CatchAR: Prototyping Partial Object Manipulation, Naturalistic Throwing Interactions, and Intuitive Navigation Systems with AR Glasses

Alejandro Romero*

Brown University

ABSTRACT

Presently, fully immersive augmented reality (AR) experiences are few and far between, with many of the most popular AR interfaces relying on user input via smartphone touchscreens as opposed to more naturalistic interactions such as real-time hand tracking. CatchAR leverages Snap Spectacles, a pair of AR glasses capable of applying a vast array of emerging computer vision techniques to create immersive AR environments, to expand upon prior research at the Brown HCI Lab. The project allows users to throw objects via real-time hand tracking and explore AR environments using a responsive and intuitive navigation system.

Keywords: Augmented Reality, Hand Tracking, HCI

1 INTRODUCTION

During my graduate studies at Brown, I worked in the HCI Lab with professor Jeff Huang and Jing Qian to develop CatchAR, a demo leveraging techniques from the lab's Portal-ble project to create a truly immersive AR experience.
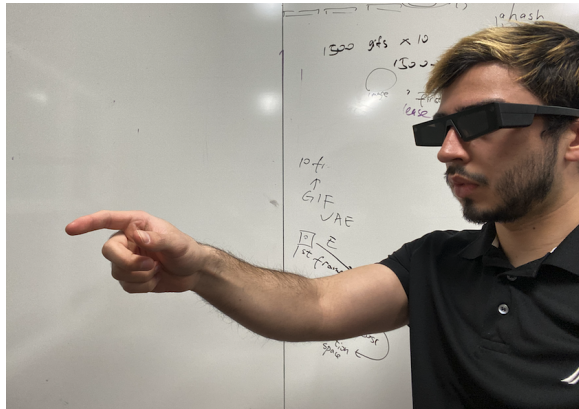

Figure 1: The small profile of the 2021 Snap Spectacles allow them to be worn by users as naturally as a pair of prescription glasses

Most current mainstream AR experiences are not truly immersive; that is, they rely on users to provide input via interfaces on their smartphones to interact with virtual objects projected onto the real world. However, Snap Inc.'s 2021 Spectacles are an example of true augmented reality. The glasses are able to project 3D environments into the user's world and support naturalistic interactions, such as hand tracking, that allow users to become truly immersed in an AR experience.

*e-mail: alejandro_romero@alumni.brown.edu

Figure 2a (Left), Figure 2b (Right): Examples of a user throwing Pokéballs in AR with their own hand.

CatchAR addresses this by taking work done for the Portal-ble project and generalizing its use cases to AR glasses. Portal-ble currently allows users to use their smartphones to engage with virtual worlds via real-time hand tracking. CatchAR takes this method one step further by eliminating the need to hold one's phone. Instead, CatchAR is presented right in front of a user's eyes, freeing their hands for a more genuine AR experience (**Figure 1**).

2 APPROACH

Pokémon Go is the biggest augmented reality app currently available. It allows users to explore their environments and interact with virtual objects projected therein. However, these features are held back by their reliance on handheld devices and limited screen space.

CatchAR takes AR navigation and object manipulation to a new level, tying these experiences to the user's physical surroundings via a lightweight head-mounted display. However, due to the shift to a device with more limited compute power, I had to re-imagine how these features would be implemented.

For instance, partial object manipulation is achieved via real-time hand tracking. Because a touchscreen is no longer needed, the potential area for hand interaction becomes larger, allowing for throws that feel much more human. Further, Pokémon Go currently utilizes a highly detailed map view to display Pokémon in the area. Inspired by first-person video games, I implemented a navigation system that uses relative coordinates of objects within the scene to show a user where to go next via a mini-map and 3D arrow, which take up a fraction of a user's view, allowing them to still see their environment and the virtual objects anchored within it.

