2014

Testing the Fences: Using Multi-Agent Reinforcement Learning to Structure the Behavior of Virtual Dinosaurs

Henry A. Meyers
Bard College

Testing the Fences:

Using Multi-Agent Reinforcement Learning to Structure the Behavior of Virtual Dinosaurs

Senior Project submitted to

The Divison of Science, Mathematics, and Computing

of Bard College

by

Henry Meyers

Annandale-on-Hudson, New York

December 2014

For my family, friends, and the *Saurian* develoment team

Table of Contents:

0. INTRODUCTION

What was it like to be a dinosaur? For many reasons, this is a phenomenological question that can never be objectively answered scientifically or logically – only coarsely approximated through educated speculation and/or good storytelling. While such speculation is best left to paleontologists and animal biologists, the question of storytelling remains far more ambiguous—what storytelling medium is best suited to answer this question, and what kinds of stories should be told? Though there is no objectively correct answer to this question, attempts at telling these stories have already been made in, literature, television, film, and the visual arts; however, there is another medium that has yet to see a serious (and widely accessible) attempt: interactive narrative.

Interactive narrative (IN) is a form of storytelling in which the audience is an active participant in determining the structure of the narrative they are perceiving. IN can be thought of as a spectrum with two poles--Strong Story and Strong Autonomy (Riedl and Stern, 2006). Strong Story ("linear[1]") interactive narratives are characterized by their reliance on an authored narrative, and by their restriction that user interactions must conform to the user's pre-determined narrative role (text adventures are a classic example of this). Strong Autonomy ("emergent") IN usually have little authored narrative, and allow the user to subjectively create their own narrative based on their open-ended interactions with an richly-simulated virtual environment. Most IN fall somewhere in-between Strong Story and Strong Autonomy. While IN is not an inherently digital medium, computers have become the dominant medium through which IN have been told since the emergence of video game arcades and personal computers in the 1960's-80's. Since then, the concepts of digital IN and games have become widely associated with one another, en though some IN are not games, and some games are not IN[2]. As such, a brief discussion of their distinctions/relationships are merited.

---

1 Branching narratives are still considered "linear" in the context of this paper, since the choices afforded by a branching narrative are still authored, and explore a different aspect of agency than emergent games.

2 One of the aspects fundamental to games are their "mechanics" – however, there is no widely accepted definition of what truly constitutes a game mechanic.

"Classical games," such as chess and checkers, are defined by their unambiguous, discrete state-spaces, rules, and outcomes – they are almost always adversarial (the player is playing against someone/something, and the game can be won, lost, or tied). Given their characteristics, classical games have been widely studied in the fields of artificial intelligence and game theory. Classical games rarely have any narrative layers, and so it is fair to generalize that the creation of classical games is strictly a design (as opposed to artistic) process.

Early video games were very reminiscent of classical games in that they were often adversarial or score-based—however, as the developers of these games began experimenting with narrative, the distinction between story/game and art/design began to blur. An early example of this can be found in the 1980's arcade game "Missile Command." The game was narratively straightforward: the player assumed the role of an anti-missile battery commander tasked with saving six cities from nuclear annihilation. Mechanically, the game was also very simple: the player controlled a crosshair that could be used to target incoming missiles to prevent them from destroying the player's cities. While the narrative and mechanics were simple on their own, "Missile Command" was significant in the way it combined them. For example: often times it would be rational (from a classical game perspective) for a player to allow several of their cities to be destroyed so that they could concentrate their resources protecting the others, but due to the game's narrative grounding, such a rational decision could be guilt-inducing and cause the player to question their status as the "hero" of the story—a designed emotional experience that artistically deviated from the usual design goals of classic games.

The interplay of game mechanics and narrative exemplified by "Missile Command" was further explored over time and eventually christened with its own term: ludonarrative. "Ludonarrative[3]" refers to a user's subjective integration of their memories of interaction with a game/IN and their memories of

3 The term "ludonarrative" was coined in the context of describing "ludonarrative dissonance" in a blog post in 2007 critiquing the popular game *BioShock*; its authored narrative was a parable in which an attempted utopia (inspired by the objectivist philosophy of Ayn Rand, which promoted selfish behavior) fails and turns into a dystopia. However, the gameplay mechanics in BioShock encouraged the user to exhibit selfish, self-preserving behavior, hence the ludonarrative "dissonance". Incidentally, there is far more debate about what truly constitutes 'ludonarrative dissonance' than what constitutes a ludonarrative.

an authored narrative presented by that game/IN, and is a very important design consideration for and IN with enough mechanics or any game with enough narrative layers[4]. A hypothetical example of a couple ideal dinosaur ludonarratives might be as follows:

You are a Triceratops in a Late Cretaceous ecosystem along the Hell Creek formation. You start alone in a forest, and notice that you're hungry, and start walking around looking for plants to eat. You eat some small herbs, but they don't satiate you much, so you try eating a nearby shrub. This time, you notice that your health has not only increased more but also is slowly increasing even more, and wonder if it has something to do with your digestive system. Then, just in time to react, you see a T. rex running towards you and turn to face it. You lunge at it in defense and miss, and the T. rex circles around you and kills you with a bite to the neck from behind.

Say you restart the game—but as a T. rex instead this time. Again, you start hungry and alone in a forest, and start looking for prey to hunt. You see some Thescelosaurs in the distance and run as fast as you can towards them, but they seem to hear you coming and scurry away into the trees before you can reach them. You try to pursue one, but your size makes it difficult to navigate through the dense forest relative to the quick, smaller dinosaur, and you quickly run out of energy. You figure that an ambush tactic might be more effective. You search for a more open area where you could lay an ambush. You find an open plain with a group of 3 Triceratops, one of which seems injured. You move as close as possible to the injured Trike from the shade of the treeline, and then charge it. You successfully bite its crest, but as you wrestle with it, one of the other two Triceratops attacks and wounds you. You try to turn and fight it, but you're crippled, and the second Triceratops lands another blow from behind you. You bleed to death rapidly[5].

Despite both of the above ludonarratives having grizzly endings[6], they exemplify desirable qualities of a dinosaur ludonarrative – offering the player a great deal of agency, but systemically

4 Of course, the distinction between these two begins to blur at a certain point.

5 At this point, the player has learned both a) that a successful T. rex needs to isolate its prey from a herd before ambushing it, and b) that a successful Triceratops might want to stay with a herd if they want to survive a T. rex attack--all derived from ludonarrative.

6 This was solely to limit the length of the example ludonarratives—a focus on violence has been an unfortunately common trope in a great deal of dinosaur art, although this trope could be thoughtfully subverted through ludonarrative

pressuring them to interact with the game in ways that are not only mechanically engaging and congruent with their authored narrative role, but also educational about paleontological theories (such as T. rex being an ambush predator, or Triceratops being a social animal and preferring a more fibrous than herbiceous diet). The only game/IN (referred to simply as "games" from here) to have attempted to create a game with ludonarratives like those described above was a travelling museum exhibit called *Be The Dinosaur*—which, though admirable in its intent, is not widely available for the majority of the audience interested in it. Now, though, another group of developers is attempting to achieve the same goals—this time, with a much broader scope in both design and distribution.

Saurian

Saurian is an independent, 3D open-world[7] dinosaur survival game that takes place in the Hell Creek ecosystem 66 million years ago, currently in development by an international team of amateur game developers and paleoartists. The user will take the role of some species of dinosaur, with only the explicit goal of surviving according to the characteristics of their chosen species (be it a small or large herbivore or carnivore, etc). I have been put in charge of designing AI for *Saurian.* Given the pivotal importance that AI has in creating successful Strong Autonomy ludonarratives, and given the many ludonarrative limitations I have experienced interacting with traditional, reactive finite state machine (FSM) based AI in other Strong Autonomy games, I will be developing another type of AI system that will ideally allow for significantly more complex, dynamic, and adaptive behavior than has ever been seen in open-world games to date. This paper will discuss the initial development and testing of this system, which will ultimately be used to bring *Saurian's* dinosaurs to life in the ludonarratives of its users.

7 A sub-genre of hybrid Strong Autonomy / Strong Story games which places much greater emphasis on the Strong Autonomy approach than Strong Story.

1. BACKGROUND

I. The Unity Engine

The majority of modern games are developed using game engines—a type of software framework that facilitates the use of many crucial tools for game development (including rendering graphics, physics, collision detection, sound, animation, scripting, A.I., etc.) Although there are many game engines, a game developer's choice of engine is usually constrained by copyright, licensing costs and the engine's features relative to the developer's resources and goals. Some game engines, such as Unreal, CryEngine, and Unity are free for non-commercial use. Due to prior experience with the software and the goals of the Saurian team, I developed this project in Unity, version 4.3.

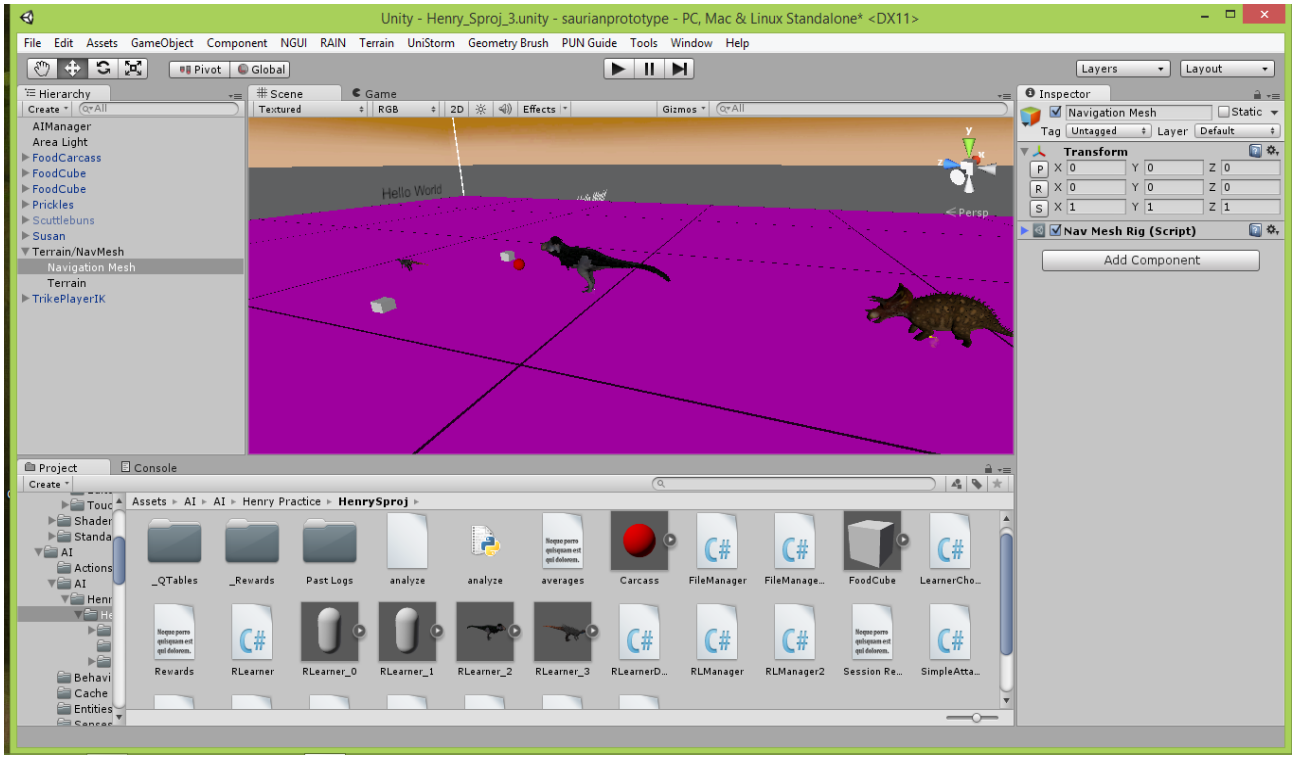


Illustration 1 : Unity 4.3 Editor
Windows: scene object hierarchies (left), inspector (right), scene, game view (center), project directory , debug console (bottom)

Two of Unity's main distinguishing characteristics are its multiple scripting language

options and its flexible, modular approach to sharing and using game assets[8]. Since Unity's utilizes Mono (the open-source implementation of Microsoft's .NET framework), it supports use of several compatible languages: C#, UnityScript (a proprietary language similar to JavaScript) and Boo (a language influenced by C# and Python). Although Unity comes with many generic assets for developers to use, it also has an online 'asset store' that allows developers to share assets with one another (commercially and non-commercially). This allows for a great degree of customization in Unity projects beyond the default features offered by the engine (the importance of which will be discussed below).

One fundamental type of object in Unity is the GameObject—which can be empty, or which can have attached components and/or other 'child' GameObjects. The most fundamental type of script (for all languages) in Unity is the MonoBehavior. A MonoBehavior is an inheritable base class that must always be attached to some GameObject as a component, and is used to execute actions as a scene is being rendered. The two default, void functions that must usually be specified in a MonoBehavior are the 'Start' and 'Update' functions. The 'Start' function is used to initialize a MonoBehavior whenever its parent GameObject is instantiated in a scene. During runtime, the 'Update' function is called by Unity engine at every frame[9], and is used to control all time-sensitive behavior of a GameObject. MonoBehaviors also have several optional functions such as 'OnDestroy' or 'OnMouseDown' that allow a programmer to handle specific events. Like any class, custom member functions can also be written in a MonoBehavior and called at the programmer's discretion.

II. RAIN

RivalTheory's RAIN is a free library for Unity intended to streamline development of many common elements of game A.I., such as behavioral evaluation/control, movement, perception, actions,

8 The term 'game assets' simply refers to specific resources used by developers, such as shaders, models, animations, libraries, etc.

9 A 'frame' refers to a variable time-step, the length of which is dictated by a) the time it takes to render graphics / physics and b) the time it takes to execute scripts. Like film and animation, motion in a game engine is thought of in terms of 'frames per second' (FPS) – most modern games are expected to run at a more-or-less steady rate of 30-60 FPS.

pathfinding, and animation. When a programmer has selected an appropriate GameObject in the Unity scene view, they may attach a RAIN 'AI Rig' as a child GameObject. When selected , all of its parameters may be viewed and edited with Unity's inspector. Here, relevant information pertaining the AI may be managed with its respective module: “Working Memory,” “Mind,” “Motor,” “Animation,” “Navigation,” “Perception,” or “Extensions”. In addition to the 'AI Rig', if any GameObject is intended to be perceivable by a RAIN agent, it must also have a RAIN 'Entity' attached as a child object. In the inspector, an Entity can be given 'Aspects' (either 'visual', 'audio', or 'custom'), which can then be perceived by the respective sensors of a RAIN agent.

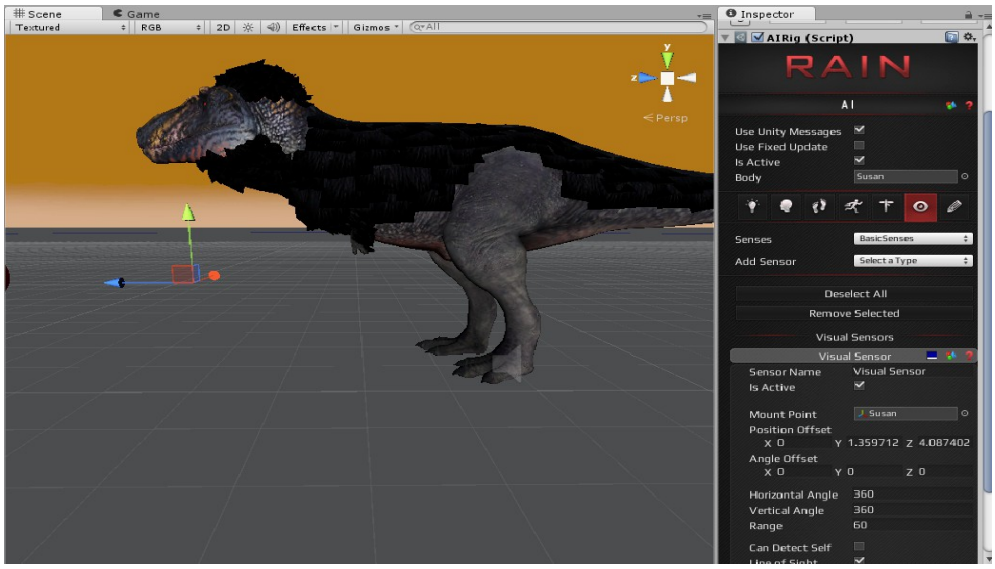


Illustration 2: RAIN AI Rig in Unity Inspector

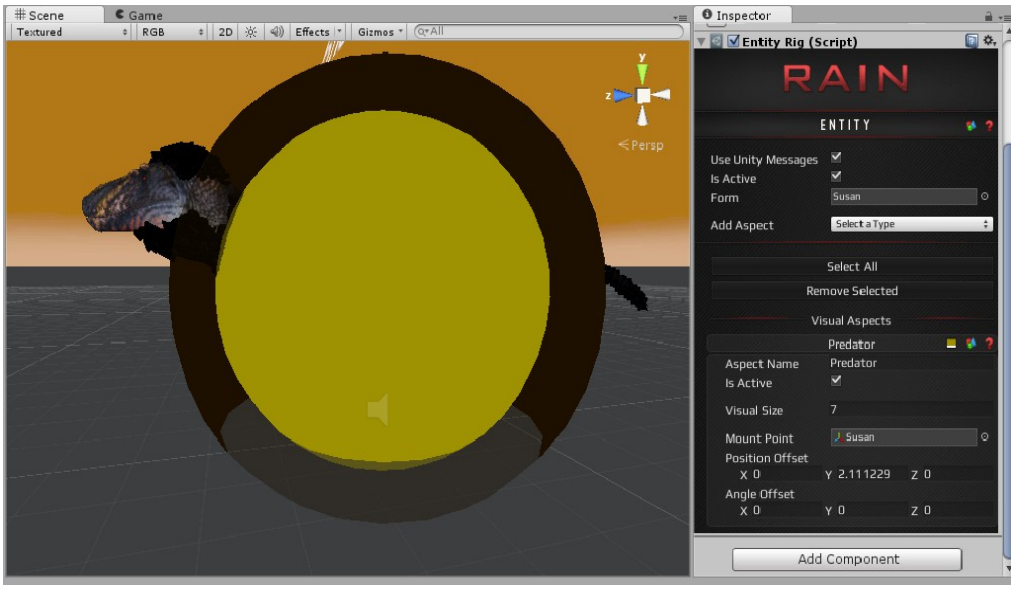


Illustration 3: RAIN Entity in Unity Inspector

Rather than explaining these components individually, the way that these modules act and interact can be more easily appreciated if another central concept in RAIN is examined in detail first: the behavior tree. In RAIN, an AI's control architecture is represented as a behavior tree—a tree in which non-leaves are 'decision nodes' that can succeed or fail based on their constraints, and leaves are 'action nodes'. This tree is traversed from the root along succeeding decision nodes until their action nodes have been executed. Behavior trees are created and modified in RAIN's Behavior Tree Editor. Since there are many different types of decision and action nodes, and an overview of them is not particularly helpful here; however, a simple, illustrated example of a behavior tree traversal will help explain the basics of using RAIN AI and should clarify how its elements are integrated.

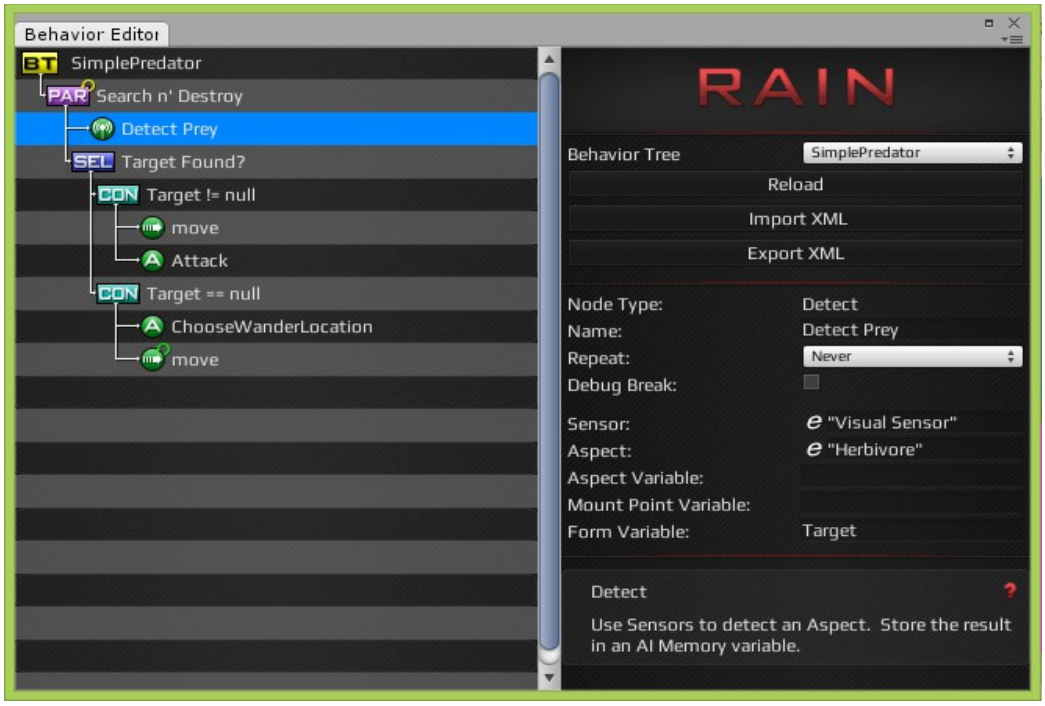


Illustration 4: RAIN Behavior Tree

The above example shows the behavior tree of a simple antagonistic AI as seen in the Behavior Tree Editor. The root node ("BT") has one child: a parallel decision node ("PAR"). A parallel decision node traverses its child nodes simultaneously (as opposed to iteratively[10]). The yellow ring at the upper-

10 The "parallel" algorithm is actually iterative, although this is a trivial detail given the parallel node's behavior relative to other decision nodes.

right of the node indicates that this decision node will repeat forever—effectively turning it into the root node for all following tree traversals[11].

At this point, both of this new root's child nodes are inspected: a 'Detect' action node (“Detect Prey”) and a selector decision node (“SEL”). As a leaf node, the detect node performs an action; in this case, it specifies a sensor with which the agent should perceive some aspect of its environment, and then stores the result of the search in the agents 'working memory' (relevant parameters are set in the Behavior Tree Editor, pictured). In this case, the agent checks its visual sensor for an aspect named “Herbivore,” storing the result as a GameObject called “Target” in 'working memory'. The value of “Target” will be the parent GameObject of said aspect's parent entity if that aspect is detected, and null otherwise.

The selector decision node traverses any of its child nodes until any succeeds or all return failure. In this tree, the selector will never see all its children fail because they are two constraint decision nodes (“CON”)--conditional nodes that execute their children sequentially if some given Boolean expression evaluates true and return failure if false—that cover all possible conditions: either there is a GameObject named “Target” in working memory, or there isn't. In the former case, two action nodes are executed: a move node causes the AI to use its motor and pathfinding modules and move to the target, then a custom action node (“Attack”) executes a C# script that damages (decreases a 'health' variable in the working memory of the RAIN AI attached to the target GameObject) or destroys the target GameObject. Likewise, in the latter case, another two nodes are executed: a custom action node calculates a semi-random Vector3 (x,y,z coordinate) and stores it in working memory, and a move node directs the agent towards it.

Though at a glance RAIN seems to have been designed with FSM architectures in minds, its flexibility is great enough to allow for exploration into other avenues of AI control architectures--in this case: temporal difference learning.

11 The absence of a colored ring indicates that the node cannot repeat until it is reached in the next traversal.

2. METHODS

I. Temporal Difference Learning

Inspired by biology and psychology, unsupervised reinforcement learning (RL) is a type of machine learning in which an agent takes random actions in an environment and attempts to maximize values computed by a pre-defined reward function that either 'rewards' or 'punishes' an agent based on its state and actions. One type of RL is temporal difference learning (TDL), in which an agent gradually learns through trial and error to associate its actions with either reward or punishment; structuring its behavior by continually choosing actions that are expected to maximize its reward according to stored values derived from its reward function and mapped to state-action pairs as a Markov Decision Process (MDP) policy.

There are several types of TDL, including Q-Learning and SARSA (Sutton and Barto, 1998). Q-Learning is an off-policy algorithm, meaning that it learns the values of its policy independently of the agents actions and can update its policy based on an estimated payoff, developing flexible behavioral sequences by separating its exploration function from its actions. SARSA, on the other hand, is an on-policy algorithm, meaning that it learns the values of its policy directly from experience; this allows an agent to learn quickly and reliably, but at the cost of flexibility and a more complex exploration process. Given Unity's real-time environment and RAIN's control structure, SARSA proved to be the more useful of these two for this task (this will be explained in the following section). SARSA, an acronym for “state-action-reward-state-action,” is named after the structure of its algorithm:

$$Q(s, a) = Q(s, a) + \alpha[R' + \gamma Q(s', a') - Q(s, a)]$$

Where 'Q(s, a)' refers to the MDP policy value of the agent's previous[12] state-action pair and 'Q(s', a')' refers to the value of the agent's current state and action. R' is the calculated reward value of the agent's current state. Alpha (α) is the rate at which total reward and policy information influences the policy values of previous states. Gamma (γ) is the 'discount rate' at which rewards of encountered states influence the policy values of previous states. As indicated by the name, the reward is only delivered after an agent in some state has taken an action that resulted in a reward. An optimal action is then chosen and the next state-action pair 'Q(s', a')' is described—at which point, the update assignment equation is computed and the optimal action is taken.

