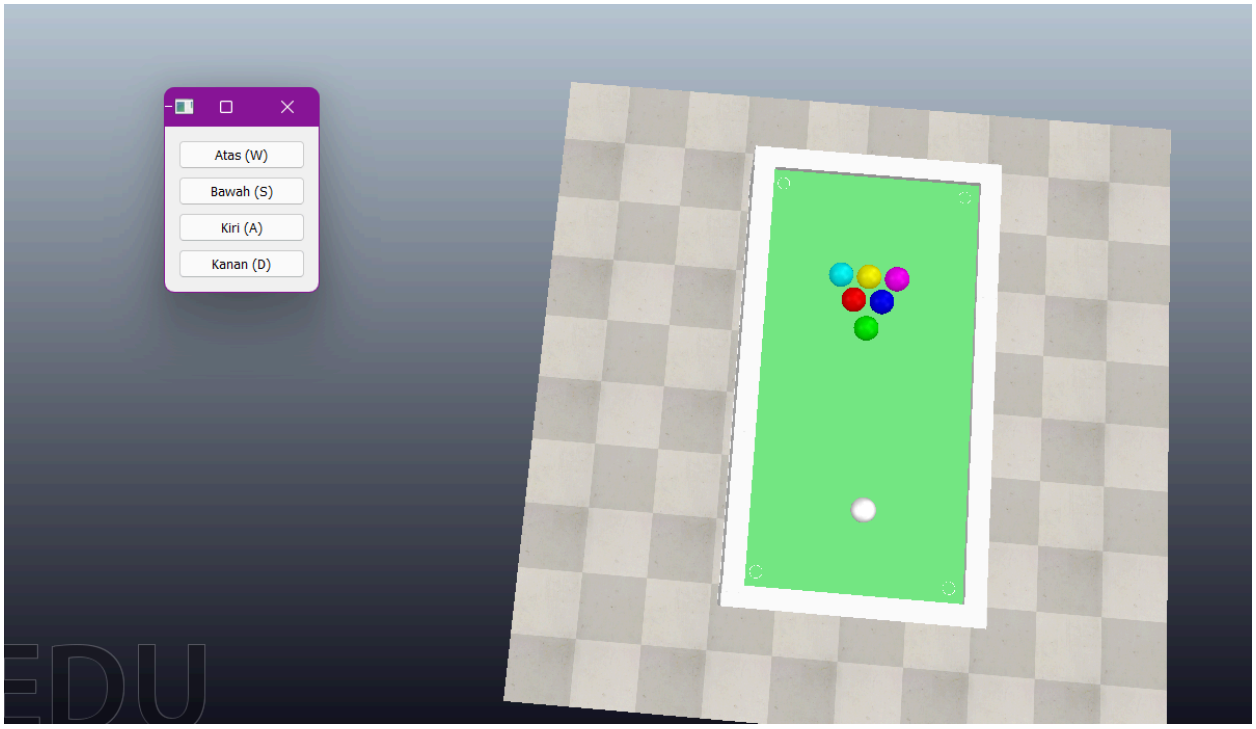
Harits Shidqi Fadhiil
5022221014



Berikut merupakan percobaan untuk membuat simulasi permainan billiard. Dimana fungsi tab dengan opsi atas, bawah, kiri, kanan digunakan untuk menggerakan bola putih sesuai dengan arah yang dipilih.

```python
from coppeliasim_zmqremoteapi_client import RemoteAPIClient
from PyQt5.QtWidgets import QApplication, QWidget, QPushButton, QVBoxLayout
from PyQt5.QtCore import QCoreApplication
import sys
import math

# Inisialisasi klien CoppeliaSim di luar kelas
client = RemoteAPIClient()
sim = client.getObject('sim')
sim.setStepping(False)
sim.startSimulation()

class BilliardApp(QWidget):
    def __init__(self):
        super().__init__()
        self.init_ui()
        self.b2_Handle = sim.getObject("/Sphere[6]")

    def init_ui(self):
        self.setWindowTitle("Kontrol Bola Putih")
        self.setGeometry(10, 10, 20, 15)
```

```
        # Buat tombol-tombol
        self.btn_up = QPushButton("Atas (W)")
        self.btn_down = QPushButton("Bawah (S)")
        self.btn_left = QPushButton("Kiri (A)")
        self.btn_right = QPushButton("Kanan (D)")

        # Hubungkan tombol dengan fungsi kontrol
        self.btn_up.clicked.connect(lambda: self.apply_force(10))
        self.btn_down.clicked.connect(lambda: self.apply_force(30))
        self.btn_left.clicked.connect(lambda: self.apply_force(20))
        self.btn_right.clicked.connect(lambda: self.apply_force(0))

        # Atur tata letak
        layout = QVBoxLayout()
        layout.addWidget(self.btn_up)
        layout.addWidget(self.btn_down)
        layout.addWidget(self.btn_left)
        layout.addWidget(self.btn_right)
        self.setLayout(layout)
```

```
    def apply_force(self, direction_angle_deg):
        force_magnitude = 5
        direction_angle_rad = math.radians(direction_angle_deg)
        force_x = force_magnitude * math.cos(direction_angle_rad)
        force_y = force_magnitude * math.sin(direction_angle_rad)
        force_vector = [force_x, force_y, 0]

        sim.addForce(self.b2_Handle, sim.getObjectPosition(self.b2_Handle, -1), force_vector)
        sim.addStatusbarMessage(f"Gaya diterapkan. Arah: {direction_angle_deg}°, Kekuatan: {force_magnitude} N")

if __name__ == '__main__':
    app = QApplication(sys.argv)
    ex = BilliardApp()
    ex.show()
    sys.exit(app.exec_())
```

Program ini pada dasarnya menciptakan antarmuka pengguna grafis (GUI) sederhana dengan empat tombol yang, ketika diklik, akan menerapkan gaya pada bola putih di lingkungan simulasi CoppeliaSim.

