# Do A Barrel Roll, But In Real Life

Joshua Pulido
jpulido@oxy.edu
Occidental College

## 1 Problem Context

With the rapid development of technology in both the virtual reality and drone fields, what were once niche experiences have now become household brands, especially thanks to companies like Meta and DJI. That begs the question: why has there not been software that brings the two spaces together in a entertaining way? While there are certainly methods for using VR headsets to view drone footage, very little effort has been put into pushing the space further and innovating in the augmented reality field using drones. My goal is to be among the first to look into the space; more specifically, I plan on creating a video game that has the player fly a real world drone, while interacting with a virtual environment using the other buttons of a game controller.

Before talking about the game itself, it is notable to mention that such a game would bend the definition of augmented reality (AR) as commonly known today. The most popular AR games, including the popular game Pokemon GO, often focus on overlaying digital objects on top of the real world, thereby ”augmenting reality”. Virtual reality (VR) loses the ”reality”, instead focusing on creating an immersive virtual world that feels as real as possible. While augmented reality is not limited to smartphones, other devices such as Microsoft’s HoloLens have not yet reached the commercial viability of their VR counterparts. This is where the use of a drone, with it’s included camera, comes into play. By combining streaming the video feed from a drone to a VR headset, an AR game becomes possible where it otherwise would not be; being able to physically fly the drone around comes as a bonus that adds to the entertainment value and allows for interaction with the real world.

It is important to note that we are still relatively early in what will likely be a long history for both the VR/AR and drone spaces of technology, and as such the two are not necessarily designed to function together. Nevertheless, I have multiple ideas for how to make them cooperate, which I will go more in depth on in the Technical Background section. The obstacles may be difficult to overcome, but through a series of compromises, I believe that a prototype game can be developed, with the goal of creating a more integrated system in the future. There are also ethical considerations that must be made, such as whether or not inexperienced pilots should be allowed to fly drones, especially while trying to play a game at the same time. Within the scope of this project, it may not be possible to address such issues; however, in the creation of a full game that would be released to the general population, they would more than likely be dealt with through a required training program before being allowed to pilot the actual drone.

## 2 Technical Background

In an effort to capture the feel of the first person view (FPV) headsets that are traditionally used to fly drones, it makes the most sense to use a VR headset. Notably, recent advances in mobile technology have led to a split in the VR headset field, with some headsets requiring a connection to a computer and others having the ability to be completely independent. The ideal way of making this game would be the development of an application that runs natively on an ”all-in-one” headset, most likely the Oculus Quest 2. However, depending on other factors that could come into play, it might not be possible to run this type of game on Android based systems. In this case, it would make the most sense to use a PC to run the game alongside a headset such as the Valve Index. Regardless of which platform ends up being preferred, the most obvious solution for developing a 3-dimensional (3D) video game would be Unity.

Although there are many popular game engines, the two that come to mind for 3D video game development are Unity and the Unreal Engine. I have chosen Unity because of its integration with the C# language, for which there are many APIs that allow for computers to interact with drones. Unity also provides many built in tools that help simplify the game design process. Regardless of what the final gameplay ends up being, one important feature I want to include is the procedural generation of levels. There are algorithms out there that can create some very detailed levels, but since my game will have to be relatively simple, I imagine that I will end up making my own. Currently, my idea is to create an algorithm that takes the dimensions of the environment as input, up to a max value equivalent to the drone’s range. Then, using a random number generator and pre-established rules for game features, I hope to create a graph, where each node is a feature, such as an enemy or a power-up, and the weights are the distance between each item. The final part of the algorithm would just perform a clean up, again based on rules that it is fed. With that being said, there is still a lot more to determine before writing this algorithm, such as where the inputs are collected. Ideally, this would be done using sensors on the drone itself, but not every drone has access to distance-sensors. There is also the challenge of mapping the real world to the virtual environment by defining what 1 meter in the real world means in game. It is common for games to exaggerate in game physical values to make them more fun and engaging, but in the case of this game, doing so could potentially detract from the immersiveness. Therefore, there will have to be a balance between the entertainment value and the realness that the game offers. There is also the potentially difficult and non-trivial issue of getting a drone to interact with a computer.

