UNIVERSIDADE FEDERAL DE MINAS GERAIS
Departamento de Ciência da Computação

Desenvolvimento de um Jogo Digital utilizando a Biblioteca SDL

Relatório Final

Marceline Lommez Rodrigues de Jesus
Orientador: Prof. Lucas N. Ferreira
Disciplina: Monografia em Sistemas de Informação II
Belo Horizonte, MG, Brasil
2026

Resumo

Resumo. Este trabalho apresenta o desenvolvimento completo de Breach Assault, um jogo digital 2D do gênero bullet heaven (também chamado de survivors-like) implementado em C++ com a biblioteca Simple DirectMedia Layer (SDL). Dando continuidade à primeira etapa do projeto, dedicada ao estudo da biblioteca, ao levantamento de requisitos e à prototipação, esta segunda etapa concentrou-se na construção do ciclo de jogo (game loop) completo, abrangendo movimentação livre em um mundo aberto, geração contínua de inimigos em ondas de dificuldade crescente, sete armas automáticas distintas, um sistema de progressão por experiência com escolhas de aprimoramento a cada nível, um confronto contra um chefe e condições de vitória e derrota bem definidas. A arquitetura do software foi estruturada de forma modular, sem o uso de uma engine pronta, aplicando padrões de projeto (Singleton, Factory, Strategy e Object Pool) e um loop principal com passo de tempo fixo. Os resultados demonstram a viabilidade técnica da SDL para sustentar centenas de entidades simultâneas a 60 quadros por segundo, além de evidenciar o valor didático do desenvolvimento de baixo nível para a compreensão profunda do funcionamento interno de um motor de jogo.

Palavras-chave: *SDL, C++, Bullet Heaven, Survivors-like, Renderização 2D, Arquitetura de Software, Padrões de Projeto, Desenvolvimento de Jogos.*

1. Introdução

O desenvolvimento de jogos digitais destaca-se como uma das áreas mais completas da computação aplicada, pois exige a integração de múltiplos domínios do conhecimento: lógica de programação, arquitetura de software, otimização de desempenho, gerenciamento de recursos, estruturas de dados, computação gráfica e experiência do usuário. Diferentemente de aplicações tradicionais, jogos operam em ciclos contínuos, processando eventos, atualizando estados e renderizando gráficos em frações de segundo. Essa característica impõe desafios próprios, como manter uma taxa de quadros estável e garantir responsividade mesmo com um grande número de entidades simultâneas na tela.

Além de sua relevância técnica, a indústria de jogos possui peso econômico e cultural inegável. Segundo a Newzoo, o mercado global movimentou cerca de US$ 187,7 bilhões em 2024, e o

Brasil figura entre os maiores mercados consumidores do mundo. Apesar disso, a produção nacional ainda é pequena frente ao consumo: o número de desenvolvedoras brasileiras cresceu de 375, em 2018, para mais de mil em 2022, mas apenas uma fração do que os jogadores brasileiros gastam é direcionada a títulos produzidos no país. Esse cenário, somado à acessibilidade cada vez maior de ferramentas e à expansão do segmento independente (indie), motiva trabalhos acadêmicos que exercitem, na prática, a criação autoral de jogos.

A escolha de desenvolver um jogo em C++ com a biblioteca SDL, em vez de recorrer a motores completos como Unity ou Unreal, justifica-se pelo equilíbrio entre acessibilidade e controle. A SDL fornece as ferramentas fundamentais (janela, renderização, eventos, áudio e temporização) sem abstrair a organização interna do software. Isso aproxima do funcionamento real dos motores usados na indústria e incentiva boas práticas de engenharia de software, como modularidade, reuso e desacoplamento entre componentes.

1.1 Objetivos

Objetivo geral: concluir o desenvolvimento de um jogo digital funcional em C++ utilizando a biblioteca SDL, aplicando princípios e boas práticas de engenharia de software, e entregando uma experiência jogável com início, meio e fim.

Objetivos específicos:

- Implementar por completo as mecânicas centrais do gênero bullet heaven: movimentação, ataques automáticos, ondas progressivas de inimigos e colisões.
- Construir o sistema de progressão do personagem, baseado em coleta de experiência, subida de nível e escolha de aprimoramentos.
- Desenvolver sistemas de suporte à jogabilidade: partículas, áudio dinâmico, HUD e telas de menu, pausa, vitória e derrota.
- Estruturar uma arquitetura modular e extensível, validando padrões de projeto adequados ao domínio.
- Garantir desempenho estável (60 FPS) mesmo em cenários de carga extrema, com grande número de entidades simultâneas.

2. Definições e Referencial Teórico

“O que é um jogo?” A pergunta admite muitas respostas, da clássica formulação de Sid Meier, que disse que “um jogo é uma série de escolhas interessantes”, a definições mais elaboradas. Jesse Schell, em *The Art of Game Design*, propõe que jogos são “atividades de resolução de problemas abordadas de forma lúdica”, e descreve quatro elementos fundamentais que estruturam qualquer jogo: mecânica, tecnologia, história e estética. Esses elementos guiaram as decisões de projeto de Breach Assault e são retomados ao longo deste relatório.

