Tugas 2 Sistem Robot Otonom: Simulasi Billiard Menggunakan Coppelia Sim dan Bahasa Pemrograman Python Dengan Perantara ZMQ Remote API

Nama: Falah Amru Dikasmara
NRP: 5022211041
Mata Kuliah: Sistem Robot Otonom

1 Pendahuluan

1.1 Latar Belakang

Proyek ini adalah simulasi permainan billiar yang dibuat menggunakan perangkat lunak coppelia sim edu dan bahasa pemrograman python. Dalam proyek ini, Program python berkomunikasi dengan perangkat lunak copelliasim menggunakan ZMQ Remote API. Proyek ini bertujuan sebagai sarana edukasi dalam memahami cara menggunakan coppelia sim untuk membangun lingkungan (scene) yang dapat mensimulasikan konsep-konsep fisika dasar seperti dinamika benda kaku (gerak translasi dan rotasi) yang mengabaikan deformasi (perubahan bentuk dan ukuran benda) dan hukum kekekalan energi dan momentum yang dibutuhkan untuk memodelkan interaksi (tumbukan) antar objek, serta cara menggunakan ZMQ Remote API yang memungkinkan pengguna untuk membaca informasi dan memberi perintah pada lingkungan simulasi menggunakan program python.

1.2 Tujuan

Adapun tujuan dari proyek ini adalah :

1. Mempelajari cara membuat lingkungan simulasi menggunakan Coppelia Sim Edu.
2. Mempelajari cara memanfaatkan ZMQ Remote API untuk membaca informasi dan memberi perintah pada lingkungan simulasi menggunakan program python.
3. Merancang program python yang memungkinkan pemain untuk menentukan kecepatan dan arah bola putih melalui terminal.
4. Merancang program python yang memungkinkan pemain untuk menentukan vektor gaya dan torsi bola putih secara manual melalui terminal.
5. Membuat laporan sebagai bahan pembelajaran mahasiswa

1.3 Fitur Proyek

Pada proyek simulasi permainan billiar ini, program diharapkan dapat memungkinkan pemain untuk menentukan kecepatan bola putih, menentukan bola target, dan memberikan gaya pada titik tertentu relatif terhadap pusat massa bola putih untuk menciptakan efek torsi saat peluncuran bola. Selain itu, program diharapkan dapat menyediakan pilihan bagi pemain untuk menentukan vektor gaya dan torsi dari bola putih secara manual. Seluruh input dari pemain dimasukkan melalui dialog pada terminal.

1.4 Software yang Dibutuhkan dan Fungsinya

Adapun beberapa software yang dibutuhkan untuk membuat proyek ini antara lain :

1. Python : Sebagai bahasa pemrograman dan interpreter
2. Coppelia Sim Edu : Sebagai software untuk membuat lingkungan simulasi yang terdiri atas papan billiar dan beberapa bola billiar
3. Mempelajari cara memanfaatkan ZMQ Remote API untuk membaca informasi dan memberi perintah pada lingkungan simulasi menggunakan program python.
4. Merancang program python yang memungkinkan pemain untuk menentukan kecepatan dan arah bola putih melalui terminal.

5. Merancang program python yang memungkinkan pemain untuk menentukan vektor gaya dan torsi bola putih secara manual melalui terminal.
6. Membuat laporan sebagai bahan pembelajaran mahasiswa

1.5 Tantangan

Dalam pengerjaan proyek ini, terdapat beberapa tantangan yang dihadapi, antara lain :

1. Menentukan algortima yang tepat agar program dapat memenuhi tujuan proyek.
2. Menentukan perintah API CoppeliaSim yang tepat untuk melakukan tugas tertentu.
3. Menentukan metode yang tepat agar koneksi antara program python dan CoppeliaSim tidak terputus saat program berjalan.

1.6 Kemampuan yang Dipelajari

1. Penggunanan CoppeliaSim untuk membuat lingkungan simulasi sederhana.
2. Pemrograman dalam bahasa python dan penggunaan ZMQ Remote API untuk membaca data dan memberi perintah pada lingkungan simulasi.
3. Penggunaan GitHub untuk dokumentasi proyek.

