Formula C: Velocidade em um Novo Ângulo

Formula C: Speed from a New Angle

Eric Albuquerque[1], **João Vitor Passos**[1], **Mircio Ferreira**[1], **Pamela Bezerra**[1], **Tiago Barros**[1]

[1]Faculdade do Centro de Estudos e Sistemas Avançados do Recife – CESAR School
Avenida Cais do Apolo, n 77 – 50.030-220 – Recife – PE – Brasil

{ega2,jvap, mfsn, ptlb, tgfb}@cesar.school

Abstract. Introduction: *The use of project-based learning has proven effective in education, facilitating content assimilation. In this context, this work presents the project from the C programming course at CESAR School: a casual game designed to support the learning of fundamental programming concepts.* ***Objective:*** *To develop a Formula 1-inspired game in C that, throughout its development, applies concepts covered in class such as linked lists, dynamic arrays, matrices, and file handling, reinforcing the learning process.* ***Methodology or Steps:*** *The development followed three stages: (1) ideation, involving research on racing games; (2) implementation of basic logic and physics using the Raylib library; and (3) finalization with testing and creation of the HUD.* ***Results:*** *A complete and functional game, with a modularized code structure that facilitates maintenance, evolution, and integration of new educational content.*
Keywords *Educational Games, C Programming, Data Structures, Computer Science Education.*

Resumo. Introdução: *O uso de aprendizagem baseada em projetos tem se mostrado eficaz no ensino, facilitando a assimilação de conteúdos. Dito isto, este trabalho apresenta o projeto da disciplina de programação em C da CESAR School: um jogo casual para apoiar o aprendizado de conceitos fundamentais da linguagem.* ***Objetivo:*** *Desenvolver um jogo em C baseado na Formula 1 que aplique durante seu desenvolvimento conceitos visto em sala, como listas encadeadas, arrays dinâmicos, matrizes e arquivos, reforçando o aprendizado.* ***Metodologia ou Etapas****: O desenvolvimento seguiu três etapas: (1) ideação, com pesquisa de jogos de corrida; (2) implementação da lógica e física básica usando a biblioteca Raylib; e (3) finalização com testes e criação da HUD.* ***Resultados:*** *Um jogo completo e funcional, com estrutura de código modularizada que facilita manutenção, evolução e integração de novos conteúdos educacionais.*
Palavras-Chave *Jogos Educacionais, Programação em C, Estruturas de Dados, Ensino de Computação.*

1. Introdução e Justificativa

Este artigo apresenta o desenvolvimento do **Formula C**, um jogo de corrida em visão **top-down**, fortemente inspirado nas competições de **Formula 1**. Sua proposta é proporcionar uma experiência lúdica que simule, de forma simplificada, a pilotagem de um carro de Fórmula 1 em circuitos oficiais. O projeto foi desenvolvido inteiramente na **linguagem**

de programação C, no contexto da disciplina de **Programação Imperativa e Funcional (PIF)**, ofertada no segundo período da graduação em Ciência da Computação da **CESAR School**. A disciplina, que tem como foco introduzir os fundamentos da linguagem C, adota como principal estratégia o desenvolvimento de um jogo casual, realizado em equipes de três integrantes ao longo de um mês. O **Formula C** foi produzido no final do semestre como exercício de consolidação dos conceitos abordados, sendo seus aspectos técnicos detalhados na Seção 3.2 *Descrição Técnica*.

2. Fundamentação

Este projeto teve como principal inspiração o *remake* **F1 Spirit – The Way to Formula-1** [Maurício Braga et al. 2004], desenvolvido por fãs, que serviu como referência visual e mecânica para a criação do **Formula C**. Embora seja dos anos 90, a nova versão renovou a experiência nostálgica. A partir dessa base, buscamos desenvolver uma proposta mais moderna, alinhada aos padrões visuais e técnicos atuais.

