PROGRAMAÇÃO ORIENTADA A OBJETOS

JOGO
C++ E RAYLIB

"LOST ASTRONAUT"

Tópicos

- RAYLIB
- O JOGO
- A INSPIRAÇÃO
- O DESENVOLVIMENTO
- DESAFIOS
- CONCLUSÃO

RAYLIB

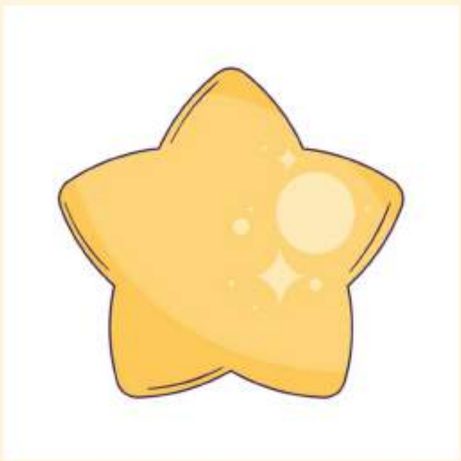
A Raylib é um biblioteca, construída em C, a qual possui código aberto e funciona em diversas plataformas.

Se destaca pela compatibilidade com diversas plataformas, IDE's, facilidade de uso (comandos simples e claros), dentre outras características.

Além disso, possui uma comunidade de desenvolvedores ativa, código aberto e diversos projetos exemplos e tutoriais em seu site oficial, no Github do seu criador, youtube e etc

O JOGO

“Lost Astronaut” é simples: consiste em um astronauta em queda livre, que deve desviar dos asteroides e coletar estrelas, até chegar a seu roover que está sobre o solo aguardando por ele.



Inspiração

O conceito do jogo foi inspirado na segunda parte da fase Sunken Ghost Ship do Super Mario World 2, no entanto, apresenta uma abordagem mais simples, sem a variação de powerups e inimigos disponíveis no origianl;

Inspiração

https://www.youtube.com/watch?v=Xf2En3IWWMs

O DESENVOLVIMENTO

ETAPA 1 - O PLANEJAMENTO

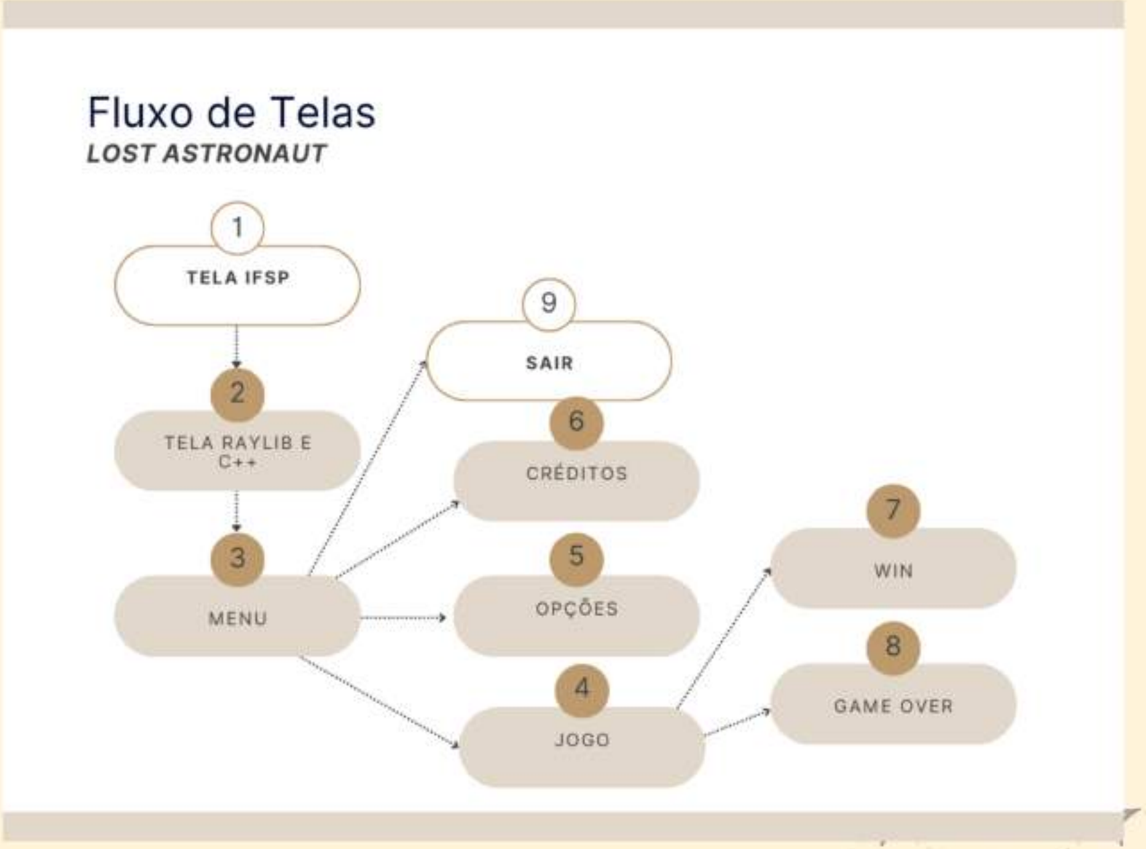
A primeira etapa consistiu em organizar a ideia, para isso, foi usado um modelo simplificado de um GDD - Game Design Document, ou seja, um documento que descreve como o jogo será, o que vai ter, qual o objetivo, os inimigos, os powerups, títulos, etc.

