Pitstop F1 - Minijogo

Grupo:

Ricardo Oliveira, 57218

Docente:
João Dias

Unidade Curricular:
Programação Criativa

Curso:
Computação Criativa e Realidade Virtual, 1º ano

Índice

Acrónimos

Game / Gameplay	Jogo / Jogabilidade
Pitstop / Pitlane	Paragem nas [boxes] (F1)
Asset(s)	Recursos digitais (imagens, sons, etc.)
Frame(s)	Fotogramas
Bug	Erro de funcionamento
Highlight	Realce / Destaque visual
Overlay	Camada sobreposta
Record / Highscore	Recorde / Melhor pontuação
Random	Aleatório
Input	Entrada (de dados ou teclas)
Output	Saída (resultados, imagem, som)
keyListener / keyPressed	Detetor de teclas pressionadas
Load / Preload	Carregar / Pré-carregar ficheiros
Canvas	Área gráfica onde o jogo é desenhado
Sprite(s)	Imagem ou elemento gráfico manipulado no jogo
Sequence	Sequência
Timing	Tempo determinado
localStorage	Armazenamento local no navegador
SVG	Gráficos vetoriais escaláveis
Debug / Debugging	Depuração – processo de identificar e corrigir erros.
Feedback	Reenvio de informação
Background	Fundo / Plano de Fundo
HTML	Linguagem de Marcação de Hipertexto

Agradecimentos

Agradeço ao professor de Programação Criativa, João Dias, pela orientação e pelo apoio prestado durante o desenvolvimento do projeto.

Resumo

O presente relatório técnico descreve o desenvolvimento do minijogo "Pitstop F1", projeto final para a unidade curricular de Programação Criativa. O objetivo geral foi criar uma experiência interativa original utilizando p5.js, que demonstrasse não só o domínio das bases da programação (estruturas de controlo e variáveis), mas também a capacidade de integrar recursos externos (SVG, mp3) e implementar uma arquitetura de gestão de estados complexa. O projeto foca-se em conjugar uma ideia lúdica e estética coerente com a robustez e a organização técnica do código.

1. Título e Tema do Projeto

1.1 Título

Pitstop F1 – Minijogo

1.2 Tema do Projeto

O projeto enquadra-se no desafio C. Minijogo Criativo, que exigia o desenvolvimento de um jogo com mecânicas simples, mas apelativas, incluindo objetivos claros e feedback em tempo real. O conceito foi escolhido por traduzir a exigência de reflexos e precisão da F1 num desafio algorítmico de sequência de teclas e timing. O jogo exige que o utilizador execute corretamente e rapidamente uma série de tarefas (troca de partes do carro) acionando a tecla correta correspondente à parte ativa.

2. Conceito e Referências Visuais

O conceito simula a pressão de um pitstop oficial. O utilizador deve responder a uma sequência aleatória de 6 tarefas de manutenção (`totalPartes = 6`), que correspondem a partes específicas do carro (pneus e asas). A pontuação é baseada no tempo de execução (`tempoAtual`) e é influenciada pela lógica de pontos/penalização (onde o erro subtrai pontos e adiciona 1 segundo ao tempo final).

2.1 Estrutura de Interatividade

A interatividade é capturada exclusivamente por `keyPressed()`. O jogo implementa dois mecanismos de interação e feedback distintos:

- Sequência de Partida: o início do jogo é precedido pela contagem de luzes acompanhada por um som de sequência (`somPartida`). O timing desta sequência (luzes que acendem a cada segundo) é crucial para a imersão.
- Lógica de Jogo (Acerto/Erro):
  - Acerto: O feedback é uma mensagem de confirmação (“Correto!”) e a geração imediata de um novo pedido/tarefa (`gerarNovoPedido`)
  - Erro: O sistema penaliza com -5 pontos e adiciona +1 segundo (`tempoInicio = tempoInicio - 1000`) ao tempo final. Se o número de erros exceder `maxErros (5)`, o jogo é terminado em falha (`terminarJogoFalha`).

A gestão do foco (`canvas.eltfocus()`) garante que o jogo corretamente as teclas premidas pelo utilizador.