Para a detecção de colisões, foi adotada a técnica de **color-based collision detection**, amplamente utilizada em jogos 2D por sua simplicidade e eficiência. Essa abordagem permite criar e customizar circuitos usando máscaras de *pixels*, onde cada cor indica um tipo de terreno ou colisão. A edição dessas máscaras pode ser feita com ferramentas simples, como o *Paint*, facilitando a construção da "física" dos circuitos sem grandes complexidades. Técnicas semelhantes já foram aplicadas em jogos como *Rayman* [Michel Ancel 1995], que utiliza máscaras de colisão em preto e branco, e títulos como *Worms Armageddon* [Andy Davidson 1999] e *Liero* [Joosa Riekkinen 1998], que adotam colisão *pixel a pixel* com base na leitura direta dos dados de imagem.

Além disso, a estrutura de código foi planejada com foco em **modularidade** e **clareza**. Mesmo sem orientação a objetos em C, aplicamos princípios do **SOLID** para manter coesão e baixo acoplamento seguinte as práticas de [Martin 2002]. Essa organização foi inspirada na disciplina de Fundamentos de Desenvolvimento de Software (FDS) a qual tem como foco ensinar boas práticas em grandes projetos.

O impacto educacional do projeto também se estendeu para além da sala de aula: o **Formula C** foi apresentado na mostra *TechDesign*[1], iniciativa da CESAR School que reúne estudantes de diversos cursos para expor seus projetos acadêmicos. Durante o evento, o jogo foi testado por alunos e professores da instituição, que puderam interagir com a equipe e conhecer o processo de desenvolvimento. Essa experiência dialoga com a visão de Gee [James Paul Gee 2003] sobre o papel dos jogos na aprendizagem e na resolução de problemas, e com a abordagem de Kafai [Yasmin Kafai 1993] no ensino de lógica de programação por meio do desenvolvimento de jogos. O projeto obteve destaque significativo, recebendo o **primeiro lugar geral** entre os trabalhos dos cursos de tecnologia, evidenciando seu papel como ferramenta de aprendizado prático e inovação no contexto educacional.

3. Metodologia

3.1. Descrição Funcional do Jogo

Formula C é um jogo de corrida com visão ***top-down***, ou seja, com perspectiva superior, em que o jogador controla um carro em pistas fechadas. O jogo oferece dois modos

[1] `https://www.instagram.com/reel/DLH8HyjOYr9/`

principais: **singleplayer**, no qual o jogador compete contra um carro fantasma *(ghost car)* que representa a melhor volta registrada; e **splitscreen**, onde dois jogadores disputam simultaneamente no mesmo computador com a tela dividida horizontalmente (demonstrada na Figura 1 abaixo).

A dinâmica do jogo é centrada em voltas cronometradas. A interface (HUD) apresenta as seguintes informações em tempo real para os jogadores:

- Velocímetro com velocidade atual;
- Minimapa com a posição de todos os carros;
- Tempo atual da volta;
- Diferença de tempo entre os competidores;
- Ranking (posição relativa dos jogadores);
- Sistema de semáforo durante a largada no modo *splitscreen*.

Figura 1. Tela dividida (*splitscreen*) com dois jogadores competindo simultaneamente.

O jogador pode escolher entre diferentes pistas e modos de jogo por meio de um menu interativo. Durante a corrida, a câmera pode operar em dois modos: um que segue o carro sem aplicar rotação na imagem, e outro que simula a perspectiva do piloto, alinhando a câmera à orientação do carro. Uma demo do jogo pode ser vista aqui : `https://youtu.be/fjHUECffkyQ`

3.2. Descrição Técnica

O jogo foi desenvolvido inteiramente em **linguagem C**, projetado para o sistema operacional **Linux**, com uso de duas bibliotecas externas: **Raylib 5.5**[Ramon Santamaria et al.], responsável pela renderização gráfica 2D e reprodução de sons, escolhida por sua praticidade e variedade de recursos; e **SDL2**[Sam Lantinga 2013], utilizada exclusivamente para a leitura de entrada dos controles (*gamepads*), melhorando a jogabilidade. As ferramentas adotadas incluíram o **Visual Studio Code**[Microsoft 2005] como ambiente de desenvolvimento e o **Git**[Torvalds 2005] como sistema de controle de versão. O público-alvo é geral, com destaque para entusiastas de automobilismo e jogos de corrida.

