HCR Simulator

IMSEA Part 2: Intelligent Methods for Scientific and Engineering Applications

COMP2S401 Service-Learning Project

August 2026

The Hong Kong Polytechnic University

Table of content

1 Service Context

Part 2 makes the HCR concept teachable before it becomes physical

The access problem.
Real robot practice depends on equipment, supervision, space, and time. A beginner also needs room to make mistakes without damaging hardware.

The Part 2 contribution.
The simulator turns the Haircut Control Robot concept into a browser-based learning environment that works without a robot or network connection.

Why simulation fits the task.
PolyU's AIR Lab describes digital twins as a way to design, test, refine, and train with robots before real-world deployment.

Service-learning requires reciprocity, not only a working app

Connect technology to a real need.
The project should reduce barriers to learning robot programming, while the intended users and partners shape what "accessible" and "useful" mean.

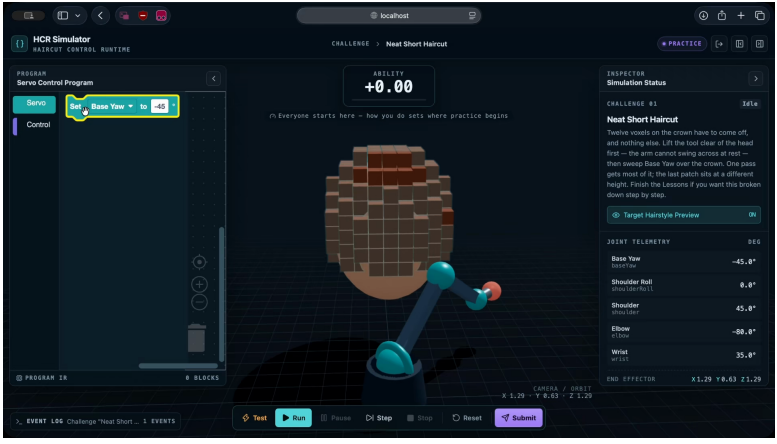
Apply academic knowledge.
Programming languages, kinematics, geometry, scoring, and interface design are combined in one learner-facing service.

Reflect and improve.
PolyU's service-learning framework links academic study, community service, reflection, and reciprocal benefit. Feedback must change the next version.

Keep the claim bounded.
This deck presents a simulated training and validation tool. It does not claim that a physical haircutting robot is safe for use on people.

2 Learner Experience

One screen closes the learning loop



The learner moves through a six-step cycle

1. Choose a guided lesson, solo challenge, or versus round.
2. Build a program with absolute joint-angle, Wait, and Repeat blocks.
3. Press Test for a fast, headless evaluation.
4. Press Run to watch the same program animate in 3D.
5. Inspect the cut, collision messages, telemetry, and score.
6. Revise the blocks and try again without consuming hardware time.

The loop rewards experimentation: a failed attempt becomes visible evidence for the next decision.

A small language keeps the reasoning visible

Three block types.
Learners set one joint to an absolute angle, wait for a duration, or repeat a sequence. Programs expand to at most 500 runtime commands.

Five controlled joints.
Base Yaw, Shoulder Roll, Shoulder, Elbow, and Wrist form a nested kinematic chain with challenge-defined ranges and speeds.

Data, not arbitrary code.
Blockly compiles to a validated Program IR. The simulator does not execute generated JavaScript or use `eval`.

Debugging is part of learning.
Pause, Resume, Step, Stop, Reset, source-block highlighting, and event logs make each command inspectable.

Eight lessons progress from movement to precision

Foundation.
First Cut introduces one command; *Mind the Head* makes collision avoidance an explicit part of the program.

Joint reasoning.
Reach Higher, *Straighten the Arm*, and *Angle the Tool* isolate Shoulder, Elbow, and Wrist decisions.

Precision.
Do Not Overcut and *Precision* show that stopping at the correct angle matters, even when a wider sweep looks nearly right.

