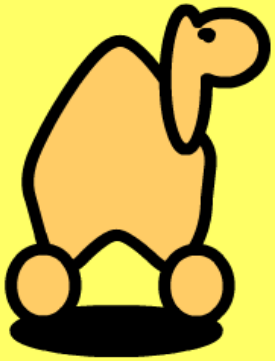
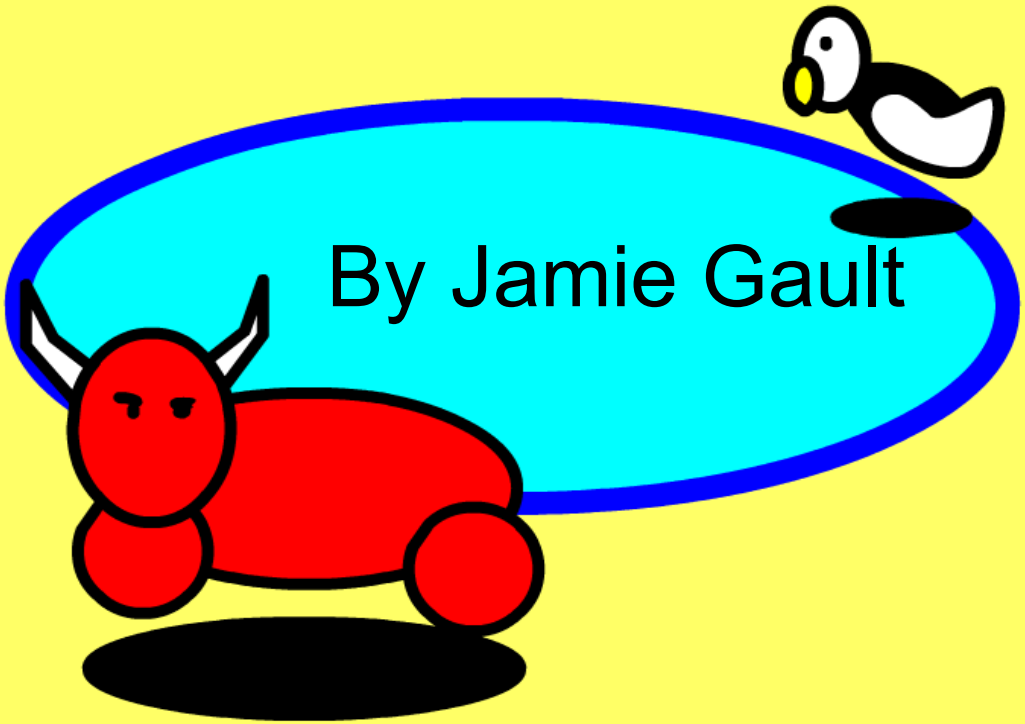
Fuzzy AI:  Animal Trail

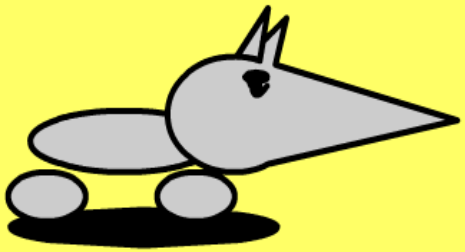


By Jamie Gault

Project Summary

- Part 1: Make game objects' and enemies' visibility based on the fuzzification and defuzzification of the states and positions of the animal characters

- Part 2: Use the same system of visibility to govern over the AI's decisions and movement.



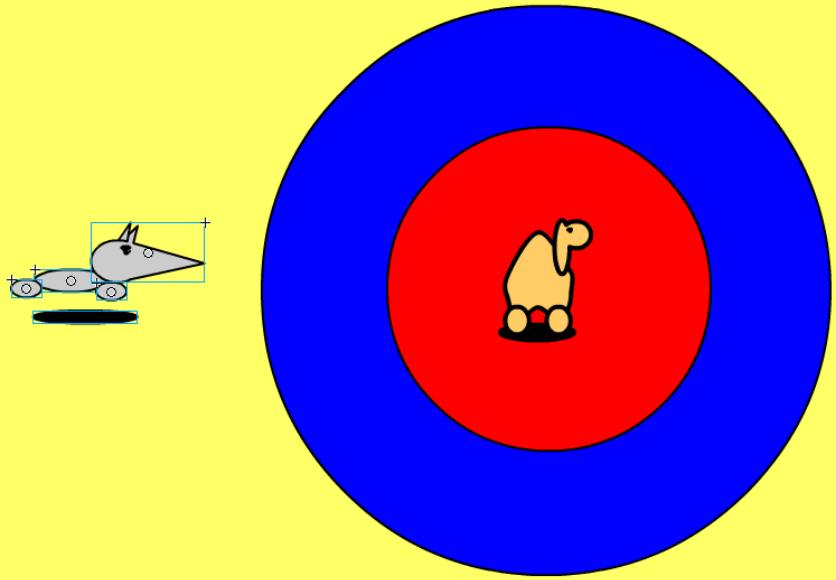
Part 1: Visibility

- Starting with the crisp set of visible and not visible, we want to declare a range of vision that allows for an interpolation between the two ranges.
- This interpolation is done by fuzzifying the data for the given distances and map to the range [0,1].
- After that, we defuzzify and map back to a range of [Not Visible, Visible]