II. Implementation and Early Development Iterations

I initially attempted to implement Q-Learning as a single MonoBehavior in with a RAIN agent (with a separate C# class file I/O) in Unity, using a C# dictionary as the policy table. However, it soon became apparent that this process was too complex to allow for easily readable code in a single MonoBehavior (due partially to the specific tasks needed, and also to a bug in Unity's MonoDevelop IDE's code folding[13]). As a result, learning was then split between two MonoBehaviors—a learning "manager" and a second inheritable script that could be attached to learning agents. (A separate class was implemented for the representation of states as well).

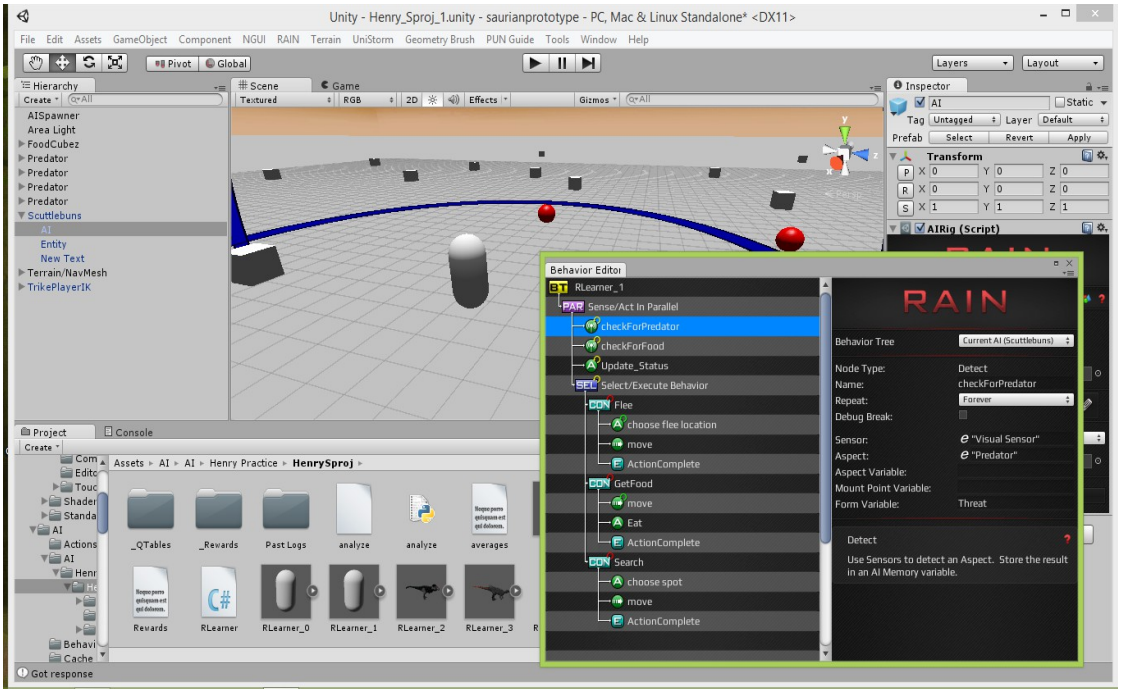
The learning "manager" is attached to an empty GameObject (same as the spawner), and serves as a hub for agent policy-information, in addition to handling file I/O for agent-specific policies. The "learner" MonoBehavior, on the other hand, uses RAIN in order to gather sensory information (for state information) and take actions for the agent it is attached to, as well as keeping track of states. Learner agents choose their actions based on the policies stored by a single manager in their scene,

12 The (s, a) tuple must be initialized (at least arbitrarily) before the first update
13 This is a feature that allows a programmer to selectively display or hide certain sections of code.

which they individually modify as they go about the world. A useful consequence of the manager/learner split was that multiple coexisting agents of the same “type” (specified as a string in their learner) all read and write to the same Q-Table (policy), allowing them to instantly transfer learned knowledge to one another and dramatically increasing policy learning rates.

At first, I attached a RAIN AI and Entity to a simple 3D capsule representing a prey species (Scuttlebuns), fitted it with a crude Q-Learning MonoBehavior, created a simple RAIN behavior tree (where constraint nodes selected behavior based on a string item in working memory), and used it to synchronize RAIN AI actions with Q-Learning actions (which were “search,” “get food,” and “flee”). States were concatenated strings of Booleans and an integer representing agent health and distance from food and predators. Scuttlebuns was trained in an environment with food (simple cubes with RAIN entities attaches) and three FSM controlled 'predators'-- its reward based on maximizing its health. This approach was unsuccessful for several reasons, including system parameter design and a few unresolved bugs, but most importantly because of Q-Learning's off-policy nature. In a real-time environment like Unity engine, it is important for an agent to learn how its actions relate to what actually happens in the environment (as opposed to what it would optimally like to happen)—this led me to restructure my code and implement SARSA instead.

Illustration 5: Early simulation shown in scene view, with RAIN BTree in view



At first, I greatly simplified my system parameters and state/reward representations for Scuttlebuns in order to verify that SARSA was working correctly—I reduced Scuttlebuns' state knowledge to food location information represented as bitstrings, changed its reward value to a simple 1 or 0 depending on its state, and removed the "flee" behavior along with all predators in the environment. I also added a 'spawner' GameObject to the scene to handle terminal events—at first, this was simply when Scuttlebuns ate food, which would result in its reward, destruction, and re-instantiation ('hungry' once again). I also separated the "manager" and "learner" classes from one another at this time. After I finally observed Scuttlebuns learning optimally[14], I moved on to attempt multi-agent learning.

At this point I introduced Susan, a predator-type A.I. into the mix. It was setup exactly like Scuttlebuns, except that its RAIN Entity's visual aspect was named "Predator" rather than "Herbivore," used a (similar, but separate) behavior tree that classified Scuttlebuns as its food, and it was embodied as a Tyrannosaurus rex. In addition, I added "flee" back into Scuttlebuns' behavioral repertoire, added a

14 This was solely from in-editor observation, see results for quantitative proof.

punishment (reward of -1) upon death, its new terminal state, and allowed it to perceive Susan (as well as food) via attached RAIN entities using RAIN's visual sensors. The results of this initial attempt were somewhat successful, but also flawed for much more interesting reasons than they were with Scuttlebuns' first Q-Learning trials. On loading a scene with empty Q-Tables for both Scuttlebuns and Susan, Susan would learn very quickly to hunt Scuttlebuns while Scuttlebuns was still learning how to get food. However, given enough time, Susan would become a completely inept hunter, while Scuttlebuns would become excellent at evading Susan and even learn to ignore it after a while. The reasons for this problem stemmed from two components of the system: the spawner and the complexity of both agents' state representations.

With only two behaviors and one perceptual signal, Susan had a much smaller state-space to explore, hence the success at first, while Scuttlebuns had a larger state-space due to its three behaviors and two perceptual signals, hence its initial (seeming) failure and ultimate success relative to Susan. The reason why Susan began to fail was that, while in pursuit of Scuttlebuns, if Scuttlebuns reached food before Susan reached it, the spawner would de-spawn (destroy) Scuttlebuns, causing an undesirable state change for Susan (from “food found / near” to “no food found”), while simultaneously increasing policy values for arbitrary actions in the “no food found” state)--if this happened enough times, Susan's policy values would be scrambled beyond salvation, and Scuttlebuns would learn that it could ignore Susan. The current environment, terminal conditions, and knowledge representations needed for a predator/prey ecosystem were too simple for these agents; they needed to know more about the world, themselves, and each other if they were going to fully utilize SARSA.

At the time of initial data collection, all agents have a “current health” value with a “max-health” ceiling value. The reward received by an agent is simply the agent's current health value, with a punishment of -1 * maximum health if the agent's health drops below 0 (at which point the agent is destroyed and re-spawned with half of its maximum health). If an agent dies, it spawns a 'carcass'

object that can be consumed only by predators. The “success” terminal condition was removed, so death is the only terminal state for all agents. On each frame, the agent's current health is decreased by a constant percentage of their maximum health, current movement speed, and a very small decimal (.000001 * maximum health; so that agents with more health will lose fractions of their health at similar intervals as agents with lower maximum health). In addition, each agent also has a 'Bite Size' value – this determines both how much health an agent gains from eating their designated food item, and how much damage can be dealt to an opponent upon chasing them down using a `Fight` behavior. To simplify Q-Table complexity, both agents could `Search` and `Get Food`, additionally with `Fight` for Susan (referred to as the “predator” from hereon) and `Flee` for Scuttlebuns (referred to as the “prey” from hereon).

The predator and the prey respectively have bite sizes of 25*[15] and 10*, and maximum health values of 200 and 100, respectively. Movement speeds for behaviors were shared by both agents, with the speeds for the behaviors `Flee, Fight, Search,` and `Get Food` being respectively 30*, 25, 15, and 10. Each agent had the same learning parameters: Alpha (the learning rate) = .1, Gamma (the discount rate) = .9, and a ¼* chance to choose a random action rather than an optimal action on either a state change or action completion. Policy updates are made only on state changes. In all trials, the number of prey food items on the map at any given time was always the same as the number of prey present.

Fixed Parameters:

Alpha	0.1
Gamma	0.9
`Search` Speed	15
`Get Food` Speed	10
Predator `Fight` Speed	25
Predator Visual Range	60

15 * indicates that certain parameters mentioned here were later changed, and will be discussed the following sections

Prey Visual Range	50
Predator Max-Health	200
Prey Max-Health	100

Parameters Varied After Validation :

Prey Bite Size	10
Prey `Flee` Speed	30
Random Behavior Chance	0.25
State Space Complexity	4^2
NavMesh Size	200

3. RESULTS

I. Validation:

Before any multi-agent trials could be conducted, a validation trial was required to verify that learning was occurring properly in a single-agent context. There are three common models used for measuring the performance of RL agents: the finite horizon model, the infinite horizon model, and the average model; of these, the average model (similar to the infinite horizon model) was most useful for this process because a) the appropriate length for training sessions is unknown in this domain (required by the finite horizon model), and b) agent discount factors do not change over time (required by the infinite horizon model). The average model measures the average reward of all agents over any course of *h* time-steps, R_{avg}.

$$\lim_{h\to\infty} \frac{1}{h} \sum_{t=0}^{h} R_{(t)}$$

Using this formulation, reward is averaged over the duration of an entire session. In theory, given enough time, all agents should converge towards Nash equilibrium—a condition in which all

moves deemed 'optimal' by the agent's policy are better than any other potential moves—represented visually by convergence with a horizontal asymptote at the maximum reward (Bowling and Veloso, 2004). Given an agent in an environment with the simple task of finding food in an environment and plotting the average rewards collected (according to the above formula) produces the following graph (Figure 1). The y-axis indicates reward (current health), and the x-axis indicates the number of samples that have been taken at a given point (with roughly 60 frames per second, and sample time length specified under the x-axis)[16].

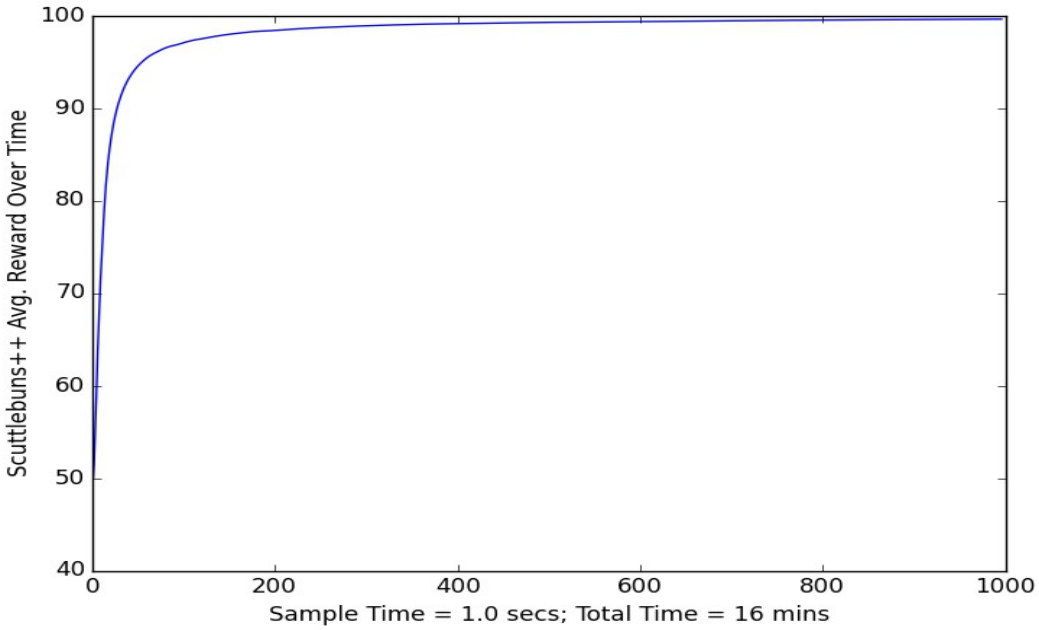


Figure 1: Single prey validation session using R_{avg} model

This approach makes interpreting the events in an agent's session significantly more difficult than it could be since it averages over the time of the entire episode, potentially smoothing over the impact of events that occur late in training. However, if only the running averages themselves are plotted, a training session with an identical agent tells a much more detailed story of the events that occurred during that session.

16 Although frames would technically seem to be the most computationally relevant time-step to analyze, this was not done because, due to the time in seconds required to complete an action and the impact of executing certain actions on overall framerate, frames could not easily be correlated with either state changes or actions.

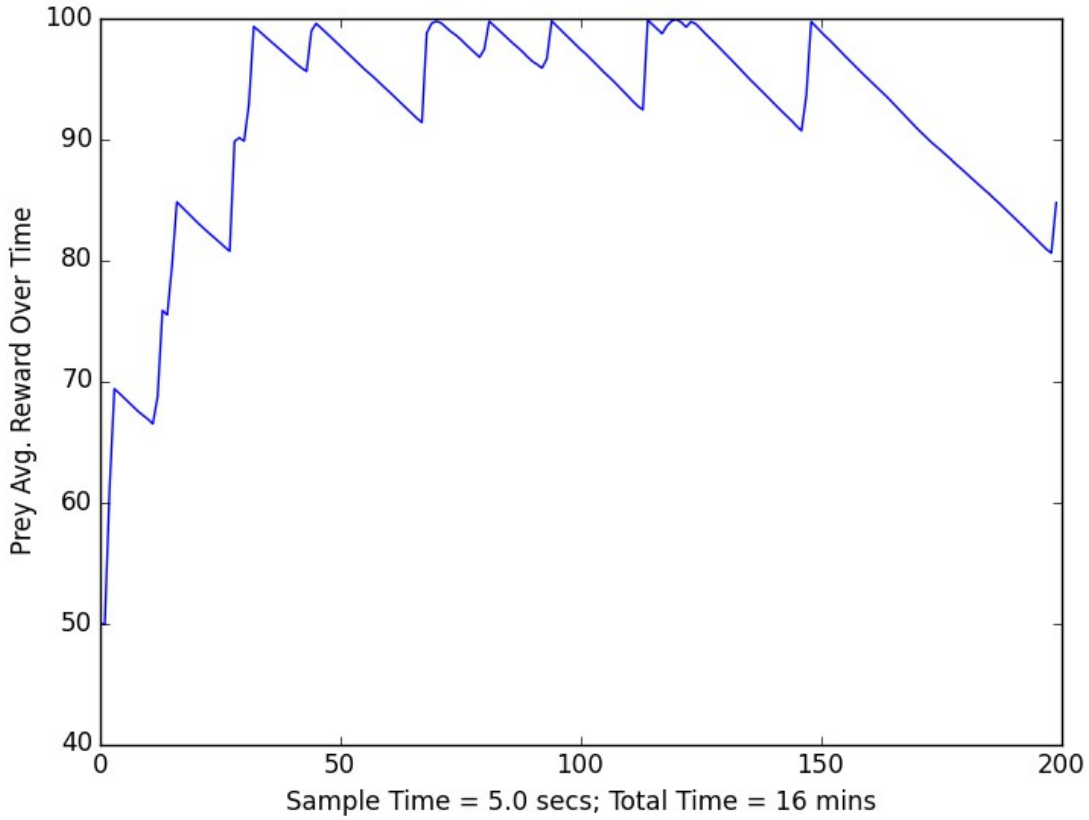


Figure 2: Single prey running average

Although this agent generally seems to be converging towards equilibrium, the average reward is significantly noisier—and features a significant drop at the end. The reason for this is simple: finding sparse food takes time, and the agent's health decreases over time. One can see this entire agent's "life history" in this graph: peaks represent finding food, and declines represent time spent wandering. However, the convergence becomes more evident when the averages of 5 of the above agents' (all following the same policy) running averages are recorded simultaneously, with available food increased proportionally.

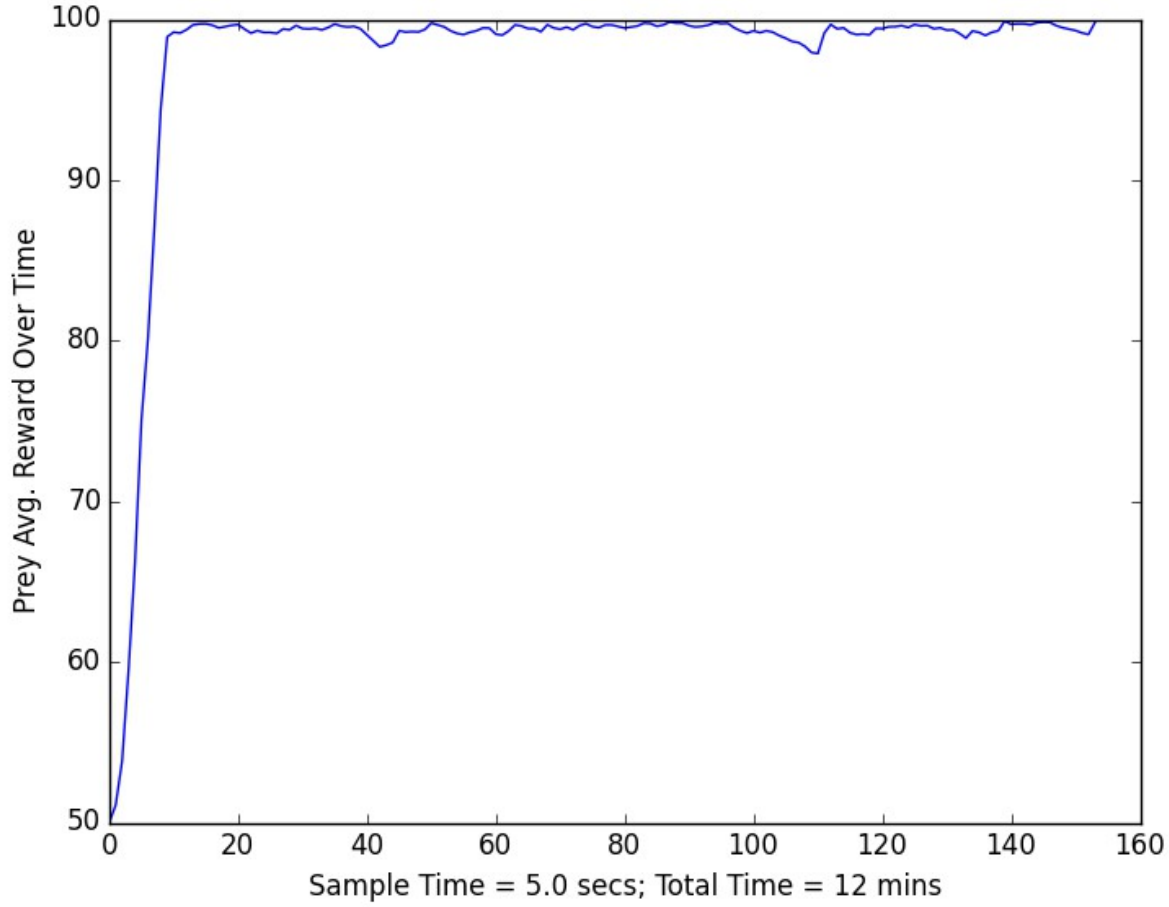


Figure 3: Five prey averaged running averages

Not only is convergence significantly easier to identify, but that it occurs significantly faster than it does with a single agent. Although individual histories can no longer be unambiguously interpreted, generalizations can still easily be made from this curve's behavior near the asymptote, such as the average time that agents were spending searching or getting food, indicated by peaks and (less-pronounced) valleys once again.

Further evidence that equilibrium was almost reached in Figure 2 and successfully reached in the Figure 3 is provided by their respective Q-Tables. For the purposes of clarity, Q-Table entries and optimal action selections will be expressed in the following format:

$$\{S_1{:}S_2{:}....{:}S_n :: A_1\} => \{S'_1{:}S'_2{:}....{:}S'_n{::}A_2\} => ... => \{S''_1{:}S''_2{:}....{:}S''_n{::}A_3\}$$

where $S_1{:}S_2{:}....{:}S_n$ refers to an agents initial state tuple, A_1 represents the optimal action chosen from that state, $\{S_1{:}S_2{:}....{:}S_n{::}A_1\}$ represents the state-action pair, '=>' represents a state-change from $S_1{:}S_2{:}....{:}S_n$ to $S'_1{:}S'_2{:}....{:}S'_n$ and A_2 represents an optimal action taken given the new state. S_1 - S_n are numbers between 0 and 3. For all the following trials, S_1 represents the agent's health : 0 is almost empty, and 3 is full; S_2 represents distance from food, with 0 meaning 'not found', 1 meaning detected but far away, and 3 meaning 'very close'. For the purposes of discussing state permutations, the symbol '*' will be used to specify irrelevant states, and the variable 'X' used to specify all state tuples that have 'X' value for some given state. For example, {X:*} refers to all state tuples where S_1 has value X, regardless of the value of S_2.

Integer	*Health Enum Meaning (S_1)*	*Distance Enum Meaning (S_2)*
0	Very Low	Not Found
1	Low	Far Away
2	Medium	Medium Distance
3	High	Very Close

For all of the above agents, all states of a policy at equilibrium can be expressed as follows:

$$\{*{:}0{::}Search\} => \{*{:}1{::}GetFood\} => \{*{:}2{::}GetFood\} => \{*{:}3{::}GetFood\} => \{*{:}0{::}Search\}$$

[Example:]

$$\{1{:}0{::}Search\} => \{1{:}1{::}GetFood\} => \{1{:}2{::}GetFood\} => \{1{:}3{::}GetFood\} => \{2{:}0{::}Search\}$$

This expresses a sequence of events where an agent searches until it finds food, then approaches the food, eats it, and begins searching again. This sequence is expected to be in the Q-Tables of both

single-agent and five-agent trials. Although present in both, in the five-agent trial there was a conspicuous deviation from this ideal policy, which merits comment. The five-agent trial contains the policy entry {3:0::Flee} which occurred simply because I forgot to omit the `Flee` behavior when collecting data. The explanation for its calculated optimality is important. In the five-agent session, agents had no knowledge of one another, and sometimes one agent would eat a food item before another could reach it. In these situations, the value of the agent's current policy would be devalued—even if their action was (correctly) `GetFood`. When these situations occur, sometimes non-optimal, randomly-chosen actions will be reinforced. In addition, if food was spawned near an agent as they were randomly attempting the `Flee` action, this would bring them to {3:X::GetFood} with $X > 0$, and would be rewarded as such[17]. Both of these occurrences were unavoidable due to constraints on agent state complexity and systemic properties, and so the resulting policy entries were inevitable. For the purposes of this paper, these types of entries will be referred to as "invalid entries."

II. Experiment 1 – Population Ratios, State Complexity, and Predator Empowerment

The first set of multi-agent simulations introduced the predator into the environment with the prey, and had three additional variables: predator bite size, session population ratios, and trial state complexity. Hour-long sessions with (arbitrary) predator:prey ratios of 1:1, 1:2, 1:3, 2:2, 2:3, 3:2, and 3:3 were run in two trials. Agents in trial 1 with a state complexity of 4^3 (S_1 and S_2 remain, with S_3 representing distance from a prey/predator agent for predator and prey respectively), while later simulations had 4^5 (representing the prior three states, plus S_4 as predator/prey's current health and S_5 as the distance enumeration from other nearest prey/predator for prey and predator respectively). In trial 1, predators had a bite size of 25, while in trial 2 they had a their bite size doubled. The reason for the

17 This conclusion was reached through observation—there is no way that this information could be indicated either in a Q-Table or in a reward graph.

latter variable can be easily understood through the contrast between the graphs of 1:1 population ratio for trials 1 and 2 (shown in figures 4 and 5).

