# ECS412: Project Work

## User-Interface for Virtual Reality-Based Human Swarm Interaction

**Name: Akshat Singh**

**Roll No: 20031**

**Date: April 10, 2024**


Department of Electrical Engineering and Computer Science,
Indian Institute of Science Education and Research (IISER) Bhopal.
India

# 1 Introduction

Interfaces designed for human swarm interaction in virtual environments are of great interest because of the convergence of virtual reality and human-computer interaction. This project explores this newly-emerging field in an effort to comprehend how a human can effectively collaborate and communicate with a multi-agent system of robots—here referred to as swarms—in immersive virtual reality settings. This project explores the design of such a user interface and its possible applications across various domains by concentrating on the development and assessment of user interfaces enabling such interactions.

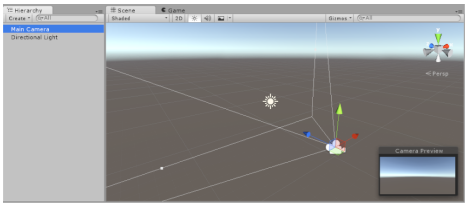
# 2 Motivation

With the advancement of VR and drone technologies, coupled with the development of various swarm algorithms, there arises a necessity for systems that facilitate human interaction with swarms without overwhelming them with vast amounts of information. While traditional methods employing 2D screens can be beneficial, there's a need to devise a system for examining the comparative efficacy of VR systems versus conventional methods. With this project, the objective of creating a system for conducting such studies is fulfilled.

# 3 Methods

## 3.1 Technology Selection

Unity was chosen as the simulation environment for the development of the virtual reality interface. Unity is a popular game engine that's well-known for its adaptability and powerful virtual reality creation tools. The project intends to design and implement a user-friendly and captivating interface for human swarm interaction by utilizing Unity's rich feature set and support for virtual reality development. It can also be used to test various different swarm algorithms as a simulation environment.



(a)

(b)

Figure 1: (a) Unity Editor and (b) VR headset with controllers

The project uses a high-performance hardware configuration that includes an Nvidia RTX 3060 graphics card and an i9-11k processor, supported by 128GB of RAM. The principal VR hardware components utilized for testing and validation are the HTC Vive Pro 2 headset and controllers. However, it is worth noting that the final product is designed to be compatible with a range of VR systems, ensuring accessibility across various platforms and configurations.

## 3.2 Scenario Setup and Objectives

The project uses a thorough 3-D model of New York City which is imported from the asset store provided by unity named Real New York City Vol 2 by Geopipe, Inc.[2]; it does not concentrate on creating the environment, as that can be done in a number of ways[7]. It consists of a specific spawn point for a leader drone that is followed by a group of follower drones. In the virtual world, groups of people are randomly positioned at one spot in the metropolis at the same time. In the main scenario, the drones are strategically deployed and navigated to find and interact with these randomly arranged human groups. This exercise facilitates exploration, coordination, and decision making in the simulated urban area while providing a hands-on example of human swarm interaction in a virtual reality setting.

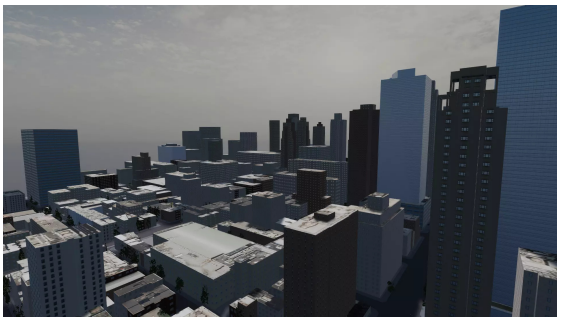


Figure 2: Environment of New York City

## 3.3 Algorithm selection

Three significant swarm algorithms were tested on the drones[3], including Couzin's model[1], the Shepherd model[5], and the Physicomimetics model[6]. For collision avoidance a customized algorithm was used. The models will be briefly described":

