

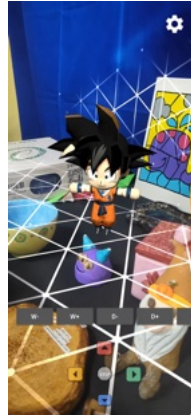
Programming with the Physical World: Object-Aware AR Interactions for Children

Aranis Das*
Stony Brook University

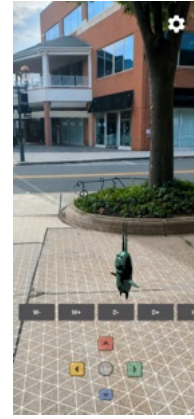
Shubendu Das†
Steamworks Studio



Pixel capybara



Goku



Plankton

Figure 1: Android AR prototype with different avatar styles placed in everyday environments. The flow begins with placing a character in the physical scene, then exploring how different virtual avatars can act as programmable agents for object-aware AR interactions. These examples illustrate the prototype’s emphasis on sandbox-like character play before connecting characters to physical-world objects and behaviors. The pixel capybara avatar uses the Minecraft Capybara (Capybara) model by rexitto, licensed under Creative Commons Attribution [11].

ABSTRACT

Physical objects can be powerful anchors for children’s AR authoring because they make spatial interaction concrete: learners can see, move around, and program against objects in their own environment. Building on constructionist tools such as Scratch and Capybara, we present an object-aware mobile augmented reality prototype that explores how everyday objects can become programmable targets for playful character-object interactions. In the prototype, learners place animated characters in an AR scene, detect everyday objects such as cups, apples, and bottles, and transform those detections into simple interactable three-dimensional proxies. Characters can then perform simple behaviors such as jumping onto, jumping over, or bouncing off these object proxies. The system intentionally uses lightweight geometry and visible interaction rules rather than high-fidelity reconstruction, making perception, anchoring, scale, collision, and event-driven behavior easier to reason about. This paper describes the design motivation, system architecture, interaction model, and planned classroom-oriented evaluation of the prototype as a low-barrier way to connect visual programming ideas with physical-world AR interaction.

Keywords: Augmented reality, XR education, spatial computing, object-aware AR, mobile AR, learning by making, 3D interaction.

1 INTRODUCTION

Capybara introduced a compelling direction for children’s AR authoring by bridging constructionist programming with the fantasy

of creating interactive augmented worlds [12]. Rather than positioning children as passive consumers of AR, Capybara frames them as creators who customize virtual characters, animate them, and program interactions with the surrounding physical world.

We build on this direction through a mobile Android AR prototype that asks how physical-world objects can become game-like, programmable elements inside children’s AR worlds. As shown in Figure 1, learners first place AR avatars into everyday environments, then connect those characters to detected objects and behaviors. For early prototyping, we included recognizable popular-media-inspired avatars, including Goku from *Dragon Ball* and Plankton from *SpongeBob SquarePants*, to explore how familiar character forms can make AR agents feel playful and immediately legible to learners [1, 8]. Our prototype extends Capybara’s object and zone interactions with lightweight 3D object proxies and simple character behaviors such as jumping onto, jumping over, or bouncing off detected objects.

This shift treats the physical environment as programming material. A cup, apple, bottle, or keyboard is not only something the camera recognizes; it can become an object that the character responds to, navigates around, jumps on, or uses in a challenge.

This paper positions the prototype as an exploratory extension of Capybara’s physical-world interaction layer. We contribute an Android-based object-proxy interaction model for turning detected physical objects into interactable AR targets for beginner-friendly character behaviors.

