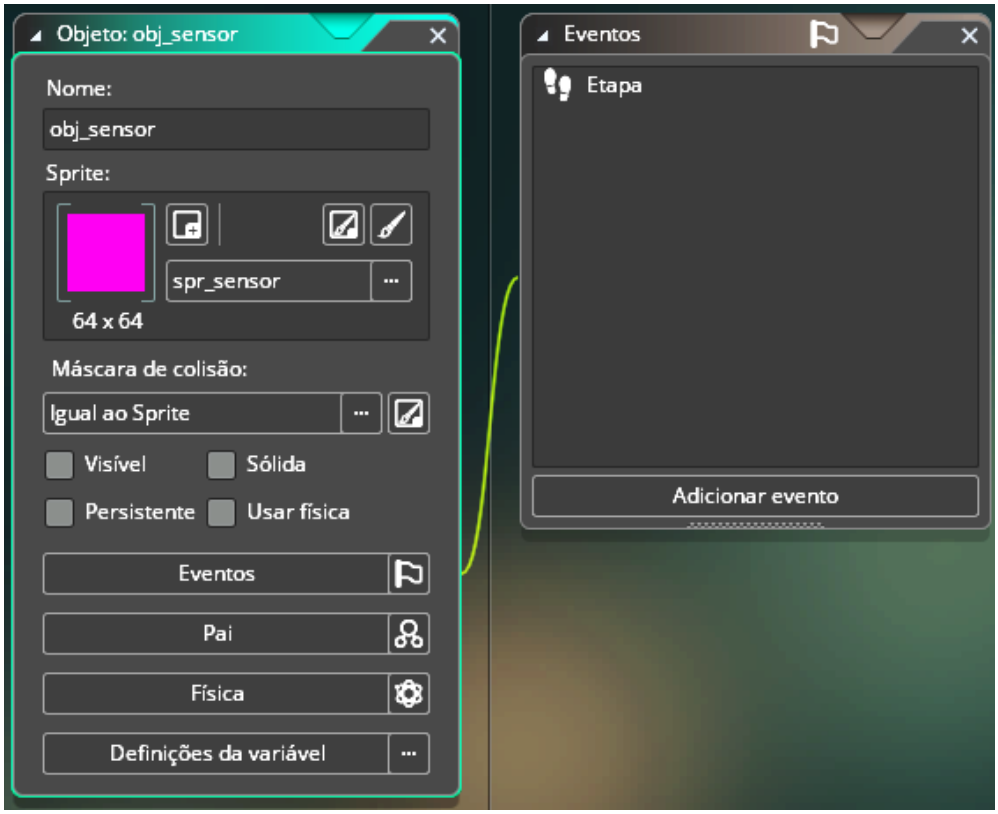
Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

5.7 Teste 7: Configurando as Salas

Como apresentado no Capítulo 4 na Figura 8 o mapa é divido em três áreas, para acessar essas áreas e interligá-las foi preciso criar um sensor e um objeto de transição. Estas configurações de objeto estão ilustradas na Figura 63.

O objeto criado é onde o jogador irá interagir com ele e mudará de sala, há vários deles pelo jogo, mas estão invisíveis aos olhos do jogador.

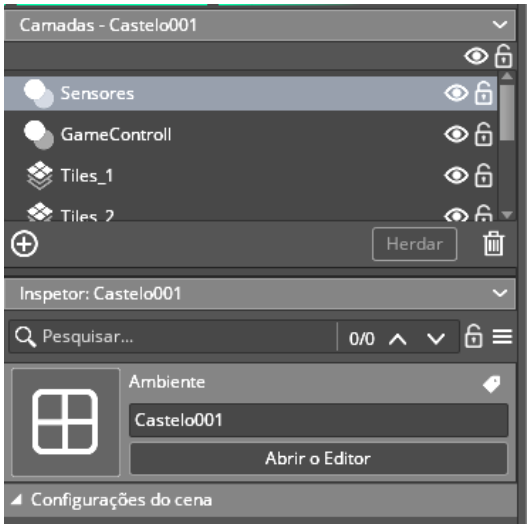
Figura 63: Criando Obj_Sensor



Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

Uma camada foi criada para os sensores, com essa camada eles irão ficar à frente de tudo que há no cenário, fazendo assim, o jogador conseguir interagir com o objeto. Esta característica esta apresentada na Figura 64.

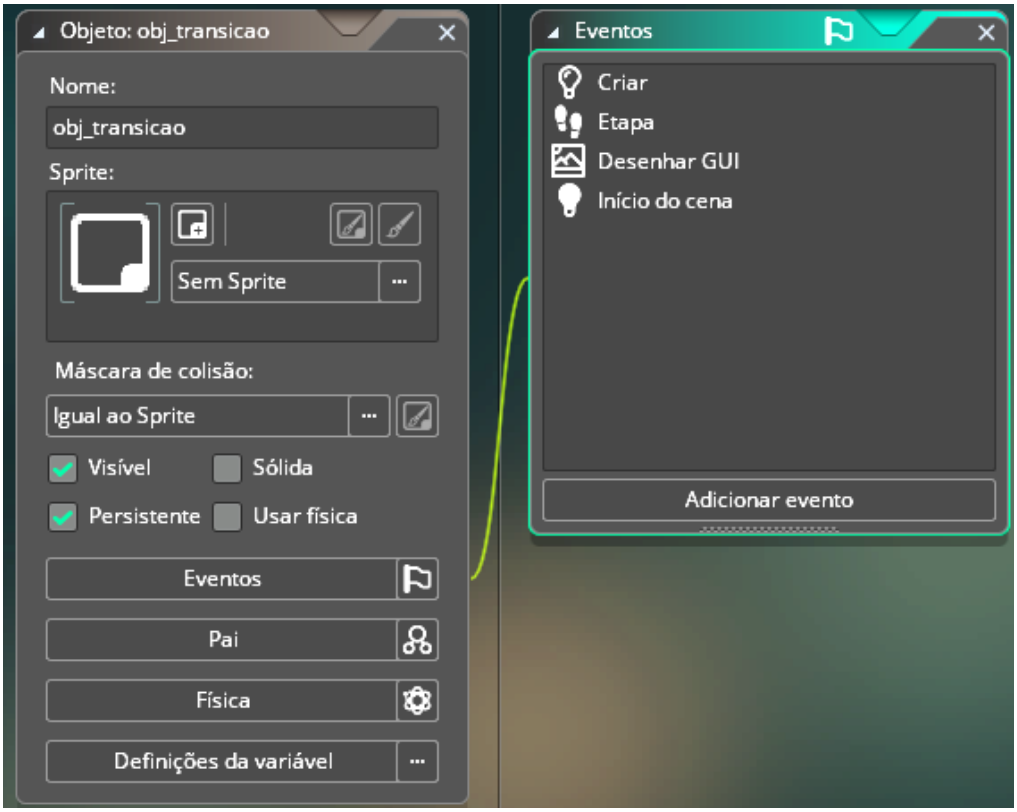
Figura 64: Camada para Sensores



Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

A Figura 65 apresenta o objeto de transição. O objeto de transição é feito para que após o jogador interagir com o sensor, ele será transportado para outra sala, esse objeto é persistente, mesmo após o jogador interagir com o objeto e mudar de sala o objeto não irá desaparecer.

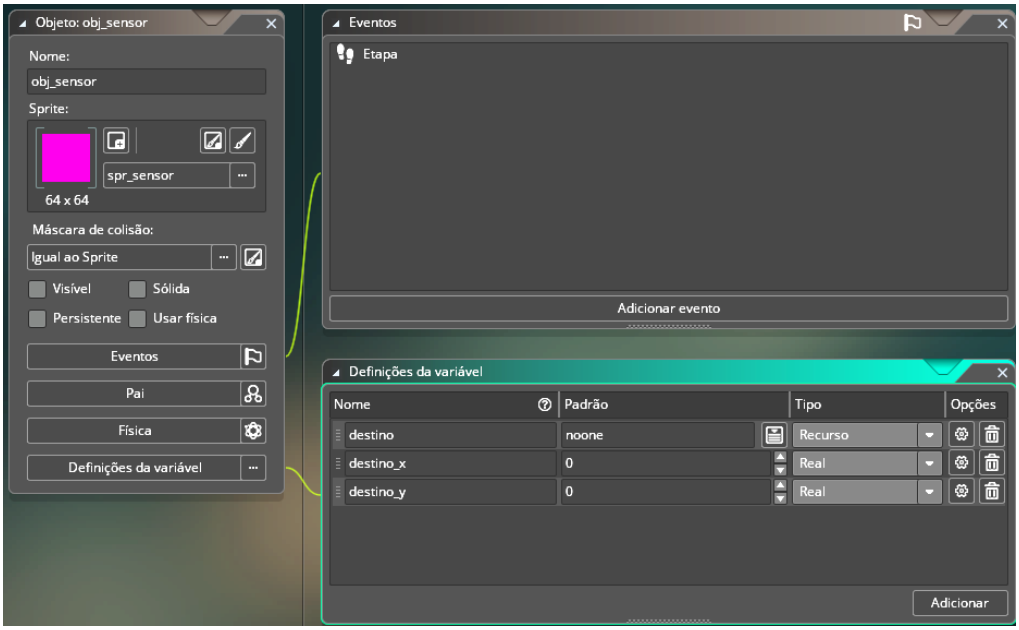
Figura 65: Criando Obj_Transição



Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

A Figura 66, ilustra as definições das variáveis de transporte, para o obj_sensor ser capaz de transportar o jogador de uma sala para outra, precisa definir variáveis no objeto, são elas: destino para qual sala o jogador irá, e destino_x e destino_y, a posição que o jogador estará após ser transportado.

