

RELATÓRIO DE PESQUISA FINAL

**CENTRO UNIVERSITÁRIO DA FUNDAÇÃO DE ENSINO
OCTÁVIO BASTOS - UNIFEOB**

João Augusto de Freitas - 25000019

Telefone: (19) 99494-3031

E-mail: joao.a.freitas@sou.unifeob.edu.br

Número do edital - 2025.2

Entrega do relatório parcial 16/03/2026

Entrega do relatório final 14/09/2026



Documento assinado digitalmente
JOAO AUGUSTO DE FREITAS
Data: 14/09/2026 22:01:07-0300
Verifique em <https://validar.iti.gov.br>

**INTELIGÊNCIA ARTIFICIAL E ENGENHARIA DE
PROMPT NO DESENVOLVIMENTO DE JOGOS
DIGITAIS**

Orientador Marcelo Ciacco Almeida

Curso Análise e Desenvolvimento de Sistemas



Documento assinado digitalmente
MARCELO CIACCO DE ALMEIDA
Data: 14/09/2026 21:56:28-0300
Verifique em <https://validar.iti.gov.br>

Assinatura do orientador

São João da Boa Vista/SP

2026

TÍTULO

INTELIGÊNCIA ARTIFICIAL E ENGENHARIA DE PROMPT NO DESENVOLVIMENTO
DE JOGOS DIGITAIS

AUTORES - João Augusto de Freitas e Marcelo Ciacco Almeida

RESUMO

A Inteligência Artificial (IA) generativa vem sendo incorporada ao desenvolvimento de jogos digitais, especialmente na criação visual, geração de código e apoio ao processo produtivo. Este trabalho investigou, por meio do desenvolvimento integral de um protótipo de jogo 2D, os limites e as contribuições da IA generativa e da engenharia de prompt, buscando identificar em quais etapas essas tecnologias podem substituir, assistir ou não atender adequadamente ao trabalho humano. A pesquisa caracteriza-se como aplicada, de abordagem predominantemente qualitativa e delineada como estudo de caso único com desenvolvimento iterativo. O protótipo Legacy Code: Override foi desenvolvido no GameMaker, com registros por versionamento de código, documentação audiovisual e experimentações sistemáticas com modelos generativos. Os resultados indicaram maior eficácia da IA em atividades com critérios predominantemente perceptuais, como concepção visual, geração de fundos e produção assistida de ativos, além de apoio em consultas e depuração localizada de código. Em contrapartida, foram observadas limitações em tarefas que exigem restrições funcionais, espaciais e sistêmicas verificáveis, especialmente na geração autônoma de level design navegável. Como resposta, adotou-se uma composição em camadas que separa os elementos visuais gerados por IA da geometria funcional construída manualmente. Conclui-se que a IA generativa pode acelerar e assistir o desenvolvimento de jogos, mas sua eficácia depende da natureza da tarefa, permanecendo a intervenção humana essencial nas decisões funcionais, arquiteturas e sistêmicas.

Palavras-chave: Inteligência Artificial generativa; Engenharia de prompt; Desenvolvimento de jogos digitais; Level design; GameMaker.

I. INTRODUÇÃO E JUSTIFICATIVA

A Inteligência Artificial generativa deixou de ser promessa tecnológica e passou a integrar o fluxo produtivo cotidiano do desenvolvimento de software. Na indústria de jogos digitais, a expectativa depositada nessas ferramentas é particularmente alta, porque a produção de um jogo concentra, em um mesmo artefato, três naturezas distintas de trabalho: a criação artística, a construção narrativa e a engenharia de sistemas interativos. O discurso corrente de mercado tende a tratar essas três frentes como igualmente automatizáveis, projetando sobre a geração de conteúdo de jogo o mesmo desempenho observado na geração de imagens ilustrativas ou de trechos isolados de código.

A literatura especializada é consideravelmente mais cautelosa. Os papéis viáveis dos Modelos de Linguagem de Grande Escala (Large Language Models, LLM) em jogos são mapeados na literatura, que identifica como desafios abertos a alucinação, a controlabilidade das saídas e a ausência de métricas consolidadas de avaliação (GALLOTTA et al., 2024). Na revisão da geração procedural de conteúdo, a integração de modelos generativos à criação de fases permanece emergente, tendo como obstáculo central a garantia de jogabilidade: gerar um cenário visualmente convincente é problema distinto de gerar um cenário efetivamente navegável (FARROKHI MALEKI; ZHAO, 2024).

Há, entretanto, uma lacuna entre o diagnóstico da literatura e a experiência concreta de quem desenvolve. Faltam registros técnicos de primeira mão que documentem, com rastreabilidade, onde exatamente a assistência automatizada acelera o trabalho e onde ela falha de forma irreversível. É nessa lacuna que este projeto se justifica. Mais do que uma revisão bibliográfica, a pesquisa produziu um artefato de software público, versionado e auditável, cujo histórico de desenvolvimento pode ser inspecionado commit a commit, acompanhado de registro audiovisual contínuo dos testes mecânicos. O código-fonte está disponível em github.com/joaoaugusto-dev/legacy-code-override e a documentação audiovisual em playlist aberta no YouTube.

O objeto escolhido reforça a pertinência do recorte. Legacy Code: Override é um jogo de plataforma 2D em pixel art cuja temática é a própria evolução de uma unidade de inteligência artificial: o jogador controla B.I.T. (Basic Intelligent Unit), entidade que percorre um sistema computacional corrompido, elimina bugs, dispara dígitos binários e utiliza a metáfora do override de código para superar obstáculos. O projeto é, simultaneamente, o instrumento e o objeto da investigação, o que permite confrontar diretamente duas acepções de inteligência

artificial que o senso comum costuma confundir: a IA que auxiliou a construir o jogo e a IA que governa o comportamento das entidades dentro dele.

Este relatório final consolida os doze meses do programa, compreendidos entre setembro de 2025 e setembro de 2026, incorporando e ampliando o conteúdo apresentado no relatório parcial de março de 2026. São aqui reunidas a fundamentação teórica da primeira etapa, a experimentação com engenharia de prompt, o desenvolvimento completo dos sistemas do protótipo e a formalização do pipeline de produção resultante.

II. OBJETIVOS

a) Objetivo geral

Investigar, por meio do desenvolvimento integral de um protótipo jogável, os limites e as contribuições efetivas da Inteligência Artificial generativa e da engenharia de prompt no processo de produção de jogos digitais 2D, delimitando tecnicamente as etapas em que essas ferramentas substituem, assistem ou não atendem ao trabalho humano.

b) Objetivos específicos

- Desenvolver e versionar publicamente o protótipo Legacy Code: Override, implementando os sistemas mecânicos essenciais de um jogo de plataforma (controlador de personagem, física, resolução de colisão, combate, progressão entre fases e inteligência artificial de comportamento dos inimigos).
- Avaliar empiricamente, sob a técnica de zero-shot prompting, a capacidade de modelos generativos de imagem de produzir cenários de plataforma que respeitem simultaneamente a coerência estética pretendida e a rigidez topológica exigida pela malha de colisão.
- Formalizar e validar um pipeline de produção reprodutível que distribua as responsabilidades entre a geração automatizada e a intervenção humana, registrando todo o processo em documentação audiovisual contínua.

III. REVISÃO DA LITERATURA

3.1 Fundamentos: aprendizado profundo e a natureza probabilística dos modelos generativos

As definições de Inteligência Artificial são agrupadas em quatro categorias (sistemas que pensam como humanos, que agem como humanos, que pensam racionalmente e que agem racionalmente), evidenciando que não há convergência plena quanto ao próprio conceito (RUSSELL; NORVIG, 2020). Operacionalmente, a IA contemporânea apoia-se em aprendizado de máquina e aprendizado profundo, aplicados sobre grandes volumes de dados (IBM, 2024; Google Cloud, 2025). O mecanismo que sustenta essa família de técnicas foi consolidado por Rumelhart, Hinton e Williams (1986), ao demonstrarem a retropropagação de erros como método de ajuste de representações internas em redes neurais.

A consequência epistemológica desse arranjo é decisiva para esta pesquisa: um modelo generativo otimiza a verossimilhança estatística de suas saídas em relação à distribuição dos dados de treino. Ele não possui intencionalidade, não representa explicitamente restrições e não verifica a validade estrutural do que produz. A plausibilidade visual é o critério interno do modelo; a funcionalidade não é.

