# Applying Grey Wolf’s Behaviours for mutil-UAVs system in surveillance and combat mission

# Nam The Pham

# z5196268

# Problem definition

The motivation for this project is to design a smart drone swarm’s tactics applying grey wolf’s behaviours for surveillance and combat mission. The real-world challenge is that drones have to conduct surveillance mission on large areas without human control. Through this simulation, we need to design a tactic that swarm of drones can rely on, therefore, they can conduct the mission effectively without human control. They are capable of making decisions without human intervention including where to scan, where to move, when to attack and conduct an ambush on the target.

# Study objectives and Hypothesis

This project is to study the effectiveness of GWB application in swarm of drones for surveillance and combat mission by simulating a battlefield of UAVs and tanks then observe output results including who wins the battle and amount of time the battle takes.

My hypothesis is that if UAVs following Grey Wolf’s behaviours (GWB) in which they are divided in groups and each group led by an alpha UAV, they will be more effective in detecting and destroying enemy’s tanks. Moreover, I will do experiments on factors that can affect the fighting effectiveness of UAVs swarm including the influence range of UAVs and the number of UAVs participating in the mission.

# System definition

Since I compare the fighting effectiveness of UAVs when it is following GWB and when it is not, I have 2 systems. The behaviours of UAVs in 2 systems are different but the tanks’ behaviours remain the same. For convenience, System 2 is where multi-UAVs system is following GWB and System 1 is where multi-UAVs system is not.

Entities:

* Troops: UAVs and tanks
* Troop size: each side has 9 units. both sides place all their UAVs or tanks in the battle at the beginning
* Fighting effectiveness:
* Tank can destroy 1 UAV at a time as soon as the UAV is in its shooting range
* 3 UAVs can group together and conduct an ambush to destroy a tank

Behaviours:

* UAVs in System 1:
* Move randomly
* Once a tank in its influence range, it will pursue the tank
* It will stop and conduct ambush on the tank if at least 2 other UAVs are pursuing the same target with it.
* Once the tank is destroyed, it continue move randomly.
* UAVs in System 2:
* Normal UAVs move in groups of 3 with one leader – alpha UAV. Alpha UAVs move randomly, finding tanks
* In their groups of 3, once every UAV keep the found tank in its influence range, they stop and conduct ambush on the tank and destroy it.
* Once the target is destroyed, they immediately moving and looking for another target
* Tanks:
* Move as a boid flocking
* Attack as soon as any UAV get in their shooting range. They can only attack 1 UAV at a time

Effects: opponents destroyed and remove from the battlefield

Relationship between entities: warring sides

Environment: a non-obstacle terrain which has a rectangle shape and no elevation.

The resolution level is force level.

# Primary and secondary representations

To represent the system defined above, I uses finite-state machine (FSM) because the system have a set of finite states and triggering events. FSM is the most suitable in this case because it can go through a number of states and state changes are triggered by events or conditions called “transitions”. The system also has a set of predefined actions that can be performed in sequence. For example, an UAV that is move randomly will start pursuing a tank if the tank is in its influence range. In above case, the UAV has 2 states (moving randomly and pursuing a tank) and state changes between 2 states are triggered by the event of a tank in its influence range.

Another method can be used to represent the system is decision tree in which the choice between number of alternatives are represented through a tree structure. There are nodes and branches in the decision tree, each internal node is a test on a single attribute and each branch represent the result of the attribute test. For example, a test of attribute occurs when a tank is moving, it checks if the distance between a tank and any UAVs is closer than the tank’s attacking range. If the test produce the answer ‘yes’, the number of UAVs is reduced by 1 since the tank shoot the UAV. Another example is when a test occurs when an UAV is pursuing a tank, the test checks if the number of UAVs are pursuing the same tank is more than 2. If the answer is ‘yes’, then the UAVs will conduct an ambush on the tank and destroy it. However, the use of decision tree in this case has some drawbacks. Firstly, the decision trees will have a lot of branches which are the same because if the test produce answer ‘no’ then the branch should be the same as the branch before the test of attribute. For example, a test occurs when an UAV is wandering, it checks if any tanks is in the UAV’s influence range and produce the answer ‘no’, then the UAV will continue wandering. Secondly, the system has a number of events and conditions which can make a tree become too big. In this case, it is difficult for users and modeller to understand such representation.

Considering the resolution level is force level and the main focus of the simulation is to produce results such as who wins the battle and amount of time the battle takes, I will focus on some important behaviours and ignore unnecessary details in the abstraction process. This is to make sure that the representation suits the purpose and the model is still valid under assumptions. Specifically, I focus on the behaviours of UAVs and tanks when moving and how they interact with each other. Meanwhile, details related to how UAVs conduct an ambush on tanks and how tanks can shoot down UAVs are ignored. To move from resolution to abstraction, I made the following assumptions:

* UAVs will successfully destroy a tank as long as they conduct an ambush in group of at least 3 UAVs
* The tank will successfully destroy an UAV as long as it is in the tank’s shooting range.
* The maximum speed of UAVs is twice as large as the maximum speed of tanks.

# Modelling

It is noticed that the behaviours of UAVs in system 1 and behaviours of alpha UAVs in system 2 are the same, therefore I used a common representation for them.

Finite state machine:

A FSM of an alpha UAV can be defined as a tuple AlphaUAV = (Σ, Q, q0, 𝛿, F)

The input to the FSM is the information sensed by the alpha UAV including:

* IR: at least a tank is in its influence range
* RS: at least 2 other UAVs are targeting the same target with it

Therefore, Σ = {{0, 0}, {1, 0}, {1, 1}}. It is noticed that {0, 1} is not included because there must be a tank in its influence range so that it can have a target and sense 2 other UAVs that target the same tank.