A lógica do jogo, desde a física, controle de carros e interação com a pista, até a organização estrutural dos modos de jogo e menus, foi implementada manualmente em C seguindo os requisitos de projeto da disciplina PIF. Em termos de estrutura de dados,

foram utilizadas **Listas encadeadas** para armazenar os carros em tempo de execução; **Arrays dinâmicos** para armazenar os dados do *ghost car* e das voltas de referência.

Cada pista possui uma imagem auxiliar (máscara de *pixels*) que contém codificações por cor representando diferentes propriedades, como, por exemplo, aderência do solo, regiões de colisão, *checkpoints*, etc. Essa imagem é processada em tempo real de acordo com a posição do carro, permitindo ajustar o comportamento físico conforme a superfície. Já a física do carro é baseada em **vetores polares**, representando separadamente a magnitude (velocidade) e direção do movimento. Essa abordagem facilita a implementação de curvas, deslizamentos e efeitos de atrito.

Durante o desenvolvimento, utilizamos *threads* para aprimorar a experiência do usuário em momentos de carregamento. Mais especificamente, implementamos o carregamento do mapa em uma *thread* separada da renderização, evitando o congelamento da interface gráfica durante esse processo. Essa abordagem permitiu a exibição contínua de elementos visuais, como animações e uma barra de progresso, enquanto os dados do mapa eram processados em segundo plano.

Para mais detalhes técnicos o código-fonte completo está disponíveis em: `https://github.com/iampassos/formula-c`.

3.3. Pontos de Inovação

O grande diferencial técnico do *Formula C* é o sistema de leitura de terreno por meio de uma **imagem auxiliar codificada por cor**, separada da imagem visível da pista. Isso possibilita, entre outras coisas: (1) Simular diferentes terrenos com coeficientes de atrito distintos; (2) Definir áreas de colisão e fora da pista com precisão; (3) Determinar *checkpoints* e áreas válidas da volta com flexibilidade.

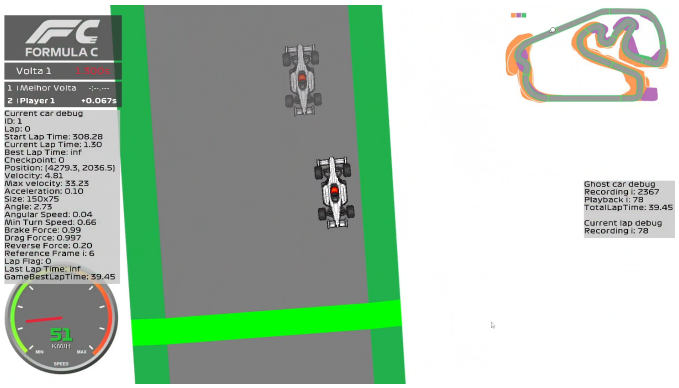

Figura 2. Modo de *debug* com o jogador competindo contra o *ghost car*.

Outro destaque é o **modo debug**, descrito na figura 2, que permite ao jogador competir contra sua melhor volta histórica (ghost car), carregada e executada em tempo real. Este recurso foi implementado com foco em leveza de dados, utilizando *arrays* dinâmicos com interpolação entre *frames* para suavizar o movimento. Além disso, o jogo oferece **duas perspectivas de câmera configuráveis**, algo incomum em jogos 2D *top-down* simples, e um **HUD completo e informativo** que simula elementos clássicos de jogos de corrida comerciais.

3.4. Processo de Desenvolvimento

O processo de desenvolvimento do *Formula C* seguiu uma abordagem incremental, orientada por testes e pela construção gradual dos subsistemas. Inicialmente, foram explorados exemplos da biblioteca Raylib para compreender seus recursos gráficos, permitindo implementar a física básica do carro e testar a movimentação na tela. Em seguida, o código foi modularizado com bibliotecas auxiliares para *arrays* dinâmicos (*arrayList*), listas encadeadas (*linkedList*), câmera e física.

Na etapa seguinte, foi criado o sistema de mapas com propriedades interativas da pista, seguido pelo modo *singleplayer*, com gravação de voltas e comparação com uma volta ideal (carro fantasma). O modo *splitscreen* foi então implementado, exigindo renderização de duas seções do mapa para cada *frame*. A **HUD** foi enriquecida com elementos visuais e sonoros, e adicionou-se uma volta de referência extra para comparações parciais. Na fase final, foram desenvolvidos o menu de seleção e a integração com *gamepads* via SDL2, substituindo o controle por teclado.