3.2 Engenharia de prompt e a técnica de zero-shot prompting

Em levantamento sistemático das técnicas de engenharia de prompt, o campo é organizado por domínio de aplicação, destacando-se que o prompt opera como instrução em linguagem natural ou como representação vetorial aprendida, permitindo adaptar o modelo a tarefas específicas sem alterar seus parâmetros (SAHOO et al., 2024). Entre as técnicas catalogadas, o zero-shot prompting consiste em submeter uma instrução descritiva ao modelo sem fornecer exemplos prévios de entrada e saída, dependendo exclusivamente do conhecimento internalizado durante o pré-treinamento.

Essa caracterização impõe um limite importante: o prompt é uma interface de exploração do espaço latente já aprendido pelo modelo, e não um mecanismo de imposição de restrições rígidas. Uma competência ausente da distribuição de treino não é recuperável por reformulação da instrução (observação que, como se verá, é central para interpretar os resultados obtidos neste projeto).

3.3 IA generativa aplicada a jogos digitais e geração procedural de conteúdo

Um levantamento abrangente dos papéis que LLM podem assumir em jogos distingue funções de jogador, de personagem não jogável, de assistente de produção e de projetista de conteúdo. Os usos consolidados concentram-se na assistência à produção, enquanto a atuação autônoma em projeto de conteúdo permanece limitada por problemas de controlabilidade e de avaliação (GALLOTTA et al., 2024).

Complementando esse quadro, a revisão da geração procedural de conteúdo (PCG) mostra que as abordagens clássicas (construtivas, baseadas em busca e baseadas em restrições) incorporam explicitamente o critério de jogabilidade ao processo de geração, seja por regras de construção, seja por função de avaliação, seja por satisfação de restrições formais. Os modelos generativos por difusão, ao contrário, não dispõem de qualquer mecanismo equivalente: otimizam semelhança perceptual com o corpus de treino, e não navegabilidade do espaço gerado. A integração entre as duas famílias de técnicas é apontada como direção promissora e ainda em aberto (FARROKHI MALEKI; ZHAO, 2024).

3.4 Limitações espaciais dos modelos texto-imagem

A explicação técnica mais direta para as dificuldades observadas na prática deste projeto está na consistência espacial de modelos texto-imagem: relações espaciais estão sub-representadas nos conjuntos de pares imagem-texto utilizados no treinamento desses sistemas. Em consequência, os modelos aprendem com muito mais competência o que deve aparecer em uma cena do que onde e em que relação os elementos devem aparecer. Para mitigar o problema, foi construído um conjunto de dados massivamente reanotado com descrições espaciais explícitas, obtendo-se ganhos significativos de fidelidade — o que confirma, por contraste, que a deficiência é de representação nos dados, e não de formulação da instrução (CHATTERJEE et al., 2025).

3.5 Confiabilidade estrutural do código gerado por LLM

Uma comparação empírica entre código gerado por LLM e código escrito por humanos, sob métricas de manutenibilidade e confiabilidade, mostra que, de modo geral, o código gerado apresenta menor incidência de defeitos superficiais e menor esforço de correção; entretanto, ao enfrentar problemas complexos, os modelos introduzem problemas estruturais ausentes do código humano. A conclusão prática é a necessidade de validação sistemática e auditoria humana antes da incorporação de trechos gerados, particularmente em sistemas com estado compartilhado (MOLISON et al., 2025).

Em direção convergente, o impacto da IA generativa sobre a prática de desenvolvimento de software mostra que essas ferramentas operam como ampliação de capacidade, e não como substituição do desenvolvedor, permanecendo limitadas justamente na compreensão contextual e na tomada de decisões arquiteturais (MALLAMPATI, 2025). Trata-se da mesma fronteira identificada empiricamente nesta pesquisa, deslocada do domínio visual para o domínio da engenharia de sistemas.

3.6 Inteligência artificial de comportamento em jogos

O campo da IA em jogos é sistematizado com a distinção entre a IA empregada na produção de conteúdo e a IA responsável pelo comportamento de agentes em tempo de execução. Registra-se que máquinas de estados finitos e árvores de comportamento permanecem dominantes na prática da indústria, apesar da disponibilidade de técnicas de aprendizado mais sofisticadas. As razões são de ordem prática: custo computacional reduzido, previsibilidade de execução, facilidade de depuração e, sobretudo, controle autoral (o comportamento de um inimigo precisa ser legível e justo para o jogador, e legibilidade é propriedade projetada, não emergente) (YANNAKAKIS; TOGELIUS, 2025).

IV. METODOLOGIA

4.1 Natureza e delineamento da pesquisa

Trata-se de pesquisa aplicada, de abordagem predominantemente qualitativa com registro quantitativo dos artefatos produzidos, delineada como estudo de caso único conduzido sob desenvolvimento iterativo. O pesquisador atua simultaneamente como desenvolvedor do artefato e como observador do processo, condição que exige, como contrapartida metodológica, rastreabilidade externa: por essa razão todo o desenvolvimento foi versionado em repositório público e documentado em vídeo, de modo que as afirmações deste relatório possam ser verificadas de forma independente.

4.2 Fases de execução

O programa foi organizado em cinco fases ao longo dos doze meses:

- Fase 1 — Fundamentação teórica (setembro a novembro de 2025): revisão dos conceitos de Inteligência Artificial, aprendizado profundo, redes neurais e modelos de linguagem, consolidada no documento PARTE 1 — Leitura IC, anexado ao repositório.
- Fase 2 — Concepção e definição de escopo (dezembro de 2025): definição da estética pixel art 2D como escolha funcional, associada à natureza binária e retro-futurista do ambiente narrado; concepção do protagonista B.I.T.; primeiras gerações de sprites; configuração da máquina de estados de animação e do sistema de entrada.
- Fase 3 — Núcleo mecânico (janeiro a março de 2026): desenvolvimento das mecânicas de jogo e aplicação da física, com implementação de gravidade, pulo duplo, resolução de colisão, escadas e suporte a gamepad; abertura do repositório público em 6 de fevereiro de 2026; entrega do relatório parcial em 16 de março de 2026.
- Fase 4 — Sistemas de jogo (abril e maio de 2026): implementação do transporte entre salas, dos inimigos com máquina de estados finitos, do sistema de combate, do sistema de vidas e da interface de estado.
- Fase 5 — Consolidação (junho a agosto de 2026): testes de jogabilidade, ajuste de *game feel*, organização da documentação de conceitos e redação deste relatório final.

4.3 Instrumentos de registro

Três instrumentos foram empregados para tornar o processo auditável:

- Versionamento de código. Repositório público no GitHub (disponível em: <https://github.com/joaoaugusto-dev/legacy-code-override>), organizado em três diretórios (src, com o projeto GameMaker completo; docs, com os relatórios e o material de conceito; e versions, com a build empacotada). Cada commit corresponde a um incremento funcional identificável e datado, o que permite reconstruir a cronologia do desenvolvimento.
- Documentação audiovisual. Playlist pública com oito devlogs no YouTube (disponível em: https://www.youtube.com/playlist?list=PLG5uEMVXHVVVe_PjFykboGsXb5euGNZH8), destinada a registrar o comportamento dinâmico do protótipo (deslizamento de sprite, peso de queda, resposta ao dano, fluidez de movimentação) que a leitura estática do código não é capaz de revelar.

- Registro de experimentação com IA. O diretório docs/conceitos preserva as gerações de sprites de primeira e segunda geração, a geração do background, o registro do loop sequencial de fundo e os documentos de solicitação de estilo e de consulta assistida à linguagem GML (GameMaker Language).

4.4 Ferramentas empregadas

O protótipo foi desenvolvido no motor GameMaker, com programação em GML. A geração de imagens utilizou o modelo Nano Banana 2, identificado tecnicamente como Gemini 3.1 Flash Image, lançado pela Google em fevereiro de 2026 (Google, 2026). O pós-processamento dos ativos foi realizado em Canva. Assistentes conversacionais baseados em modelos de linguagem foram utilizados para consulta sintática e depuração de GML. O controle de versão empregou Git e GitHub; a captura e publicação dos devlogs, ferramentas de gravação de tela e o YouTube.