There are 3 states including:

* wandering (W), the alpha UAV is moving randomly to search for tanks
* pursuing (P), the alpha UAV lock on a target and it pursue its target to keep the target in its influence range
* shooting (S), the alpha UAV stop and conduct ambush to destroy the tank with at least other 2 UAVs

Therefore, Q = {W, P, S}. At the beginning of the battle, the alpha UAV move randomly to search for the tank, hence q0 = W.

Consequently, an action is associated with each state:

* state W, action: MR. the alpha UAV move randomly.
* state P, action: FP. The alpha UAV follow its target and keep the target in its influence range.
* state S, action: SS. The alpha UAV stop and conduct the ambush to destroy its target with at least other 2 UAVs.

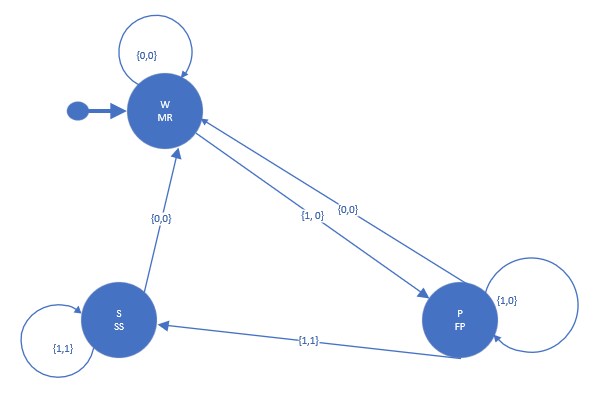


Figure 1. FSM for modelling alpha UAVs’ behaviours to find and destroy tanks

A True Table of the FSM for modelling alpha UAVs’ behaviours to find and destroy tanks

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| State | Actions | IR | RS | Next state |
| W | MR | 1 | 0 | P |
| W | MR | 0 | 0 | W |
| P | FP | 1 | 1 | S |
| P | FP | 0 | 0 | W |
| P | FP | 1 | 0 | P |
| S | SS | 0 | 0 | W |
| S | SS | 1 | 1 | S |

A FSM of an UAV can be defined as a tuple UAV = (Σ, Q, q0, 𝛿, F)

The input to the FSM is the information sensed by the UAV including:

* PA: the position of alpha UAVs and/or other UAV(s) in its group
* PT: the position of target locked by its alpha UAV (if the alpha UAV has locked on a target)
* RS: at least 2 other UAVs are targeting the same target with it

Therefore, Σ = {{0, 0, 0}, {1, 0, 0}, {1, 1, 0}, {1, 1, 1}}. It is noticed that if the UAV cannot sense the position of its alpha UAV (which means that it does not have one), the UAV cannot sense a position of a target or decide to attack any target. In different words, if not “PA” then not “PT and not “RS”. Moreover, if a UAV does not have a target, it does not matter if any UAVs target the same tank with it because it does not have one. In different words, if not “PT” then not “RS”.

There are 4 states including:

* wandering (W), the UAV is wandering to find an alpha UAV that it can follow
* following (FL), the UAV is following its alpha UAV
* pursuing (P), the UAV is pursuing the target that its alpha UAV has chosen.
* Shooting (S), the UAV stop and conduct ambush to destroy the tank with its group

Therefore, Q = {W, FL, P, S}. At the beginning of the battle, each UAV will follow its designated alpha UAV, hence q0 = FL.

Consequently, an action is associated with each state:

* State W, action: MR. The UAV moves randomly and search for an alpha UAV to follow, it must avoid any tanks in its influence range since it is alone by itself and cannot conduct ambush.
* State FL, action: MB. The UAV follow its alpha UAV and moves as a group with other UAVs that have the same alpha UAV. It interact with other UAVs in the group depend on its alignmentRange, collisionRange and cohesionRange.
* Cohesion: each tank moves towards the average position of its neighbouring tanks
* Alignment: each tank turns to the average heading of its neighbouring tanks
* Collision: each tank steers to avoid collisions with its neighbouring tanks
* State P, action: P. The UAV pursue the tank that its alpha UAV has targeted.
* State S, action: SS. The UAV stop and conduct the ambush to destroy its target with at least other 2 UAVs.

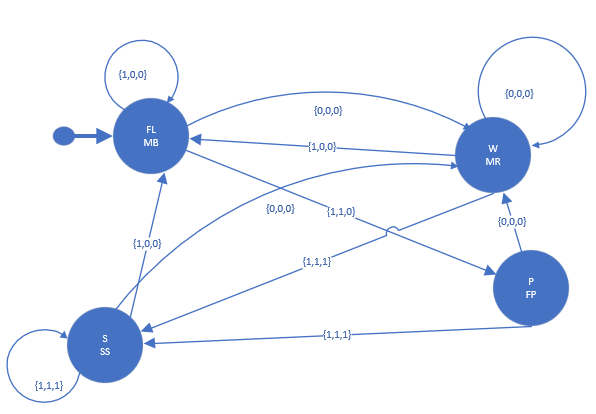


Figure 2. FSM for modelling UAVs’ behaviours to find and destroy tanks

A True Table of the FSM for modelling alpha UAVs’ behaviours to find and destroy tanks

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| State | Actions | PA | PT | RS | Next state |
| FL | MB | 1 | 0 | 0 | FL |
| FL | MB | 0 | 0 | 0 | W |
| FL | MB | 1 | 1 | 0 | P |
| W | MR | 0 | 0 | 0 | W |
| W | MR | 1 | 0 | 0 | FL |
| W | MR | 1 | 1 | 1 | S |
| P | FP | 0 | 0 | 0 | W |
| P | FP | 1 | 1 | 1 | S |
| S | SS | 1 | 1 | 1 | S |
| S | SS | 1 | 0 | 0 | FL |
| S | W | 0 | 0 | 0 | W |

A FSM of a tank can be defined as a tuple Tank = (Σ, Q, q0, 𝛿, F)

The input to the FSM is the information sensed by the tank including:

* SUAV: a UAV is in its attacking range

Therefore, Σ = {{0},{1}}.

