CS313 - Birds - Final Report

Dustin Schmidt, Karl Hinder

November 30, 2012

Contents

1 Introduction

The initial goal of the Birds game was to emulate flocking behavior without the typical need to assign entities to a flock, ie allow the flock to form as it would naturally with individual entities making informed decisions about their surrounding environment. While this is still a plausible goal our objective adapted towards developing and effective AI mechanism that would lay the groundwork for such flocking behavior. We quickly decided on Collaborative Diffusion as the basis for our mechanism as it showed promise for facilitating the desired behavior. Our final game product allows a player to witness the emergent behaviors of entities acting independently on weighted metrics in their surrounding environment. The player guides a group of birds to the nest by attracting them with strategically placed food items. Meanwhile there are hawks which seek to eat the birds before they can return to the nest.

2 Game Play

The game loads with a map populated by some birds, nests and hawks. Entities can be placed in a limited quantity (based on the contents of the map data file) by left, right, and middle clicking on the map. For development purposes entities can be removed by clicking them with any button. A nest may be placed in the game map file.

2.1 Game Overview

- Game Rules
  - Birds pursue and eat food
  - Hawks pursue and eat Birds
  - Birds flee Hawks
  - Player strategically places food to guide Birds to the Nest
- Scoring Options
  - Some number of birds reach the nest
  - Birds eat some number of food items

2.2 Key Mappings

p	Pause/Unpause the game
t	Toggle metric plotting (will slow the game play significantly)
b	Show the Bird metric layer
h	Show the Hawk metric layer
f	Show the Food metric layer(Default Plot parameter)
s	Pause and step through game one frame at a time

3 AI Component

Our game needed an AI strategy that accomplishes decisive path finding. Entities within the game are AI units which make decisions about where to go based on evaluation of various metrics their surrounding environment.

3.1 AI Objectives

- Decisive Pathfinding
- Allow Birds to find Food
- Allow Hawks to find Birds
- Allow Birds to flee Hawks

3.2 Collaborative Diffusion (CD)

3.2.1 CD Overview

The backbone of the AI for Birds is a goal-driven technique called Collaborative Diffusion (CD). CD can be thought of as a diffusion of scents throughout a grid. These "scents" are just floating-point values that are assigned to a cell in the play-grid. They originate at point of interest (such as a food item, or an enemy), and they spread (diffuse) throughout the map using very simple local rules. Here are the basic steps of CD:

For each frame (or turn):
1. Assign the highest possible diffusion value to the cells that hold features of interest. Cells can have more than one "type" of value, indicating different goals.
2. Iterate through each cell in the play-grid. For each cell, assign it diffusion value(s) as a weighted sum of its neighbor's diffusion values. (Optionally, some cells can be always given a diffusion value of 0, such as walls and barriers).
3. Repeat step 2 for a desired number of iterations. The accuracy and usefullness of the diffusion grid increases with the number of iterations, so the number of iterations is usually chosen only for performance considerations.
4. After determining all diffusion values for all grid cells, simply move the agents in the game (in this case, birds), to the adjascent square with the highest diffusion value corresponding to their current goal (or a weighted combination of several goals). With enough iterations, this path approaches an optimal shortest-path from any agent to a point of interest.

3.2.2 Benefits of Collaborative Diffusion Approach

Not only is collaborative diffusion an efficient way to find shortest paths for many agents to many goals, but it is also known to result in unpredictable, emergent behavior, especially as many goals are being diffused.

Division of labor can be acheived by making certain agents attracted to certain "scents" (diffusion goals). For example, some agents could be dedicated explorers, while others could be dedicated to seeking out and engaging combatants.

Dynamic goal-changing comes for free with collaborative diffusion when movement choices are made as a function of multiple diffusion value types. Depending on game-state and circumstance, more weight can be given to different values. For instance, food diffusion goals can be given more weight as strength decreases for a bird. Thus, if such a weak bird is presented with a choice of following a path to food or a path to fight an enemy, food will be preferred by simply changing one weight paramater.

3.3 Extention to collaborative diffusion

Collaborative diffusion is an intriguing tool but it has a number of limitations which we aimed to eliminate in order to impart more biological plausiblity.

3.3.1 Limitations of Collaborative Diffusion

Collaborative diffusion as outlined above allows diffusion values in map cells to grow in value with no bound. In each round the value of any given cell is increased by the weighted summation of its neighbors. In this case all cells will continue to grow in value. This works well if there is only one metric of interest, or even two where one is considered a goal and another an anti-goal. The pursuit of the maximum (or avoidance of the minimum) value in a neighborhood would still lead to the goal (or away from the anti-goal) because the goal/anti-goal cell is guaranteed to have the maximum absolute value of all cells in its layer.