2 RELATED WORK

Constructionist programming environments show the value of giving children low-floor, high-ceiling tools for personally meaningful computational projects [10]. Capybara extends this tradition into mobile augmented reality, connecting the constructionist spirit of Scratch with character customization, animation, and physical-

*e-mail: aranis.das@stonybrook.edu

†e-mail: shubendu@steamworksstudio.com

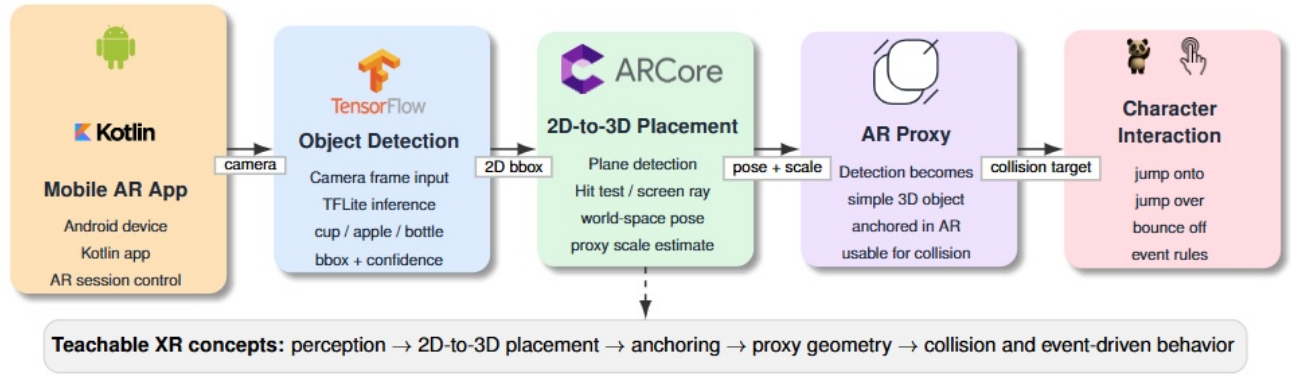


Figure 2: Detection-to-proxy pipeline from camera frame to interactable AR object.

world sensing [12]. Our work follows this framing, but focuses on how everyday physical objects can become programmable material inside an AR scene.

Earlier AR programming systems also connect child-friendly authoring with physical context. AR Scratch extended Scratch so children could mix real and virtual spaces through camera-based interaction, while AR-Maze used tangible blocks and AR feedback to support young children’s programming activities [9, 4]. Work on tangible programming and physical interactive storyrooms similarly shows how children can compose computational behavior through objects, blocks, and room-scale physical interaction rather than only through screen-based code [3, 7].

Prior work suggests that young people want AR creation tools that blend imagination with real-world context, make actions visible, invite social connection, and support learning about the physical world [5]. This motivates object-aware authoring tools that turn recognized objects into meaningful parts of an authored world.

This view is reinforced by mobile AR systems that treat physical context, bodies, objects, and shared space as resources for playful, embodied, co-located interaction [2]. Recent tangible AR prototyping work further shows the value of using everyday objects as interaction material for rapid AR authoring [6]. Our prototype adopts this view by treating detected objects and placed AR proxies as resources for play, movement, and programming.

Together, this work motivates our focus on Capybara’s object and zone interactions as a starting point for explicit object proxies and character-object behaviors [12, 5, 2].

3 SYSTEM ARCHITECTURE

Our prototype translates Capybara’s physical-world interaction idea into an Android-based mobile AR pipeline. Capybara was implemented as an iOS application using SwiftUI, ARKit, and RealityKit, and its study was conducted on an iPad Pro (13-inch, M4 Chip, 2024) [12]. That platform choice gave Capybara access to Apple’s AR stack, including ARKit tracking, body motion capture, and high-end tablet hardware. Our prototype explores a complementary question: how much of Capybara’s object-aware, child-facing AR interaction model can be made legible on a commodity Android AR device using ARCore, Kotlin, and on-device object detection?

This shift is not a claim of feature parity with Capybara. Rather than emphasizing high-fidelity reconstruction or advanced character generation, the Android prototype focuses on a visible pipeline: camera input, object detection, 2D-to-3D placement, proxy geometry, and character-object interaction. Figure 2 shows how the prototype turns a camera detection into a selected object, an AR proxy, and finally an interactable scene element.