2.2 Referências Visuais

A estética F1 é alcançada através de:

- Paleta de Cores: predominância do preto (menus), com o uso estratégico do vermelho (botão e luzes) para o feedback visual.
- Composição: o carro de F1 foi desenhado estaticamente no centro para servir de ponto focal, enquanto a sequência de tarefas é apresentada num subtítulo superior horizontal, otimizando a leitura visual e a rapidez de processamento por parte do utilizador.

2.2.1 Sistema de Luzes da Fórmula 1

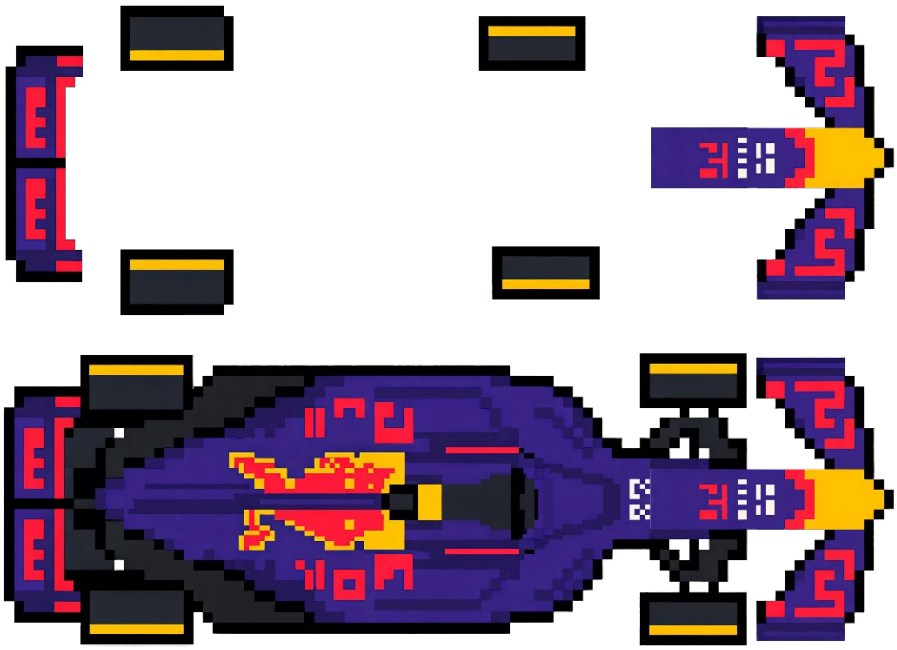
Inspirado no mecanismo oficial de largada, com cinco luzes vermelhas que acendem em sequência. O arranque acontece após todas as luzes se apagarem. No minijogo, este comportamento marca o início da corrida contra o tempo.

2.2.2 Pixel Art e Sprites SVG

Optei por *sprites* em SVG, derivados do carro Redbull RB21, pela sua escalabilidade e nitidez. Foram criadas imagens individuais para realizar as tarefas aleatórias solicitadas pelo jogo:

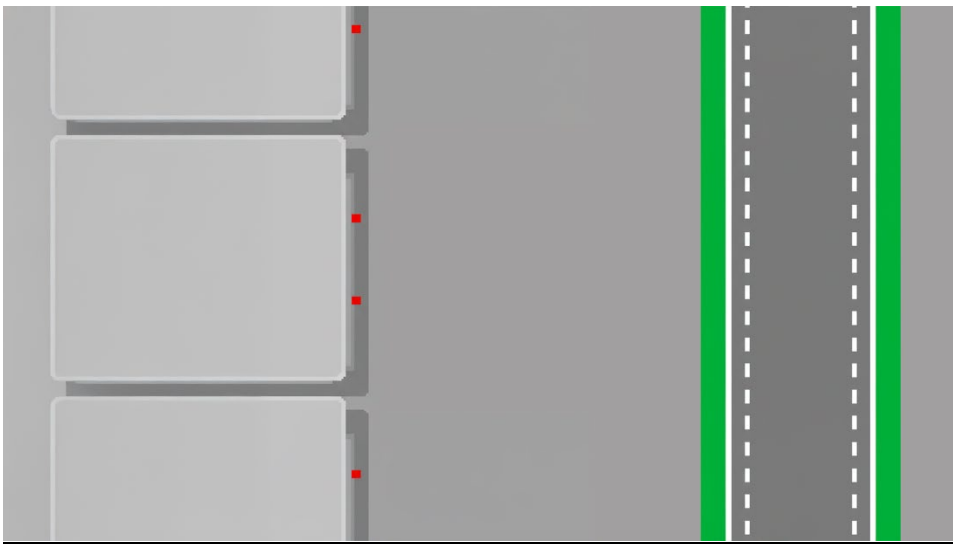
- Pneus Frontais e Traseiros (Esquerdo e Direito)
- Asa dianteira
- Asa Traseira
- Corpo completo do carro