There are 2 states including:

* Moving as boid flocking (MB): the tanks is moving as a boid flocking by interacting with other tanks depend on its alignmentRange, collisionRange and cohesionRange.
* Cohesion: each tank moves towards the average position of its neighbouring tanks
* Alignment: each tank turns to the average heading of its neighbouring tanks
* Collision: each tank steers to avoid collisions with its neighbouring tanks
* Shooting (S): The tank will immediately shoot any UAVs in its attacking range. The tank can only shoot one UAV at a time.

Therefore, Q = {MB, S}. At the beginning of the battle, all tanks are moving together as a boid flocking, hence q0 = MB.

Consequently, an action is associated with each state:

* State MB, action: M.
* State S, action: SM. The tank can shoot and then continue moving after target is destroyed.

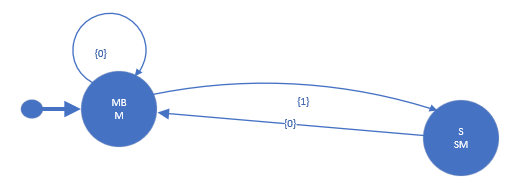


Figure 3. FSM for modelling tanks’ behaviours

A True Table of the FSM for modelling tanks’ behaviours

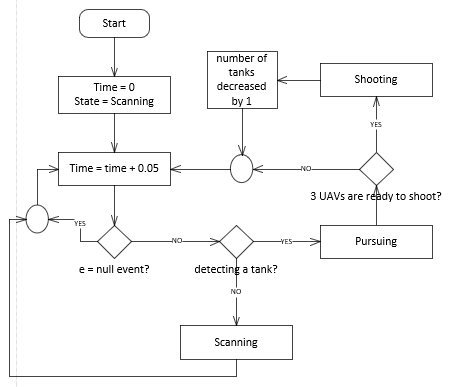
|  |  |  |  |
| --- | --- | --- | --- |
| State | Actions | SUAV | Next state |
| MB | M | 1 | S |
| MB | M | 0 | MB |
| S | SM | 0 | MB |

# Simulator

The rules of the simulation are as followings:

* The UAVs which are not alpha UAV must move as a group which is led by their corresponding alpha UAVs. The alpha UAVs and each UAV’s alpha UAV- its leader will be chosen before the simulation starts. This simulated the behaviour of the wolf pack where other wolves are commanded by the alpha wolf and the hunt is led by the alpha wolf.
* It is always need to be at least 3 UAVs target at the same tank before they can change state to “shooting” and the tank can change state to “being attacked”. This simulated the hunting behaviour of grey wolf when they need to outnumber the prey. The reason why they need to outnumber the prey is that wolf’s target are normally bigger than wolves and might be stronger than them. Therefore, they need to attack in group to be able to kill the prey. This trait is similar to UAVs when UAVs’ targets are tanks that are armoured strongly and difficult to be destroyed. Moreover, tanks are equipped with machine gun and are capable of destroying UAVs which can make the ambush of UAVs unsuccessfull. Therefore, there must be a sufficient number of UAVs conducting ambush so that the assumption (in which UAVs always successfully destroy tanks) is true.

Time advance mechanism for this simulation is fixed-increment time advance. I can use a flowchat to represent discrete time simulation approach.



Discrete time simulation approach represented by flowchart

Base on the concept of Agent-based simulation: ABS = {A,I,E,T}:

1. A: the set of agents include 12 UAVs and 20 tanks
2. I: interaction space between agents is in 2D Euclidean space
3. E: agents live in a rectangle size 500x500 in 2D Euclidean space
4. T: discrete time-advance mechanism. After every 0.05 seconds, the positions and states of agents are updated.

I simulated 2 types of agents which are UAV and tank.

An UAV agent can be formalised as a = {B,S,D,N,M}:

1. B: set of behaviours of UAVs

* Wandering: alpha UAVs will lead a team of 3 UAVs and it move randomly to scan for targets. Other UAVs in the group move as a boid following alpha UAV.

Alpha’s UAV’s movement is done by the following equations:





It is noticed that normal UAVs also have ‘wandering’ behaviour which is the same as alpha UAVs’. In addition to that, it has ‘following’ state when it has an alpha UAV:

* Following: normal UAVs following alpha UAVs and hence their movement are done by the following equations:





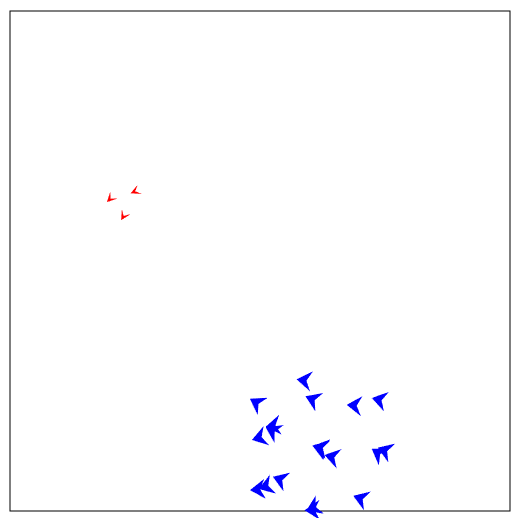
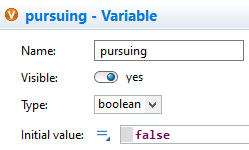


Fig 4. UAVs move in group led by alpha UAV

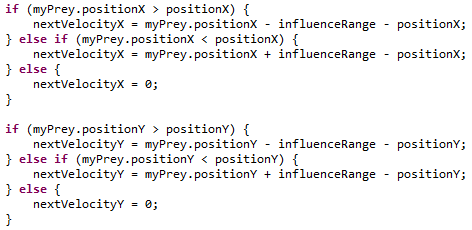
* Pursuing: alpha UAVs lock on a target and make a chase

This state is determined by the following variable

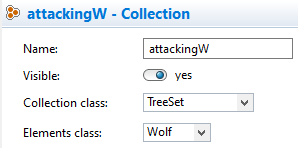


‘pursuing’ is a boolean and its value becomes ‘true’ only if the alpha UAV is pursuing a tank.

