GPU Bee-Havior: Honey Bee foraging AI simulated on a GPU

Lasse Laursen

June 15, 2007

Contents

1 Introduction **3**

2 Reading Instructions **4**

3 Bee-haviour **4**
- 3.1 Perception and orientation of the world . . . 5
- 3.2 Activities . . . 6
  - 3.2.1 Scouting . . . 6
  - 3.2.2 Communication . . . 7
  - 3.2.3 Nectar gathering . . . 9

4 Previous/Related Work **10**
- 4.1 Mathematical Models . . . 10
- 4.2 Agent based Models . . . 11
- 4.3 GPU executed AI . . . 12

5 Bee-havior and World Model **13**
- 5.1 Bee State Machine and Transitions . . . 15
  - 5.1.1 Probability Model . . . 15
  - 5.1.2 Extended Model . . . 18
- 5.2 Simulated World Scenario . . . 21
- 5.3 Bee perception and scouting . . . 22
- 5.4 Model limitations . . . 23

6 Simulating Bee Foraging on a GPU **26**
- 6.1 From CPU to GPU . . . 26
- 6.2 GPU limitations . . . 27
- 6.3 Bee State-machine execution . . . 28
  - 6.3.1 Bees as fragments . . . 28
  - 6.3.2 State-Machine Cg Pipeline . . . 29
- 6.4 Persistent GPU State Management . . . 30
- 6.5 Bee rendering . . . 31

7 Program Description **32**

8 Results and discussion **34**
- 8.1 Probability Model . . . 34
- 8.2 Extended Model . . . 36
  - 8.2.1 Extended Vision . . . 36
  - 8.2.2 Narrow Vision . . . 39
  - 8.2.3 Regular sized colony . . . 40
- 8.3 Performance comparison . . . 40
- 8.4 Future exploration and improvements . . . 41
- 8.5 Summary . . . 44

References **45**

1 Introduction

Since the original discovery and decoding of bees dancing language by Karl von Frisch [26], the amount of research regarding the behavior pattern of bees has steadily increased.

The overall goal of this project is to simulate the foraging behavior of a colony of honey-bees (*Apis mellifera*) on a GPU.

Foraging is act of search for food or provisions. The term is used loosely in this project. For example, bees gathering food at a source are still referred to as foraging bees although they have clearly found food and are no longer searching.

The intention of the aforementioned goal is further specified in the following points:

- To reproduce foraging behavior of honey bees similar to the empirical data collected by Seeley et al. in their article [14].
- To examine which techniques can be used in conjunction with the simulation of AI on a GPU.
- To measure the performance impact of various functionality on the GPU.

I believe the project has met the specified goals on most accounts. Initially it was my intention to convert an existing model simulating the foraging behavior of honey bees from the CPU to the GPU. However, lack of documentation and GPU limitations forced me to create my own models with which to simulate the foraging behavior. These models are still based upon existing models and research related to honey bees. A consequence of creating my own models is that the amount of techniques and performance comparisons is smaller than I had originally intended. However, the models still contain differing techniques as well as performance differences which are discussed in this project. The results of the simulated test runs using the models are presented in Section 8.

General purpose programming on a GPU is not new, but will in my opinion become more widespread as the GPU's increase in functionality and versatility.

GPUs have existed since the beginning of the 1980s, initially charged with handling graphics related tasks. As technology has progressed, so have the GPUs, becoming more powerful and further offloading the CPUs more complex graphic related tasks. Arguably one of the biggest breakthroughs in GPU technology occurred in the mid-1990s where CPU-assisted real-time 3D rendering was completely offloaded on to 3D capable GPUs.

This evolution in technology eventually led to the introduction of programmable shaders in 2001. Programmable shaders laid the groundwork for what was essentially the beginning of general purpose programming on the GPU. Each iteration of graphics cards since has brought with it more speed, a newer shader model, and consequently more flexibility.

While the initial shader models were arguably restricted to purely graphical customizations, the 3.0 shader model introduced 32-bit floating-point precision, dynamic branching, and the possibility for writing longer shaders. Essentially removing many general purpose programming barriers.