Currently, the industry standard for piloting drones is through the use radio frequencies to transmit the required data to control a drone. This includes the instructions given to the drone by a controller, as well as the video feed given to an FPV headset by an onboard camera. Therefore, interacting with most drones in the market through a computer would require radio-capable hardware that is uncommon on PCs. With that said, drone company DJI has set a precedent of using the WiFi-standard 2.4 and 5 Ghz frequencies for their drones, allowing computers to easily connect through their WiFi chips, which potentially eliminates an entire step in the process. It is important to note, however, that latency plays a huge issue when working with any device over wireless communication, and in this case it is especially important to evaluate the latency of the video stream. One study on latency[3], performed by researchers at the National Institute of Information and Communications Technology in Yokosuka, Japan, found that the use of the 2.4 GHz radio band caused major latency issues due to the large amounts of interference. Although their research focused more on beyond-line-of-sight flight, the latency still poses an issue at closer distances, and worsens with the amount of wireless devices in the area. Theoretically, even with the interference and latency issues, the video output should be sufficiently fast enough to provide for a playable experience, but in practice this could end up not being the case. While it would be preferred to have a perfect product, compromises will clearly have to be made to make this project possible. For this reason, I have decided to use DJI and Ryze's Tello drone, which has a public SDK and a large developer community. While the video quality is not the highest resolution, at 720p, the latency should be reasonable within approximately 100 meters from the player. This might restrict what is possible with gameplay somewhat, but it should allow for a working end product, and support for other drones could be added after the fact, although that is likely outside the scope of this prototype.

## 3 Prior Work

In the department of video games that interact with real drones, very little has been accomplished. Only a single commercial game can be found using search engines that matches the description of this type of game. Named Drone Prix AR[5], this game by developer Edgybee was the first known attempt at creating an augmented reality game using a drone, in this case developed for the Android and iOS platforms. Despite lacking the immersiveness that a VR headset would bring, the game largely accomplishes a similar feel to my idea. However, it is important to note that the game no longer exists on both Google's Play Store and Apple's App Store, and there is very little evidence that the game ever existed. Perhaps this is not by coincidence - creating a game like this is something that few have done before, and given the current size of the drone market, such a product is likely not commercially viable. Nevertheless, the game and its idea, while novel, is one that many would gladly play. In fact, although there are very few video games that use AR technology with drones, there have been many other projects that run parallel to this idea. For example, one group of researchers[6] sought to create an augmented reality spectating application for drone FPV racing, and was written several years ago when the technology was not as capable as it is today. Their work mostly involved overlaying a visual UI and smart assistant that explained the race as it progressed, so it shares some similarities with my concept. However, they also used external motion detectors and other devices that will not be used in my project.

When it comes to working with Tello drones, the specific drone that I will be using in this project, there are many interesting projects that developers have made publicly available on Github. While most of these have nothing to do with video games, there are a couple that stand out as ones I can take some ideas from. One such project, created by user carter-james89[2], allows for the Tello to follow a "virtual drone" in Unity, with real time video output in the Unity game scene. Although this is not necessarily a video game, it proves that interaction with the drone in Unity using the C# library[4] created by user Kragrathea is possible. Another really interesting library is Joy[1], a simple tool developed by GiovanniGrieco that allows for the Tello drone to be flown using a traditional game controller, and written in Python. Unlike most of the other projects that use DJI's own API, Joy sends packets directly to the drone, which allows for faster responsiveness, an absolute necessity for my game to be playable. I believe that these libraries will provide me with sufficient information to make the Tello work with the game, but there is an active community of developers, including an entire website, dedicated to the Tello that

I can go to for any help if need be.