Figura 66: Definindo Variáveis para o Obj_Sensor



Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

A Figura 67 ilustra o *script* empregado para a associação do sensor com as variáveis de transporte. O código do objeto funciona com a checagem do objeto *player*, após entrar em contato e o jogador pressionar a tecla *backspace* para poder interagir com o sensor, ele será ser transportado para o destino que o objeto sensor irá transportá-lo.

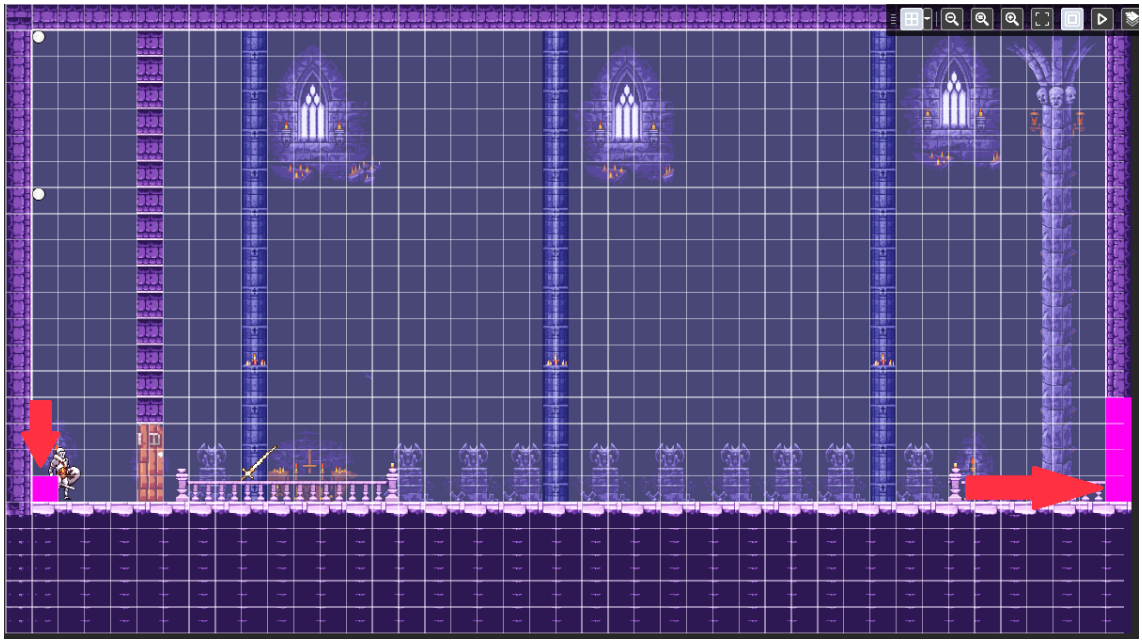
Figura 67: Etapa do Obj_Sensor

```
1  //Checar se o player está colidindo
2  var player = place_meeting(x, y, obj_character);
3
4  //Espaço
5  var espaco = keyboard_check_released(vk_backspace);
6
7  //O Player está colidindo
8  if (player && espaco)
9  {
10     //Código da transição
11     var tran = instance_create_layer(0, 0, layer, obj_transicao);
12     tran.destino = destino;
13     tran.destino_x = destino_x;
14     tran.destino_y = destino_y;
15 }
```

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

Ilustrado na Figura 68, as duas setas indicam a localização dos sensores que transportarão o jogador após interagir com o sensor da sala anterior.

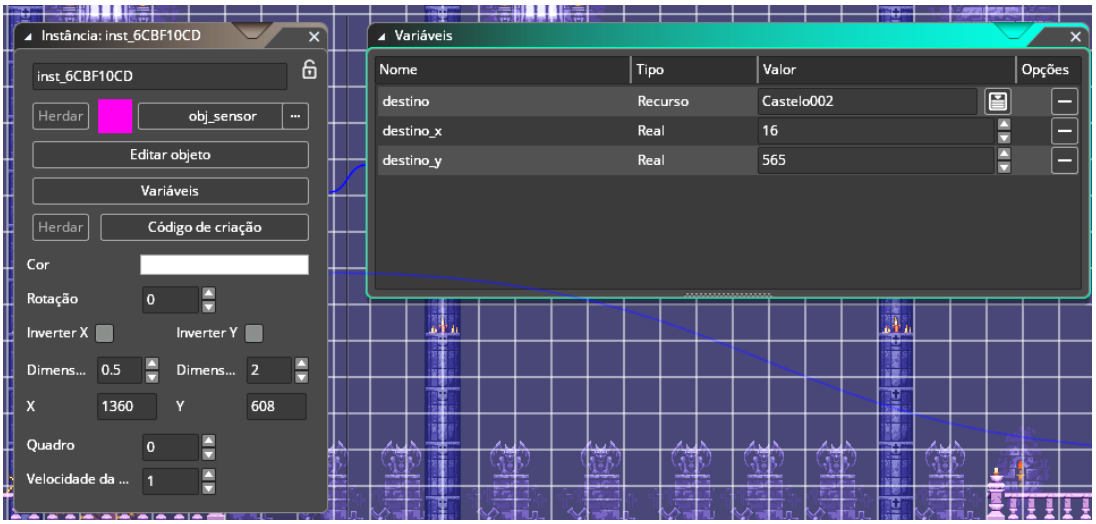
Figura 68: Objeto Sensor na Sala



Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

A Figura 69 apresenta as configurações dos sensores. Cada sensor tem um destino diferente que leva para uma sala diferente, grande parte das salas do jogo possuem dois sensores para o jogador voltar, cada sensor leva para um lado diferente da sala, fazendo com que o jogador possa voltar caso não deseje explorar o jogo.

Figura 69: Destino da Transição



Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

A Figura 70 apresenta o *script* da variável de transição. As variáveis presentes no objeto de transição são a sua transparência, fazendo com que o objeto não apareça para o jogador. A

função “mudei” mudei, que é a variável confirmando para o objeto de transição que o jogador mudou de sala.

Figura 70: Variáveis do Obj_Transição

```
//Transparência
alpha = 0;

//Se eu já mudei de sala
mudei = false;
```

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

A Figura 71 apresenta o *script* da transição do destino do jogador. Nesta parte do código do objeto são acionados: a velocidade da tela de transição, o controle da posição do jogador após chegar no destino, e a destruição do objeto de transição após ser acionado.

Figura 71: Evento Etapa Obj_Transição

```
//Já mudei de sala
if (mudei)
{
    alpha -= .02;
}
else //Ainda não mudei de sala
{
    alpha += .02;
}

//Quando o alpha passar de 1, muda de sala
if (alpha >= 1)
{
    room_goto(destino);

    //Controlando a posição do jogador
    obj_character.x = destino_x;
    obj_character.y = destino_y;
}

//Destruindo Objeto de Transição
if (mudei && alpha <=0)
{
    instance_destroy();
}
```

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

A Figura 72, apresenta o código de operação da tela de transição do jogo, após o jogador interagir com o sensor. A tela ficará preta e após um curto tempo a tela retorna ao normal e o jogador estará na outra sala.

Figura 72: Tela de Transição de Sala

```
1 //Desenhando um retangulo na tela toda
2 draw_set_color(c_black);
3 draw_set_alpha(alpha);
4 draw_rectangle(-1, -1, view_wport[0] + 1, view_hport[0] + 1, false);
5
6 draw_set_color(-1);
7 draw_set_alpha(1);
```

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

Após toda a transição de sala feita pelo código do objeto transição, a variável mudei se torna verdadeira, confirmando que o jogador conseguiu ir para a outra sala. Esta transição pode ser verificada na Figura 73.