2.1 Conceitos Fundamentais

Para a compreensão do restante do texto, definem-se a seguir os principais termos do contexto de jogos utilizados neste trabalho:

- **Gameplay (jogabilidade):** a experiência de jogar, abrangendo regras, mecânicas, objetivos e a interação do jogador com o jogo.
- **Mecânica:** conjunto de procedimentos e regras que descrevem o que o jogador pode fazer e o que acontece quando o faz.
- **Condições de vitória e derrota:** acontecimentos que encerram uma partida (run) com, respectivamente, o êxito ou o fracasso do jogador.
- **HP (pontos de vida):** recurso de agentes (jogador e inimigos); quando chega a zero, o agente morre.
- **XP (experiência):** recurso coletado ao derrotar inimigos; ao acumular o suficiente, o jogador sobe de nível e recebe aprimoramentos.
- **Game loop (laço de jogo):** ciclo contínuo que processa entradas, atualiza o estado e renderiza cada quadro.
- **HUD (Heads-Up Display):** interface sobreposta que comunica informações de estado (vida, experiência, tempo, armas) ao jogador.

2.2 O Gênero Bullet Heaven (Survivors-like)

Breach Assault pertence ao gênero *bullet heaven*, também conhecido como *survivors-like*. Trata-se de um subgênero de ação que ganhou enorme popularidade a partir de 2022 com o sucesso de Vampire Survivors. Suas características definidoras são: o jogador controla apenas a movimentação, enquanto os ataques ocorrem de forma automática; hordas crescentes de inimigos cercam o jogador continuamente; e a progressão se dá pela coleta de experiência e pela escolha incremental de armas e atributos, construindo gradualmente uma "build" capaz de produzir grande volume de dano em tela.

Esse desenho de jogo é particularmente adequado a um trabalho de engenharia de software, pois foca em questões de desempenho (centenas de entidades simultâneas), gerenciamento de memória e modularidade, exatamente os aspectos identificados como críticos já na primeira etapa do projeto.

2.3 Inspirações e Jogos de Referência

A concepção de Breach Assault foi diretamente influenciada pela análise de títulos de referência do gênero. A seguir, discutem-se os principais, com aspectos de design e de produção.

Vampire Survivors (poncle, 2022). Desenvolvido inicialmente por Luca Galante, é o título que consolidou o gênero. Sua fórmula, baseada em ataques automáticos, subida de nível com três cartas de escolha, fusão de armas e partidas de cerca de 30 minutos contra hordas infinitas, serviu de base direta para o loop de jogo, o sistema de progressão e a estrutura de escolha de aprimoramentos de Breach Assault.

Brotato (Blobfish, 2023). Reinterpreta o gênero em partidas curtas, organizadas em ondas com tempo limitado e um intervalo de loja entre elas. A ênfase em runs mais curtas e ritmadas

inspirou a opção de Breach Assault por uma partida objetiva de cinco minutos, em vez de uma sobrevivência indefinida.

20 Minutes Till Dawn (flanne, 2022). Diferencia-se por exigir mira do jogador e por estabelecer um objetivo temporal explícito: sobreviver até o amanhecer. A ideia de um objetivo de tempo claro, com clímax marcado por um inimigo poderoso, está refletida no surgimento do chefe e no limite de tempo de Breach Assault.

HoloCure: Save the Fans! (Kay Yu, 2022). Título gratuito que se popularizou ao aplicar a fórmula survivors-like ao universo das VTubers do grupo Hololive. Reforça a importância da clareza do feedback visual e da diversidade de armas e itens combináveis, além de demonstrar como o gênero acomoda uma forte identidade temática, aspecto que inspirou a coesão entre narrativa, arte e mecânicas em Breach Assault.

Figura 1: Jogos de referência que inspiraram o design de Breach Assault.

3. Tecnologias e Ferramentas

3.1 Linguagem C++

A linguagem C++ (C++17) foi adotada por seu alto desempenho e pelo controle direto sobre o gerenciamento de memória, características essenciais para um jogo que precisa atualizar e renderizar centenas de entidades a cada quadro. O projeto faz uso intensivo de recursos modernos da linguagem, como ponteiros inteligentes (unique_ptr) para gerência automática de tempo de vida das entidades, contêineres da biblioteca padrão (vector), expressões lambda para encapsular comportamentos de aprimoramento e *optional* para representar resultados que podem não existir.

3.2 Biblioteca SDL2 e Extensões

A Simple DirectMedia Layer (SDL), em sua versão 2, é uma biblioteca multiplataforma que atua como camada intermediária entre o hardware e o software, encapsulando diferenças entre sistemas operacionais sem ocultar completamente a complexidade interna. Foram utilizados os seguintes módulos:

- **SDL2 (núcleo):** criação de janela e contexto de renderização acelerada por hardware, captura de eventos de teclado e temporização de alta precisão (SDL_GetPerformanceCounter).
- **SDL2_image:** carregamento de texturas em formato PNG (sprites, tilesets e elementos de interface).
- **SDL2_mixer:** reprodução simultânea de música e efeitos sonoros, com controle de canais e de volume por categoria.
- **SDL2_ttf:** renderização de fontes TrueType para todos os textos do jogo (HUD, menus, narrativa e créditos).