With regards to mainstream applications, modern day GPUs can be considered a powerful calculatory resource. Even though GPUs still carry certain restrictions in comparison with modern CPUs (see Section 6.2), it comes as no surprise that GPUs are now being utilized in a wide array of non-graphic areas including: Grid Computing, Neural Networks, Cryptography and many others. This project being no exception, which aims to simulate bee foraging AI on a GPU (specifically the GeForce 7600 GT [28]).

As previously mentioned, the foraging behavior of honey bees has been the focus of many recent studies [14, 16, 11]. The past century has brought with it new insight into how a bee hive functions and how a colony with limited centralized control can still manage to optimize its foraging behavior in a dynamically fluctuating environment.

I hope to be able to simulate bees to the extent that the results will be comparable with empirically collected data. This type of simulation on a GPU is arguably still a mostly uncharted region of computer science. Therefore the goal of this project is as much intended to simulate bees on a GPU, as it is intended to find out if and how this can be achieved while producing empirically comparable results.

2 Reading Instructions

It is recommended that this document be read in sequential order. In Section 3 I explain the current understanding of bee behavior, relevant for this project, as observed by the scientific community. The section will mainly focus on how bees forage and scout for food. Section 4 begins by examining past projects involving bee simulation. Section 5 explains the behavior model to be implemented in this project, including all variations of behavior as well as sources of inspiration. Section 6 elaborates on existing limitations regarding the GPU used for the project. However, the main focus of the section is the techniques and methods used in order to simulate the bee model described in Section 5, on a GPU. In Section 7, in-depth details will be provided regarding specific sections of the program source that require further explanation. Finally, Section 8 evaluates a selection of test runs in order to establish which AI lead to the most realistic simulation of bees as well as which technologies proved to be most useful for achieving this goal.

3 Bee-haviour

As mentioned in the introduction, several advances have been made regarding the behavioral characteristics of bees. Ranging from the discovery of the dancing language of the bees [26] to more detailed knowledge regarding all aspects of bee cooperation, hive management and social behavior.

This section will detail the current understanding of bee behavior relevant for this project, including some basic details regarding bee orientation. Most areas of bee behavior that this project simulates has achieved a broad consensus within the scientific community. However, it is important to note that this consensus is not the always the product of indisputable proof. Research on live insects do not allow for the same clinical noise-free environments that other areas of

research support. Only recently have certain long-standing hypothesis gained directly-related empirical proof to further support their credibility [12].

Consequently, various areas of bee behavior have not been understood to the level of detail that a simulation would require. Fortunately, this has not proven to be an insurmountable problem as several simulations (detailed in Section 4) have produced results close to available empirical data.

Details regarding which approximations and assumptions the behavior model in this project has are detailed in Section 5.

3.1 Perception and orientation of the world

A bees visual perspective of the world is very different when compared to that of a human being. Instead of a single eye, bees view the world through compound eyes: arrays of hundreds of single eyes. Each with its own lens and each looking in its own direction. Because of this, it is perhaps not surprising that bees use different means to orient themselves, than humans do.

According to most recent studies [19], bees navigate by using a combination of techniques involving, the sun as a compass, polarized light, landmarks and distance measurement.

Studies show that a newly hatched bee performs a number of "orientation flights" before proceeding to forage at its first site. An article [11] that measures these initial flight patterns by bees via radar supports this hypothesis and suggests that the bee learns the landscape in a progressive fashion. The article also supports the notion that bees use a combination of cues to orient themselves in the world. Researches have yet to determine exactly how a bee combines the previously mentioned techniques to navigate through the world.

Below I will describe the main methods of orientation a honey bee uses:

- **Sun compass** - The first discovered navigation method of the honey bee. Each bee is capable of approximating its horizontal angle to the sun. The angle to the sun is crucial when bees inform each other where to find a source of nectar (as detailed in Section 3.2.2).

- **Polarized light** - Research has shown that bees do not actually need to see the sun in order to navigate by it. As long a small patch of blue sky is visible, the bees can approximate where the sun is, due to polarized light. Bees are capable of interpreting polarized light (blue-ultraviolet light) due to the different color spectrum they can perceive, when compared to humans.