O DESENVOLVIMENTO

ETAPA 1 - O PLANEJAMENTO

Nessa primeira etapa também foi definido o “Fluxo de Telas” do JOGO, o que facilitou pensar em como seria a lógica e a ligação de entre telas.

A primeira versão foi rascunhada em papel e a versão final, num aplicativo online (LUCIDCHART)



O DESENVOLVIMENTO

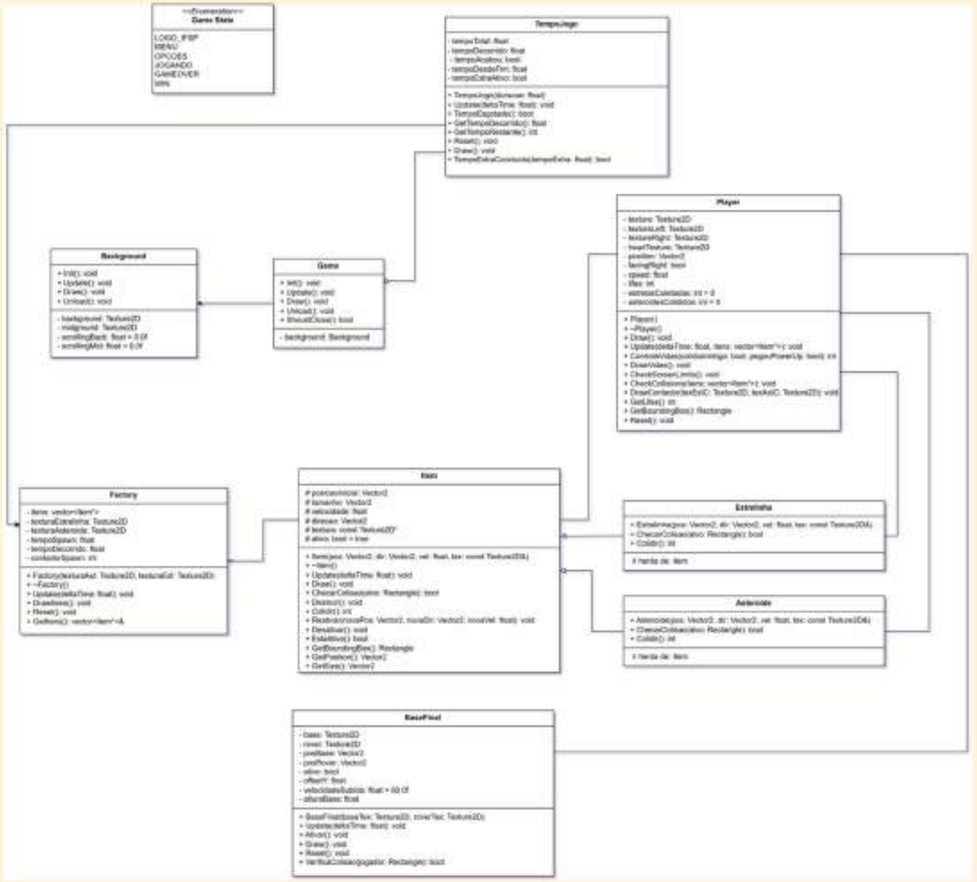
ETAPA 2 - A PROGRAMAÇÃO ORIENTADA A OBJETOS