Apesar da asa traseira não possa ser trocada num *pitstop* na realidade, foi adicionada no minijogo de forma a torná-lo mais interativo.



2.2.3 *Pitlane* (*Background*)

Desenhei um *background* imitando o *pitlane*, que reforça a ambientação sem distrair do *gameplay*.



3. Arquitetura do Código e Principais Funções

3.1 Visão Geral da Arquitetura

O minijogo desenvolvido segue uma arquitetura simples, mas eficiente, adequada ao requisito de utilizar a biblioteca p5.js. A estrutura do programa foi organizada em torno de funções independentes, cada uma com responsabilidades claramente delimitadas, garantindo a clareza do código e facilitando futuras expansões. Apesar de não recorrer a classes ou modularização avançada, o código mantém uma divisão lógica entre carregamento de recursos, inicialização, ciclo principal, renderização visual e lógica do jogo.

3.2 Principais Funções

3.2.1 Função `preload()`

A função `preload()` é responsável por carregar todos os recursos essenciais antes do início do minijogo. Este processo inclui a importação das imagens SVG que compõem o carro, as suas partes individuais e o fundo do *pitlane*, bem como o ficheiro de áudio que reproduz o som da sequência de arranque. Este carregamento prévio garante que, durante a execução do jogo, não existem atrasos, falhas de renderização ou interrupções sonoras.

3.2.2 Função `setup()` – Inicialização e Preparação do minijogo

Após os recursos serem carregados, a função `setup()` é executada uma única vez no arranque da aplicação. É nesta fase que o *canvas* é criado com o tamanho total da janela do navegador, sendo colocada dentro do elemento HTML correspondente. O `setup()` também atribui automaticamente o foco ao *canvas*, permitindo que o jogo possa capturar eventos de teclado sem exigir cliques do utilizador. A função carrega igualmente o melhor tempo guardado no `localStorage`, criando persistência entre sessões do jogador. Finalmente, apresenta-se no ecrã uma mensagem inicial que orienta o utilizador antes do início da sequência de partida.

3.2.3 Função `draw()` – Ciclo Principal de Atualização

A função `draw()` é o núcleo operativo do minijogo, sendo executada continuamente e aproximadamente 60 *frames* por segundo. Nesta função, é feita a gestão visual e lógica de cada estado do jogo: momento de espera, sequência de partida ou pitstop em curso. Quando o jogo está parado, a função limita-se a desenhar o fundo e a mensagem inicial. Durante a sequência de partida, `draw()` chama as funções responsáveis pelas luzes e pela sua temporização. Quando o pitstop está ativo, esta função atualiza o tempo de jogo, mostra a pontuação, erros e instruções, e chama a função responsável por desenhar o carro e destacar a parte correspondente.

3.2.4 Função `iniciarJogo()` – Reinicialização e Arranque do *Pitstop*

A função `iniciarJogo()` executa a preparação necessária para o início de uma nova ronda. Ela reinicia o sistema de pontuação, os erros, a contagem do tempo e o progresso das partes. Além disso, a função gera o primeiro pedido do jogador através de `gerarNovoPedido()`. Esta função marca a transição entre a fase de partida e o *gameplay* principal, garantindo que nenhuma variável residual interfere com a nova sessão de jogo.

3.2.5 Função `gerarNovoPedido()` – Seleção aleatória de Partes

Sempre que o jogador acerta a tecla correspondente à parte atual, é necessário escolher uma nova parte para continuar o pitstop. A função `gerarNovoPedido()` seleciona uma parte aleatória da lista predefinida, atribuindo-a à variável global `pedidoAtual`. Esta função contribui para a imprevisibilidade e dinamismo do minijogo, garantindo que o jogador não pode antecipar qual peça surgirá a seguir.