Aside from the technology side to this game, equally important is the actual gameplay. While I am still unsure exactly what genre of game I want to play, the core of this game will be an immersive flight experience, a necessity when a real drone is being flown. The biggest inspiration for my game is No Man's Sky, a space exploration game, particularly the space flight mechanics that exist. In the game, the player can fly around to collect resources, and can also optionally engage in combat with other starships. This is not the only game to explore this genre; however, it is one of the most immersive experiences, and also has a VR mode to add to the experience. Regardless of the actual gameplay and mechanics that end up making it into the final product, I really hope to capture the same feel of flight from No Man's Sky, where it is arcade-like yet still feels very real. Other games with similar gameplay that have inspired me include Star Wars: Squadrons and the Star Fox series from Nintendo. On the AR side of things, there have been relatively few games that have achieved commercial success, but games like Pokemon Go have proven that augmented reality can be viable if done right. Although Pokemon Go is of a completely different genre, its ability to captivate players through interaction with the real world, in this case through the use of GPS, is the driving force behind the use of a real drone in the game I will be making. The entire game itself can be made in full VR, but the addition of the camera view and real drone provide for a similar engaging feature that will add to the experience.

## 4 Methods

In order to create a game in the time span of approximately 4 months with a team size of 1, a strict schedule will have to be followed to ensure that game systems and design are developed in time for the release of a prototype. There are distinct pieces of this game that can be taken as their own parts, which all come together to form the final product, notably the control scheme, game design, implementation of game systems in Unity. In parallel with the development of these features, user testing should be run throughout to ensure the final product is both playable and fun.

### 4.1 Control Scheme

Arguably one of the most important design elements for this game is the control scheme, which needs to be intuitive enough such that anyone can pick up a controller and learn how to fly the drone through trial and error. As a result, the initial focus of the project will be testing different configurations for flight, including multiple different controllers, in particular a traditional game controller and the Oculus Quest controllers. Potentially, the best option might be to mimic a standard drone controller; however, these controls might not have the right feel for a game. Once several control schemes have been created, the best way to decide one will be through user testing and selecting which has the best feel, both for controller type and scheme.

### 4.2 Game Design

While the control scheme is being designed, it is also necessary to determine the exact design of the game. The first step in this will be deciding a concept, which currently is to make a game that emulates the feel of a Star Wars space fight, although the final concept might be limited slightly by what is actually possible in Unity. The next step would be to decide on the primary mechanics of the game; in this case, it is obvious that flight is one of the key mechanics, but others that are more gameplay-specific also must be decided. This could be a shield and weapon for a space-fighting game, or a speed boost for a racing game, and will depend on other decisions made about the game's key features.

Once these key elements of the game's design have been decided, a few different processes can be done in parallel. Much like in all other parts of the game's development process, user testing, or in this case surveying, should be done to see if the game design is something that a diverse group of players would actually want to experience. The responses most likely we lead to some adjustments, or potentially even an entire redesign, but this would hopefully lead to an end product that is entertaining. Level design can also begin at this stage, and will consist more of deciding what goes into a level and how to balance the game, rather than designing specific layouts. Since the game involves the real world, it might be difficult to create levels that work everywhere, so a procedurally generated system might work best, with the estimated size of the playfield being taken as an input.

### 4.3 Unity Implementation

With regards to implementation in Unity, the first (and most important) part would be to get the virtual world in Unity to display in front of the camera output from the drone. There are a couple of different ideas here: either the video stream can somehow be placed as an actual background, which would be the ideal solution, or the video stream would be placed in world, but far away and very large, to simulate it being a background. Perhaps there is another solution, but this problem is the first that must be addressed, as it is the determining factor for what is and is not possible while using a real drone.

Other game features, such as the primary mechanics, could also begin development immediately after the game design has been finalized, and would for the most part follow the standard procedure for making a game in Unity. This only gets complicated when the real drone is introduced, so movement and any other game mechanics that involve the drone would be require a more in depth development cycle. The first step in this would be getting the controller to interact with the drone through the various libraries that will be used, and making sure that the drone moves relatively smoothly. The end goal would be to tune the movement to where the game feels truly immersive, and user testing would assist in this endeavor. Alongside the mechanics, other core games systems, such as any AI for enemies, or the implementation of levels, can also take place. This would mostly involve writing algorithms and implementing them with Unity's built in systems. It is still unclear what these might be, but TODO: FINISH THIS