2 Dasar Teori

2.1 Python

Python merupakan bahasa pemrograman yang diciptakan Guido van Rossum dan pertama kali dirilis pada tahun 1991. Python digolongkan sebagai bahasa pemrograman tingkat tinggi karena cara penulisannya lebih dekat dengan bahasa manusia dan lebih jauh dari bahasa mesin, sehingga memberikan beberapa kelebihan. Pengguna python tidak perlu mengatur detail teknis yang berhubungan dengan perangkat keras seperti alamat memori, register CPU, atau instruksi mesin, sehingga membuat baris perintah python lebih ringkas daripada bahasa pemrograman lain untuk tugas yang sama. Sintaks pada python juga mudah dipahami karena mirip dengan bahasa inggris. Python dapat dijalankan pada berbagai sistem operasi karena memiliki interpreter yang dapat menerjemahkan kode ke instruksi yang sesuai dengan sistem yang dipakai. Berbagai kelebihan tersebut membuat penggunaan python sangat populer bagi banyak orang dan mendorong dikembangkannya berbagai pustaka dan modul siap pakai untuk berbagai tugas, sehingga memungkinkan programmer fokus ke logika aplikasi daripada detail implementasi rendah.

2.2 CoppeliaSim

CoppeliaSim adalah sebuah perangkat lunak simulasi robotika yang banyak digunakan di bidang penelitian, pendidikan, dan pengembangan sistem otonom. Aplikasi ini memungkinkan pengguna membuat model robot, sensor, aktuator, serta lingkungan virtual yang realistis, kemudian menguji algoritma kendali atau kecerdasan buatan secara aman tanpa perlu langsung menggunakan perangkat keras. CoppeliaSim dilengkapi dengan mesin fisika untuk mensimulasikan dinamika gerakan, benturan, serta interaksi antar objek. Selain itu, software ini mendukung berbagai antarmuka pemrograman (API) seperti Python, C++, Java, MATLAB, dan ROS (Robot Operating System), sehingga memudahkan integrasi dengan proyek robotika nyata. Karena fleksibilitas dan kelengkapan fiturnya, CoppeliaSim menjadi salah satu platform simulasi robot paling populer untuk eksperimen akademik maupun industri sebelum implementasi di dunia nyata.

2.3 Application Programming Interface (API)

Application Programming Interface (API) adalah suatu antarmuka yang menyediakan sekumpulan aturan, fungsi, dan protokol yang memungkinkan satu perangkat lunak berkomunikasi dengan perangkat lunak lainnya. API berfungsi sebagai jembatan penghubung antara aplikasi dengan sistem operasi, pustaka, layanan, atau aplikasi eksternal sehingga pengembang tidak perlu memahami detail internal dari sistem yang digunakan. Dengan API, pertukaran data dan instruksi dapat dilakukan secara standar melalui mekanisme request dan response. Dalam praktiknya, API banyak digunakan, misalnya pada aplikasi cuaca yang mengambil data dari server penyedia layanan meteorologi. Keberadaan API mempermudah proses pengembangan perangkat lunak karena memungkinkan pemanfaatan fungsi dan layanan yang sudah ada, mendukung interoperabilitas antar sistem, serta mempercepat implementasi teknologi baru.

2.4 ZMQ Remote API

ZMQ Remote API adalah antarmuka komunikasi pada perangkat lunak CoppeliaSim yang dibangun di atas teknologi ZeroMQ (ZMQ) untuk memungkinkan interaksi antara simulasi dengan program eksternal. Melalui API ini, CoppeliaSim dapat dijalankan sebagai server yang menerima perintah dari program client yang ditulis dalam bahasa pemrograman seperti Python, C++, Java, atau MATLAB. Mekanisme kerjanya berbasis pertukaran pesan (message-based communication) yang bersifat asinkron, sehingga proses pengiriman instruksi dan penerimaan data dapat berjalan cepat, efisien, serta lintas platform. Dengan adanya ZMQ Remote API, pengguna tidak hanya dapat memulai dan menghentikan simulasi, tetapi juga mengontrol objek, menggerakkan robot, membaca sensor, dan melakukan integrasi dengan sistem lain seperti Robot Operating System (ROS). Keunggulan utama pendekatan ini adalah pemisahan antara logika kendali robot yang dijalankan di luar simulasi dengan lingkungan virtual yang ada di dalam CoppeliaSim, sehingga memberikan fleksibilitas tinggi, kemudahan integrasi, serta kondisi eksperimen yang lebih mendekati penerapan nyata pada sistem robotika.

