

## Programa IT Academy – Processo Seletivo – Edição #22

Nome Completo: Gabriel Alves Bohrer

E-mail: gabrielalvesbohrer@gmail.com

### Exercício Técnico

### APRESENTAÇÃO EM VÍDEO DA SOLUÇÃO

<https://youtu.be/3FTHcieyqfl>

### RESUMO DA SOLUÇÃO

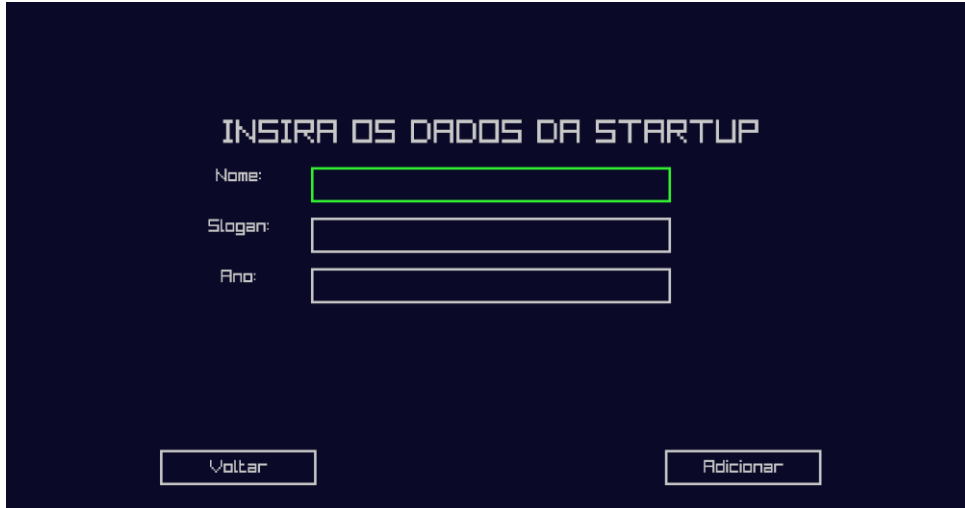
Minha solução consiste em um software desenvolvido integralmente em C++ e com a biblioteca Raylib, uma facilitadora na criação de interfaces gráficas em C. Todos os sistemas foram implementados sem a utilização de frameworks ou componentes prontos, de modo que todos os elementos visuais e estruturais foram desenvolvidos do zero. Isso inclui os componentes de interface do usuário (*UIObjects*), o sistema de gerenciamento de estado e demais artefatos que normalmente seriam abstraídos no uso de *toolkits* (como o GTK para C).

O software executa a simulação de batalhas entre startups, administrada por um usuário. A solução permite, então, a criação de Startups, a visualização das batalhas aleatoriamente geradas, a seleção de eventos específicos para qualificar as startups, a apresentação dos resultados finais de forma modular e detalhada, e a criação automática de startups-exemplo comandada pelo usuário, se este desejar.

A estrutura do programa é centralizada em uma única classe, **Manager**, que armazena as informações do torneio, de estados do programa e da lógica de controle. É criada uma instância global do Manager, que chama os métodos **Update()** e **Draw()** no loop principal. A função principal de Update() é selecionar o handler de estado correto. Em Draw(), todos os UIObjects do estado atual são desenhados na tela, assim como as mensagens de erro do usuário. Dentro de cada handler, a função de ordem maior **Handle\_UI** é chamada, possibilitando a identificação de com qual UIObject o usuário tentou interagir. Com base nisso, os botões permitem o usuário a trafegar por um fluxo de telas e a disparar ações específicas, como começar o torneio, salvar a avaliação de uma batalha ou ver os detalhes do relatório final.

