**Universidade Federal de Pernambuco**

**Centro de Informática**

Professores:

Filipe Calegário, Sérgio Soares, Ricardo Massa

LimoRaCIn

Sistema interativo em Python

Equipe 7:

Arthur Conegundes, Breno Santos, Ian Felipe, Pedro Souza, Rodrigo Castro, Wilson Freitas

INTRODUÇÃO

A ideia do projeto consiste em um sistema interativo criado em Python, que estrutura um jogo simples em que o usuário controla um carro e interage com outros objetos que aparecem na tela, seja desviando ou coletando-os.

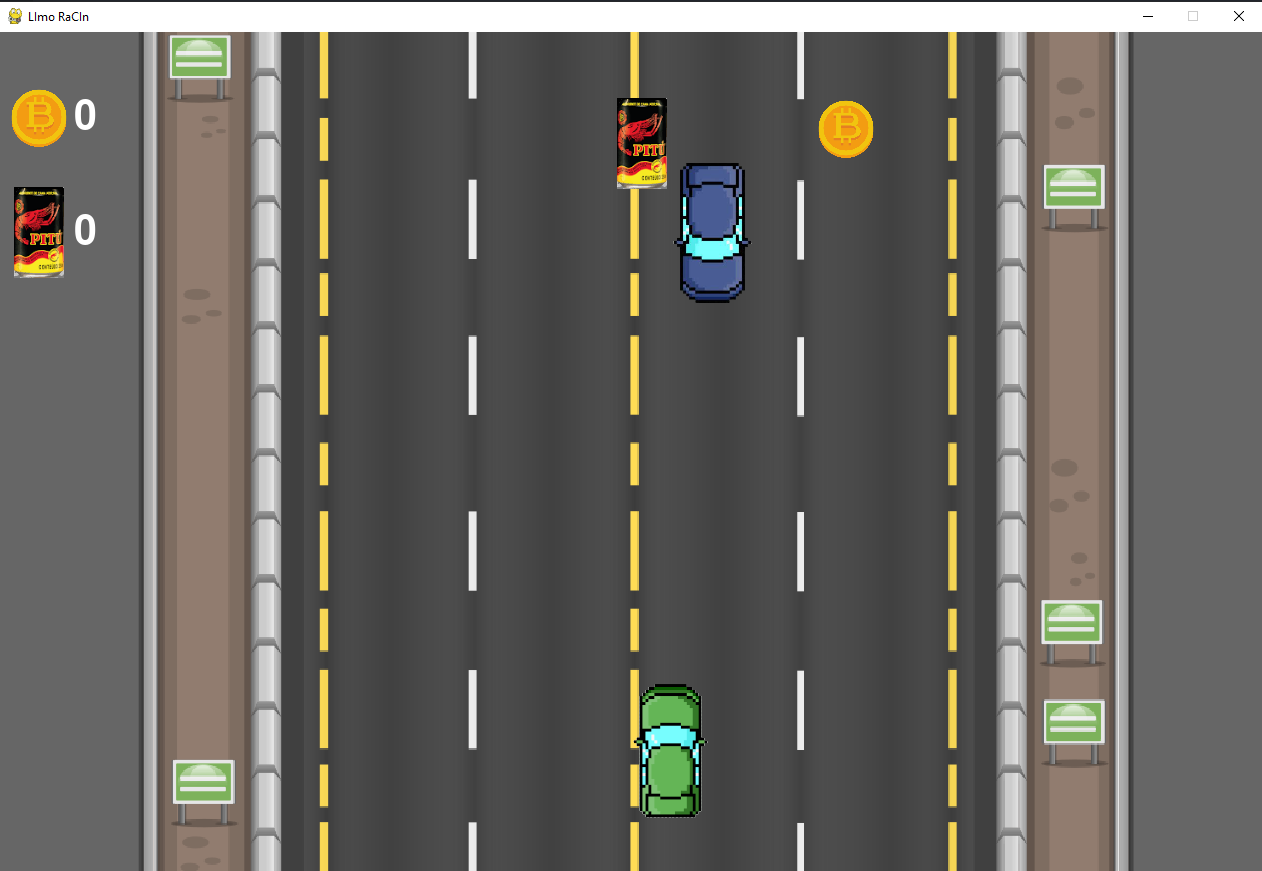
Ao iniciar o programa, é exibido um menu interativo, contendo o nome do jogo (LimoRacin) e opções de “Start” e “Quit”.



O menu recebe como input o click do mouse, se o usuário clicar no botão referente ao comando “Quit” o programa é encerrado e a janela do jogo é fechada.

Para cliques fora dos botões, o programa não retorna nada.

Caso o usuário clique no botão referente ao comando “Start” o código referente ao jogo em si é executado e o usuário começa sua interação com o programa.



Aqui, a interação do usuário com o programa se dá pelo pressionamento das teclas ‘W, A, S, D’ que movem o carro verde nos eixos horizontal e vertical pela tela.

Os objetos interativos são 3: Um carro azul, uma lata e uma moeda. Que se comportam da seguinte forma:

1 - O contato do carro azul gera uma colisão, que significa o fim do jogo, fechando a janela e encerrando o programa.

2 - O contato com a lata atualizado o contador no lado esquerdo da tela e gera um aumento de velocidade nos objetos

3 - O contato com a moeda atualiza o contador no lado esquerdo atuando como um sistema de pontuação.

