Nuclear Pool Manipulator Simulator – Prototype

Team: StudentTechLab

Author: Waseem Imad Fanous

Eötvös Loránd University

Hepenix Task Designing a Functioning Nuclear Pool Manipulator

Date: 10/14/2025

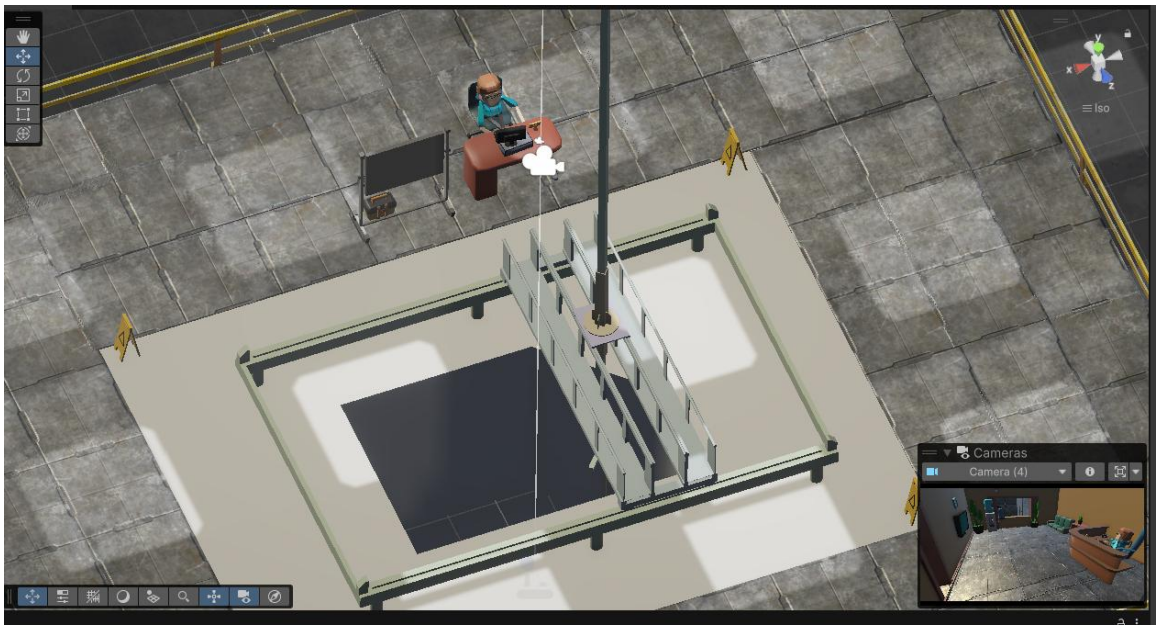
Contents

Executive Summary

This prototype delivers a Unity-based simulator for a four-degree-of-freedom (4-DOF) nuclear pool manipulator. It features clamped motion limits, a diagonal wall constraint to stop clipping, multiple camera perspectives, and an in-scene overlay menu that lets the operator switch pool modules on the fly through additive scene loading. The project was built with TechTogether Fall 2025 in mind, emphasizing smooth performance on a laptop, a clean modular structure, and a clear presentation—exactly what the competition values in both implementation quality and live demonstration.

Task Interpretation

The HEPENIX challenge called for a simplified manipulator simulator with four controllable axes (X, Y, Z, Fi). It must run in real time, support multiple viewing angles, and operate inside a convincing industrial environment with an interface simple enough for short demo sessions. The resulting tool needed to be stable, intuitive, and adaptable for future upgrades such as module swapping or additional human-machine interface (HMI) options.



-

System Architecture

Input Layer
Keyboard controls handle translation and rotation for now, with clear room for a future joystick setup. Each input channel feeds its own axis controller so that motion logic remains independent from device handling, improving test coverage and portability.

Motion Layer
Dedicated controllers for X, Y, Z, and Fi update each frame, adjusting target positions while respecting whether inputs are currently enabled. When the overlay menu is active, these inputs

suspend to avoid conflicting commands—a small but important detail for predictable demo behavior.