3.3 Ferramentas de Apoio e Recursos

O mapa do jogo foi construído no editor Tiled, exportado em formato JSON e carregado em tempo de execução por um leitor próprio. A compilação é orquestrada pelo CMake, que organiza as dependências e copia automaticamente a pasta de assets e o mapa para junto do executável. Os recursos artísticos e sonoros provêm de pacotes licenciados para uso, com destaque para o conjunto de sprites Tech Dungeon: Roguelite (Trevor Pupkin), efeitos sonoros de Helton Yan (Pixel Combat) e do pacote 400 Sounds (Chequered Ink), interface sonora de JDSherbert e trilhas de Clement Panchout e AlkaKrab (Sci-Fi Music Pack). As fontes empregadas são BoldPixels (YukiPixels) e Silver (Poppy Works).

4. Game Design e Planejamento do Jogo

4.1 Conceito e Narrativa

Breach Assault tem como subtítulo narrativo “Assalto à Fábrica”. O jogador assume o papel de um caçador de recompensas contratado por uma organização anônima para invadir a Merol Muspi, uma das maiores empresas de robótica do mundo, sob a suspeita de que estaria realizando experimentos com humanos. Após conseguir invadir o laboratório de pesquisas e obter os planos secretos, os alarmes disparam e a missão se transforma em uma fuga: o jogador precisa resistir e escapar enquanto enxames de robôs de segurança convergem sobre ele.

Essa premissa é apresentada por meio de uma introdução em texto rolante, no estilo das aberturas cinematográficas, inspirado por Star Wars. A narrativa, embora enxuta, fornece justificativa interna para as mecânicas centrais: o caráter automático dos ataques (o personagem é um especialista agindo sob pressão), a horda infinita (a segurança da fábrica) e o objetivo temporal (escapar antes de ser derrotado).

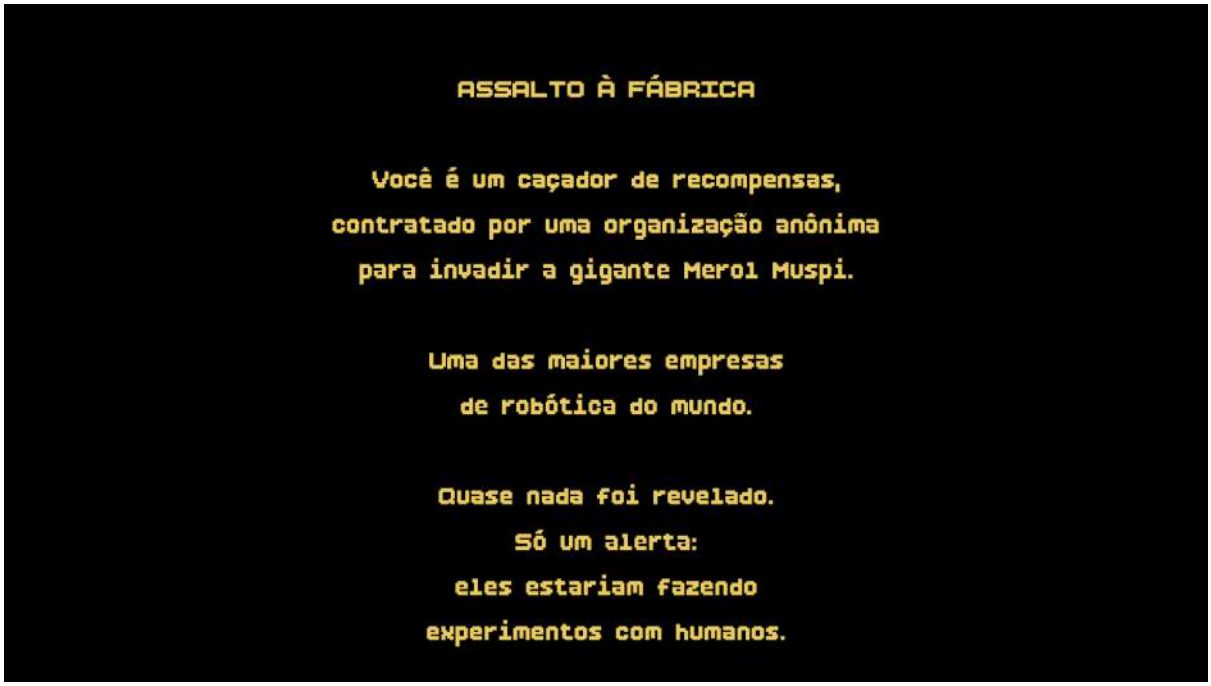

Figura 2: Introdução narrativa apresentada antes do início da partida.

4.2 Fluxo do Jogo e Ciclo de Jogabilidade

Uma partida (run) de Breach Assault dura no máximo cinco minutos. Durante esse período, o jogador movimenta-se livremente por uma arena ampla, enquanto suas armas disparam automaticamente contra os inimigos mais próximos. Os inimigos são gerados continuamente em posições fora do campo de visão, em ondas em que a frequência e composição se intensificam com o tempo. Ao derrotar inimigos, o jogador coleta gemas de experiência que, acumuladas, elevam seu nível e fazem surgir a tela de aprimoramento.