The equations implement movement of alpha UAV is different since now the UAV’s movement steer towards its target. I have to mention that the UAV will pursue the target but still keep it in a safe distance because the tank can destroy the UAV if it comes too close and enter the tank’s attacking range.



Moreover, each tank have a collection that stores all UAVs that are currently targeting it.



Therefore, if it reachs this ‘pursuing’state the alpha UAV will be added to ‘attackingW’ collection of its target.

* Pursuing: normal UAVs will pursue its alpha UAV’s target as soon as the alpha UAV lock on a target. Same with alpha UAV, it will pursue the target but still keep it in a safe distance. The normal UAV will be added to ‘attackingW’ collection of its target as soon as its target is in its influence range.
* Shooting: alpha UAV will wait for the rest of the group to move closer to the tank so that the tank is in their attacking range. Then, alpha UAV decides to attack: every UAVs in the group will head towards the tank and shoot.

The ‘stop’ action is simulated by giving the UAV a stopTick (=20) so that it cannot move in the next 20 times that trigger-to-move event occurs.

Firstly, ‘stopTick’ is assigned with 20.

Then, in the function simulating movement of UAVs



……(functions implementing movements of UAVs)



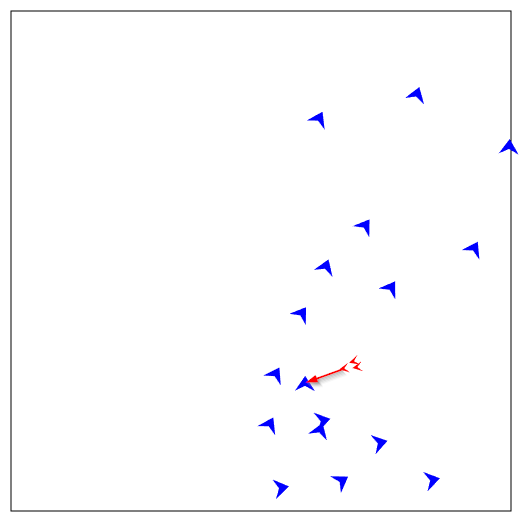


Fig 5. UAVs stopped and they are heading towards a target and shooting

1. S: set of static attributes

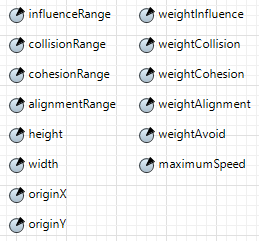


Fig 6. Set of UAVs’ static attributes

1. D: set of dynamic attributes

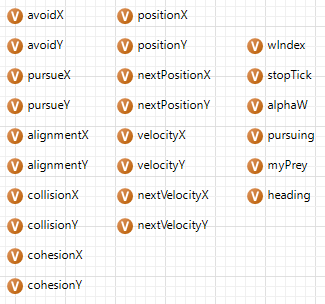


Fig 7. Set of UAVs’ dynamic attributes

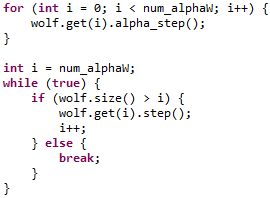
1. N: set of neighbouring agents

* The alpha UAVs will interact with tanks if any tanks is in its influence range. If the alpha UAVs decided to pursue the tank, it will run after the tank with other UAVs in its group. If alpha UAVs decided to attack the tank, it will head toward the tank and stop to start shooting until shooting phase is completed and tank is destroyed.
* The UAVs that are not alpha will follow alpha UAVs, they will try to stay cohesively and align with the alpha UAV, they also try to avoid other UAV in the same group.

1. M: mechanism for updating the agent’s states.

After every 0.05 second, a function alpha\_step() will be called and move the alpha UAVs to a proper position depends on its current state. After that, fuction step() will be called and move normal UAVs to a calculated position cosidering the collision effect, alignment effect, cohesion effect, influence effect and it also depends on the state of their corresponding alpha UAVs at the time whether they are scanning, pursuing or shooting the tank.

Alpha UAVs move and then the normal UAVs follow its corresponding alpha UAVs:



The UAVs start simulation with scanning state, after the alpha UAVs decide to pursue a tank because the tank is in its detect range, other UAVs in the corresponding group will pursue the tank. The UAVs change to shooting state when all UAVs in the group have the tank in their shooting range. The UAVs change back to scanning after shooting for 1 second and the progress start all over again.

A tank agent can be formalised as b = {B,S,D,N,M}:

1. B: set of behaviours of a tank:

* Move: the tank mimic the flocking behaviour, they move as a boid.

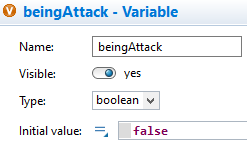
Their movements are done by the following equations:





* Being attacked: if the tank is targeted by at least 3 UAVs, it stop for 1 second before getting destroyed.

This state is determined by the following variable:



‘beingAttack’ is a boolean and its value becomes true only if the tank is being attacked.

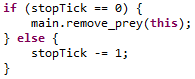
The ‘stop’ action is simulated by giving the tank a stopTick (=20) so that it cannot move in the next 20 times that trigger-to-move event occurs.

Firstly, ‘stopTick’ is assigned with 20.

Then, in the function simulating movement of the tank



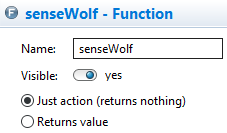
….( functions implementing movements of tanks)

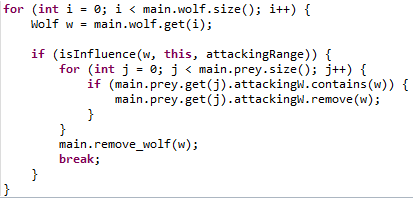


The stopTick count down to 0 and when it reachs 0, the tank will be removed from the battlefield.