3 ARCHITECTURE AND IMPLEMENTATION

3.1 Setup

I used Lens Studio, Snap Inc.'s proprietary development environment, to develop the project. All code was written in JavaScript, and assets were edited using Blender prior to being imported into the final project directory. The experience (known on Snapchat as a lens) is supported on both Spectacles and smartphones via the Snapchat app. It is optimized for both platforms.
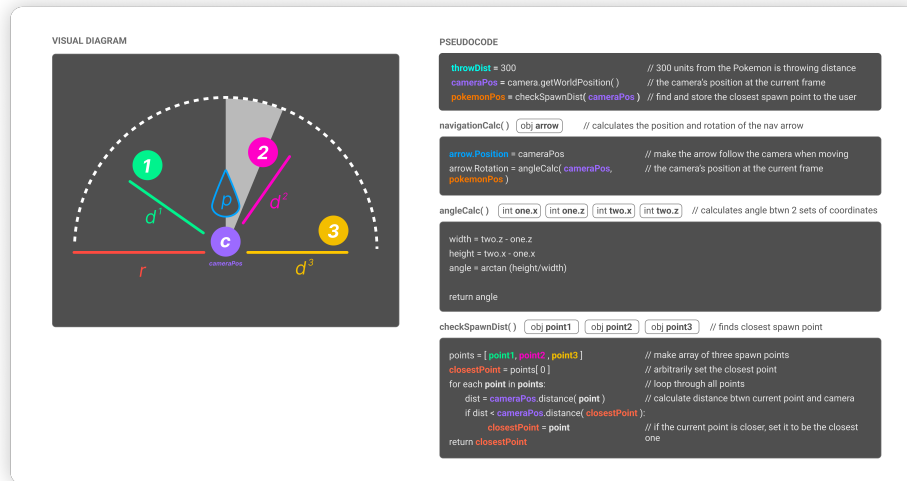
Figure 3: Diagram and accompanying pseudocode for the real-time navigation system graphically leading users to the nearest Pokémon spawn location.

3.2 Throwing in AR

Lens Studio currently has a template for using their hand tracking package, which I used as a foundation for the lens' solution to throwing Pokéballs to catch Pokémon. Using the device's camera feed, the algorithm uses distinct hand features (i.e. joints of each finger) to approximate the real-world location of the user's hand. These coordinates are then normalized to correspond to the virtual world such that the user's hand can manipulate rendered objects.

Previous work on Portal-ble included the conception of a throwing algorithm that calculates the expected velocity and trajectory of a projectile given the user's hand motion data. To do this, I created a ring buffer of the user's ten latest hand positions and calculated a velocity vector based on the average of these positions, which was then fed into a physics algorithm to make the ball move.

This approach was necessary as Lens Studio did not have a physics engine at the time that development on this project started. However, once Lens Studio's physics engine was released, we refined the formula and fed our velocity vector into this physics simulation instead, providing more compelling results. This also allowed me to create more realistic interactions with the environment, using rigid body colliders to have balls bounce off of Pokémon and the ground plane upon contact.

The act of throwing was an important consideration in itself, as determining an object's point of release is a deceptively nontrivial task. The queue of hand positions is updated every frame. A release event in our lens is defined as a frame in which the tip of the thumb and tip of the index finger exceed a certain distance from one another. Because of the variability in human hand sizes, this distance is calculated proportionally to the size of the rendered hand. Once a release is detected, a Pokéball, which is initialized as a child of the hand's transform, becomes un-parented, allowing for the physics engine to act upon it as per the velocity vector calculations provided by our calculations. The result is a method for realistically throwing virtual objects that generalizes to different throwing movements (e.g. overhead or underhand throws) *(Figure 2)*.

3.3 Navigation

Initially, I wanted to implement a path finding system that would route users to the nearest Pokémon by projecting a path onto the ground *(Figure 5)*. However, given the limitations of Lens Studio,

I soon realized that a I would have to consider a different solution. This resulted in two features: a mini-map showing users the relative location of Pokémon spawn locations and a 3D arrow indicating which direction the nearest of these spawn locations was in.

The mini-map was implemented using an orthographic camera above the scene that can see UI objects hidden from the lens' main camera. These UI objects were parented to their 3D counterparts in the 3D scene, so they moved with them upon scene updates. This allowed for optimization in the lens by significantly reducing the number of calculations that had to be performed during each frame update. The center icon of the mini-map moves in response to the device's orientation.
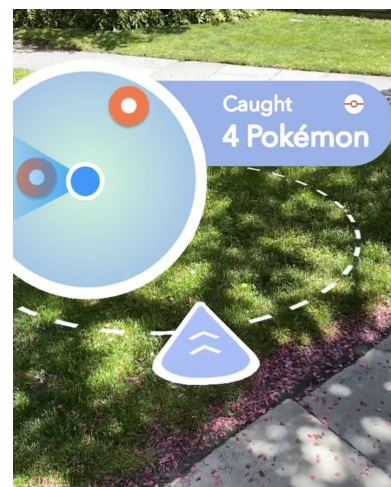

Figure 4: The lens' UI, which includes a mini-map, directional arrow indicator, and a counter recording the number of Pokémon caught.

The 3D arrow indicator was more challenging in implementation, but a useful asset in enhancing the user experience. At any given time, three potential Pokémon spawn locations are visible. These spawn locations are randomized within 500 units of the main

camera. The arrow indicator is rendered by taking these three spawn locations and calculating the one with the least distance to the camera's current position. Its orientation is then calculated by finding the angle between the camera coordinate and the nearest spawn point. If the user travels closer to a different spawn point, the calculation is rerun, and the arrow is updated in real-time *(Figure 3)*.

Once a user catches a Pokémon, the spawn point disappears and is re-randomized to a new location within 500 units of the user. This allows for the illusion of continuous exploration while constraining the number of objects instantiated in the scene.
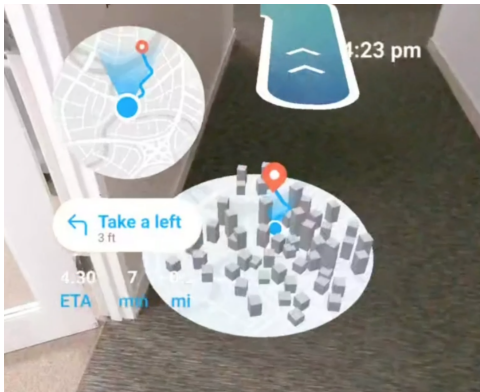

Figure 5: A test lens I developed to prototype routing navigation in AR.

3.4 Gamification Features

To enhance the user experience, I added features to gamify the experience and reward the user for trying out the lens.

When a user gets close to a Pokémon spawn point (300 units away), the navigation features are toggled off, and a Pokémon appears (randomized from a list of four possible Pokémon meshes). This mode toggling was implemented to optimize the use of limited computer power on Spectacles and also acts as a way to de-clutter the screen from a user's perspective.

Once a user is within this throwing distance, they can then use hand tracking to throw a Pokéball. When the ball's rigid body makes contact with that of the Pokémon, audiovisual cues are played to signify success, and the user's score is incremented. Additionally, the Pokéball changes color with subsequent catches.

I also learned from user feedback that it would be beneficial to implement a tutorial screen to educate users about hand tracking (a fairly new interaction method to most people), as well as the other features of the lens. This took the form of a brief 10 second video played once the lens is initialized.

4 Conclusion and Future Work

CatchAR is a step towards fully immersive interactive experiences that bridge the gap between the virtual and physical worlds. By leveraging fully AR glasses such as Snap Spectacles, we eliminate the need for handheld devices, allowing users to interact with virtual objects almost as intuitively as they would with real ones.

There are many potential future extensions of the current work, including avenues such as adaptive throwing algorithms to account for human inaccuracy, the use of speech recognition, enhancing the mini-map via surface detection and visualization, realistic occlusion, the introduction of social features, among many more. The framework developed for CatchAR can be modified to create similar experiences leveraging waypoint navigation, such as for educational purposes.

5 Useful Links

- GitHub Repo: `https://github.com/brownhci/Spectacles`

- Interactive Demo: `https://lens.snapchat.com/9eff1581da8e4ba7a5208787e08348f2`

- Case Study: `https://alejandroromero.me/proj/spectacles.html`

- Portal-ble: `https://portalble.cs.brown.edu/`