Já na segunda etapa, com o GDD em mãos e algumas ideias de design mais definidas, foi hora de modelar a 1º versão do diagrama de classes UML, avaliar quais seriam as classes, seus métodos e as relações.

Para essa etapa foi usado o Drawio, para fazer o diagrama inicial, no entanto durante outras etapas do desenvolvimento esse diagrama sofreu mudanças.

O DESENVOLVIMENTO

ETAPA 2 - A PROGRAMAÇÃO ORIENTADA A OBJETOS



O DESENVOLVIMENTO

ETAPA 3 - RAYLIB - INSTALAÇÃO, TUTORIAIS E WIKI

A terceira fase se uniu a quarta, já nessa foi o momento de instalar e estudar o básico da biblioteca, os comandos de criação da tela, como funciona a estrutura e também os comandos mais básicos.

Foi nessa etapa também, que para o desenvolvimento foi decidido que seria utilizado o CodeBlocks e não o Visual Studio (como previsto no GDD), por questões de facilidade.

O DESENVOLVIMENTO

ETAPA 3 - RAYLIB - INSTALAÇÃO, TUTORIAIS E WIKI

Ainda, foi a fase de ver os tutorias básicos para conhecer a biblioteca, nesse caso, a Raylib conta com vários exemplos disponíveis em seu site, além de diversas séries de tutoriais no youtube.