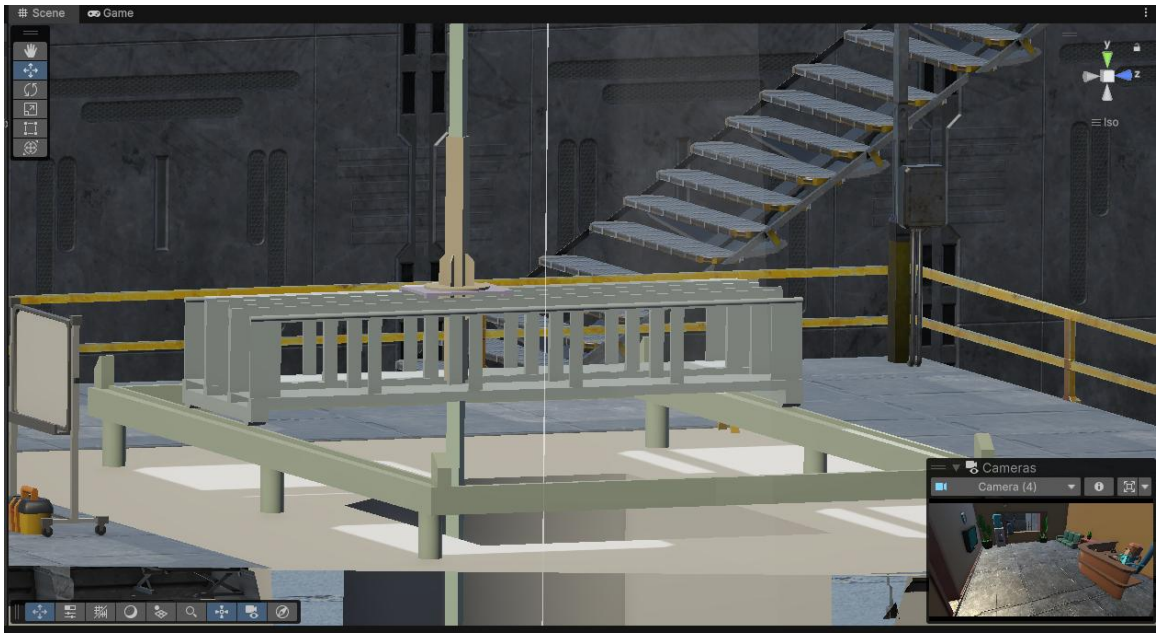
Constraint Layer
Movement is bounded by hard limits and by a slanted-wall glide rule. This rule projects the manipulator's X–Z coordinates onto a defined diagonal segment and advances the travel parameter according to actual speed and wall length, keeping the arm aligned with the barrier and eliminating mesh intersection without heavy physics processing.

View / UI Layer
A bank of cameras supplies different perspectives. The overlay interface pauses the scene, unlocks the cursor, and allows the judge to switch both cameras and environment modules through on-screen buttons, maintaining consistent control across views.

Scene Layer
Using additive scene loading, pool modules can be swapped during runtime without leaving the main scene. It shortens reset times and keeps memory usage stable—a major benefit when every minute counts in front of judges.

Environment and Modules

The main scene houses the operator area and static props. Each pool configuration sits in its own additive scene, loadable from the menu. Switching uses LoadSceneAsync in Additive mode, followed by setting the new scene active and unloading the previous one with UnloadSceneAsync. This structure allows immediate transitions while retaining a steady frame and memory footprint over long demonstrations.



Manipulator DOFs and Limits

- **X-Axis:** Rail translation bounded by adjustable inspector limits. The clamps maintain camera orientation and prevent the manipulator from leaving view—important for consistent presentations.
- **Y-Axis:** Vertical or bridge motion with symmetric clamping keeps the model centered and visible at all times.
- **Z-Axis:** Trolley translation that, within a defined region, follows the diagonal wall path. This prevents geometry overlap and illustrates precise positional control without a physics overhead.
- **Fi (Rotation):** End-effector rotation constrained to a realistic ergonomic range. Camera positioning highlights the tool's motion, reinforcing the safe and understandable operation expected in a competition setting.

Controls and Cameras

Pressing "1" moves the display to Camera 1—the main operator view. The menu mirrors these controls with clearly labeled camera buttons so judges can switch perspectives without memorizing shortcuts. Extra top and side cameras support motion validation and orientation clarity. The same menu also triggers module swaps, showing how constraints persist between different layouts, and also an extra fun workers camera perspective.

UI / HMI Main Menu

The overlay uses a Screen Space – Overlay Canvas with one EventSystem and a GraphicRaycaster to prevent duplicated input sources. Input handling is unified under the chosen UI module, resolving earlier cases of unresponsive buttons. Opening the menu reveals the cursor, unlocks its state, and sets Time.timeScale to 0 to pause safely while selections are made. Resuming restores normal play, providing a clean, predictable workflow for the presentation.

Implementation Highlights

- **Additive Modules:** LoadSceneAsync(scene, Additive) plus SetActiveScene ensures proper context and lighting. Old modules unload cleanly with UnloadSceneAsync, keeping transitions smooth.
- **Camera Switching:** Only the active camera remains enabled (or prioritized in a virtual stack). Buttons map directly to camera names for quick, error-free control.
- **Diagonal Glide:** Position data along the X–Z plane is projected to the wall segment, clamped, and advanced using normalized speed. The logic maintains precise alignment and stable motion without jitter.

Testing and Results

Bench tests confirmed reliable clamps on all axes, consistent glide behavior along the diagonal wall, and seamless additive scene swaps. The main scene, UI, and inputs remained intact across repeated cycles. Eliminating extra EventSystem components and outdated listeners fixed the intermittent click failures noted earlier.

Roadmap

Upcoming tasks include joystick integration, CSV-based macro playback to show repeatable paths, and live limit adjustments through the inspector-to-UI link. Additional polish will improve lighting, materials, and load times. Long-term extensions could involve optional Rigidbody. MovePosition for physics-accurate contact zones and an introductory overlay that guides first-time users through controls.

Build and Run

The simulator is built in Unity with URP and TextMeshPro for a consistent laptop deployment. All modules are listed in Build Settings, so name-based loading works without errors. A brief control card appears on-screen and, in the report, ensuring anyone—judge or viewer—can operate the prototype confidently.

References

- **HEPENIX TechTogether Fall 2025**: Official task description defining simulator scope, DOFs, and demo criteria.

- **Unity Documentation**: SceneManager APIs for additive loading and unloading, and UI/EventSystem guidelines for stable pointer input and Canvas setup.