### 3.3.1 Shepherding model

The shepherding model has been validated experimentally on real world sheep and sheep-dog hence it was chosen as a test model. In this model, the collective motion of sheep as a swarm is influenced by five forces. Consider a scenario as shown in Figure 3(a), where an influenced agent (green circle) is affected due to the presence of a predator(blue square). Due to the presence of the predator, a force $F$ acts on the agent(located at $\bar{S}_i$) to move away from the predator(located at $\bar{P}_j$ ) which can be calculated using Equation 1), a force $C$ acts on the agent to move towards its neighbors which can be evaluated using Equation 2. There is always an inter-agent repulsion(denoted by vector $Re$ in Figure 3(a) and Equation 3) when the sheep comes within a threshold(denoted by $r_a$) of another sheep . Also, there is a tendency for the agent to continue in a previously moved direction due to inertia (Equation 4) and finally, a small error in a random direction (Equation 5). The resultant of all these forces decide the direction of the sheep's next step. Therefore, from these equations of motion, we can find that the sheep become cohesive in the presence of a shepherd (predator), otherwise they disperse and roam freely. The various parameters involved in the formulation is given in Table 1.[3]

The shepherding model proved challenging to control within the context of leader-follower dynamics, and as a result, it was not employed in the swarm.

$$F = -\rho_s \frac{(\bar{P}_j - \bar{S}_i)}{\left\|\bar{P}_j - \bar{S}_i\right\|}, \quad \forall P_i \in \mathcal{P} \tag{1}$$

$$C = \frac{c}{n} \sum_{j=1}^{n} \bar{S}_j \tag{2}$$

$$R_e = \begin{cases} \rho_a \frac{(\bar{S}_i - \bar{S}_j)}{\left\|\bar{S}_i - \bar{S}_j\right\|} & , \forall \left|\bar{S}_i - \bar{S}_j\right| \leq r_a \\ 0 & , \text{otherwise} \end{cases} \tag{3}$$

$$I = h\dot{\bar{S}}_i \tag{4}$$

$$S_e = p \cdot e \cdot v_s, \quad \forall S_i \in S_g \tag{5}$$

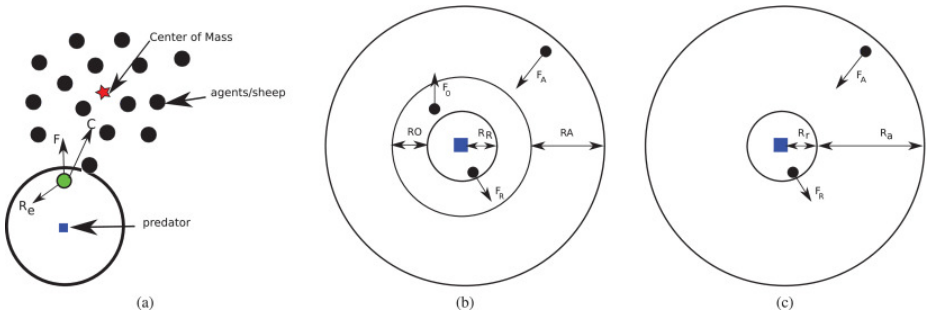



Figure 3: (a) Forces acting on an agent (green colored) under a predator influence for a shepherding model (b) Radii and Forces involved in the Couzin's model. (c) Forces involved in the Physicomimetic model

| Parameter | Description | Parameter | Description |
|---|---|---|---|
| $h$ | Relative strength of inertial effect | $R_e$ | Inter-agent repulsive vector |
| $n$ | Number of nearest neighbours | $I$ | Inertial vector |
| $C_M$ | Centre of mass of flock | $C$ | Clustering vector |
| $S_l$ | Furthest sheep position | $F$ | Predatory vector |
| $v_s$ | Speed of sheep | $p$ | Probability of random movement |
| $e$ | Strength of angular noise | $c$ | Strength of clustering effect |
| $\rho_s$ | Strength of predatory effect | $\rho_a$ | Relative strength of inter-agent repulsion |