2.5 Gaya, Momentum, Impuls, dan Hukum Kekekalan Momentum

Gaya adalah interaksi berupa dorongan atau tarikan yang dapat menyebabkan perubahan gerak maupun bentuk suatu benda. Secara teoritis, hubungan gaya dengan gerak dijelaskan oleh Hukum II Newton:

$$\vec{F} = m \cdot \vec{a},$$

dengan $\vec{F}$ menyatakan gaya (Newton), m adalah massa benda (kg), dan $\vec{a}$ adalah percepatan (m/s^2). Hal ini berarti gaya bekerja sebagai penyebab perubahan kecepatan benda. Jika resultan gaya nol, benda akan tetap diam atau bergerak lurus beraturan sesuai Hukum I Newton. Dalam pengalaman nyata, gaya dapat muncul dalam berbagai bentuk seperti *gaya gesek*, *gaya normal*, *gaya tegangan*, *gaya pegas*, hingga *gaya medan* seperti gravitasi dan elektromagnetik.

Momentum adalah besaran vektor yang menggambarkan kuantitas gerak suatu benda. Momentum secara matematis dirumuskan sebagai

$$\vec{p} = m \cdot \vec{v},$$

dengan $\vec{p}$ adalah momentum (kg·m/s), m massa, dan $\vec{v}$ kecepatan. Secara nyata, momentum menunjukkan "kesulitan" suatu benda untuk dihentikan atau diubah arah geraknya. Misalnya, truk yang bergerak pelan tetap lebih sulit dihentikan daripada bola pingpong yang bergerak cepat, karena truk memiliki momentum lebih besar. Momentum juga searah dengan kecepatan benda, sehingga mengubah arah gerak sama artinya dengan mengubah arah momentum, yang selalu membutuhkan gaya tambahan.

Hubungan antara gaya dan momentum dijelaskan lebih umum oleh persamaan:

$$\vec{F} = \frac{d\vec{p}}{dt}.$$

Artinya, gaya merupakan penyebab perubahan momentum terhadap waktu. Jika massa benda konstan, persamaan ini setara dengan $F = m \cdot a$, namun dalam kasus massa berubah (seperti pada roket yang membuang bahan bakar), bentuk persamaan dalam momentum lebih tepat digunakan. Dengan demikian, gaya tidak hanya mempercepat benda, tetapi lebih luas: gaya adalah sumber perubahan momentum.

Impuls adalah besaran yang menyatakan hasil kali gaya dengan waktu kerjanya. Secara matematis didefinisikan sebagai:

$$\vec{I} = \int_{t_1}^{t_2} \vec{F}\, dt,$$

dan bernilai sama dengan perubahan momentum:

$$\vec{I} = \Delta\vec{p}.$$

Hal ini berarti gaya yang bekerja dalam suatu selang waktu tertentu akan menyebabkan perubahan momentum pada benda. Secara nyata, semakin besar gaya atau semakin lama gaya bekerja, semakin besar perubahan momentum yang terjadi. Contohnya, saat seorang pemain menendang bola, gaya besar dalam waktu singkat menghasilkan impuls yang mengubah momentum bola, sehingga bola melesat cepat. Sebaliknya, pada sistem keselamatan seperti airbag atau bantalan, waktu tumbukan diperpanjang sehingga gaya rata-rata yang diterima tubuh lebih kecil, meskipun perubahan momentum (dari bergerak hingga berhenti) sama.

Hukum Kekekalan Momentum menyatakan bahwa momentum total suatu sistem tertutup (tanpa gaya luar) akan tetap konstan sebelum dan sesudah interaksi. Prinsip kekekalan momentum inilah yang menjadi dasar dalam banyak peristiwa fisis maupun teknologi. Salah satu contoh nyata dari **Hukum Kekekalan Momentum** dapat dilihat pada permainan bola biliar. Misalkan sebuah bola biliar A bergerak dengan kecepatan tertentu dan menumbuk bola biliar B yang semula diam. Sebelum tumbukan, momentum sistem hanya berasal dari bola A:

$$p_{\text{awal}} = m_A \cdot v_A,$$

karena $v_B = 0$. Setelah tumbukan, kedua bola bergerak dengan kecepatan berbeda, sehingga momentum total sistem adalah

$$p_{\text{akhir}} = m_A \cdot v'_A + m_B \cdot v'_B,$$

dengan v'_A dan v'_B masing-masing adalah kecepatan bola A dan B setelah tumbukan. Menurut hukum kekekalan momentum, berlaku

$$p_{\text{awal}} = p_{\text{akhir}},$$

atau

$$m_A \cdot v_A = m_A \cdot v'_A + m_B \cdot v'_B.$$

Secara nyata, hal ini berarti momentum yang semula dimiliki oleh bola A *ditransfer* sebagian kepada bola B saat tumbukan. Akibatnya, bola A melambat atau bahkan berhenti, sementara bola B bergerak. Jika kedua bola identik ($m_A = m_B$) dan tumbukan bersifat lenting sempurna (*elastic collision*), maka bola A akan berhenti total setelah tumbuk