In the interest of accurate biological modeling our game allows for multiple goals or anti-goals. In order for a game entity to distinguish among multiple simultaneous goals it is necessary to have some guarantee about the maximum value in a particular metric layer. If a hawk wishes to pursue birds as well as food but prefers to pursue birds, then it is necessary to the hawks decision that it be able to detect proximity. If a food item has been occupying a cell far away from the hawk for a longer time(more diffusion iterations) than a nearby bird has occupied its cell, the values in the food metric layer may have grown significantly larger than the values in the bird metric layer. In this situation the hawk could end up pursuing the distant food despite his preference for nearby birds simply because the food has been occupying its cell longer. Thus the metric values must be bounded.

Our first attempt to accomplish this bounding was to simply exclude the metric seeds from further diffusion. We skipped diffusing occupied cells. This quickly introduced a new problem. Consider an empty cell that is surrounded on all

sides by food. The food cells will not increase in value but the empty cell will continue to receive contributions from its occupied neighbors. Thus we bounded the occupied cells but the empty cells remained unbounded, making them frequently more attractive than the cells where the food actually existed!

To deal with the new problem we attempted to bound the diffusion rate. Supposing that the diffusion rate, ie the amount that one cell bleeds in to another, was bounded by the inverse of the number of neighbor cells ($1/n$ where n is the number of neighbor cells contributing to the diffusion) we could ensure that no diffused cell would contain a value greater than any of its neighbors. One objection to this approach is that it would be more intuitive to allow diffusion rates in the range $[0, 1)$ rather than $[0, 1/n)$. But an even bigger problem arose. It had already become obvious that in order to prevent diffusion through obstacles their cells should always have a value of zero. The new bounding of diffusion rate combined with zero valued obstacle cells introduced yet another problem.

To put it simply n varies throughout the map. Cells adjacent to obstacles have fewer contributing neighbors than cells in unobstructed areas. A corner cell with only two contributing neighbors should be subject to a diffusion coefficient bound at 0.5 rather than 0.25 (in the case of 4 contributing neighbors, our final implementation uses all 8 neighbors). The result is that cells adjacent to obstacles exhibited significantly smaller diffusion values than others. In most cases this was not a problem as it simply meant that entities would tend not to find themselves adjacent to obstacles, but situations did occur where an entity might pursue a food item placed in a corner cell and become trapped in the poorly diffused area. Furthermore excluding obstacle adjacent cells excludes some possible shortest paths.

The final limitation of basic collaborative diffusion is that it uses an agent based approach; Cells themselves are treated as independently acting agents of different types. That is to say, environment cells have some functionality, they can report or compute different values based on their type. Given that environment cells are really just representations of partitions of space it seems unrealistic to impart them with intelligence, even though they are essentially processors of diffusing metric values. It is more realistic to treat the environment as the processing agent rather than its partitions. It seems apparent that a diffusion algorithm should exist which treats environment cells for what they are: boxes with values in them. Thus it was a stretch goal to make our algorithm function using matrices to represent the environment rather than functional entities and limit the algorithm to elementwise matrix algebra as much as possible.

Another stretch goal was to capture more parameters about the diffusion. In addition to the amount which a metric bleeds into other cells it should be possible to capture the rate at which that bleed occurs in terms of the number of cells per time unit. Considering the metric of the smell of pizza in a room the bleed rate is how strong the smell of the pizza is (is it plain cheese or garlic?)

and the diffusion spread per time unit is how fast it crosses the room (is there a fan blowing?). Given these considerations the following requirements were formulated for a diffusion algorithm that would allow for increased realism and biological plausiblity.

3.3.2 Requirements for Extended Algorithm

1. Metric seed cells should never grow in value once they are seeded (ie the values of seeded cells should be bounded by their initial value)
2. Non Metric seed cells should never exceed the value of seed cells (ie the values of unoccupied cells should also be bounded by the seeded value(s))
3. Metrics should not diffuse through obstacles
4. Metrics should properly diffuse through corner cells and other cells adjacent to obstacles
5. Bleed rate and diffusion rate in cells per time unit should be independently controllable
6. The algorithm should function on matrices using elementwise matrix algebra

3.3.3 Extended Algorithm

- Sum of Neighbors Operator (4 neighbors):

$$S_4(X) : S_{i,j} = X_{i+1,j} + X_{i,j+1} + X_{i-1,j} + X_{i,j-1}$$

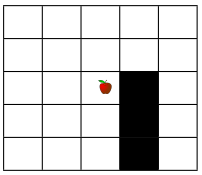
- Sum of Neighbors Operator (8 neighbors):

$$S_8(X) : S_{i,j} = X_{i+1,j}+X_{i,j+1}+X_{i-1,j}+X_{i,j-1}+X_{i-1,j-1}+X_{i+1,j+1}+X_{i+1,j-1}+X_{i-1,j+1}$$

