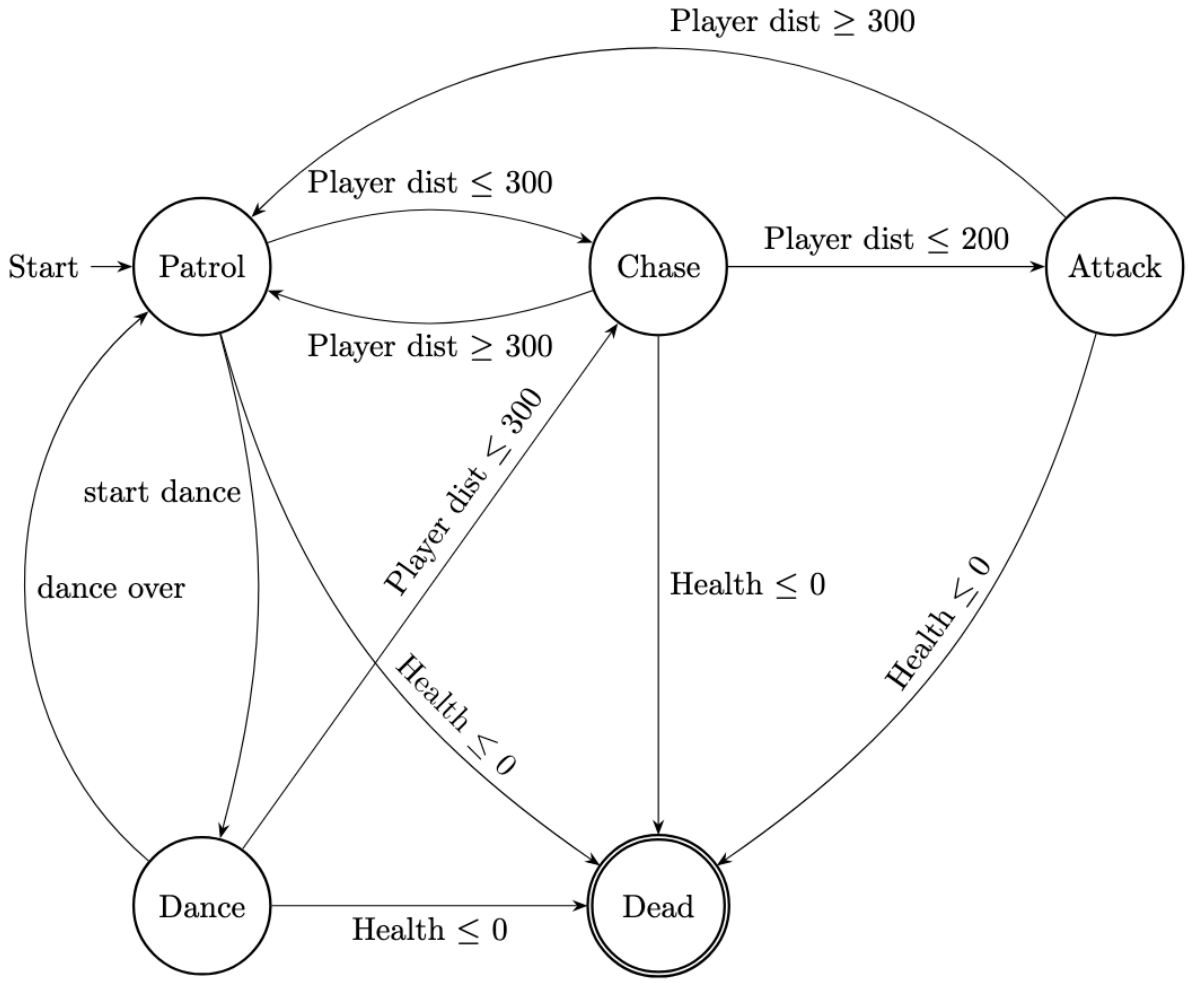
FSM diagram



(start dance and dance over are based on timers that tick in the patrol and dance states, respectively)

Implementation Logic Algorithm (steps 3, 4, and 5)

Change Tank Color based on State

1. Make a New Shader that Declares a Color Variable:
    - expose a single color variable that can be changed from a script.
2. Modify FSM Script to Map States to Colors
    - The modified script should create a map, linking each AI state to a specific color
3. Modify FSM Script to Update the Shader
    - Make it so that when the AI's state changes, the script finds the corresponding color from its map and sends it to the shader's color variable.
4. Make Shader Paint the Object
    - Program the shader's fragment function so it simply paints every pixel of the object with the color it gets from the script.

Implementing the 'Dance' State

1. Create the New State
    - Make a Dance state and add it to the FSMState enum.
2. Make the 'Dance' Action
    - Make a new function for updating the dance state. The code inside this function should define the dancing behavior.
3. Add an Entry Condition
    - Modify the UpdatePatrolState() function so that after a random amount of time passes (Tank is bored), the script tells the AI to switch from Patrol to the new Dance state.
4. Add Exit Conditions
    - Modify the UpdateDanceState() to exit the dance. The AI stops dancing and returns to Patrol after its dance timer runs out, or it will immediately switch to Chase if the player gets close.

Implementing the On-Screen Health Bar

1. Link the UI Element in the Editor
    - Add an exposed image variable to the script that can be linked in the inspector.
2. Convert Health into a Property
    - This allows us to run code when its value is changed.
3. Update UI When Health Changes
    - Inside the set block of the new health property, calculate the health percentage and sets the visual fill of the health bar accordingly.