Handling Visibility Cases

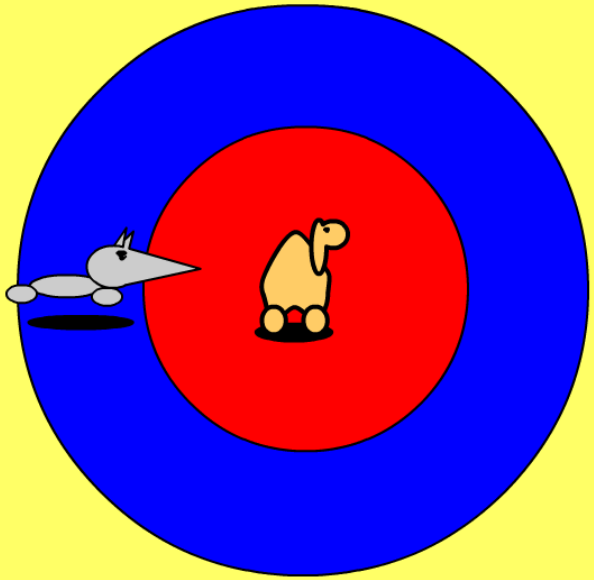
- The crisp set we're using has 2 elements of the set, so that means we have 3 cases to handle:
  - The enemy is not able to be seen.
  - The enemy can kind of be recognized.
  - The enemy can be clearly seen.

Case 1: The enemy is not able to be seen.



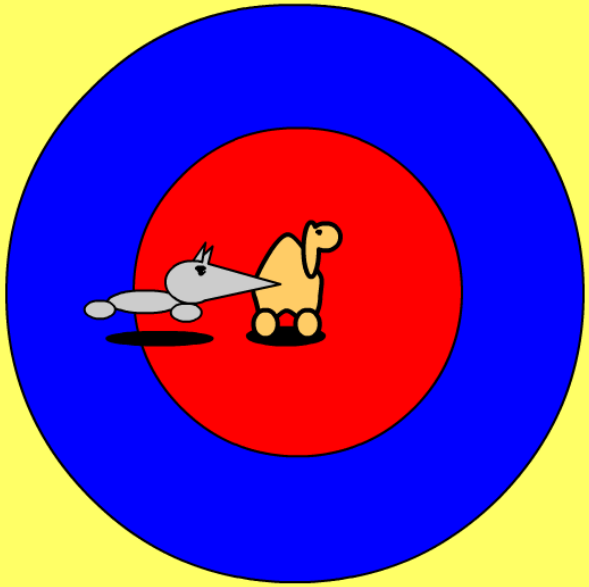
This case shows the enemy being out of range of the camel’s vision.

Case 2: The enemy can kind of be recognized.



This case shows the enemy in the fuzzified range. It is only partly visible.

Case 3: The enemy can be clearly seen.



This case shows the enemy is fully visible

Calculation of vision

- Based on the simple line equation:

Distance t € [r, s]

Scalar a € [0, 1]

$t = (1 - a)*r + a*s$

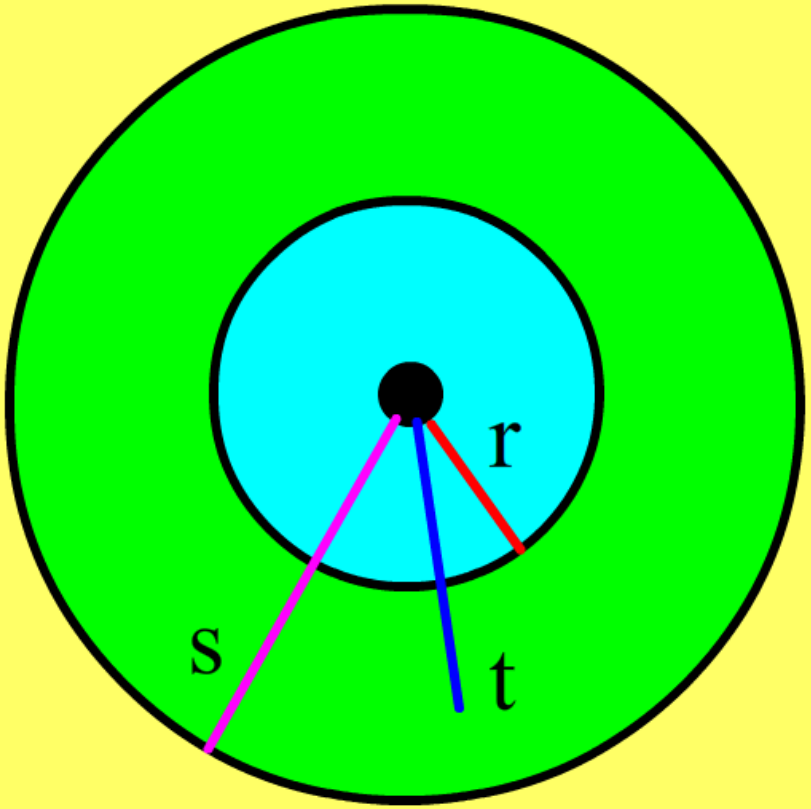
Fuzzy function:

$$a = \begin{cases} 0, & t <= r \\ -(t - r)/(s - r), & r < t < s \\ 1, & s <= t \end{cases}$$

Defuzzification function:

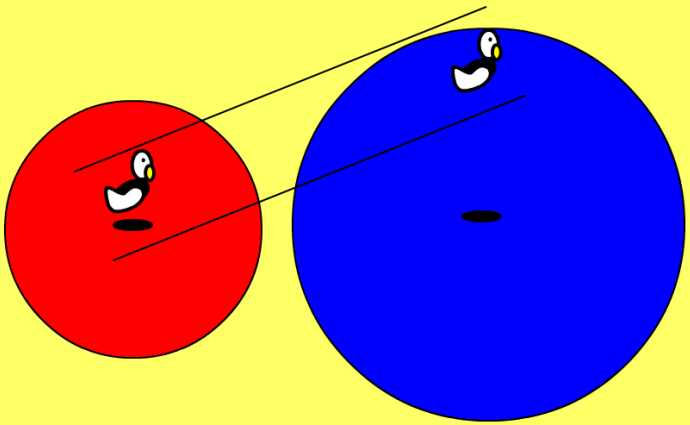
Alpha = (1 – a) * 100

This now maps the alpha value to the range of [0, 100], and is assigned to the animation of each enemy and the Oasis.



