Das raízes do arcade ao roguelike: uma releitura de Space Invaders

Title: From Arcade Roots to Roguelike: A Reinterpretation of Space Invaders

Luís Santos[1], Anderson Gabriel[1], Débora Souza[1], Pamela Bezerra[1], Tiago Barros[1], Carlos Santos[2]

Faculdade do Centro de Estudos e Sistemas Avançados do Recife – CESAR School
Avenida Cais do Apolo, n° 77 – 50.030-220 – Recife – PE – Brasil
{agvms, dcas, lfms, ptlb, tgfb}@cesar.school

Escola Politécnica de Pernambuco – POLI
Rua Benfica, 455 – Madalena – 50720-001 – Recife – PE – Brasil
chms@ecomp.poli.br

Abstract. Introduction: *We report on the development of Distress Beacon, a game written in the C programming language, created within the context of the course Imperative and Functional Programming.* ***Objective:*** *To analyze how project-based learning, centered on a motivating challenge, acts as a catalyst for the autonomous pursuit of knowledge that goes beyond the curricular content.* ***Methodology:*** *Through the development of the Distress Beacon game, we describe the practical challenges faced by the student team and the software engineering solutions (modular architecture, design patterns, and memory management) that were researched and applied to overcome them.* ***Results:*** *The experience resulted not only in a functional prototype, but, more importantly, in evidence that the need to solve concrete problems fostered autonomous learning that exceeded the curricular content.*
Keywords *Project-Based Learning, Game Development, C (Programming Language), Software Engineering, Shoot 'em up*

Resumo. Introdução: *Relatamos o desenvolvimento de Distress Beacon, um jogo em linguagem C criado no contexto da disciplina Programação Imperativa e Funcional.* ***Objetivo:*** *Analisar como a aprendizagem baseada em projetos, centrada em um desafio motivador, atua como um catalisador para a busca autônoma de conhecimento que extrapola o conteúdo curricular.* ***Metodologia:*** *Através do desenvolvimento do jogo Distress Beacon, descrevemos os desafios práticos enfrentados pela equipe de estudantes e as soluções de engenharia de software (arquitetura modular, padrões de projeto e gerenciamento de memória) que foram pesquisadas e aplicadas para superá-los.* ***Resultados:*** *A experiência resultou não apenas em um protótipo funcional, mas principalmente na comprovação de que a necessidade de resolver problemas concretos fomentou um aprendizado autônomo que excedeu o conteúdo curricular.*
Palavras-Chave *Aprendizagem Baseada em Projetos, Desenvolvimento de Jogos, Linguagem C, Engenharia de Software, Shoot 'em up*

1. Introdução e Justificativa

A disciplina de Programação Imperativa e Funcional (PIF), ofertada no segundo período do curso de Ciência da Computação da CESAR School, utiliza a Aprendizagem Baseada em Projetos para conectar seus fundamentos teóricos a um desafio prático. Neste contexto, foi proposto o desenvolvimento de um jogo em linguagem C, cujo propósito pedagógico é motivar os estudantes a superarem os desafios técnicos da linguagem por meio de um objetivo complexo e estimulante: a criação de um produto de software funcional e interativo.

Este artigo apresenta o relato de experiência do desenvolvimento do jogo *Distress Beacon*. Inspirado em clássicos da ficção científica e em jogos como *Space Invaders* [Taito, 1978], a narrativa coloca o jogador no papel de um explorador espacial que atende a um sinal de socorro e enfrenta ondas de inimigos para salvar um povo de uma ameaça iminente. O objetivo deste trabalho é analisar como a metodologia de aprendizagem baseada em projetos, quando aplicada a um desafio motivador, atua como um catalisador para a aprendizagem autônoma que extrapola o conteúdo curricular.

Para isso, primeiro descrevemos as funcionalidades e a arquitetura do jogo e, em seguida, discutimos os principais desafios práticos encontrados durante o processo. As soluções de engenharia de software aplicadas são apresentadas como evidência do aprendizado originado da necessidade de entregar um produto funcional, destacando as lições aprendidas ao longo do desenvolvimento.