4.5 Protocolo de teste de engenharia de prompt

A avaliação da capacidade dos modelos generativos de produzir cenários de plataforma seguiu protocolo de três rodadas para cada cenário pretendido. Na primeira rodada, aplicou-se prompt-to-image em regime de zero-shot, descrevendo em linguagem natural a atmosfera e a estrutura desejadas. Na segunda, o prompt foi reformulado com restrições topológicas explícitas: plataformas horizontalmente alinhadas, caminho contínuo de ponta a ponta, ausência de vãos superiores ao alcance do pulo e bordas retas. Na terceira, recorreu-se a sketch-to-image, fornecendo ao modelo um esboço manual da planta da fase como condicionamento estrutural.

O critério de aceitação foi definido previamente e de forma objetiva: o cenário gerado seria aproveitado como geometria funcional apenas se suas plataformas pudessem ser convertidas diretamente em máscara de colisão sem redesenho (isto é, mantendo alinhamento consistente, bordas retas e caminho navegável integral). Cenários que exigissem reconstrução manual da geometria seriam classificados como falha, ainda que esteticamente satisfatórios.

V. RESULTADOS

5.1 Artefato de software e rastreabilidade

O principal resultado da pesquisa é o protótipo jogável Legacy Code: Override (<https://github.com/joaoaugusto-dev/legacy-code-override>), publicamente disponível e auditável. O repositório reúne onze commits registrados entre 6 de fevereiro e 10 de maio de 2026 (período em que o desenvolvimento passou a ser versionado; a produção anterior,

correspondente às fases de concepção e configuração inicial, está documentada nos devlogs). O projeto contabiliza oito objetos, 631 linhas de GML distribuídas em dez eventos programados, e onze sprites totalizando vinte quadros de animação.

5.2 Controlador de personagem, física e resolução de colisão

O objeto `obj_bit` concentra a maior parte da lógica do jogo, com 265 linhas apenas no evento de passo. A resolução de colisão foi implementada por aproximação incremental de um pixel: ao detectar sobreposição projetada na velocidade corrente, o objeto avança progressivamente na direção do movimento até o último ponto livre e zera a componente correspondente da velocidade. O método garante encaixe exato contra a geometria independentemente da magnitude da velocidade, eliminando penetração e tremulação, e é aplicado uniformemente contra os três tipos sólidos do jogo (`obj_chao`, `obj_ponta` e `obj_teleporter`).

O sistema de escada exigiu tratamento específico. A subida só é habilitada quando a sobreposição horizontal entre o personagem e a escada atinge ao menos trinta por cento da largura do sprite, o que evita o engate acidental quando o jogador apenas roça a borda do objeto. Durante a subida, os pulos disponíveis são recarregados, permitindo que o jogador salte a partir da escada.

O suporte a gamepad varre os doze primeiros slots de controle, aplica zona morta de 0,25 ao analógico e aceita entrada simultânea de teclado, analógico e direcional. O retorno háptico foi parametrizado de forma contextual, de modo que cada evento do jogo produza uma assinatura tátil distinta, conforme a Tabela 1.

Tabela 1 — Parametrização do retorno háptico por evento de jogo

Evento	Intensidade	Duração	Função de leitura
Pulo simples	0,30	10 quadros	Confirmação discreta do salto
Pulo duplo	0,60	15 quadros	Sinaliza esgotamento dos pulos
Disparo	0,40	8 quadros	Confirmação imediata de tiro
Dano recebido	0,60	15 quadros	Alerta de perda de vida
Morte	1,00	continua até reinício	Encerramento da tentativa
Carregamento do teleporte	progressiva (0 a 0,50)	120 quadros	Indica progresso da transição

Fonte: elaborado pelo autor a partir do código-fonte do protótipo (`obj_bit`, evento Step).

5.3 Inteligência artificial de comportamento dos inimigos

O objeto `obj_bug` implementa, em 140 linhas, uma máquina de estados finitos de quatro estados (Figura 1): IDLE, de espera; WANDER, de patrulha em direção sorteada; CHASE, de perseguição ao jogador; e DYING, de dissolução após ser atingido. A transição entre patrulha e perseguição é governada por dois limiares distintos de distância euclidiana: o engajamento ocorre a 150 pixels e o desengajamento apenas a 250 pixels. Essa diferença constitui histerese deliberada e resolve um defeito recorrente em máquinas de estado de limiar único, no qual o agente oscila rapidamente entre estados quando o jogador permanece exatamente na fronteira do campo de visão.

As velocidades foram diferenciadas por estado (1,5 pixel por quadro em patrulha e 2,5 em perseguição) para que o estado interno do inimigo seja visualmente legível pelo jogador antes do contato. A preservação espacial do agente é assegurada por sondagem à frente da caixa de colisão, oito pixels adiante da borda e quatro pixels abaixo da base: quando o solo termina, o inimigo em patrulha inverte a direção, enquanto o inimigo em perseguição apenas interrompe o avanço, permanecendo na borda. A regra foi necessária porque a mesma rotina de gravidade e colisão do jogador é aplicada ao inimigo, o que, sem a sondagem, o levaria a se lançar em quedas durante a perseguição.

O estado DYING dura trinta quadros, com transição cromática progressiva e tremor crescente, encerrando-se com a emissão de quinze faíscas e um anel de energia. Durante toda a dissolução o inimigo deixa de causar dano, decisão que elimina a colisão injusta imediatamente posterior ao acerto (um problema de justiça percebida que só se manifestou em teste de jogabilidade).

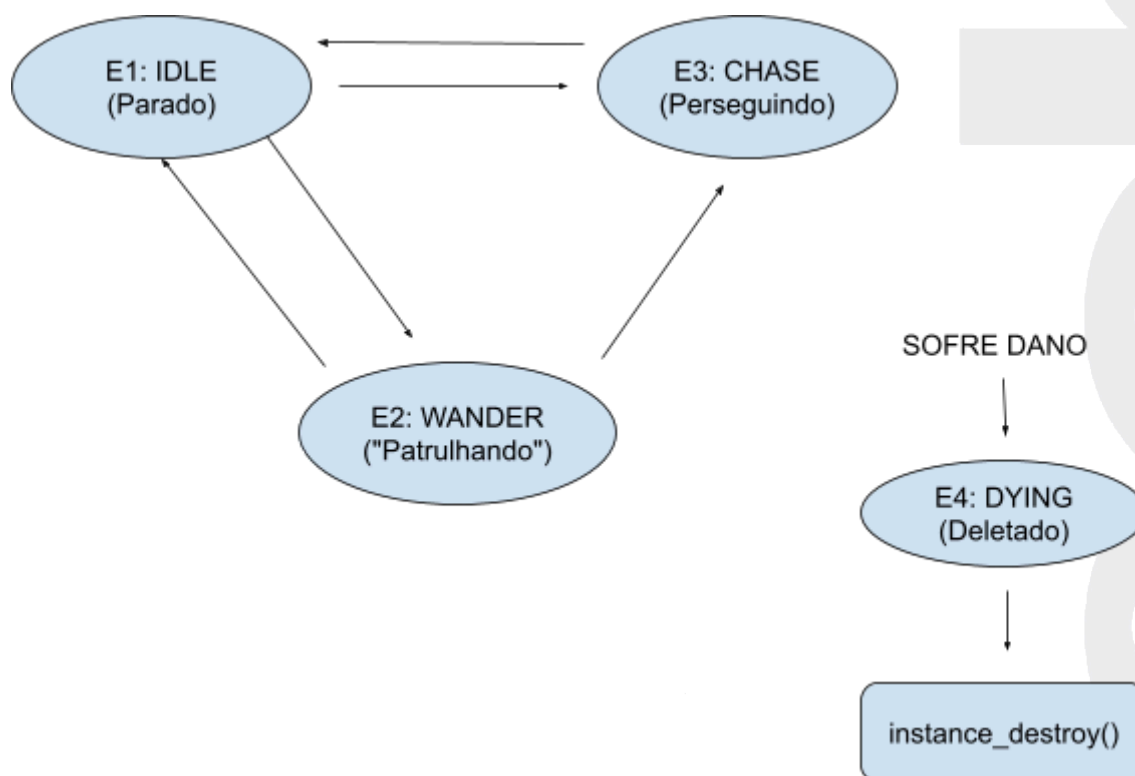


Figura 1 — Máquina de estados finitos do comportamento do inimigo (obj_bug): transições entre patrulha, perseguição e dissolução