3.2.6 Função `keyPressed()` – Captura e Tratamento de Teclas

A função `keyPressed()` é central para a jogabilidade, uma vez que interpreta todas as teclas pressionadas pelo utilizador. Quando o jogador pressiona a tecla de espaço, e se o jogo ainda não estiver ativo, inicia-se a sequência das luzes. Casa contrário, se o jogo se encontra em curso, a função compara a tecla pressionada com a tecla esperada para a parte atual.

3.2.7 Função `terminarJogo()` e `terminarJogoFalha()` – Finalização da Ronda

A função `terminarJogo()` é chamada quando o jogador completa todas as partes necessárias. Ela calcula o tempo total da ronda, compara-o com o melhor tempo guardado e, se aplicável, atualiza o recorde no `localStorage`. Em contraste, `terminarJogoFalha()` é ativada quando o número de erros ultrapassa o limite definido. Ambas as funções encerram a ronda, removendo o pedido ativo e apresentam uma mensagem correspondente ao resultado obtido.

3.2.8 Função `atualizarSequenciaPartida()` – Temporização das Luzes de Arranque

Durante a sequência inicial de luzes, a função `atualizarSequenciaPartida()` controla quanto tempo passou desde o início da contagem, determinando quantas luzes devem estar acesas. A função também é responsável por sincronizar o áudio com o acender de cada luz, garantindo uma experiência consistente. Quanto todas as luzes completam o ciclo, a função encerra a sequência e chama automaticamente `iniciarJogo()`.

3.2.9 Função `desenharLuzesPartida()` – Representação Visual da Sequência

Enquanto a sequência de partida decorre, esta função desenha graficamente as cinco luzes vermelhas no centro do ecrã. As luzes acendem de forma gradual conforme calculado pela função anterior, proporcionando um indicador visual claro e inspirado diretamente nas partidas reais de F1.

3.2.10 Função `desenharCarro()` – Renderização do Carro e Partes

A função `desenharCarro()` desenha o carro completo no centro do ecrã, ajustando automaticamente o seu tamanho com base na resolução disponível. Após o desenho do carro base, a função aplica um `tint()` verde e sobrepõe apenas a parte correspondente ao pedido atual, destacando-a visualmente. Este mecanismo permite ao jogador identificar rapidamente a zona correta do carro que deve ser manipulada, sem necessidade de textos complexos ou setas adicionais.

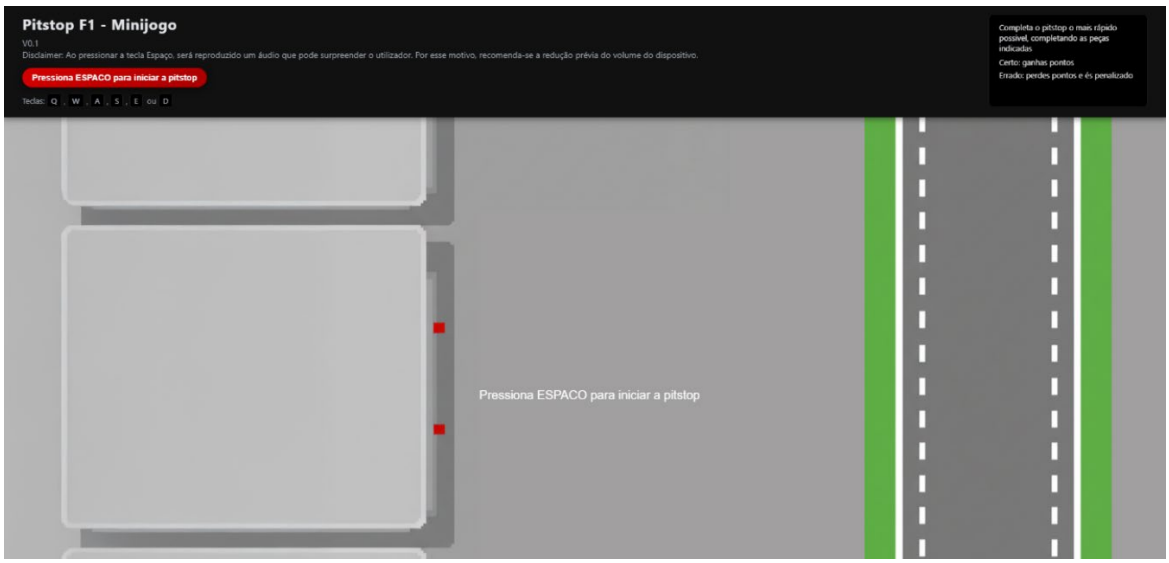
3.2.11 Função `windowResized()` – Adaptação a Mudanças de Tamanho