* Shoot: tank can shoot and destroy any UAV as soon as the UAV is in its attacking range. It can only shoot one UAV at a time.

This is done by the function senseWolf()





1. S: set of static attributes

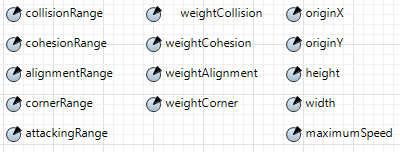


Fig 8. Set of tanks’ static attributes

1. Set of dynamic attributes

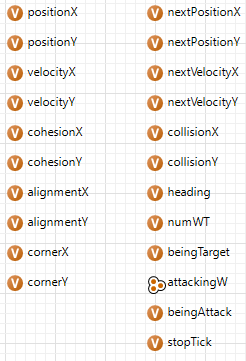


Fig 9. Set of dynamic attributes

1. N: set of neighbouring agents

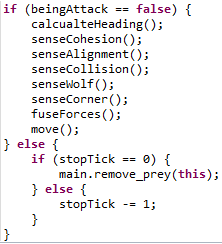
The tanks interact with other tanks depend on its alignmentRange, collisionRange and cohesionRange.

* Cohesion: each tank moves towards the average position of its neighbouring tanks
* Alignment: each tank turns to the average heading of its neighbouring tanks
* Collision: each tank steers to avoid collisions with its neighbouring tanks

The tanks interact with UAVs if UAV is in its influenceRange by steering to run away from UAVs

1. M: mechanism for updating agent’s states:

After every 0.05 second, a function step() will be called and move the tanks to a calculated position cosidering the collision effect, alignment effect, cohesion effect and influence effect.



Function step()

The tanks start the simulation with “move” state. Then, if any tank is targeted by at least 3 UAVs, the tank is changed to “being attacked” state, it stops for 1 second and get destroyed by the UAVs.

There are two conditions for the simulation to finish running:

* The number of UAVs is less than 3. When the number of UAVs is less than 3, they cannot conduct a successful ambush and hence cannot destroy any tanks. This means that the UAVs’ mission has failed.
* The number of tanks is equal to 0. This means that the UAVs’ mission has been compeleted successfully.

The detailed explanation of the main variables in the simulation is shown in Appendix A. The instructions on how to run the simulator is shown in Appendix B

To ensure that the simulator reproduce the behaviours of the intended system, I conduct testing by changing variables related to the behaviours of the agents and observe how the simulator runs. This process is demonstrated in Appendix C.

# Experimental design

My hypothesis is that if we apply the GWB for multi-UAVs system in survelliance and combat mission then the system will fight more effectively (have a higher chance of winning and complete the mission in shorter period of time). Moreover, the simulation also study:

* if we deploy more UAVs participating in the mission then they will fight more effectively
* if we increase the UAVs’ influence range then they will fight more effectively.

I will use 2k factorial design as a factor screening method to estimate factors’ effects on the response. There are three main factors that can cause large response variations in the experiments including:

* Whether the UAVs are following GWB. This is Factor 1, ‘+’ means that UAVs are following GWB and ‘-’ means that UAVs are not.
* The number of UAVs participating in the mission. This is Factor 2, ‘+’ means that 12 UAVs are participating in the mission and ‘-’ means that there are only 6 UAVs
* Influence range of UAVs. This is Factor 3, ‘+’ means that the influence range of UAVs are 100 units and ‘-’ means that the influence range of UAVs are 50 units.

There are two responses associating with two outputs produced by the simulator:

* The percentage that UAVs successfully complete the mission. This is Response 1.
* The amount of time that UAVs take to destroy all tanks. This is Response 2.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Design points | Factor 1 | Factor 2 | Factor 3 | Response 1 | Response 2 |
| 1 | - | - | - | 10.8 | 639.39 |
| 2 | + | - | - | 77.8 | 154.7 |
| 3 | - | + | - | 96.5 | 129.97 |
| 4 | + | + | - | 98.1 | 90.48 |
| 5 | - | - | + | 76.8 | 167.51 |
| 6 | + | - | + | 89.3 | 140.1 |
| 7 | - | + | + | 100 | 70.26 |
| 8 | + | + | + | 99.8 | 83.83 |

For design points, I make the simulator run for 1000 times and take average as the responses’ value.

Considering Response 1 – the percentage that UAVs complete the mission successfully. The main effect of all 3 factors are calculated by the equations:

e1\_1 = ((R1\_2 – R1\_1) + (R1\_4 - R1\_3) + (R1\_5 - R1\_6) + (R1\_8 – R1\_7)) / 4 = 20.225

e1\_2 = ((R1\_3 – R1\_1) + (R1\_4 – R1\_2) + (R1\_7 – R1\_5) + (R1\_8 – R1\_6)) / 4 = 34.925

e1\_3 = ((R1\_5 – R1\_1) + (R1\_6 – R1\_2) + (R1\_7 – R1\_3) + (R1\_8 – R1\_4)) / 4 = 20.675

Considering Response 2 – the amount of time that UAVs take to destroy all tanks. The main effect of all 3 factors are calculate by the equations:

e2\_1 = ((R2\_2 – R2\_1) + (R2\_4 – R2\_3) + (R2\_5 – R2\_6) + (R2\_8 – R2\_7)) / 4 = -134.505

e2\_2 = ((R2\_3 – R2\_1) + (R2\_4 – R2\_2) + (R2\_7 – R2\_5) + (R2\_8 – R2\_6)) / 4 = -181.79

e2\_3 = ((R2\_5 – R2\_1) + (R2\_6 – R2\_2) + (R2\_7 – R2\_3) + (R2\_8 – R2\_4)) / 4 = -138.21

According to the results obtained above, the number of UAVs participating in the mission is the most important factor. An increase in the number of UAVs contributes effectively to the probability for a mission to be successful and the amount of time taken to complete the mission.