Figura 73: Início de Cena

```
1 mudei = true;
```

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

5.8 Teste 8: Deixando o Personagem Invencível

No próximo teste foi realizado o efeito de deixar o personagem invencível após levar dano. A Figura 74 ilustra os valores das variáveis para tal fim.

As variáveis criadas fazem com que o jogador se torne imune a dano por um determinado tempo, o qual seria a velocidade da sala multiplicado por três.

Figura 74: Variáveis para deixar o Jogador Invencível

```
19 invencivel = false;
20 invencivel_timer = room_speed * 3;
21 tempo_invencivel = invencivel_timer;
```

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

No evento Etapa do jogador é aplicado um código dentro do estado de dano onde após o jogador receber dano irá para o estado de invencível, conforme ilustra o código na Figura 75.

Figura 75: Ativando Estado Invencível

```
168             //Deixando Invencivel
169             invencivel = true;
170             tempo_invencivel = invencivel_timer;
```

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

O código presente na Figura 76 é o controle de quanto tempo o jogador ficará no estado invencível, quando o jogador está invencível sua imagem fica transparente por um tempo determinado depois que o a transparência acaba o jogador volta a receber dano.

Figura 76: Controle de Invencibilidade

```
43  //Controle da Invencibilidade
44  if (invencivel && tempo_invencivel > 0)
45  {
46      tempo_invencivel--;
47      image_alpha = .5;
48  }
49  else
50  {
51      invencivel = false;
52      image_alpha = 1;
53  }
```

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

No objeto dano foi implementado um código que quando o objeto dano detecta que o jogador está no estado invencível ele ignora o jogador, conforme ilustra o comando na Figura 77.

Figura 77: Dano ignora Jogador enquanto está Invencível

```
16  //Checando se o alvo está invencivel
17  if (atual.invencivel)
18  {
19      continue;
20  }
```

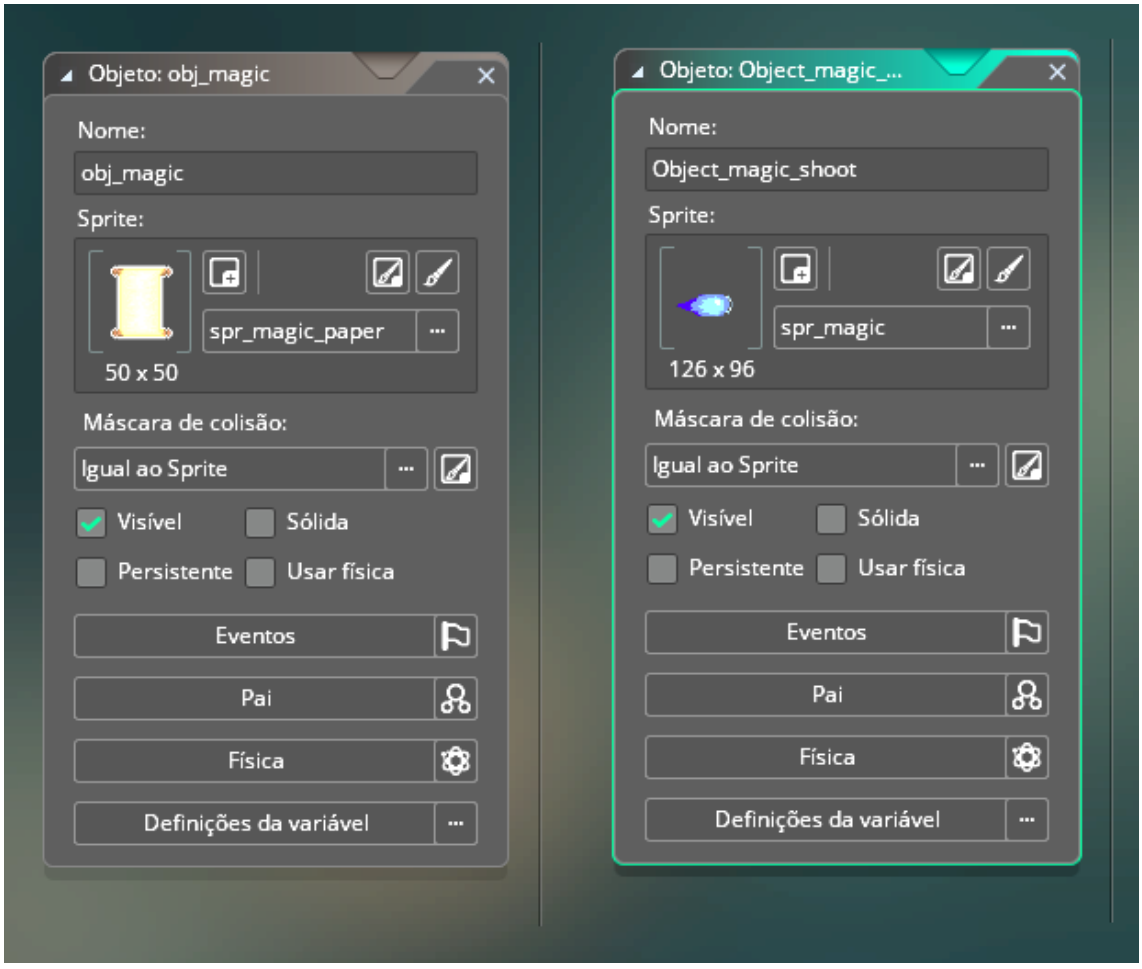
Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

5.9 Teste 9: Coletando e Aplicando Efeito dos Itens

No teste número 9 foi feito a implementação do efeito após coletar os itens do jogo, sendo esse item o tiro mágico. A Figura 78 apresenta as telas de configuração utilizadas.

Foram criados dois objetos para os dois itens do jogo. O poder de magia, foi preciso criar dois objetos para o item. Um para ser o objeto que o jogador irá interagir, e o outro o disparo da magia.

Figura 78: Criando os Objetos de Itens



Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

A Figura 79 apresenta as variáveis de controle. Foram criadas variáveis para o controle de coleta de itens, foi aplicado a variável global por causa da instancia criada para os objetos de itens, o valor dos itens é falso, pois só são ativados após o jogador coletá-los, e a variável *bullets*, representa quantos disparos de magia o jogador pode usar.

Figura 79: Controle de Itens

```
23 //Controle dos Itens
24 //Itens [Escudo, Magia]
25 global.power_ups = [false, false];
26
27 global.bullets = 5;
```

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

A Figura 80 ilustra o *script* no evento Etapa do Jogador há uma região de código localizada na linha 11 a 31, onde está a ativação do efeito do item de magia, que após a coleta do item ele poderá atirar projéteis mágicos.

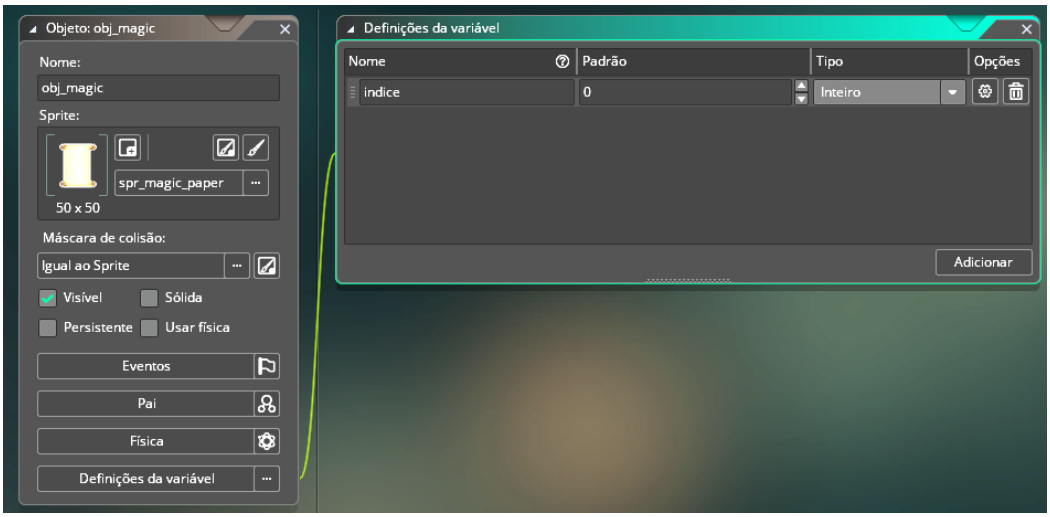
Figura 80: Efeito ao Coletar o Item de Magia

```
11  #region Magia
12
13  var flipped = direction;
14  var gun_x = (x+2) * (flipped)
15  var _xx = x + lengthdir_x(15, image_angle)
16  var y_offset = lengthdir_y(-20, image_angle)
17  if (magic and global.bullets > 0 && global.power_ups[1])
18  {
19      audio_play_sound(sound_magic, 1, 0)
20      with (instance_create_layer(x , y + -50, "Magic", Object_magic_shoot))
21      {
22          global.bullets--;
23          //Velocidade
24          speed = 10;
25          //Direção
26          direction = -90 + 90 * other.image_xscale;
27          //Angulo
28          image_angle = direction;
29      }
30  }
31  #endregion
```

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

A Figura 81 ilustra a opção de escolha de valores dos prêmios do item coletado. Cada objeto possui uma definição de variável chamada de índice que é do tipo inteiro, esta variável permite que o jogador após interagir com o item ele irá adquirir um valor do índice, permitindo que ele libere a habilidade do item, cada item possui um valor de índice diferente.