## APRESENTE OS TESTES DAS FUNCIONALIDADES DESENVOLVIDAS

### FUNCIONALIDADE 1 – Cadastro de startups




INSIRA OS DADOS DA STARTUP

Nome:

Slogan:

Ano:

Tela de cadastro de startups. Apresenta três campos de prompt (nome, slogan e ano de fundação) que o usuário ativa posicionando o mouse dentro da "hitbox" de cada um. A ação de *hover* do mouse faz com que campos mudem de cor. Cada um deles possui regras de validação: o campo de nome faz a validação de duplicatas (se já existir outra startup cadastrada com o mesmo nome) e não permite o usuário cadastrar nomes muito grandes; o campo de slogan permite até 32 caracteres; e o campo de ano só aceita dígitos numéricos válidos (não é possível inserir valores negativos ou maiores do que o ano atual). Se dados inapropriados são inseridos, o sistema dispara uma mensagem de erro personalizada para o usuário quando ele tenta adicionar a startup. O usuário pode usar *backspace* para corrigir algum campo e, somente após todos os campos estarem preenchidos, que o botão "Adicionar" salva o cadastro e volta para o menu.



INSIRA OS DADOS DA STARTUP

Nome:

Slogan:

Ano:

Insira um ano válido!

## FUNCIONALIDADE 2 – Sorteio de batalhas e estrutura do torneio



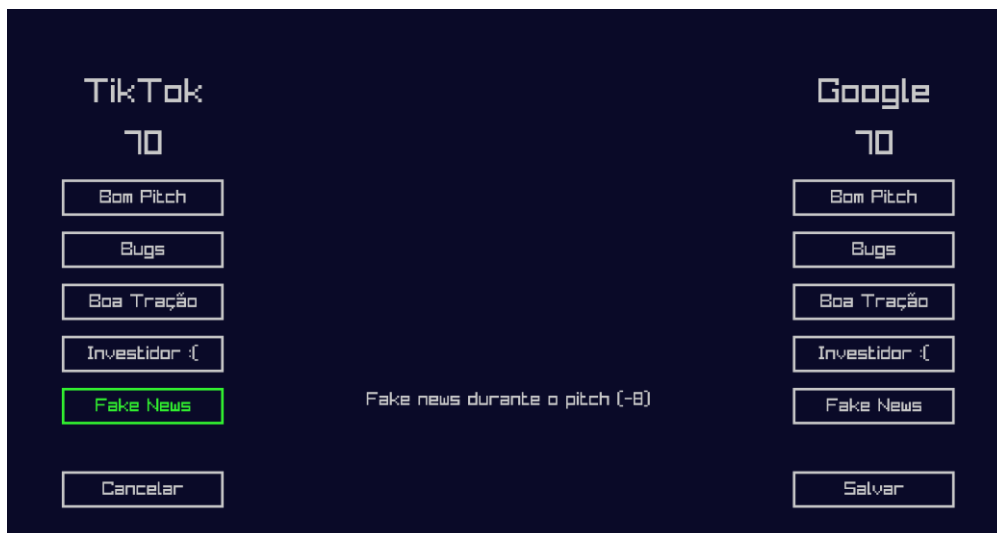
Assim que a quantidade permitida de startups é atendida, o usuário consegue iniciar um torneio a partir da tela inicial do sistema. Antes do sistema atualizar para a próxima tela, ele realiza a lógica de formação de batalhas. Sempre que uma etapa de batalhas termina, esta lógica é chamada. O algoritmo seleciona duas startups disponíveis (que ainda não foram desclassificadas) aleatoriamente e cria uma instância de batalha com ambas. Assim que não restar mais nenhuma startup disponível, as batalhas são alocadas para um vetor no torneio e o manager seleciona para qual tela irá baseado na quantidade total de batalhas criadas.

A classe de torneio, **Tournament**, tem os seguintes atributos: um vetor com os eventos de batalha registrados, um vetor de **StartupEntry**, que armazena as informações pertinentes da startup em relação ao torneio (pontuação, eventos, status, etc.), o vetor com as batalhas atuais e outros atributos de controle, como a startup campeã, a batalha atual cuja está sendo manipulada pelo usuário e quantidade máxima/mínima de startups permitidas no torneio.