O ritmo da partida é crescente. Aos três minutos, surge o chefe, um inimigo poderoso cujo aparecimento é anunciado por um alerta sonoro e por uma mudança de trilha. A partir dos 280 segundos, um aviso visual pulsante em vermelho sinaliza a aproximação do fim. As condições de término da partida são:

- **Vitória:** derrotar o chefe. Quando isso ocorre, todos os inimigos restantes são eliminados em uma sequência de vitória e o jogador conclui a fuga.
- **Derrota:** ter o HP reduzido a zero a qualquer momento.
- **Limite de tempo:** alcançar os 300 segundos sem ter derrotado o chefe encerra a partida (o personagem é derrotado).

Figura 3: Tela principal de jogo, mostrando a horda de inimigos e o HUD.

4.3 Personagem, Experiência e Progressão

O jogador inicia cada partida apenas com a arma básica e com atributos-base: 100 de HP, velocidade de movimento de 220 unidades por segundo e uma leve regeneração de vida. A cada subida de nível, o jogo pausa e apresenta três cartas de escolha, geradas pelo sistema de aprimoramento. As opções podem ser: uma nova arma (até o limite do arsenal), o aprimoramento de uma arma já possuída (até o nível 5) ou uma melhoria de atributo. As melhorias de atributo disponíveis são apresentadas na Tabela 1.

Atributo	Efeito do aprimoramento
Max HP	Aumenta o HP máximo em 25 pontos.
Speed	Aumenta a velocidade de movimento em 8%.
Cooldown	Reduz o tempo de recarga das armas em 8% (até um limite mínimo).
Damage	Aumenta o dano de todas as armas em 12%.
Area	Aumenta a área de efeito das armas em 10%.
Pickup	Amplia em 30% o raio de coleta de experiência.
Regen	Aumenta a taxa de regeneração de vida.

Tabela 1: Aprimoramentos de atributo oferecidos ao subir de nível..

A curva de experiência é progressiva: cada nível exige aproximadamente 25% mais experiência que o anterior, partindo de 20 pontos. Essa progressão equilibra a sensação de poder crescente do jogador contra a intensificação das ondas inimigas, mantendo a tensão característica do gênero.

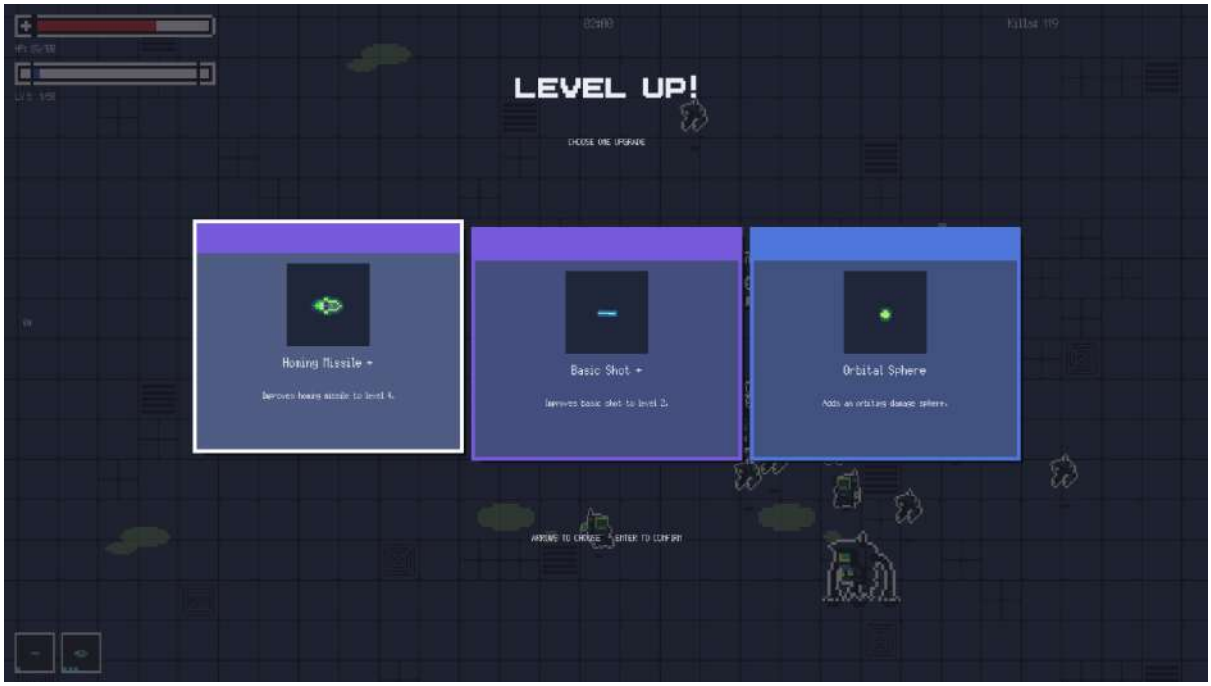

Figura 4: Tela de aprimoramento exibida a cada subida de nível.

