# Chapter 3 – Methods

## 3.1 Development Environment

### 

### 3.1.1 Unity

Unity has been chosen as the game engine to within which CardsAR will be developed. This was selected for its extensive support for smartphone-based AR development. Unity has multiple options for AR plugins as well as an active community and marketplace for models and plugins. This choice aligns with choices made by game development studios; both Niantic and Rovio Entertainment develop their AR games in Unity (FRANCIS, 2016) (DAVENPORT, 2022).

### 3.1.2 AR Toolset

Multiple software development kits (SDKs) are available in Unity for AR. Unity has support for both iOS’s ARKit and Android’s AR core. CardsAR was developed primarily on an iPhone, originally it was planned to be tested on Android during Build 3, but time did not allow for this. Therefore a cross-platform AR SDK solution is required. AR Foundation meets the requirements, it allows for AR apps for Android and iOS to be built from the same code. However this SDK does not have the breadth of features that Lightship ARDK has.

Lightship ARDK from Pokémon GO creators Niantic is a cross platform AR toolkit for Unity that adds to the feature sets found in ARKit, AR Core, and AR Foundation (Niantic, 2022). In addition to 3D plane detection, Lightship provides built-in multiplayer netcode and session management. The toolkit also provides an in-editor world simulator for Unity. This allows for AR features to be tested without needing to build to an external device. These features allow for more time to be spent on feature development and less time on multiplayer testing.

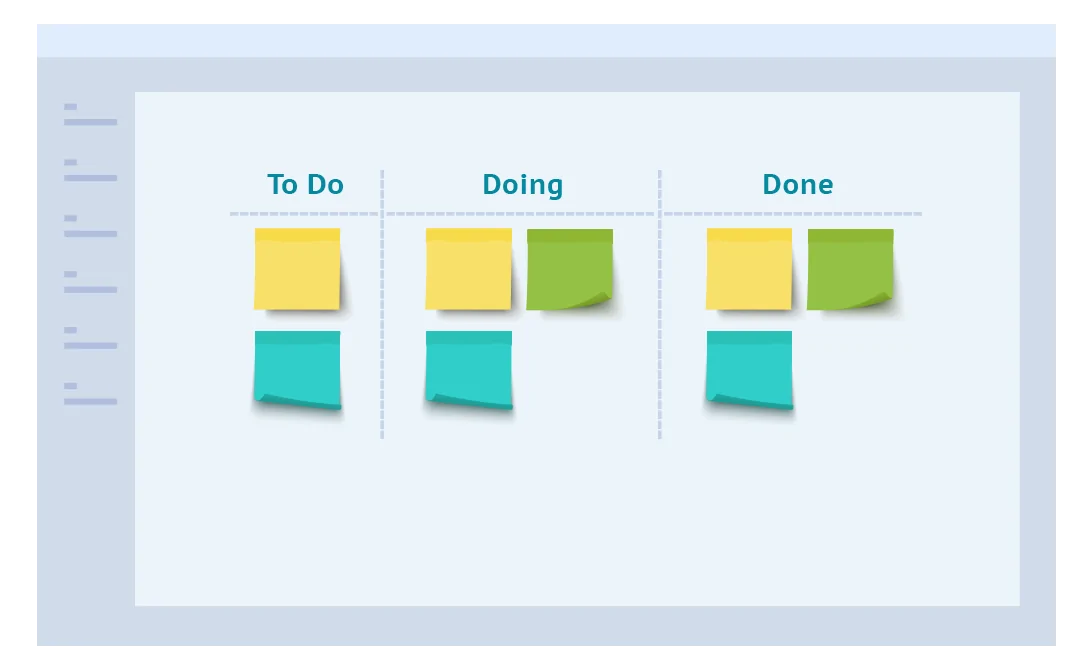
### 3.1.3 Video and Audio

For the video call component of CardsAR, Agora was chosen. The Agora Video and Voice SDKs are both available as plugins in the Unity Asset store for free. This is the only video call plugin in the Unity store that has been marked as a “Verified Solution” by Unity. Verified Solutions are Unity packages that are both compatible with the latest versions of Unity, and recognised as having long term dependability (Agora.io, 2022).