## FUNCIONALIDADE 3 – Administração de batalhas individualmente



Na tela de administração do torneio, o usuário consegue selecionar qual batalha irá avaliar. Ao lado de cada botão, o sistema mostra as duas startups atribuídas à determinada batalha. Se o usuário clicar em “Cancelar”, o torneio atual é desfeito e o sistema retorna para a tela inicial, gerando uma nova chave de batalhas caso o usuário inicie o torneio novamente. A seguir, a tela de batalhas:



Quando o usuário seleciona uma das batalhas, o sistema transfere o estado de execução para a tela de batalhas; e, através do atributo de “batalha atual” de Tournament, o manager sabe quais informações de startup exibir. Aqui, o usuário visualiza os pontos atuais das startups e os botões da esquerda/direita, que atribuem os eventos às startups respectivas. Quando o usuário passa o mouse em cima de um dos botões de eventos, o sistema mostra uma breve descrição e quantos pontos o evento vale. O usuário consegue deixar ativado/desativado cada um dos botões, que atribuem corretamente os pontos em tempo real. Se estiver ativado o botão fica escuro, com uma aparência de pressionado. E, se clicado novamente, “desativa”, não contabilizando mais seus valores nos pontos.

Se o usuário clicar em “Cancelar”, todo o progresso da batalha será apagado e o sistema voltará para a seleção de batalhas em vigência. O botão “Salvar” confirma as atribuições feitas pelo usuário, finalizando a batalha. Caso ocorra o “shark fight”, o sistema exibe uma mensagem ao usuário, indicando que o desempate foi realizado.

#### FUNCIONALIDADE 4 – Avanço automático de fase



Ao finalizar uma batalha, o sistema trava o botão correspondente a ela, não permitindo o usuário a modificá-la mais, e evidencia que foi “concluída” com o vencedor ao lado. Quando a última batalha sem avaliação for finalizada, o sistema avança para a próxima etapa: todas as batalhas são desfeitas, e novas batalhas são formadas apenas com as startups classificadas. A tela então mostra as batalhas e botões disponíveis, tudo dinamicamente formatado com a quantidade correta de startups e batalhas das fases consequentes.

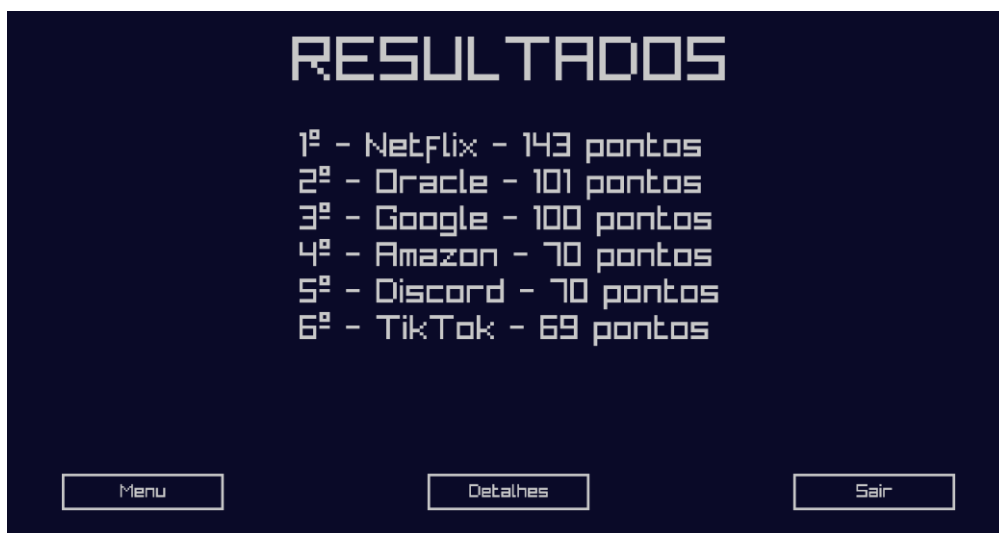
Quando um torneio é iniciado com 6 startups, a primeira etapa de batalhas termina com três delas classificadas. Isso exige um tratamento especial na geração aleatória de batalhas, pois sobraria uma startup sem dupla para batalhar. Por isso, criei uma nova regra para o sistema: as duas startups com PIOR qualificação (menos pontos) são selecionadas do vetor de startups, adicionando 1 ponto para uma delas de forma aleatória. A startup selecionada então é classificada, e a outra é desclassificada. Quando isso ocorre, o sistema deixa claro ao usuário esta decisão. Ao clicar no botão “Prosseguir”, ocorre a execução desta regra e uma próxima rodada de batalhas é criada para o usuário avaliar.