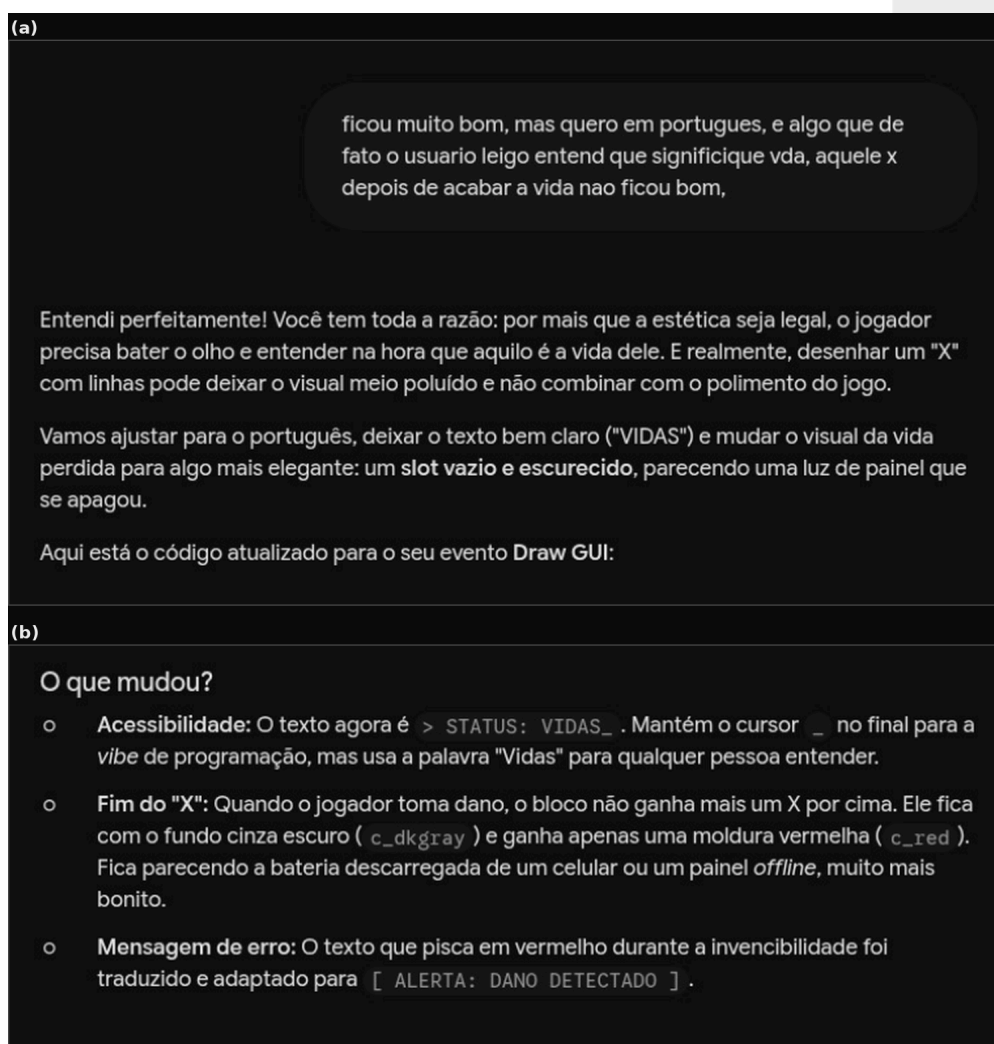
5.4 Combate, dano e progressão

O projétil obj_tiro desloca-se a oito pixels por quadro na direção do olhar do personagem, com intervalo mínimo de quinze quadros entre disparos e autodestruição cem pixels além dos limites da sala, o que evita acúmulo de instâncias. A detecção de acerto emprega colisão circular com raios assimétricos: seis pixels contra o cenário e quinze pixels contra o inimigo. A assimetria é intencional e constitui decisão de *game feel* (o acerto no alvo é mais permissivo do que a obstrução pela geometria, o que torna o combate mais responsivo sem tornar o cenário permeável).

A representação visual do projétil realiza a metáfora central do jogo por desenho procedural, sem uso de sprite: cada tiro é um dígito binário sorteado a cada quadro, acompanhado de um rastro de oito caracteres com opacidade decrescente e oscilação vertical, desenhado em três camadas sobrepostas (sombra de contraste, halo difuso e núcleo luminoso). O jogador, literalmente, dispara código.

O sistema de vidas emprega variável global persistente entre salas, inicializada em três. A colisão com um inimigo ativo desconta uma vida, concede 150 quadros de invulnerabilidade com piscar em vermelho e aplica impulso vertical de afastamento. Contato com o objeto letal ou queda abaixo do limite inferior da sala provocam morte imediata. Um alarme decide entre reiniciar apenas a sala corrente, quando ainda restam vidas, ou reiniciar o jogo. A interface, desenhada na camada de GUI, apresenta o rótulo de terminal STATUS: VIDAS e três blocos de estado com preenchimento e contorno diferenciados, além de alerta intermitente durante a invulnerabilidade. A Figura 2 exemplifica a natureza dessa assistência: a primeira versão proposta pelo modelo foi entregue em inglês e com uma marcação de vida perdida julgada ilegível para o jogador leigo, tendo sido reformulada apenas depois de crítica explícita do desenvolvedor.

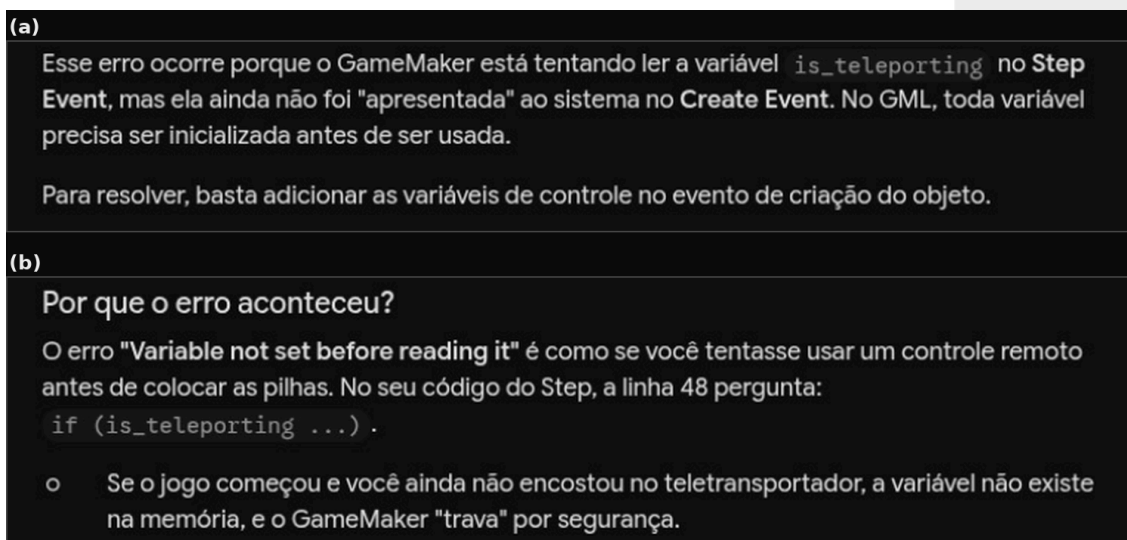
Figura 2 — Consulta assistida para o indicador de vidas: crítica do desenvolvedor à proposta inicial (a) e síntese das alterações aplicadas pelo modelo (b)



Fonte: registros de consulta assistida à linguagem GML conduzida pelo autor, preservados em docs/conceitos.