2. Fundamentação

A Aprendizagem Baseada em Projetos, como discutido por [Mendes et al. 2020], é uma abordagem pedagógica na qual o conhecimento acontece de forma colaborativa e centrada no aluno. Essa metodologia é particularmente valorizada por conectar a teoria à prática através da resolução de problemas reais, preparando os estudantes para a sua futura atuação profissional [Fernandes 2014]. No ensino de programação, essa conexão é frequentemente realizada por meio de jogos educacionais e ferramentas especializadas [Monclar et al. 2018], e a revisão sistemática conduzida por [Wu e Wang 2012] indica que o aprendizado de frameworks de jogos contribui para um desempenho significativamente melhor dos alunos.

O design de *Distress Beacon* foi concebido a partir da fusão de dois gêneros: o *shoot 'em up* (*shmup*) e o *survivor-like*. O primeiro, um dos mais tradicionais dos videogames, estabelecido por clássicos como *U.N. Squadron* [Capcom, 1989], foca em reflexos rápidos e na precisão dos disparos manuais contra hordas de inimigos. Em contraste, o *survivor-like* — subgênero recente também conhecido como *bullet heaven* (Figura 1) e popularizado por *Vampire Survivors* [Poncle, 2022] — inova ao automatizar os ataques. Essa abordagem transfere o foco da microgestão dos tiros para a macrogestão do posicionamento estratégico e da evolução do personagem.

3. Processo de Desenvolvimento

Esta seção detalha o desenvolvimento do jogo Distress Beacon em duas partes. A Seção 3.1 descreve as funcionalidades do jogo sob a perspectiva do jogador, abordando suas mecânicas e sistemas. Em seguida, a Seção 3.2 apresenta o processo de desenvolvimento

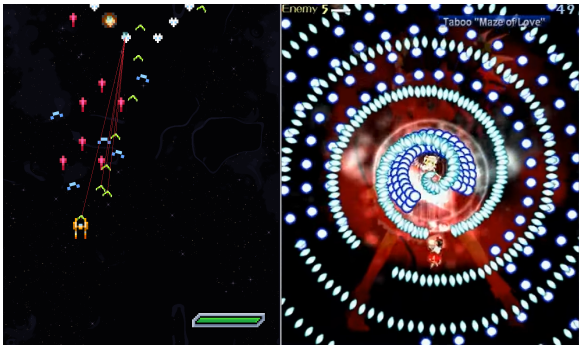

Figura 1. (A) *Bullet Heaven*, (B) *Bullet Hell* [*Touhou 6*, Team Shanghai Alice].

como um relato de experiência, destacando os desafios práticos enfrentados e como a busca por soluções exigiu a aplicação de conceitos de engenharia de software que extrapolaram o escopo curricular, evidenciando o aprendizado motivado pelo projeto.

3.1. Descrição Funcional do Jogo

O jogador controla a movimentação da nave pela tela, enquanto os disparos das armas são automáticos. Cada nave também possui uma habilidade especial única, que pode ser utilizada estrategicamente durante a partida (detalhes na Figura 2).

Os inimigos, assim como em *Space Invaders*, surgem a partir do topo da tela. No entanto, em *Distress Beacon*, reforços aparecem em intervalos de tempo definidos, e cada inimigo segue um padrão de movimentação próprio. À medida que o jogador progride, novos inimigos são introduzidos progressivamente, acumulando-se aos já presentes e criando assim combinações imprevisíveis que exigem manobras de evasão constantes para sobreviver. Uma vez derrotados, os inimigos concedem pontos de experiência.

A progressão do personagem segue uma mecânica central dos *survivor-likes*: ao acumular experiência e subir de nível, o jogador escolhe uma entre três melhorias aleatórias. Essa escolha contribui para a construção de um conjunto de aprimoramentos criando sinergias únicas. Alinhado aos pilares do design *roguelike*, esse sistema, combinado com a oferta de cinco naves distintas e dez *power-ups*, garante que cada sessão de jogo seja uma experiência distinta.

3.2. Desafios Práticos do Desenvolvimento

Desenvolvido para atender aos requisitos da disciplina, o projeto do jogo deveria ser concluído no prazo de um mês, exigindo a implementação de listas encadeadas, manipulação de arquivos, alocação dinâmica de memória, uma tabela de placar e compatibilidade com Linux. Para a parte gráfica, optou-se pela biblioteca Raylib [Santamaria 2025], cujos materiais de referência facilitaram a compreensão de conceitos como *game loop*, renderização de *assets* e captura de entradas do usuário.