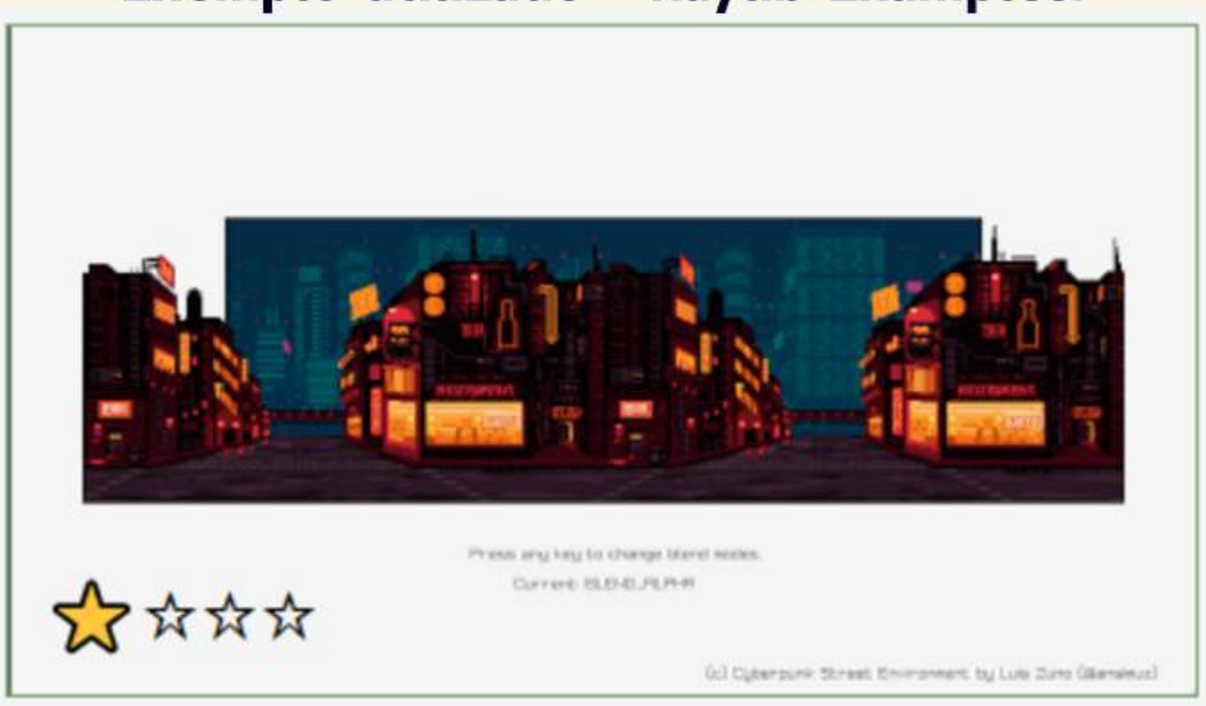
Um destaque para os vídeos:
“Get Started in raylib in 20 minutes!“ e **Getting Started with Raylib - C Tutorial.**
Ambos simples que dão uma boa visão inicial da biblioteca.

O DESENVOLVIMENTO

ETAPA 4 - CODIFICAÇÃO DO JOGO

A etapa 4 começou com a construção do background, um modelo com movimentação, que tem tutorial na aba examples do site da Raylib, mas invertido em relação ao tutorial - tela com movimento horizontal no tutorial e vertical no jogo.

Exemplo utilizado - Raylib Examples:



O DESENVOLVIMENTO

ETAPA 4 - CODIFICAÇÃO DO JOGO

Depois de criar o background foram adicionadas as vidas, a princípio 3 corações, embora renderizados, nessa 1º parte, eles não estavam integrados com nada, mas serviram de suporte para redimensionar os outros assets do jogo.

Asset do Coração do Jogo:

O DESENVOLVIMENTO

ETAPA 4 - CODIFICAÇÃO DO JOGO

Antes de continuar o desenvolvimento do jogo em si, foi hora de montar o fluxo de telas inicial e menu, utilizando uma lógica de switch para as escolhas do menu e um ENUM com “Estados de Jogo” para controlar as telas, junto com um controle de tempo simples.

Vale lembrar, a cheatSheet no site oficial da Raylib contém os comandos da linguagem com uma breve explicação, isso com auxílio de alguns vídeos foram a base para construção desse modelo!

O DESENVOLVIMENTO

ETAPA 4 - CODIFICAÇÃO DO JOGO

Com o menu previamente criado, textos simples nas outras páginas e o botão de sair configurado. Foi feita a criação da classe Player, com o Astronauta, protagonista no jogo.

A princípio com movimentação para todos as direções e sem a configuração de colisão.

Lembrando que o background do jogo já estava implementado, então o personagem já “flutuava” sem rumo nessa etapa.

O DESENVOLVIMENTO

ETAPA 4 - CODIFICAÇÃO DO JOGO

Depois do astronauta implementado e se movimentando livremente pela tela, veio um grande desafio.

Criar Estrelas e Asteroides, que surgem em lugares randômicos na tela e se movimentam de forma aleatória.

