Projeto Final

Aluno: Bruna Lima Farias

Quis Star Wars

Propósito:

O projeto é uma combinação de um sistema de cadastro de planetas, personagens e eventos do universo Star Wars, um sistema de perguntas e respostas que testa o conhecimento dos jogadores sobre esses planetas e um mapa em 2d da galáxia do universo dos filmes

Funcionamento:

- **Cadastro de Planetas, Personagens e Eventos:**

  O usuário pode cadastrar informações sobre planetas do universo Star Wars, incluindo o nome do planeta, personagens que nasceram nele e eventos relacionados a esse planeta

  Isso é feito usando a opção "1. Cadastrar Planeta" no menu.

  Os planetas cadastrados são armazenados em um dicionário chamado cadastro_planetas, neste já possuindo dados pré cadastrados

- **Cadastro de Perguntas sobre Planetas:**

  Os usuários podem cadastrar perguntas relacionadas a um planeta específico. Cada pergunta deve incluir a pergunta, as opções de respostas, a resposta correta e uma explicação

  Isso é feito usando a opção "2. Cadastrar Pergunta para Planeta" no menu

  As perguntas são associadas a um planeta específico e armazenadas nos dados do planeta no dicionário cadastrar_planeta_pergunta

- **Lista Planetas:**

  Exibe uma lista de planetas que estão cadastrados quando selecionado a opção "3. Lista Planetas" no menu

- **Quiz:**

  O jogador pode iniciar o quiz relacionado a um planeta específico, respondendo a perguntas sobre esse planeta, selecionando a opção "4. Iniciar Quiz"

  Os pontos são calculados com base nas respostas corretas e registrados para o jogador

  Os resultados do quiz são armazenados no dicionário cadastro_planetas com o nome do jogador

- **Galáxia 2D**

  A opção "5. Mostrar Galaxia 2D" no menu uma representação simplificada da galáxia do universo de Star Wars

- **Pontuação**

  Depois de terminar de cadastrar planetas, perguntas e realizar o quiz, o jogador pode ver sua pontuação final

  A opção "6. Encerrar" no menu encerra o programa e exibe as pontuações finais de todos os jogadores listados

Evidências:

- O código contém funções para validar entradas não vazias, cadastrar planetas, cadastrar perguntas

```python
def entrada_nao_vazia(mensagem):
    while True:
        entrada = input(mensagem).strip()
        if entrada:
            return entrada
        else:
            print("Por favor, insira uma entrada valida")
```

- O quiz exibe perguntas, opções resposta e uma explicação sobre as respostas

```python
for pergunta_data in perguntas:
    print(pergunta_data["pergunta"])
    for resposta in pergunta_data["respostas"]:
        print(resposta)

    resposta = input("Sua resposta (a, b ou c): ")

    if resposta == pergunta_data["resposta_correta"]:
        print("Correto! Você ganhou um ponto.")
        pontuacao += 1
    else:
        print("Errado. A resposta correta é:", pergunta_data["resposta_correta"])
        print(pergunta_data["explicacao"])
```

Testes:

Cadastro de Planetas:

Adicionado da maneira correta

```
Menu:
1. Cadastrar Planeta
2. Cadastrar Pergunta
3. Lista de Planetas
4. Iniciar Quiz
5. Mostrar Galaxia 2D
6. Encerrar
Escolha uma opção: 1
Digite o nome do planeta: Naboo
Digite os personagens que nasceram neste planeta: Padme Amidala
Digite os eventos relacionados a este planeta: Nascimento do Chanceler Palpatine
```

Input (“ ”) sem nada dá como entrada invalida

```
Menu:
1. Cadastrar Planeta
2. Cadastrar Pergunta
3. Lista de Planetas
4. Iniciar Quiz
5. Mostrar Galaxia 2D
6. Encerrar
Escolha uma opção: 1
Digite o nome do planeta:
Por favor, insira uma entrada valida
```

Cadastro de Perguntas:

Adicionado da maneira correta

```
Escolha uma opção: 2
Digite o nome do planeta para cadastrar a pergunta: Naboo
Digite a pergunta: Qual jedi foi morto durante o ataque separatista em Naboo?
Digite a alternativa (a): Qui-Gon
Digite a alternativa (b): Obi-Wan
Digite a alternativa (c): Anakin
Digite a resposta correta (a, b, c): a
Digite uma explicação da resposta: Qui-Gon foi morto durante a luta comtra Darth Maul
```

Quiz:

Mostrado da maneira correta, com as perguntas e validando a resposta

```
Escolha uma opção: 4
Digite o nome do planeta para o quiz: Naboo
Digite o nome do jogador: Bruna
Quiz sobre o planeta Naboo para Bruna:

Qual jedi foi morto durante o ataque separatista em Naboo?
Qui-Gon
Obi-Wan
Anakin
Sua resposta (a, b ou c): a
Correto! Você ganhou um ponto.
```

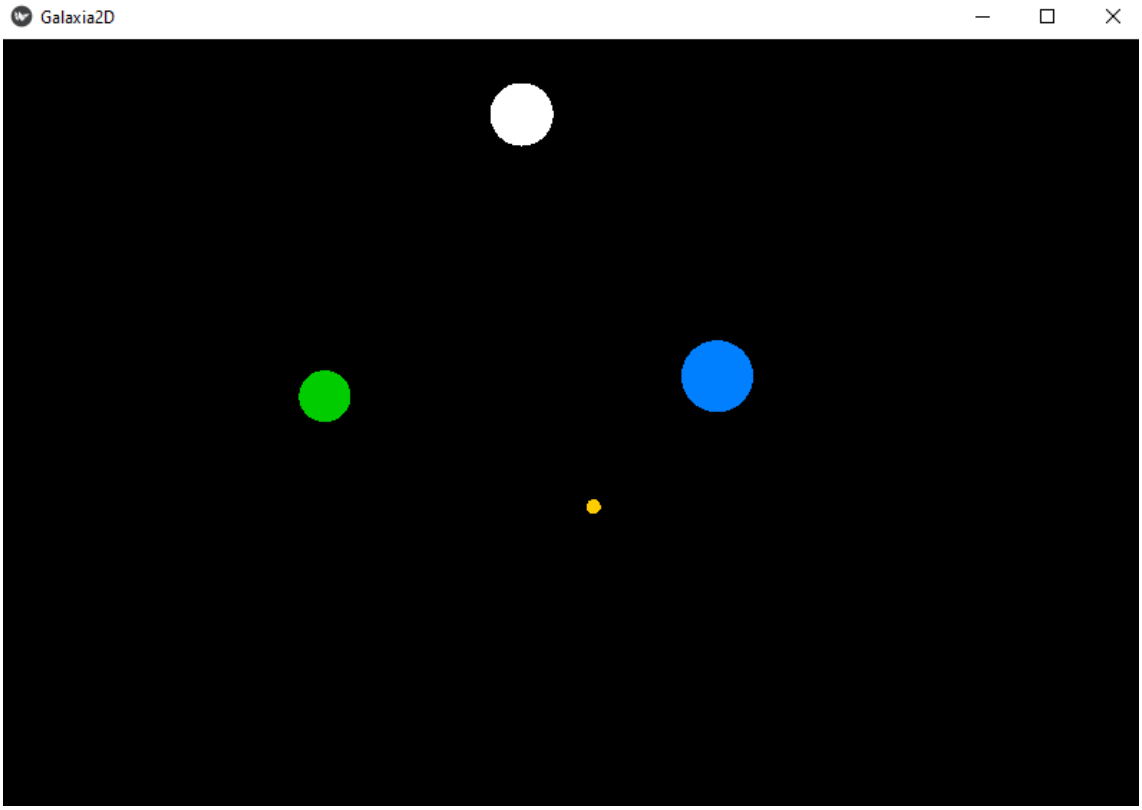
Lista de Planetas:

É mostrada a lista com os planetas cadastrados

```
Escolha uma opção: 3

Lista de Planetas:

- Tatooine

- Endor
```

Mostrar Galáxia 2D:

```
Escolha uma opção: 5
[INFO   ] [Window      ] Provider: sdl2
[INFO   ] [GL          ] Using the "OpenGL" graphics system
[INFO   ] [GL          ] GLEW initialization succeeded
[INFO   ] [GL          ] Backend used <glew>
[INFO   ] [GL          ] OpenGL version <b'4.5.13542 Compatibility Profile Context 25.20.14012.9002'>
[INFO   ] [GL          ] OpenGL vendor <b'ATI Technologies Inc.'>
[INFO   ] [GL          ] OpenGL renderer <b'AMD Radeon(TM) Vega 10 Mobile Graphics'>
[INFO   ] [GL          ] OpenGL parsed version: 4, 5
[INFO   ] [GL          ] Shading version <b'4.50'>
[INFO   ] [GL          ] Texture max size <16384>
[INFO   ] [GL          ] Texture max units <32>
[INFO   ] [Window      ] auto add sdl2 input provider
[INFO   ] [Window      ] virtual keyboard not allowed, single mode, not docked
[INFO   ] [Base        ] Start application main loop
```



Encerramento:

Ao encerrar é mostrada a pontuação de acordo com o jogador e o planeta

Códigos:

https://github.com/BrunaLimaFarias

projeto_star_wars.py

```
"""
Cadastro sobre filmes de Star Wars com um quiz sobre os filmes I II III
IV V VI
"""
# Sistema de Cadastro de Planetas
from galaxia_2d import Galaxia2DApp


# dicionario base
cadastro_planetas = {
    "Tatooine": {
        "personagens": "Anakin Skywalker, Luke Skywalker",
        "eventos": "Pod race, Duelo com Darth Vader",
        "perguntas": [
            {
                "pergunta": "Qual é o planeta natal de Anakin
Skywalker?",
                "respostas": ["a) Tatooine", "b) Coruscant", "c) Naboo"],
                "resposta_correta": "a",
                "explicacao": "Anakin Skywalker nasceu em Tatooine."
            }
        ]
    },
    "Endor": {
        "personagens": "Leia Organa",
        "eventos": "Batalha de Endor",
        "perguntas": [
            {
                "pergunta": "Onde os Ewoks vivem?",
                "respostas": ["a) Alderaan", "b) Coruscant", "c) Endor"],
                "resposta_correta": "c",
                "explicacao": "Os Ewoks vivem no planeta Endor."
            }
        ]
    }
}

# dicionário separado para as pontuações
pontuacoes = {}


# tratamento de erro, verifica e entrada nao esta vazia
def entrada_nao_vazia(mensagem):
```

```
    while True:
        entrada = input(mensagem).strip()
        if entrada:
            return entrada
        else:
            print("Por favor, insira uma entrada valida")

def cadastrar_planeta():
    nome_planeta = entrada_nao_vazia("Digite o nome do planeta: ")
    personagens = entrada_nao_vazia("Digite os personagens que nasceram
neste planeta: ")
    eventos = entrada_nao_vazia("Digite os eventos relacionados a este
planeta: ")

    # adiciona no dicionario
    cadastro_planetas[nome_planeta] = {
        "personagens": personagens,
        "eventos": eventos,
        "perguntas": []
    }

def cadastrar_pergunta_planeta():
    nome_planeta = entrada_nao_vazia("Digite o nome do planeta para
cadastrar a pergunta: ")

    # verificar se planeta existe no cadastro
    if nome_planeta in cadastro_planetas:
        pergunta = entrada_nao_vazia("Digite a pergunta: ")
        respostas = [input("Digite a alternativa (a): "), input("Digite a
alternativa (b): "), input("Digite a alternativa (c): ")]
        resposta_correta = entrada_nao_vazia("Digite a resposta correta
(a, b, c): ")
        explicacao = entrada_nao_vazia("Digite uma explicação da
resposta: ")

        # adiciona no dicionario
        cadastro_planetas[nome_planeta]["perguntas"].append({
            "pergunta": pergunta,
            "respostas": respostas,
            "resposta_correta": resposta_correta,
            "explicacao": explicacao
        })
    else:
        print(f"O planeta {nome_planeta} não está cadastrado. Cadastre
primeiro.")

# Funcao para mostrar lista de planetas que estao no dicionario de
cadastro
def lista_planetas():
```

```
    print("\nLista de Planetas:")
    for planeta in cadastro_planetas.keys():
        print(f"\n- {planeta}")

# retorna perguntas sobre um planeta especifico
def get_pergunta(nome_planeta):
    return cadastro_planetas.get(nome_planeta, {}).get("perguntas", [])

def executar_quiz():
    nome_planeta = input("Digite o nome do planeta para o quiz: ")

    # verifica se o planeta existe no cadastro
    if nome_planeta in cadastro_planetas:
        jogador = input("Digite o nome do jogador: ")

        # usando as perguntas cadastradas para cada planeta específico
        pontuacao = 0
        perguntas = cadastro_planetas.get(nome_planeta,
{}).get("perguntas", [])

        if not perguntas:
            print(f"Não há perguntas cadastradas para o planeta
{nome_planeta}.")
            return

        print(f"Quiz sobre o planeta {nome_planeta} para {jogador}:\n")

        # dicionário sobre uma pergunta especifica relacionada a um
planeta
        for pergunta_data in perguntas:
            print(pergunta_data["pergunta"])
            for resposta in pergunta_data["respostas"]:
                print(resposta)

            resposta = input("Sua resposta (a, b ou c): ")

            if resposta == pergunta_data["resposta_correta"]:
                print("Correto! Você ganhou um ponto.")
                pontuacao += 1
            else:
                print("Errado. A resposta correta é:",
pergunta_data["resposta_correta"])
                print(pergunta_data["explicacao"])

        if jogador not in pontuacoes:
            pontuacoes[jogador] = {}

        pontuacoes[jogador][nome_planeta] = pontuacao
    else:
```

```
        print(f"O planeta {nome_planeta} não está cadastrado. Cadastre-o
primeiro.")

def executa_estrela_da_morte():
    print("  .           __---__")
    print(".     .   .-'...:...'-.               .          .")
    print("         / .  . : .__
.\                                      .")
    print("  .     /........./  \
.\  .   .                              .")
    print("       / :  :   :| () | :\                 .        .")
    print("      :...........\__/....:         .")
    print("      |____________________|              .
.")
    print("      |...................|              .")
    print(".     :  :  :   :   :   :
:                          .           .")
    print("    .  \................/      .            .")
    print("        \  .  . :
.  .  /   .                              .")
    print("     .   \._........._./  .        .                  .")
    print("           -..___..-               .         .")
    print("")

def main():
    while True:
        print("\n")
        print("                    8888888888  888    88888")
        print("                   88     88   88 88   88  88")
        print("                    8888  88  88   88  88888")
        print("                       88 88 888888888 88   88")
        print("                88888888  88 88     88 88    888888")
        print(" ")
        print("                88  88  88   888    88888    888888")
        print("                88  88  88  88 88   88  88  88")
        print("                88 8888 88 88   88  88888    8888")
        print("                 888  888 888888888 88   88     88")
        print("                  88  88  88     88 88    8888888")
        print("\nMenu:")
        print("1. Cadastrar Planeta")
        print("2. Cadastrar Pergunta")
        print("3. Lista de Planetas")
        print("4. Iniciar Quiz")
        print("5. Mostrar Galaxia 2D")
        print("6. Encerrar")
        opcao = input("Escolha uma opção: ")

        if opcao == "1":
            cadastrar_planeta()
```

```
        elif opcao == "2":
            cadastrar_pergunta_planeta()

        elif opcao == "3":
            lista_planetas()

        elif opcao == "4":
            executar_quiz()

        elif opcao == "5":
            galaxia_app = Galaxia2DApp()
            galaxia_app.run()

        elif opcao == "6":
            break

        else:
            print("Opção inválida")


    print("Pontuações finais:")
    for jogador, dados in pontuacoes.items():
        for planeta, pontuacao in dados.items():
            print(f"{jogador} - Planeta {planeta}: Pontuação:
{pontuacao}")


    print(executa_estrela_da_morte())


if __name__ == "__main__":
    main()
```