- **Landmarks** - In case the entire sky is covered by clouds, bees have to use alternate methods to navigate and from their hive. Even though the bees sight is "only" comprised of over 8500 separate "receptors" (as opposed to the millions of nerves in the human eye), they are able to recognize various landmarks by which they can navigate.

- **Distance measurement** - A crucial part of bees navigation is their inbuilt distance measure capability. When learning of a potential source for nectar, bees measure the distance to the source from the hive. Being able to estimate distances allows the bee for a much better chance of both finding a given source and communicating its location to other bees. Even

though this sense, like the other senses, is rarely 100 percent accurate, it will give the bee an informed guess at where to start searching.

The methods mentioned above are not the only ones by which a bee navigates. Research suggests that bees both mark and smell surroundings in order to better find what they are looking for.

A arguably separatist group of the scientific community argues that bees find their forage locations by relying only chemical navigation (smell). It is beyond the scope of this project to enter into this discussion.

3.2 Activities

Through a bee's entire life span of approximately 30 days, it will undertake several different tasks such as cleaning cells, tending to the queen, eating pollen, patrolling or resting. Research suggests that some bees specialize in certain activities while other bees may avoid a particular activity entirely. For example, one bee may never guard the hive whereas another will specialize in it intently [24]. On average, every bee will spend approximately one third of its entire life foraging. While bees in general perform a large variety of tasks every day, after the age of fifteen days, most bees will devote all of their time to foraging.

The foraging behavior of a bee colony is an activity that has been the origin of many recent studies. Perhaps because it is one of the easiest to observe, but probably also because it is the one activity bees devote most time to.

This project's main goal is simulate the foraging behavior (of nectar) of bees. As such, I will limit the discussion of "real-life" bee activities to foraging and closely related activities. These activities include the communication between bees as well as scouting performed by them to find new and known sources to forage from.

The average honey bee hive consumes approximately 20 kg of nectar a year. All this nectar has to be efficiently collected if the hive is to stand any chance of survival. The fact that the availability and position of nectar sources constantly changes makes this challenge even greater.

The bee use a system partially composed of positive and negative feed-back loops informing bees which sources are lucrative to forage at and which are less interesting. Additional bees are recruited to the nectar sources with most yield. Sources with a lower yield are automatically abandoned by the forager bees.

The process of foraging can roughly be separated into three separate sections. Scouting, communication and actual nectar gathering. Bees scout for new and unknown sources of nectar, they communicate their discovery of a new (or even old) source to other bees by dancing on a "bee dance floor". Finally, the bees depart in order to collect the food.

All of these activities are described in further detail in the following chapter.

3.2.1 Scouting

The scouting behavior of honeybees varies greatly depending on the sources available. Approximately 10 percent of bees actively scout when looking for a new source (as opposed to following information communicated on the dance floor). This percentage is highly dependent on the amount of bees present on

the dance floor communicating their own discoveries. Research has shown that less dancing bees leads to more scouts.

Studies suggest that bees scout a large area surrounding the hive when searching for new sources. Instances of bees flying up to and beyond 10 km in order to forage from a source have been reported. Based on several studies, a search radius of 6 km for an average hive is to be expected [24]. Researchers speculate that while this large radius (covering over 100 km^2) may be needed in order to satisfy demand, it is perhaps also used in order to achieve better efficiency by foraging more lucrative resources.

The search pattern used by each individual bee is at the time of writing still largely unknown. However, since recent studies [12] have begun tracking individual bees via radar, it is in this authors opinion only a matter of time before the search pattern is accurately mapped.

Bees do not only performing scouting when looking for unknown sources of nectar, but also for a known sources. Certain inaccuracies exist when one bee informs another of a source of nectar, and therefore the newly recruited bee must still search for the advertised source once arriving at its supposed location.

I will devise a selection of simple search methods that may or may not be similar to how a bee actually searches. These methods are described in detail in Section 5.3.

3.2.2 Communication

The most widely researched and arguably complex of the three areas is the communication among the bees.