3.1 Platform Shift: From ARKit to ARCore

Capybara provides the closest system foundation for our work because it shows that children can use a block-based AR environment to customize virtual characters and define interactions between those characters and physical surroundings [12]. In Capybara, object detection and zone detection are used as sensing blocks: learners select a target object or region, the system detects whether the virtual character touches that physical referent, and subsequent code blocks execute. Their implementation uses YOLOv11s for object detection, runs inference every 10 frames with a confidence threshold of 0.6, and detects touch by raycasting from sampled points in the selected object’s 2D bounding box into the AR scene [12].

Our Android prototype preserves the same conceptual arc but changes the implementation substrate. The app is built as a native Kotlin AR application using ARCore for tracking, plane detection, hit testing, and world-space placement. Object detection is handled with an on-device TensorFlow Lite model over the live camera frame. The current prototype has been tested on a Pixel-class ARCore-capable Android phone, making it closer to a low-cost workshop or classroom deployment scenario than a high-end tablet-centered setup. This platform shift means the prototype cannot assume LiDAR-quality depth, full scene reconstruction, or Apple’s body-tracking stack, so it relies on plane hit tests, screen rays, target filtering, and visible proxy objects.

3.2 Detection-to-Proxy Pipeline

The system loop begins after ARCore has established tracking and detected usable planes. The learner places a character, selects a target object class, and starts object detection. The detector returns labels, confidence scores, and 2D bounding boxes from the live camera frame. The app filters these detections by the selected class and locks onto the most stable target.

Once a target is selected, the app projects the center of the 2D bounding box into the AR view and uses ARCore hit testing to estimate a world-space pose. The detection is then represented as a simple 3D proxy at that pose. This proxy is not a reconstruction of the object; it is a lightweight spatial handle that lets the system attach interaction behavior to the detected physical object.

3.3 Character-Object Interaction

After the proxy is placed, it becomes the target for simple character behaviors. The current prototype supports scripted actions such as jumping onto an object, jumping over it, bouncing off it, or moving around it. These behaviors use the proxy as the spatial reference for motion and collision checks.

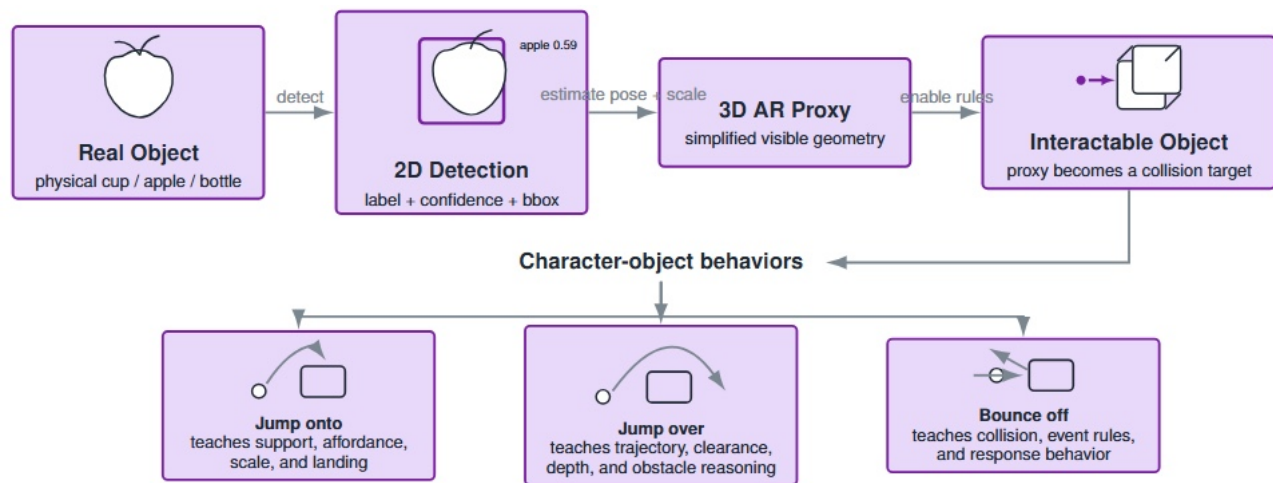


Figure 3: Object-aware AR flow from real object detection to proxy creation and character-object behaviors.