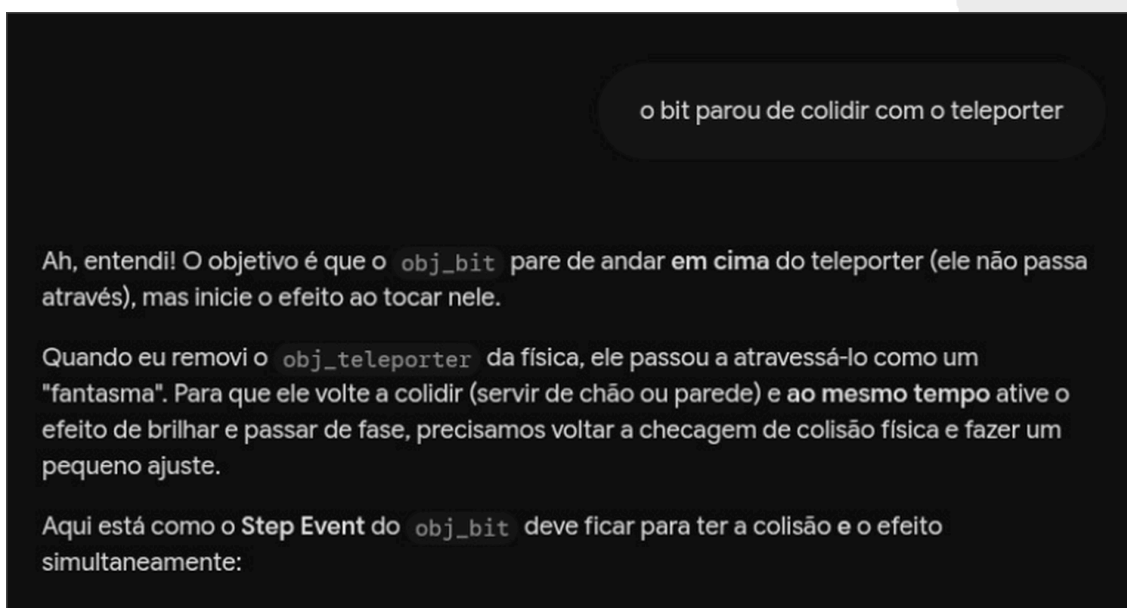
A progressão entre fases é feita pelo `obj_teleporter`, que exige permanência de 120 quadros sobre a plataforma e fornece retorno progressivo (variação cromática, tremor a partir da metade do carregamento e vibração proporcional). O destino é configurado por instância, com retorno à sala seguinte como comportamento padrão, o que permite encadear novas fases sem alteração de código. A implementação desse sistema evidencia os limites da assistência automatizada em código: a primeira solução sugerida produziu erro de execução por variável não inicializada (Figura 3) e a correção seguinte introduziu regressão na malha de colisão, fazendo o personagem atravessar o teleportador (Figura 4). O comportamento correto só foi obtido após a depuração conduzida pelo desenvolvedor.

Figura 3 — Erro de execução decorrente de código sugerido pelo modelo: variável de controle do teleportador utilizada no evento Step sem inicialização no evento Create



Fonte: registros de consulta assistida à linguagem GML conduzida pelo autor, preservados em docs/conceitos.

Figura 4 — Regressão introduzida pela correção subsequente: ao retirar o `obj_teleporter` da malha de colisão, o personagem passou a atravessá-lo, exigindo nova rodada de ajuste



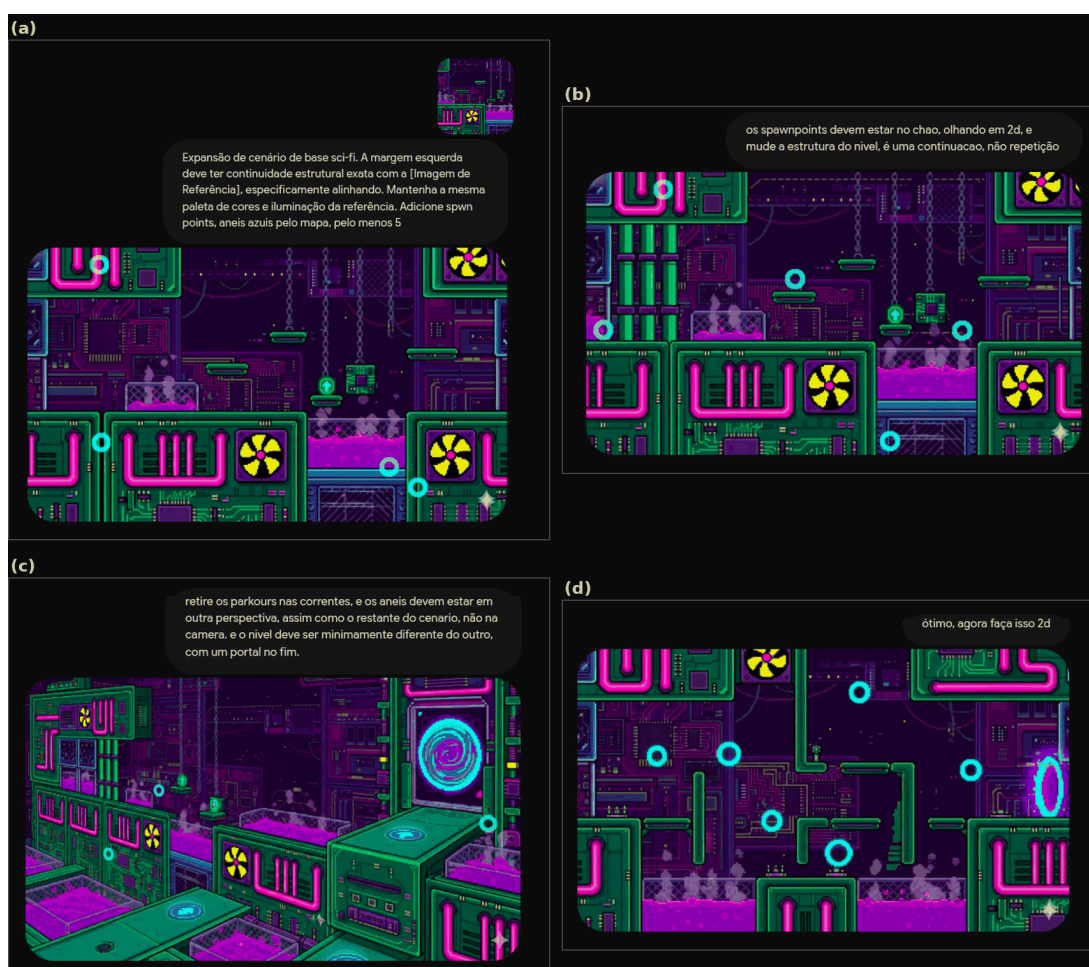
Fonte: registros de consulta assistida à linguagem GML conduzida pelo autor, preservados em docs/conceitos.

5.5 Experimentação com engenharia de prompt e formulação da Composição Modular em Camadas

O resultado mais relevante da pesquisa é negativo. Nas três rodadas do protocolo descrito na seção 4.5, nenhum cenário gerado atingiu o critério de aceitação estabelecido. Em

prompt-to-image sob zero-shot, as saídas apresentaram atmosfera coerente e alta qualidade estética, mas com plataformas dispostas em alturas incompatíveis com o alcance do pulo, bordas irregulares e caminhos interrompidos. A reformulação do prompt com restrições topológicas explícitas produziu variação estilística, não estrutural: o modelo incorporou vocabulário da instrução sem alterar a organização espacial do resultado. A Figura 5 documenta essa sequência: as instruções corretivas alteraram perspectiva, enquadramento e densidade de elementos, mas não produziram a geometria alinhada exigida pelo critério de aceitação.

Figura 5 — Rodadas sucessivas de prompt-to-image para o cenário da fase, com instruções corretivas de continuidade, perspectiva e posicionamento dos pontos de surgimento

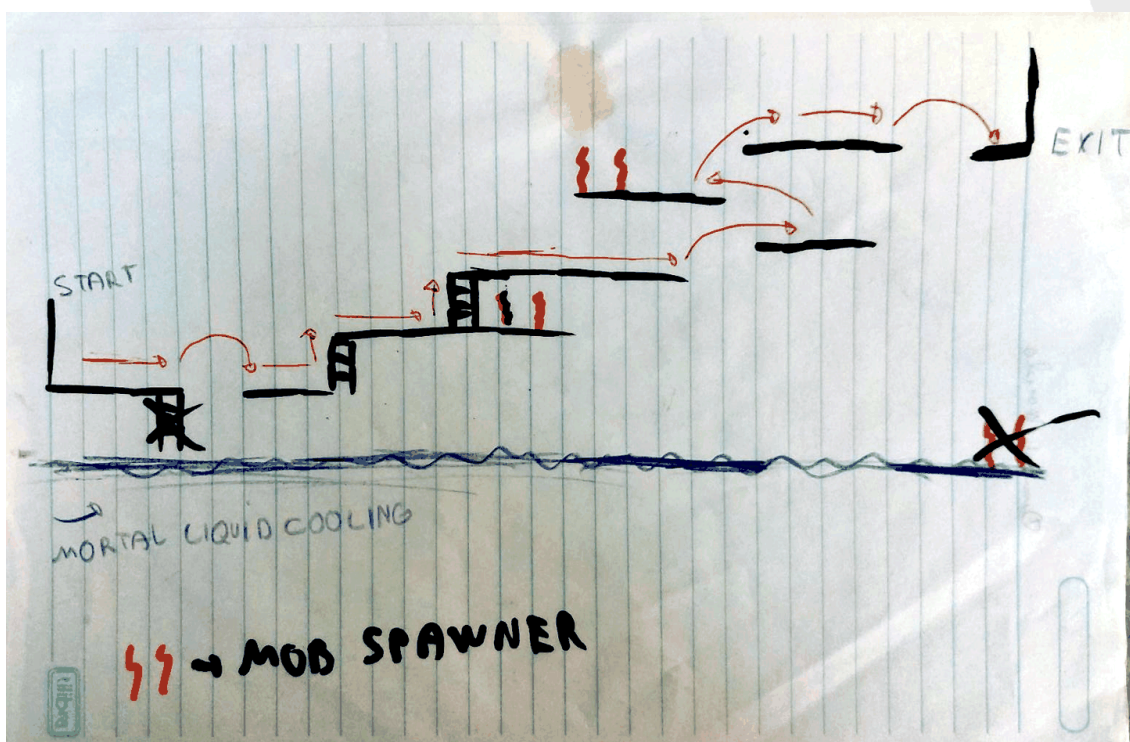


Fonte: registros de experimentação do autor com o modelo Nano Banana 2 (Gemini 3.1 Flash Image), preservados em docs/conceitos.