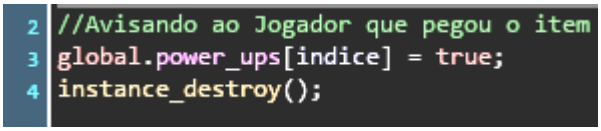
Figura 81: Definindo Variável do Objeto Item



Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

A Figura 82 apresenta a configuração da programação dos eventos de colisão. Nos itens do jogo estão programados eventos de Colisão, após o jogador colidir com o objeto, o item desaparece e faz o jogador usar a habilidade do item.

Figura 82: Colisão do Item com o Jogador



Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

5.10 Teste 10: Chefes do Jogo

No teste de número dez foi feito a criação dos chefes do jogo. Sua base de programação é semelhante à dos inimigos comuns do jogo tendo algumas diferenças.

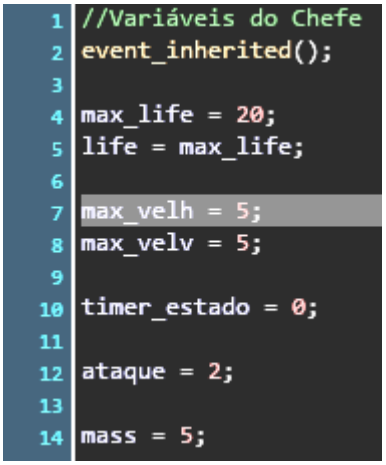
Na Figura 83 mostra o obj_boss002 que é o chefe principal do jogo, ele herdou os eventos do obj_entidade_inimigo, e possui alguns eventos próprios que o diferencia dos inimigos padrões do jogo.

Figura 83: Criação das Entidades Chefes

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

A Figura 84 apresenta os valores das variáveis do personagem. Suas variáveis se assemelham com a dos inimigos, tendo a diferença de seus atributos serem maiores como vida, velocidade e dano, foi também adicionando uma massa no chefe representada pela variável *mass* na linha 14.

Figura 84: Variáveis do Chefe



Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

A Figura 85 apresenta as configurações gerais dos chefes inimigos. Os chefes possuem algumas diferenças no comportamento de estado que se encontram durante o jogo, enquanto o chefe está no estado parado, ele ficará esperando o jogador aparecer na tela, após o jogador entrar na tela do chefe ele começará a perseguir o jogador e após atingir uma certa distância do jogador, ele iniciará um ataque.

Figura 85: Estados Especiais dos Chefes

```
        //Condições de troca de estado
        //Checando se o jogador está na sala
        if (instance_exists(obj_character))
        {
            var _dist = point_distance(x, y, obj_character.x, obj_character.y);
            //Se o jogador está perto o chefe o persegue
            if (_dist < 300)
            {
                estado = "movendo";
            }
        }


        break;
    }

    case "movendo":
    {
        //Criar a lógica do estado movendo
        //Nesse estado inicia a luta de chefe
        if (sprite_index != spr_boss002_walk)
        {
            sprite_index = spr_boss002_walk;
            image_index = 0;
        }
        //Perseguindo o jogador
        if (instance_exists(obj_character))
        {
            //Distância do chefe para o jogador
            var _dist = point_distance(x, y, obj_character.x, obj_character.y);
            var _dir = point_direction(x, y, obj_character.x, obj_character.y);

            if (_dist > 50)
            {
                //Definindo a velocidade
                velh = lengthdir_x(max_velh, _dir);
            }
            else
            {
                //Após chegar ao jogador inicia o ataque
                velh = 0;
                estado = "ataque";
            }

        }

        break;
```

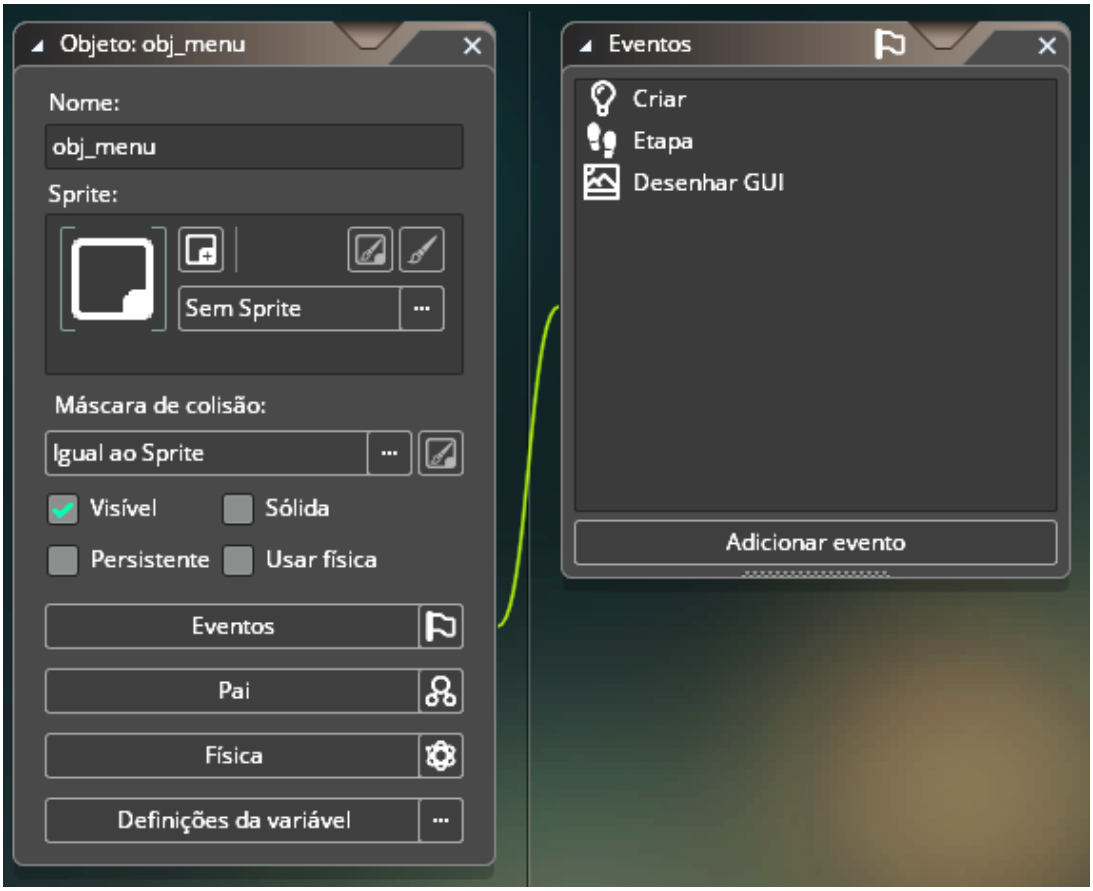
Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

5.11 Teste 11: Criando *Menu* do Jogo

No teste realizado foi criado o *Menu* do jogo. E suas configurações estão ilustradas na Figura 86.

Foi criado o objeto de nome obj_*Menu* que é responsável por ativar o *Menu* da sala.