Experiment 1, Trial 1 Parameters

Predator Bite Size	25
State Space Complexity	4^3

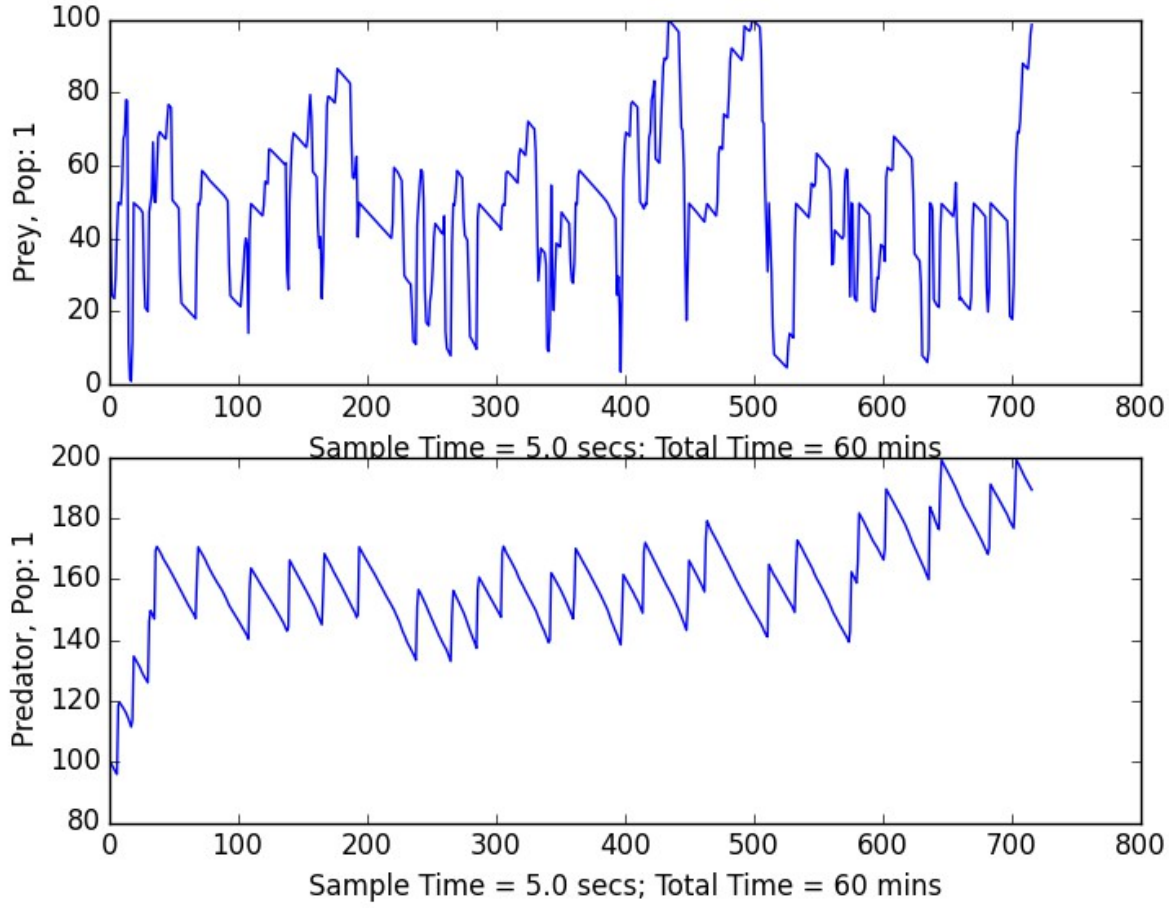


Figure 4: 1:1 Multi-Agent Session, Experiment 1, Trial 1

Experiment 1, Trial 2 Parameters

Predator Bite Size	50
State Space Complexity	4^5

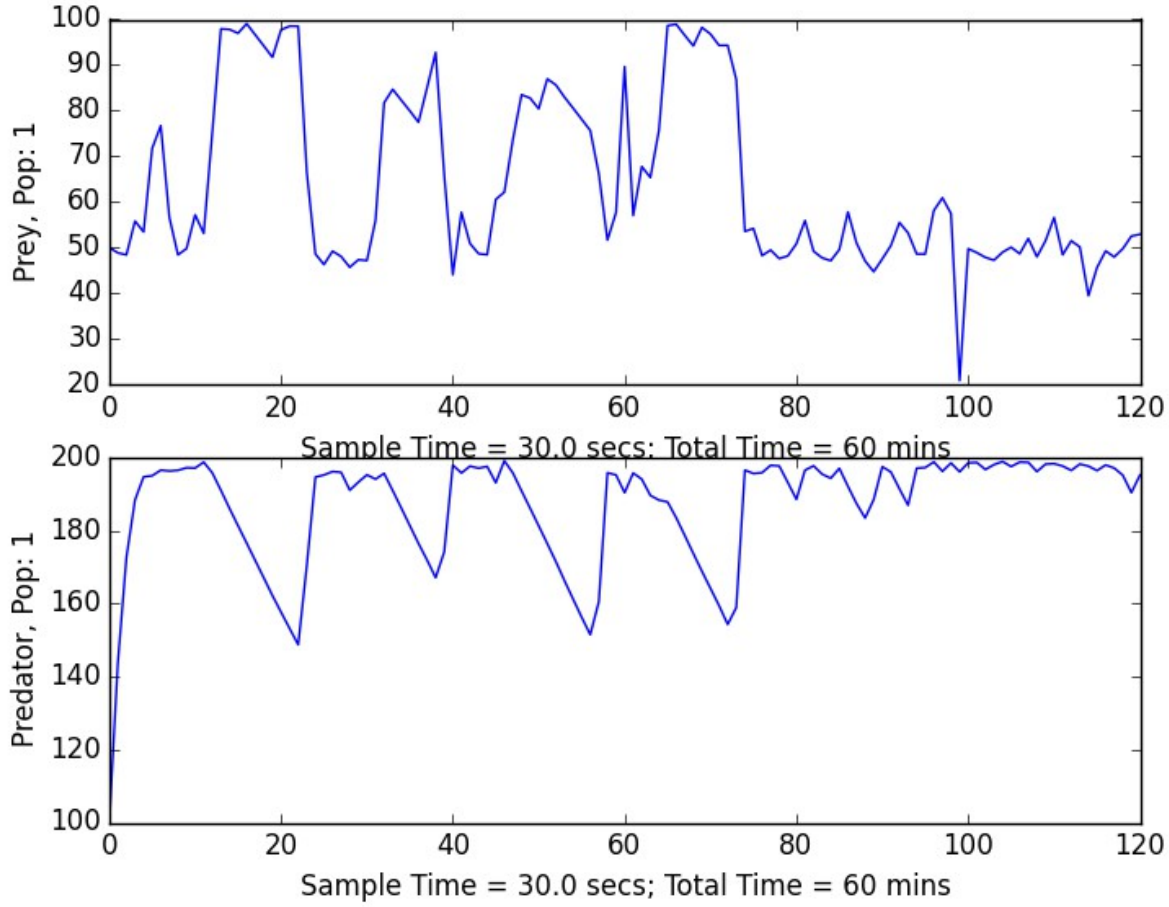


Figure 5: 1:1 Multi-Agent Session, Experiment 1, Trial 2

Even with an exponentially larger state-space than agents in trial 1, the trial 2 predator converged while the trial 1 predator did not. The reason for this was that the trial 1 predator could only kill and eat a prey after it had attacked it a maximum of 4 times and received 1/8th its total health upon eating, while the trial 2 predator had to attack twice at maximum and received ¼ its total health —this drastically increased the complexity of the learning to hunt and choosing between wandering and fighting for predator 2, and is evident from the relative jaggedness of the predator curve in Figure 4 relative to Figure 5. The 'injuries' (rapid decreases in reward) sustained by the prey in trial 1 are visibly aligned with jumps in predator health, although they are not clearly aligned as they are in the trial 2 graph—this is because they were being injured frequently but only rarely died. Counter-intuitively, the

apparent asymptote at roughly half-health for the prey in Figure 5 represents frequent prey deaths. The running average counts by time-steps, and the 'death' state itself only lasts a moment before the agent is de-spawned, and so their punishment is largely overshadowed by their health value of 50 when they re-spawn. As a result, dips below 50 in reward represent a chase in which a prey had been injured but managed to stay alive for at least a few moments afterwards. For this reason, although Figure 5's predator converges, the prey clearly does not.

The most illustrative of results in the remaining trials with multiple agents of the same type[18] were the 3:2 sessions.

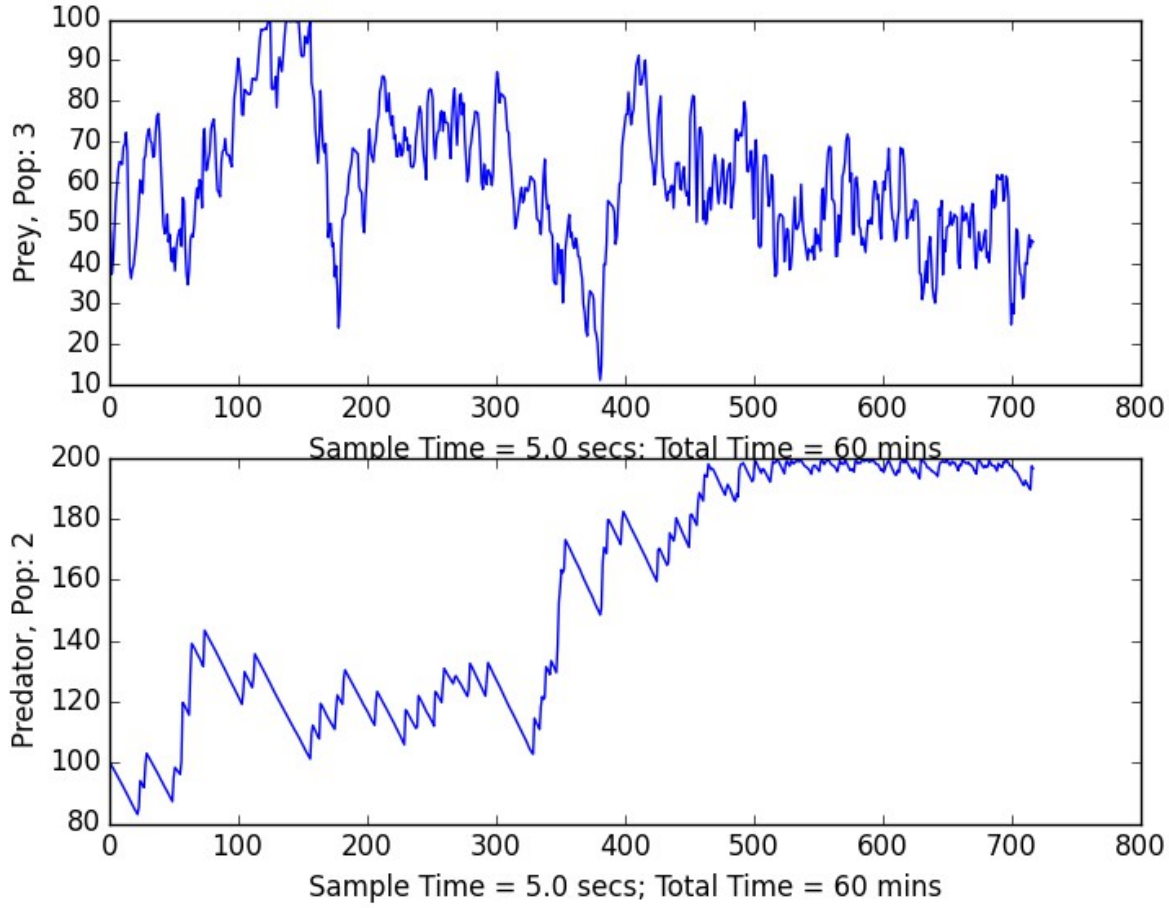


Figure 6: 3:2 Multi-Agent Session, Experiment 1, Trial 1

18 See appendix for others

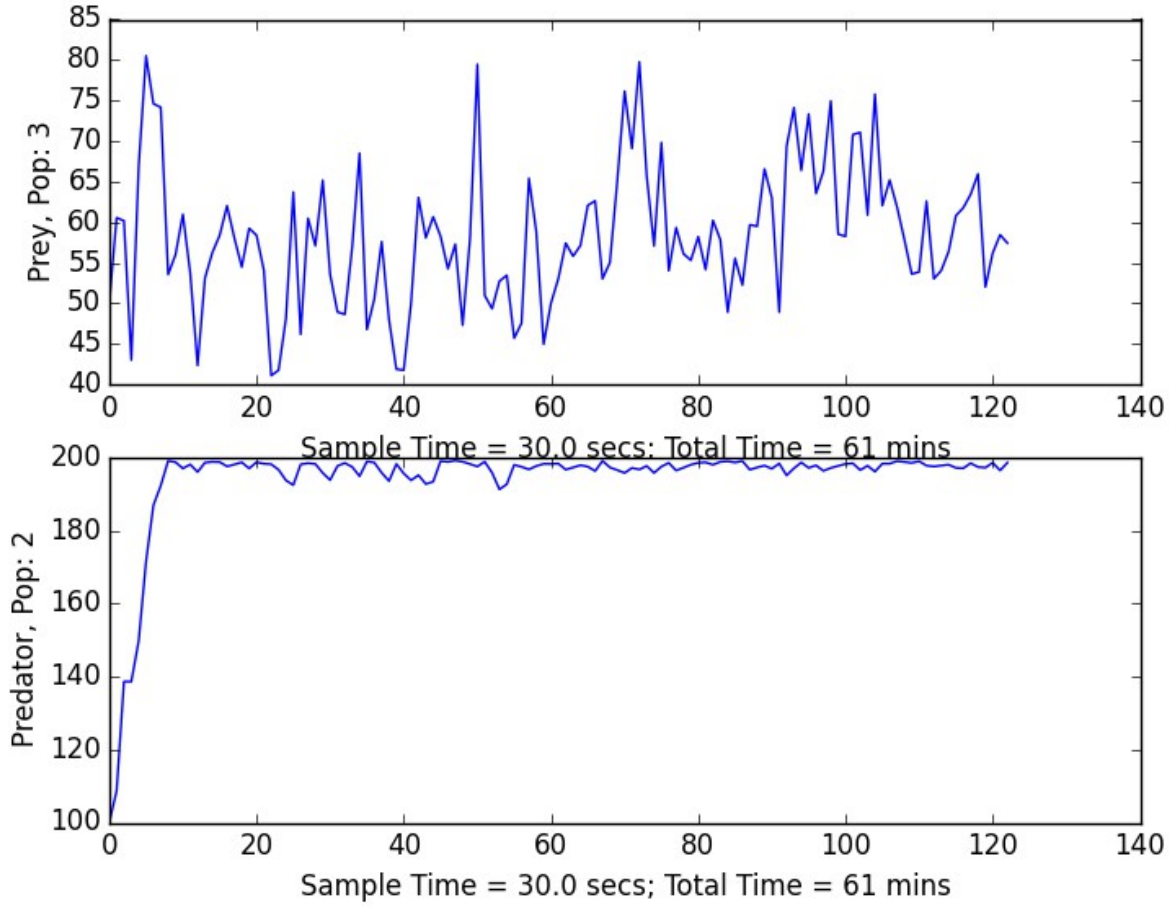


Figure 7: 3:2 Multi-Agent Session, Experiment 1, Trial 2

Once again, clear, fast convergence is seen in predator 2, while predator 1 only begins to converge around halfway through the session. In this case, although both predators and prey learn faster due to their increased numbers, it appears that prey 1 was successfully learning to flee at first (evident correspondence of prey 1's peaks with predator valleys, as well as average prey health dipping below 50 for significant periods of time), but then begins wavering around 55 as the predator reaches equilibrium—indicating that the predator was successfully chasing down prey very frequently.

Why, after reaching equilibrium, did predators seem to dominate prey so disproportionately? Based on both data and observation, it is clear that the prey's fleeing behavior isn't very effective

against the predator's fighting behavior. Additionally, are multiple predators in trial 1 significantly less effective than those of trial 2 despite their smaller state-exploration requirements? The third set of runs examined these questions by enhancing prey survival behaviors and environmental parameters accomodating them.

III. Experiment 2 – Empowered Prey and Continuing Q-Tables:

At this point, I began Experiment 2 in an attempt to even the odds for the prey, reduce the number of invalid entries, and also to observe the behavior of agents using the same Q-Table for several training sessions of different length. For this trial, populations were fixed at a 10:10 ratio. Predator bite size was changed to 40, while prey bite size was changed to 20 (allowing for them to extract more health from food). State complexity was made 4^4, with the first 3 states in the state tuple being the same as those in trial 1, and S_4 being a distance enumeration from the nearest agent of the same type. Lastly, the chance to choose a random behavior was increased from 1/4 to 1/10, the size of the navigation mesh was tripled, and spawn radius of both agents and food items was greatly increased in proportion with the NavMesh being tripled in size.

Experiment 2 Parameters

Predator Bite Size	40
Prey Bite Size	20
State Space Complexity	4^4
Random Behavior Chance	0.1
NavMesh Size	600
Prey `Flee` Speed	35

State Variables	S_1	S_2	S_3	S_4
Meaning	Health Enum (0-3)	Food Distance Enum (0-3)	Foe Distance Enum (0-3)	Friend Distance Enum (0-3)

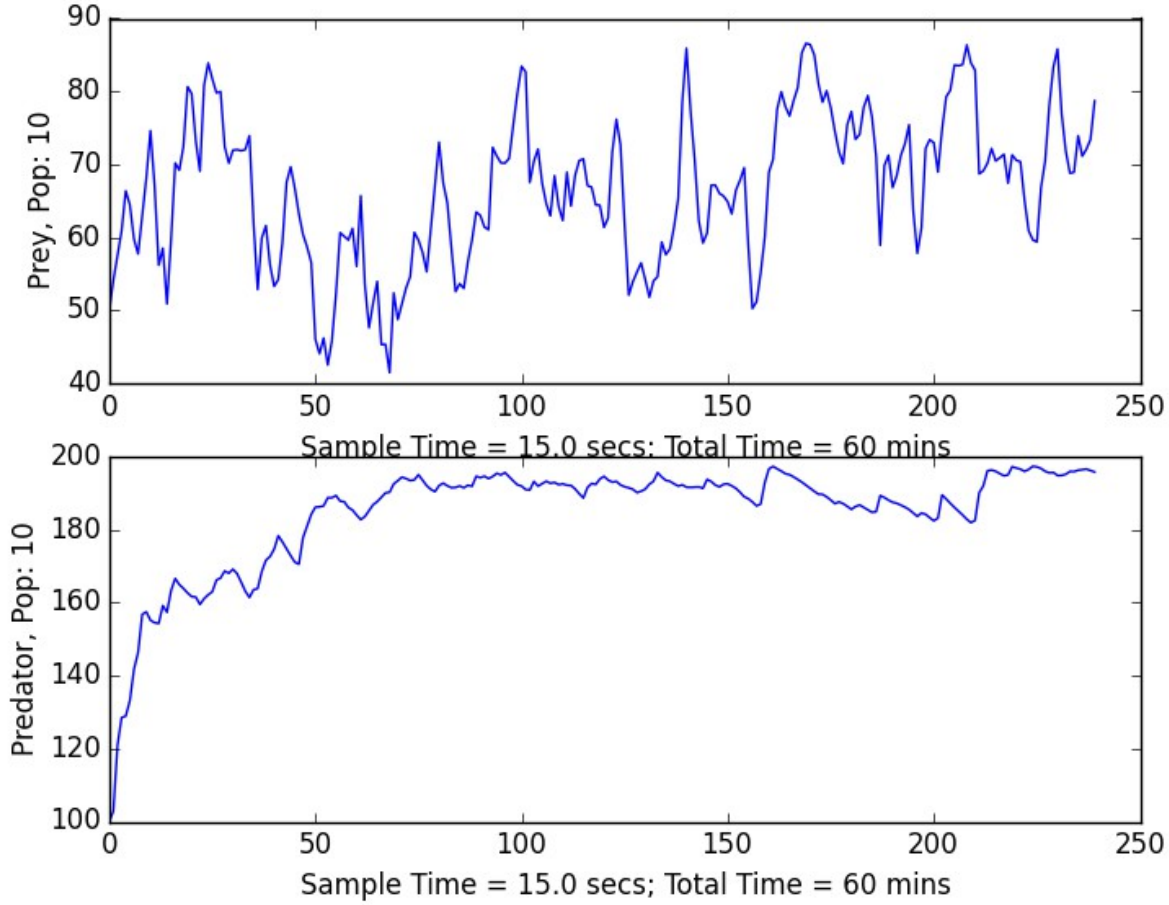


Figure 8: Experiment 2, Hour 2

Since the first hour of training had already been observed several times in Experiment 1, recording for Experiment 2 began after the first hour. The prey's successes relative to the predator's successes were only somewhat noticeable in the graph of the second hour. Although the predator remained visibly closer to equilibrium than the prey, it took significantly longer to converge with 10 predators in parallel than it did with any of the smaller groups in Experiment 1, trial 2—compared with the initially high rewards of the prey, this indicates that the prey was probably fleeing successfully in at least some states by the 2nd hour. In addition, another signal indicating chases appears in the steepness of some deviations from the predators convergence—this indicates the average of predator movement

speeds, with `Flee` and `Fight` behaviors causing significantly steeper slopes than `Search` or `Get Food` behaviors. The trend of delayed convergence for predators continued in the 3rd hour.

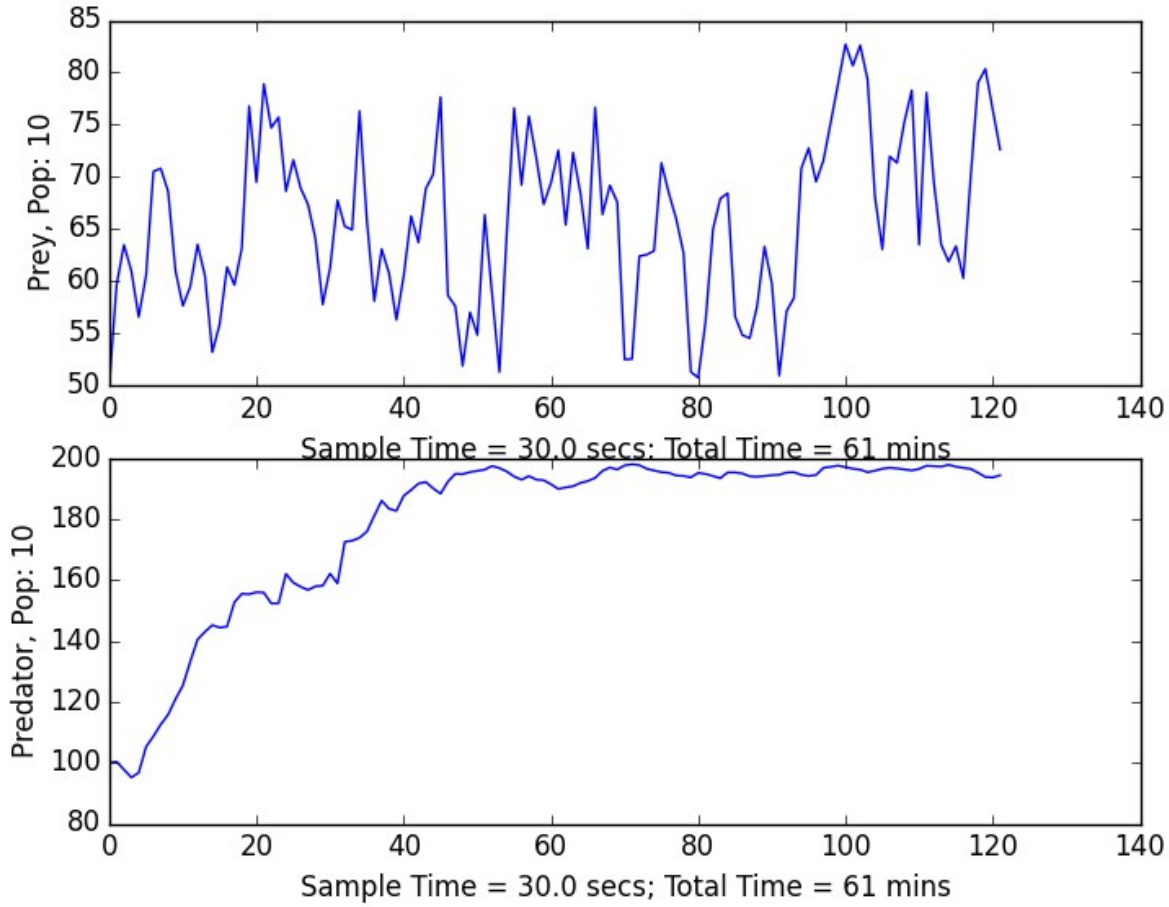


Figure 9: Experiment 2, Hour 3

Notice that the lines on these graphs are significantly smoother—this is because the sample time for running averages was increased from 15 to 30 seconds. As a result, detailed information about time spent chasing versus wandering are harder to determine, but indications of predator domination remained through its curves' repeated convergence relative to the prey's erratic reward curves. Hours four and five appeared very similar in appearance. At this point, puzzled by the similarity of these graphs to those of trials 1 and 2 in Experiment 1, I ran a final, 8-hour training session.