RELATÓRIO DO PROJETO

## Link para download do LimoRaCIn:

O seguinte link leva a um repositório do GitHub que contém os arquivos necessários para rodar o jogo:

## A organização do código:

Durante o desenvolvimento adotamos diversas abordagens de administração de tempo e de otimização do trabalho em equipe, valendo destacar a adoção de metodologias ágeis como dailies e sprints semanais. A organização do código foi se adaptando a essa estratégia, sendo primeiro criado sem o OOP, para entendermos e praticarmos com o pygame. Nas sprints seguintes, optamos por tornar o código 100% orientado a objetos, separando ele em 4 arquivos principais: Variables, responsável por inicializar as variáveis; gameEngine, que contém a lógica (‘engrenagem’) do jogo; main, que inicializa, finalmente, uma classe definindo os atributos do objeto e criando um método de inicialização do jogo(chamando gameEngine) e, por último, a run.py, que como o nome sugere ‘roda’ o jogo, ela cria um objeto que, por meio do polimorfismo, tem embutida as propriedades da classe em main e, através do super, cria uma função que utiliza da lógica do jogo agora modificada pela classe e restrita a seus atributos. Ademais, há também a exibição gráfica do jogo na pasta run.py, que ao mudar o estado(clicar no botão de iniciar) chama o método final do objeto e, ao ‘perder’ o jogo, faz a recursividade nesse método'.

## Ferramentas e recursos utilizados:

WhatsApp - O grupo utilizou o Whatsapp como canal de comunicação primário, para marcar reuniões, tirar dúvidas pontuais e acertar pequenos ajustes.

Discord - Foi o canal de comunicação principal, onde as reuniões aconteciam e onde o trabalho se desenvolveu. O grupo se utilizou das ferramentas do Discord para realizar o projeto, principalmente por meio de compartilhamento de telas, podendo discutir e avançar no projeto de forma conjunta.

Git e GitHub - Essencial para o desenvolvimento do projeto, foi no GitHub que colocamos os arquivos necessários para o desenvolvimento do projeto, e por meio do Git, foi possível ir atualizando os arquivos a cada implementação, para que os membros que fossem trabalhar posteriormente no projeto já tivesses no repositório a última versão do que foi feito.

Pygame - Biblioteca do python utilizada fundamentalmente para a criação do jogo, atuando desde a criação da janela em que o jogo é apresentado até a reprodução dos sons do jogo e na execução de sua lógica.

Random - Biblioteca do python utilizada para obter valores aleatórios, utilizados para gerar os objetos interativos em lugares variados na tela, adicionando dinamicidade ao jogo.

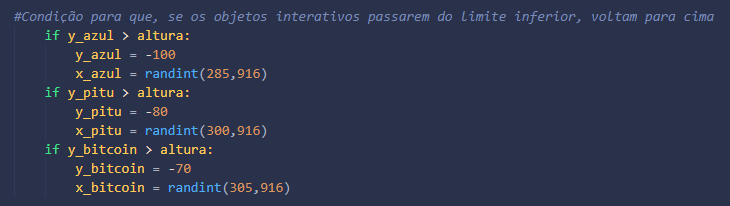
## Divisão de trabalho dentro do grupo:

Realizamos várias reuniões para dividir as tarefas e dividimos o grupo de forma que uma parte ficaria responsável por trabalhar com os conceitos de POO (programação orientada a objeto) e outras pelas partes conceituais estéticas e sonoras do jogo , sendo a primeira parte do código (sem POO) feita coletivamente por todos os integrantes .

## Conceitos apresentados durante a disciplina e utilizados no projeto:

Foram utilizados diversos conceitos apresentados na disciplina, alguns deles foram:

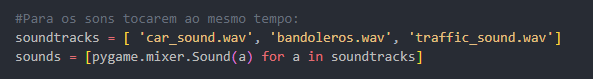
Condicionais: Estruturas de condição foram utilizadas para fundamentar a lógica do jogo e determinar eventos específicos. Exemplo: No corpo do código referente a lógica, foi criada uma condição para que, se a posição vertical do objeto ultrapassasse a altura da tela, esse objeto deveria retornar ao topo, em um ponto x aleatório.



**Laços de repetição:** Foram utilizados laços de repetição para realizar eventos repetidos, e fazê-los parar por meio de uma condição de parada.



**Listas :** Foram utilizadas listas em conjunto com laços para facilitar o trabalho de inserção de sons e trilhas sonoras dentro do jogo.



**Qual foi o maior erro cometido durante o projeto? Como vocês lidaram com ele?**

Na medida em que o código escalava, sofremos por não utilizar boas práticas de arquitetura de código, se debatendo com o erro de criar funções muito grandes e responsáveis por quase toda a lógica.

**Qual foi o maior desafio enfrentado durante o projeto? Como vocês lidaram com ele?**

Adaptar-se a tecnologias novas, com destaque para o git. Com muita pesquisa e ajuda de membros mais experientes, fomos nos familiarizando com o processo de versionamento. Como também com os comandos próprios das bibliotecas utilizadas no projeto.

**Quais as lições aprendidas durante o projeto?**

Com o processo de criação do projeto foi possível aprender a trabalhar em conjunto, pois nós nos reunimos para construir partes do código e assim evoluí-lo com o passar do tempo.