# Overview

The game consists of 2 AI entities: swarmers and leaders. Swarmers are dumb with hundreds of them at any given time, while leaders are more intelligent and fewer in numbers. A leader will recruit swarmers at their spawner before going to fight the player, then return to get more after enough of them have died.

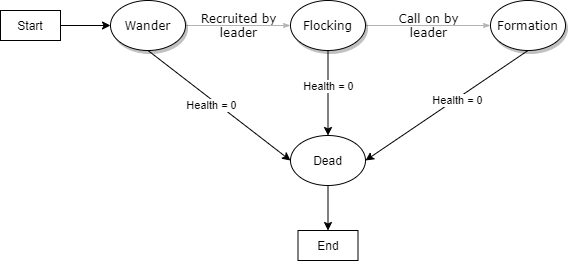
Swarmers have 4 states:

* **Wander:** After being created by a spawner, it will wander around nearby.
* **Flock:** When a leader recruits a swarmer, it enters a flocking state, becoming a boid while also seeking the leader.
* **Formation:** When a leader tells it to, a swarmer will enter a formation state, where it will have a greater impact on other boids and follow the formation set by the leader.
* **Dead:** When a swarmer’s health drops to 0, it dies and is destroyed.

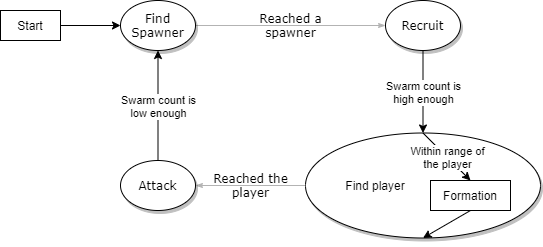
Leaders have 5 states:

* **Find spawner:** The leader follows a path to the nearest spawner.
* **Recruit:** After reaching the spawner, it will recruit swarmers into its swarm.
* **Find player:** After the swarm is large enough, the leader follows a path to the player.
* **Formation:** When the leader is close to the player, it will gather the swarmers into a ‘U’ shape to surround the player.
* **Attack:** Upon reaching the player, the leader will send the swarm to fight the player, until too many have died, and it must go recruit more.

***Swarmer***:



***Leader:***



%3CmxGraphModel%3E%3Croot%3E%3CmxCell%20id%3D%220%22%2F%3E%3CmxCell%20id%3D%221%22%20parent%3D%220%22%2F%3E%3CmxCell%20id%3D%222%22%20style%3D%22edgeStyle%3DorthogonalEdgeStyle%3Brounded%3D0%3BorthogonalLoop%3D1%3BjettySize%3Dauto%3Bhtml%3D1%3BexitX%3D1%3BexitY%3D1%3BexitDx%3D0%3BexitDy%3D0%3B%22%20edge%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20relative%3D%221%22%20as%3D%22geometry%22%3E%3CmxPoint%20x%3D%22663.8379480147864%22%20y%3D%22212.7959108208015%22%20as%3D%22sourcePoint%22%2F%3E%3CmxPoint%20x%3D%22637.5%22%20y%3D%22220%22%20as%3D%22targetPoint%22%2F%3E%3C%2FmxGeometry%3E%3C%2FmxCell%3E%3C%2Froot%3E%3C%2FmxGraphModel%3E

Raylib will be used as the graphics library to create the game primarily due to my familiarity with it, however, its efficiency lends itself well to this project due to the number of entities that must be simulated and displayed. This permits the use of even larger numbers of swarmers to be used.

The only licence applicable is raylib’s, being a ‘zlib/libpng’ license, essentially stating that anything can be done with it, as long as it does not misrepresent the original software and contains the licence. Its impact on this project is simply that its licence be included with the library, which it is.

# Implementation

The leader entity uses a finite state machine to implement its logic, while the swarmer uses compound behaviours that can be activated or deactivated to switch state. The leader has two instances of pathfinding: the find player and find spawner states. The find player state has a single target, making A\* the appropriate algorithm to use. The find spawner state looks for the closest spawner, however, making Dijkstra’s algorithm a better choice for it. Pathfinding is implemented under the game manager, as it allows an entity to access it.

## Swarmer

Swarmers are very basic, mostly being controlled by a leader; it stays in wander until recruited and stays in flocking until ordered by a leader. The only exception is when it dies, which can be triggered at any time and results in it being destroyed. This, in addition to the number of swarmers being simulated, makes a state machine unnecessary. Instead, swarmers will use behaviours that can be activated or deactivated by its public functions. These functions are used by a leader to transition a swarmer between states.

Wander is only used after being created by a spawner and consists of a wander behaviour using a random target within an area around where its created. The behaviour will also have a reference to a vector containing all leaders and checks if any of them are recruiting by checking a flag. If a leader is recruiting and within range, it will call a function on the swarmer to join the leader’s swarm. This function will call a function on the leader, adding it to their swarm, and change the active behaviour to a specialised flocking behaviour.

This behaviour has 2 phases; when the swarmer is too far away from the leader, it will seek towards them until within the radius, then it will use a combination of wander and flocking steering forces, with its neighbourhood being the leader’s vector representing the swarm. The effect of this is an erratic, bug-like swarm, while preventing it from drifting away from the leader.

The leader can make a swarmer enter a formation state by calling one of its functions. This function takes a target, offset, and importance value. The function passes the first two values to a formation behaviour and sets it to its active behaviour, which uses the arrival steering force to keep the swarmer in formation. Importance is used by the alignment and cohesion flocking forces, multiplying a boid by its importance before adding it to the total force. This is done to make certain agents able to steer or shape a flock.

When a swarmer has taken enough damage to reduce its health to 0, it is considered dead and cleans up after itself before being destroyed. It will call a function on its leader to remove it from its swarm and delete any smart pointers it holds. The abstract class GameObject will then remove it from the game manager.

## Leaders

Leaders have a full state machine, populated by 4 states. The recruit and attack states are very similar; stay at a location and wait for the swarm count to reach a threshold, as are the find spawner and find player states; get a list of nodes from its current position to the target and follow them.

Recruiting starts by setting a member Boolean to true, flagging swarmers to join its swarm. Swarmers do this using a public method that increments a swarm count and adds the swarmer to a vector of swarmer pointers, then returns a pointer to it. When the swarm count has passed a set threshold, a transition to the find player state will trigger.

The find player state also contains the formation sub-state. It starts by getting a path from its current position to the player’s position from the game manager. It then uses an arrival steering force to move to each node, moving to the next node after coming in range of the current one. A transition is triggered when the final node is reached, moving to the attack state.

The formation sub-state is just a process that is called by the find player state when its within a certain distance from the player. The function selects swarmers from the leader’s swarm, adding them to a separate vector of formation members, and calls a function for them to enter a formation around the leader and increase their importance. The formation they form is a ‘U’ shape with the leader at its base. The effect of this is to shape the swarm into the same shape, surrounding the player.

The attack state starts by changing the formation from following the leader to following the player near and further increasing their importance to have the swarm gather on the player. The leader keeps their distance from the player using an arrival and modified flee steering force producing a ring around the player where the forces cancel out. As swarmers die, the swarm count is reduced, and when it goes below a threshold, a transition to the find spawner state is triggered.

The find spawner in the state that leaders are instantiated with, but immediately transition to the recruitment state as they are created at a spawner. The state starts by getting a path from the leader’s current position to the nearest spawner from the game manager. The path is followed in the exact same way the find player state does; using an arrival steering behaviour. The transition to the recruiting state is triggered when the final node has been reached.