### 3.1.4 Hand Recognition

For hand gesture recognition the ManoMotion hand tracking SDK was used. This was selected as it was free to use in a university project, and it provided enough demo projects that the feasibility of hand gesture-based control could be tested without significant work (ManoMotion, 2022).

## 3.2 Software Development Practices

Throughout development a Git repo has been maintained on GitHub.com. Build 1 is focussed on research and third-party plugin experimentation which was tracked in the CardsAR repository since no code from the tests was be carried through to Builds 2 and 3. Work on Cards AR was tracked with the feature development in Build 2 and 3. At the end of every day a commit was made to the main repository and uploaded to Github.com  
  
All builds have been planned with the software project management tool Jira. The builds and their associated features have been entered into the Jira database. Two views were set up for managing the development of CardsAR: a Kanban board and a roadmap. Kanban is a framework for managing Agile software development projects within a team. Tasks are displayed on a board with columns for their status (“to do”, “in progress”, “done”), and as the project progresses, tasks are moved between columns (Digite, 2022).   
  


The roadmap view allows for planning of the project timeline. Each feature has an allotted time, and each week the allocated times are adjusted depending on the work achieved that week.

[Roadmap screenshot]

## 3.3 Design

## 

### 3.3.1 Scope

CardsAR is intended to be as close to real world card playing as possible. The game contains a single deck of playing cards. As no predefined game rules have been designed into CardsAR, the interaction model attempts to cover all the actions that a player may need. Taking inspiration from video call social gaming, CardsAR has also be designed to be easy to drop in and out of. Players should be able to join and leave a game without breaking the current ongoing game. To keep development time down, there is no multiplayer session interface, the final build auto-joins players into the gameplay session with hardcoded authentication credentials.

There is no 3D physics model. Cards are either held by a player or placed on the table. A dedicated physics model is unnecessary for the cards AR feature set. Cards do not animate between positions, they are either statically placed or dragged by players across the table.

### 3.3.2 Card Gameplay Model

The card gameplay model has the following features.

* Cards can be stacked on top of one another or individually placed on the table.
* Stacks of cards can be fanned across the table to allow users to see each card’s suit and number.
* Cards can be flipped individually or a whole stack can be flipped.
* Cards can be visible to all players or just a single player.
* Cards and stacks card can be highlighted by players
* When one player highlights a card/stack, no other player can highlight it.
* When a card/stack is highlighted one or more cards can be picked up and held by the player.
* A held card/stack can be placed onto the table again or added to an existing stack.
* Stacks of cards can be sorted and shuffled.

This feature set allows for cards to be arranged and placed in any order on the table. Card visibility allows users to have a card “hand”; a set of cards only visible to themselves. A dedicated deal card function was not added but card dealing can be achieved by picking up a set number of cards from a larger stack.

### 3.3.3 Environment

When players start CardsAR they set up their AR environment. As is common in most AR games, the virtual world needs to be aligned with the real world. Most AR games start with the user looking around their environment to allow the AR system to detect surfaces. The player is then asked to place a virtual marker on a detected surface. This marker is used to orientate the virtual objects or gameplay area. CardsAR starts off with a virtual marker placement. This marker is be used to orientate the virtual table on which the cards are placed.   
  
Diagram

Description automatically generated

Figure a: Player places virtual marker in real world

A picture containing text

Description automatically generated

Figure b: Virtual marker sets position of table

The real world camera feed is also used by the AR tools to place the table. Contrary to most AR applications, the camera feed is not shown during gameplay after the marker has been placed. The entire game takes place in a virtual room that players join. Setting the game in a purely virtual room allows for other players to be represented in the game without needing to consider each other’s physical spaces. If the camera feed was used as the gameplay backdrop, moving player avatars would break the AR illusion as their movements wouldn’t be restricted by each other’s physical space. If the physical environment were to be used to display cards, players would be required play on tables of a specific size, a virtual room is a simpler solution for inconsistencies between player environments.

### 3.3.4 Interaction Models

Built into the development process of CardsAR is an investigation into different interaction techniques. The first investigation was into hand recognition and whether players can hold their phone with one hand and interact with cards using pinch/grab gestures with their free hand. This was not feasible, as a result the primary interaction technique is a combination of on-screen buttons and using the phone as a pointing device for selecting cards.

Diagram

Description automatically generated

Figure c: Using the camera to detect the player hand (left) Using the camera to point to items in on the play area (right)