1. Impor Pustaka (`Imports`)

Kode ini mengimpor pustaka yang diperlukan:

- `RemoteAPIClient`: Digunakan untuk menghubungkan dan berkomunikasi dengan CoppeliaSim.
- `PyQt5.QtWidgets`: Untuk membuat antarmuka pengguna grafis (GUI), seperti jendela, tombol, dan tata letak.

- `PyQt5.QtCore`: Untuk fungsionalitas inti Qt.
- `sys`: Untuk interaksi dengan sistem, seperti mengelola aplikasi PyQt.
- `math`: Untuk perhitungan matematis, khususnya konversi sudut.

---

2. Inisialisasi Klien CoppeliaSim

- **`client = RemoteAPIClient()`**: Membuat instance klien untuk terhubung ke CoppeliaSim.
- **`sim = client.getObject('sim')`**: Mendapatkan objek `sim` yang merupakan antarmuka ke fungsi-fungsi simulasi.
- **`sim.setStepping(False)`**: Mengatur simulasi ke mode *real-time* (bukan *stepping*).
- **`sim.startSimulation()`**: Memulai simulasi di CoppeliaSim.

---

3. Kelas `BilliardApp`

Ini adalah kelas utama yang mengelola GUI dan logika aplikasi.

- **`__init__`**: Metode konstruktor yang dipanggil saat objek dibuat.
  - `super().__init__()`: Memanggil konstruktor kelas induk `QWidget`.
  - `self.init_ui()`: Memanggil metode untuk inisialisasi antarmuka.
  - `self.b2_Handle = sim.getObject("/Sphere[6]")`: Mendapatkan *handle* (pengenal unik) dari objek bola putih di CoppeliaSim. `/Sphere[6]` adalah jalur hierarki (path) yang mengacu pada bola putih.

---

4. Metode `init_ui`

Metode ini bertanggung jawab untuk membangun antarmuka pengguna:

- Menyiapkan jendela (`setWindowTitle`, `setGeometry`).
- Membuat empat tombol (`QPushButton`) dengan label **"Atas (W)"**, **"Bawah (S)"**, **"Kiri (A)"**, dan **"Kanan (D)"**.
- Menghubungkan sinyal `clicked` dari setiap tombol ke metode `self.apply_force()` dengan argumen sudut yang berbeda.
- Mengatur tombol-tombol tersebut dalam tata letak vertikal (`QVBoxLayout`).

---

5. Metode `apply_force`

Ini adalah inti dari logika simulasi:

- Menerima `direction_angle_deg` (sudut dalam derajat) sebagai masukan.
- **`force_magnitude = 5`**: Menentukan besaran (kekuatan) gaya yang akan diterapkan. Nilai ini tetap konstan.
- **`direction_angle_rad = math.radians(direction_angle_deg)`**: Mengonversi sudut dari derajat ke radian karena fungsi trigonometri Python menggunakan radian.
- **`force_x` dan `force_y`**: Menghitung komponen gaya pada sumbu X dan Y menggunakan trigonometri ($Fx = F \cdot \cos(\theta)$, $Fy = F \cdot \sin(\theta)$).
- **`force_vector = [force_x, force_y, 0]`**: Membuat vektor gaya 3D dengan komponen Z bernilai 0 (asumsi gerakan hanya pada bidang X-Y).
- **`sim.addForce(...)`**: Fungsi utama dari CoppeliaSim API yang menerapkan gaya.
    - **`self.b2_Handle`**: Objek yang akan dikenai gaya (bola putih).
    - **`sim.getObjectPosition(...)`**: Titik penerapan gaya. Dalam kasus ini, gaya diterapkan di pusat massa bola.
    - **`force_vector`**: Vektor gaya yang telah dihitung.
- **`sim.addStatusbarMessage(...)`**: Menampilkan pesan di bilah status CoppeliaSim untuk memberikan umpan balik kepada pengguna.

---

6. Eksekusi Program Utama

- **`if __name__ == '__main__':`**: Memastikan kode hanya berjalan saat skrip dieksekusi secara langsung.
- **`app = QApplication(sys.argv)`**: Membuat instance aplikasi PyQt.
- **`ex = BilliardApp()`**: Membuat objek `BilliardApp`.
- **`ex.show()`**: Menampilkan jendela GUI.
- **`sys.exit(app.exec_())`**: Memulai *event loop* aplikasi dan memastikan aplikasi ditutup dengan benar saat pengguna keluar.

Secara ringkas, kode ini menjembatani interaksi pengguna (klik tombol pada GUI) dengan simulasi fisik (menerapkan gaya pada bola) melalui API CoppeliaSim. Ini adalah contoh sederhana dari aplikasi *real-time* yang mengintegrasikan GUI dengan simulasi robotika/fisika.