4.4 Arsenal

Breach Assault dispõe de sete armas, cada uma com um padrão de ataque distinto e cinco níveis de aprimoramento. A diversidade do arsenal incentiva diferentes estratégias de build. Todas as armas disparam automaticamente, respeitando seus respectivos tempos de recarga. A Tabela 2 resume o arsenal.

Arma	Comportamento
Basic Shot	Dispara um projétil no inimigo mais próximo; arma inicial, com dano e cadência crescentes por nível.
Spread Shot	Lança um leque de projéteis simultâneos; níveis aumentam a quantidade de projéteis e a cadência.
Orbital Sphere	Esfera que orbita o jogador, causando dano por contato contínuo.
Pulse Wave	Onda de choque que se expande a partir do jogador, atingindo tudo ao redor em área.
Lightning Chain	Raio que salta encadeado entre inimigos próximos; níveis aumentam o número de saltos.
Homing Missile	Míssil teleguiado que persegue alvos automaticamente.
Poison Gas	Nuvem de gás que causa dano por segundo a inimigos em sua área.

Tabela 2: Armas disponíveis e seus padrões de ataque.

4.5 Inimigos e Chefe

O jogo conta com cinco tipos de inimigos comuns e um chefe, cada qual com atributos e comportamento próprios. A composição das ondas evolui ao longo do tempo: os primeiros 45 segundos apresentam apenas Walkers; em seguida, somam-se Runners, depois Tanks e, por fim, Shooters, com pesos de probabilidade ajustados por faixa de tempo. A Tabela 3 detalha os inimigos.

Inimigo	HP	Veloc.	Dano	Comportamento
Walker	30	70	10	Avança em linha reta em direção ao jogador; inimigo básico.
Runner	15	160	8	Rápido e frágil; movimenta-se em zigue-zague, dificultando a mira.
Tank	200	35	25	Lento e resistente; alto dano por contato.
Shooter	40	50	12	Mantém distância e dispara projéteis à distância.
Boss	1500	60	40	Alterna entre investida (charge) e rajadas circulares de projéteis.

Tabela 3: Tipos de inimigos, atributos principais e comportamento.

O chefe é o ponto principal da partida. Seu comportamento é organizado em duas fases que se alternam: uma fase de *investida*, em que persegue o jogador por alguns segundos, e uma fase de *rajada*, em que permanece parado disparando tiros circulares de oito projéteis em ângulos rotativos. Sua barra de vida é exibida no topo da tela, e derrotá-lo é a única forma de vencer.

Figura 5: Galeria de inimigos: Walker, Runner, Tank, Shooter e o Chefe.

4.6 Direção de Arte e Áudio

A identidade visual adota o estilo pixel art com ambientação de ficção científica industrial, combinando com a narrativa de invasão a uma fábrica de robótica. Personagens, inimigos e cenário são animados por *sprite sheets* com múltiplos estados: *idle*, movimento, ataque e morte. O áudio é dinâmico e reage ao estado do jogo: a trilha do menu e da introdução troca para a música de combate durante a partida, que por sua vez é substituída por um tema de tensão quando o chefe surge, e por temas diferentes nas telas de vitória e derrota. Efeitos sonoros acompanham cada arma, impacto, coleta de experiência e subida de nível, com canais de áudio dedicados.

5. Metodologia: Arquitetura e Implementação Técnica

O desenvolvimento foi organizado em módulos, dividindo as responsabilidades em sistemas independentes e interconectados para garantir clareza, facilidade de manutenção e extensibilidade, princípios definidos como requisitos desde a primeira etapa do projeto.

5.1 Visão Geral da Arquitetura

O código é organizado em módulos de responsabilidade bem delimitada: core (loop principal, configuração e tipos), entities (jogador, inimigos, projéteis e gemas), managers (renderização, áudio, entrada, entidades e recursos), systems (colisão, ondas, aprimoramento, partículas e

mapa), weapons (as sete armas), sprites (animação) e UI (telas de menu, pausa, HUD e fim de jogo). Um arquivo central de configuração (Config.h) concentra todas as constantes de balanceamento e de apresentação, facilitando o ajuste do jogo sem alterações pelo código.