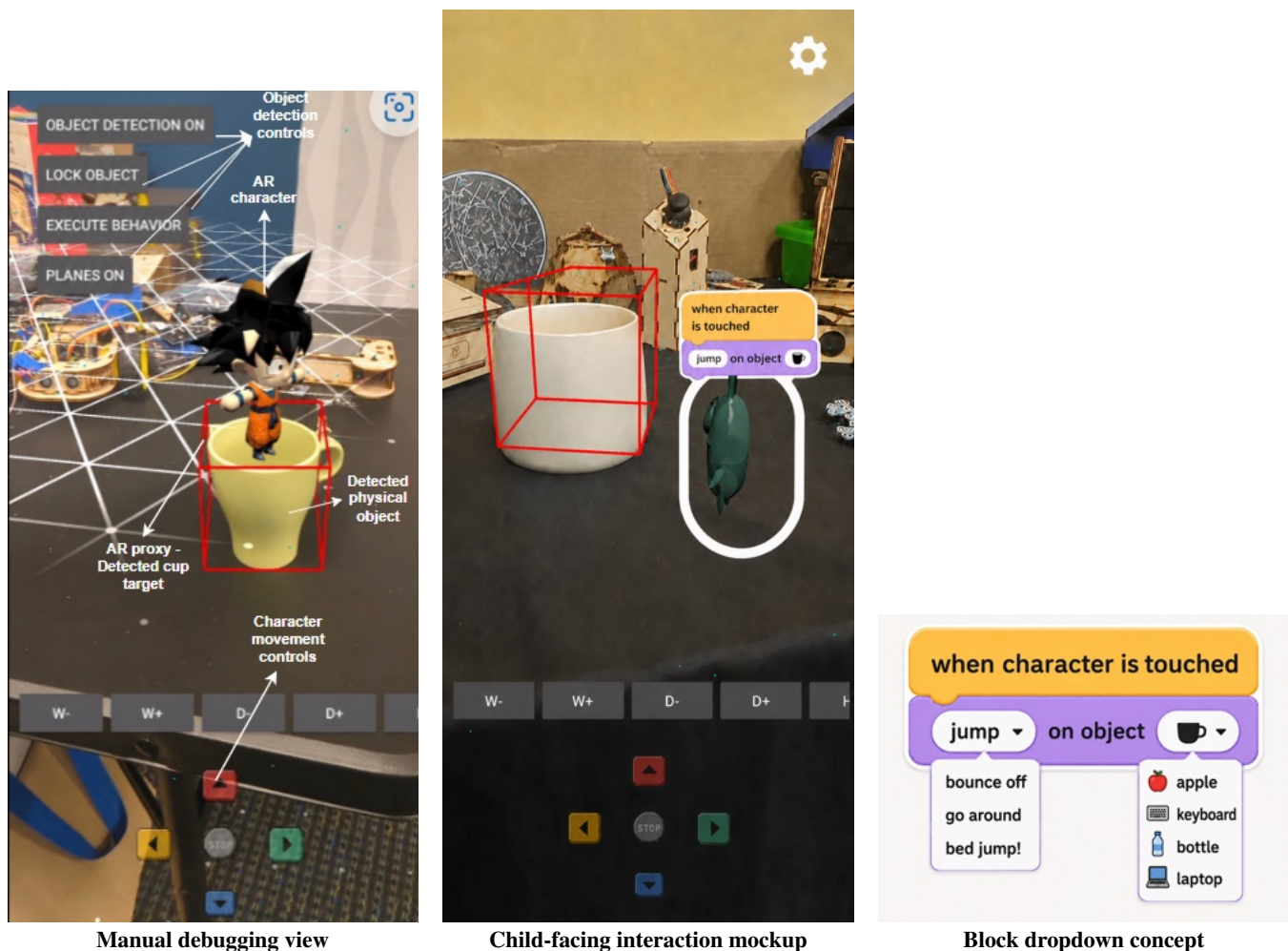


Figure 4: Prototype views used to guide the design goals. The manual-control view exposes object detection, locking, execution, AR planes, character motion controls, and the detected cup proxy. The child-facing mockup shows how the same pipeline could become a block-based interaction where a character jumps on a detected cup. The dropdown concept shows how future blocks could expose both object choices and executable behaviors.

The behaviors are currently triggered through manual controls rather than a full block editor.

4 DESIGN GOALS

Our design goals come from reading Cappybara as both a system and an invitation for further exploration. Cappybara shows that children can author AR experiences by combining block-based programming, virtual characters, and sensing blocks that connect those characters to the physical world [12]. We focus on two extensions of that idea.

First, we explore a more broadly deployable Android path. Cappybara was built around Apple's AR ecosystem and high-end tablet hardware, while our prototype uses Kotlin, ARCore, TensorFlow Lite, and a Pixel-class ARCore-capable phone. This shift lets us examine how Cappybara-like interaction ideas might transfer to a lower-barrier mobile AR setting.

Second, we make object detection explicitly interactable. Cappybara notes that its object detection pipeline is limited in object coverage and does not infer complete 3D bounding boxes [12]. Our response is not full reconstruction, but an intentionally approximate proxy that turns a detected object into a spatial target for character behavior.

Figure 4 shows the design bridge from the current debug interface to a future child-facing block interface. The manual-control panel exposes detection, locking, plane visualization, movement, and behavior execution. The Plankton concept and dropdown mockup show how the same pipeline could become a Cappybara-like rule such as "when character is touched" followed by "jump on object," with object choices such as cup, apple, keyboard, bottle, and laptop, and behavior choices such as jump onto, jump over, bounce off, go around, and bed jump.