Differences in Vision

- The Eagle character's main ability is to fly high, increasing his range of vision.
- This plays a big part in the AI in that as the Eagle is able to see things more easily, so will the AI be more able to see the Eagle



Part 2: AI

- Again, this is about taking a crisp set and fuzzifying it to interpolate the range of the vision that the enemy has and the defuzzifying it so that they can react accordingly to the player's actions.
- Start by making cases and using those cases to make the piece-wise fuzzy function.

Objectives

- The AI has certain objectives it tries to maintain that it bases it's actions off of using the data given from the player's characters.

- Certain aspects like knowing when to approach one character while running away from another, and then knowing when it can stop.

Vulture

- The simpler of the 2 enemies, the Vulture's actions are based on two objective points: the Eagle and the Vulture's hub point.

- The Eagle objective is handled similiarly to the player character's visibility. The Vulture will not attack the Eagle unless it sees it, needs to interpolate between that range.

Vulture: Eagle Objective

Using the fuzzy function that we have for visibility of enemy character for the player characters, we find A € [0, 1] that represents the range of

[Not Visible, Visible]

The value A simply multiplies the unit vector of the Vulture's position to the ealges position for it's movement

(i.e. A * BirdDir = VultureVeolocity)

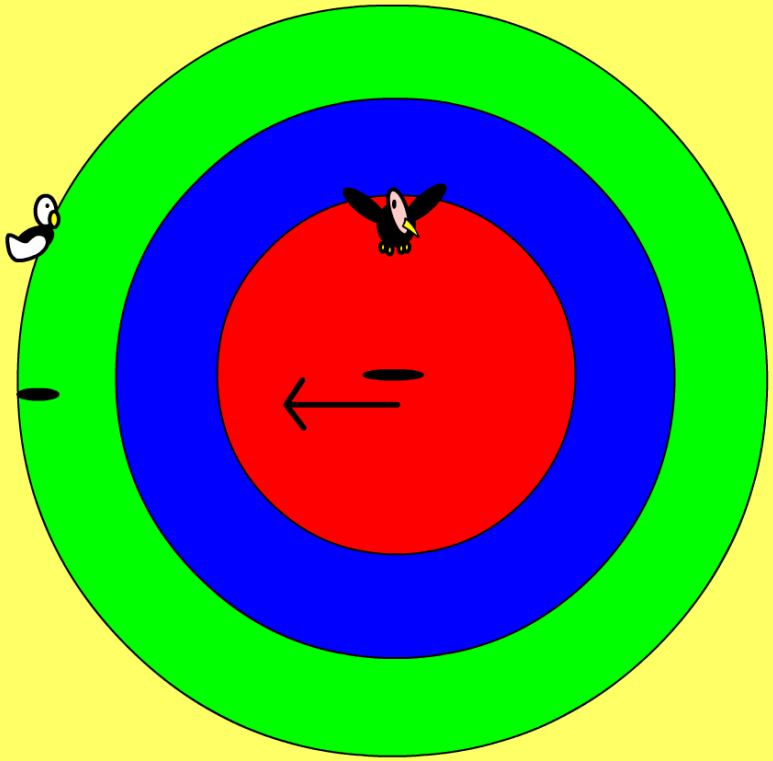
Vulture: Hub Point Objective

- All vultures are created with a hub point that they are not to stray too far from.
- In order to handle making a fuzzy function for it, again we created cases:
  - The vulture is close to the hub point.
  - The vulture is kind of close to the hub point.
  - The vulture is not close to the hub point.

Case: The vulture is close to the hub point.

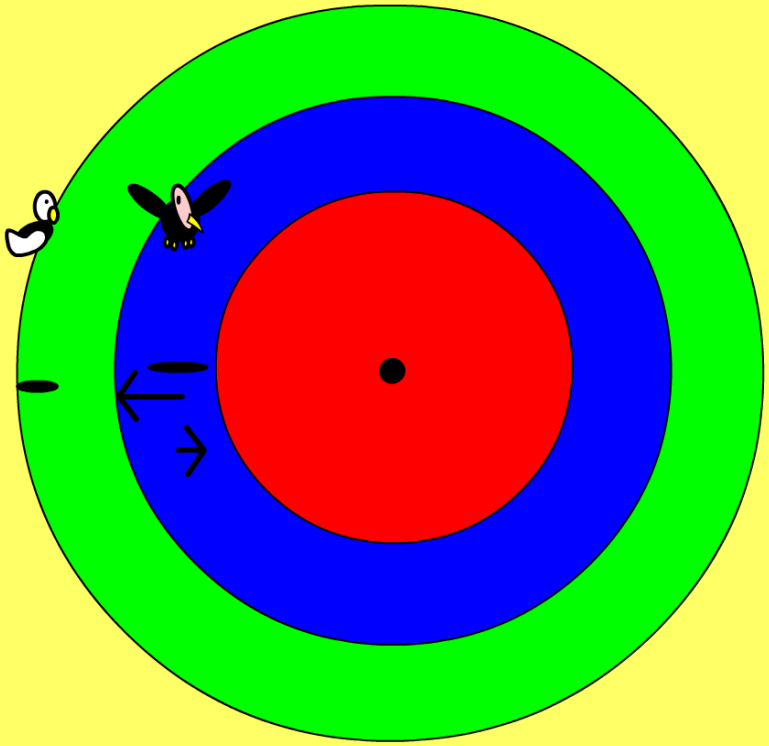
As the Eagle becomes in range of sight, the vulture advances toward the Eagle, unrestrained.