A rodada de sketch-to-image foi a mais elucidativa. Fornecido um esboço manual da planta da fase, o modelo tratou o desenho como sugestão estilística e não como restrição geométrica: preservou a atmosfera e a paleta, mas reposicionou, fundiu ou dissolveu as

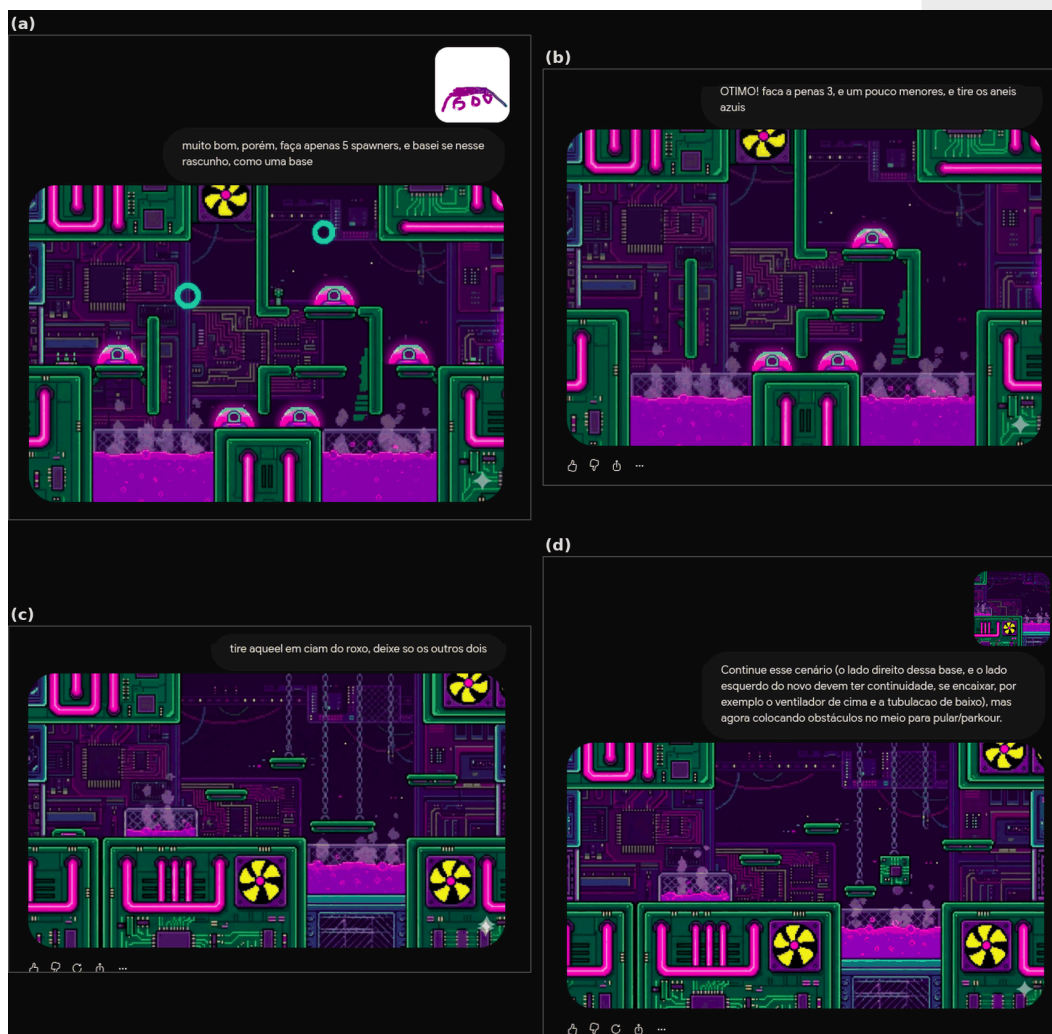
plataformas indicadas. A hipótese inicial do projeto (de que o condicionamento por esboço seria suficiente para que o modelo respeitasse regras físicas implícitas) foi assim refutada de forma consistente. O esboço fornecido ao modelo é reproduzido na Figura 6, e a Figura 7 registra a rodada, do condicionamento pelo desenho às tentativas subsequentes de correção pontual por instrução textual.

Figura 6 — Esboço manual da planta da fase, elaborado pelo autor e fornecido ao modelo como condicionamento estrutural na rodada de sketch-to-image



Fonte: elaborado pelo autor.

Figura 7 — Rodada de sketch-to-image: saída obtida a partir do esboço da Figura 6 (a) e refinamentos subsequentes por instrução textual (b, c, d)



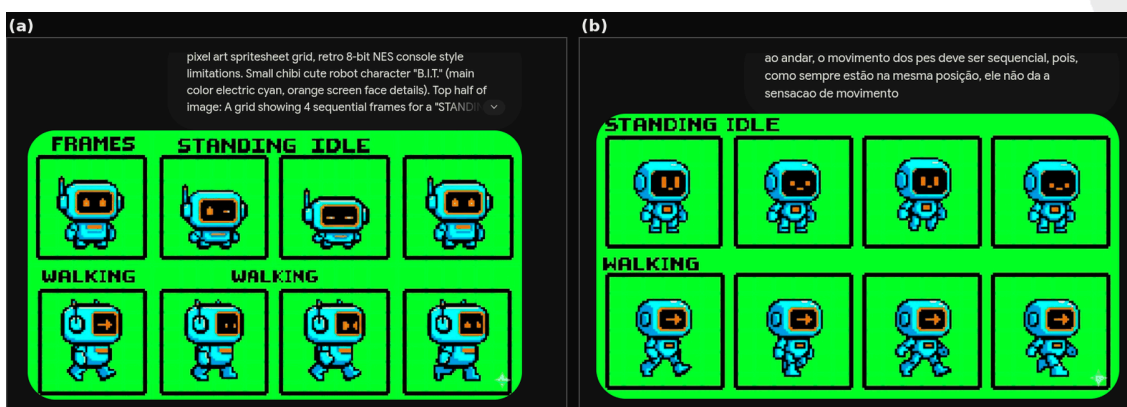
Fonte: registros de experimentação do autor com o modelo Nano Banana 2 (Gemini 3.1 Flash Image), preservados em docs/conceitos.

A resposta arquitetural a esse impasse foi a Composição Modular em Camadas, implementada diretamente na estrutura de camadas do motor. A camada de fundo, situada em profundidade 100, recebe exclusivamente o backdrop gerado por IA (o ativo backgroundMainframe, de 4800 por 896 pixels). A camada de instâncias, em profundidade zero, recebe integralmente a geometria funcional, construída e posicionada manualmente. A separação é estrita: nenhum elemento gerado participa da malha de colisão e nenhum elemento de colisão depende da imagem gerada.

