3D Drag Doll Application - Project Instructions

Objective

The objective of this challenge is to evaluate your technical skills across multiple dimensions, including design choices, logic implementation, code quality, component architecture, and software design principles.

Task Overview

Develop a comprehensive 3D drag doll application using **Svelte**, **Threlte**, and **Rapier** physics engine. The application has 2 modes: Editor and Run mode, during editor mode you can do as you wish to your scene, during Run mode you can interact with the assets as Drag doll behaviour.

Features

1. Scene Asset/Object Management UI

- Implement a complete **CRUD system** for managing scene elements:
  - Load local 3D assets
  - Add default Three.js primitives
  - Manage lights, cameras, and other scene objects
- Provide intuitive controls for asset manipulation (you can customize the appearance of the default threejs component if you want).

2. Drag and Drop Functionality

- Enable drag-and-drop interaction for assets from a management menu
- Ensure smooth asset placement within the 3D scene

3. Scene Tree Hierarchy UI

- Display a hierarchical view of all scene elements (lights, cameras, assets, etc.)
- Take inspiration from theater while implementing creative layout and visual design
- Provide clear organization and navigation of scene components

4. Property Editor UI

- Create an interface for modifying component properties, including:
  - Position and orientation
  - Illumination settings
  - Material properties
  - Other relevant object attributes

5. Rapier Physics Integration

- Implement core Rapier physics functionalities:
  - Collision detection systems
  - Rigid body physics simulation
  - Proper physics material assignment

6. Interactive Drag Doll Simulation

- Provide a "Run" mode that enables:
  - Mouse-based object manipulation during simulation
  - Real-time physics interactions
  - Object collision and response when dragged into proximity

Bonus Features

Advanced Joint Systems

- Implement joint constraints for multi-limb objects (e.g., robot assets)
- Ensure kinematic constraint compliance:
  - Limbs should respect maximum extension limits
  - Models should maintain structural integrity when constraints are reached
  - Prevent model breakage during extreme manipulations

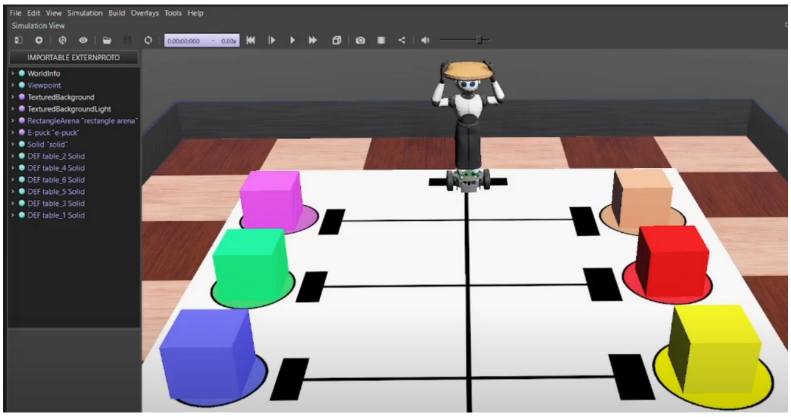
Design Guidelines

- **Creative Freedom**: You have complete autonomy over UI/UX design decisions
- **Inspiration Source**: We highly recommend studying **Webots** for design patterns and user interaction paradigms
- **Professional Standards**: Ensure clean, maintainable code architecture and intuitive user interfaces

Example references

- Webots interface and simple functionalities: https://cyberbotics.com/#webots

- Drag doll like behaviour: https://research.mels.ai/ide?mels=UnitreeGo1.qkazy
- https://zalo.github.io/mujoco_wasm/


File Edit View Simulation Build Overlays Tools Help
Simulation View
0:00:00:000 - 0.00x
IMPORTABLE EXTERNPROTO
WorldInfo
Viewpoint
TexturedBackground
TexturedBackgroundLight
RectangleArena "rectangle arena"
E-puck "e-puck"
Solid "solid"
DEF table_2 Solid
DEF table_4 Solid
DEF table_6 Solid
DEF table_5 Solid
DEF table_3 Solid
DEF table_1 Solid