(This is all done through the same style of fuzzy function as the visibility)



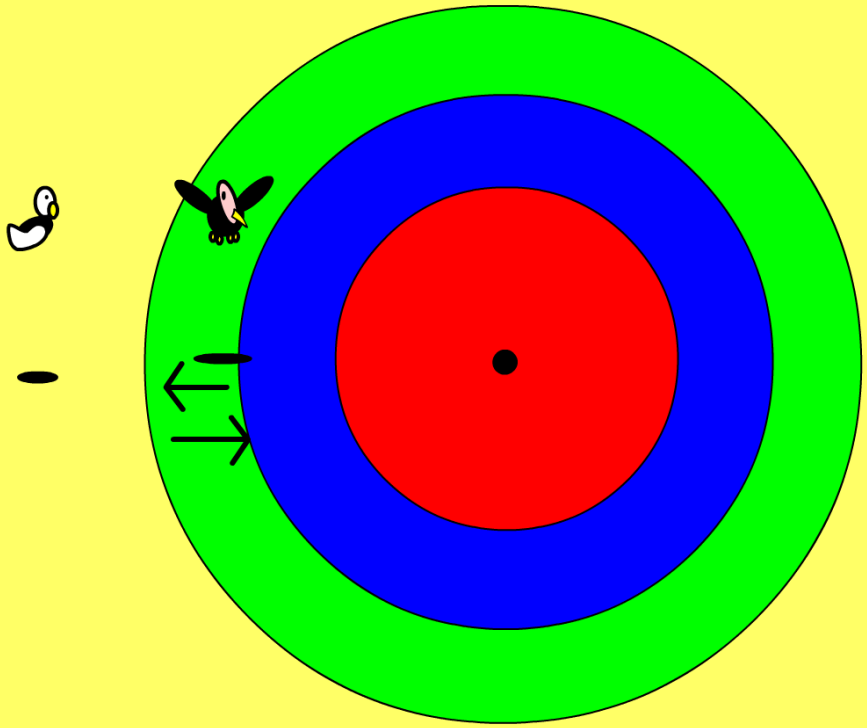
Case: The vulture is kind of close to the hub point.

The Vulture is now in a range where he will be hindered by a vector forcing him to stay close to his hub.



Case: The vulture is not close to the hub point.

This range is where the Vulture has maxed out his distance that he can leave his Hub, and is now being forced back by a vector equal to that of the vector pointing him toward the Eagle.



Vulture Movement Calculations

- The Fuzzy Values from both objectives' functions are then used in this function:

  VulDir = (A * BirdDir + B * HubDir)/2.0,

  where:
  - A is the fuzzy visibility value based on distance from the Eagle to the Vulture, (Objective 1)
  - B is the fuzzy distance value for how far the Vulture is for it's hub point (Objective 2),
  - BirdDir is the direction vector toward the bird, and
  - HubDir is the vector direction towards the Hub Point

  (The division by two is purely to slow the Vulture down to make it more interesting to play against.)

Coyote

- The coyote is the main AI for handling reactions to all three characters as well as responding to other enemies so it can know when or when not to act as a team.

- Its objectives are made to be exact counters to the player's objectives

  (such as if the killing the player when the player attempts to avoid dying and vice versa)