A fase Mainframe foi construída sob esse regime, ocupando 5760 por 1080 pixels (aproximadamente quatro telas de largura) com 47 instâncias posicionadas manualmente: treze

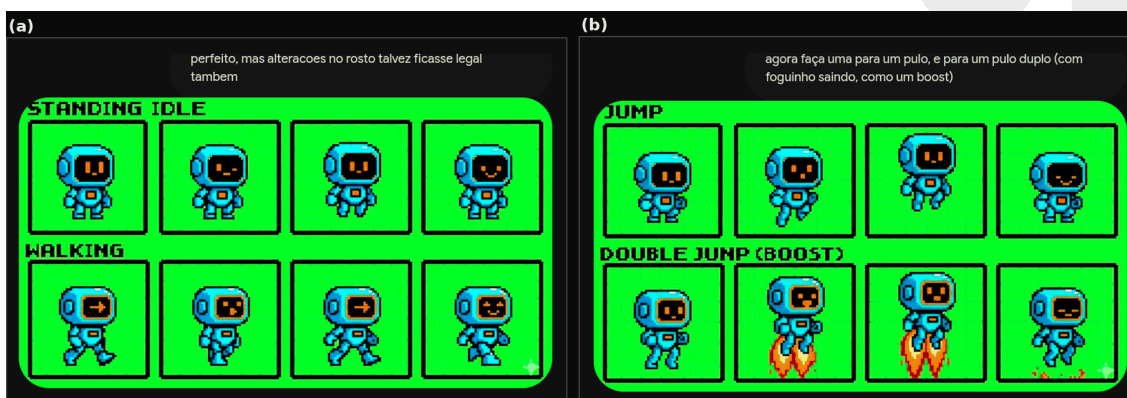
plataformas, doze terminações de plataforma, sete segmentos de escada, cinco inimigos e um teleportador. O arranjo resultante mantém a identidade visual pretendida sem qualquer concessão na precisão mecânica. Em contrapartida, a geração de ativos isolados (sprites de personagem, de inimigo e de objetos de cenário) mostrou-se produtiva, ainda que igualmente dependente de correção humana iterativa, como registram as Figuras 8 a 10.

Figura 8 — Geração do spritesheet do protagonista B.I.T.: primeira saída do modelo (a) e resultado após instrução corretiva sobre a sequência do ciclo de caminhada (b)



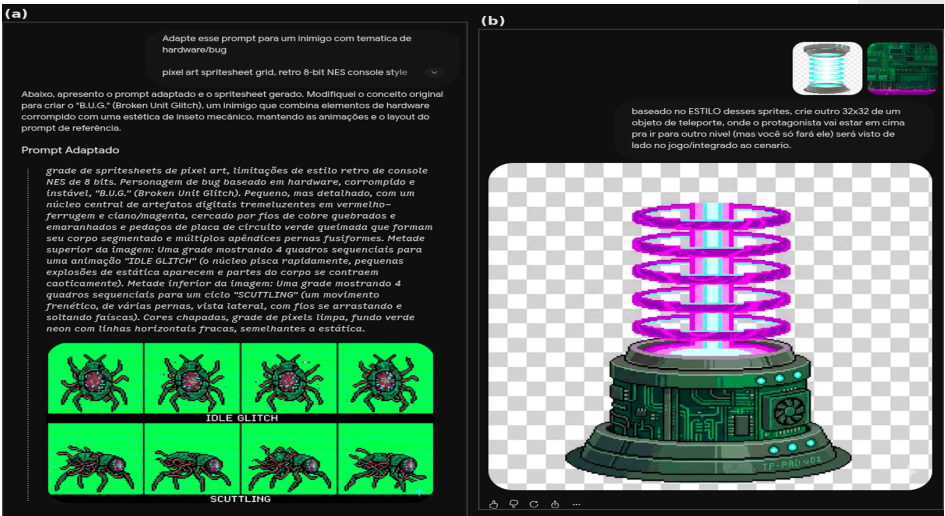
Fonte: registros de experimentação do autor com o modelo Nano Banana 2 (Gemini 3.1 Flash Image), preservados em docs/conceitos.

Figura 9 — Refinamentos posteriores do mesmo ativo: variação de expressão facial (a) e ciclos de pulo e de pulo duplo com propulsão (b)



Fonte: registros de experimentação do autor com o modelo Nano Banana 2 (Gemini 3.1 Flash Image), preservados em docs/conceitos.

Figura 10 — Derivação de ativos por adaptação de prompt: o inimigo B.U.G. (a) e o teleportador obtido por transferência de estilo a partir de sprites de referência (b)



Fonte: registros de experimentação do autor com o modelo Nano Banana 2 (Gemini 3.1 Flash Image), preservados em docs/conceitos.

A Tabela 2 sintetiza a eficácia observada da assistência automatizada em cada etapa do pipeline de produção.

Tabela 2 — Eficácia da IA generativa por etapa do pipeline de produção

Etapa	Eficácia	Observação
Concepção visual e definição de estética	Alta	Exploração rápida de direções artísticas; saída aproveitada como referência.
Geração de backdrops	Alta	Critério puramente perceptual; ativo incorporado à camada de fundo sem redesenho.
Geração de sprites do personagem	Média	Base aproveitável, porém com necessidade de tratamento <i>pixel perfect</i> , remoção de artefatos e unificação de paleta.
Consulta sintática e depuração em GML	Média-alta	Respostas úteis em problemas locais, sempre sujeitas a auditoria e reescrita.
Arquitetura de sistemas interdependentes	Baixa	Sugestões corretas isoladamente, mas inconsistentes quando havia estado compartilhado entre objetos.
Level design e topologia navegável	Nula	Nenhuma das três rodadas do protocolo atingiu o critério de aceitação.

Fonte: elaborado pelo autor a partir dos registros de experimentação do projeto.

5.6 Documentação audiovisual

A playlist pública

(https://www.youtube.com/playlist?list=PLG5uEMVXHVVVe_PjFykboGsXb5euGNZH8)

reúne oito devlogs que cobrem a trajetória do protótipo, do primeiro teste de movimentação e da correção do deslizamento de sprite conhecido como *moonwalk*, passando pela implementação da gravidade e do pulo duplo, até a build atual com composição híbrida de camadas. O registro em vídeo cumpre função metodológica específica: torna verificáveis comportamentos dinâmicos que a inspeção estática do código não evidencia, como a sensação de peso na queda, a legibilidade do estado do inimigo e a resposta perceptual ao dano.

VI. DISCUSSÃO

A refutação da hipótese inicial encontra explicação precisa na literatura. Relações espaciais estão sub-representadas nos pares imagem-texto utilizados no treinamento de modelos texto-imagem, de modo que tais sistemas aprendem com muito mais competência o que deve aparecer em uma cena do que onde e em que relação os elementos devem aparecer (CHATTERJEE et al., 2025). A falha observada neste projeto, portanto, não é falha de formulação do prompt: é ausência de competência no modelo. Nenhuma reformulação em linguagem natural recupera uma capacidade que não está representada na distribuição de treino. Esse ponto reposiciona a engenharia de prompt em termos mais sóbrios do que o discurso corrente sugere (como instrumento de exploração de um espaço latente já aprendido, e não como mecanismo de imposição de restrições verificáveis (SAHOO et al., 2024)).

O resultado converge também com o diagnóstico sobre geração procedural de conteúdo (FARROKHI MALEKI; ZHAO, 2024). As técnicas clássicas de PCG garantem jogabilidade porque incorporam o critério de jogabilidade ao próprio processo de geração, por regras construtivas, função de avaliação ou satisfação de restrições. Modelos de difusão não possuem mecanismo equivalente: otimizam verossimilhança perceptual, não navegabilidade. A Composição Modular em Camadas formulada neste trabalho é uma resposta pragmática a essa assimetria (admite a IA onde o critério de qualidade é perceptual e a exclui onde o critério é funcional e verificável).

No plano da codificação, a experiência do projeto reproduziu o mesmo padrão observado na literatura: sugestões automatizadas mostraram-se corretas e úteis em problemas locais e bem delimitados, mas degradaram diante de estado compartilhado entre objetos (MOLISON et al., 2025). O caso mais ilustrativo foi a rotina de resolução de colisão: ela precisou ser deliberadamente mantida idêntica, com os mesmos três alvos sólidos, entre o personagem e o inimigo. Sugestões automatizadas tratavam cada objeto isoladamente e

produziam divergências de comportamento (inimigos atravessando o teleportador, por exemplo) que só se revelavam em execução. A consistência sistêmica exigiu decisão humana de arquitetura, exatamente o tipo de problema estrutural que a literatura identifica como limite dos modelos em tarefas complexas e como a fronteira persistente da assistência automatizada: a decisão arquitetural (MALLAMPATI, 2025).