Aqui veio a segunda grande mudança no diagrama de classes. (A primeira foi o enum para controle de telas)

O DESENVOLVIMENTO

ETAPA 4 - CODIFICAÇÃO DO JOGO

Para fazer isso funcionar foi criada a Classe FACTORY, que gerava ITENS (a superclasse de Asteroide e Estrelinha) que tem como função criar e fazer os itens se movimentar em uma proporção pré-definida (9:1 - A cada 9 asteroides = 1 estrelinha)

Foi a etapa que levou mais tempo, a construção das duas classes, a eliminação de bugs, erros de lógica, tutoriais e outros detalhes levou cerca de 3 semanas!

O DESENVOLVIMENTO

ETAPA 4 - CODIFICAÇÃO DO JOGO

Com o spawn de itens funcionando corretamente foi hora de trabalhar as colisões, criar as caixas de colisão com a ajuda da Raylib é simples, gerenciar as colisões foi uma tarefa mais complexa.

O erro de “Perda de 2 Vidas” foi o bug mais difícil de descobrir no jogo, graças ao posicionamento errado do método de destruição do asteroide, o jogador perdeu duas vidas em colisões durante 1 semana.

Agradecimentos especiais ao fórum Game Maker, que tem muitas perguntas/respostas que ajudaram a solucionar o problema

O DESENVOLVIMENTO

ETAPA 4 - CODIFICAÇÃO DO JOGO

Com o spawn de itens funcionando e graças ao problema das 2 vidas, foi implementado a contagem de itens, um jeito de gerar possíveis pontuações e controlar melhor o jogo. Depois disso, foi instaurado o temporizador, com 10 segundos p testes, com o final do jogo sendo chamado pela mudança de estado (Game::Win) se o jogador sobrevivesse.

Junto com o Game::Win foi implementado o Game::GameOver, assim o jogo sempre tinha um final.

O DESENVOLVIMENTO

ETAPA 4 - CODIFICAÇÃO DO JOGO

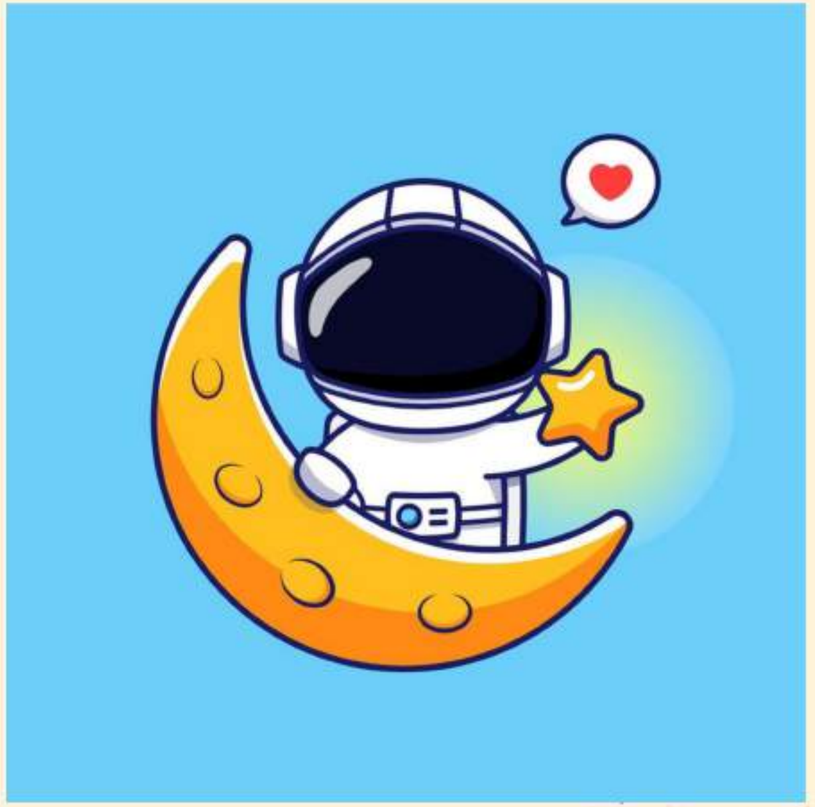
Só depois de ter as telas finais a base foi implementada, como são duas texturas diferentes, (base e rover), fazer as duas conversarem e “subirem” a tela juntas, como se o astronauta fosse se aproximando do solo, foi a parte mais difícil, no entanto, foi utilizado apenas uma subtração entre os as posições y para que as duas texturas funcionassem como uma única!