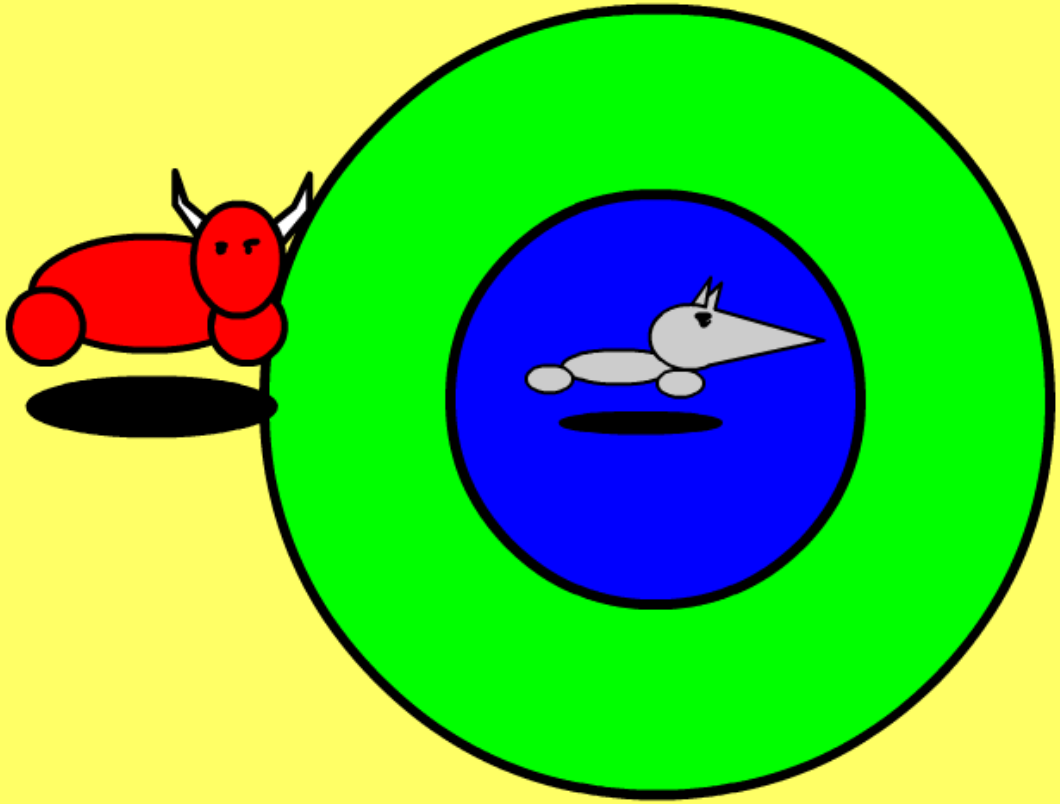
Coyote Objectives

- Staying hidden from the Bull and the Bird, unless a team is formed.
- Stay alive and not let the Bull attack unless there is enough members on the team to kill the Bull.
- Attacking the Camel when it is with in sight and not being guarded by the Bull.

Movement and Visibility Handling for the Coyote

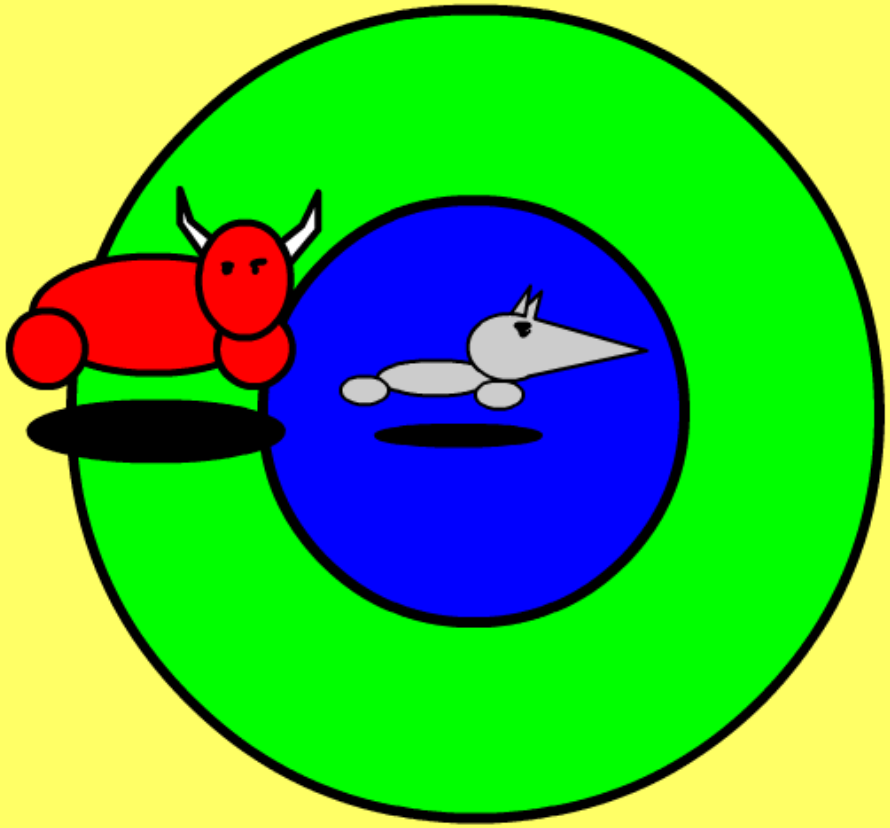
- Just like the Vulture, a coyote will have each objective be checked for visibility and direction.
- In the case of the Camel, the direction of the movement vector will be towards the Camel.
- In the case of the Bull and the Eagle, the direction of the movement vectors will be away from both. (with the exception of when a team is made)

Case: Bull/Bird Out of Range



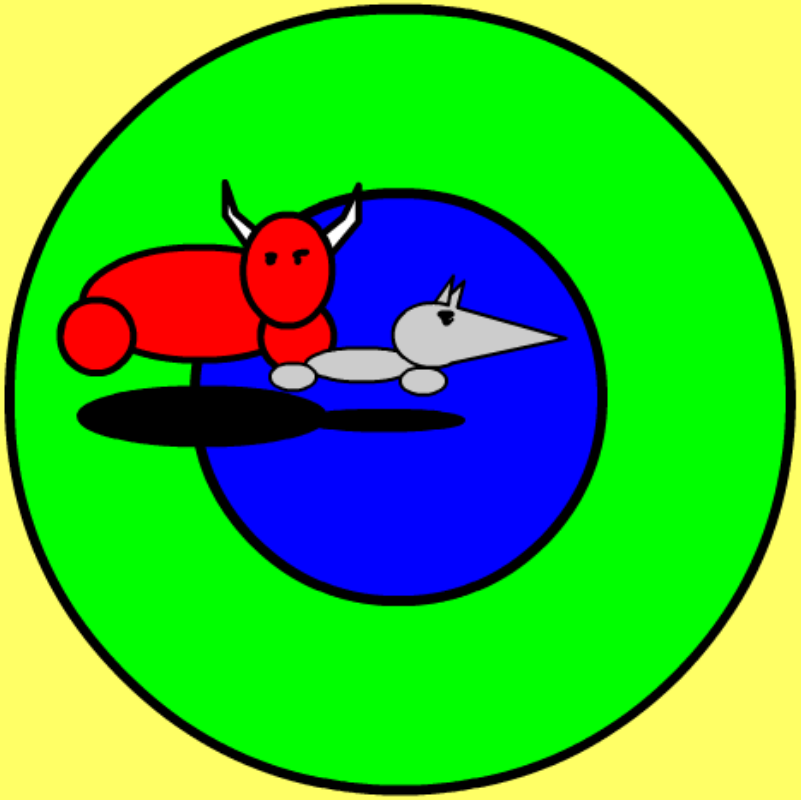
They cyote does not see the bull, so therefore should not react.