Table 1: Description of Shepherd mopdel

### 3.3.2 Couzin's model

Couzin's model captures fish collective behavior, featuring three zones: repulsion ($R_R$), orientation ($R_0$), and attraction ($R_A$). In the repulsion zone, agents repel each other with force $F_R$; in the orientation zone, they align with neighbors' average direction with force $F_O$; and in the attraction zone, they are drawn together with force $F_A$. Agents under repulsion don't orient or attract. The model computes the average of attraction and orientation vectors for agents not in the repulsion zone as shown in Fig 3(b).[3]

$$F_R = -\sum_i \frac{r_{ij}}{|r_{ij}|}, \quad \forall i,j \in R_R \tag{6}$$

$$F_O = \sum_i \frac{v_i}{|v_i|}, \quad \forall i \in R_O \tag{7}$$

$$F_A = \sum_i \frac{r_{ij}}{|r_{ij}|}, \quad \forall i,j \in R_A \tag{8}$$

| Parameter | Description |
|---|---|
| $F_R$ | Repulsion force in the repulsion zone of agent |
| $F_O$ | Orientation force in the orientation zone of agent |
| $F_A$ | Attraction force in the attraction zone of agent |
| $R_R$ | Radius of repulsion zone of the agent |
| $R_O$ | Radius of orientation zone of the agent |
| $R_A$ | Radius of attraction zone of the agent |
| $r_{ij}$ | distance between two agents i and j |
| $v_i$ | normalized direction of motion of the agent i |

Table 2: Description of Couzin's model

The Couzin's model offered precise control over the follower drones but lacked energy efficiency since the followers continued moving even when the leader drone stopped. As a result, this model was not selected.

### 3.3.3 Physicomimetic model

Physicomimetics is a physics based model and hence this model is stationary when there is no external influence, this provides excellent energy efficiency over the Couzin's model. In this model, each agent experiences an attraction or repulsion force induced by other agents within a finite neighborhood. The model is defined by two zones – repulsion zone ($R_r$) and zone of attraction ($R_a$) as shown in Figure 3(c). All the agents within $R_r$ repel each other, while the other agents that are within $R_a$ tend to move towards each other. The force F, is defined by the Newton's Law of Gravitation, which is dependent on the mass of the particles and the inverse of the square of the distance between the particles as shown in Equation 9.

For an agent $i$, the forces are given by Equation 9

$$F_{ij} = c\frac{Gm_i m_j}{r_{ij}^2}, \quad \forall i, j \in \mathcal{P} \tag{9}$$

$$F_P = \sum_{j=1}^{n} F_{ij}, \quad \forall i \in \mathcal{P} \tag{10}$$

$$c = \begin{cases} 1 & , \forall i \in R_A \\ -1 & , \forall i \in R_R \\ 0 & , \text{otherwise} \end{cases} \tag{11}$$

| Parameter | Description |
|---|---|
| $F_{ij}$ | Force on an agent $i$ by another agent $j$ |
| $F_P$ | Net physicomimetic force on an agent |
| $G$ | Strength of physicomimetic force |
| $m_i$ | mass of agent $i$ |
| $r_{ij}$ | distance between agents $i$ and $j$ |
| $\mathcal{P}$ | Set of all agents |

Table 3: Description of Physicomimetic model

Among these, the Physicomimetics model was chosen due to its superior energy efficiency when applied to real drones. Additionally, it required less attention on the follower drones and focused primarily on the leader drone.

### 3.3.4 Collision Avoidance

To mitigate the risk of collisions in the urban environment filled with buildings, a collision avoidance algorithm is integrated into the drone's operations. This algorithm leverages Unity's collider system to identify the closest point on nearby objects in a given radius, a process that can be translated to real-world drones using SONAR technology. Subsequently, a force is exerted on the drone, directed opposite to the identified point, effectively steering it away from potential collisions.

$$F_C = -\rho_c \frac{1}{\|\vec{r_d} - \vec{r_p}\|^3}(\vec{r_d} - \vec{r_p}) \tag{12}$$

| Parameter | Description |
|---|---|
| $F_C$ | Force on agent due to collision avoidance |
| $\rho_c$ | Strength of collision avoidance force |
| $\vec{r_d}$ | Position Vector of agent |
| $\vec{r_p}$ | Position vector of identified point |