O primeiro desafio consistiu em viabilizar o trabalho colaborativo entre os três membros originais da equipe. Para isso, foi necessário reorganizar o código em múltiplos arquivos, o que acarretou problemas como dependências circulares e conflitos de nomes. Esses obstáculos foram superados por meio do uso de arquivos de cabeçalho (`.h`), *forward declarations* e do modificador `static`[1] para restringir o escopo de

[1]Em C, o modificador `static` restringe a visibilidade de uma variável ao arquivo onde foi definida.

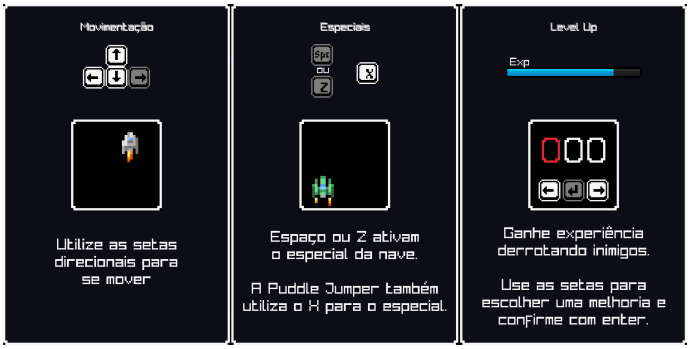

Figura 2. A tela de tutorial apresenta as principais mecânicas do jogo.

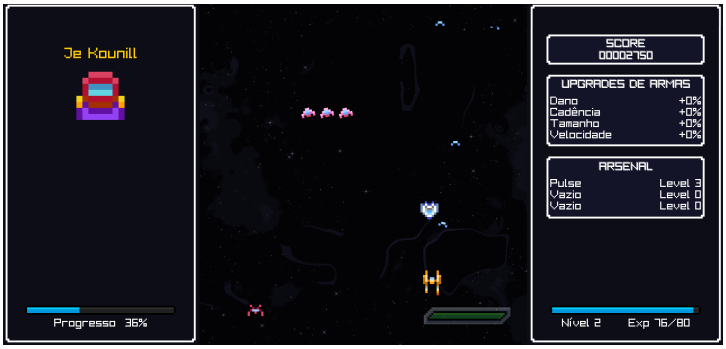

Figura 3. Interface completa do *Distress Beacon*.

funções e variáveis. Durante esse processo, evidenciou-se a importância do Princípio da Responsabilidade Única (SRP), adaptado ao paradigma imperativo, o que também contribuiu para agilizar a depuração.

A estrutura modular resultante exigiu o gerenciamento centralizado de recursos compartilhados, como texturas e sons. Para garantir instâncias únicas desses gerenciadores, aplicamos o padrão de projeto Singleton [GeeksforGeeks 2025]. A complexidade das animações e dos comportamentos dos inimigos foi tratada com o uso de máquinas de estado finitas, evitando condicionais aninhadas de difícil manutenção. Buscando organização e redução de redundâncias, exploramos a composição de `structs`[2]. Para controlar o acesso a dados entre módulos, utilizamos o modificador `extern`[3] e implementamos funções de acesso, introduzindo, na prática, o conceito de encapsulamento.

Para assegurar a compatibilidade multiplataforma, foram realizados testes no Windows e Linux, o que demandou atenção às particularidades de cada sistema e à conformidade com o padrão C. Já as dificuldades com alocação manual de memória e o risco de vazamentos (*memory leaks*) motivaram a criação de um módulo genérico de listas com ciclo de vida rastreado por um gerenciador central, permitindo a liberação de toda a memória alocada por meio de uma única função ao final da execução.

O projeto completo, incluindo assets, funcionalidades e respectivos créditos aos autores, está disponível em nosso GitHub [Santos et al. 2025].

[2]Estrutura de dados composto que define uma lista de variáveis agrupadas em um bloco de memória.
[3]Modificador para acesso global em C.