Case: Bull/Bird sort of in range



They cyote begins to notice and starts to progressively react

Case: Bull/Bird in range

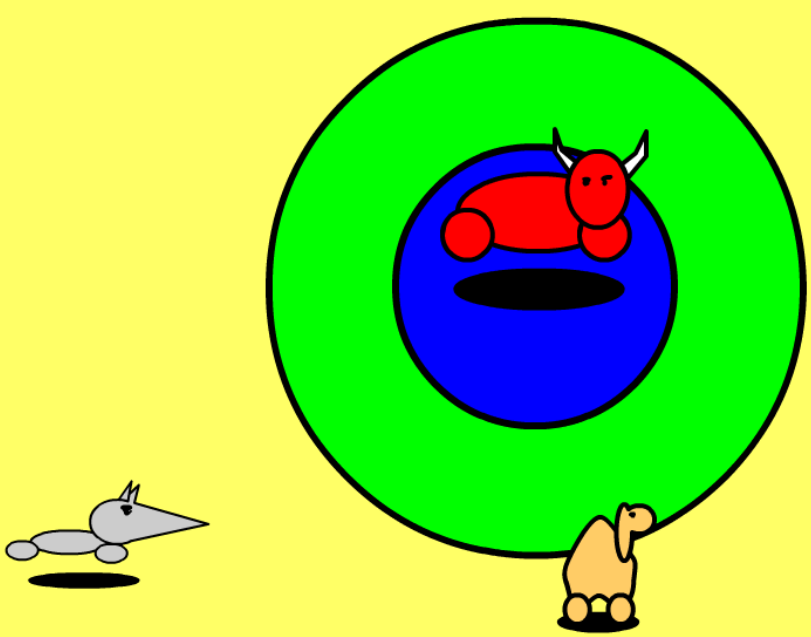


They cyote completely sees the bull and it running full speed.

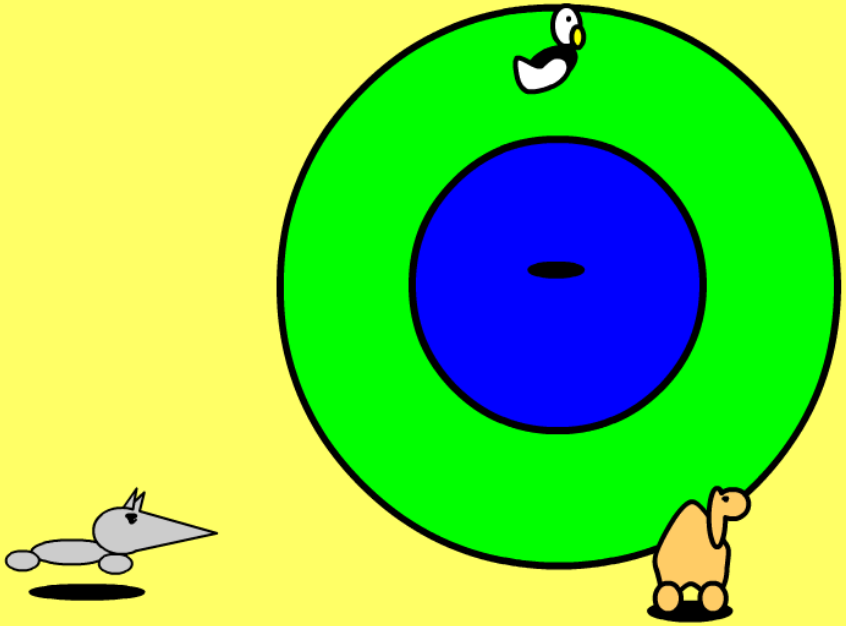
This Guard Factor

- Being that the coyotes objective is to hide and survive, and it handles this with the guard factor.
- We then create a fuzzy value telling how much the coyote should consider the other animals when going after the camel
- Guard Cases:
  - Character is guarding the Camel
  - Character is kind of guarding the Camel
  - Character is not guarding the Camel