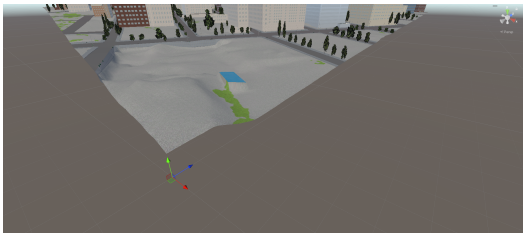
Table 4: Description of Collision Avoidance

# 4 Assets

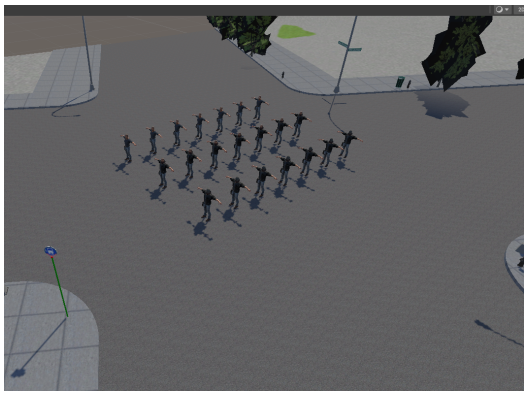
Following assets are implemented in the project

## 4.1 Environment

The environment encompasses a region of roughly 800m x 800m area within New York City, with one corner serving as the origin point and a defined global x, y, and z axes. The y-axis represents vertical direction, while the x and z axes denote horizontal directions. Within this space, there are 12 designated spawn locations for either drones or groups of humans. The task finishes when the leader drone approaches this group of humans.



(a) Origin of the environment with the three axes z(blue), x(red) and y(green)

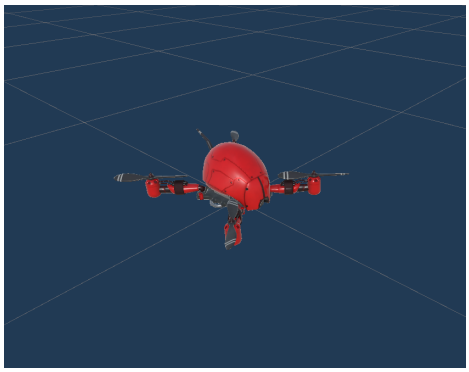


(b) Humans Spawned in the environment

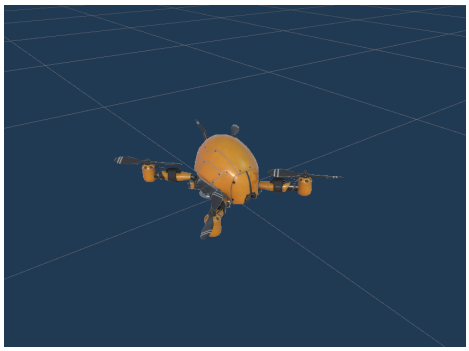
Figure 4

## 4.2 Drone

The 3D drone model utilized in this project is sourced from Indie-Pixel's tutorial, as referenced in [4]. However, the drone's controls are not adopted from their tutorial due to distinct purposes in this project. Additionally, the identical drone model serves both as the leader and follower drones, with a modification in the leader drone's body color to red for easy differentiation from the follower drones. The leader drone is spawned at a specified height among the 12 locations and the follower drones are spawned randomly within a circular area surrounding the leader drone.



(a) Leader Drone



(b) Follower Drone

Figure 5: 3-D models of leader and follower drones

## 4.3 Drone Controls

For basic movement and algorithm study, the drones' vertical movement is limited. They are initially placed at a set altitude and maintain this height throughout their operation. This restriction on vertical motion can be described as follows:

$$F_{vertical} = mg \tag{13}$$

| Parameter | Description |
|---|---|
| $F_{vertical}$ | Net force acting in vertical up direction |
| $m$ | Mass of the drone |
| $g$ | Gravitational acceleration |

Table 5: Description of Vertical force on leader drone

$F_{vertical}$ : Net force acting in vertical up direction

$m$ : Mass of the drone

$g$ : Gravitational acceleration

The horizontal movement of the leader and follower drones is governed by distinct control methods. Each drone has predefined maximum speeds for linear movement and angular motion.