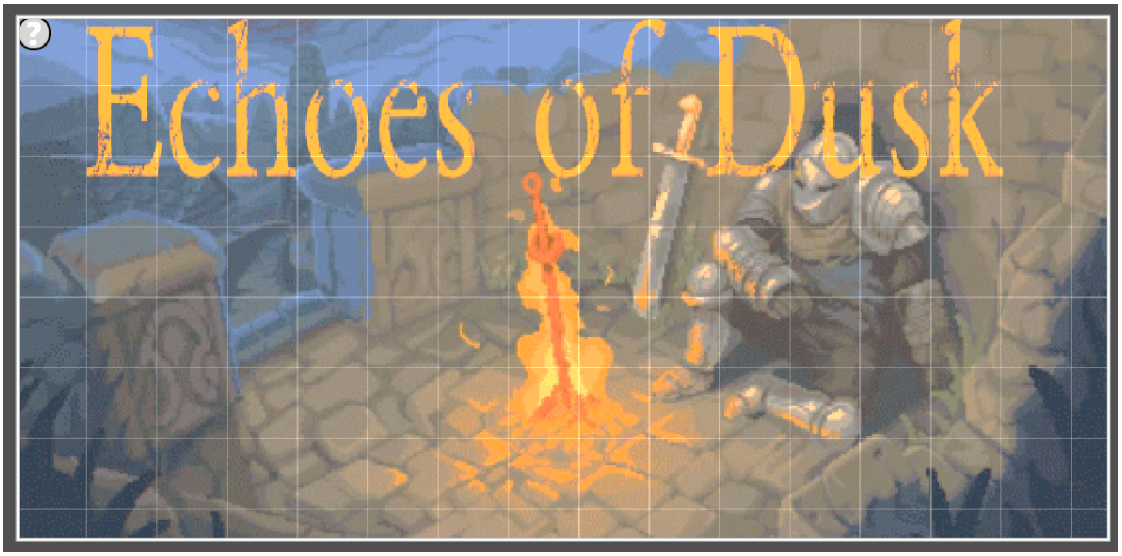
Figura 86: Criando Obj_*Menu*



Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

Para iniciar o *Menu* so jogo foi criado uma sala onde inicia o jogo, foi adicionado ao fundo da sala um GIF e o título do jogo *Echoes of Dusk*, e dentro da sala foi colocado o obj_*Menu* para iniciar as opções do *Menu* do jogo. Essa tela pode ser visualizada na Figura 87.

Figura 87: Sala do *Menu*



Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

Na Figura 88 ilustra os valores empregados na configuração da tela e a posição do *Menu*. O *Menu* possui as seguintes variáveis, sel = 0 sendo o valor de seleção das opções do *Menu*, marg_val = 0, sendo o valor inicial da margem, e marg_total = 32, o valor total da margem que é 32.

Figura 88: Variáveis do Objeto *Menu*

```
1  //Criando Menu
2
3  //Seleção de Menu
4  sel = 0;
5  marg_val = 0;
6  marg_total = 32;
7
```

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

As Figuras 89 e 90 indicam os valores das variáveis de início e fim do jogo. No *Menu* foram adicionadas duas opções para o jogador escolher, “Iniciar”, que inicia o jogo, e “Sair”, que fecha o jogo. Também foi aplicado onde cada opção do jogo leva o jogador, sendo destino de “Iniciar” inicia_jogo, e o destino de “Sair” fechar_jogo.

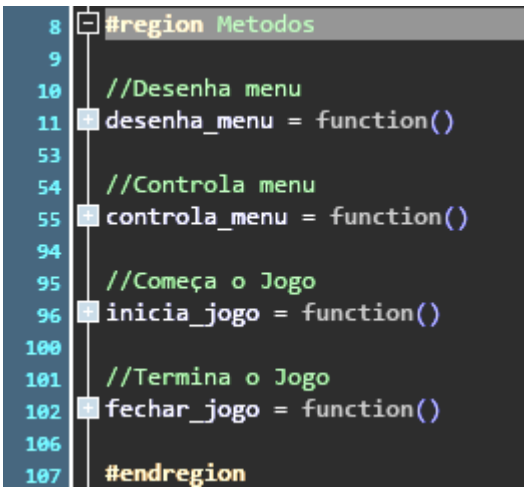
Figura 89: Opções e Destinos do *Menu*

```
111  menu =  [
112              ["Iniciar", menu_acoes.roda_metodo, inicia_jogo],
113              ["Sair", menu_acoes.roda_metodo, fechar_jogo]
114          ];
```

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

No obj_*Menu*, foi criada uma região na qual estão organizadas as ações que ocorrem no *Menu*. Dentro dessa região, encontra-se as funções: Desenha *Menu*, Controle *Menu*, Começa o Jogo e Termina o Jogo.

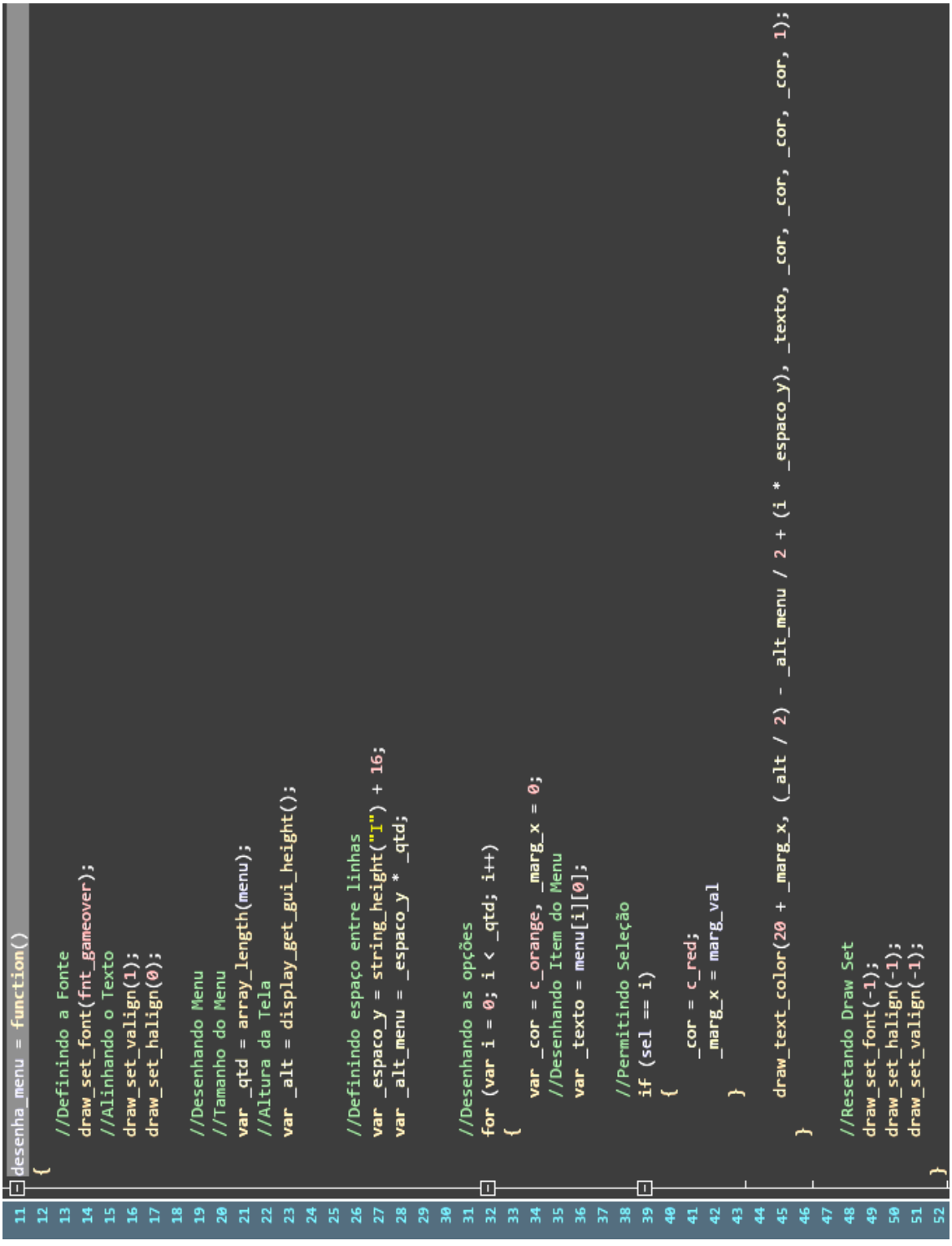
Figura 90: Iniciando Metodos do *Menu*



Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

Nas Figuras 91, 92, 93 e 94, são apresentados os *scripts* dos métodos de ativação das funções *Menu*. O primeiro método é responsável por inserir as opções do *Menu*, foi usada a mesma fonte utilizada na tela de *game over* apresentada na Figura 62 no tópico 5.6 deste trabalho, foram configuradas o tamanho, altura e espaçamento entre linhas das opções, neste método também foi feita a seleção de opções que o jogador pode alternar, e por último aplicando as cores, laranja como cor padrão das opções e vermelha para sinalizar qual opção o jogador está selecionando.