Case: Camel not guarding

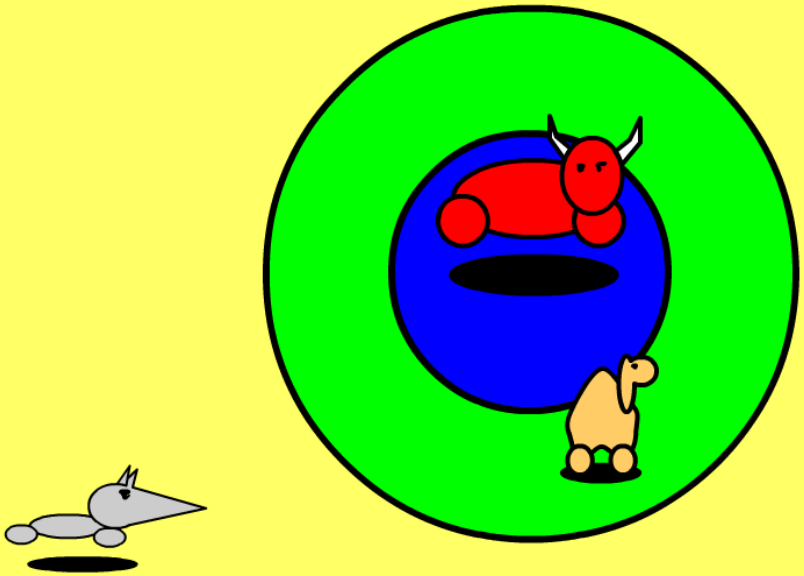


The Bull is not guarding the Camel, so go after the Camel without considering the Bull.

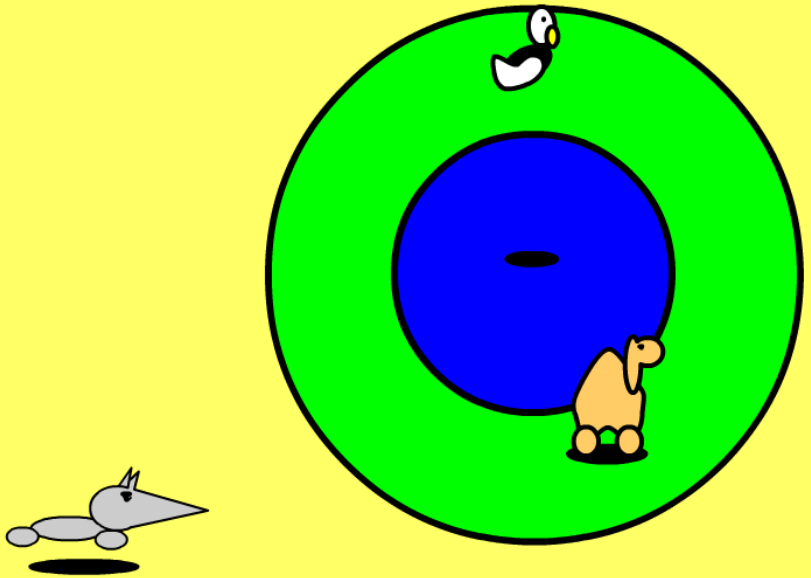


The Eagle is not guarding the Camel, so Cyote should consider the bird so that it stays out of sight

Case: Camel sort of guarded

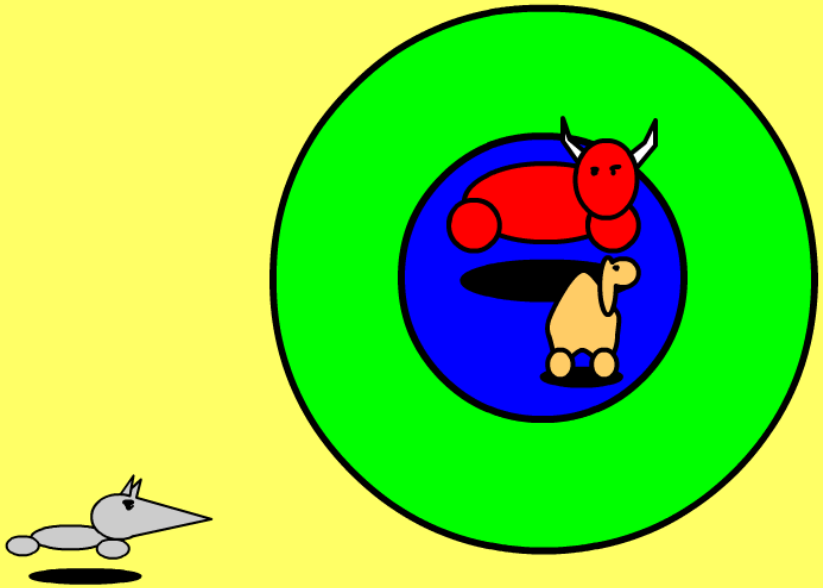


The Bull is kind of guarding the Camel, so the Coyote should slow down its pursuit.

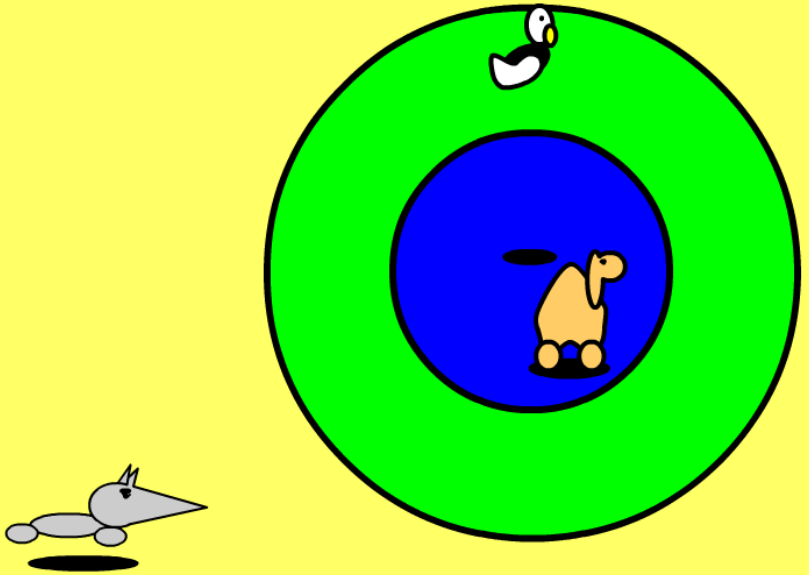


The Eagle is kind of guarding the Camel, so the Coyote should slow down its pursuit.