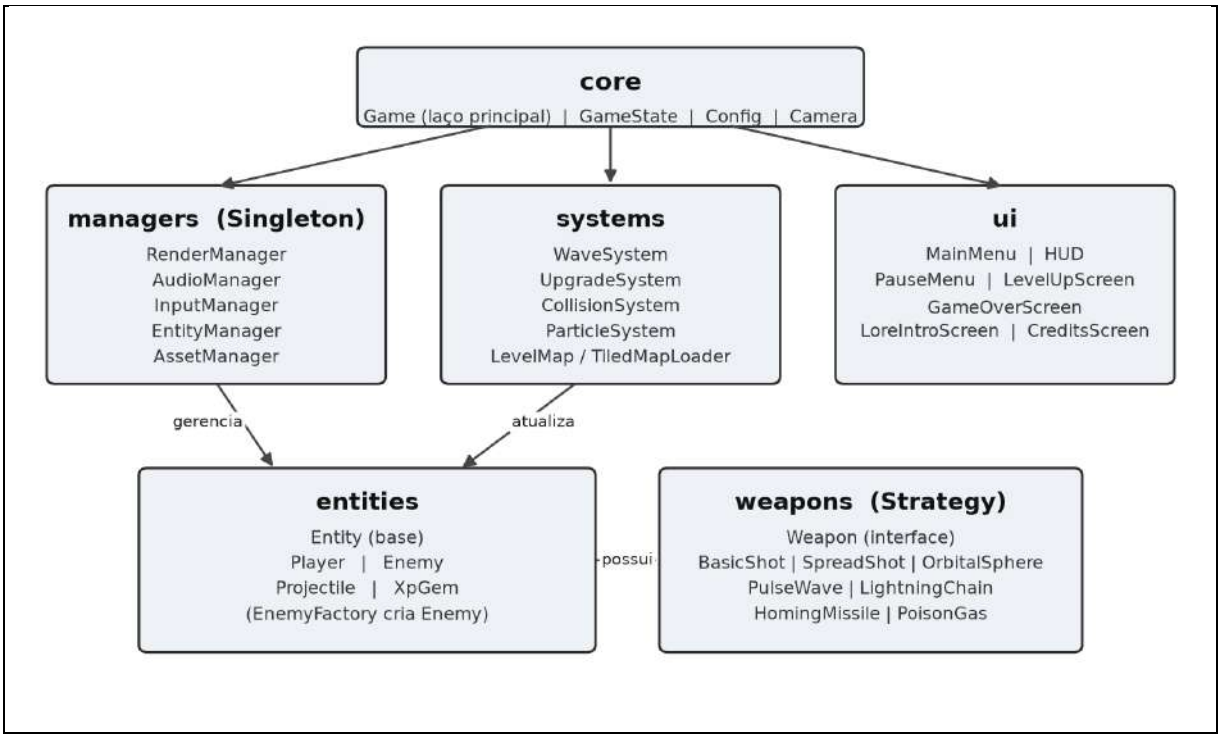

Figura 6: Visão geral da arquitetura do Breach Assault.

5.2 Loop Principal e Passo de Tempo Fixo

O loop de jogo segue o fluxo de processamento de eventos, atualização de estado e renderização. Para garantir que a simulação seja determinística e independente do hardware, adotou-se a técnica de passo de tempo fixo (fixed timestep): o tempo decorrido entre quadros é acumulado e a lógica é atualizada em incrementos constantes de 1/60 de segundo, enquanto a renderização ocorre na maior taxa possível. Um teto de tempo por quadro evita a chamada "espiral da morte" em situações de travamento momentâneo. Essa decisão garante que o comportamento de armas, inimigos e colisões seja idêntico em máquinas distintas.

5.3 Máquina de Estados do Jogo

O fluxo entre telas é controlado por uma máquina de estados explícita, com os estados: menu principal, créditos, introdução narrativa, em jogo, pausado, subindo de nível, sequência de vitória, vitória e fim de jogo. Cada estado define como entradas são tratadas e o que é renderizado, e as transições são acionadas por eventos do jogador ou do próprio jogo (morte, derrota do chefe, esgotamento do tempo). Essa separação mantém a lógica de cada tela isolada e facilita a adição de novos estados.