This leads to the central design goal: physical objects should become programmable parts of an AR world. A cup should not only be recognized by the camera; it should become a spatial object that learners can reason about, attach rules to, and use in playful challenges.

5 PLANNED EVALUATION AND DISCUSSION

We plan to evaluate a more mature version of the prototype in a short workshop or classroom activity with learners approximately 10–14 years old. Participants may be novices or have limited prior experience with Scratch-like block programming; prior programming experience will be recorded, while prior XR development experience will not be required. Unlike the current implementation, which uses manual controls, execute buttons, and debug UI, the evaluated version will move closer to Cappybara's child-facing authoring model by implementing blocks for object selection and character behaviors. Participants will create a small object-aware scene by selecting an everyday object, inspecting and adjusting its proxy, programming a character to jump onto or avoid it, predicting the outcome, and debugging a deliberately mis-scaled or mis-anchored interaction.

Our research question is: *How does authoring with visible 3D object proxies support learners' explanations of the relationships among object detection, anchoring, scale, collision, and event-driven character behavior?* Intended learning outcomes are that learners can distinguish a camera detection from its world-space proxy, explain how anchoring and scale affect collision, trace an event rule from a sensed object to a character action, and revise a proxy or rule after an incorrect prediction. We plan to collect pre/post concept questions, task completion and debugging success, authored programs and proxy settings, screen recordings and interaction logs, facilitator observations, and brief interviews. Analysis will compare pre/post concept responses and task measures descriptively and, if the sample supports it, with paired nonparametric tests. Learner explanations, debugging strategies, and miscon-

ceptions will be coded qualitatively and interpreted alongside their artifacts and logs, with prior programming experience considered when examining patterns. These procedures are future work; no participant study or results are reported here.