A inspiração para o rover veio do livro/filme “Perdido em Marte”, já que o veículo foi o lar e melhor amigo do astronauta na trama.

O DESENVOLVIMENTO

ETAPA 4 - CODIFICAÇÃO DO JOGO

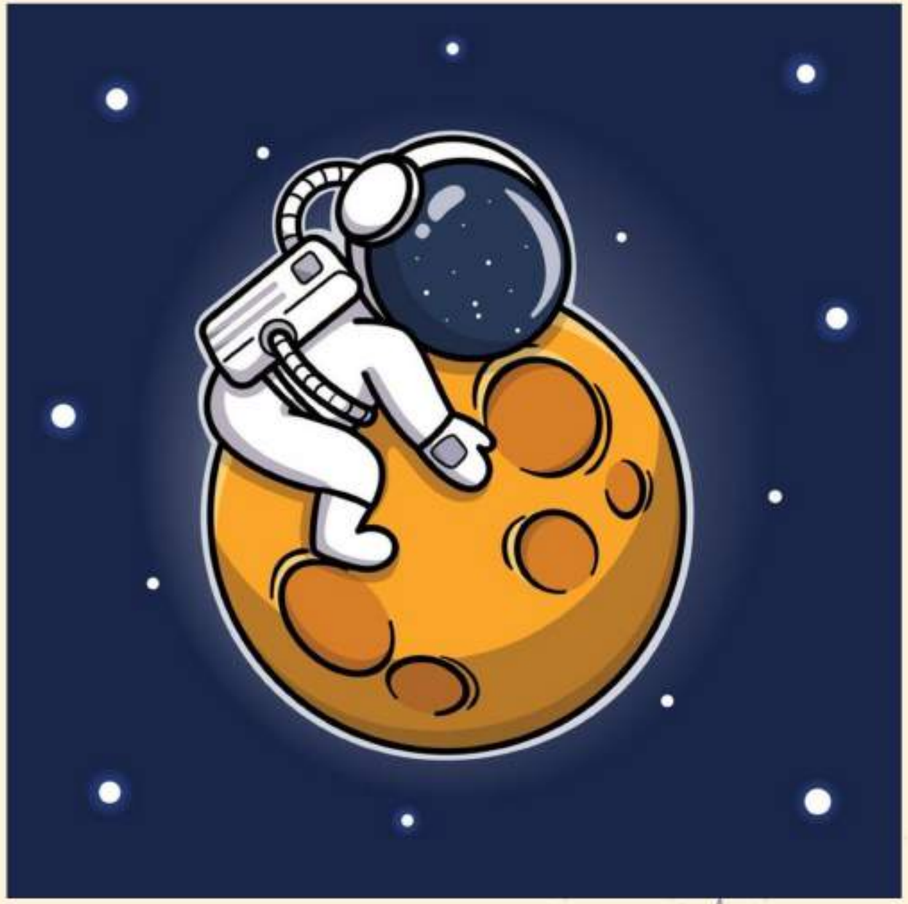
Com isso construído o jogo finalmente chegou a uma versão “Funcional”!



O DESENVOLVIMENTO

ETAPA 5 - AJUSTES

Com o “esqueleto” pronto, foi hora de começar os ajustes, verificar se a velocidade dos itens e dos players, ajustar a quantidade de vidas, a proporção de itens (a principio era 15: 2), dentre outros detalhes - Nessa etapa foram descobertos vários bugs.