3.3. Inovações como Resultado do Aprendizado Aplicado

A superação dos desafios técnicos descritos na seção anterior não apenas resultou em uma arquitetura robusta, mas também foi o que viabilizou a implementação das mecânicas que diferenciam Distress Beacon. Embora algumas dessas ideias existissem conceitualmente desde o início, sua concretização só foi possível porque a complexidade de cada uma exigiu que a equipe pesquisasse e aplicasse soluções de engenharia de software de forma autônoma. Assim, os pontos de inovação a seguir são, em si, uma evidência do aprendizado aprofundado e motivado pelo projeto:

1. **Hibridização de Gêneros:** Combina elementos de *shmup* com a progressão estratégica de jogos do tipo *survivor-like*. Dessa forma, o jogador precisa se posicionar ativamente e tomar decisões significativas de longo prazo relacionadas à sua *build*, ambas igualmente determinantes para o sucesso na partida.
2. **Sistema de Arsenal Flexível:** Diferentemente dos *shmups* clássicos, em que o jogador conta com uma única arma principal, *Distress Beacon* permite um arsenal de até três armas distintas. Essa mecânica amplia significativamente a profundidade estratégica e a rejogabilidade, ao possibilitar combinações adaptadas a diferentes estilos de jogo e situações.
3. **Sistema de habilidades especiais:** Em *shmups* tradicionais, é comum o uso de habilidades limitadas, com alto poder destrutivo e que concedem invulnerabilidade temporária. Em *Distress Beacon*, cada nave conta com habilidades especiais que podem ser utilizadas repetidamente, limitadas apenas por um tempo de recarga. Essas habilidades combinam elementos ofensivos e defensivos, exigindo decisões estratégicas entre maximizar o dano causado ou utilizá-las como recurso de sobrevivência em situações críticas.

4. Considerações Finais

O desenvolvimento do jogo *Distress Beacon* foi muito além da aplicação de conceitos discutidos em sala de aula: constituiu uma jornada de aprendizado guiada tanto pela curiosidade quanto pela necessidade. A experiência relatada demonstra que um projeto de jogo, com seus desafios técnicos e apelo motivacional, pode atuar como um catalisador poderoso para a aprendizagem autônoma. Cada obstáculo — da organização do trabalho colaborativo ao gerenciamento de memória em C — transformou-se em oportunidade para pesquisar, compreender e aplicar conceitos de engenharia de software considerados avançados para o estágio inicial do curso.

A complexidade da arquitetura final, que à primeira vista poderia parecer excessiva para uma disciplina introdutória, demonstra o potencial da abordagem. No nosso caso, mostrou que estudantes engajados em resolver problemas que consideram relevantes são capazes de ir além do currículo formal, alcançando um nível de aprofundamento incomum em contextos acadêmicos convencionais.

Como perspectiva futura, as limitações impostas pela linguagem C despertam o interesse em explorar novas tecnologias. Entre as possibilidades estão a implementação de um modo cooperativo local e a adoção de uma linguagem com suporte a orientação a objeto, permitindo produzir implementações mais elegantes para padrões hoje simulados. A arquitetura modular já existente favorece essa transição, viabilizando a evolução do protótipo atual para uma versão mais completa, com maior profundidade e rejogabilidade.

Referências

Fernandes, S. R. G. (2014). Preparing graduates for professional practice: Findings from a case study of project-based learning (pbl). *Procedia-Social and Behavioral Sciences*, 139:219–226.

GeeksforGeeks (2025). Singleton design pattern in system design. `https://www.geeksforgeeks.org/system-design/singleton-design-pattern/`. Acessado em: 6 de julho de 2025.

Mendes, M. H., Langhi, C., Peterossi, H. G., Rubim, L., et al. (2020). Conectando a aprendizagem baseada em projetos com a experiência do aluno: uma análise do pbl à luz de dewey. *Interfaces Científicas-Educação*, 9(1):161–170.

Monclar, R. S., Silva, M. A., e Xexéo, G. (2018). Jogos com propósito para o ensino de programação. In *Anais do XVII Simpósio Brasileiro de Jogos e Entretenimento Digital (SBGames)*, pages 1132–1140.

Santamaria, R. (2025). Raylib - a simple and easy-to-use library to enjoy videogames programming. `https://www.raylib.com/`. Acessado em: 1 de julho de 2025.

Santos, L., Souza, D., Gabriel, A., e Santos, C. (2025). Distress beacon. `https://github.com/filipe-ms/Distress-Beacon`. Acessado em: 09 de agosto de 2025.

Wu, B. e Wang, A. I. (2012). A guideline for game development-based learning: A literature review. *International Journal of Computer Games Technology*, 2012(1):103710.