Já em termos de divisão de tarefas, a equipe, composta por três estudantes de Ciência da Computação, organizou-se de modo que um integrante ficou responsável pela física do carro, pelos testes de jogabilidade e pela documentação; outro desenvolveu as bibliotecas auxiliares, o sistema de menu, partes dos modos de jogo e a câmera; e o terceiro integrou o SDL2 para controle, criou a *HUD* e a funcionalidade do *ghost car*, além de colaborar no desenvolvimento dos modos de jogo.

Durante o desenvolvimento, enfrentamos desafios técnicos que trouxeram importantes aprendizados. A implementação da física vetorial com vetores polares reforçou o entendimento da matemática aplicada a jogos. Trabalhar com estruturas em C, como listas encadeadas e *arrays* dinâmicos, consolidou conhecimentos sobre alocação dinâmica de memória e manipulação em baixo nível. A leitura das cores da pista via buffer de imagem para definir regras de física e colisão mostrou como soluções criativas simplificam problemas complexos. Por fim, a modularização dos subsistemas evidenciou a importância de manter código legível, escalável e fácil de manter, essenciais em projetos maiores.

4. Considerações Finais

O desenvolvimento do *Formula C* impôs desafios técnicos e organizacionais que estimularam o amadurecimento da equipe, funcionando como um laboratório prático para consolidar conceitos aprendidos. Embora a busca por uma arquitetura bem estruturada tenha sido constante, enfrentamos dificuldades para manter a consistência do código e a integração entre módulos, o que reforçou a importância de compreender e aplicar princípios de qualidade de software. A padronização das **estruturas de dados** e a **definição precisa de responsabilidades** mostraram-se decisivas não apenas para a eficiência técnica, mas também como exercício de colaboração, comunicação e gestão — habilidades frequentemente negligenciadas no ensino, mas essenciais na prática profissional.

Entre os próximos passos, destacam-se melhorias nas mecânicas, aplicação de **IA**, recursos de corrida avançados e integração online, cuja implementação exigirá sólida base técnica e aprofundamento que serão desenvolvidos ao longo do curso.

5. Referências

Andy Davidson (1999). Worms Armageddon (1999). https://www.team17.com/games/worms-armageddon-anniversary-edition. Jogo desenvolvido pela Team17. Acesso em: 07 jul. 2025.

James Paul Gee (2003). What video games have to teach us about learning and literacy. https://dl.acm.org/doi/abs/10.1145/950566.950595. A importância do uso dos jogos na aprendizagem e resolução de problemas. Acesso em 07 agosto. 2025.

Joosa Riekkinen (1998). Liero (1998). https://www.liero.be, note = Jogo de código aberto inspirado em Worms. Acesso em: 07 jul. 2025.

Michel Ancel (1995). Rayman (1995). https://raym.app/. Jogo desenvolvido por Ubisoft. Acesso em: 07 jul. 2025.

Yasmin Kafai (1993). Minds in play. https://www.proquest.com/openview/c57b7cfc2f4e37f3532bf2ac07ad3979/1pqorigsite=gscholarcbl=18750diss=. Jogos na aprendizagem de lógica de programação. Acesso em 07 agosto. 2025.

Martin, R. C. (2002). *Agile Software Development: Principles, Patterns, and Practices*. Prentice Hall.

Maurício Braga et al. (2004). F1 spirit – the way to formula-1. https://f1spirit.jorito.net/. Acesso em: 03 jul. 2025.

Microsoft (2005).

Ramon Santamaria et al. Raylib: A simple and easy-to-use library to enjoy videogames programming. https://www.raylib.com/. Acesso em: 07 jul. 2025.

Sam Lantinga (2013). Simple directmedia layer 2 (sdl2). https://www.libsdl.org/. Acesso em: 07 jul. 2025.

Torvalds, L. (2005). Git - distributed version control system. https://git-scm.com/. Acesso em: 07 jul. 2025.