O DESENVOLVIMENTO

ETAPA 6 - ASSETS DE SONS, MÚSICAS E FONTES

Quase na última etapa foram feitas as inserções de som, música e novas fontes para o projeto. Uma pequena na mudança no comando Draw e o acréscimo de algumas linhas de código garantiram a música e os sons funcionando.

O DESENVOLVIMENTO

ETAPA 7 - LOOP DO MENU

Com tudo referente ao jogo inserido, foi hora de acrescentar o loop que permite o retorno ao menu e uma nova partida.

A última mudança nas classes foi feita aqui, com a criação dos métodos resets para iniciar as partidas de novo.

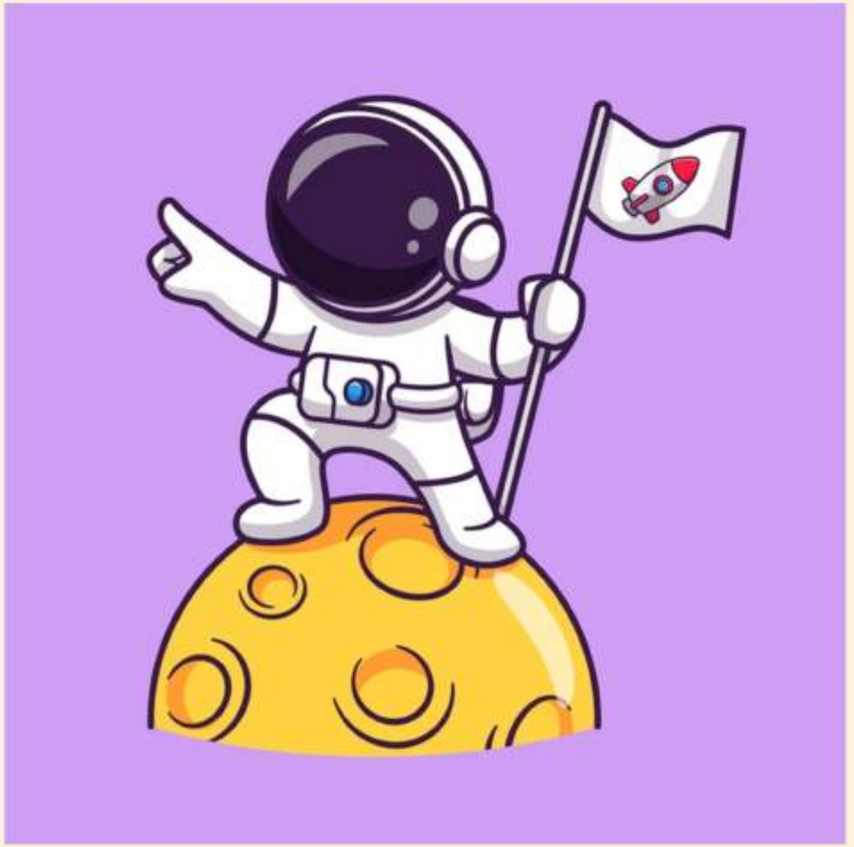
O DESENVOLVIMENTO

ETAPA 8 - SPLASH DE VITÓRIA, GAME OVER