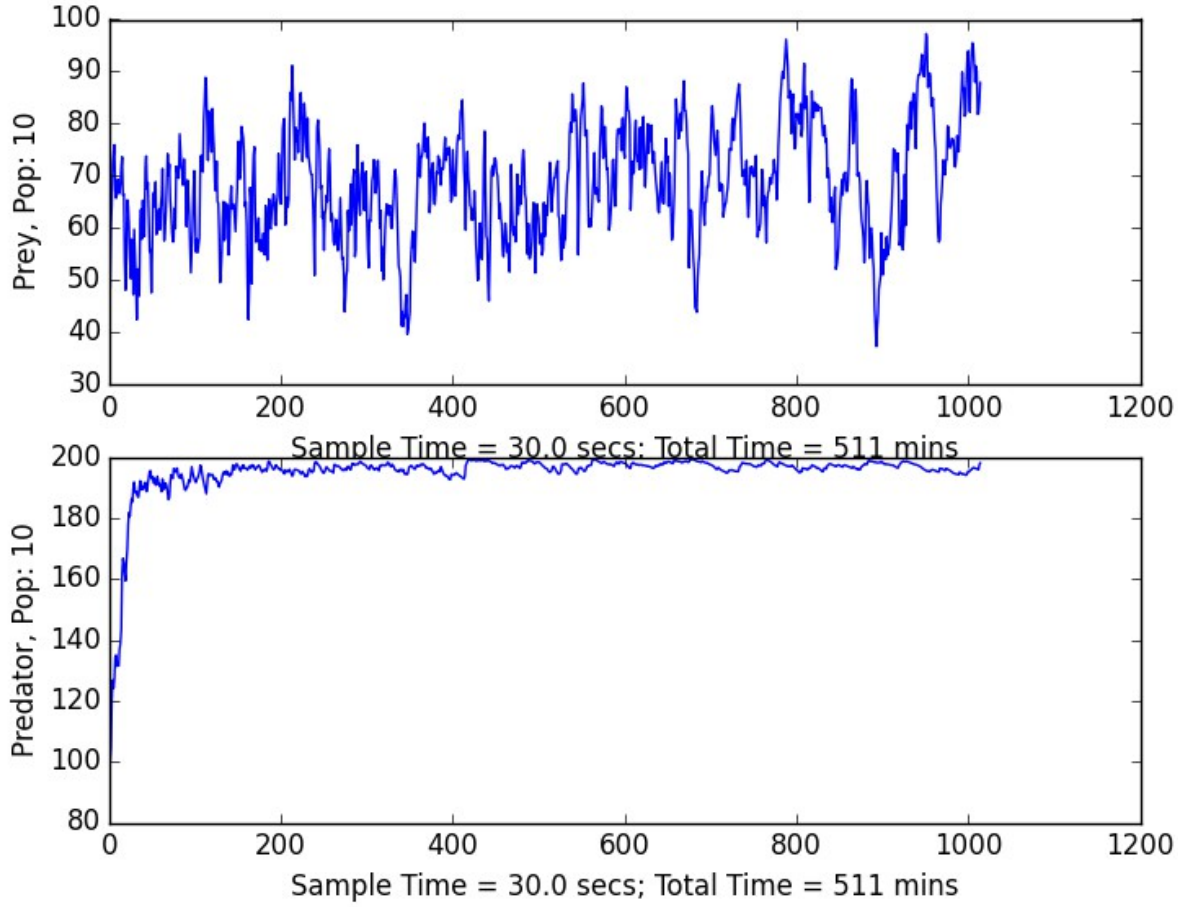


Figure 10: Experiment 2, Last 8 Hours

Although there are points in this graph that seem potentially meaningful, this visualization alone is not enough to understand what was happening—predator and prey Q-Tables must be analyzed. For prey, 22% of states had `Flee` as an optimal policy, with 7 (out of 58, with 256 total) of them invalid. The predator, on the other hand, only had 18.7% of states with `Fight` as an optimal policy, with 6 (out of 39, with 208 total) invalid (the significance of these percentages will be made clearer in analysis of Experiment 3). This indicates that the prey agents were reinforcing `Flee` slightly more than predators were reinforcing `Fight`--suggesting that these prey were learning to flee successfully, although evidently not successfully enough. For example, it contained {1:0:3:X::Flee} for all values of X, but

for the state tuple {1:1:X:0}, it had `Get Food` optimal for X = 0-2, but only `Flee` for {1:1:3:0}, when food was significantly farther away than the predator—a risky strategy. Although hypothetically possible, the prey didn't seem to use their knowledge of one another for any sort of emergent cooperative strategy (other than sometimes ignoring predators if other prey is nearby). The predators, on the other hand, display emergent cooperation clearly, both to an observer and in their Q-Tables. For example, the prey Q-Table state {2:0:3:X}, for X = 1-3 the optimal action is `Fight`, while for X = 0 , the optimal action is `Get Food` (which is invalid), indicating that the predator was significantly more successful in their attempts to hunt when there were other predators present, and even notably unsuccessful when attempted on their own. This makes intuitive sense—if a prey gets cornered by two predators (their most vulnerable situation), it's significantly less likely to escape than with just one[19]. Since states don't allow knowledge of how many of a certain type of agent is present, and prey don't gain an analogous advantage from their state representation as the predators do, this emergent strategy is only feasible for the predators with their current state variables. The occurrence of these strategies is interesting, but it still doesn't offer insight on the lingering question of whether or not prey agents can reach equilibrium in this system. However, after some further research, I realized there was a crucial component for multi-agent convergence that I had been missing.

IV. Experiment 3: Epsilon Greedy

Up until this point, random behaviors were chosen with either a constant 0.25 or 0.1 chance on every state transition for the entire duration of every training session—this is not the standard process for RL agents. One exploration function commonly used in TDL is the *ε-greedy* algorithm, in which actions are randomly chosen with the probability of a variable, ε, which decreases over time. In Experiment 3, I implemented ε-greedy as a replacement, with ε being the quotient of a time-sensitive exponentiation, represented as the four variables B, P, T, and C. B and P were related as B^P, with B

19 Although not indicated by graphs or Q-Tables, this can be seen occurring in supplemental video footage, so this statement is not solely based on intuition.

values being either 2 or 3 and P (initial values) values ranging from .3 to 2. C represents the amount that P is incremented every T minutes; C ranged from .3 to 1, and T ranged from 5 to 15. As such, chance for every agent of choosing a random action on a state chance was now ($1 / B^P$). Agents were trained for 6 more hour-long (or longer) sessions. The hypothesis was that, if predator and prey displayed fluctuating success rates, there would be some values of B^P that, by chance, could "freeze" prey as they approach equilibrium at the expense of predators (before predators learn to hunt in groups).

Experiment 3 Parameters:

Random Behavior Chance	$1 / (B^P)$; P increased by C every T minutes

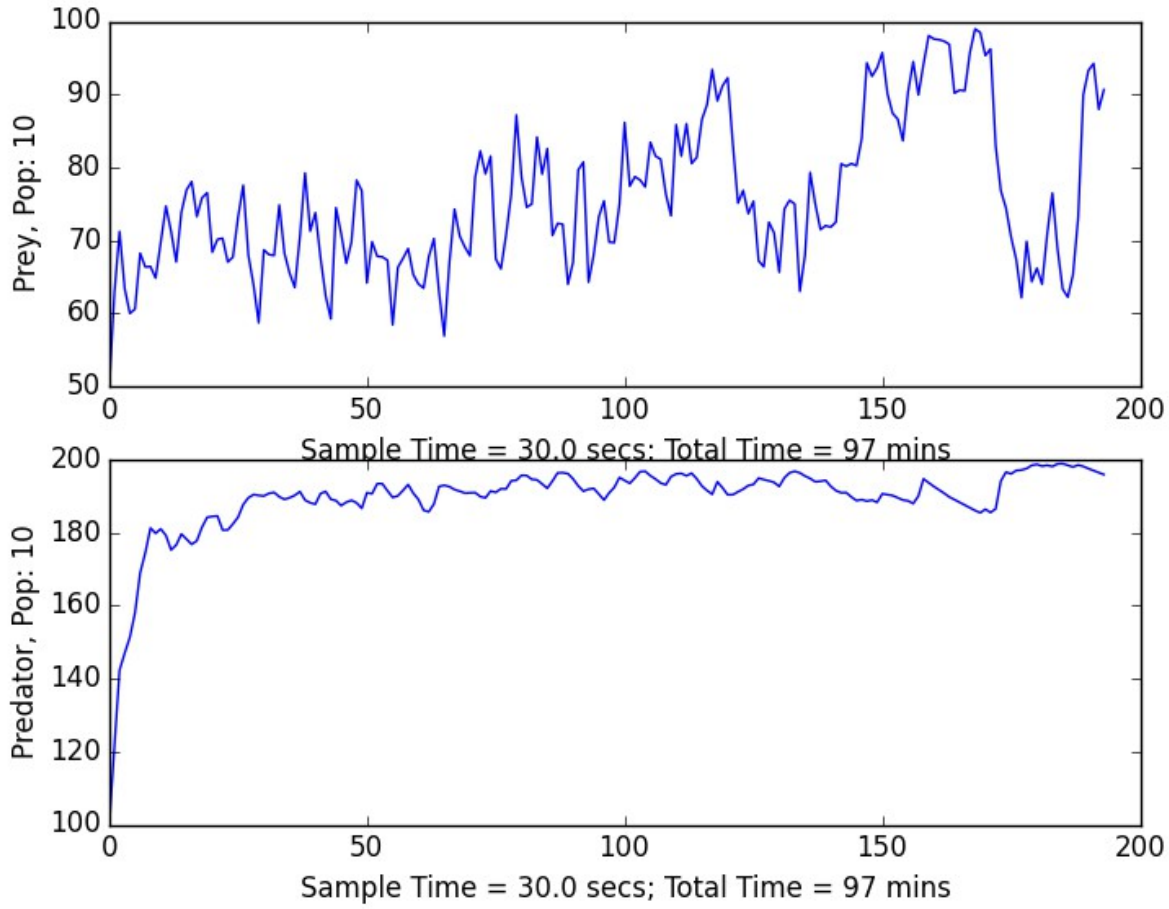


Figure 11: Experiment 3, B = 2, P = 2, C = 1, T = 10

In the session depicted by the above graph, T was 10 minutes, B was 2, P was 2, and C was 1. It appears very similar to graphs from the previous trials, although with some conspicuous dips in predator reward correlated with peaks in prey reward. `Flee` was optimal for 16.2% percent of states in the prey Q-Table, and `Fight` for 16.3% in the predators'. These facts, on their own, mean little. However, when trained again with only B changed (from 2 to 3), the following data was produced.

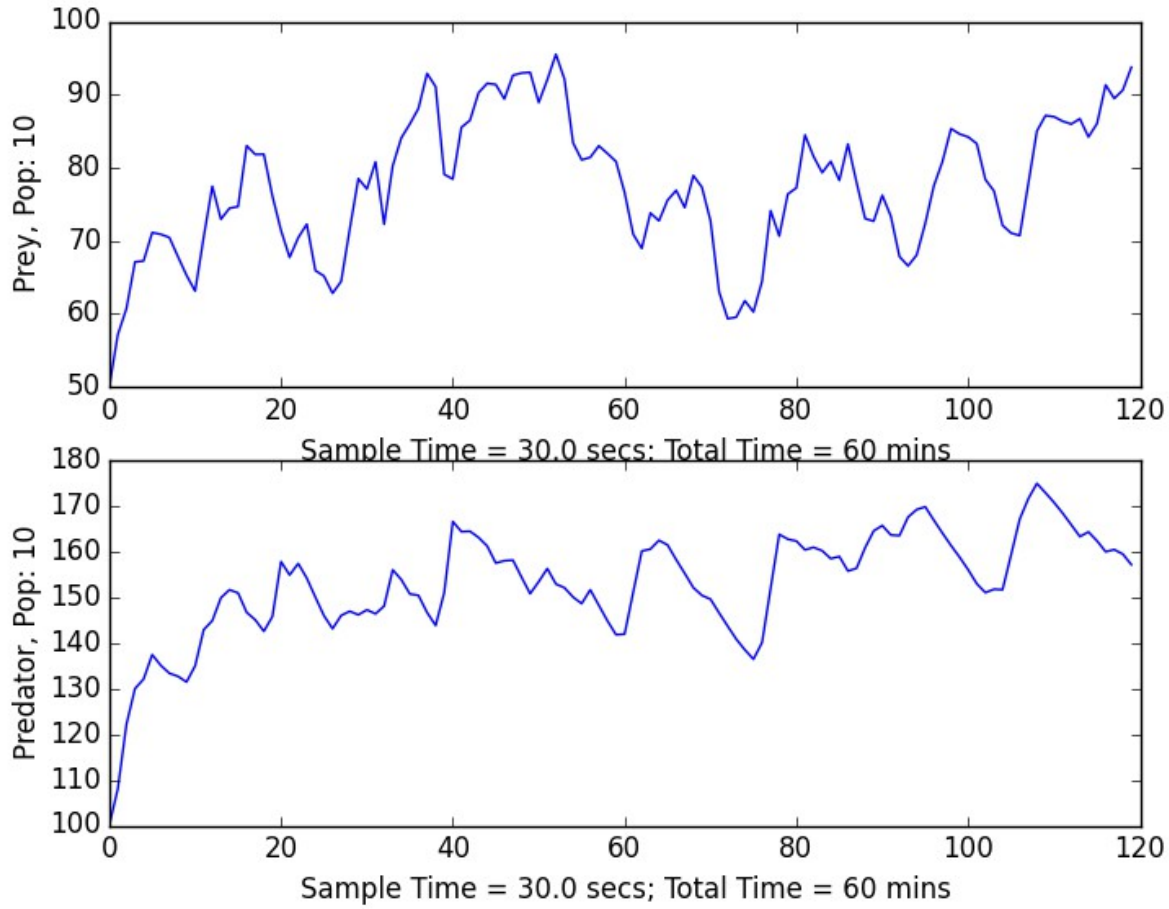


Figure 12: Experiment 3, B = 3, P = 2, C = 1, T = 10

In this graph, the convergence of the predator is visibly stunted due to the faster decrease in random behavior chances, and prey are appearing closer to equilibrium than they have yet—but are still visibly stunted. This is supported by the percentages for `Flee` (11%) and `Fight` (14%), as compared to those found in the Q-Tables of the previous agents whose graphs resembled those seen in previous

trials. For the session depicted in the following graph, the only parameters changed from the above were P (to 1.5) and C (to .5).

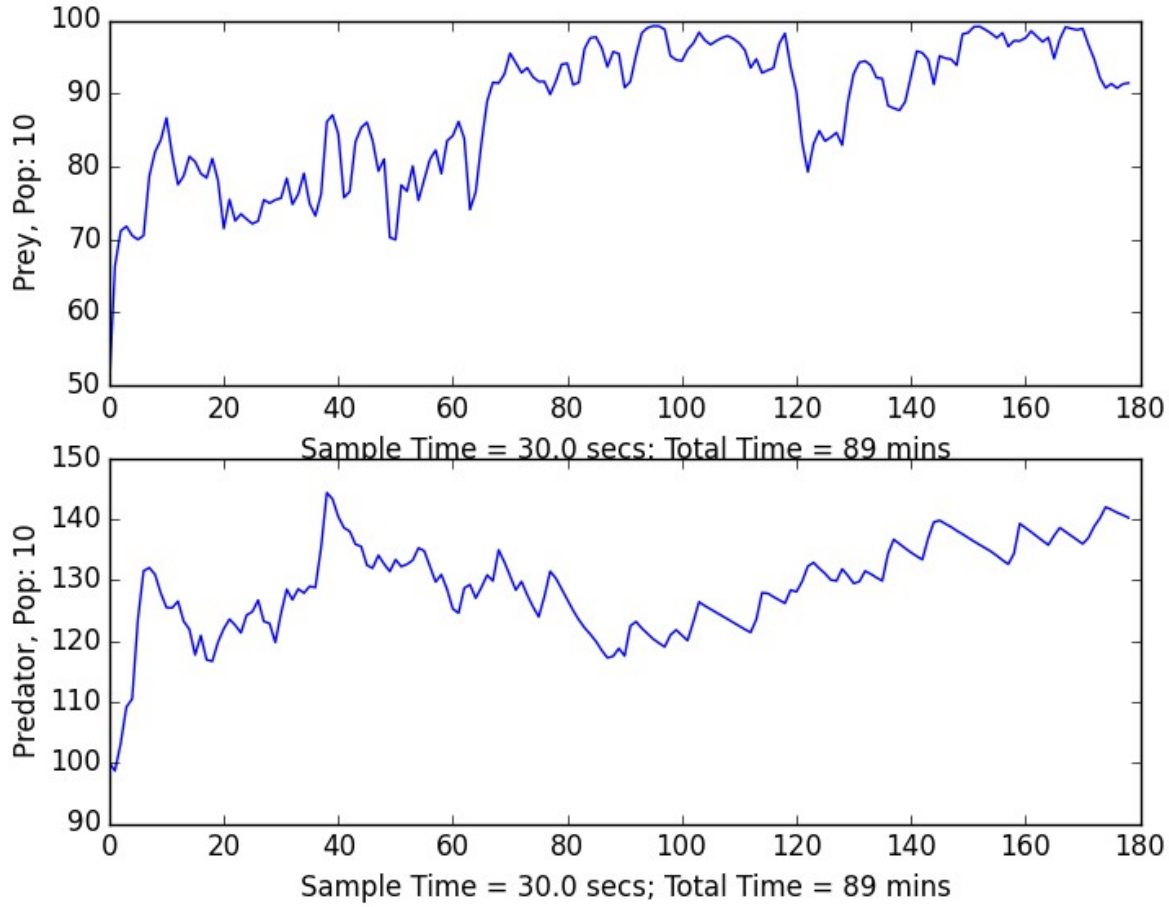


Figure 13: Experiment 3, B = 3, P = 1.5, C = .5, T = 10

This graph shows a dramatic departure from the prior ones, with prey approaching Nash equilibrium and predators hovering between ½ and ¾ of their usual equilibrium value. Further, the percentage of `Flee` as optimal in the prey Q-Table was 22.6% (significantly higher than previous percentages) and the percentage of `Fight` as optimal in the predator Q-Table was 8.4% (significantly lower than previous Q-Tables). Combined, this data suggests that, at some point in time, the prey were in fact converging to Nash equilibrium at the expense of the predator's success.

4. DISCUSSION

First and foremost, the bottom line of these sessions was demonstrated in the validation stage of data collection: reinforcement learning in Unity is possible. Still, especially given the wide range of variable parameters explored, the multi-agent trials still merit discussion. The first half of this discussion will discuss the results, and the second half will discuss potential means of improvement on this system.

Of the many variables explored during data collection, the first that should be addressed is session time. The duration of all training sessions was constrained by my personal schedule and project deadlines—one hour sessions were the most manageable option, and longer sessions (such as the 8 hour session in trial 3) were recorded overnight. As such, there is no current way to know how long it would take to train an agent's entire policy to equilibrium in a multi-agent setting—or even if equilibrium is guaranteed at all. However, the expected training time for shared-policy agents has been proposed to be $\Omega(1/N)$ for N agents (Ming Tan, 1993)--meaning that training speed decreases asymptotically with the number of agents in best-case situations. This seems to be supported by the required time to reach equilibrium for individual agents relative to multiple agents in Experiment 1, and suggests that the shared-policy approach is well-suited for decreasing required training time given a lack of knowledge of total training required. Further, it will be important to learn how much more training time is required by an agent as their state-space complexity increases through some approximation function. There was not enough time to collect data on this matter.

The lack of difference in predator convergence speeds superficially attributable to the variations in state-space complexity in between trials in Experiment 1 is pronounced, and can be explained by two factors: restricted policies, and physical/rational limitations. A restricted policy occurs when is when an agent only explores a fraction of its total policy in a session; this was manifested in a novel fashion

during many of the one-hour sessions where predators reached equilibria quickly, but then never left the 'high health' ({X:*:*:*}; X=3,2) state, and so had very underdeveloped policies for 'low health' ({X:*:*:*}; X=1,0)} levels. This, in combination with prey learning to flee, is probably responsible for the delayed predator equilibria in Experiment 2, when predators that reached equilibrium in a prior session (with underdeveloped hunting strategies at lower health levels) were reset to half of their maximum health. Although restricted policy spaces have been explored by arbitrarily limiting potential actions of otherwise-rational learning agents in stochastic games (Bowling, 2004), their occurrence this simulation was novel and unexpected. Bowling and Veloso discuss the concept of 'restricted equilibria' accompanying restricted policies (which will be necessary for understanding prey results), but the concepts of physical and rational limitations must be explained in-depth first if that is to be understood.

As defined by Bowling and Veloso, a limitation of an agent is “anything that can restrict the agent from learning or playing optimal policies.” Physical limitations stem from an agent's interactions with its environment. An example of a physical limitation would be the inability of prey to reliably choose a direction to flee in upon being cornered by a predator or group of predators--the emergent 'mobbing' behavior displayed by predators in the 2nd and 3rd Experiments, though, represents not only a physical limitation, but a rational limitation. Rational limitations stem from knowledge representation / behavior control constraints specified by the designer of the agent. In the case of predator mobbing in the 2nd and 3rd experiments, predators exploited the facts that a) they could sense the nearest other predator, b) they had a higher chance of cornering prey in groups, and c) they could deal more damage to prey in groups. Prey, on the other hand, only had a random chance of fleeing in an optimal direction upon being cornered, and also had no way to sense how many predators were present, hence the limitation. Other examples of rational limitations in this simulation are the inability of agents to know the distance between two sensed agents, or the requirement that agents move at different speeds when they're executing different behaviors.

When agents are limited in one or more of these ways, they are only sometimes guaranteed to converge to a "restricted equilibrium" at which they have maximized their actions optimally given said limitations. This concept has been explored with turn-based stochastic games, but it is unclear at this time whether or not the relationship of prey graphs if/when when predators converge (at least to a local optimum) in Unity indicate a true restricted equilibrium, another type of equilibrium, or no equilibrium at all—the same can be said for the predator in the Experiment 3 session where the prey's reward visibly converged at the expense of the predator. However, the observation that one agent converged consistently at the expense of the other (rather than finding some sort of clear, competitive co-equilibrium) must be explained before any conclusions on restricted equilibria can be supported.

Agent exploitation of rational and physical limitations was pervasive through all experiments, and should be regarded as an inevitability of the system. Many of these exploitations were desired (such as prey being more vulnerable when they attempted to get food, or prey being able to flee faster than predators could chase), but some emerged unexpectedly and merit scrutiny. One way to understand unintended agent exploitation of rational limitations is through analysis of sub-optimal action sequences and invalid policy entries—especially in Experiments 2 and 3. In terms of invalid states, their occurrences in predator policies are telling: almost all of the time, invalid entries consist of 'Get Food' being deemed optimal when there is no food but there are other predators around. Again, this action is 'invalid' because 'Get Food' cannot be executed when there is no food item perceived. This appeared to occur for one main reason – when other predators and prey were around, food could appear[20] for agents when they were taking a sub-optimal action (much like the problem with food spawning and prey perception noticed in the validation trials), but could disappear just as quickly—and all of these state changes (triggering policy modification) were caused by the actions of other agents. This type of invalid entry should hypothetically disappear given enough training time, which is supported by the overwhelming majority of invalid predator entries occurring in less-explored, lower

20 This would happen if the spawner created a food item next to prey, or if a prey died of starvation near a predator

health level states $\{X:*:*:*;\ X = 0,1\}$ in their policies[21].

Prey invalid entries, though they too appeared mostly at the lowest health level state in their policies, could not be attributed as much to lack of exploration, especially since prey were often injured and did a fair bit of exploration at lower health levels. Rather, these represented a fundamental rational limitation: agents could hypothetically starve to death. When, on the brink of death, and perceiving no food or predators, a prey's only valid action is to search. However, if they try to search but starve in the process, the policy entry for 'search' in that state will be significantly punished, leading to another action becoming the maximum and making the policy entry invalid. Like any rational limitation, this indicated a flaw in the simulation's setup, not the effectiveness of SARSA. Simlarly, when prey were killed during an unsuccessful 'flee' action, occasionally sub-optimal or invalid actions would become the maximum. When this happened, it allowed predators to hunt with much more success as the prey repeatedly need to relearn the value of 'flee' in some states, effectively exploiting invalid entries in prey policy to optimize their own. However, when the inverse occurred (predators unsuccessfully hunting prey due to a random or sub-optimal action choice), prey would learn sub-optimal policies by ignoring predators in some situations; this demonstrated another significant rational limitation of prey agents and another avenue in which predators were able to optimize their behavior at the expense of their prey's[22]. (See video supplement for footage of many of the scenarios described.)

Despite many explanations that have been / could be made regarding the situations described above, the problem of limited training time makes conclusive answers difficult to obtain, and leaves many questions unanswered. Perhaps the most interesting of these questions is: given enough time and no implementation of ε-greedy, will both agents reach either a Nash or restricted equilibrium, or will

21 One interesting exception to this was in the 8-hour session of trial 3, where more invalid entries were located at the highest health level. Although some of these appeared trivial, some seemed to alternately indicate cooperation and even competition between predators.

22 While it is hypothetically possible that, in some cases, these 'invalid' actions were optimal in some counter-intuitive sense (such as that, when agents aren't moving due to invalid actions, they are losing less health), there is no concrete evidence with which to support this, and this behavior is undesired for the specific task these agents are ultimately designed for.

optimality of each agent's policies oscillate continuously over time with phases of exploiting/being exploited (like the Lotka-Volterra formulation of predator-prey population dynamics)? As demonstrated in Experiment 3, ε-greedy causes agents to converge various (potentially sub-optimal) equilibria, effectively 'freezing' policies in place—an apparent 'phasing' of predator/prey optimality can result in one agent type's cemented dominance over another (including prey over predators in the final session of Experiment 3[23]). From this, another, more pragmatic question is raised: when (if ever) should agents stop learning? Given that no agent ever clearly 'completed' their policy, but total required training time for this task is unknown, it follows that a conclusion to this question should not be drawn from data presented here. Ultimately, more training time and more rigorous data collection are needed if the dynamics of this system are to be fully understood. However, there remain many systemic variables that remain to be explored before more focused data collection is merited, in order to attempt to solve some of the systemic problems identified.

At the end of the day the fact remains that, while classical games present a problem that can be optimally solved by an A.I., storytelling does not. As such, optimality or convergence towards Nash equilibrium cannot inherently be used as a measure of agent success from a ludonarrative standpoint; only as a confirmation that learning was, in fact, occurring. Sub-optimal policies may be desirable properties for agents in certain circumstances due to training time constraints, but also for both narrative and mechanical reasons such as characterization (an making a 'mistake' can still result in an interesting story) and difficulty balancing (player mistakes against an optimal learning agent opponent my cause the opponent to exploit the player in seemingly 'unfair' ways). With that said, even a desired sub-optimal policy will still be *mostly* optimal, and so a means to ensure this must be created, while further expanding the potential of this architecture to foment meaningfully complex agent behavior.