O contraste conceitualmente mais instrutivo do trabalho, contudo, é interno ao próprio artefato. Legacy Code: Override contém inteligência artificial em dois sentidos incomunicáveis entre si. A IA que auxiliou a produzi-lo é um modelo generativo probabilístico, opaco, não determinístico e avaliado por plausibilidade média. A IA que habita o jogo (o comportamento do `obj_bug`) é uma máquina de estados finitos determinística, de quatro estados e cerca de 140 linhas, integralmente inspecionável, depurável e reproduzível. A dominância persistente da segunda na indústria, mesmo com técnicas mais sofisticadas disponíveis, é explicada por previsibilidade, custo e controle autoral: um inimigo precisa ser justo e legível, e legibilidade é propriedade projetada, não emergente (YANNAKAKIS; TOGELIUS, 2025). A mesma tensão é registrada ao inventariar os papéis viáveis de modelos de linguagem em jogos, situando os usos consolidados na assistência à produção e não na substituição do projetista (GALLOTTA et al., 2024).

Cabe explicitar as limitações desta pesquisa. Trata-se de estudo de caso único, conduzido por um único desenvolvedor, sem grupo de controle e sem medição quantitativa comparada de tempo de produção com e sem assistência automatizada; as conclusões sobre produtividade são, portanto, qualitativas. Não foram realizados testes de jogabilidade com usuários externos, de modo que as decisões de *game feel* documentadas refletem o julgamento do próprio desenvolvedor. Por fim, os modelos avaliados constituem um recorte temporal de um campo em transformação acelerada (o modelo utilizado na geração de imagens foi lançado em fevereiro de 2026, no curso desta pesquisa), razão pela qual os limites aqui documentados devem ser lidos como limites de uma geração de modelos, e não como limites de princípio.

VII. CONCLUSÕES

Os objetivos propostos foram cumpridos. O protótipo Legacy Code: Override foi desenvolvido, versionado publicamente e documentado em vídeo, contemplando controlador de personagem, física, resolução de colisão, escadas, combate, sistema de vidas com interface, progressão entre salas e inteligência artificial de comportamento dos inimigos. A capacidade dos modelos generativos de imagem de produzir cenários de plataforma funcionais foi avaliada sob

protocolo explícito e critério de aceitação previamente definido. O pipeline de produção resultante foi formalizado e aplicado à construção integral de uma fase.

A resposta à pergunta central da pesquisa é uma delimitação, e não um veredito binário. A Inteligência Artificial generativa mostrou-se produtiva onde o critério de qualidade é perceptual e tolerante à variação (concepção visual, backdrops, bases de sprite, consulta sintática e depuração local). Mostrou-se inadequada onde o critério é funcional e verificável (topologia de fase, malha de colisão e coerência entre sistemas interdependentes). A fronteira entre os dois regimes não é o grau de dificuldade da tarefa, mas a natureza do critério de aceitação: onde existe uma resposta certa verificável, a correspondência probabilística de padrões não substitui a engenharia.

A contribuição prática do trabalho é a Composição Modular em Camadas, protocolo de baixo custo, reprodutível e transferível a qualquer projeto 2D que pretenda incorporar ativos gerados sem comprometer a precisão mecânica. A contribuição documental é o conjunto aberto formado pelo repositório de código, pelos devlogs e pelo material de conceito, que permite a outros desenvolvedores independentes verificar, replicar e contestar os achados aqui apresentados.

Como desdobramentos futuros, indicam-se: a realização de testes de jogabilidade com usuários externos, com coleta de métricas de dificuldade e legibilidade; um estudo comparativo com medição quantitativa de tempo de produção com e sem assistência automatizada; a investigação de arranjos híbridos que combinem geração de layout por algoritmos com garantia formal de navegabilidade e revestimento estético por modelos de difusão (FARROKHI MALEKI; ZHAO, 2024); e a implementação da mecânica narrativa de override, na qual o ato de instruir uma inteligência artificial se converte em verbo de jogo, aproximando definitivamente o tema da pesquisa da experiência do jogador.

O achado que melhor sintetiza os doze meses de trabalho está inscrito no próprio protótipo. As duas inteligências artificiais do projeto (a que ajudou a construí-lo e a que o habita) não se parecem. A primeira precisa acertar em média; a segunda precisa acertar sempre.

VIII. REFERÊNCIAS

CHATTERJEE, A.; STAN, G. B. M.; AFLALO, E.; PAUL, S.; GHOSH, D.; GOKHALE, T.; SCHMIDT, L.; HAJISHIRZI, H.; LAL, V.; BARAL, C. Getting it right: improving spatial consistency in text-to-image models. In: COMPUTER VISION – ECCV 2024. Cham: Springer, 2025. p. 204-222. DOI: 10.1007/978-3-031-72670-5_12. Disponível em: https://link.springer.com/chapter/10.1007/978-3-031-72670-5_12. Acesso em: 28 ago. 2026.

FARROKHI MALEKI, M.; ZHAO, R. Procedural content generation in games: a survey with insights on emerging LLM integration. In: AAAI CONFERENCE ON ARTIFICIAL INTELLIGENCE AND INTERACTIVE DIGITAL ENTERTAINMENT (AIIDE-24), 20., 2024, Lexington. Proceedings [...]. Washington: AAAI Press, 2024. DOI: 10.1609/aiide.v20i1.31877. Disponível em: <https://ojs.aaai.org/index.php/AIIDE/article/view/31877>. Acesso em: 28 ago. 2026.

GALLOTTA, R.; TODD, G.; ZAMMIT, M.; EARLE, S.; LIAPIS, A.; TOGELIUS, J.; YANNAKAKIS, G. N. Large language models and games: a survey and roadmap. IEEE Transactions on Games, 2024. Disponível em: <https://arxiv.org/abs/2402.18659>. Acesso em: 28 ago. 2026.

GOOGLE. Build with Nano Banana 2, our best image generation and editing model. 26 fev. 2026. Disponível em: <https://blog.google/innovation-and-ai/technology/developers-tools/build-with-nano-banana-2/>. Acesso em: 28 ago. 2026.

GOOGLE CLOUD. O que é Inteligência Artificial (IA)?. 2025. Disponível em: <https://cloud.google.com/learn/what-is-artificial-intelligence?hl=pt-BR>. Acesso em: 28 ago. 2026.

IBM. What is Artificial Intelligence (AI)?. 2024. Disponível em: <https://www.ibm.com/think/topics/artificial-intelligence>. Acesso em: 28 ago. 2026.

MALLAMPATI, B. The role of generative AI in software development: will it replace developers?. World Journal of Advanced Research and Reviews, v. 26, n. 1, p. 2972-2977, 2025.

DOI: 10.30574/wjarr.2025.26.1.1387. Disponível em: https://wjarr.com/sites/default/files/fulltext_pdf/WJARR-2025-1387.pdf. Acesso em: 28 ago. 2026.

MOLISON, A. S.; MORAES, M.; MELO, G.; SANTOS, F.; ASSUNÇÃO, W. K. G. Is LLM-generated code more maintainable & reliable than human-written code?. arXiv:2508.00700, 2025. Disponível em: <https://arxiv.org/abs/2508.00700>. Acesso em: 28 ago. 2026.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. Nature, v. 323, n. 6088, p. 533-536, 1986. DOI: 10.1038/323533a0. Disponível em: <https://doi.org/10.1038/323533a0>. Acesso em: 28 ago. 2026.

RUSSELL, S. J.; NORVIG, P. [Artificial intelligence: a modern approach](#). 4. ed. Hoboken: Pearson, 2020.

SAHOO, P.; SINGH, A. K.; SAHA, S.; JAIN, V.; MONDAL, S.; CHADHA, A. A systematic survey of prompt engineering in large language models: techniques and applications. arXiv:2402.07927, 2024. Disponível em: <https://arxiv.org/abs/2402.07927>. Acesso em: 28 ago. 2026.

YANNAKAKIS, G. N.; TOGELIUS, J. Artificial intelligence and games. 2. ed. Cham: Springer, 2025. DOI: [10.1007/978-3-031-83347-2](https://doi.org/10.1007/978-3-031-83347-2).

APÊNDICE — ACESSO AOS ARTEFATOS DA PESQUISA

Repositório de código-fonte (GitHub): <https://github.com/joaoaugusto-dev/legacy-code-override>

Documentação audiovisual (playlist de devlogs no YouTube): https://www.youtube.com/playlist?list=PLG5uEMVXHVVVe_PjFykbogsXb5euGNZH8

Os relatórios teórico e prático produzidos nas etapas anteriores, bem como o material de concepção visual e os registros de experimentação com IA generativa, encontram-se no diretório docs do repositório.