Integration.
The Whole Crown combines setup, multiple sweeps, and a height change. Every lesson target is derived from a program that reaches 100.

The same engine supports learning, practice, and competition

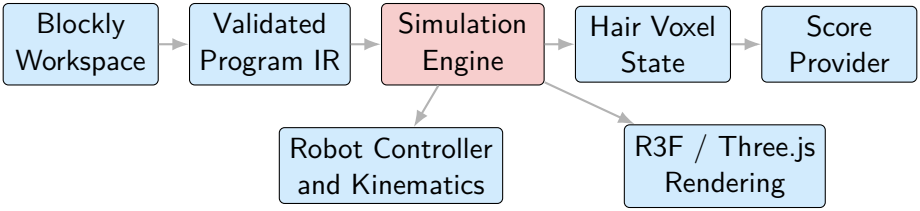
Tutorial and lessons.
Eight guided tutorial steps teach the language, followed by eight constructively solvable challenges that each isolate one skill.

Solo Practice.
Offline mode uses a clearly labelled fixed sequence. With the backend connected, the next challenge can be selected from the learner's estimated ability.

Versus Round.
Players receive the same hidden challenge at the same time. Online results come from server replay; the offline version is labelled practice with scripted bots.

3 Engineering Method

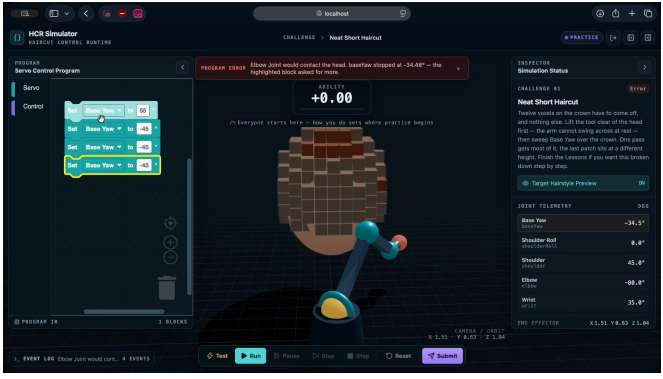
Clear boundaries keep one result reproducible



One engine, two speeds.
Animated Run and headless Test execute the same commands. Rendering speed does not change the score, command count, or estimated duration.

Providers isolate deployment choices.
Local services keep the learning loop offline; HTTP providers enable adaptive sessions and authoritative multiplayer without rewriting the workbench.

Unsafe simulated poses are refused at the last safe angle

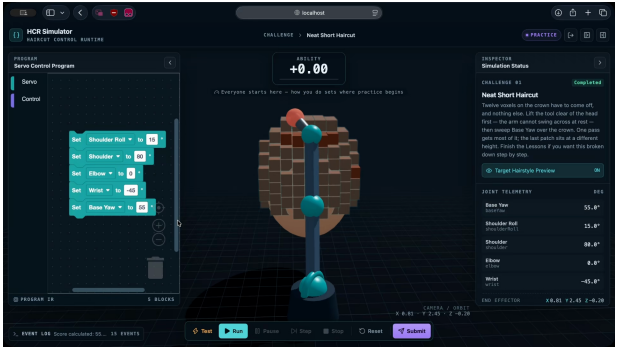


Deterministic checks.
The robot is approximated by spheres and capsules, while the head is an expanded ellipsoid.

Recoverable refusal.
Motion advances in small angular substeps, refines the boundary, keeps the last safe pose, and highlights the source block.

Scope.
Simulation only, not a physical interlock.

The haircut becomes measurable geometry



241 discrete hair voxels.
Hair exists as a logical set as well as a 3D mesh, so removal and rendering cannot silently disagree.

Swept contact.
The tool's path is tested against voxel bounds, preventing fast movement from tunnelling through hair.

A precise current task.
The default crown challenge asks for exactly 12 voxels to be removed.