Figura 91: Metodo Desenha *Menu*

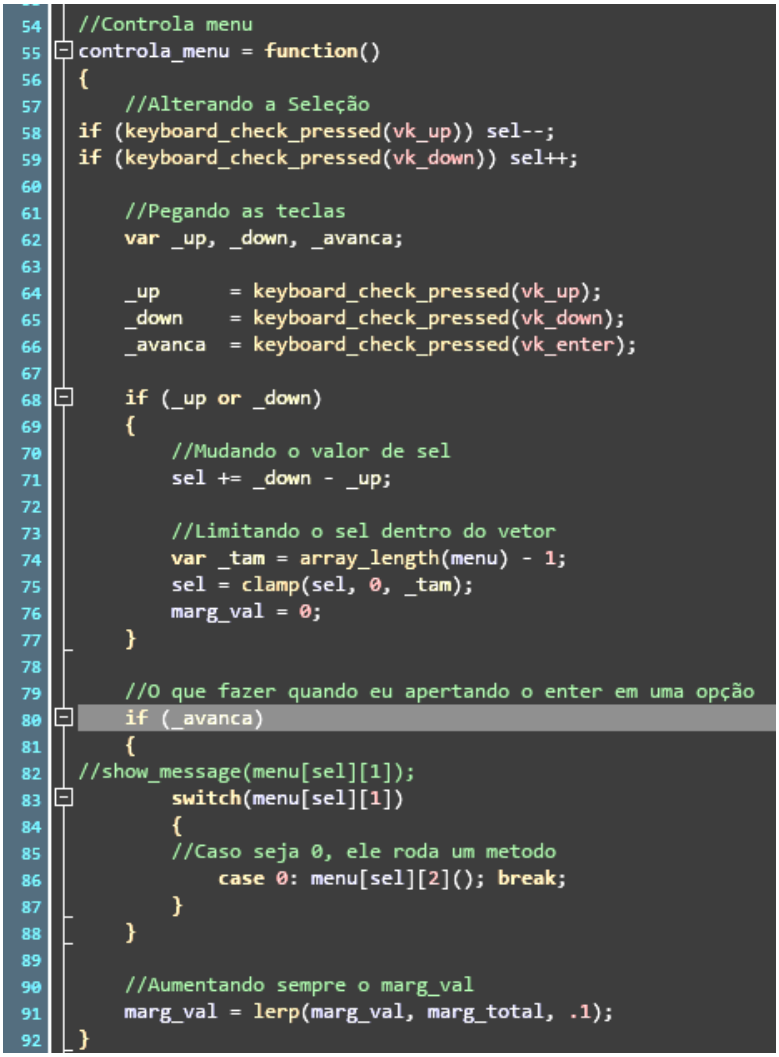


```
desenha_menu = function()
{
    //Definindo a Fonte
    draw_set_font(fnt_gameover);
    //Alinhando o Texto
    draw_set_valign(1);
    draw_set_halign(0);

    //Desenhando Menu
    //Tamanho do Menu
    var _qtd = array_length(menu);
    //Altura da Tela
    var _alt = display_get_gui_height();

    //Definindo espaço entre linhas
    var _espaco_y = string_height("I") + 16;
    var _alt_menu = _espaco_y * _qtd;

    //Desenhando as opções
    for (var i = 0; i < _qtd; i++)
    {
        var _cor = c_orange, _marg_x = 0;
        //Desenhando Item do Menu
        var _texto = menu[i][0];

        //Permitindo Seleção
        if (sel == i)
        {
            _cor = c_red;
            _marg_x = marg_val
        }

        draw_text_color(20 + _marg_x, (_alt / 2) - _alt_menu / 2 + (i * _espaco_y), _texto, _cor, _cor, _cor, _cor, 1);
    }

    //Resetando Draw Set
    draw_set_font(-1);
    draw_set_halign(-1);
    draw_set_valign(-1);
}
```

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

O segundo método, permite que o jogador controle qual opção ele pode selecionar no *Menu*. Para isso aplica o comando das teclas e as ações ao pressionar, sendo elas "*_up*" para ir para cima, "*_down*" para baixo, e a opção "_avanca" que confirma a opção selecionada pelo jogador.

Figura 92: Metodo Controla *Menu*

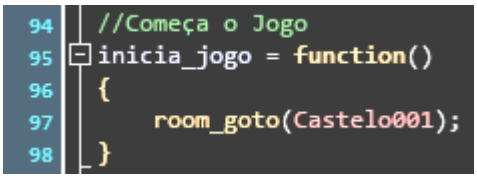


```
//Controla menu
controla_menu = function()
{
    //Alterando a Seleção
if (keyboard_check_pressed(vk_up)) sel--;
if (keyboard_check_pressed(vk_down)) sel++;

    //Pegando as teclas
    var _up, _down, _avanca;

    _up      = keyboard_check_pressed(vk_up);
    _down    = keyboard_check_pressed(vk_down);
    _avanca  = keyboard_check_pressed(vk_enter);

    if (_up or _down)
    {
        //Mudando o valor de sel
        sel += _down - _up;

        //Limitando o sel dentro do vetor
        var _tam = array_length(menu) - 1;
        sel = clamp(sel, 0, _tam);
        marg_val = 0;
    }

    //O que fazer quando eu apertando o enter em uma opção
    if (_avanca)
    {
//show_message(menu[sel][1]);
        switch(menu[sel][1])
        {
        //Caso seja 0, ele roda um metodo
            case 0: menu[sel][2](); break;
        }
    }

    //Aumentando sempre o marg_val
    marg_val = lerp(marg_val, marg_total, .1);
}
```

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

O terceiro método como apresentado na Figura 93, fará com que após o jogador selecionar a opção Iniciar, irá começar o jogo, transportando o jogador para a primeira sala do jogo.

Figura 93: Metodo Inicia Jogo



```
//Começa o Jogo
inicia_jogo = function()
{
    room_goto(Castelo001);
}
```

Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

O último método presente na região dentro do *Menu* foi o Termina o Jogo, onde o jogador após selecionar a opção Sair o jogo irá fechar.

Figura 94: Metodo Fechar Jogo

```
100  //Termina o Jogo
101  fechar_jogo = function()
102  {
103      game_end();
104  }
```

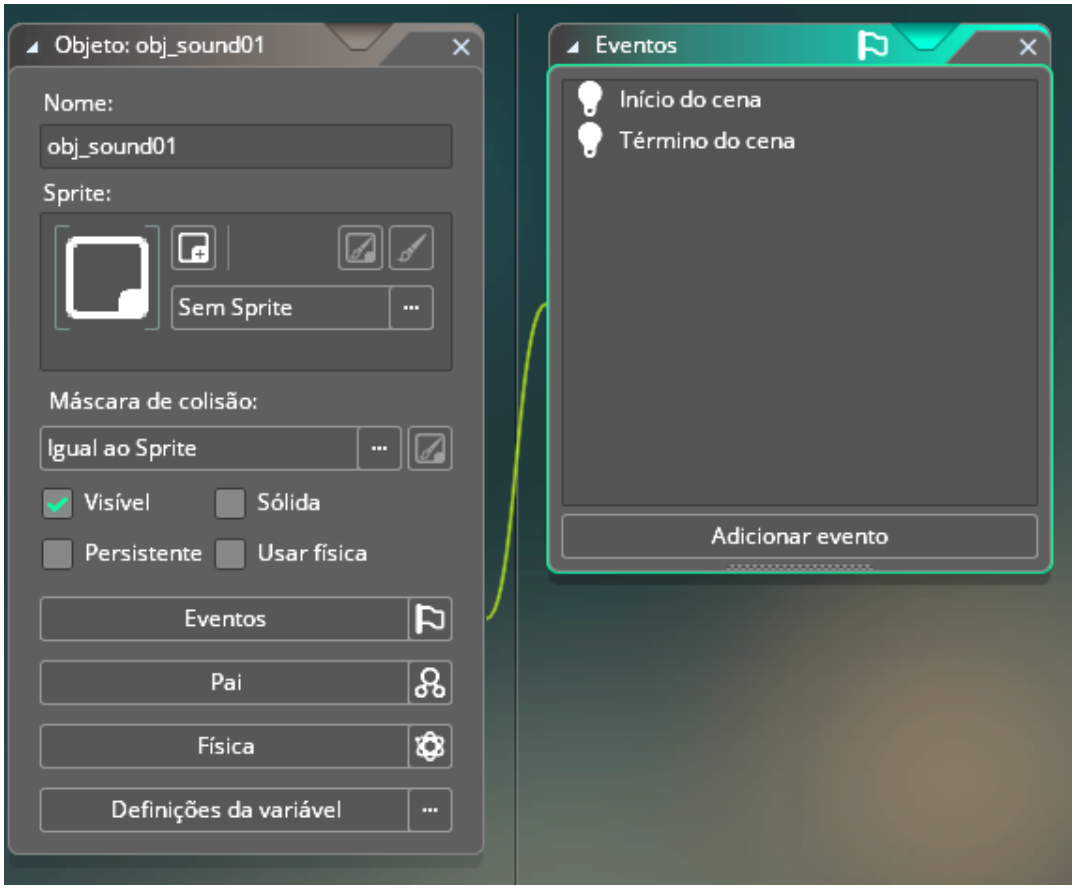
Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

5.12 Teste 12: Aplicando Trilha Sonora

No teste a seguir foi implementada a trilha sonora no jogo. A Figura 95 apresenta a sua tela de configuração.