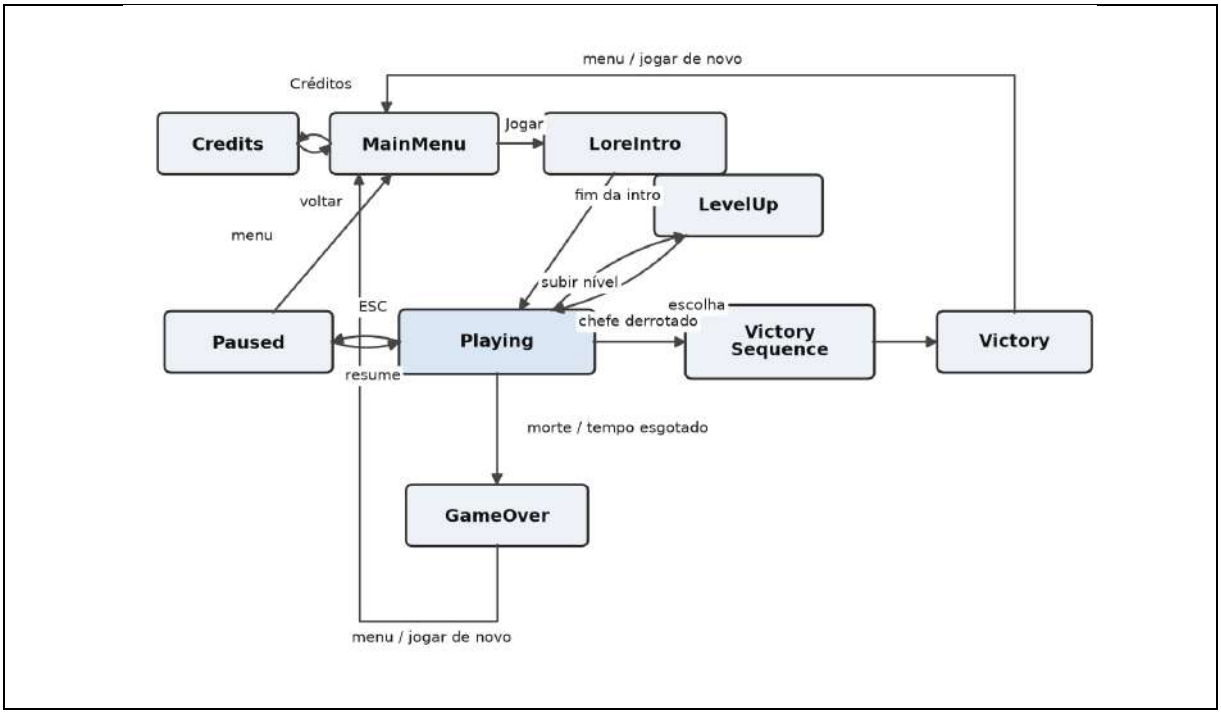

Figura 7: Máquina de estados que controla o fluxo do jogo.

5.4 Sistema de Entidades

Todas as entidades dinâmicas (jogador, inimigos, projéteis e gemas) derivam de uma classe-base comum, que reúne posição, velocidade, raio de colisão e estado de vida. O EntityManager centraliza a criação, atualização, desenho e destruição dessas entidades, armazenando-as em vetores de ponteiros inteligentes. A cada quadro, ele atualiza todas as entidades, remove as inativas e expõe consultas úteis a outros sistemas (por exemplo, a obtenção do chefe ativo).

5.5 Padrões de Projeto Aplicados

Durante a primeira etapa, três padrões foram identificados como adequados para o projeto; nesta etapa eles foram implementados:

- **Singleton:** os gerenciadores de acesso global e instância única (renderização, áudio, entrada, recursos e entidades) são implementados como Singletons, garantindo um ponto de acesso a serviços compartilhados.
- **Factory (Fábrica):** a EnemyFactory encapsula a criação dos diferentes tipos de inimigos. O sistema de ondas solicita um inimigo por tipo, sem conhecer os detalhes de construção de cada classe concreta, o que simplifica a adição de novos inimigos.
- **Strategy (Estratégia):** cada arma é uma subclasse de uma interface Weapon comum, com seu próprio algoritmo de ataque. O jogador mantém uma coleção de armas e as atualiza uniformemente, sem conhecer o comportamento específico de cada uma.
- **Object Pool (Reservatório de Objetos):** o sistema de partículas pré-aloca um reservatório fixo de partículas, reutilizando em vez de criar e destruir objetos a cada efeito visual. Isso elimina alocações dinâmicas no caminho de desempenho.

5.6 Sistemas de Jogo

Sistema de ondas (WaveSystem). Controla a geração de inimigos. Calcula o intervalo entre surgimentos de forma decrescente com o tempo, respeita um teto de inimigos ativos simultâneos (180) e seleciona o tipo de inimigo por sorteio conforme a faixa de tempo da partida. Posições de surgimento são escolhidas em um anel ao redor do jogador, fora de seu campo de visão e validadas contra obstáculos do cenário.

Sistema de aprimoramento (UpgradeSystem). Gera as três opções apresentadas a cada nível, combinando novas armas, aprimoramentos de armas existentes e melhorias de atributo, com embaralhamento aleatório. Cada opção carrega uma função que aplica seu efeito ao jogador quando escolhida, mantendo o sistema aberto a novas opções sem alterar a lógica de seleção.

Sistema de colisão (CollisionSystem). Resolve, a cada quadro, as colisões círculo-círculo entre jogador, inimigos, projéteis e gemas, além das colisões dos agentes com os obstáculos sólidos do mapa. Projéteis perfurantes registram os inimigos já atingidos para não causar dano repetido, e gemas de experiência iniciam uma animação de atração quando entram no raio de coleta do jogador.

Sistema de partículas (ParticleSystem). Produz feedback visual para impactos, mortes e subidas de nível, a partir do reservatório pré-alocado mencionado na Seção 5.5.

Mapa e cenário (LevelMap e TiledMapLoader). Carrega o mapa produzido no Tiled, distinguindo o piso de obstáculos sólidos e definindo os limites da arena. A renderização do cenário aplica recorte por visibilidade, desenhando apenas o que está dentro da câmera.

5.7 Renderização e Câmera

O RenderManager encapsula todas as chamadas de desenho da SDL, oferecendo primitivas de alto nível (retângulos, círculos, linhas, texto e quadros de sprite) em coordenadas de mundo ou de tela. Uma câmera segue o jogador, convertendo coordenadas de mundo em coordenadas de tela e limitando-se aos contornos da arena. A renderização mantêm resolução lógica fixa com filtragem nearest-neighbor, preservando a nitidez característica da pixel art independentemente da resolução real da janela, que pode operar em tela cheia ou em modo janela.

5.8 Áudio

O AudioManager gerencia música e efeitos sonoros por meio do SDL2_mixer. Cada categoria de som possui um canal dedicado e um volume próprio, e efeitos de alta frequência possuem um intervalo mínimo entre reproduções, evitando distorção. As transições de música (menu, combate, chefe, vitória e derrota) são acionadas pela máquina de estados, com efeitos de fade.