- Input for game map of size $m \times n$:
  - $M_{m\times n} : M_{i,j} = \begin{cases} 0 & \text{if no metric seed exists at cell } (i,j) \\ > 0 & \text{otherwise} \end{cases}$
    Metric Seed: The originating metric values to be diffused
  - $O_{m\times n} : O_{i,j} = \begin{cases} 0 & \text{if an obstacle exists at cell } (i,j) \\ 1 & \text{otherwise} \end{cases}$
    Obstacle Mask: A mask to hide obstacles. Has value of 0 where obstacles exist, 1 everywhere else
  - $D \in \mathbb{R}_{m\times n}$
    Diffusion Matrix: Existing diffusion array to be further diffused. Initially all zeros
  - $d : d \in \mathbb{R}, d < 1$
    Diffusion Rate:Diffusion rate scalar, controls how much one cell bleeds into another
  - $i : i \in \mathbb{Z}, i > 0$
    Iteration count:Numer of diffusion iterations to apply. Controls rate at which metric diffuses through environment
- Return Value: $\hat{D} \in \mathbb{R}_{m\times n}$ Diffusion Matrix: Metric seed values diffused
- Precomputed Values:
  - $\bar{M}_{m\times n} : \bar{M}_{i,j} = \begin{cases} 1 & \text{if } M_{i,j} = 0 \\ 0 & \text{otherwise, may be constant or cell occupant skill} \end{cases}$
    Metric Mask: A mask to hide metric seed cells.
    Need only be computed when M changes a 0 valued cell to non-zero value.
    Has value of 0 where metric seeds exist, 1 everywhere else
  - $\hat{N}_4 \in \mathbb{R}_{m\times n}$
    Neighbor coefficient matrix for 4 (or possibly 8) neigbhors: Coefficients to compute the average of neighboring cells, has value of 0 for obstacles.
- Diffusion Algorithm ($\odot$ deonotes element-wise matrix multiplication, may use either S_4 or S_8):
  $\hat{D} \leftarrow D$
  FOR $j = 1 : i$
  $\hat{D} \leftarrow \hat{D} + d(S_4(\hat{D}) \odot \bar{M} \odot \hat{N}_4) + M$
  RETURN $\hat{D}$

3.3.4 Example

Environment:



Simple environment depicting a food item placed on a non-wrapping game map with obstacles

Metric Seed Matrix: $M =$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

A numeric matrix with 0 at empty cells and 1 at cells containing food

Metric Mask: $\bar{M} =$

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

A mask to prevent modifying values in food containing cells. Helps ensure that values of diffusion array are bounded

Obstacle Mask: $O =$

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 \end{bmatrix}$$

A mask to prevent diffusion through obstacle cells

Neighbor Count: $N = S_4(O) =$

$$\begin{bmatrix} 2 & 3 & 3 & 3 & 2 \\ 3 & 4 & 4 & 3 & 3 \\ 3 & 4 & 3 & 3 & 2 \\ 3 & 4 & 3 & 2 & 2 \\ 2 & 3 & 2 & 2 & 1 \end{bmatrix}$$

A count of the contributing neighbors for each cell

O/N Coefficient: $\hat{N}_{i,j} = \frac{O_{i,j}}{N_{i,j}} =$

$$\begin{bmatrix} 0.5 & 0.33 & 0.33 & 0.33 & 0.5 \\ 0.33 & 0.25 & 0.25 & 0.33 & 0.33 \\ 0.33 & 0.25 & 0.33 & 0 & 0.5 \\ 0.33 & 0.25 & 0.33 & 0 & 0.5 \\ 0.5 & 0.33 & 0.5 & 0 & 1 \end{bmatrix}$$

A composed mask to prevent diffusion through obstacles and properly scale the diffusion of each cell based in its number of contributing neighbors.

4 Game Representation

4.1 Entity, Metric, and Win Parameters

Agent behavior is captured in a `.data` file which is parallel to the game map. This file specifies the metric layers, metric diffusion properties, entity parameters, entitiy behavior, entity placement limits, and win condition. The `.data` file is a string describing a Python dictionary with keys and values as follows. If a data file is not found then the default `maps/map.data` file is used.