Case: Bull not guarding



The Bull is guarding the Camel, so don't consider attacking the Camel.

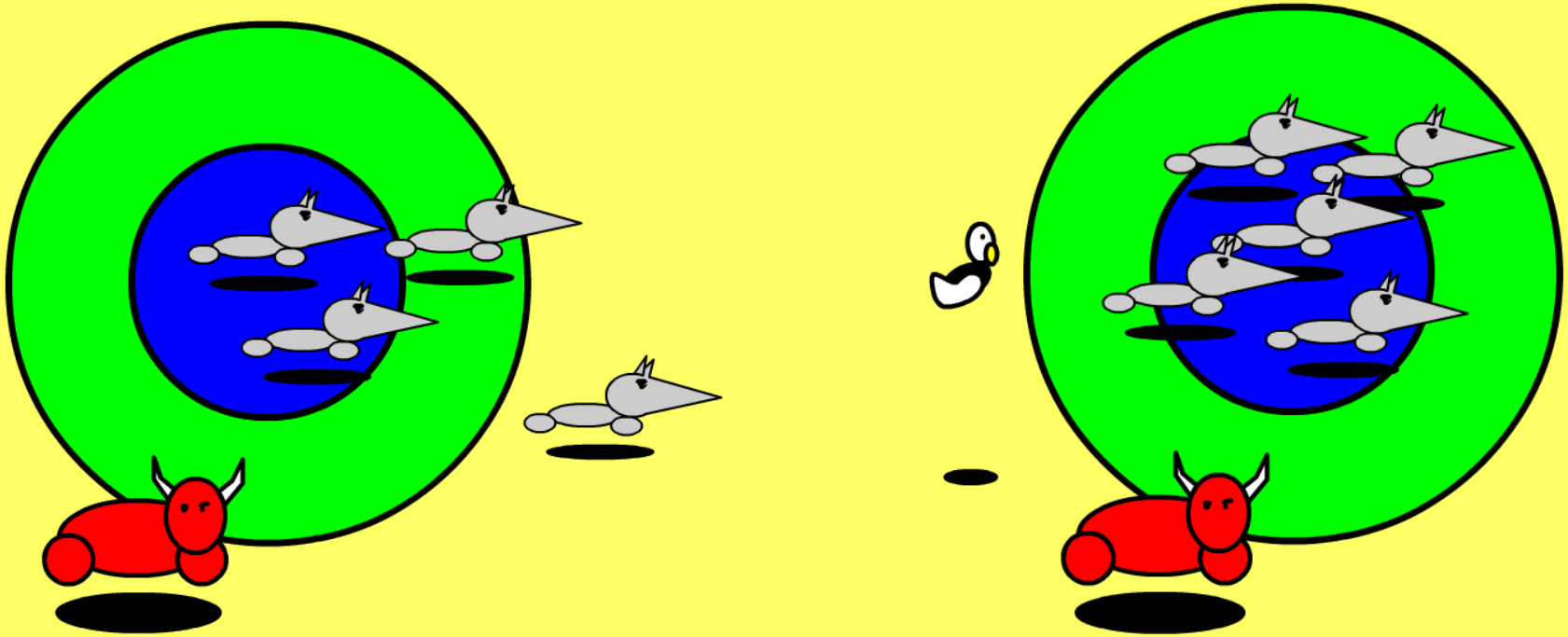


The Eagle is guarding the Camel, so disregard the Eagle and go for the Camel.

Team Factor

- A coyote on it’s own can’t exactly survive very well, but when in a group, they have a better chance.
- Every so often, each coyote checks for how many other coyotes it sees and how close they are as a way of generating a team value.
- This is instrumented into the hiding from the Eagle and the Bull aspect.

Coyote Team Formations



Ranged formation

Complete formation

Team Factor Calculations

- It first takes the fuzzy visibility values of all near by coyotes and finds the total for each coyote. This is value is then entered into the fuzzy functions for the Eagle and the Bull to consider the team aspect.

- The Eagle's value is then used to scale the direction value like in all the other cases.

- The Bull's team value is then defuzzified in a function that maps the initial fuzzy value from [0,1] to [-1,1]. This is to allow for the enemies to recognize that if they have a large enough team, they can go after the Bull and take it down simply by using a value that scales the direction vector of the bull be something in a range more similar to [Attack the Bull, Run from the Bull].

Bringing it all together

A = Fuzzy Value for how much the coyote is able to see the Camel.
[0, 1] -> [Not Visible, Visible]
B = Fuzzy Value for how much the coyote is able to see the Bull.
[0, 1] -> [Not Visible, Visible]
C = Fuzzy Value for how much the coyote is able to see the Eagle.
[0, 1] -> [Not Visible, Visible]
D = Fuzzy Value for how much the Bull is guarding the Camel.
[0, 1] -> [Guarded, Not Guarded]
E = Fuzzy Value the team value based on the Bull.
[-1, 1] -> [Attack the Bull, Run from the Bull]
F = Fuzzy Value for how much the Eagle is guarding the Camel.
[0, 1] -> [Guarded, Not Guarded]
G = Fuzzy Value the team value based on the Eagle.
[0,1] -> [Don't run from the bird, run from the bird]

CyoDir = A * D * CamDir + B * E * BullDir + C * F * G * BirdDir

And now to the game…