Por fim, a função `windowResized()` assegura que o canvas é reajustado sempre que o utilizador altera o tamanho da janela do navegador. Esta capacidade garante que o jogo permanece visualmente consistente e jogável em diferentes resoluções.

4. Printscreens

4.1 Ecrã inicial

Mostra o *canvas* vazio, instruções e botão de início. Simplicidade e clareza na interface

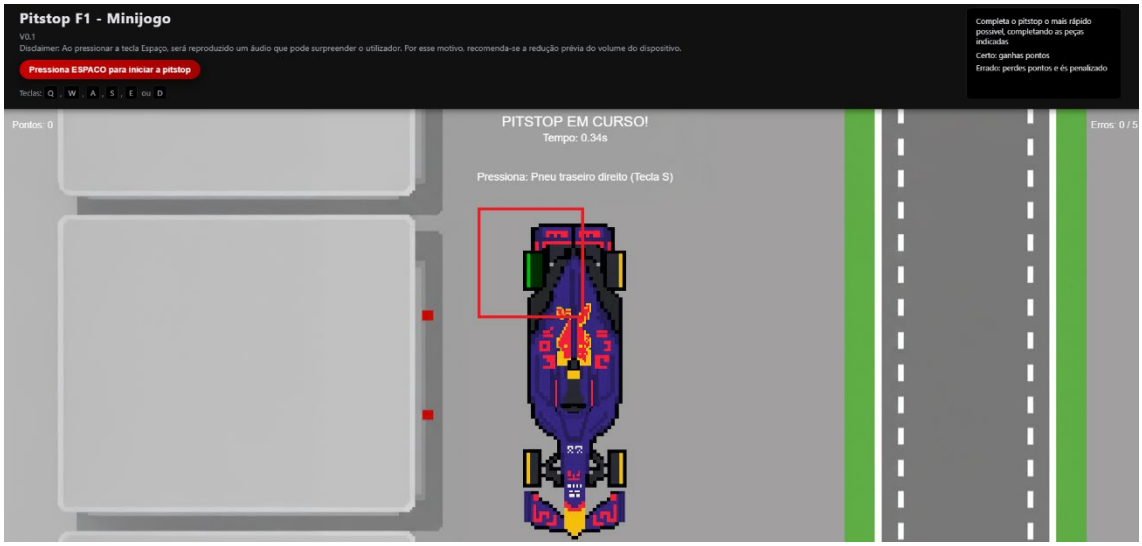


4.2 Sequência de Luzes

Indicadores visuais que simulam a verdadeira largada de F1. O áudio sincronizado aumenta a imersão