## FUNCIONALIDADE 5 – Relatórios e resultados



Quando a última batalha ocorre, deixando todas as startups desclassificadas com exceção de uma, o sistema elege a startup campeã. O nome, a quantidade total de pontos acumulados e o seu slogan são mostrados ao usuário. Ao clicar em “Resultados”, o sistema prossegue e exibe uma lista com todas as startups participantes, ordenadas pelo ranque no torneio. Caso duas ou mais startups terminem sua participação no torneio com a mesma quantidade de pontos, o sistema realiza o desempate pelo ano de fundação da startup, onde startups mais antigas possuem prioridade.



Se o usuário clicar no botão “Detalhes”, poderá visualizar uma relação detalhada de cada uma das startups participantes. A nova tela exibirá quantas vezes cada um dos eventos de batalha foram atribuídos a cada uma das startups através das rodadas:

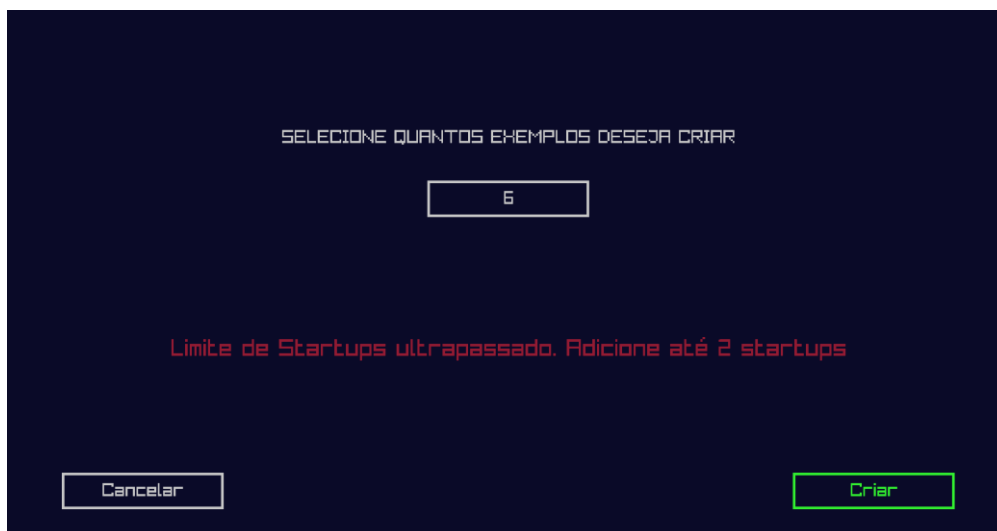


Se o usuário desejar, pode sair do programa, finalizando a execução do sistema, ou voltar para o Menu, podendo iniciar um novo torneio com as mesmas startups já cadastradas.

#### FUNCIONALIDADE 6 – Criação Automática de Startups (FEATURE EXTRA)



A funcionalidade extra adicionada ao sistema foi a possibilidade de o usuário selecionar uma quantidade específica de startups, previamente criadas, ao torneio. Escolhi esta funcionalidade pois foi algo que implementei no back-end, que me pouparia tempo e facilitaria minhas rodadas de testes. Assim, esta funcionalidade também acaba ajudando os usuários, caso estejam sem ideias ou não queiram cadastrar suas próprias startups. Na tela de cadastro de startups, basta o usuário clicar no botão “Exemplos”.



SELECIONE QUANTOS EXEMPLOS DESEJA CRIAR

6

Limite de Startups ultrapassado. Adicione até 2 startups

Cancelar Criar

O botão com um número indica quantas startups serão criadas e adicionadas ao torneio. O botão é rotativo; então o usuário pode clicar nele até alcançar a quantidade que deseja. Se o usuário tentar adicionar um número de startups que resulte em um número de cadastros totais maior do que o limite do torneio, a ação é barrada e uma mensagem de erro é exibida para o usuário.