Of the current system parameters, several that are worth experimenting further with are alpha

23 The second session of Experiment 3 also hinted at a potential middleground, but this may have been due to a largely unexplored policy space, as well.

and gamma in the learning equation, more complex reward signals, and a more complex epsilon function. The effects of altering alpha, gamma, and reward cannot be fruitfully speculated on yet, but given the lack of knowledge about required training, one possible alternative ε-greedy function comes readily to mind. Rather than the probability of choosing a random action decreasing predictably over time, if said probability oscillated over time, then the relationships between agent policies would be periodically 'frozen' and 'thawed' repeatedly, potentially allowing for more reliable analysis of agent policies during 'frozen' periods and allowing those policies to continue to evolve in a 'thawing' period if they are not satisfactory. Also, for these trials, agent learning only occurred on state transitions: is there another event that could be used as a trigger for learning that would be more effective? Other rational limitations and parameters such as state representation and behavior design will require more rigorous design and balancing between agents—especially as agents are given increasingly diverse repertoires of states and behaviors.

On another note, invalid entries must be dealt with; there are multiple avenues through which this might be achieved. One of these is reward shaping—when a human helps fill out an agent's policy in order to speed up training time and also allow for greater designer control over the behavior exhibited by learners. This is desirable from an artistic standpoint (allowing more authorial control) as well as a practical one (speeding up training time); a program must be written to make this process as efficient and powerful as possible. As demonstrated from the state-tuple syntax earlier, certain types of states and their actions are easily generalized about—this fact should make reward shaping for large Q-Tables more feasible using this program. Aside from the ability to increase and decrease the value of actions from any given state for any given agent, one function that this program should have is the ability for a human designer to 'lock' a policy in place—once an optimal action has been found in a state, it can be 'locked' so that no other action can be chosen. This will be indispensable for dealing with invalid and sub-optimal entries, although, there are several potential complications that may arise

(such as agents decreasing their value of other optimal policy entries due to another being locked at a lower value than it might have achieved if it was 'unlocked'). Another avenue for reducing the problem of invalid entries could be the altering of the SARSA algorithm employed for this paper to a SARSA(λ) algorithm—which is almost identical to SARSA except that it has a few extra steps that are intended to reduce the propagation of high-error policy entries through the rest of the agent's policy through eligibility tracing.

Another issue that must be dealt with (both for reasons of artistic control and expediting training time) is policy complexity, especially as more complex state representations become desirable. One way I have thought of doing this is with a hybrid reinforcement learning / finite state machine agent, in which states and actions are constructed as before, but one action is a 'meta-action' that may execute a variety of other behaviors given the circumstance. For example, the actions of the agents studied in this paper could be reconstrued as “Fight,” “Flee,” and “[Defer]”, and “[Defer]” would consist of “Get Food” and “Search” being executed by a reactive FSM agent. Hypothetically, this should be able to support many behaviors wedged into a “[Defer]” state so long as the learning agent has some knowledge of its relationship to the goal designed by the FSM. This would allow for a more complex repertoire of behaviors to be employed by learning agents in addition to simplifying state-space, and for straightforward behaviors to be reliably authored by a designer rather than having to be learned; both of these outcomes are highly desirable.

Lastly, from a much more long-term view, there remain two further experiments that might prove fruitful. One of these is the notion of crowd-sourced training. In addition to reward shaping and a hybrid FSM, if many humans were able to log into a single server hosting the learning agents, their policies, and their environment (with those humans taking on similar constraints and roles occupied by the agents), agents might not only learn faster (since more computers rendering different regions of the enviroment means more learners could be learning in parallel) but agents might also begin learning

how to optimize their behavior with respect to both other agents *and* human players. Since ultimately these learning agents will end up coexisting in their environment with human agents, there is an incentive to train these agents partially with humans, as well. Another topic left to explore is motivated reinforcement learning, in which an agent employs a 'motivation' function instead of / in conjunction with their reward function in order to bias an agent towards exploring rare or novel occurrences (Maher and Merrick, 2009). The motivation function essentially models the agent's 'attention focus' in addition to reward, and can be used to both speed up agent training and/or allow an agent to be more successfully adaptive in a dynamic environment. While the potential of all the system modifications and optimizations described is exciting, there remains the high probability that there remain potential optimizations I have yet to discover, and so the future development of this system remains a frontier waiting to be explored and tested.

5. CONCLUSION

Overall, as an empirically-based proof of concept that multi-agent reinforcement learning is a viable means to structure AI behavior in Unity (using RAIN), this project was successful—but that's not the question this paper initially set out to help answer. In the end, this AI was brought into existence to provide characters that will help generate entertaining and educational dinosaur ludonarratives; so that those characters could be more interesting and complex than they could have feasibly been using hand-coded FSMs (the status quo AI architecture in most modern game design). While AI wildlife in featured other open-world games found success utilizing between roughly[24] 3 - 10 discrete states considered in behavior selection, my first iteration goal for this AI in *Saurian* is to have between 1,000 to 10,000 of them[25]. Development of this will be attempted by employing a several of the potential avenues described in the discussion section in parallel to reduce total training time as much as possible, while maximizing policy-size[26] and balancing agent parameters. This endeavor will signify one of the first times that authorship of AI characters in a narrative game can be said to have been shared by their human designers with the character AI itself. With luck, these characters will help tell the best dinosaur stories ever told—at least for those who have an interest in that have ever found themselves wondering, "what was it like to be a dinosaur?"

24 Estimate made from experimentation with agents within open world games Far Cry 3 and S.T.A.L.K.E.R.: Call of Pripyat in their intended environmental context, as well as through map editors and mods

25 I shouldn't say much about it here simply because I signed an NDA and modern game PR can be very complicated

26 Not only will the number of states each agent has dwarf those of agents in comparable modern games, but it will be trivial to use different sets of agent parameters at once, allowing for the creation of different policies reflecting "personality types" within single species—each of which can simply be saved and loaded in Q-Table form once trained/shaped, rather than having to be hand-coded individually.

6. BIBLIOGRAPHY

Bowling, M., & Veloso, M. M. (2004). *Existence of multiagent equilibria with limited agents*. Technical report CMU-CS-02-104, Computer Science Department, Carnegie Mellon University.

Merrick, K., & Maher, M. (2009). *Motivated Reinfocement Learning: Curious Characters for Multiuser Games*. Berlin: Springer.

Riedl, M. O., and Stern, A. (2006). *Believable agents and intelligent story adaptation for interactive storytelling*. In 3rd International Conference on Technologies for Interactive Digital Storytelling and Entertainment (TIDSE 2006), 1–12. Darmstad, Germany: Springer

Russell, S., & Norvig, P. (2010). *Artificial Intelligence: A Modern Approach.* (3rd ed., pp. 645-684, 830-853). Upper Saddle River, NJ: Pearson Education.

Sutton, R., & Barto, A. (1998). *Reinforcement Learning*. Cambridge, Mass.: MIT Press.

Tan, M. (1993). *Multi-agent reinforcement learning: independent vs. cooperative agents*. In Proceedings of the Tenth International Conference on Machine Learning, Amherst, Massachusetts. Morgan Kaufmann.

7. APPENDIX

a. Graphs

Experiment 1

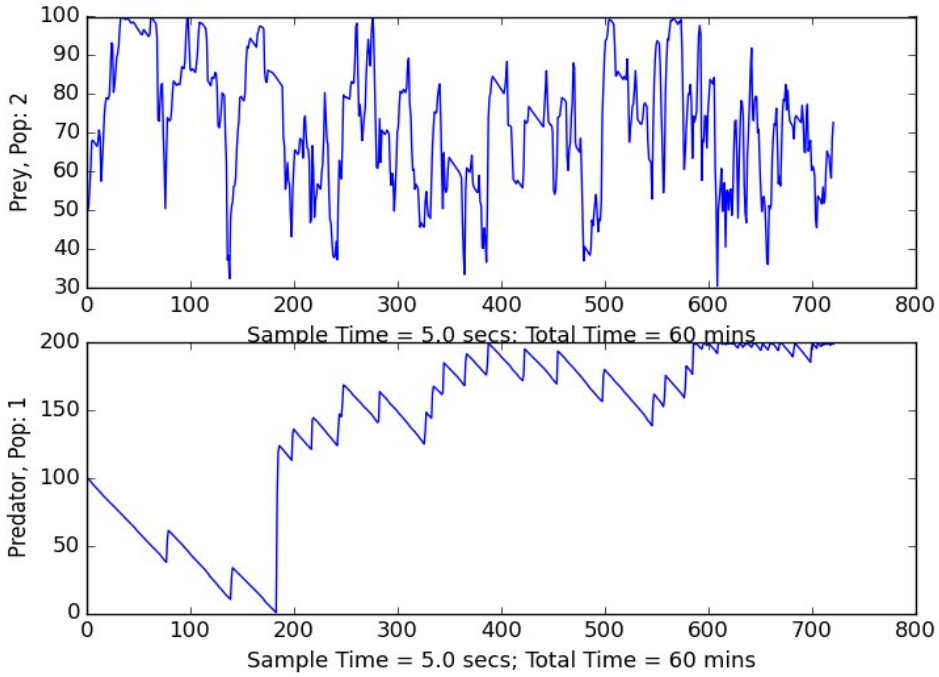


Illustration 6: 2:1 Trial 1

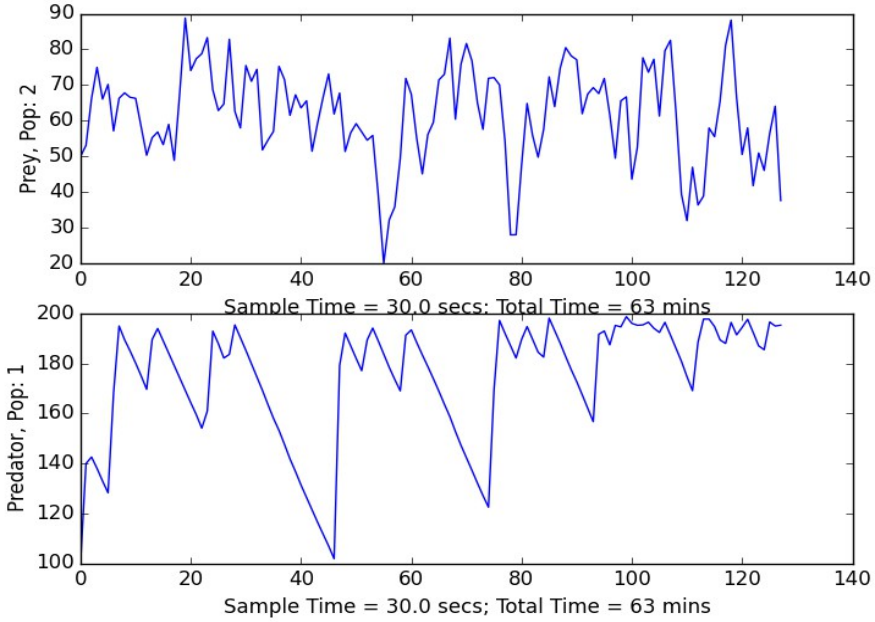


Illustration 7: 2:1 Trial 2

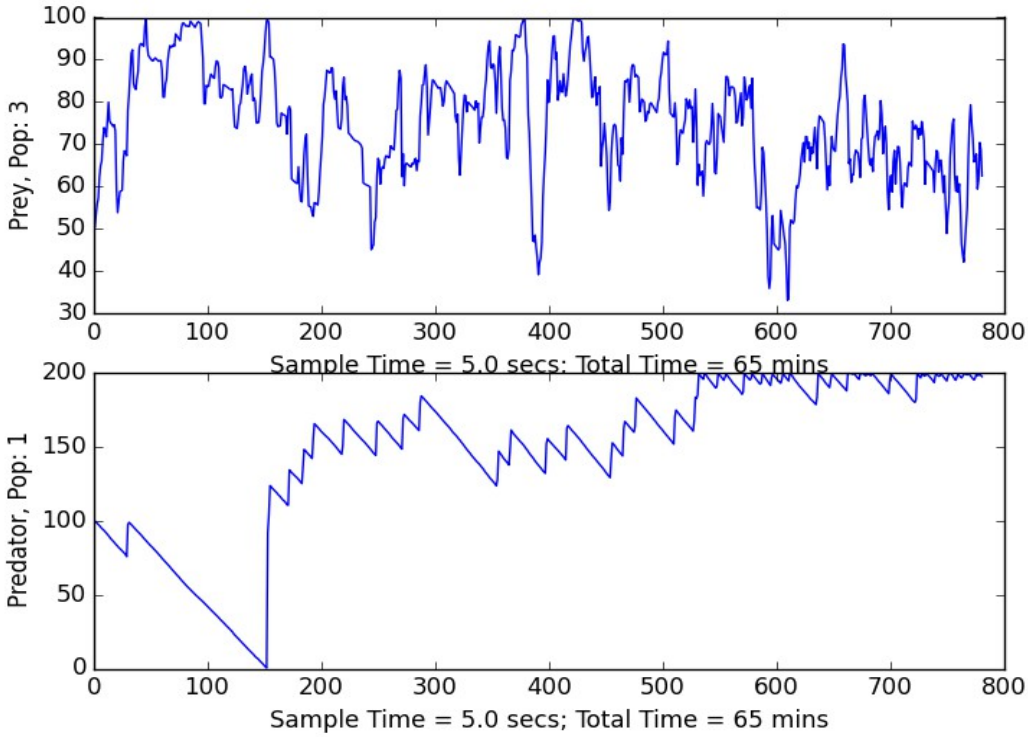


Illustration 8: 3:1 Trial 1

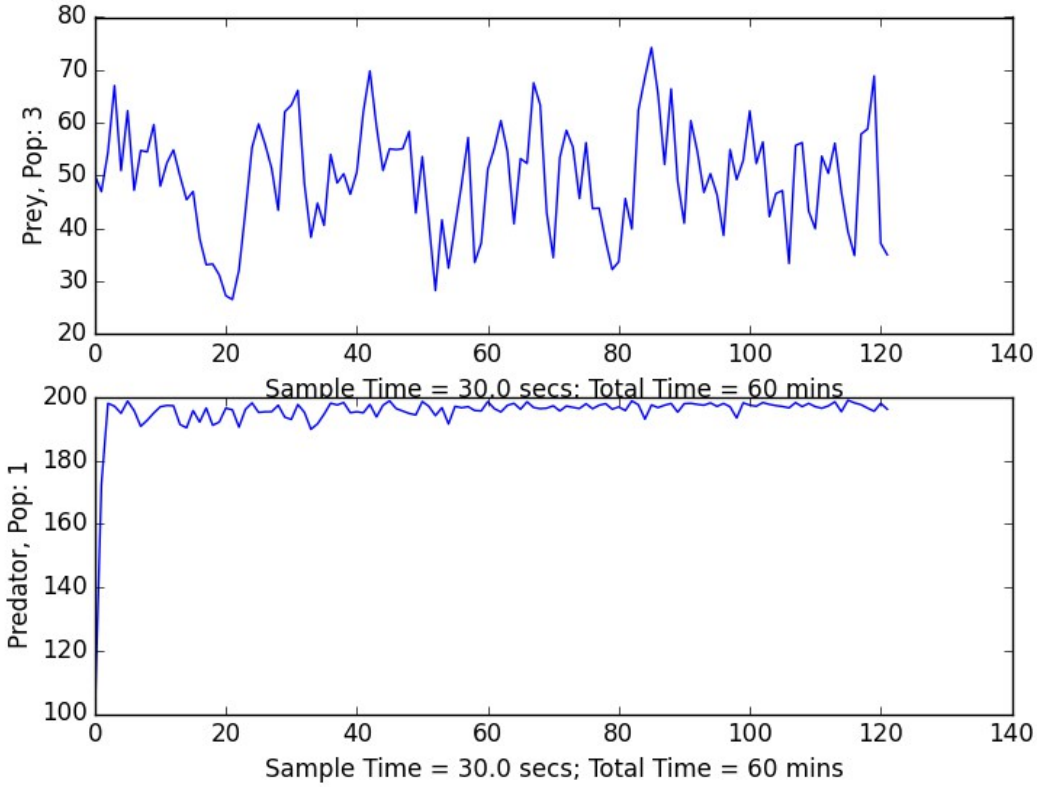
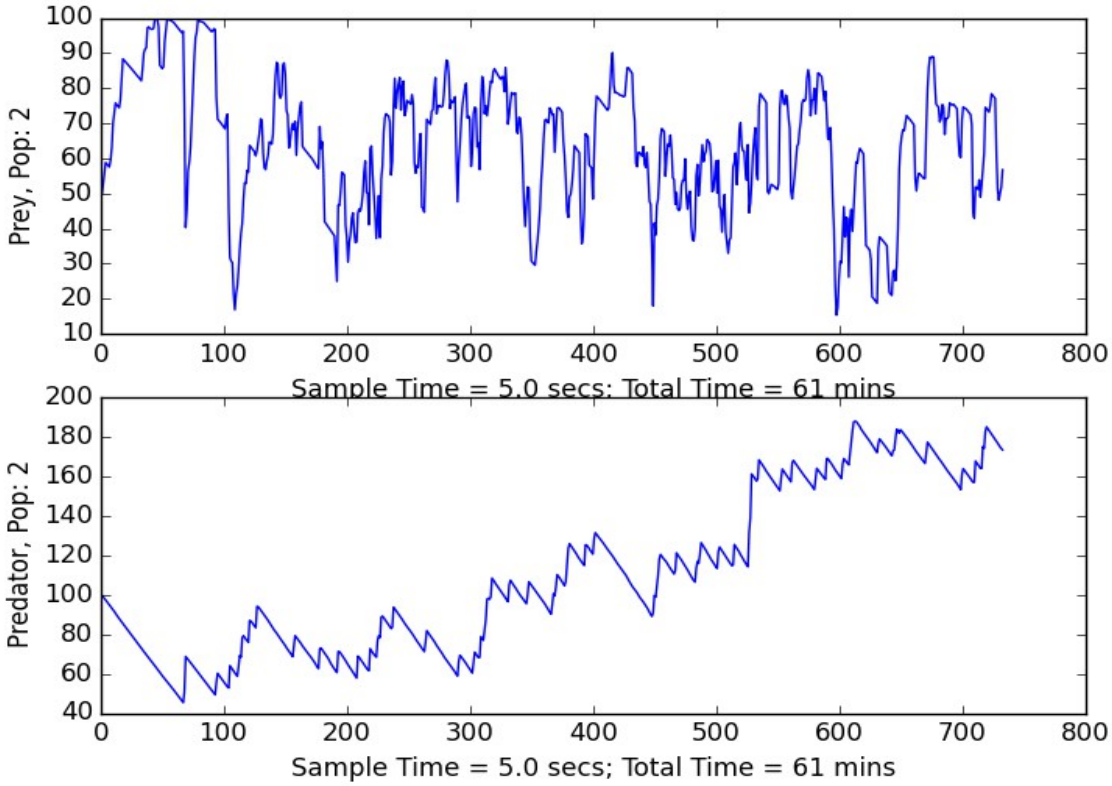