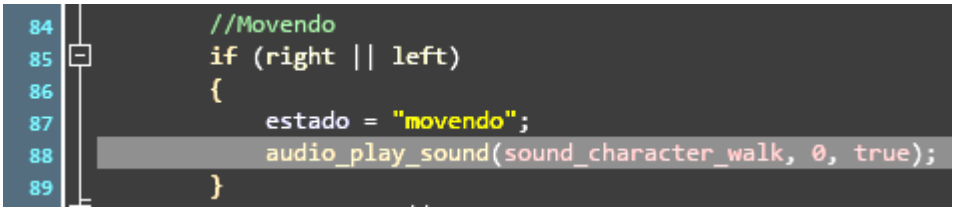
Para cada sala no jogo foi criado um objeto de som contendo uma trilha sonora diferente em cada seguimento, cada objeto está posicionado nas salas para quando o jogador entrar na sala a trilha sonora começar a tocar.

Figura 95: Criando Objeto de Som das Salas



Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

Figura 96: Iniciando Som após Ação



Fonte: Capturado pelo autor do trabalho a partir de *Game*Maker 2 (2025)

A Figura 96 ilustra uma amostra de um *script*, para ativar o som do personagem após realizar a ação “movendo”. Além dos objetos criados, cada entidade do jogo possui seus próprios efeitos sonoros que após realizarem certas ações ou após entrarem em certos estados o som irá ativar.

6 CONSIDERAÇÕES FINAIS

O objetivo desse trabalho foi conseguir adaptar o jogo *Dark Souls* de gênero *Soulslike* para o gênero *Metroidvania* em um protótipo de jogo digital.

O jogo foi feito com inspiração no jogo do gênero RPG e de estética *Dark Fantasy* de nome *Dark Souls*, resgatando elementos presentes no jogo original e adaptando outros para se adequar a estética de troca de gênero de *Soulslike* para *Metroidvania*, com o objetivo de aplicar técnicas de UX (Experiência de Usuário).

Foi analisado para a conclusão deste trabalho análise de elementos essenciais do *game design* para obter uma UX positiva com foco nas mecânicas de jogo, *interface*, narrativa e ambientação.

Foi analisado o jogo *Dark Souls* para identificar práticas de *game desing* que favoreceram a experiência do usuário, permitindo concluir que o *desing* contribui para a criação de uma experiência intuitiva acessível e envolvente. Neste contexto foi desenvolvido um protótipo de jogo.

A metodologia adotada neste trabalho foi qualitativa e descritiva, com o objetivo de analisar e compreender a relação entre o *game design* e a experiência do usuário em jogos digitais. Essa metodologia abordou como a atenção do usuário pode ser chamada por elementos que geram uma curiosidade, fantasias medievais chamam bastante atenção por suas mitologias e criação de universos fantasiosos, e aplicando a estética *Dark Fantasy* gera ainda mais uma atenção do usuário, uma curiosidade pode despertar no jogador querendo saber como esse o universo sombrio e gótico se tornou assim, desbravar a história contada pelo jogo e ficando mais imerso naquele universo.

O trabalho foi dividido em capítulos. No capítulo 2 foi abordado as definições da teoria de experiência de usuário e os seus fatores. No capítulo 3 foram descritas as características específicas para se obter uma experiência de usuário em jogos digitais.

No capítulo 4 foram apresentados os procedimentos metodológicos e as ferramentas necessárias para elaboração deste projeto, tais como: *hardware* utilizado e a *engine Game Maker*, ferramenta bastante empregada por criadores de jogos digitais. Esta *engine* tem vantagem de ser uma ferramenta gratuita tem bastante flexibilidade na elaboração de *game desing*.

No capítulo 5 foram executados 12 testes e apesentados os seus resultados para melhor descrever o funcionamento e o próprio desenvolvimento deste trabalho. Foram encontradas algumas dificuldades, a destacar:

O objeto dano não se comportava de maneira correta, resultando na falha de absorção de dano pelos inimigos e assim tornando-os invencíveis;

As Entidades permaneciam presas ao solo em função de algumas posições dos *sprites* dos personagens;

O campo de visão dos inimigos não detectava o jogador;

Após pesquisas em plataformas digitais as soluções encontradas para estes problemas foram:

Criar um objeto dano diferente em outro projeto de jogo para identificar o problema, e assim, foi descoberta uma falha na leitura das variáveis das entidades. A solução encontrada foi criar duas entidades pai, uma para o jogador e outra para os inimigos;

Para impedir que as entidades permanecessem presas ao solo, foi necessário mudar as posições dos eixos x e y em cada *sprite* das entidades;

Para possibilitar que os inimigos detectassem o jogador em seu campo de visão foi preciso reduzir a altura do campo de visão;

As informações bibliográficas sobre a *engine Game Maker* na internet ainda são bastante escassas. Nos aplicativos de Inteligência Artificial e grupos de discussão há pouca informação disponível. Os canais do *Youtube* são os que apresentam maior diversidade de informações.

6.1 Sugestões de trabalhos futuros

Uma contribuição que este trabalho pode proporcionar para a comunidade de jogos digitais é a apresentação de uma ideia de mudança de gêneros de jogos, porém mantendo sua essência original.

A sugestão para trabalhos futuros é o desenvolvimento de uma versão completa do jogo, ou a criação de novos ambientes, adaptando outras partes do jogo *Dark Souls* e com algumas mudanças de *level desing*, aplicando ainda mais a mudança do gênero *Soulslike* para *Metroidvania*.

REFERÊNCIAS

ADIFA, Marcelo. Comunicação, Marketing e Estratégias de Texto: *Storytelling* e a Jornada do Herói. 1. ed. [*S. l.*: *s. n.*], 2019. 34 p. v. 1. ISBN 3186833106.

ALURA - UX e Usabilidade aplicados em Mobile e Web. 1. ed. [*S. l.*]: Alura, s.d. 159 p. v. 1.

ANTI. *Dark Fantasy Tiktok Song (slowed + reverb) Yamaha - Dorian concept*. YouTube, 2023. Disponível em: https://www.youtube.com/watch?v=MAbSVYjobOc. Acesso em: 11 mar. 2025.

BAGGINS, Nicholas. The Legacy of Dark Souls: *Design*, Difficulty, and Community. Journal of Video *Game* Culture, v. 12, n. 3, p. 25-40, 2016

BORGES, Bosco. *et al*. Experiência do Usuário em Jogos Digitais: Uma Catalogação de Instrumentos de Avaliação. Introdução ao Desenvolvimento de *Game*s, Universidade Federal do Ceará Fortaleza, Ceará, Brasil, v. 1, n. 1, ed. 1, p. 1-10, 24 nov. 2024.

CRAFTPIX.NET. [S. l.], 2016. Disponível em: https://craftpix.net. Acesso em: 10 fev. 2025.

DARK Souls Bonfire GIF. Tenor: MrEpicWonder, 2020. Disponível em: https://tenor.com/pt-BR/view/dark-souls-bonfire-rest-warm-gif-16368928?utm_source=share-button&utm_medium=Social&utm_content=pinterest. Acesso em: 31 maio 2025.

DREAMSCAPE. *øneheart x reidenshi – snowfall*. YouTube, 2022. Disponível em: https://www.youtube.com/watch?v=LlN8MPS7KQs. Acesso em: 11 maio 2025.

EMBOAVA, Valdecir. O que é um jogo *Soulslike*?. Confira tudo sobre o gênero!, *Site* oficial Tecmundo, v. 1, ed. 1, 5 jul. 2024. Disponível em: https://www.tecmundo.com.br/voxel/286695-jogo-*Soulslike*-confira-tudo-o-genero.htm. Acesso em: 14 nov. 2024.

FREE *Pixel* Art Forest 2D Nature Unity Asset Store. Anloveov.click, 2025. Disponível em: https://anloveov.click/product_details/111083073.html. Acesso em: 31 maio 2025.

FREESOUND. [S. l.], 2005. Disponível em: https://freesound.org. Acesso em: 11 mar. 2025.

GAME MAKER. *Como tocar música e efeitos sonoros no GameMaker*. YouTube, 2022. Disponível em: https://www.youtube.com/watch?v=NGNJWpiVS1M. Acesso em: 31 maio 2025.