## AUTOAVALIAÇÃO

Você concluiu a implementação de 100% das funcionalidades solicitadas?

(X) Sim      ( ) Não

Para as 6 principais funcionalidades solicitadas, como você avalia a sua solução?

	Inexistente/ Insuficiente	Pouco satisfeito(a)	Satisfeito(a)	Muito satisfeito(a)
Funcionalidade 1			X	
Funcionalidade 2				X
Funcionalidade 3				X
Funcionalidade 4				X
Funcionalidade 5				X
Funcionalidade 6				X

## Principais dificuldades

Minhas principais dificuldades ocorreram na integração dos componentes de interface (UIObjects) com as entidades do torneio. Devido a minha solução encontrada para o sistema interativo e com interface gráfica, todos os UIObjects são criados e armazenados nos estados de execução respectivos na inicialização do manager; logo, possuem uma natureza estática. Porém, todos os outros artefatos, em especial as informações de batalhas e pontuação das startups, assumem uma natureza dinâmica. Por causa disso, funções de “print” foram criadas para imprimir diretamente na tela as informações, ao invés de UIObjects serem devidamente criados em cada tela e atualizados conforme ações dos usuários.

Por exemplo, nos métodos `SelectWinner()` ou `ResetBattle()`: tais métodos precisam lidar tanto com objetos de interface quanto com variáveis e instâncias das entidades, o que gera uma complexidade exagerada. Além disso, o uso de cópias, referências e ponteiros para atualizar os dados corretamente nestes métodos me fez quebrar a cabeça.

Outro ponto que vale evidenciar é no uso da Raylib e a dificuldade que tive com seu Makefile. Este arquivo serve para criar um ambiente de desenvolvimento apropriado no VSCode, incluindo caminhos necessários para o uso do compilador e das header files. Contudo, mesmo tendo passado horas estudando o arquivo, não alcancei o completo entendimento do que todas as suas linhas executam. Por causa disso, alterações no software que eu desejava fazer não foram possíveis, como o empacotamento de todos os objetos (.o) em um único executável ou a mudança do nome do programa principal.

## Auto avaliação sobre o desempenho Geral

Atribuiria uma nota de 9.0/10.0 à solução desenvolvida.

Apesar do software conter todas as funcionalidades requisitadas, acho que a interface possui alguns pontos a desejar. O sistema de mensagens de erro ao usuário (PopUpMessage), por exemplo, não foi completamente desenvolvido, não interagindo de maneira satisfatória com o restante da lógica dos estados de execução. Outro ponto é a funcionalidade de cadastro de startups. Os campos de prompt são bastante rudimentares, apresentando deficiências em termos de usabilidade: os campos de entrada não permitem o uso fluido do *backspace* e não mostram um cursor piscante dentro do campo selecionado — aliás, o sistema força o usuário a manter o mouse sobre o campo desejado para digitar, algo nada natural. Por fim, ainda acredito que toda a arquitetura e lógica de controle e modelagem de componentes de interface/estados poderia ser melhorada, uma vez que essas foram minhas ideias, baseadas nas minhas experiências em trabalhos acadêmicos, profissionais e pessoais.

Sobre minha organização e tecnologias utilizadas, vejo como vários conteúdos aprendidos em meu curso de graduação se fizeram úteis neste projeto. Desde o aprendizado de máquinas de estado e autômatos, que me auxiliaram na concepção do fluxo de telas, do uso do GIT para integração contínua, documentação e controle de todo o projeto, dos fundamentos de orientação à objetos e técnicas de construção de programas (garantindo que meu código ficasse o mais coeso, legível e desacoplado possível), até o conhecimento de propriedades avançadas em linguagens de programação, como funções anônimas, operadores ternários ou estruturas de dados complexas como mapas e tuplas. Sinto que nunca havia posto em prática tantos conceitos quanto agora! Em suma, fiquei muito satisfeito e surpreso com meus resultados, comemorando cada método que compilava de primeira (mesmo que não executasse *exatamente* o que eu queria).