Illustration 9: 3:1 Trial 2



Illustr

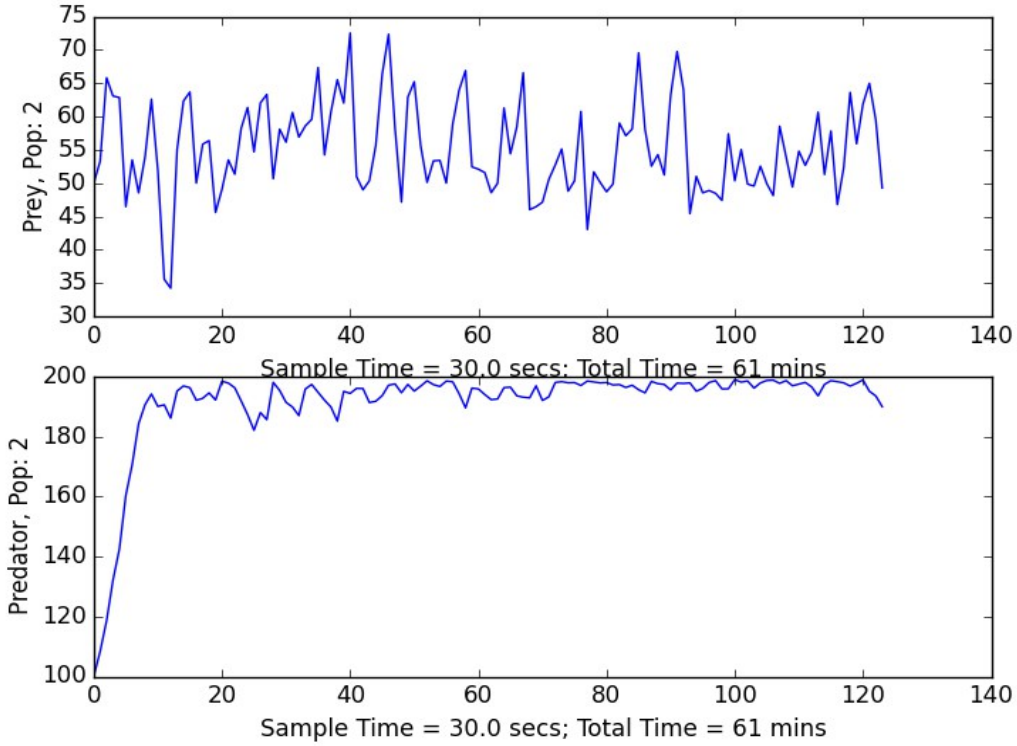


Illustration 11: 2:2 Trial 2

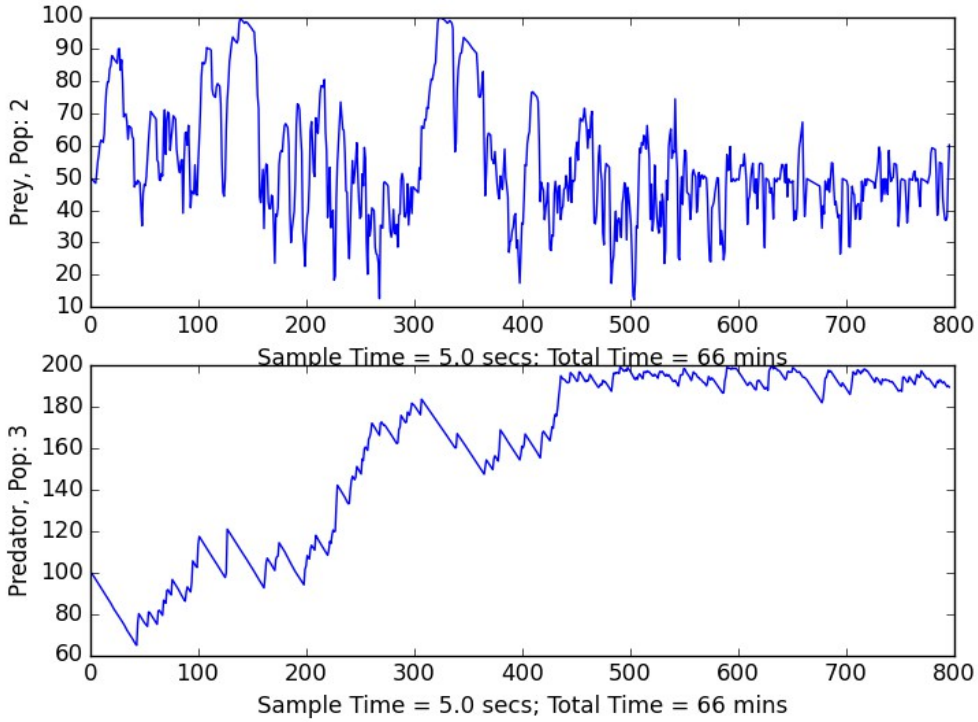


Illustration 12: 2:3 Trial 1

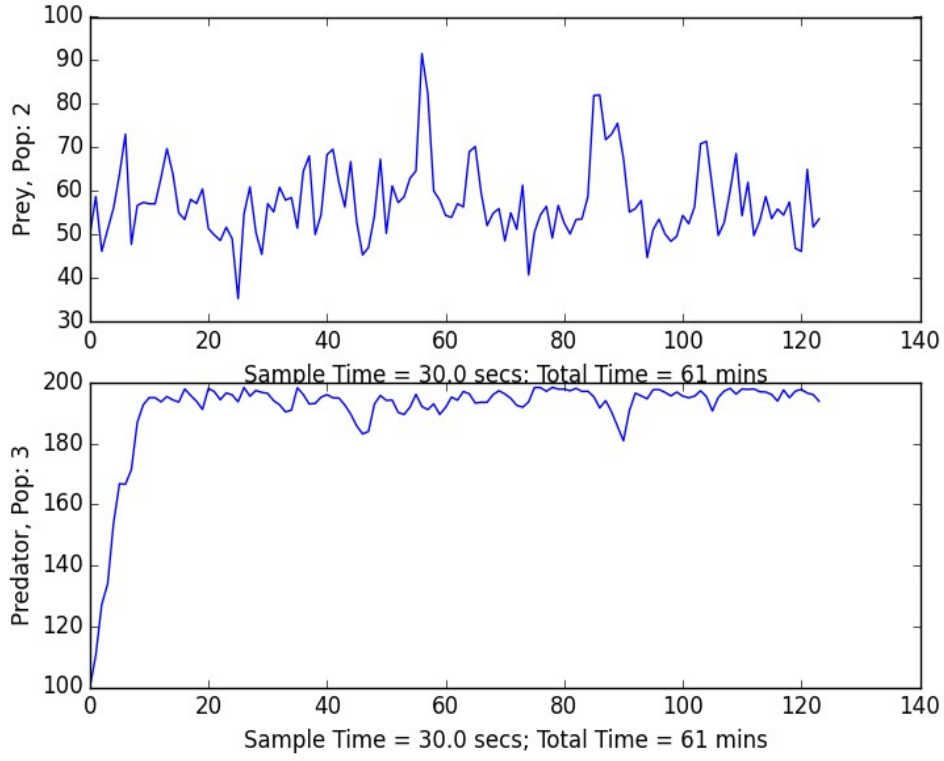


Illustration 13: 2:3 Trial 2

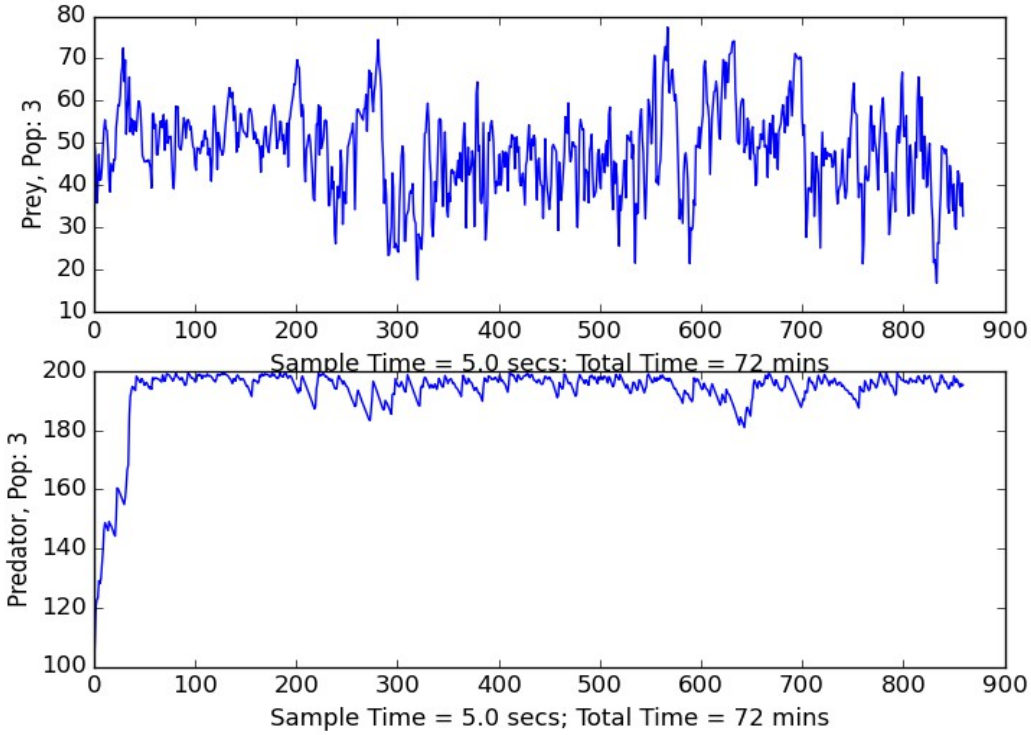


Illustration 14: 3:3 Trial 1

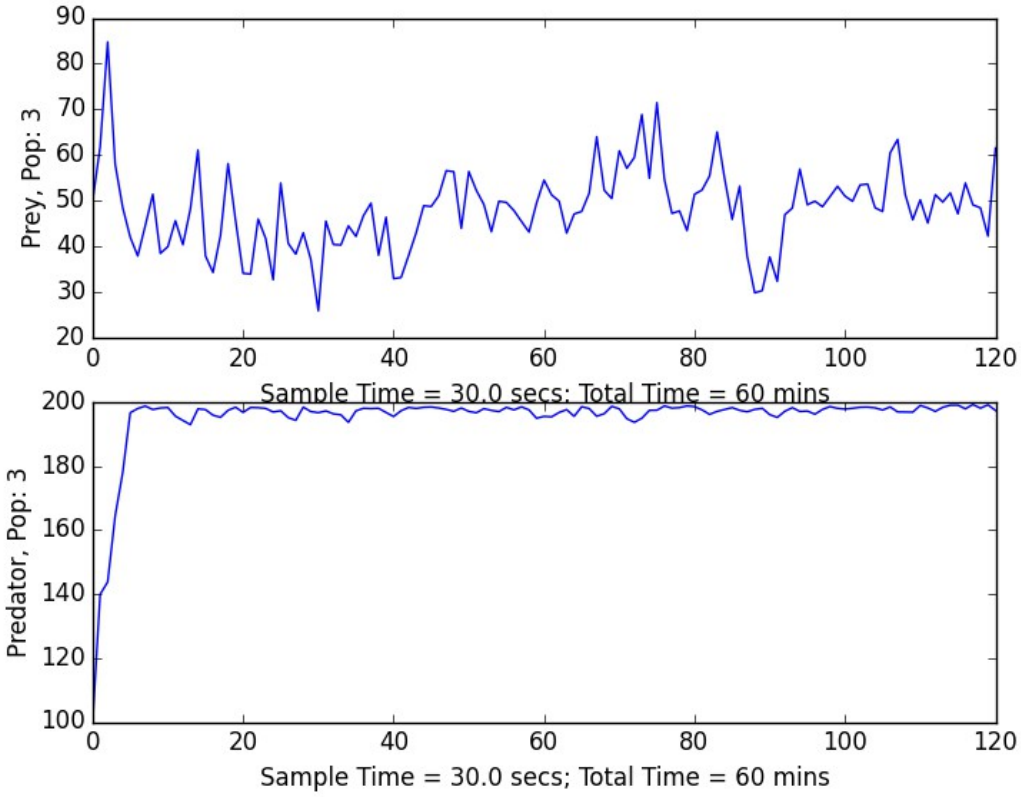


Illustration 15: 3:3 Trial 2

Experiment 2:

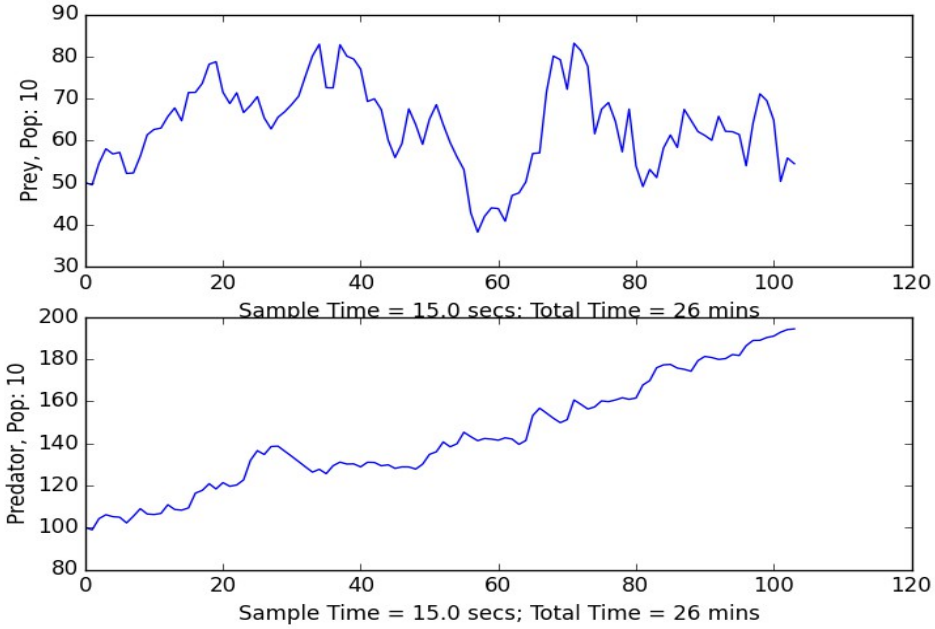


Illustration 16: First half hour

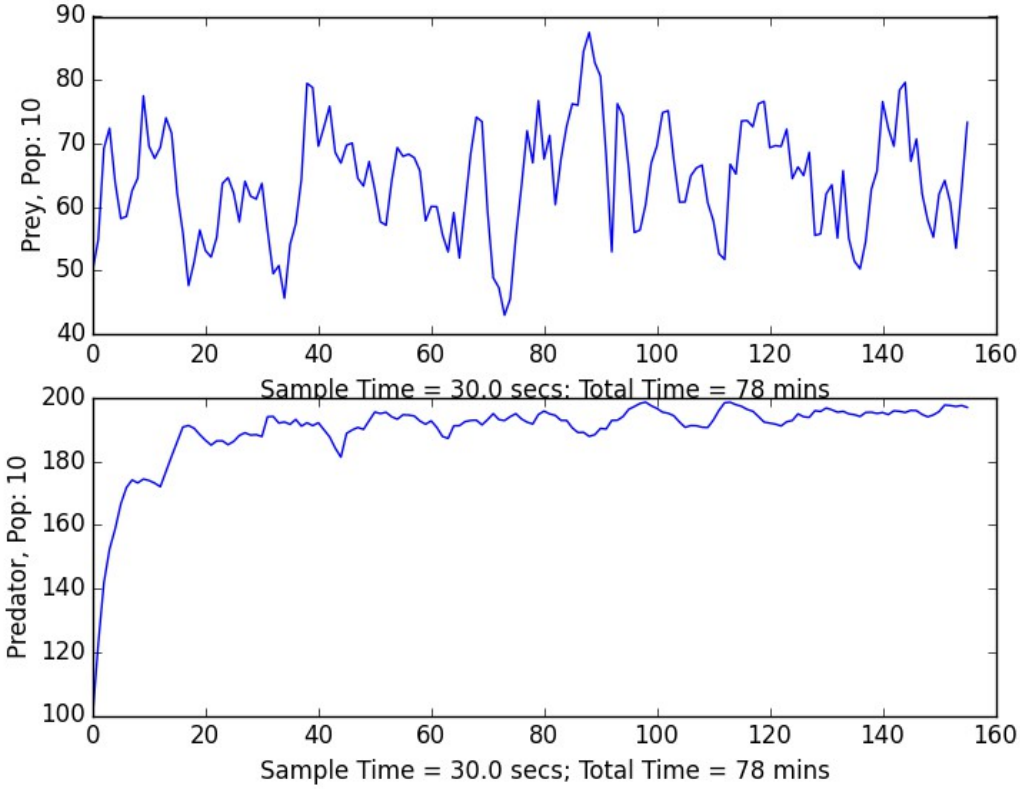


Illustration 17: Fifth hour

Experiment 3:

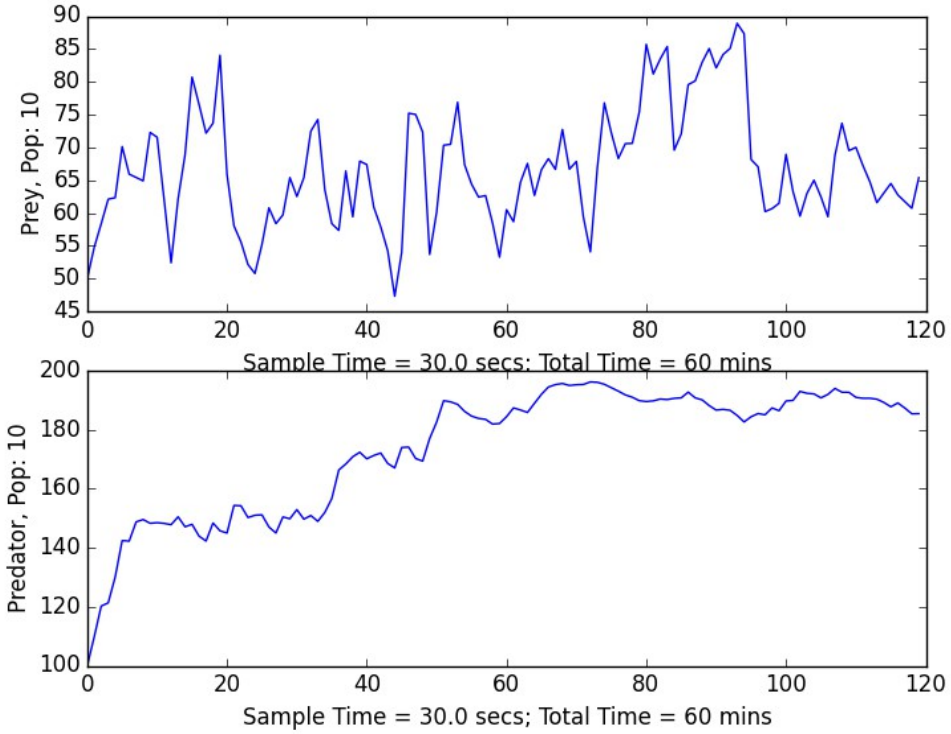


Illustration 18: B = 2, P = 2, C = 1, T = 6

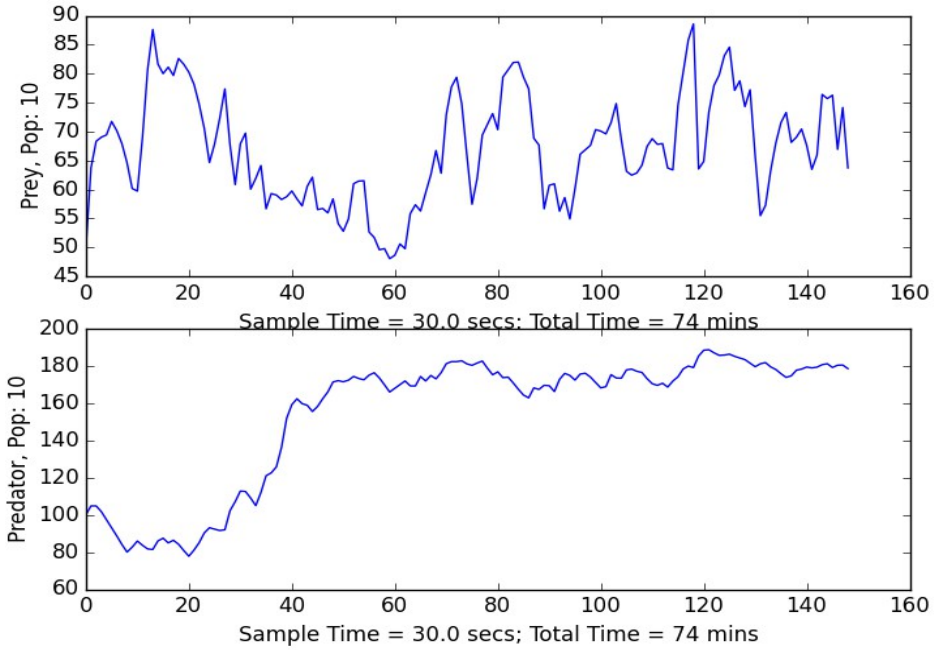


Illustration 19: B = 3, P = 2, C = 1, T = 15

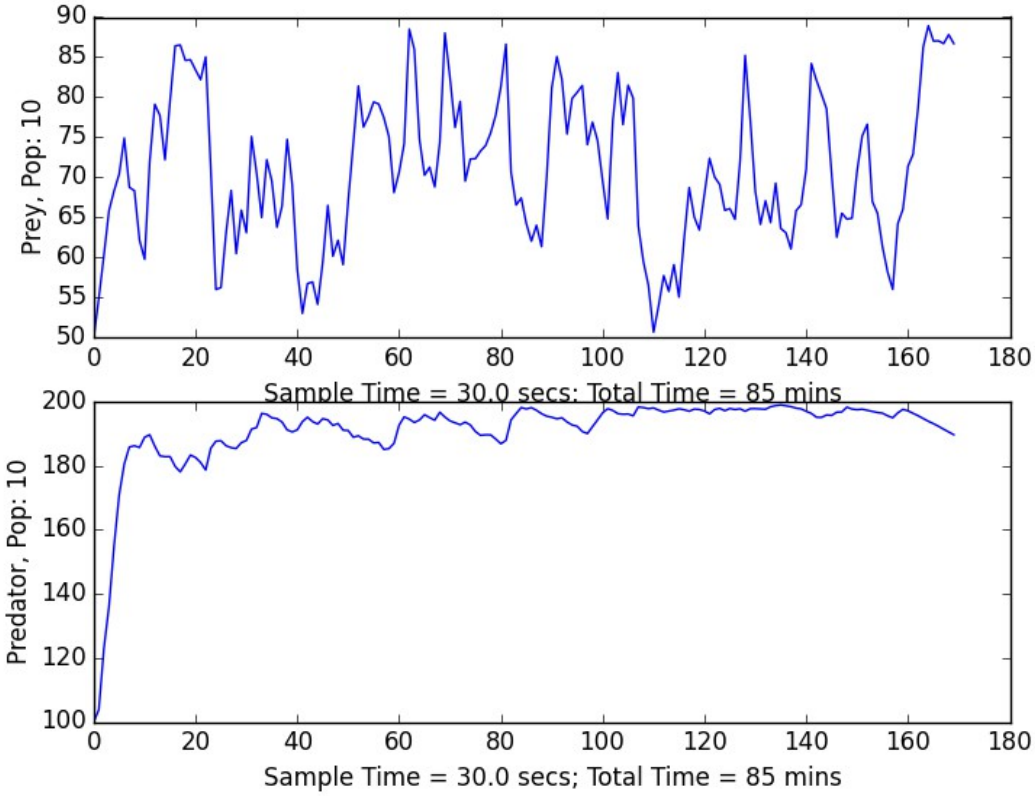


Illustration 20: B = 3, P = 1.5, C = .5, T = 10

b) Q-Tables

Note : States are in the left column, and actions are sorted from the left by highest value. The “F” followed by a number indicates the difference between the highest and second-highest entries. Also note that sometimes the same action will be listed twice after the top action – this was due to a data processing bug, and simply means that both alternative actions have a zero value associated with them.

I. Validation

TOTAL SIZE : 12

100 :: Search : 37.52372 --- Flee : 17.22717 --- GetFood : 0 --- ### F = 20.30
110 :: GetFood : 22.83295 --- Search : 0 --- Search : 0 --- ### F = 22.83
120 :: GetFood : 21.95606 --- Search : 16.0246 --- Flee : 0 --- ### F = 5.93
130 :: GetFood : 30.22511 --- Search : 0 --- Search : 0 --- ### F = 30.23
200 :: Search : 73.64182 --- GetFood : 20.78385 --- Flee : 0 --- ### F = 52.86
210 :: GetFood : 55.45303 --- Search : 11.65053 --- Flee : 0 --- ### F = 43.80
220 :: GetFood : 64.40755 --- Search : 27.6986 --- Flee : 0 --- ### F = 36.71
230 :: GetFood : 70.63702 --- Search : 17.97764 --- Flee : 0 --- ### F = 52.66
300 :: Flee : 727.2042 --- Search : 685.0306 --- GetFood : 678.7315 --- ### F = 42.17
310 :: GetFood : 709.4166 --- Search : 665.1716 --- Flee : 174.6643 --- ### F = 44.25
320 :: GetFood : 730.8027 --- Search : 663.906 --- Flee : 374.223 --- ### F = 66.90
330 :: GetFood : 717.5525 --- Search : 669.0595 --- Flee : 120.6328 --- ### F = 48.49

II. Experiment 2 – Prey

TOTAL SIZE : 256

0000 :: Search : 67.6768 --- Flee : 61.84586 --- GetFood : 37.21073 --- ### F = 5.83
0001 :: Search : 96.08649 --- GetFood : 30.41841 --- Flee : 15.20777 --- ### F = 65.67
0002 :: Search : 107.0729 --- Flee : 38.53244 --- GetFood : 5.953106 --- ### F = 68.54
0003 :: Search : 98.41073 --- Flee : 7.343667 --- GetFood : -10 --- ### F = 91.07
0010 :: Flee : 113.1799 --- Search : 89.03091 --- GetFood : 85.061 --- ### F = 24.15
0011 :: Search : 99.30512 --- Flee : 69.04684 --- GetFood : 65.27785 --- ### F = 30.26
0012 :: Search : 134.0787 --- Flee : 64.97105 --- GetFood : 53.61763 --- ### F = 69.11
0013 :: Search : 117.0427 --- Flee : 48.64071 --- GetFood : 39.42953 --- ### F = 68.40
0020 :: Flee : 116.22 --- Search : 93.33569 --- GetFood : 70.26997 --- ### F = 22.88
0021 :: GetFood : 103.9893 --- Flee : 88.06084 --- Search : 87.9817 --- ### F = 15.93
0022 :: Flee : 102.2112 --- GetFood : 75.7423 --- Search : 75.131 --- ### F = 26.47
0023 :: Flee : 122.872 --- Search : 64.95717 --- GetFood : 47.29853 --- ### F = 57.91
0030 :: Search : 110.1493 --- Flee : 59.79434 --- GetFood : 49.52063 --- ### F = 50.35
0031 :: Flee : 103.0316 --- Search : 63.94558 --- GetFood : 51.03719 --- ### F = 39.09
0032 :: Search : 100.4217 --- Flee : 96.71503 --- GetFood : 36.05755 --- ### F = 3.71
0033 :: Search : 129.1195 --- Flee : 40.87999 --- GetFood : 18.2906 --- ### F = 88.24
0100 :: GetFood : 53.74468 --- Search : 7.935718 --- Flee : -37.27856 --- ### F = 45.81
0101 :: Search : 53.61889 --- Flee : -19.11027 --- GetFood : -19.32399 --- ### F = 72.73
0102 :: Search : 52.29031 --- GetFood : 4.856452 --- Flee : 3.615269 --- ### F = 47.43
0103 :: GetFood : 13.09255 --- Flee : 0 --- Search : -10 --- ### F = 13.09
0110 :: Flee : 78.65779 --- GetFood : 45.67843 --- Search : 43.50354 --- ### F = 32.98
0111 :: GetFood : 54.80378 --- Search : 17.50574 --- Flee : 12.1513 --- ### F = 37.30

0112 :: GetFood : 72.53156 --- Search : 15.11104 --- Flee : 7.708738 --- ### F = 57.42
0113 :: GetFood : 42.3735 --- Flee : 6.795072 --- Search : 4.495862 --- ### F = 35.58
0120 :: GetFood : 91.53928 --- Search : 72.00924 --- Flee : 68.00868 --- ### F = 19.53
0121 :: GetFood : 82.52023 --- Search : 47.24268 --- Flee : 43.83266 --- ### F = 35.28
0122 :: Search : 72.4566 --- Flee : 31.91933 --- GetFood : 30.04095 --- ### F = 40.54
0123 :: Flee : 62.40491 --- Search : 13.99664 --- GetFood : 9.829952 --- ### F = 48.41
0130 :: Flee : 91.87353 --- GetFood : 59.1102 --- Search : 57.23759 --- ### F = 32.76
0131 :: GetFood : 81.50374 --- Search : 24.72481 --- Flee : 16.7634 --- ### F = 56.78
0132 :: Search : 63.10063 --- GetFood : 5.165821 --- Flee : 2.755835 --- ### F = 57.93
0133 :: Flee : 50.98614 --- Search : 9.52101 --- GetFood : 1.651319 --- ### F = 41.47
0200 :: GetFood : 40.54389 --- Search : -27.08778 --- Flee : -38.93747 --- ### F = 67.63
0201 :: Search : 5.695435 --- GetFood : -12.61178 --- Flee : -15.33796 --- ### F = 18.31
0202 :: Search : 52.4115 --- Flee : -3.780509 --- GetFood : -10.68732 --- ### F = 56.19
0203 :: Search : 31.80936 --- Flee : -1.958056 --- GetFood : -6.334149 --- ### F = 33.77
0210 :: Search : 66.81268 --- GetFood : 48.49282 --- Flee : 32.26796 --- ### F = 18.32
0211 :: GetFood : 50.21678 --- Flee : 21.21751 --- Search : 14.44671 --- ### F = 29.00
0212 :: Search : 67.27359 --- GetFood : 26.07664 --- Flee : 23.06919 --- ### F = 41.20
0213 :: Search : 45.1456 --- GetFood : 10.74136 --- Flee : 0 --- ### F = 34.40
0220 :: Search : 87.80352 --- Flee : 63.36256 --- GetFood : 61.13007 --- ### F = 24.44
0221 :: GetFood : 77.55539 --- Flee : 41.24164 --- Search : 36.62907 --- ### F = 36.31
0222 :: Flee : 80.25166 --- Search : 50.58014 --- GetFood : 33.19976 --- ### F = 29.67
0223 :: Search : 68.98602 --- Flee : 7.770561 --- GetFood : 3.964829 --- ### F = 61.22
0230 :: Flee : 88.74117 --- GetFood : 58.90759 --- Search : 47.1473 --- ### F = 29.83
0231 :: GetFood : 68.72029 --- Flee : 35.02755 --- Search : 34.31396 --- ### F = 33.69
0232 :: GetFood : 71.65993 --- Flee : 33.30827 --- Search : 13.80995 --- ### F = 38.35
0233 :: Flee : 54.51094 --- GetFood : 9.578693 --- Search : 3.781654 --- ### F = 44.93
0300 :: GetFood : 74.29652 --- Search : -36.48111 --- Flee : -46.59507 --- ### F = 110.78
0301 :: Flee : 11.00077 --- Search : -16.51689 --- GetFood : -25.79238 --- ### F = 27.52
0302 :: Flee : 20.01214 --- Search : -27.16531 --- GetFood : -30.60103 --- ### F = 47.18
0303 :: GetFood : 21.88531 --- Search : -13.68266 --- Flee : -14.67734 --- ### F = 35.57
0310 :: GetFood : 93.78444 --- Search : 49.06796 --- Flee : 33.31086 --- ### F = 44.72
0311 :: GetFood : 62.47161 --- Search : 6.450073 --- Flee : 0 --- ### F = 56.02
0312 :: Flee : 48.82199 --- Search : 11.16797 --- GetFood : 6.581468 --- ### F = 37.65
0313 :: GetFood : 55.59904 --- Flee : 1.840342 --- Search : 1.404241 --- ### F = 53.76
0320 :: Flee : 88.62923 --- Search : 68.39048 --- GetFood : 59.71414 --- ### F = 20.24
0321 :: Search : 73.90187 --- GetFood : 20.82001 --- Flee : 18.39926 --- ### F = 53.08
0322 :: GetFood : 75.16652 --- Search : 29.41099 --- Flee : 18.80024 --- ### F = 45.76
0323 :: GetFood : 73.32706 --- Search : 27.03074 --- Flee : 18.96578 --- ### F = 46.30
0330 :: Flee : 74.02936 --- GetFood : 53.73329 --- Search : 51.68687 --- ### F = 20.30
0331 :: GetFood : 57.22306 --- Search : 34.41155 --- Flee : 4.375122 --- ### F = 22.81
0332 :: GetFood : 51.76521 --- Flee : 9.796912 --- Search : 8.983916 --- ### F = 41.97
0333 :: Flee : 60.41762 --- GetFood : 32.8528 --- Search : 3.182885 --- ### F = 27.56
1000 :: Search : 433.6383 --- GetFood : 397.1736 --- Flee : 374.2727 --- ### F = 36.46
1001 :: Search : 434.6396 --- GetFood : 410.5652 --- Flee : 380.4487 --- ### F = 24.07
1002 :: Search : 438.8299 --- Flee : 418.9492 --- GetFood : 389.46 --- ### F = 19.88
1003 :: Search : 442.5466 --- Flee : 428.5598 --- GetFood : 400.2452 --- ### F = 13.99
1010 :: GetFood : 437.5595 --- Flee : 435.2439 --- Search : 431.7252 --- ### F = 2.32
1011 :: Flee : 434.6463 --- Search : 406.4792 --- GetFood : 379.5339 --- ### F = 28.17
1012 :: Flee : 443.4216 --- GetFood : 430.4921 --- Search : 425.7394 --- ### F = 12.93
1013 :: Flee : 442.8084 --- Search : 410.6366 --- GetFood : 410.1712 --- ### F = 32.17
1020 :: GetFood : 413.1824 --- Flee : 397.3582 --- Search : 383.3504 --- ### F = 15.82
1021 :: Search : 434.6052 --- GetFood : 384.8432 --- Flee : 377.9509 --- ### F = 49.76
1022 :: Flee : 430.8717 --- Search : 421.8928 --- GetFood : 389.7988 --- ### F = 8.98
1023 :: Flee : 429.6317 --- Search : 400.3912 --- GetFood : 323.944 --- ### F = 29.24
1030 :: Flee : 423.7148 --- Search : 394.2582 --- GetFood : 384.8987 --- ### F = 29.46
1031 :: Flee : 429.532 --- Search : 377.232 --- GetFood : 299.3895 --- ### F = 52.30
1032 :: Flee : 438.5064 --- Search : 386.6541 --- GetFood : 171.4656 --- ### F = 51.85
1033 :: Flee : 432.3193 --- Search : 347.1692 --- GetFood : 180.3088 --- ### F = 85.15