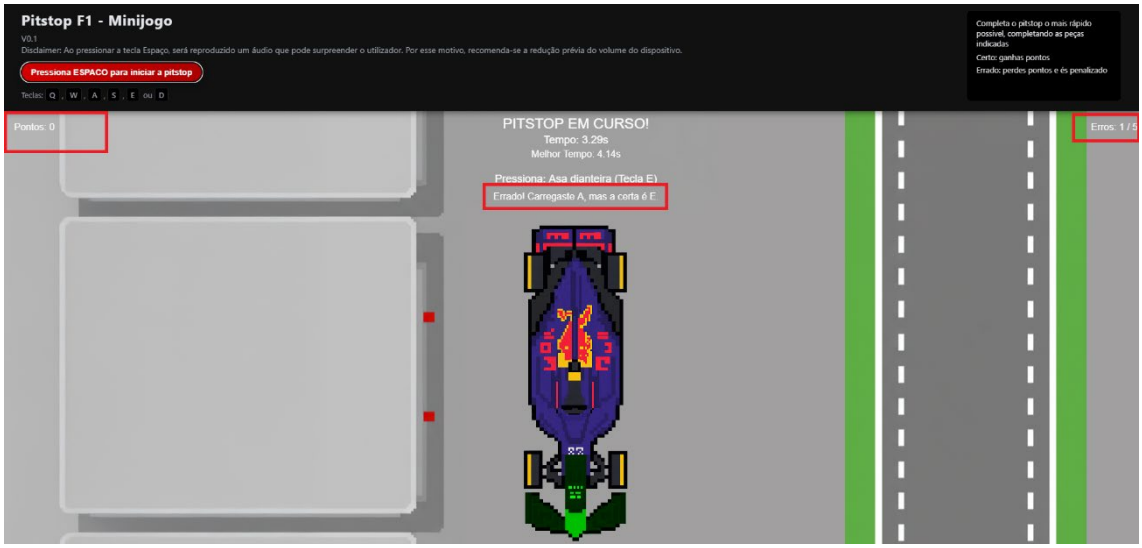
4.3 Parte ativa Destacada

O *tint* verde destaca a peça correta, facilitando a perceção visual.



4.4 Erro do Jogador

Mensagem clara e penalização reforçam o aspeto competitivo.



5. Análise Crítica e Conclusão

O principal desafio técnico foi garantir a sincronia do áudio e da lógica, especificamente na sequência de partida. Foi necessário garantir que o `userStartAudio()` era ativado pela interação do utilizador (ao premir Espaço) e que o som só era reproduzido no momento exato em que a luz acendia (`numLuzesAcesas > ultimaLuzTocada`). Outra dificuldade foi a correta proporção e desenho do carro (`desenharCarro()`) em diferentes resoluções, resolvida com a utilização de proporções dinâmicas baseadas na altura do ecrã (`let alturaCarro = height * 0.6`).

O projeto permitiu consolidar diferentes competências técnicas adquiridas ao longo do semestre. Ao longo do desenvolvimento, foi possível compreender de forma aprofundada o ciclo de vida de uma aplicação gráfica, bem como a importância de organizar o código em funções bem definidas, garantindo legibilidade e facilidade de manutenção.

A construção do sistema de pedidos aleatórios, da deteção de teclas e da lógica de validação revelou-se essencial para reforçar o domínio da programação orientada a eventos. Da mesma forma, o uso de imagens SVG de alta resolução e a necessidade de adaptar automaticamente o jogo ao tamanho do ecrã, evidenciaram também a importância de trabalhar com escalabilidade e responsividade.

O resultado final é um minijogo funcional, visualmente apelativo e tecnicamente coerente, que cumpre com os objetivos definidos inicialmente e que abre portas para futuras melhorias, como novos modos de jogo, mais partes, animações adicionais ou um sistema completo de pontuação global. Em suma, o projeto representa não apenas a aplicação prática de conhecimentos adquiridos, mas também um exercício de criatividade, organização e resolução de problemas.

6. Declaração de utilização de Inteligência Artificial

Este trabalho recorreu parcialmente a ferramentas de Inteligência Artificial generativa, utilizadas como apoio à pesquisa, revisão linguística e estruturação do documento. A utilização da ferramenta teve como objetivo complementar a recolha e organização de informação relativa às dificuldades sentidas com sincronização do áudio e da lógica, assim como, com a proporção a utilizar.

Todas as informações geradas foram revistas, traduzidas e ajustadas, de modo a garantir a conformidade com as normas de citação e integridade académica (APA, 7.ª edição).

Referências Bibliográficas

McCarthy, L., & p5.js contributors. (2023). *p5.js: A JavaScript library for creative coding*. Disponível em https://p5js.org/

MDN Web Docs - JavaScript:

Mozilla Foundation. (2024). JavaScript documentation. MDN Web Docs. https://developer.mozilla.org/

W3C - SVG Specifications:

World Wide Web Consortium. (2022). *Scalable Vector Graphics (SVG) 2*. https://www.w3.org/TR/SVG2/

Áudio e Timing em JavaScript:

Flanagan, D. (2020). *JavaScript: The Definitive Guide* (7th ed.). O'Reilly Media.