6. Resultados e Avaliação Crítica

A principal contribuição é a criação de um produto jogável e coeso, dos conceitos planejados na primeira fase. O protótipo inicial, limitado a movimentação e testes de viabilidade, evoluiu para um jogo completo, com início, meio e fim, validando as mecânicas centrais do gênero bullet heaven e a adequação da SDL ao escopo proposto. Em testes de carga, a arquitetura

sustentou o teto de 180 inimigos ativos, somados a projéteis e partículas, mantendo a taxa-alvo de 60 quadros por segundo, o que confirma as expectativas levantadas na etapa anterior quanto ao desempenho da biblioteca.

Gestão de escopo. O desafio mais relevante foi conter o *feature creep*, a tendência de acumular funcionalidades além do necessário. Isso exigiu decisões constantes de priorização, focando na entrega de um ciclo de jogo fechado e polido em vez de uma lista extensa de conteúdo.

Desempenho e desacoplamento. Manter o desempenho com muitas entidades exigiu atenção ao gerenciamento de memória (ponteiros inteligentes e reservatório de partículas) e à organização do loop de atualização. A centralização das constantes em Config.h foi decisiva para o balanceamento iterativo, permitindo ajustar dificuldade, dano e cadências sem tocar na lógica.

Atuação multidisciplinar. Como projeto individual, o trabalho exigiu a alternância constante entre programação, integração de recursos artísticos e sonoros e design de jogo. A arquitetura modular e o planejamento prévio foram determinantes para que isso não comprometesse a estabilidade do produto final.

7. Conclusão e Trabalhos Futuros

Este trabalho concluiu o desenvolvimento de Breach Assault, um jogo digital do gênero bullet heaven implementado em C++ com a biblioteca SDL, sem recurso a motores prontos. Os objetivos propostos foram alcançados: as mecânicas centrais, o sistema de progressão, os sistemas de suporte e as telas de interface foram implementados, resultando em uma experiência jogável completa e em desempenho estável. Mais do que o produto, o processo mostrou o valor didático do desenvolvimento de baixo nível para a compreensão de como sistemas interativos em tempo real são estruturados, consolidando boas práticas de engenharia de software: modularidade, reuso, desacoplamento e aplicação consciente de padrões de projeto.

Com a base técnica consolidada, é planejado as seguintes direções futuras:

1. Expansão de conteúdo: novos tipos de inimigos, armas e múltiplos chefes, aproveitando a extensibilidade da Factory e da interface Weapon.
2. Refinamento de *game feel*: efeitos visuais adicionais, como tremor de tela e maior variedade de partículas, e polimento da interface.
3. Sistemas de progressão: desbloqueios persistentes entre partidas e diferentes personagens jogáveis, ampliando a rejogabilidade.
4. Otimizações: introdução de particionamento espacial para as colisões, reduzindo a complexidade em cenários de carga ainda maior.
5. Suporte a gamepad e acessibilidade, ampliando o alcance do jogo.

Por fim, o projeto cumpriu seu propósito acadêmico e prático, demonstrando a viabilidade do desenvolvimento autoral de jogos no meio acadêmico e servindo como estudo de caso de técnicas de engenharia de software.

Referências Bibliográficas

[1] SCHELL, J. *The Art of Game Design: A Book of Lenses*. 3. ed. Boca Raton: CRC Press, 2019.

[2] HUNICKE, R.; LEBLANC, M.; ZUBEK, R. MDA: A Formal Approach to Game Design and Game Research. In: *Proceedings of the AAAI Workshop on Challenges in Game AI*. San Jose, 2004.

[3] NYSTROM, R. *Game Programming Patterns*. Genever Benning, 2014. Disponível em: https://gameprogrammingpatterns.com. Acesso em: jun. 2026.

[4] SHIFFMAN, D. *The Nature of Code*. The Nature of Code, 2012. Disponível em: https://natureofcode.com. Acesso em: jun. 2026.

[5] SDL. *Simple DirectMedia Layer: documentação oficial (SDL2 Wiki)*. Disponível em: https://wiki.libsdl.org. Acesso em: jun. 2026.

[6] NEWZOO. *Global Games Market Report 2024*. Amsterdã: Newzoo, 2024.

[7] CARDOSO, M. V.; GUSMÃO, C.; HARRIS, J. J. *Pesquisa da Indústria Brasileira de Games 2023*. São Paulo: ABRAGAMES, 2023.

[8] FORTIM, I. et al. *Pesquisa Indústria Brasileira de Games 2022*. São Paulo, 2022.

[9] PONCLE. *Vampire Survivors* [jogo digital]. Desenvolvido por Luca Galante. poncle, 2022.

[10] FLANNE. *20 Minutes Till Dawn* [jogo digital]. Erabit Studios, 2022.

[11] YU, K. *HoloCure: Save the Fans!* [jogo digital]. Kay Yu, 2022.

[12] BLOBFISH. *Brotato* [jogo digital]. Blobfish, 2023.

[13] THOMPSON, J. Collision Detection. Disponível em: https://www.jeffreythompson.org/collision-detection/index.php. Acesso em: jun. 2026.

[14] PUPKIN, T. *Tech Dungeon: Roguelite* [pacote de assets de pixel art]. Pupkin Assets.