galaxia_2d.py

```
# galaxia_2d.py
# import do kivy para a interface grafica

from kivy.app import App
from kivy.uix.widget import Widget
from kivy.graphics import Ellipse, Color
from kivy.clock import Clock
from math import cos, sin, radians

# tela para visualização do modelo 2D
```

```
class Galaxia2DWidget(Widget):
    def __init__(self, **kwargs):
        super(Galaxia2DWidget, self).__init__(**kwargs)

        # define os planetas, velocidade, distancia, raio e cor
        self.planets = {
            'Tatooine': {'radius': 5, 'distance': 30, 'color': (1, 0.8,
0), 'angle': 5, 'orbit_speed': 10},
            'Coruscant': {'radius': 25, 'distance': 120, 'color': (0,
0.5, 1), 'angle': 0, 'orbit_speed': 6},
            'Endor': {'radius': 18, 'distance': 180, 'color': (0, 0.8,
0), 'angle': 0, 'orbit_speed': 8},
            'Hoth': {'radius': 22, 'distance': 250, 'color': (1, 1, 1),
'angle': 0, 'orbit_speed': 7},
        }

        for planet, info in self.planets.items():
            with self.canvas:
                Color(*info['color'])
                info['ellipse'] =
Ellipse(pos=self.calculate_position(planet), size=(info['radius'] * 2,
info['radius'] * 2))

        Clock.schedule_interval(self.update, 1 / 60.0)

    # posição do planeta
    def calculate_position(self, planet):
        info = self.planets[planet]
        angle_rad = radians(info['angle'])
        x = self.width / 2 + info['distance'] * cos(angle_rad)
        y = self.height / 2 + info['distance'] * sin(angle_rad)
        return (x - info['radius'], y - info['radius'])

    # animação dos planetas, atualizando a posição da elipse
    def update(self, dt):
        for planet, info in self.planets.items():
            info['angle'] += dt * info['orbit_speed']  # ajuste
velocidade do movimento orbital
            info['ellipse'].pos = self.calculate_position(planet)

# chama a tela para abrir o aplicativo de visualização
class Galaxia2DApp(App):
    def build(self):
        return Galaxia2DWidget()

if __name__ == "__main__":
    galaxia_app = Galaxia2DApp()
    galaxia_app.run()
```