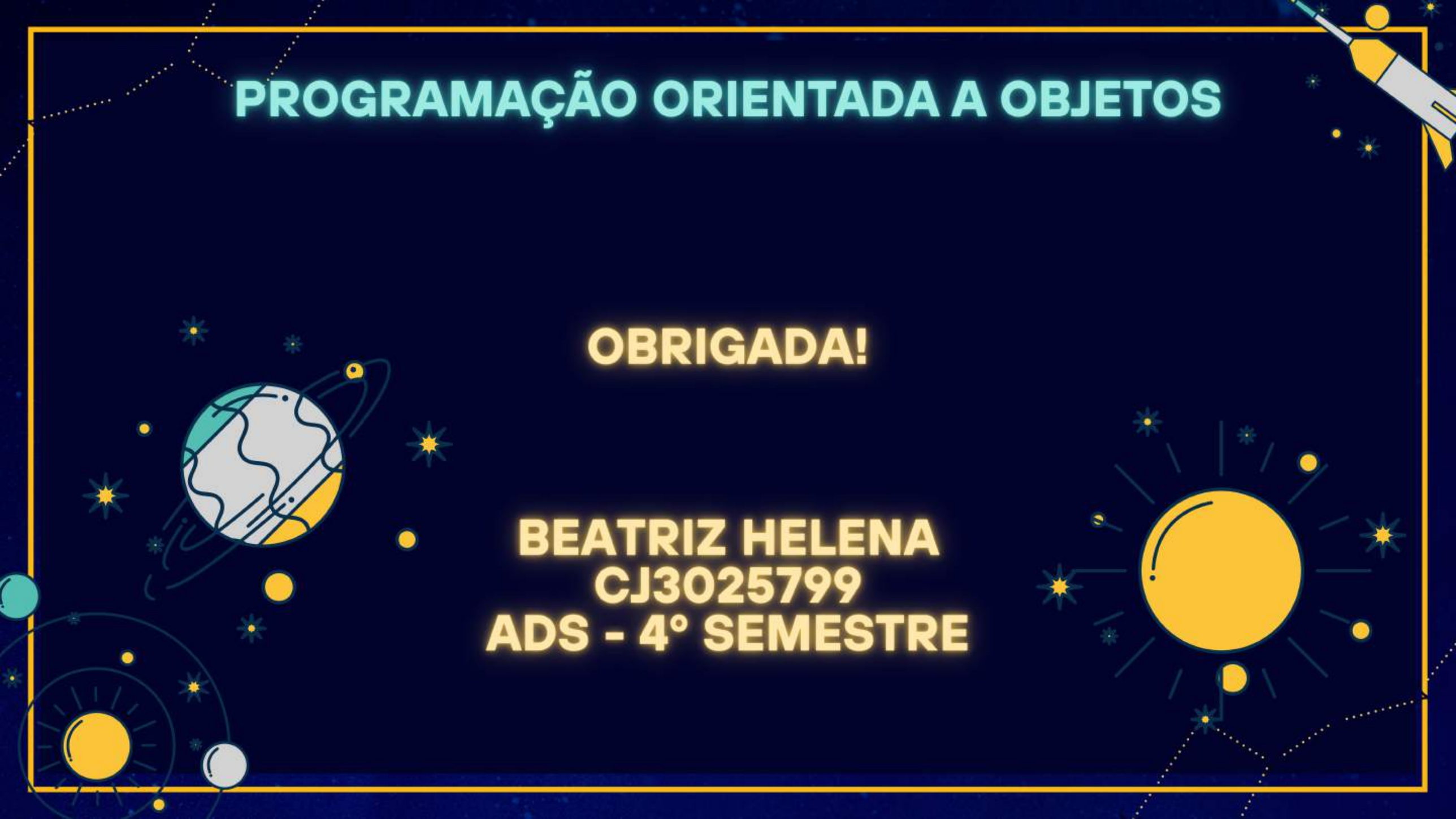
Por último, as funções que exibiam as telas de vitória (WIN) e GameOver no jogo foram adicionadas. Mudando pouca coisa nas funções que chamam os estados de jogo.

CONCLUSÃO

A construção de um jogo utilizando Programação Orientada a Objetos e uma biblioteca, que a princípio, era desconhecida ajudou a fixar conceitos, como a Raylib, ajudou a fixar os conceitos aprendidos em aula e até compreender melhor outros. Os desafios durante o desenvolvimento, embora obstáculos, ajudaram também a entender melhor como as classes interagem, por meio dos itens do jogo.



PROGRAMAÇÃO ORIENTADA A OBJETOS



OBRIGADA!

BEATRIZ HELENA
CJ3025799
ADS - 4° SEMESTRE