1100 :: Search : 398.3738 --- GetFood : 348.0575 --- Flee : 237.779 --- ### F = 50.32
1101 :: Search : 402.26 --- GetFood : 345.9488 --- Flee : 0 --- ### F = 56.31
1102 :: GetFood : 385.6151 --- Search : 238.0244 --- Flee : 62.7371 --- ### F = 147.59
1103 :: GetFood : 389.4997 --- Search : 137.6517 --- Flee : 62.7241 --- ### F = 251.85
1110 :: GetFood : 410.6668 --- Search : 366.7741 --- Flee : 342.2086 --- ### F = 43.89
1111 :: GetFood : 367.4231 --- Search : 238.7783 --- Flee : 218.6462 --- ### F = 128.64
1112 :: Search : 380.7953 --- GetFood : 263.614 --- Flee : 242.2081 --- ### F = 117.18
1113 :: GetFood : 378.7502 --- Search : 192.6964 --- Flee : 120.1025 --- ### F = 186.05
1120 :: GetFood : 401.2755 --- Search : 383.1865 --- Flee : 350.7429 --- ### F = 18.09
1121 :: GetFood : 371.9836 --- Search : 349.8612 --- Flee : 289.845 --- ### F = 22.12
1122 :: GetFood : 362.6861 --- Flee : 295.954 --- Search : 285.6378 --- ### F = 66.73
1123 :: GetFood : 355.1264 --- Search : 150.7285 --- Flee : 124.0371 --- ### F = 204.40
1130 :: Flee : 408.1174 --- Search : 385.0528 --- GetFood : 376.2338 --- ### F = 23.06
1131 :: GetFood : 374.7838 --- Flee : 231.4781 --- Search : 148.1888 --- ### F = 143.31
1132 :: GetFood : 341.9267 --- Flee : 178.8627 --- Search : 176.6861 --- ### F = 163.06
1133 :: GetFood : 347.1285 --- Search : 41.76891 --- Flee : 41.52378 --- ### F = 305.36
1200 :: GetFood : 367.6472 --- Search : 302.3455 --- Flee : 208.6368 --- ### F = 65.30
1201 :: GetFood : 395.0636 --- Search : 332.2296 --- Flee : 91.19788 --- ### F = 62.83
1202 :: GetFood : 394.8707 --- Search : 320.9801 --- Flee : 31.07145 --- ### F = 73.89
1203 :: GetFood : 411.8607 --- Search : 300.0818 --- Flee : 112.8441 --- ### F = 111.78
1210 :: Search : 398.6452 --- GetFood : 357.8165 --- Flee : 348.5683 --- ### F = 40.83
1211 :: GetFood : 303.8906 --- Flee : 264.5417 --- Search : 165.478 --- ### F = 39.35
1212 :: GetFood : 391.1035 --- Search : 330.4518 --- Flee : 306.0111 --- ### F = 60.65
1213 :: Flee : 385.0592 --- GetFood : 159.7243 --- Search : 122.8963 --- ### F = 225.33
1220 :: Flee : 364.3105 --- GetFood : 362.1278 --- Search : 355.434 --- ### F = 2.18
1221 :: GetFood : 385.7865 --- Flee : 315.5934 --- Search : 238.9744 --- ### F = 70.19
1222 :: Flee : 390.1239 --- GetFood : 367.5782 --- Search : 341.9985 --- ### F = 22.55
1223 :: GetFood : 386.3775 --- Search : 266.2544 --- Flee : 146.1637 --- ### F = 120.12
1230 :: Search : 403.1772 --- Flee : 372.1822 --- GetFood : 368.0725 --- ### F = 31.00
1231 :: GetFood : 382.0698 --- Search : 278.1501 --- Flee : 228.7517 --- ### F = 103.92
1232 :: Flee : 387.6239 --- Search : 295.7699 --- GetFood : 170.5477 --- ### F = 91.85
1233 :: GetFood : 372.7401 --- Flee : 157.0881 --- Search : 91.9814 --- ### F = 215.65
1300 :: GetFood : 409.0806 --- Search : 295.7209 --- Flee : 130.6461 --- ### F = 113.36
1301 :: GetFood : 394.747 --- Search : 335.7157 --- Flee : 74.55482 --- ### F = 59.03
1302 :: GetFood : 364.5919 --- Search : 343.9262 --- Flee : 145.2469 --- ### F = 20.67
1303 :: GetFood : 415.5237 --- Search : 339.7341 --- Flee : 76.99896 --- ### F = 75.79
1310 :: GetFood : 419.5699 --- Search : 346.2994 --- Flee : 340.3509 --- ### F = 73.27
1311 :: Search : 382.0869 --- GetFood : 229.1596 --- Flee : 135.3105 --- ### F = 152.93
1312 :: GetFood : 414.684 --- Flee : 235.5762 --- Search : 173.5813 --- ### F = 179.11
1313 :: GetFood : 417.6284 --- Flee : 141.5298 --- Search : 133.0614 --- ### F = 276.10
1320 :: GetFood : 421.2016 --- Search : 381.7028 --- Flee : 381.3022 --- ### F = 39.50
1321 :: Search : 381.1617 --- GetFood : 319.7399 --- Flee : 266.4756 --- ### F = 61.42
1322 :: GetFood : 354.4002 --- Flee : 249.497 --- Search : 188.5945 --- ### F = 104.90
1323 :: GetFood : 375.7108 --- Flee : 256.8661 --- Search : 251.9142 --- ### F = 118.84
1330 :: Flee : 417.8731 --- GetFood : 369.5386 --- Search : 357.441 --- ### F = 48.33
1331 :: GetFood : 407.0674 --- Search : 223.6551 --- Flee : 173.2142 --- ### F = 183.41
1332 :: GetFood : 380.5991 --- Flee : 169.9653 --- Search : 138.1375 --- ### F = 210.63
1333 :: GetFood : 401.055 --- Flee : 212.562 --- Search : 175.8355 --- ### F = 188.49
2000 :: GetFood : 624.2587 --- Search : 554.5581 --- Flee : 523.6235 --- ### F = 69.70
2001 :: Search : 595.1813 --- Flee : 583.1152 --- GetFood : 552.6971 --- ### F = 12.07
2002 :: Search : 613.3079 --- Flee : 550.0271 --- GetFood : 440.0892 --- ### F = 63.28
2003 :: Search : 599.7221 --- Flee : 505.1119 --- GetFood : 461.4718 --- ### F = 94.61
2010 :: GetFood : 637.4601 --- Search : 604.5548 --- Flee : 600.9202 --- ### F = 32.91
2011 :: Search : 616.7015 --- Flee : 567.1905 --- GetFood : 491.1083 --- ### F = 49.51
2012 :: Flee : 606.7726 --- Search : 574.2184 --- GetFood : 443.925 --- ### F = 32.55
2013 :: Search : 595.9675 --- Flee : 527.2646 --- GetFood : 302.0045 --- ### F = 68.70
2020 :: Search : 646.4199 --- Flee : 606.825 --- GetFood : 588.0436 --- ### F = 39.59
2021 :: Search : 638.1871 --- Flee : 575.233 --- GetFood : 523.0101 --- ### F = 62.95

2022 :: Flee : 608.3479 --- Search : 564.5928 --- GetFood : 547.9089 --- ### F = 43.76
2023 :: Flee : 564.4157 --- Search : 560.7734 --- GetFood : 333.5269 --- ### F = 3.64
2030 :: Search : 637.0021 --- Flee : 570.0624 --- GetFood : 563.2758 --- ### F = 66.94
2031 :: Flee : 628.683 --- Search : 579.192 --- GetFood : 321.6413 --- ### F = 49.49
2032 :: Flee : 619.5232 --- Search : 564.5164 --- GetFood : 299.0457 --- ### F = 55.01
2033 :: Search : 553.6935 --- Flee : 421.4183 --- GetFood : 230.3663 --- ### F = 132.28
2100 :: Search : 581.7008 --- GetFood : 473.2309 --- Flee : 199.3679 --- ### F = 108.47
2101 :: GetFood : 540.4061 --- Search : 286.4034 --- Flee : 139.0886 --- ### F = 254.00
2102 :: GetFood : 521.1207 --- Search : 131.5227 --- Flee : 85.17423 --- ### F = 389.60
2103 :: GetFood : 513.5322 --- Search : 149.8533 --- Flee : 0 --- ### F = 363.68
2110 :: GetFood : 568.2894 --- Search : 467.2669 --- Flee : 449.3871 --- ### F = 101.02
2111 :: GetFood : 482.8531 --- Flee : 166.2985 --- Search : 142.2274 --- ### F = 316.55
2112 :: GetFood : 514.7574 --- Search : 248.0382 --- Flee : 104.2907 --- ### F = 266.72
2113 :: GetFood : 469.0626 --- Flee : 91.94101 --- Search : 59.4602 --- ### F = 377.12
2120 :: Flee : 573.3975 --- Search : 494.2429 --- GetFood : 491.7103 --- ### F = 79.15
2121 :: GetFood : 541.639 --- Search : 283.0046 --- Flee : 174.4266 --- ### F = 258.63
2122 :: GetFood : 517.4149 --- Flee : 168.1653 --- Search : 58.06083 --- ### F = 349.25
2123 :: GetFood : 484.06 --- Search : 112.7509 --- Flee : 56.92928 --- ### F = 371.31
2130 :: GetFood : 563.3736 --- Flee : 426.5674 --- Search : 331.809 --- ### F = 136.81
2131 :: Search : 539.9806 --- GetFood : 164.038 --- Flee : 116.4696 --- ### F = 375.94
2132 :: GetFood : 530.8609 --- Flee : 222.2144 --- Search : 165.9832 --- ### F = 308.65
2133 :: GetFood : 506.0919 --- Search : 191.1198 --- Flee : 124.5709 --- ### F = 314.97
2200 :: GetFood : 549.3781 --- Search : 500.2416 --- Flee : 277.7586 --- ### F = 49.14
2201 :: GetFood : 557.8718 --- Search : 416.3615 --- Flee : 0 --- ### F = 141.51
2202 :: GetFood : 555.8188 --- Search : 427.4954 --- Flee : 2.908391 --- ### F = 128.32
2203 :: GetFood : 501.2462 --- Search : 210.9052 --- Flee : 56.47057 --- ### F = 290.34
2210 :: Flee : 551.5403 --- GetFood : 493.2812 --- Search : 468.7856 --- ### F = 58.26
2211 :: GetFood : 535.2707 --- Flee : 166.5725 --- Search : 152.566 --- ### F = 368.70
2212 :: GetFood : 542.9139 --- Flee : 241.5171 --- Search : 149.6806 --- ### F = 301.40
2213 :: Flee : 510.9534 --- Search : 194.8811 --- GetFood : 8.649668 --- ### F = 316.07
2220 :: GetFood : 559.6757 --- Search : 457.3696 --- Flee : 403.2913 --- ### F = 102.31
2221 :: GetFood : 536.1545 --- Flee : 287.5442 --- Search : 183.4742 --- ### F = 248.61
2222 :: GetFood : 542.3109 --- Flee : 412.1371 --- Search : 304.1551 --- ### F = 130.17
2223 :: Search : 509.8629 --- Flee : 305.24 --- GetFood : 260.4227 --- ### F = 204.62
2230 :: GetFood : 553.623 --- Flee : 452.1865 --- Search : 389.7465 --- ### F = 101.44
2231 :: GetFood : 516.66 --- Search : 82.82858 --- Flee : 0 --- ### F = 433.83
2232 :: Flee : 535.6907 --- GetFood : 307.3366 --- Search : 288.5838 --- ### F = 228.35
2233 :: Search : 469.179 --- GetFood : 186.1275 --- Flee : 182.1139 --- ### F = 283.05
2300 :: GetFood : 578.1304 --- Search : 482.7605 --- Flee : 232.7641 --- ### F = 95.37
2301 :: GetFood : 562.8048 --- Search : 489.5073 --- Flee : 0 --- ### F = 73.30
2302 :: GetFood : 577.3409 --- Search : 202.9829 --- Flee : 97.72883 --- ### F = 374.36
2303 :: GetFood : 560.2493 --- Search : 211.713 --- Flee : 0 --- ### F = 348.54
2310 :: GetFood : 542.2371 --- Search : 355.7136 --- Flee : 270.9176 --- ### F = 186.52
2311 :: GetFood : 572.4917 --- Search : 93.23929 --- Flee : 0 --- ### F = 479.25
2312 :: GetFood : 562.191 --- Search : 121.5206 --- Flee : 107.9453 --- ### F = 440.67
2313 :: GetFood : 573.2365 --- Search : 0 --- Search : 0 --- ### F = 573.24
2320 :: GetFood : 593.9312 --- Flee : 517.7893 --- Search : 508.7797 --- ### F = 76.14
2321 :: GetFood : 572.3752 --- Search : 185.6445 --- Flee : 71.13696 --- ### F = 386.73
2322 :: GetFood : 528.948 --- Search : 247.6254 --- Flee : 158.2382 --- ### F = 281.32
2323 :: GetFood : 557.1848 --- Flee : 223.9506 --- Search : 223.6958 --- ### F = 333.23
2330 :: Flee : 573.4584 --- GetFood : 503.2586 --- Search : 501.6144 --- ### F = 70.20
2331 :: GetFood : 564.7976 --- Flee : 248.4331 --- Search : 56.54542 --- ### F = 316.36
2332 :: GetFood : 567.1957 --- Flee : 181.8614 --- Search : 161.5907 --- ### F = 385.33
2333 :: GetFood : 543.9799 --- Flee : 102.5679 --- Search : 51.23581 --- ### F = 441.41
3000 :: Search : 883.4476 --- GetFood : 830.5949 --- Flee : 788.5438 --- ### F = 52.85
3001 :: Search : 890.2482 --- Flee : 867.973 --- GetFood : 851.3511 --- ### F = 22.28
3002 :: Flee : 889.6651 --- GetFood : 851.9407 --- Search : 851.354 --- ### F = 37.72
3003 :: Search : 892.0158 --- GetFood : 714.6619 --- Flee : 656.2188 --- ### F = 177.35

3010 :: Search : 919.0258 --- GetFood : 894.6717 --- Flee : 887.6974 --- ### F = 24.35
3011 :: Search : 909.7937 --- Flee : 876.7662 --- GetFood : 864.4269 --- ### F = 33.03
3012 :: Search : 903.9445 --- Flee : 875.0416 --- GetFood : 852.3389 --- ### F = 28.90
3013 :: Search : 850.4387 --- Flee : 835.9765 --- GetFood : 830.4774 --- ### F = 14.46
3020 :: Flee : 912.911 --- GetFood : 897.2678 --- Search : 894.9764 --- ### F = 15.64
3021 :: Search : 903.981 --- Flee : 888.8898 --- GetFood : 814.5061 --- ### F = 15.09
3022 :: Flee : 907.4611 --- Search : 869.8149 --- GetFood : 838.0294 --- ### F = 37.65
3023 :: Flee : 894.0165 --- Search : 830.2756 --- GetFood : 724.2388 --- ### F = 63.74
3030 :: Flee : 911.7467 --- GetFood : 867.0475 --- Search : 862.9394 --- ### F = 44.70
3031 :: Search : 901.7274 --- Flee : 843.468 --- GetFood : 627.2207 --- ### F = 58.26
3032 :: Search : 888.5911 --- Flee : 842.1132 --- GetFood : 680.8818 --- ### F = 46.48
3033 :: Search : 872.7261 --- Flee : 718.5526 --- GetFood : 324.5664 --- ### F = 154.17
3100 :: GetFood : 852.6887 --- Search : 810.6321 --- Flee : 390.4254 --- ### F = 42.06
3101 :: Search : 872.0189 --- GetFood : 759.7095 --- Flee : 241.9877 --- ### F = 112.31
3102 :: Search : 846.5273 --- GetFood : 770.8082 --- Flee : 355.6323 --- ### F = 75.72
3103 :: GetFood : 797.2773 --- Search : 450.4236 --- Flee : 394.0544 --- ### F = 346.85
3110 :: GetFood : 876.7971 --- Flee : 845.9648 --- Search : 794.4919 --- ### F = 30.83
3111 :: GetFood : 854.3307 --- Flee : 643.8903 --- Search : 557.3774 --- ### F = 210.44
3112 :: GetFood : 812.6708 --- Flee : 485.107 --- Search : 408.7365 --- ### F = 327.56
3113 :: Flee : 810.829 --- Search : 388.8719 --- GetFood : 253.1391 --- ### F = 421.96
3120 :: Flee : 903.718 --- Search : 853.5847 --- GetFood : 847.8734 --- ### F = 50.13
3121 :: GetFood : 845.3744 --- Search : 591.6348 --- Flee : 455.04 --- ### F = 253.74
3122 :: GetFood : 823.231 --- Flee : 418.9142 --- Search : 372.3594 --- ### F = 404.32
3123 :: GetFood : 805.2582 --- Flee : 363.5122 --- Search : 265.1472 --- ### F = 441.75
3130 :: Flee : 907.8232 --- Search : 849.4351 --- GetFood : 778.806 --- ### F = 58.39
3131 :: GetFood : 797.4667 --- Flee : 573.5258 --- Search : 374.068 --- ### F = 223.94
3132 :: GetFood : 782.5999 --- Flee : 442.0612 --- Search : 61.28492 --- ### F = 340.54
3133 :: GetFood : 747.6069 --- Flee : 293.3836 --- Search : 170.1331 --- ### F = 454.22
3200 :: Search : 828.7202 --- GetFood : 765.4115 --- Flee : 522.9572 --- ### F = 63.31
3201 :: Search : 822.6288 --- GetFood : 759.686 --- Flee : 141.1123 --- ### F = 62.94
3202 :: GetFood : 832.0201 --- Search : 655.3051 --- Flee : 76.534 --- ### F = 176.71
3203 :: GetFood : 814.6638 --- Search : 477.4548 --- Flee : 138.863 --- ### F = 337.21
3210 :: GetFood : 879.2403 --- Search : 828.1778 --- Flee : 811.0156 --- ### F = 51.06
3211 :: GetFood : 761.8619 --- Flee : 658.9205 --- Search : 497.0631 --- ### F = 102.94
3212 :: GetFood : 822.2244 --- Search : 596.743 --- Flee : 551.4569 --- ### F = 225.48
3213 :: GetFood : 844.5024 --- Search : 217.29 --- Flee : 97.46698 --- ### F = 627.21
3220 :: Flee : 890.7592 --- GetFood : 863.1976 --- Search : 858.8035 --- ### F = 27.56
3221 :: GetFood : 820.6613 --- Search : 507.0254 --- Flee : 490.241 --- ### F = 313.64
3222 :: GetFood : 848.0522 --- Search : 601.6143 --- Flee : 502.5114 --- ### F = 246.44
3223 :: GetFood : 827.7344 --- Flee : 330.4744 --- Search : 329.65 --- ### F = 497.26
3230 :: GetFood : 895.2264 --- Flee : 857.783 --- Search : 842.4412 --- ### F = 37.44
3231 :: GetFood : 838.5887 --- Search : 624.0637 --- Flee : 489.4661 --- ### F = 214.52
3232 :: GetFood : 833.8545 --- Flee : 503.3544 --- Search : 359.5835 --- ### F = 330.50
3233 :: GetFood : 827.0735 --- Flee : 486.0686 --- Search : 377.2144 --- ### F = 341.00
3300 :: GetFood : 874.8489 --- Search : 811.7193 --- Flee : 299.7139 --- ### F = 63.13
3301 :: GetFood : 875.5035 --- Search : 763.4509 --- Flee : 284.3194 --- ### F = 112.05
3302 :: GetFood : 855.0672 --- Search : 636.5231 --- Flee : 147.7897 --- ### F = 218.54
3303 :: GetFood : 874.9932 --- Search : 673.0272 --- Flee : 83.70952 --- ### F = 201.97
3310 :: GetFood : 887.8547 --- Search : 827.1988 --- Flee : 812.4969 --- ### F = 60.66
3311 :: Flee : 817.8864 --- Search : 546.1307 --- GetFood : 419.8 --- ### F = 271.76
3312 :: GetFood : 852.1449 --- Flee : 594.9869 --- Search : 522.913 --- ### F = 257.16
3313 :: GetFood : 864.9803 --- Search : 557.0828 --- Flee : 516.429 --- ### F = 307.90
3320 :: GetFood : 908.8582 --- Search : 860.0905 --- Flee : 855.5051 --- ### F = 48.77
3321 :: GetFood : 831.072 --- Search : 541.429 --- Flee : 464.2608 --- ### F = 289.64
3322 :: Flee : 815.4852 --- GetFood : 765.6981 --- Search : 672.1049 --- ### F = 49.79
3323 :: Flee : 856.1265 --- GetFood : 746.5079 --- Search : 666.4738 --- ### F = 109.62
3330 :: Flee : 902.9284 --- GetFood : 873.409 --- Search : 833.0674 --- ### F = 29.52
3331 :: Search : 855.9854 --- GetFood : 626.078 --- Flee : 339.7191 --- ### F = 229.91

3332 :: GetFood : 829.5944 --- Search : 473.6772 --- Flee : 433.9673 --- ### F = 355.92
3333 :: GetFood : 844.9758 --- Search : 556.134 --- Flee : 466.004 --- ### F = 288.84

Experiment 2, Predator

TOTAL SIZE : 208

0000 :: Search : 123.7664 --- Fight : 4.735794 --- GetFood : 0 --- ### F = 119.03
0001 :: Search : 109.531 --- GetFood : 0 --- GetFood : 0 --- ### F = 109.53
0002 :: Search : 108.5151 --- Fight : 53.84974 --- GetFood : 12.42868 --- ### F = 54.67
0003 :: GetFood : 74.86867 --- Search : 65.38679 --- Fight : 0 --- ### F = 9.48
0010 :: Search : 17.99293 --- GetFood : 0 --- GetFood : 0 --- ### F = 17.99
0012 :: Fight : 60.22528 --- Search : 5.378207 --- GetFood : 0 --- ### F = 54.85
0013 :: Fight : 20.80301 --- Search : 0 --- Search : 0 --- ### F = 20.80
0020 :: Fight : 37.56985 --- Search : 0 --- Search : 0 --- ### F = 37.57
0021 :: Search : 42.53068 --- Fight : 4.991496 --- GetFood : 0 --- ### F = 37.54
0022 :: Search : 97.5994 --- GetFood : 4.979097 --- Fight : 0 --- ### F = 92.62
0023 :: GetFood : 37.82624 --- Search : 10.17206 --- Fight : 9.885984 --- ### F = 27.65
0030 :: Search : 5.906847 --- GetFood : 0 --- GetFood : 0 --- ### F = 5.91
0031 :: Fight : 19.9465 --- Search : 0 --- Search : 0 --- ### F = 19.95
0032 :: Fight : 58.28767 --- Search : 4.979397 --- GetFood : 0 --- ### F = 53.31
0033 :: Search : 64.22591 --- GetFood : 38.70033 --- Fight : 9.779606 --- ### F = 25.53
0100 :: GetFood : 4.737763 --- Search : 0 --- Search : 0 --- ### F = 4.74
0200 :: GetFood : 4.720773 --- Search : 0 --- Search : 0 --- ### F = 4.72
0203 :: Search : 4.979241 --- GetFood : 4.971446 --- ### F = 0.01
0300 :: GetFood : 11.05647 --- Search : 0 --- Search : 0 --- ### F = 11.06
0302 :: GetFood : 4.611 --- Search : 0 --- Search : 0 --- ### F = 4.61
0303 :: GetFood : 6.2932 --- Search : 0 --- Search : 0 --- ### F = 6.29
0322 :: GetFood : 7.098159 --- Search : 0 --- Search : 0 --- ### F = 7.10
0330 :: GetFood : 5.235285 --- Search : 0 --- Search : 0 --- ### F = 5.24
0332 :: GetFood : 3.676306 --- Search : 0 --- Search : 0 --- ### F = 3.68
0333 :: GetFood : 10.29579 --- Search : 0 --- Search : 0 --- ### F = 10.30
1000 :: Search : 593.5673 --- GetFood : 434.7494 --- Fight : 416.0707 --- ### F = 158.82
1001 :: Search : 571.692 --- GetFood : 143.5966 --- Fight : 112.3047 --- ### F = 428.10
1002 :: Search : 569.483 --- Fight : 304.4828 --- GetFood : 0 --- ### F = 265.00
1003 :: Search : 586.6212 --- Fight : 210.7294 --- GetFood : 95.856 --- ### F = 375.89
1010 :: Fight : 697.3644 --- Search : 544.6411 --- GetFood : 190.062 --- ### F = 152.72
1011 :: Search : 626.7846 --- Fight : 366.0525 --- GetFood : 135.2228 --- ### F = 260.73
1012 :: Fight : 686.2851 --- Search : 357.1532 --- GetFood : 0 --- ### F = 329.13
1013 :: Search : 699.735 --- Fight : 269.0194 --- GetFood : 60.70551 --- ### F = 430.72
1020 :: Fight : 750.374 --- Search : 657.8948 --- GetFood : 500.0365 --- ### F = 92.48
1021 :: Fight : 641.8987 --- Search : 580.4487 --- GetFood : 41.03767 --- ### F = 61.45
1022 :: Search : 705.4992 --- Fight : 629.7577 --- GetFood : 145.5963 --- ### F = 75.74
1023 :: Fight : 696.575 --- Search : 637.788 --- GetFood : 131.1145 --- ### F = 58.79
1030 :: Search : 712.4153 --- Fight : 708.3566 --- GetFood : 697.9444 --- ### F = 4.06
1031 :: Search : 664.8051 --- Fight : 524.2526 --- GetFood : 385.5746 --- ### F = 140.55
1032 :: Search : 692.1246 --- Fight : 503.6339 --- GetFood : 413.5763 --- ### F = 188.49
1033 :: Fight : 698.4496 --- Search : 617.6286 --- GetFood : 458.2782 --- ### F = 80.82
1100 :: GetFood : 132.501 --- Search : 0 --- Search : 0 --- ### F = 132.50
1101 :: GetFood : 200.8222 --- Search : 0 --- Search : 0 --- ### F = 200.82
1102 :: GetFood : 241.4488 --- Search : 0 --- Search : 0 --- ### F = 241.45
1110 :: GetFood : 40.41218 --- Search : 0 --- Search : 0 --- ### F = 40.41
1111 :: GetFood : 86.55866 --- Search : 0 --- Search : 0 --- ### F = 86.56

1112 :: GetFood : 30.27708 --- Search : 0 --- Search : 0 --- ### F = 30.28
1113 :: GetFood : 10.4862 --- Search : 0 --- Search : 0 --- ### F = 10.49
1120 :: Fight : 92.67607 --- GetFood : 8.433162 --- Search : 0 --- ### F = 84.24
1121 :: GetFood : 251.5655 --- Search : 0 --- Search : 0 --- ### F = 251.57
1123 :: GetFood : 62.50968 --- Search : 0 --- Search : 0 --- ### F = 62.51
1130 :: GetFood : 30.81355 --- Search : 0 --- Search : 0 --- ### F = 30.81
1131 :: GetFood : 296.5627 --- Search : 0 --- Search : 0 --- ### F = 296.56
1133 :: GetFood : 146.1107 --- Search : 0 --- Search : 0 --- ### F = 146.11
1200 :: GetFood : 391.5038 --- Search : 0 --- Search : 0 --- ### F = 391.50
1201 :: GetFood : 106.3475 --- Search : 0 --- Search : 0 --- ### F = 106.35
1202 :: GetFood : 463.8088 --- Fight : 52.09239 --- Search : 0 --- ### F = 411.72
1203 :: GetFood : 552.3368 --- Search : 42.69621 --- Fight : 0 --- ### F = 509.64
1210 :: GetFood : 130.5679 --- Fight : 9.184556 --- Search : 0 --- ### F = 121.38
1211 :: GetFood : 94.93235 --- Search : 0 --- Search : 0 --- ### F = 94.93
1212 :: GetFood : 289.504 --- Fight : 30.9982 --- Search : 0 --- ### F = 258.51
1213 :: Search : 198.3844 --- GetFood : 10.49586 --- Fight : 0 --- ### F = 187.89
1220 :: GetFood : 77.96557 --- Fight : 11.8821 --- Search : 0 --- ### F = 66.08
1221 :: GetFood : 20.81385 --- Search : 0 --- Search : 0 --- ### F = 20.81
1222 :: GetFood : 105.3744 --- Search : 0 --- Search : 0 --- ### F = 105.37
1223 :: GetFood : 175.1214 --- Search : 0 --- Search : 0 --- ### F = 175.12
1230 :: GetFood : 91.93062 --- Search : 0 --- Search : 0 --- ### F = 91.93
1232 :: GetFood : 156.0181 --- Search : 0 --- Search : 0 --- ### F = 156.02
1233 :: GetFood : 210.6367 --- Search : 0 --- Search : 0 --- ### F = 210.64
1300 :: GetFood : 636.137 --- Search : 69.73306 --- Fight : 0 --- ### F = 566.40
1301 :: GetFood : 279.4827 --- Search : 0 --- Search : 0 --- ### F = 279.48
1302 :: GetFood : 468.0918 --- Search : 115.9738 --- Fight : 0 --- ### F = 352.12
1303 :: Search : 554.4175 --- GetFood : 289.6693 --- Fight : 81.10085 --- ### F = 264.75
1310 :: GetFood : 566.1059 --- Fight : 72.68613 --- Search : 9.693366 --- ### F = 493.42
1311 :: GetFood : 156.4259 --- Search : 0 --- Search : 0 --- ### F = 156.43
1312 :: Fight : 398.0958 --- GetFood : 96.69836 --- Search : 57.49237 --- ### F = 301.40
1313 :: GetFood : 566.2855 --- Search : 127.7582 --- Fight : 99.95739 --- ### F = 438.53
1320 :: GetFood : 447.9552 --- Search : 0 --- Search : 0 --- ### F = 447.96
1321 :: GetFood : 138.3833 --- Search : 0 --- Search : 0 --- ### F = 138.38
1322 :: GetFood : 204.6492 --- Fight : 25.0073 --- Search : 24.85761 --- ### F = 179.64
1323 :: Search : 545.7039 --- GetFood : 210.6403 --- Fight : 62.01367 --- ### F = 335.06
1330 :: GetFood : 177.0216 --- Search : 0 --- Search : 0 --- ### F = 177.02
1331 :: GetFood : 96.35671 --- Search : 0 --- Search : 0 --- ### F = 96.36
1332 :: GetFood : 178.468 --- Fight : 9.888362 --- Search : 0 --- ### F = 168.58
1333 :: Fight : 421.7042 --- Search : 108.2243 --- GetFood : 43.8889 --- ### F = 313.48
2000 :: Search : 1063.993 --- Fight : 526.3864 --- GetFood : 251.6682 --- ### F = 537.61
2001 :: Search : 899.2629 --- Fight : 445.4574 --- GetFood : 358.7269 --- ### F = 453.81
2002 :: Search : 882.0396 --- GetFood : 307.5345 --- Fight : 255.8686 --- ### F = 574.51
2003 :: Search : 943.8842 --- Fight : 463.7631 --- GetFood : 383.5329 --- ### F = 480.12
2010 :: Fight : 1058.65 --- Search : 908.4358 --- GetFood : 431.4298 --- ### F = 150.21
2011 :: Search : 934.4836 --- Fight : 793.5319 --- GetFood : 185.5219 --- ### F = 140.95
2012 :: Search : 969.1678 --- Fight : 703.6196 --- GetFood : 110.7098 --- ### F = 265.55
2013 :: Fight : 913.0655 --- Search : 605.522 --- GetFood : 46.66436 --- ### F = 307.54
2020 :: Fight : 1184.572 --- GetFood : 1045.295 --- Search : 1026.043 --- ### F = 139.28
2021 :: Search : 961.8152 --- Fight : 830.6615 --- GetFood : 294.7834 --- ### F = 131.15
2022 :: Fight : 1053.802 --- Search : 671.7295 --- GetFood : 317.0298 --- ### F = 382.07
2023 :: Fight : 1068.526 --- Search : 992.7766 --- GetFood : 494.7574 --- ### F = 75.75
2030 :: GetFood : 1196.19 --- Search : 1072.459 --- Fight : 1058.8 --- ### F = 123.73
2031 :: Fight : 941.2366 --- Search : 571.9232 --- GetFood : 0 --- ### F = 369.31
2032 :: Fight : 1075.786 --- Search : 725.7516 --- GetFood : 135.9669 --- ### F = 350.03
2033 :: Search : 1140.273 --- Fight : 1003.843 --- GetFood : 799.6873 --- ### F = 136.43
2100 :: GetFood : 173.1898 --- Search : 12.09037 --- Fight : 0 --- ### F = 161.10
2101 :: GetFood : 509.4061 --- Search : 0 --- Search : 0 --- ### F = 509.41
2102 :: GetFood : 206.6247 --- Search : 16.10314 --- Fight : 0 --- ### F = 190.52

2103 :: GetFood : 99.13619 --- Search : 0 --- Search : 0 --- ### F = 99.14
2110 :: Fight : 318.41 --- GetFood : 24.63465 --- Search : 0 --- ### F = 293.78
2111 :: GetFood : 463.0922 --- Search : 0 --- Search : 0 --- ### F = 463.09
2112 :: GetFood : 99.46234 --- Search : 0 --- Search : 0 --- ### F = 99.46
2120 :: GetFood : 240.3893 --- Search : 26.57374 --- Fight : 16.59341 --- ### F = 213.82
2121 :: GetFood : 434.1856 --- Fight : 95.76872 --- Search : 0 --- ### F = 338.42
2122 :: Fight : 303.3982 --- GetFood : 15.5564 --- Search : 0 --- ### F = 287.84
2130 :: Search : 199.2907 --- GetFood : 0 --- GetFood : 0 --- ### F = 199.29
2131 :: GetFood : 134.3367 --- Search : 0 --- Search : 0 --- ### F = 134.34
2132 :: GetFood : 288.2424 --- Search : 0 --- Search : 0 --- ### F = 288.24
2133 :: GetFood : 280.9471 --- Search : 0 --- Search : 0 --- ### F = 280.95
2200 :: GetFood : 359.6081 --- Search : 0 --- Search : 0 --- ### F = 359.61
2201 :: Search : 116.0665 --- GetFood : 56.27738 --- Fight : 0 --- ### F = 59.79
2202 :: GetFood : 595.4685 --- Search : 52.34286 --- Fight : 0 --- ### F = 543.13
2203 :: Search : 435.7528 --- GetFood : 46.96478 --- Fight : 0 --- ### F = 388.79
2210 :: GetFood : 185.7032 --- Fight : 25.91629 --- Search : 0 --- ### F = 159.79
2212 :: Fight : 116.5707 --- GetFood : 52.09564 --- Search : 0 --- ### F = 64.48
2213 :: Search : 271.6823 --- GetFood : 0 --- GetFood : 0 --- ### F = 271.68
2220 :: GetFood : 154.9623 --- Search : 0 --- Search : 0 --- ### F = 154.96
2222 :: GetFood : 209.7934 --- Search : 88.55923 --- Fight : 17.50694 --- ### F = 121.23
2223 :: Search : 428.7239 --- GetFood : 0 --- GetFood : 0 --- ### F = 428.72
2230 :: Search : 19.55744 --- ### F = 19.56
2232 :: GetFood : 258.0049 --- Search : 0 --- Search : 0 --- ### F = 258.00
2233 :: Search : 161.1576 --- GetFood : 0 --- GetFood : 0 --- ### F = 161.16
2300 :: GetFood : 958.2222 --- Search : 36.01936 --- Fight : 0 --- ### F = 922.20
2301 :: GetFood : 535.4435 --- Search : 58.15261 --- Fight : 0 --- ### F = 477.29
2302 :: GetFood : 675.0434 --- Search : 24.847 --- Fight : 0 --- ### F = 650.20
2303 :: GetFood : 856.4442 --- Search : 181.8918 --- Fight : 81.80735 --- ### F = 674.55
2310 :: GetFood : 617.3358 --- Fight : 70.63921 --- Search : 0 --- ### F = 546.70
2311 :: GetFood : 281.6337 --- Search : 0 --- Search : 0 --- ### F = 281.63
2312 :: Search : 431.5286 --- Fight : 15.17231 --- GetFood : 13.64271 --- ### F = 416.36
2313 :: GetFood : 837.7651 --- Search : 192.0537 --- Fight : 166.0175 --- ### F = 645.71
2320 :: GetFood : 511.7909 --- Search : 0 --- Search : 0 --- ### F = 511.79
2321 :: GetFood : 223.3406 --- Search : 44.17953 --- Fight : 30.70793 --- ### F = 179.16
2322 :: GetFood : 438.0403 --- Search : 42.65719 --- Fight : 0 --- ### F = 395.38
2323 :: GetFood : 749.4269 --- Search : 102.7308 --- Fight : 0 --- ### F = 646.70
2330 :: GetFood : 71.1343 --- Search : 0 --- Search : 0 --- ### F = 71.13
2331 :: GetFood : 271.7822 --- Fight : 15.95312 --- Search : 0 --- ### F = 255.83
2332 :: GetFood : 383.1021 --- Search : 0 --- Search : 0 --- ### F = 383.10
2333 :: GetFood : 491.5636 --- Search : 0 --- Search : 0 --- ### F = 491.56
3000 :: Search : 1953.107 --- GetFood : 1946.556 --- Fight : 1944.811 --- ### F = 6.55
3001 :: Fight : 1952.726 --- Search : 1950.63 --- GetFood : 1935.012 --- ### F = 2.10
3002 :: GetFood : 1951.038 --- Search : 1947.705 --- Fight : 1891.578 --- ### F = 3.33
3003 :: Fight : 1948.925 --- Search : 1944.264 --- GetFood : 1941.564 --- ### F = 4.66
3010 :: Fight : 1959.506 --- GetFood : 1948.699 --- Search : 1946.768 --- ### F = 10.81
3011 :: Search : 1950.976 --- Fight : 1948.366 --- GetFood : 1922.938 --- ### F = 2.61
3012 :: Search : 1949.387 --- Fight : 1946.83 --- GetFood : 1944.592 --- ### F = 2.56
3013 :: Fight : 1948.453 --- GetFood : 1947.813 --- Search : 1947.748 --- ### F = 0.64
3020 :: GetFood : 1959.675 --- Fight : 1951.405 --- Search : 1951.283 --- ### F = 8.27
3021 :: GetFood : 1947.978 --- Fight : 1947.714 --- Search : 1934.651 --- ### F = 0.26
3022 :: Fight : 1949.231 --- Search : 1942.67 --- GetFood : 1937.876 --- ### F = 6.56
3023 :: Fight : 1948.513 --- GetFood : 1929.574 --- Search : 1921.492 --- ### F = 18.94
3030 :: Fight : 1952.945 --- GetFood : 1948.518 --- Search : 1947.931 --- ### F = 4.43
3031 :: GetFood : 1949.465 --- Search : 1948.465 --- Fight : 1948.102 --- ### F = 1.00
3032 :: Fight : 1949.724 --- Search : 1944.475 --- GetFood : 1934.026 --- ### F = 5.25
3033 :: Fight : 1950.215 --- Search : 1945.401 --- GetFood : 1911.333 --- ### F = 4.81
3100 :: GetFood : 1810.253 --- Search : 900.4374 --- Fight : 309.4564 --- ### F = 909.82
3101 :: GetFood : 1800.821 --- Search : 980.5211 --- Fight : 317.7884 --- ### F = 820.30

3102 :: GetFood : 1772.575 --- Search : 364.9373 --- Fight : 166.4805 --- ### F = 1407.64
3103 :: GetFood : 1671.128 --- Search : 611.7633 --- Fight : 185.053 --- ### F = 1059.36
3110 :: GetFood : 1866.528 --- Fight : 1626.421 --- Search : 1355.832 --- ### F = 240.11
3111 :: GetFood : 1826.81 --- Search : 987.3522 --- Fight : 650.4532 --- ### F = 839.46
3112 :: Fight : 1834.591 --- Search : 654.7202 --- GetFood : 507.0669 --- ### F = 1179.87
3113 :: GetFood : 1788.738 --- Fight : 501.764 --- Search : 172.2272 --- ### F = 1286.97
3120 :: Fight : 1908.058 --- GetFood : 1585.937 --- Search : 1380.921 --- ### F = 322.12
3121 :: GetFood : 1800.953 --- Fight : 1023.359 --- Search : 906.7678 --- ### F = 777.59
3122 :: Search : 1793.012 --- GetFood : 1346.979 --- Fight : 1148.466 --- ### F = 446.03
3123 :: GetFood : 1680.578 --- Fight : 996.699 --- Search : 327.5295 --- ### F = 683.88
3130 :: GetFood : 1888.326 --- Fight : 1600.584 --- Search : 1286.826 --- ### F = 287.74
3131 :: Search : 1805.334 --- GetFood : 478.9517 --- Fight : 249.2059 --- ### F = 1326.38
3132 :: GetFood : 1776.926 --- Search : 445.8824 --- Fight : 318.6334 --- ### F = 1331.04
3133 :: GetFood : 1675.711 --- Fight : 546.7304 --- Search : 456.1378 --- ### F = 1128.98
3200 :: GetFood : 1823.824 --- Search : 1451.068 --- Fight : 401.3226 --- ### F = 372.76
3201 :: GetFood : 1874.728 --- Search : 988.303 --- Fight : 0 --- ### F = 886.43
3202 :: GetFood : 1923.644 --- Search : 1101.616 --- Fight : 372.0232 --- ### F = 822.03
3203 :: GetFood : 1915.512 --- Search : 1228.594 --- Fight : 992.1511 --- ### F = 686.92
3210 :: Fight : 1854.899 --- GetFood : 1725.752 --- Search : 1724.999 --- ### F = 129.15
3211 :: GetFood : 1816.447 --- Fight : 922.8217 --- Search : 318.5028 --- ### F = 893.63
3212 :: GetFood : 1865.714 --- Search : 1341.147 --- Fight : 491.9912 --- ### F = 524.57
3213 :: GetFood : 1874.234 --- Search : 1193.613 --- Fight : 345.6716 --- ### F = 680.62
3220 :: GetFood : 1882.994 --- Fight : 1663.334 --- Search : 1607.104 --- ### F = 219.66
3221 :: GetFood : 1802.598 --- Search : 309.4332 --- Fight : 245.8915 --- ### F = 1493.16
3222 :: GetFood : 1836.921 --- Fight : 1110.354 --- Search : 1099.508 --- ### F = 726.57
3223 :: GetFood : 1812.343 --- Search : 1477.646 --- Fight : 1064.983 --- ### F = 334.70
3230 :: GetFood : 1791.076 --- Fight : 1024.58 --- Search : 593.4374 --- ### F = 766.50
3231 :: GetFood : 1673.746 --- Fight : 334.8036 --- Search : 66.38159 --- ### F = 1338.94
3232 :: Search : 1924.25 --- Fight : 1041.998 --- GetFood : 1015.019 --- ### F = 882.25
3233 :: GetFood : 1881.107 --- Search : 1153.088 --- Fight : 1072.974 --- ### F = 728.02
3300 :: GetFood : 1939.447 --- Search : 1868.283 --- Fight : 1446.427 --- ### F = 71.16
3301 :: GetFood : 1936.133 --- Search : 1099.972 --- Fight : 488.3645 --- ### F = 836.16
3302 :: GetFood : 1941.889 --- Search : 1810.685 --- Fight : 693.6811 --- ### F = 131.20
3303 :: Search : 1931.233 --- GetFood : 1902.214 --- Fight : 1411.463 --- ### F = 29.02
3310 :: GetFood : 1920.464 --- Fight : 1793.555 --- Search : 1792.276 --- ### F = 126.91
3311 :: GetFood : 1922.717 --- Fight : 578.4097 --- Search : 542.3124 --- ### F = 1344.31
3312 :: GetFood : 1880.458 --- Search : 1353.045 --- Fight : 1006.976 --- ### F = 527.41
3313 :: GetFood : 1918.119 --- Search : 1783.248 --- Fight : 1698.333 --- ### F = 134.87
3320 :: GetFood : 1948.201 --- Fight : 1848.757 --- Search : 1824.157 --- ### F = 99.44
3321 :: GetFood : 1901.966 --- Search : 924.9729 --- Fight : 627.829 --- ### F = 976.99
3322 :: GetFood : 1907.084 --- Search : 1148.264 --- Fight : 999.8345 --- ### F = 758.82
3323 :: GetFood : 1935.987 --- Fight : 1805.738 --- Search : 1772.108 --- ### F = 130.25
3330 :: GetFood : 1908.097 --- Search : 1510.424 --- Fight : 1188.437 --- ### F = 397.67
3331 :: Fight : 1744.113 --- GetFood : 652.6238 --- Search : 282.2049 --- ### F = 1091.49
3332 :: Search : 1887.404 --- Fight : 947.8662 --- GetFood : 773.0244 --- ### F = 939.54
3333 :: Search : 1834.848 --- Fight : 1653.131 --- GetFood : 1530.358 --- ### F = 181.72

c) Misc. Images

Illustration 21: The RAIN Behavior Tree used with the learners

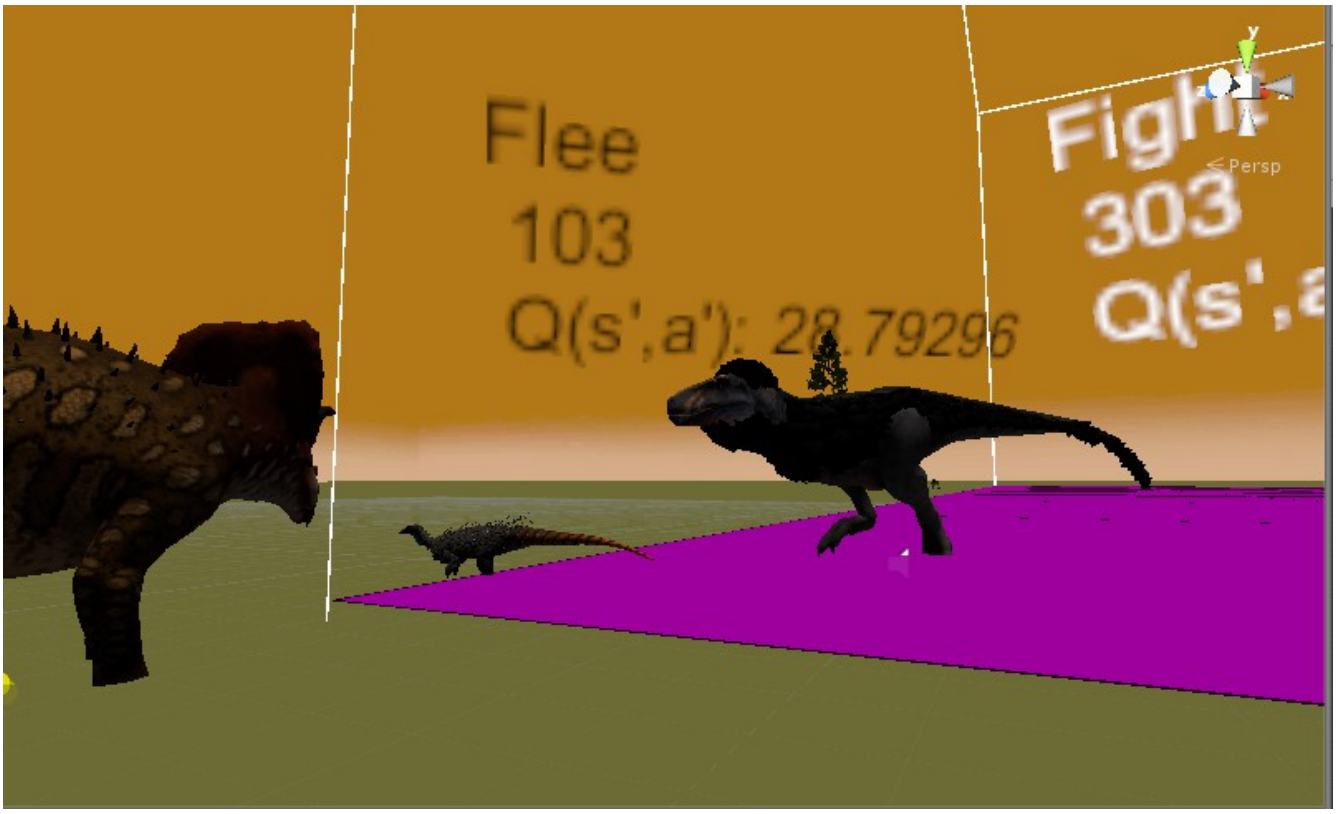


Illustration 22: Prey (embodied as the small dinosaur, Thescelosaurus) trying to escape from the predator (the T. rex) at the corner of the NavMesh (visualized by the purple plane), moments before being killed and eaten. This was before a bug fix that gave the prey a better chance at escaping these situations. I am the Triceratops standing outside of the NavMesh.

d) Link to supplemental video footage from Experiment 2*:

(*referred to as 'Trial 3' in video, which was edited and uploaded before final edits on this paper were made)

http://bit.ly/1y2AyI7