Mujin Backend System Challenges

Your task is to create a simple RESTful API server that allows the management of a collection of OpenRAVE robot files in COLLADA format.

Requirements

Try to meet as many of the following requirements as possible:

1. Start with a `debian:bullseye` Docker image and build on top of it.
2. Compile the `production` branch of OpenRAVE [1] and use it to manipulate robot files (hint: see examples [2]).
3. Build an HTTP server using either Python or C++.
4. Your HTTP server should support the following APIs:
    - List of all robots in the collection, including properties, such as `name`, `dof` (degree of freedom): `GET /api/robot` returns JSON response.
    - Get a particular robot properties: `GET /api/robot/filename` returns JSON response.
    - Add a new robot to the collection by uploading the robot file [3]: `POST /api/robot` includes the file in the request body, returns a JSON response.
    - Modify properties, such as `name`, of a robot in the collection: `PUT /api/robot/filename` takes JSON request body and returns a JSON response.
    - Download robot file of a robot in the collection: `GET /api/robot/filename/download` returns file content in response.
    - Remove a robot from the collection: `DELETE /api/robot/filename`.
    - BONUS: Get a preview image of the robot: `GET /api/robot/filename/preview` returns image content in response.
5. Add automated tests for your server.

Deliverables

1. All source code and documentation you have created, you should commit them to a private git repository and share the repository access.
2. A docker image that is pushed to docker hub and can be run by `docker run`.
3. Documentation on how to run and test your deliverables.

---

1. Please use `production` branch of OpenRAVE from GitHub at https://github.com/rdiankov/openrave. Compiling OpenRAVE is a major part of the challenge. Not all component of OpenRAVE needs to be built. Use `7dddd054628e42ab973bdbd1f9ab94535beb4d03` commit for `rapidjson`. If you want to use Python, you will need to compile with pybind11, and you will need to cherry pick these 2 commits `94824d68a037d99253b92a5b260bb04907c42355` and `98c9f77e5481af4cbc7eb092e1866151461e3508` in pybind11. ↩
2. Example usage of OpenRAVE in Python can be found at http://openrave.org/docs/latest_stable/examples/, you may also refer to examples in the OpenRAVE source. ↩
3. A collection of OpenRAVE robots in COLLADA format can be found at https://github.com/rdiankov/collada_robots, these are zip files that contain the actual COLLADA files. You may want to convert them to JSON before using them. Once loaded in OpenRAVE, you can do `env.Save("newFilename.json")` to save to JSON format. ↩