*GAME*Maker 2: Ferramenta de Criação do Jogo. [S. l.], n.d.a. Disponível em: https://twinery.org. Acesso em: 4 fev. 2025.

GOLDENBOY , Felipe. O que é *Metroidvania*?. O que é *Metroidvania*?, [*S. l.*], v. 1, n. 1, p. 1-5, 26 maio 2022. Disponível em: https://canaltech.com.br/*game*s/o-que-e-*Metroidvania*-217200/. Acesso em: 13 nov. 2024.

IGLESIAS, Juan J. Vargas. Making Sense of Genre: The Logic of Video *Game* Genre Organization. Making Sense of Genre: The Logic of Video *Game* Genre Organization, Sage Journals, ano 2018, v. 15, n. 2, p. 1-21, 7 fev. 2018. DOI 10.1177/1555412017751803. Disponível em: https://us.sagepub.com/en-us/nam/journals-permissions. Acesso em: 13 nov. 2024.

ITCH.IO. [S. l.], 2013. Disponível em: https://rvros.itch.io. Acesso em: 10 fev. 2025.

K PLAYZ. *Iron Golem - Dark Souls Soundtrack 10*. YouTube, 2020. Disponível em: https://www.youtube.com/watch?v=b1cBxN1YNXM. Acesso em: 11 mar. 2025.

LOSADA, João Paulo. ANÁLISE: Dark Souls Prepare To Die. ANÁLISE: Dark Souls Prepare To Die, *Site* oficial Adrenaline, 10/09/2012. Disponível em: https://www.adrenaline.com.br/analise/outros/analise-dark-souls-prepare-to-die/. Acesso em: 15 nov. 2024.

METATRON Omega | Hierosgamos. Youtube: VwrytheҪҪҪ, 2015. Disponível em: https://www.youtube.com/watch?v=LlN8MPS7KQs. Acesso em: 11 mar. 2025.

NATUREWORLD1986. *DARK SEWER (Dark Ambient Music) Creepy Horror Music*. YouTube, 2023. Disponível em: https://www.youtube.com/watch?v=o4-leE6iJXA. Acesso em: 11 mar. 2025.

NIGHTMARE PILLS Font. Fontspace: Herofonts, 2020. Disponível em: https://www.fontspace.com/nightmare-pills-font-f30706. Acesso em: 31 maio 2025.

NONECLASS. *Criando um Metroidvania com o Game Maker Studio 2*. YouTube, 2019. Vídeo. Disponível em: https://www.youtube.com/playlist?list=PLKTRv0drNjJ-D6CuTsb2pC8zMUuxk3XBf. Acesso em: 4 fev. 2025.

OPEN*GAME*ART. [S. l.], 2009. Disponível em: https://open*game*art.org/content/sewer-tileset. Acesso em: 25 fev. 2025.

OTSUKA, Famitsu‘s Kadoman. Dark Souls 1 - *Design* Works Interview. Dark Souls 1 - *Design* Works Interview, [*S. l.*], p. 114-125, 22 jan. 2018. Disponível em: https://www.barnardtranslations.com/post/dark-souls-*design*-works. Acesso em: 20 nov. 2024.

PEREIRA, Rogério. User Experience *Design*: Como criar produtos digitais com foco nas pessoas. 1. ed. [*S. l.*]: Casa do Código, s.d. 194 p. v. 1. ISBN 978-85-94188-66-3, 978-85-94188-67-0, 978-85-94188-68-7.

RABIN, Steve (ed.). Introdução ao Desenvolvimento de *Game*s: Entendendo o Universo dos Jogos. 2. ed. Condomfnio E·Business Park Rua Werner Siemens, 111 - Predlo 20 Espat;o 04 - Lapa de Baixo1) 3665·9900 - Fax: (11) 3665·990 CEP 05069·900-Sao Paulo- SP: Marileide Gomes, s.d. 153 p. v. 1. ISBN 13: 978·85·221-1143·5 , 10: 85·221·1143-X.

RENATITO. Conheça os gêneros de jogos e seus subgêneros: Será que você sabe diferenciar cada um deles?. *Game*Vicio, [*S. l.*], p. 1-50, 31 maio 2021. Disponível em: https://www.*game*vicio.com/noticias/2021/05/conheca-os-generos-de-jogos-e-seus-subgeneros/. Acesso em: 13 nov. 2024.

ROBERT, Christopher. *Abandoned Tower (Dark Ambient Music)*. Youtube: RobChristopherMusic1, 2016. Disponível em: https://www.youtube.com/watch?v=CxMxFEN6Lsw. Acesso em: 11 mar. 2025.

SCHELL, Jesse. The Art of *Game Design*: A Book of Lenses. 1. ed. [*S. l.*]: CRC Press, 2008. 520 p. v. 1. ISBN 9780123694966, 0123694965.

SOFTWARE, From. Dark Souls: *Design* Works. 1. ed. [*S. l.*]: Udon Entertainment, 2014. 125 p. v. 1. ISBN 1926778898,9781926778891.

STATI, Cesar Ricardo; SARMENTO, Camila Freitas. Experiência do Usuário (UX). 1. ed. Rua Clara Vendramin, 58 . Mossunguê . CEP 81200-170 . Curitiba . PR . Brasil: Inter Saberes, 2021. 244 p. v. 1. ISBN 978-65-5517-913-2.

TARI. *PLATAFORMA #3 - Tiros [Game Maker Studio 2]*. YouTube, 2019. Disponível em: https://www.youtube.com/watch?v=fo88XfeZV3w. Acesso em: 29 maio 2025.

TEIXEIRA, Fabrício. Introdução e boas práticas UX *Design*. São Paulo: Casa do Código, [s. d.].

TEKINBAS, Katie Salen; ZIMMERMAN, Eric. Rules of Play: *Game Design* Fundamentals. [*S. l.*]: The MIT Press, 2003. 1354 p. v. 1. ISBN 9780262299930.

TOMITO. *Dark Souls OST – Ornstein and Smough [Extended]*. YouTube, 2022. Disponível em: https://www.youtube.com/watch?v=76EYcQ5NfP0&t=55s. Acesso em: 11 mar. 2025.

YABLONSKI, Jon. Leis da Psicologia Aplicadas em UX: Usando a psicologia para projetar produtos e serviços melhores. Rua Luís Antônio dos Santos 110 02460-000 – São Paulo, SP – Brasil: Novatec Editora Ltda., 2020. 152 p.

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
GABINETE DO REITOR

Av. Universitária, 1069 ● Setor Universitário
Caixa Postal 86 ● CEP 74605-010
Goiânia ● Goiás ● Brasil
Fone: (62) 3946.1000
www.pucgoias.edu.br ● reitoria@pucgoias.edu.br

RESOLUÇÃO n° 038/2020 – CEPE

ANEXO I

APÊNDICE ao TCC

Termo de autorização de publicação de produção acadêmica

O(A) estudante LUÍS FERNANDO FREITAS PAGOTTI do Curso de ENGENHARIA DE COMPUTAÇÃO, matrícula 2020.1.0033.0071-4, telefone: (62) 98405-8333 e-mail lffpagotti@gmail.com, na qualidade de titular dos direitos autorais, em consonância com a Lei nº 9.610/98 (Lei dos Direitos do autor), autoriza a Pontifícia Universidade Católica de Goiás (PUC Goiás) a disponibilizar o Trabalho de Conclusão de Curso intitulado DESENVOLVIMENTO DE UM JOGO DIGITAL: ADAPTAÇÃO DO JOGO DARK SOULS PARA O GÊNERO METROIDVANIA, gratuitamente, sem ressarcimento dos direitos autorais, por 5 (cinco) anos, conforme permissões do documento, em meio eletrônico, na rede mundial de computadores, no formato especificado (Texto (PDF); Imagem (GIF ou JPEG); Som (WAVE, MPEG, AIFF, SND); Vídeo (MPEG, MWV, AVI, QT); outros, específicos da área; para fins de leitura e/ou impressão pela internet, a título de divulgação da produção científica gerada nos cursos de graduação da PUC Goiás.

Goiânia, 18 de junho de 2025.

Assinatura do(s) autor(es): Luís Fernando Freitas Pagotti

Nome completo do autor: LUÍS FERNANDO FREITAS PAGOTTI

gov.br Documento assinado digitalmente
ANGELICA DA SILVA NUNES
Data: 18/06/2025 11:20:13-0300
Verifique em https://validar.iti.gov.br

Assinatura do professor-orientador: _____

Nome completo do professor-orientador: ANGÉLICA DA SILVA NUNES