#### 4.3.1 Leader Drone

The leader drone is spawned at one of the 12 spawn points randomly, while making sure it doesn't spawn at the goal position. It has slightly less max speed than the followers which is controlled by a multiplier on leader drone. For the forward and backward motion, user input is received via the controllers, which ranges from -1 to 1. This input value is then used to adjust the drone's velocity according to the following scheme:

$$V_{FB} = PS_{max}M, \quad P \in [-1, 1] \tag{14}$$

| Parameter | Description |
|---|---|
| $V_{FB}$ | Velocity in forward direction, negative means backward direction |
| $P$ | Input "Pitch" received from the user |
| $S_{max}$ | Maximum value of speed the drone can have |
| $M$ | Multiplier for leader drone |

Table 6: Description of Speed of leader drone in forward direction

The drone can also rotate along its vertical axis, this yaw value is calculated as follows:

$$\Delta\theta = Y\omega_{max}\Delta t, \quad Y \in [-1, 1] \tag{15}$$

| Parameter | Description |
|---|---|
| $\theta$ | Angle of drone with the z-axis |
| $Y$ | Input "Yaw" received from the user |
| $\omega_{max}$ | Maximum value of angular speed the drone can have |
| $\Delta t$ | Time slice |

Table 7: Description of Rotation of leader drone

The user can use a touchpad on the controller to adjust the "Pitch" and "Yaw" of the Leader Drone.

#### 4.3.2 Follower Drone

The follower drones are spawned randomly within a circular area surrounding the leader drone, and the user does not have direct control over them. The physicomimetics[6] model collision avoidance algorithm are then implemented on these follower drones. The magnitudes of maximum force and maximum velocity these drones are fixed.

$$F = F_{vertical} + min(F_P + F_C, F_{max}) \tag{16}$$

$$\Delta V = \frac{F}{m\Delta t} \tag{17}$$

$$V = min(V_{calculated}, V_{max}) \tag{18}$$

| Parameter | Description | Parameter | Description |
|---|---|---|---|
| $F$ | Net force on the drone | $F_{vertical}$ | Net Vertical force on the drone |
| $F_P$ | Physicomimetics force on the drone | $F_C$ | Collision avoidance force on the drone |
| $F_{max}$ | Maximum force of the drone | $\Delta V$ | Change in velocity in a given time slice $\Delta t$ |
| $m$ | mass of the drone | $\Delta t$ | Time slice |
| $V$ | Velocity of the drone | $V_{calculated}$ | Velocity of drone according to equation 17 |
| $V_{max}$ | Maximum velocity of the drone | | |

Table 8: Description of controls of Follower drone

## 4.4 Pointer

A pointer is linked to the user's right-hand controller, capable of extending indefinitely and directing towards the first object it encounters, as depicted in the figure 6. This pointer serves for movement, elaborated upon in section 4.6, and for landmark selection, detailed in section 4.8.

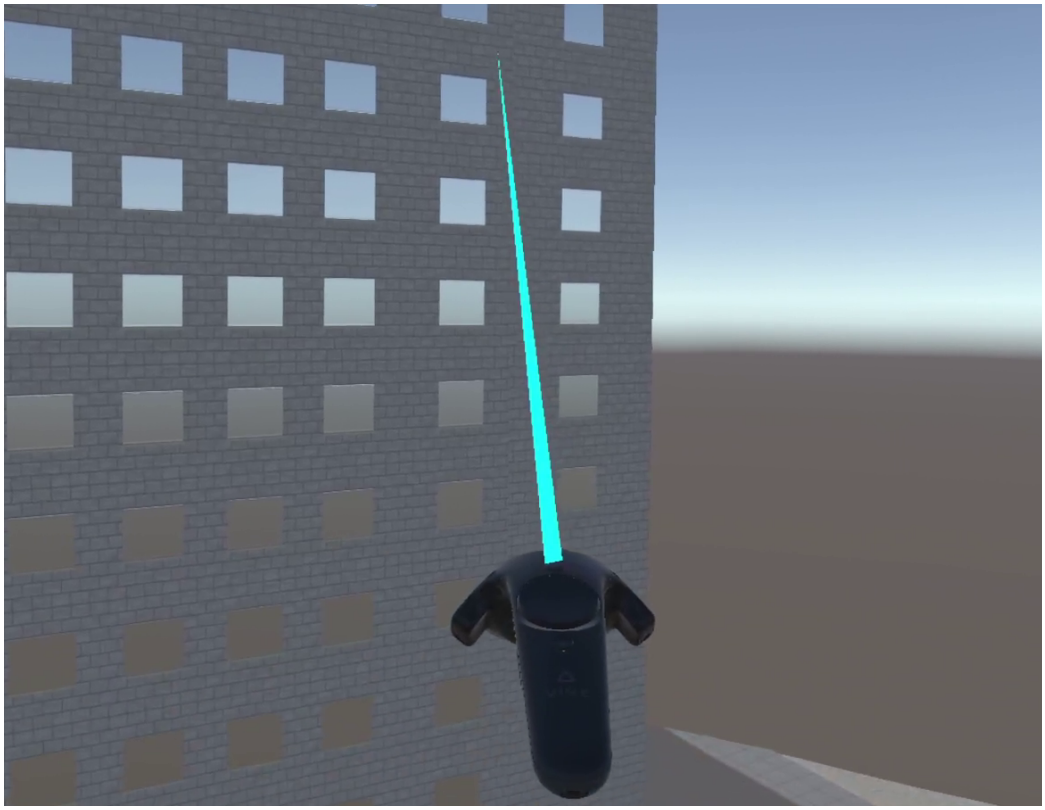


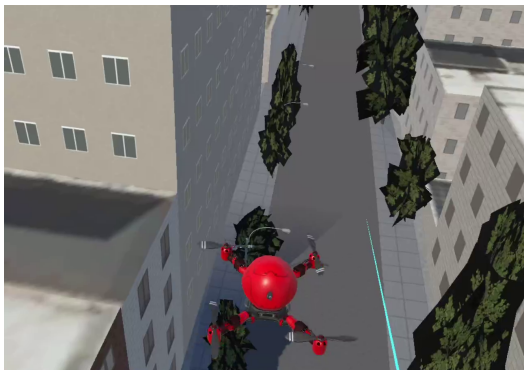
Figure 6: Pointer

## 4.5 Perspective Selection

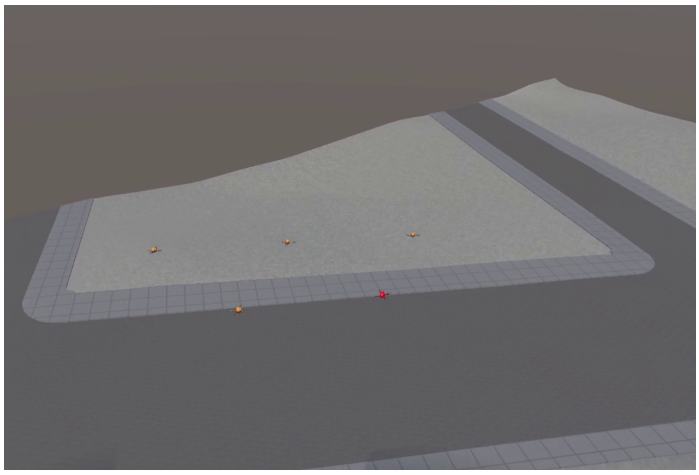
In the project, users are presented with three perspectives to control the drone: First-Person Perspective (FPP), Third-Person Perspective (TPP), and Fixed Perspective for the user to have different views of the drone in order to control it.

- **FPP:** The First-Person Perspective (FPP) places the user as if they were a camera attached directly below the leader drone, as illustrated in Fig. 7. In this view, the user can maneuver the drone and look around in all directions but is unable to move away from the drone.
- **TPP:** In the Third-Person Perspective (TPP), the user is situated just behind the leader drone, maintaining a connection to it. This setup enables the user to observe the leader drone, control its movements, and look around, but it doesn't allow the user to move independently of the drone.