Accuracy leads the score, while efficiency shapes good solutions

Completion measures the requested cut.
Let *Asked* be Initial minus Target, and *Removed* be Initial minus Result. Completion is their intersection-over-union:

$$\text{Completion} = \frac{|\text{Asked} \cap \text{Removed}|}{|\text{Asked} \cup \text{Removed}|} \times 100$$

The final score uses transparent weights.

$$\text{Final} = 0.60\,\text{Completion} + 0.25\,\text{Efficiency} + 0.15\,\text{Time}$$

Doing nothing scores zero completion, and removing hair that should stay is penalised in the union.

Every shipped target is checked before a learner sees it

Winnable by construction.
The current 12-voxel crown target and all eight lesson targets come from solution programs that actually execute in the engine.

Reachability is measured separately.
The committed geometric audit records 241 total voxels, 150 reachable voxels, and a 91-voxel dead zone.

Tests protect the guarantee.
Every authored target must request only voxels in the measured reachable set, and solution-derived targets are replayed to confirm a perfect result.

Why this matters.
A previous hand-drawn item could look reasonable while asking for hair the arm could not cut. The design now treats solvability as evidence, not an assumption.

4 Evidence and Reflection

The current build is demonstrable, with clear remaining work

Fresh automated verification.
Type checking, linting, and the production build pass. The current run passed 135 unit and integration checks plus 7 Chromium end-to-end scenarios.

The end-to-end path is covered.
Tests exercise reproducible scoring, collision refusal, pause and step controls, reset behaviour, target preview, WebGL recovery, and desktop control visibility.

The boundary is documented.
Manual visual acceptance in Chrome and Edge at 1280 by 720 and 1920 by 1080 remains pending, and Blockly drag-and-drop authoring is not automated end to end.

Readiness claim.
The software is ready for a supervised service-learning demonstration, not for unsupervised physical deployment.

Why this is service-learning, not just software

Community need and reciprocity.
Confirm the barriers learners and partners actually face, then let their feedback define priorities and acceptable trade-offs.

Academic knowledge in service.
The project applies programming, geometry, simulation, assessment, and human-computer interaction to a concrete learning environment.

Ethical and accessible delivery.
Plain-language guidance, visible errors, offline operation, privacy notices, and honest safety boundaries are part of the service, not optional polish.

Reflection becomes the next specification.
Record what users could not understand or complete, analyse why, and turn that evidence into the next design and service plan.

The next iteration should begin with users, then approach hardware

Validate learning and accessibility.
Observe first-time users, test keyboard and screen-reader paths, and measure where learners stop, recover, or ask for help.

Co-design the service model.
Agree with partners who facilitates sessions, how feedback is collected, what data may be retained, and how participants benefit after the project ends.

Finish deployment readiness.
Complete cross-browser visual acceptance, reduce the large JavaScript bundle, and test lower-powered classroom devices.

Separate simulation from physical safety.
Any connection to real machinery requires independent hardware limits, emergency stops, consent procedures, and supervised trials.

Part 2 turns simulation into a service-learning instrument

Learn.
Blockly makes robot programming approachable, while live 3D feedback reveals how code, geometry, and motion interact.

Experiment safely in the digital environment.
Deterministic collision refusal, instant testing, and resettable challenges let learners explore without consuming robot time.

Improve with evidence.
Transparent scoring, solvable targets, automated verification, and community feedback create a cycle from technical work to service and back again.

Program the arm. Observe the result. Reflect. Improve.

Evidence used in this deck

PolyU service-learning context.
COMP Service-Learning and PolyU Service-Learning Requirement.

Why digital twins support training.
PolyU AIR Lab: Digital Twin and Robot Simulation.

Project and implementation evidence.
HCR Simulator Frontend, its current code, tests, reachability fixture, and the recorded 49.8-second simulator demonstration.

Presentation system.
Azusa LaTeX Beamer Template.