# Analysis, Discussion and Conclusion

The output datas for all design points are stored in text files. For example, design point 1 have output data stored in “test1.txt” and design point 2 have output data stored in “test2.txt”.

Responses 1 from 5 Replications

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Design point | 1 | 2 | 3 | 4 | 5 |
| 1 | 10.8 | 11.3 | 11.5 | 9.4 | 10.0 |
| 2 | 77.8 | 78.1 | 77.6 | 75.8 | 79.8 |
| 3 | 96.5 | 96.5 | 96.6 | 96.2 | 95.6 |
| 4 | 98.1 | 98.7 | 98.0 | 98.30 | 98.8 |
| 5 | 76.8 | 74.1 | 75.9 | 75.3 | 76.7 |
| 6 | 89.3 | 89.4 | 89.8 | 88.4 | 89.6 |
| 7 | 100 | 100 | 100 | 100 | 100 |
| 8 | 99.8 | 99.6 | 99.9 | 99.8 | 99.8 |

Response 2 from 5 Replications

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Design point | 1 | 2 | 3 | 4 | 5 |
| 1 | 639.39 | 589.96 | 1073.22 | 336.12 | 340.96 |
| 2 | 154.7 | 165.09 | 158.35 | 155.5 | 156.39 |
| 3 | 129.97 | 130.23 | 129.46 | 130.91 | 129.72 |
| 4 | 90.48 | 97.03 | 95.30 | 91.36 | 92.55 |
| 5 | 167.51 | 156.92 | 158.92 | 158.73 | 161.54 |
| 6 | 140.1 | 143.59 | 140.96 | 137.68 | 137.37 |
| 7 | 70.26 | 71.83 | 73.05 | 70.06 | 70.83 |
| 8 | 83.83 | 85.98 | 85.97 | 83.74 | 82.63 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Effect | 1 | 2 | 3 | 4 | 5 | ej(5) | Sj^2(5) |
| e1\_1 | 20.225 | 20.975 | 20.325 | 20.35 | 21.425 | 20.66 | 0.216 |
| e2\_1 | -134.505 | -114.3125 | -238.5175 | -56.885 | -58.5275 | -120.5495 | 4410 |

Given that t(4,0.95) = 2.13

E(e1\_1) = 20.66 +/- 0.4427

E(e2\_1) = -120.5495 +/- 63.258

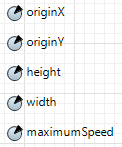
Thus all effects appear to be significant because their confidence intervals do not include zeros. The effect of GWB significantly contributes to the fighting effectiveness of UAVs in surveillance and combat misison. Therefore, I can conclude that my hypothesis is true. In other words, if the multi-UAVs system follows the GWB, it will be more effective in surveillance and combat mission.

The key finding of this simulation is that we can significantly increase the fighting effectiveness of UAVs in surveillance and combat mission by applying GWB for multi-UAVs system. Probably, this is the much simpler method than 2 other methods mentioned above including increasing the number of participating UAVs and increase the influence range of UAVs. While increasing the number of UAVs participating in the mission is sometimes impossible, increasing the influence range of UAVs requires a technological breakthrough that rarely happens.

# Appendix A

# Explanation of main variables

Below are common variables for all agents including tanks, normal UAVs and alpha UAVs:

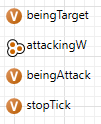


‘originX’ and ‘originY’ mark the top left corner of the battlefield.

‘height’ and ‘width’ define the size of the battlefield.

‘maximumSpeed’ stores the maximum speed of each agents. (the maximum speed of UAVs is twice as large as the maximum speed of tanks)

Below are the main variables of a tank



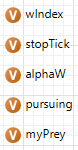
‘beingTarget’ is a boolean. Its value become ‘true’ only if a tank is being pursued by at lease 1 UAV

‘attackingW’ is a collection (TreeSet) that stores all wolves that are currently pursuing the tank

‘beingAttack’ is a boolean. Its value become ‘true’ only if a tank is being attacked by at lease 3 UAVs.

‘stopTick’ is an integer. Its value represents the number of steps that this tank have to stand still because it is being attacked.

Below are the main variables of an UAV



‘wIndex’ is an integer. It represents the index of the UAV. It is used to compare UAVs when they are added into collection ‘attackingW’ – TreeSet

‘stopTick’ is an integer. Its value represent the number of steps that this UAVs have to spend on conducting ambush on its target.

‘alphaW’ is an object. It represent the alpha UAV of this normal UAV. If this UAV is alpha UAV then ‘alphaW’ is empty

‘pursuing’ is a boolean. Its value become true only if this UAV is pursuing a tank

‘myPrey’ is an object. It represent the tank that this UAV is pursuing. If this UAV is not pursuing any tank then ‘myPrey’ is empty.