(a) First-Person Perspective (FPP)



(b) Third-Person Perspective (TPP)



(c) Fixed Perspective

Figure 7: Perspectives

- **Fixed Perspective:** In this perspective, the user remains stationary at their last position, and the drones are not tethered to the user. The user has the freedom to move around within this perspective, with further details discussed in section 4.6.

The user has a dedicated button on the controller for switching between the different perspectives.

## 4.6 User Movement (Zip)

In the fixed perspective, as described in section 4.5, the user has the ability to move within the environment. To facilitate this, the pointer can be directed to any location in the environment, and upon pressing the "Trigger" button on the controller, the user is swiftly transported (or "zipped") to the targeted location. This "zipping" action signifies that the user immediately begins moving towards the targeted location from their current position, following a straight path until they reach the destination.

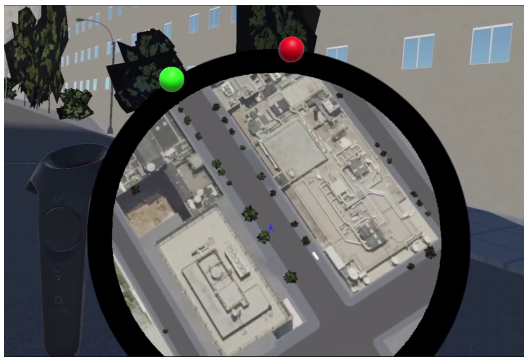
The process involves using linear interpolation of the user's position and the pointed position at each frame until the user reaches a threshold distance from the destination.

$$\vec{r_p} = \vec{r_s} + (\vec{r_d} - \vec{r_s})t \tag{19}$$

where:

$r_p$ : Current position of the user,
$r_s$ : Starting position of the user,
$r_d$ : Destination postion of the user,
$t$ : Multiplier that determines how far the user moves

(a) User(blue arrow), Leader drone(Red arrow) and follower drones(yellow circle)



(b) Green dot determining Landmark's direction, Red dot determining Leader drone's direction

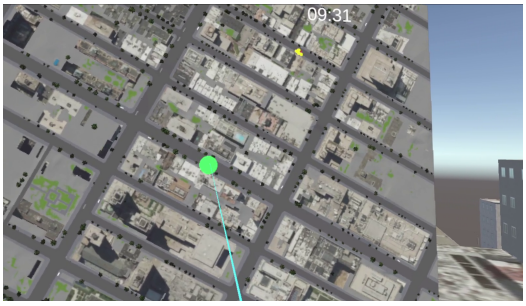
(c) Red trail left behind by Leader drone

Figure 8: Minimap

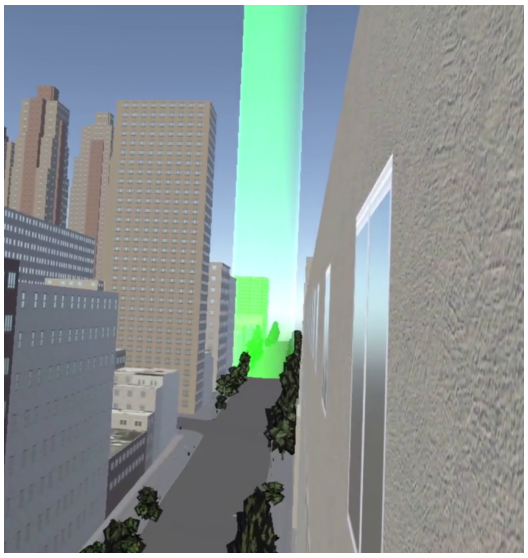
## 4.7 Minimap

In the project, users are provided with a 3D view of their environment, which can sometimes lead to difficulties in self-localization due to the information overload. To address this issue, a minimap is integrated into the user's left-hand controller as shown in Figure 8, offering the following essential details and aiding in navigation:

- **Top down view:** The minimap provides a top-down view of the environment around the user at all times.
- **User Location:** The user stays at the center of the minimap at all times as a blue arrow, and the minimap rotates accordingly to ensure that the user's direction is consistently oriented forward on the minimap.
- **Drone Location:** The leader and follower drones are represented on the minimap by a red arrow (for the leader) and a yellow circle (for the follower) when they are near the user. If they are distant, a red dot is shown to indicate the direction of the leader drone relative to the user.
- **Landmark Location:** The minimap will display the landmark if selected as a green circle on the minimap when it is within the user's proximity. In cases where the landmark is not nearby, a green dot will appear on the minimap to indicate the direction of the landmark relative to the user's position as shown in Fig 8b. Further details about the landmark are given in section 4.8.