2.6 Torsi

Torsi adalah besaran fisis yang menggambarkan kecenderungan suatu gaya untuk memutar benda terhadap suatu titik atau sumbu putar. Dalam kehidupan nyata, torsi muncul ketika gaya diberikan tidak tepat pada sumbu, melainkan pada jarak tertentu dari sumbu rotasi. Contoh sehari-hari adalah saat kita membuka pintu dengan mendorong gagangnya: semakin jauh posisi gagang dari engsel (sumbu putar), semakin mudah pintu terbuka. Secara matematis, torsi ($\vec{\tau}$) dirumuskan sebagai hasil perkalian silang antara vektor lengan gaya ($\vec{r}$) dengan vektor gaya ($\vec{F}$):

$$\vec{\tau} = \vec{r} \times \vec{F},$$

dengan besar torsi

$$\tau = r \cdot F \cdot \sin\theta,$$

di mana r adalah jarak dari sumbu putar ke titik tangkap gaya, F adalah besar gaya, dan θ adalah sudut antara vektor $\vec{r}$ dan $\vec{F}$.

Secara nyata, torsi dapat dipahami sebagai "gaya putar". Besarnya torsi bergantung pada tiga faktor: (1) besar gaya yang diberikan, (2) jarak lengan gaya dari sumbu, dan (3) arah gaya relatif terhadap lengan gaya. Semakin besar gaya dan semakin jauh jaraknya dari sumbu, semakin besar torsi yang dihasilkan.

Dalam dinamika rotasi, torsi berperan sama seperti gaya pada gerak translasi. Torsi menjadi penyebab percepatan sudut (α) pada benda tegar yang berotasi. Hubungan ini dinyatakan dengan persamaan:

$$\sum \tau = I \cdot \alpha,$$

dengan I adalah momen inersia ($\text{kg}\cdot\text{m}^2$) yang menyatakan kelembaman benda terhadap rotasi, dan α adalah percepatan sudut (rad/s^2). Artinya, semakin besar momen inersia, semakin sulit suatu benda diputar oleh torsi tertentu, mirip dengan massa yang menahan percepatan pada gerak lurus.

Dalam permainan biliar, **torsi** berperan penting ketika tongkat biliar (cue stick) mengenai bola tidak tepat di pusat massanya. Jika gaya pukulan diberikan tepat di pusat bola, maka bola hanya akan bergerak translasi (maju lurus tanpa berputar). Namun, ketika gaya diberikan dengan sedikit penyimpangan dari pusat (misalnya lebih ke atas, bawah, kiri, atau kanan permukaan bola), maka gaya tersebut menghasilkan torsi terhadap pusat bola. Akibatnya, bola tidak hanya bergerak translasi tetapi juga berotasi (menggelinding atau berputar). Semakin jauh titik pukulan dari pusat, semakin besar torsi yang dihasilkan, dan semakin besar percepatan sudut bola. Hubungan ini menjelaskan bagaimana variasi posisi pukulan dapat mengubah gaya putar bola.

Sebagai contoh perhitungan sederhana, misalkan sebuah bola biliar homogen dengan massa $m = 0.16\,\text{kg}$ dan jari-jari $R = 0.028\,\text{m}$ (diameter sekitar 56 mm). Tongkat biliar memukul bola dengan gaya $F = 20\,\text{N}$ pada titik yang berjarak $r = 0.01\,\text{m}$ di bawah pusat bola. Dengan demikian, gaya tidak hanya memberikan gerak translasi, tetapi juga menghasilkan torsi.

Besar torsi dihitung sebagai:

$$\tau = r \cdot F = 0.01 \times 20 = 0.20\,\text{N·m}.$$

Momen inersia bola padat terhadap sumbu melalui pusatnya adalah:

$$I = \frac{2}{5}mR^2 = \frac{2}{5} \times 0.16 \times (0.028)^2 \approx 5.0 \times 10^{-5}\,\text{kg·m}^2.$$

Dengan demikian, percepatan sudut bola (α) akibat torsi adalah:

$$\alpha = \frac{\tau}{I} = \frac{0.20}{5.0 \times 10^{-5}} \approx 4000\,\text{rad/s}^2.$$

Hasil ini menunjukkan bahwa pukulan dengan gaya cukup besar dan sedikit menyimpang dari pusat dapat menghasilkan percepatan sudut yang tinggi, sehingga bola berputar dengan cepat selain bergerak maju secara translasi. Dalam praktik permainan biliar, efek ini tampak pada pukulan *back spin* (bola mundur setelah tumbukan), *top spin* (bola tetap bergerak maju setelah kontak), maupun *side spin* (bola berbelok setelah mengenai bantalan meja). Semua efek ini merupakan hasil kombinasi translasi dan rotasi yang muncul karena torsi dari pukulan tongkat biliar.

3 Alur Kerja Program Simulasi Permainan Biliar

Program ini dibuat untuk mengendalikan simulasi permainan biliar pada CoppeliaSim menggunakan Python dan Remote API. Alur kerjanya dapat dijelaskan sebagai berikut:

1. **Inisialisasi Koneksi dan Simulasi**
   - Program melakukan koneksi ke server Remote API CoppeliaSim menggunakan `RemoteAPIClient`.
   - Simulasi dimulai dengan perintah `sim.startSimulation()`.
   - Mode stepping diatur OFF secara default agar simulasi berjalan bebas kecuali saat diperlukan sinkronisasi.
2. **Penemuan dan Persiapan Bola**
   - Semua bola pada papan biliar ditemukan berdasarkan daftar path (`BALL_NAMES`).
   - Bola putih (cue ball) diidentifikasi secara khusus.
   - Parameter dasar seperti massa, diameter, dan radius cue ball dibaca dari GUI CoppeliaSim.
   - Cue ball direset ke keadaan diam (tanpa gaya maupun kecepatan awal).
3. **Dialog Interaktif Satu Giliran**
   - Program masuk ke loop interaktif untuk setiap giliran.
   - Pemain memilih mode bidik:
     (a) **Manual**: pemain memasukkan komponen gaya (F_x, F_y, F_z) dan torsi (T_x, T_y, T_z) secara langsung. Opsional, gaya dapat diberikan pada titik offset relatif terhadap pusat massa (COM). Offset ini otomatis dikurangi (clamped) agar tidak melebihi radius bola.
     (b) **Target**: pemain memilih bola sasaran (`by_name` atau `nearest`). Program menghitung vektor arah cue → target, lalu menerapkan impuls sekali. Jika mode offset dipilih, pemain memasukkan vektor offset relatif terhadap COM.
   - Gaya atau impuls diterapkan secara sinkron (stepping ON → STEP → OFF).
4. **Free-Run dan Monitoring**
   - Setelah gaya atau impuls diterapkan, simulasi dilepas ke mode free-run (stepping OFF).
   - Simulasi berjalan bebas selama durasi tertentu (misalnya 4 detik).

- Kecepatan linear dan angular cue ball dapat dimonitor dan ditampilkan di terminal.

5. **Akhir Giliran**
   - Setelah free-run, semua bola dihentikan (kecepatan dan gaya dinolkan).
   - Posisi akhir semua bola dicetak pada terminal.
   - Pemain ditanya apakah ingin melanjutkan giliran berikutnya (`y/n`).
6. **Akhir Permainan**
   - Jika pemain memilih berhenti, loop interaktif dihentikan.
   - Semua bola kembali dihentikan untuk memastikan kondisi stabil.
   - Simulasi dihentikan dengan `sim.stopSimulation()`.

Secara keseluruhan, program ini memungkinkan simulasi permainan biliar interaktif dengan dua mode tembakan (manual dan target), dukungan offset gaya yang dikontrol radius bola, serta kontrol sinkronisasi penuh menggunakan mekanisme stepping pada Remote API CoppeliaSim.

4 Tutorial Penggunaan Program

Berikut adalah langkah-langkah penggunaan program untuk simulasi permainan biliar menggunakan Python dan CoppeliaSim:

1. **Persiapan Lingkungan**
   - Pastikan CoppeliaSim sudah terpasang dan dapat dijalankan.
   - Pastikan Python 3 sudah terinstal pada komputer.
   - Instal pustaka Remote API Python untuk CoppeliaSim:

     ```
     pip install coppeliasim-zmqremoteapi-client
     ```

   - Buka CoppeliaSim, muat scene yang berisi bola-bola biliar (`/Sphere[0]`, `/Sphere[1]`, ..., `/Sphere[6]`).
2. **Menjalankan Program**
   - Jalankan file Python melalui terminal atau command prompt.
   - Program akan melakukan koneksi ke CoppeliaSim, menemukan semua bola, dan menampilkan informasi massa serta diameter bola putih (cue ball).
   - Posisi awal semua bola akan ditampilkan di terminal.
3. **Interaksi Selama Simulasi**
   - Pada setiap giliran, program akan menampilkan pilihan *mode bidik*:
     (a) **Manual**: pengguna memasukkan gaya (F_x, F_y, F_z) dan torsi (T_x, T_y, T_z) secara langsung. Pengguna dapat memilih apakah gaya diberikan pada titik offset relatif terhadap pusat massa. Jika offset melebihi radius bola, program otomatis melakukan *clamp*.
     (b) **Target**: pengguna memilih bola sasaran, baik dengan nama (`by_name`) atau bola terdekat (`nearest`). Program menghitung arah cue → target, lalu menerapkan impuls. Jika mode offset dipilih, pengguna memasukkan vektor offset (r_x, r_y, r_z).
   - Setelah input dimasukkan, program akan menampilkan ringkasan gaya/impuls yang diterapkan.
4. **Fase Free-Run**
   - Setelah gaya/impuls diterapkan, simulasi dilepas ke mode bebas (`free-run`).
   - Bola bergerak sesuai hukum fisika, dan kecepatan bola putih akan dipantau selama beberapa detik.
   - Durasi free-run dapat ditentukan oleh pengguna.
5. **Akhir Giliran**
   - Setelah free-run selesai, semua bola dihentikan (reset velocity).
   - Posisi akhir semua bola ditampilkan pada terminal.

- Program menanyakan apakah pengguna ingin melanjutkan giliran berikutnya:
  - `y` → permainan dilanjutkan ke giliran baru.
  - `n` → permainan dihentikan.

6. **Mengakhiri Permainan**
   - Jika pengguna memilih `n`, maka:
     (a) Semua bola dihentikan agar tidak ada pergerakan tersisa.
     (b) Program menghentikan simulasi di CoppeliaSim.
     (c) Program menampilkan pesan bahwa permainan selesai.

5 Kesimpulan

Berdasarkan implementasi dan pengujian yang telah dilakukan, dapat disimpulkan bahwa:

1. Program simulasi permainan biliar berbasis Python dan CoppeliaSim berhasil dibuat dengan memanfaatkan *Remote API (ZeroMQ)*.
2. Program mampu berinteraksi secara *real-time* dengan objek di dalam scene CoppeliaSim, khususnya bola biliar, dengan melakukan pembacaan massa, diameter, dan posisi bola secara otomatis dari properti objek di CoppeliaSim.
3. Program mampu menyediakan penargetan interaktif dengan dua mode tembakan (manual dan target), dukungan offset gaya yang dikontrol radius bola, serta kontrol sinkronisasi penuh menggunakan mekanisme stepping pada Remote API CoppeliaSim.
4. Sistem *stepping* yang diterapkan (`setStepping(True)` saat kontrol sinkron, dan `setStepping(False)` untuk free-run) membuat simulasi dapat dijalankan secara stabil tanpa membuat komunikasi antara program python dan coppeliasim terputus, sekaligus memungkinkan pengamatan efek gaya, torsi, serta pergerakan bola secara lebih detail.
5. Program mendukung permainan secara bergiliran, sehingga dapat dipakai sebagai prototipe interaktif sederhana untuk simulasi permainan biliar berbasis fisika.

- Link Github :

  https://github.com/amrufal/Billiard_Coppeliasim/tree/main

- Linkedin :

  https://www.linkedin.com/posts/falah-amru-9a192a385_github-amrufalbilliardcoppeliasim-activity-737
  utm_source=share&utm_medium=member_desktop&rcm=ACoAAF7_5wkBHEm5eAEQ0al5sPz8ty6cgqgTe08