# Appendix B

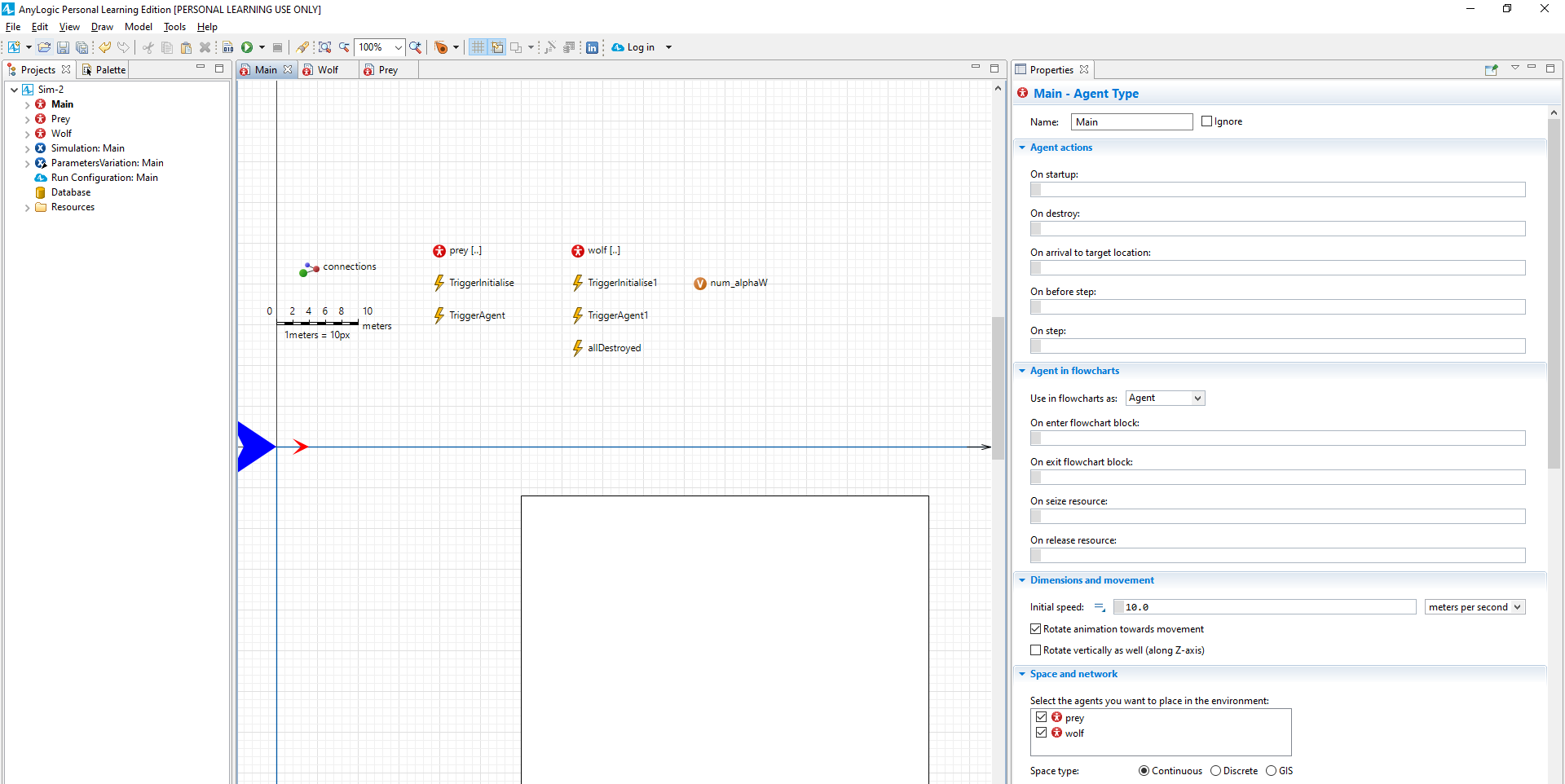
# Instructions to run the simulator

The simulators are these 2 files:

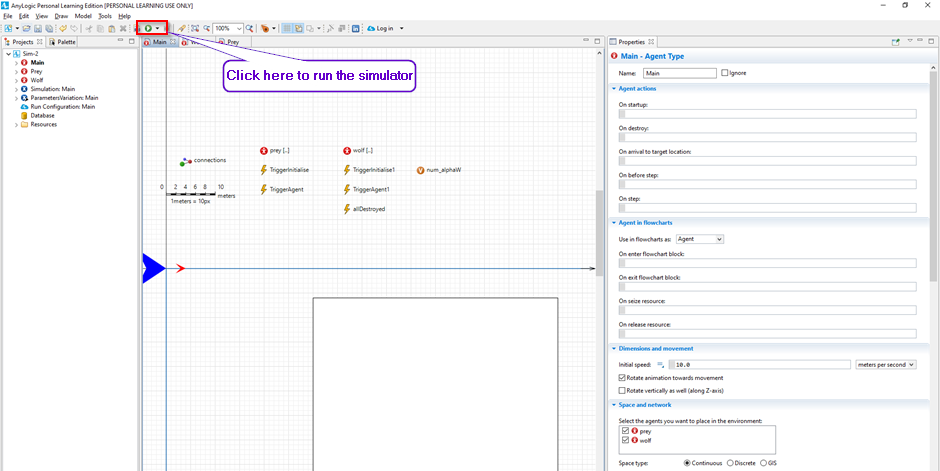


‘Sim-1’ is Simulation 1 and ‘Sim-2’ is Simulation 2

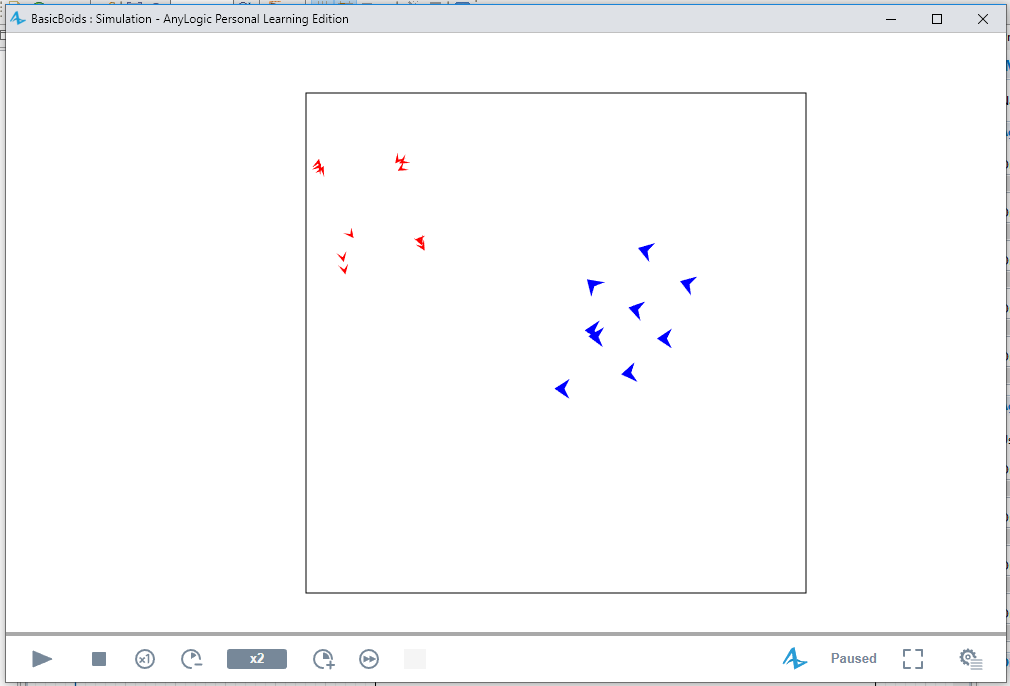
Click to open the simulators:



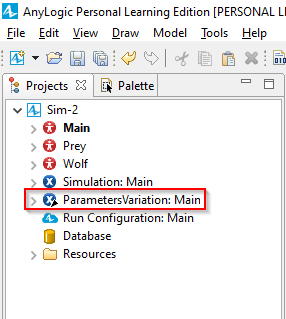
To run a single simulator, simply click run:



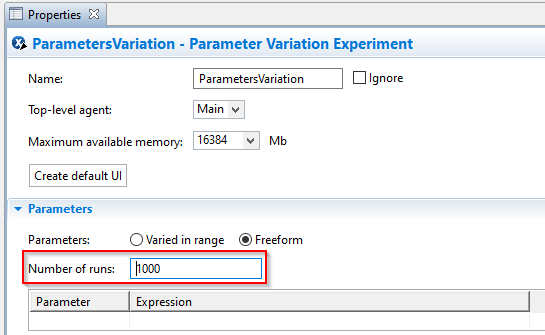
A window will pop up as following:



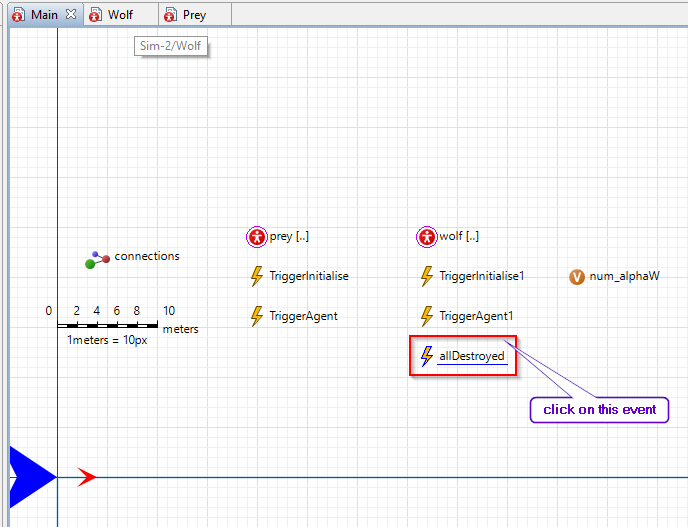
To run the simulator in a while loop, click on this on the left panel:



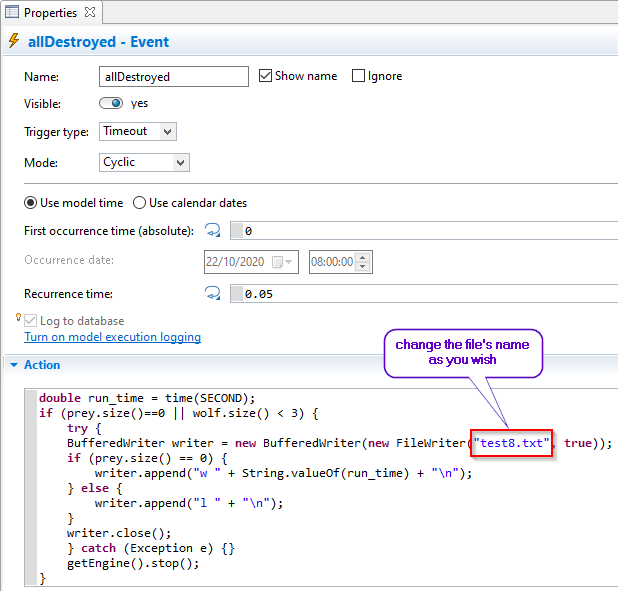
Enter the number of times the simulator will run by inserting number in the right panel:



Notice: the ouput data (including who wins the battle and the battle’s duration) is stored in the a predefined file which can be changed. Do the following steps to change the file that stores the output data:

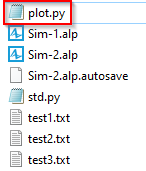


Then, change the file name here: (please delete any file before creating a file with same name since the data will be appended and can not be read by other program)

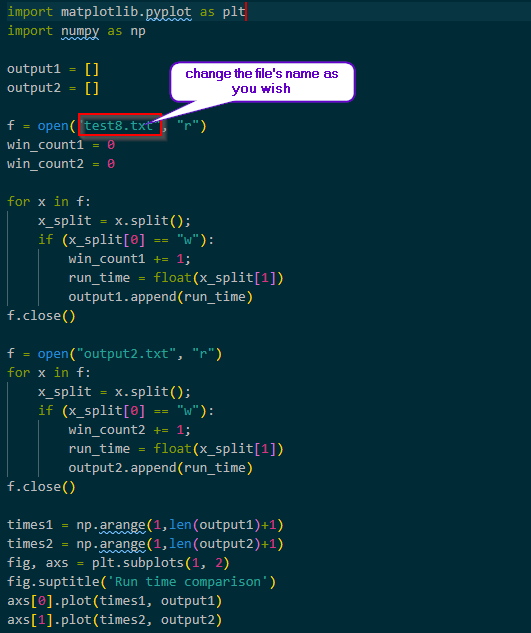


You can plot graph to visualize the result. Do the following steps to plot the graph:

Open this python file by any python IDE



Change the name of the file that contains your output data:



Then run the python program in terminal:



Graph will pop up as following:

