Group 7:

Liem Nguyen

Bruce Jiang

Meisheng (Jenny) Liu

Dobutsu Daycare Development Report

Programming Handheld Systems (CMSC436) is a course that provides students with the fundamental principles that are involved with the programming of mobile phones, tablets, and personal digital assistants. Naturally, students will be tasked with multiple projects throughout the course that will assist in their comprehension of the main concepts of limited display size, power, memory, and CPU speed. As a 400 level course, Programming Handheld Systems task students with a semester-long project to apply in-class and out-of-class concepts to develop an interactive app of their own. Students form their groups, brainstorm ideas for innovative apps, and begin their semester-long development for the app they've worked so hard for. This semester's project is a great idea for students to step away from detailed project descriptions and create a project of their own. As future computer scientists, it is important to learn how to innovate, develop new ideas, and act on those ideas. Throughout this paper, I will be describing the evolution of our ideas, the back-and-forth changes during our development, any necessary compromises, and our final product.

After forming my group at the beginning of the semester, it took a while for us to figure out what to do. We gathered together and discussed if any of us had completed personal projects and were willing to share. The idea was to gauge how much programming experience each of us had outside of the courses at UMD to get a general sense of how difficult and complex our group project will be. It turns out we all had a decent amount of experience to create a fairly challenging app. The next step was to look for inspiration and gather the tools necessary to achieve our goals.

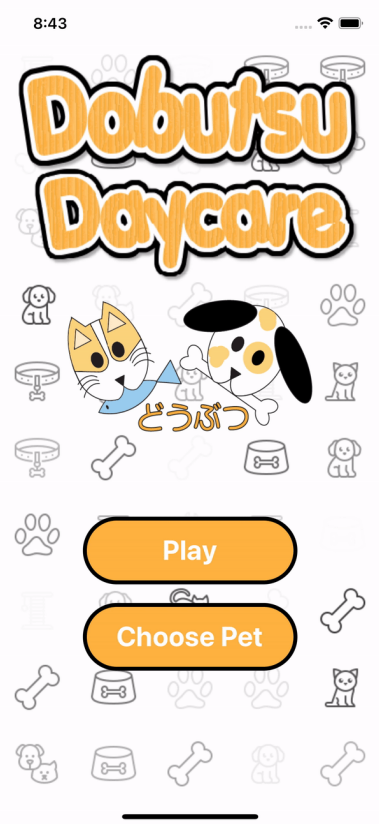
Initially, we had planned to create a simple game like Flappy Bird, a fast-paced game that involves tapping to move like Temple Run or a game that involves a balancing system using the phone's gyroscope mechanism. However, after a few days of discussing back and forth, we decided to give ourselves a bigger challenge and tackle ARKit and RealityKit. There were a lot of different games we researched that gave us a general idea of what we wanted to do. Games like Minecraft Earth and Pokemon Go were the biggest inspirations for our app. As fans of the old game Nintendogs by Nintendo, we thought it would be a great idea to make an Augmented Reality version of that game called "My Virtual Pet". We wanted to take the aspect of tossing objects and having these actions interact with our pet models that are placed within our camera view. Similar to Nintendogs, actions like petting, feeding and playing are all actions that we expected to implement in the game to create an app that closely simulates owning and caring for a pet. Beyond the core mechanics of the game, our stretch goals included more interactivity between players and a scoring system with a leaderboard. If we were able to implement the main functions of the game we could use the

extra time to allow players to have more incentive to play more by rewarding them with points and the satisfaction of caring for their pet.

To begin development, we first had to learn more about working with the ARKit and RealityKit frameworks before planning a cohesive plan for our app. The first two weeks after the groups were formed (10/7-10/24) were spent learning more about the fundamentals of ARKit and RealityKit. Some mini demos and small practice projects from Youtube or articles on related topics were crucial in our understanding of these frameworks. With a decent base understanding of the frameworks, it was time to begin the planning and blueprinting stage of our app development. However, before we blindly started creating tests and demos with Apple's models in Reality Composer, we had to divide the work amongst ourselves. One of us would work on the user interface that can be generated through SwiftUI; another would work in the Reality Composer to create triggers, behaviors, models, and animations; the remaining member would work on any art needed on the user interface and would assist in both SwiftUI and Reality Composer if needed.

For the starting screen, we planned to create a visually appealing and dynamic background. The starting screen would have a "Play" button and a "Choose Pet" button that allowed the player to choose what pet they wanted to interact with. When a player would tap on the "Choose Pet" button, a ScrollView would appear with the available pet options. Our initial title "My Virtual Pet" was too long and took up most of the space on the start screen and left minimal room for the logo and buttons, so we changed our game title to "Dobutsu Daycare." Along with the start screen, the screen that involved the AR processes would

possibly have any buttons and views generated from SwiftUI as well. One button would be a “Go Back” button, used to go back to the main menu and change the pet. Later on, we found that it would be better to not clutter the AR view with views that are larger than buttons. Thus, we opted to have most of the UI options be generated through the Reality Composer.

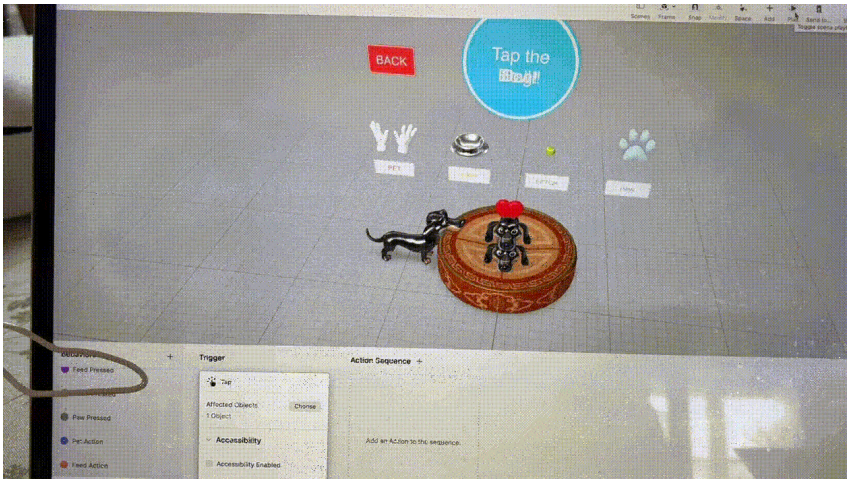


The core part of the game would be found within the Reality Composer, where all the models, animations, action triggers, and behaviors would be found. As our project heavily relies on the usage of 3D models, we had to acquire our models, and additionally in a format that would be acceptable to the Reality Composer file. After some research, it was discovered that USDZ files are the required format for AR-related tasks in XCode. This led

to the required use of Autodesk Maya to convert some animations and models into the required USDZ format such as the dog and cat's models. As we needed two separate scenes for our cat and dog, we created two separate scenes within the Reality Composer and determined which one would be loaded based on our settings in our ContentView file. After placing our models in their desired positions, we moved on to the behavior of our scene. Reality Composer offers a wide range of action detection and responses which greatly aided in the development of our application. To trigger activities, we detected touch with our menu, loaded our required game objects as a result, and the relevant animations. Each activity is unique and offers a different way to interact with our pets.

As with every development process, we ran into a few issues that we managed to resolve. However, there were several things that we hoped to implement that, unfortunately, could not be included in the final product, including the stretch goals. Although Reality Composer allowed us a straightforward method in development, we did come across a few issues along the way. Finding models to import into Reality Composer posed a significant problem. Apple did not offer any USDZ models for animals, and we found that creating our models and animations would take too much time as none of us were experienced in any form of 3D modeling and animation.. Fortunately, we found dog and cat models with animations that were available for the public and imported those into our project. However, there is no native USDZ converter in XCode that streamlines conversion with animations, which requires us to use a new skill in Autodesk Maya to consolidate our animations. The models came with their animations in one playthrough as USDZ files only allow one animation to be linked to a model. Unfortunately, this meant that we would have to manually

chop the animations in Maya. We quickly learned that it was only possible to extract the first animation set as the models would only rig on the first frame of our animation clip. Although Reality Composer made augmented reality development much simpler, it wasn't without its drawbacks.



Demo in Reality Composer

As ambitious as we were with our app, we were not able to work on our stretch goals and spent most of our time debugging model and animation issues. The semester project enabled us to apply what we learned in SwiftUI to make a visually-appealing user interface, explore the ARKit and RealityKit frameworks to gain experience with augmented reality, and collaborate with a team to produce a presentable product. After the semester, we plan to continue developing Dobutsu Daycare and implement our stretch goals. We hope to provide players with more interactive options, a larger variety of pets, and a polished user interface to enhance playability and offer a more immersive experience.