(a) Full map with selected landmark



(b) Landmark visible in the environment

Figure 9: Full-map and Landmark

- **Drone Trail:** The minimap displays a red trail indicating the path traveled by the drone, providing a visual representation of its previous movements, as depicted in Fig. 8.

### 4.8 Full map with Landmark selection

The user has access to a top-down view of the entire environment in the form of a map, accessible by pressing a designated button, as illustrated in Fig. 9. This full-map incorporates all the features discussed in section 4.7 regarding the minimap, including the ability to select landmarks. Using the pointer, the user can designate any location on the map and use the "Trigger" button to mark it as a 'Landmark' within the environment. The chosen location is represented by a green circle on both the map and minimap. Additionally, a tall cylindrical structure (as shown in Fig. 9b) is generated at the landmark, disappearing upon contact with any drone.

## 5 Conclusion and Future work

This project successfully developed a user interface for human interaction with swarms of drones in a virtual reality environment. The interface was implemented using the Unity game engine and tested with an HTC Vive Pro 2 VR headset and controllers.

The project utilized a detailed 3D model of New York City as the virtual environment. The main scenario involved strategically deploying and navigating drones to interact with randomly placed groups of humans within the city. Three major swarm algorithms were evaluated - Couzin's model, the Shepherding model, and the Physicomimetics model. Ultimately, the Physicomimetics model was selected due to its energy efficiency and ease of controlling the follower drones relative to the leader drone controlled by the user.

Key features of the user interface include multiple viewing perspectives (first-person, third-person, and fixed), a pointer tool for navigation and selecting landmarks, a minimap for spatial awareness, and the ability to quickly "zip" to different locations. The interface provides an immersive means for a human to monitor and direct a swarm of drones in a virtual urban setting.

Overall, the project demonstrates the feasibility of creating intuitive user interfaces that enable effective human-swarm interaction within virtual reality environments. This has potential applications across various domains such as search and rescue, surveillance, environmental monitoring, and more. Further research could explore more advanced swarm algorithms, additional interface elements, and user studies to evaluate and refine the system.

# References


[1] IAIN D. COUZIN et al. "Collective Memory and Spatial Sorting in Animal Groups". In: *Journal of Theoretical Biology* 218.1 (2002), pp. 1–11. ISSN: 0022-5193. DOI: https://doi.org/10.1006/jtbi.2002.3065. URL: https://www.sciencedirect.com/science/article/pii/S0022519302930651.

[2] Inc. Geopipe. *Real New York City Vol. 2*. Unity Asset Store. 2023. URL: https://assetstore.unity.com/packages/3d/environments/urban/real-new-york-city-vol-2-222827.

[3] Raghavv Goel et al. "Leader And Predator Based Swarm Steering For Multiple Tasks". In: *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*. 2019, pp. 3791–3798. DOI: 10.1109/SMC.2019.8913942.

[4] Indie-Pixel. *Unity Drone Controller*. 2020. URL: https://www.youtube.com/playlist?list=PL5V9qxkY_RnK7R1pjh0cByUCB0Adiwown.

[5] Jyh-Ming Lien et al. "Shepherding behaviors". In: *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*. Vol. 4. 2004, 4159–4164 Vol.4. DOI: 10.1109/ROBOT.2004.1308924.

[6] William M Spears and Diana F Spears. *Physicomimetics: Physics-based swarm intelligence*. Springer Science & Business Media, 2012.

[7] Chuanqi Zheng, Annalisa Jarecki, and Kiju Lee. "Integrated system architecture with mixed-reality user interface for virtual-physical hybrid swarm simulations". In: *Scientific Reports* 13.1 (2023), p. 14761.