The prototype highlights a pedagogical tradeoff in educational XR tools. High-fidelity reconstruction can make an AR scene feel realistic, but it can also hide the abstractions learners need to inspect. A simplified box proxy exposes what the system detected, where it anchored the detection, the scale it assigned, and the volume used for collision. Learners can compare the proxy with the physical referent, resize or reposition it, and observe how behavior changes, making hidden XR state available for prediction, testing, and revision. In constructionist terms, the proxy becomes editable material through which learners build and debug a personally meaningful world [10, 12]; connecting that material to movement around physical objects also supports embodied reasoning about spatial relationships [2].

The current system remains limited. Object depth is approximate, character behaviors are scripted rather than physically simulated, the supported object set is small, and the full block editor has not yet been implemented.

REFERENCES

- [1] Bird Studio/Shueisha and Toei Animation. Dragon Ball Official Site. <https://en.dragon-ball-official.com/>, 2026. Accessed 2026-07-06.
- [2] E. Dagan, A. M. Cardenas Gasca, A. Robinson, A. Noriega, Y. J. Tham, R. Vaish, and A. Monroy-Hernandez. Project irl: Playful co-located interactions with mobile augmented reality. *Proceedings of the ACM on Human-Computer Interaction*, 6(CSCW1):1–27, 2022. doi: 10.1145/3512909
- [3] M. S. Horn and R. J. K. Jacob. Designing tangible programming languages for classroom use. In *Proceedings of the 1st International Conference on Tangible and Embedded Interaction*, TEI '07, pp. 159–162, 2007. doi: 10.1145/1226969.1227003
- [4] Q. Jin, D. Wang, X. Deng, N. Zheng, and S.-C. Chiu. Ar-maze: A tangible programming tool for children based on ar technology. In *Proceedings of the 17th ACM Conference on Interaction Design and Children*, IDC '18, 2018. doi: 10.1145/3202185.3210784
- [5] A. Liaqat, F. Liu, B. Berengard, J. Cao, and A. Monroy-Hernandez. Understanding young people's creative goals with augmented reality, 2024.
- [6] K. Monteiro, R. Vatsal, N. Chulpongsatorn, A. Parnami, and R. Suzuki. Teachable reality: Prototyping tangible augmented reality with everyday objects by leveraging interactive machine teaching. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, CHI '23, 2023. doi: 10.1145/3544548.3581449
- [7] J. Montemayor, A. Druin, G. Chipman, A. Farber, and M. L. Guha. Tools for children to create physical interactive storyrooms. *Computers in Entertainment*, 2(1), 2004. doi: 10.1145/973801.973821
- [8] Nickelodeon. SpongeBob SquarePants. <https://www.nick.com/fan-hub/spongebob-squarepants>, 2026. Accessed 2026-07-06.
- [9] I. Radu and B. MacIntyre. Augmented-reality scratch: A children's authoring environment for augmented-reality experiences. In *Proceedings of the 8th International Conference on Interaction Design and Children*, IDC '09, pp. 210–213, 2009. doi: 10.1145/1551788.1551831
- [10] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, and Y. Kafai. Scratch: Programming for all. In *Communications of the ACM*, vol. 52, pp. 60–67, 2009. doi: 10.1145/1592761.1592779
- [11] rexitto. Minecraft Cappybara (Capibara). Sketchfab model page, 2025. Sketchfab 3D model, Creative Commons Attribution license. Accessed 2026-07-06.
- [12] L. Zhang, S. Zhou, A. Liaqat, T. Mak, B. Berengard, E. Qian, and A. Monroy-Hernandez. Empowering children to create ai-enabled augmented reality experiences. In *Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology*, UIST '25, 2025. doi: 10.1145/3746059.3747662