### 4.4 User Testing

For the various stages of user testing, the goal is to conduct tests and surveys with a group of people who all are knowledgeable in one field of the project, such as flying drones or making games. This would be useful to get specific feedback on particular parts of the game. For example, someone knowledgeable in flying drones could properly evaluate whether or not the control scheme provides proper control over the drone, whereas a game developer could provide better feedback on how intuitive the control scheme is to learn and play with. However, I recognize that this provides for a consolidated base, so I would also like to include a few people who can provide different viewpoints. User testing will take course constantly during the development process, and each test will have specific features that are targeted alongside open ended questions where users can provide any additional feedback. After each test concludes, all feedback will be taken into consideration as changes are made to the game.

## 5 Evaluation

Given that this project aims to create a video game, the main criteria for evaluation is ”fun”. However, such ”fun” comes from different perspectives, and as such they must be considered in order to properly evaluate whether or not the game was a success. In particular, once a final build is complete, a final run of user testing should be conducted to get thoughts on whether or not players think its fun, and the game must also be in a playable state.

### 5.1 Beta Testing

Unlike the user testing outlined above, this final run of testing focuses more on the actual gameplay and whether or not it is enjoyable. As a part of this, each player will be asked to first play the game, then rate the game based on how much they enjoyed it in a few factors. Perhaps the best way of doing this is to split it into a few scales based on the different parts of gameplay, including the feel of the flight, level design, and overall entertainment value. While these values are arbitrary, in general higher numbers would hint towards a more well designed game, whereas lower numbers would indicate significant room for improvement in the design and implementation of the game. One important piece of the user testing is also the playability of the game, otherwise known as how well the game performs. While a part of this is objective, there is also a subjective part to this, where every played might have a different experience in how they were able to play. Some might find it frustrating to fly a drone while playing, while others might find it too easy and desire a higher difficulty.

### 5.2 Playability

Alongside the subjective playability measures outlined above, there are also some objective measurements that can be taken. Given that this game is running on an Oculus Quest, the performance of the game on the VR headset could struggle if the game is not optimized, and so the average FPS should be measured and compared to the target of 30. In addition to the average FPS, the peaks and valleys should also be measured to identify any outliers, but they are unreliable and therefore should not necessarily be used to measure playability. Bugs and unintended gameplay should also be measured for playability; a buggy game is often not considered to be enjoyable. However, through optimization, user testing, and bug-fixing, both performance and bugs should be stable by the point of evaluation.

There are other measurements that should be taken which can not necessarily be remedied. The delay from the camera streaming from the drone to the computer, and the opposite direction, from the controller to the drone, are two of the most critical points of failure. If the perceived latency is too long, the game could end up being frustrating rather than entertaining. As a result, effort should be put in to minimize both these numbers as well as the effects they have. Since latency is in the nature of wireless technology, it might not be possible to eliminate this delay, so some should be expected. However, if it gets to a point where it is negatively impacting gameplay constantly, then that should be considered during the evaluation.

## References

[1] Grieco, Giovanni. *Joy*. 2021. URL: `https://github.com/GiovanniGrieco/Joy`.

[2] James, Carter. *UnityControllerForTello*. 2021. URL: https://github.com/carter-james89/UnityControllerForTello.

[3] Kagawa, T. et al. *A study on latency-guaranteed multi-hop wireless communication system for control of robots and drones*. Tech. rep. 2017. URL: https://ieeexplore.ieee.org/abstract/document/8301849.

[4] Kragrathea. *Tello Lib*. 2018. URL: https://github.com/Kragrathea/TelloLib.

[5] Murison, M. *Edgybees Launches Drone Prix Augmented Reality Game for DJI Pilots*. 2017. URL: https://dronelife.com/2017/05/02/edgybees-augmented-dji-drone-prix/.

[6] Sueda, K. et al. *Research and development of augmented FPV drone racing system*. Tech. rep. 2018. URL: https://dl.acm.org/doi/10.1145/3283289.3283322.