4.1.1 Metric

A dictionary keyed by Metric name with key value pairs describing the diffusion bleed rate and number of diffusion iterations to apply each round

```
'Metrics':{
  'FoodMetric':{'rate':0.9,'iters':5},
  'BirdMetric':{'rate':0.9,'iters':15},
  'HawkMetric':{'rate':0.7,'iters':20},
  'NestMetric':{'rate':0,  'iters':0}
}
```

4.1.2 Entities

A dictionary keyed by Entitiy name where each entry is a dictionary describing an entity.

```
'Entities':{
  'Bird':{ 'Eats':['Food'],
    'Weights':{'FoodMetric':1, 'HawkMetric':-1},
    'MapChar':'F',
    'Image':'images/apple.png',
    'Affects':{'FoodMetric':1},
    'Moves':False,
    'StartSkill':0
  },
```

4.1.3 InsertEntity

A List specifying names of entities which can be placed on the map by mouse click and a count of how many such entities may be placed

```
    'InsertEntity':[{'entity':'Food','label':'food','count':15},{'entity':'Bird','label':'bi
```

4.2 Map Layout

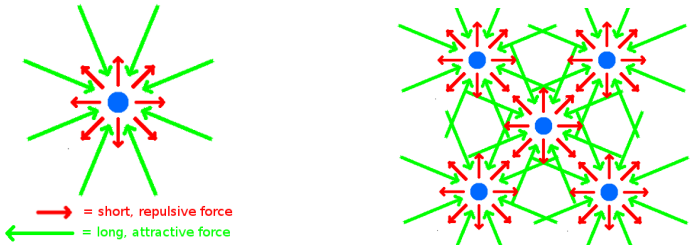
The `.map` file specifies the placement of entities and obstacles in a map. Also the size of the map

```
OOOOOOOOOO
O........O
O..NF....O
O........O
O........O
O...O....O
O.B.O....O
O...O....O
OOOOOOOOOO
```

5 Future Work

5.1 Emergent Flock Cohesion

Flocking behavior requires some form of cohesion to keep the flock in tact. Extended Collaborative Diffusion will facilitate flock cohesion through the use of two goal metrics.



Firstly birds will be attracted to other birds through a highly diffused goal metric. A well fed bird will present an attractive goal to other birds. A particular bird's attractiveness to other birds may be proportional to how well fed the attracting bird is. Secondly each bird will have a weakly diffused anti-goal metric which will repel neighboring birds in order to maintain some degree of flock separation. The repulsion anti-goal may be conceptualized as the sound of other birds squawking to maintain reasonable separation of the flock members. In order to implement this functionality our diffusion algorithm must be modified to allow agents to subtract their own influence on metric layers which they both affect and evaluate. Each entity type could maintain a matrix representing their effect on an empty neighborhood. This would allow entities to subtract their own effect from their surrounding neighborhood leaving only the effects of other entities to inform their decision. Particular entities which independently

affect a metric layer with a value proportional to their skill need only multiply their effect matrix by a scalar value (presumably their own skill)

5.2 Continuous Motion and Decisive Thrust Vectors

The current implementation allows for entities occupying and moving within a discrete tiled map. It would be a nice enhancement to allow entities to accomplish smoother continuous motion while maintaining the discrete representation of metrics. The entity's coordinates in continuous space could be easily hashed into discrete environment cells. Once continuous motion over a discrete map is accomplished it would be a modest extension use the metric neighborhood to create a summed thrust or velocity vector rather than a position vector. An entity's decision in the current implementation is effectively a vector affecting it's position; ie "I wish to go from this cell to that one." With continuous motion comes the option to use high order vectors such as velocity or acceleration(thrust). In this way an agent could be simultaneously drawn to food with a certain velocity or thrust vector magnitude and fleeing a hawk with another magnitude. The resultant effect of summing of these vectors could create rich behaviors. This enhancement opens a plethora of physical properties such as bird size or flight characteristics which which will further affect the emergent behaviors.

5.3 Genetic Algorithm to Evolve Metric Weights

It seems plausible that one could use a genetic algorithm to evolve the weights assigned to different metric layers. Given a population of entities subjected to a number of different fitness measurements it would be possible to coevolve a collection of entities to exhibit predator/prey/plant types of interactions. For example a predator entity fitness measure could be how much skill it obtains from eating other entities; A prey entity fitness measure would be how long it survives; A plant food entity fitness measure would be how often it gets eaten by other entities (emulating the biological need for some plants to spread seeds through forager droppings).

6 References

- Collaborative Diffusion Paper
  http://www.cs.colorado.edu/ ralex/papers/PDF/OOPSLA06antiobjects.pdf
- Images
  nest.png `http://images.wikia.com/frontierville/images/9/9c/Bird's_Nest-icon.png`
  flying-bird-icon.png `http://icons.iconarchive.com/icons/artdesigner/tweet-my-web/256/flying-bird-icon.png`