Inter-bee communication is what enables a honeybee hive to properly work together and forage from the best available resources. The basic process of how one bee informs another of a source of nectar is "simple". Once the bee returns from a source it has found while scouting, it will perform a so-called waggle dance. This dance is performed on the dance floor, informing other bees of its discovery. Any non-dancing bee on the dance floor has the opportunity to observe the dance and learn of this newly discovered source. Although the location where bees dance is referred to as a dance floor, it is actually performed on a hanging bee-comb, close to the hive entrance. The bees are technically dancing on a wall, not a floor.

The dance performed by the bee involves running through a small figure-eight pattern. The bee begins by performing a waggle run where it wiggles its abdomen vigorously while moving straight forward. The bee then turns right to circle back to the original starting point. Another waggle run is performed, but this time the bee circles left afterwards, circling back to the starting point. The bee dances in this fashion alternating between turning left and right until the dance is finished. The direction and duration of a single waggle run informs the other bees in which direction and how far away a source of nectar is to be found. Flowers that are directly in line with the sun are represented by waggle runs performed vertically upward. Any deviation to the left or right are interpreted as a corresponding deviation left or right of the sun. The length of an individual waggle run is interpreted as the distance to the advertised source. The further away the source is, the longer the waggle run, with a rate of increase of about 75 milliseconds per 100 meters. A example of a waggle dance can be seen in Figure 1.

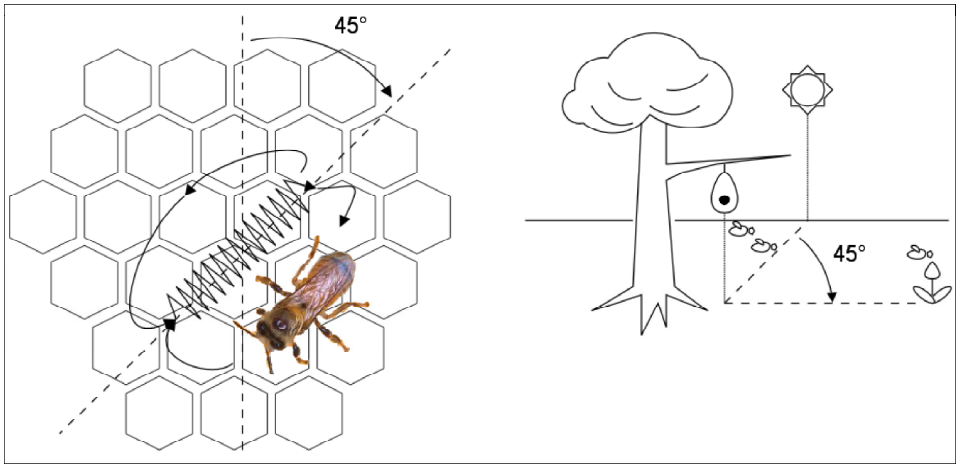

Figure 1: Visualization of the waggle-dance. *Left*: A bee performing a waggle-dance at an angle of 45. *Right*: The corresponding angle from the sun to the advertised source of nectar. Images re-drawn and slightly modified from Seeley's book [24]. Modified image of bee credited to Bauer [1].

If and how long a bee decides to dance for depends on several factors. Source profitability, personal bee dance-threshold, the time to unloading the nectar once the hive has been entered.

Bees are able to measure how lucrative a source is straight from birth and several studies [24] [6] suggest that bees judge the profitability of a source according to energetic efficiency.

$$(G - C)/C \tag{1}$$

Formula 1 shows the energetic efficiency calculated as the energetic gain minus the energetic cost, divided by the cost again. Using this formula, bees may prefer to exploit closer sources, rather than ones with higher yield that are further away. In addition to measuring the profitability of a source, each bee has its own threshold for when they perform dances on the dancefloor. Figure 2 which shows empirical data (originally from [24]) concerning six individual bee dance thresholds. Each line represents an approximation of individual dance-responses measured from the same bee. As the graph shows, bee nr.6 performs feeble dances (dances with few waggle runs), almost regardless of how profitable the source may be. Bee nr.1 on the other hand, dances vigerously for poor sources and even more vigeriously for rich ones.

Researchers speculate that bees possess this individual threshold in order to ensure a broader dynamic range when responding to nectar sources. However, preliminary studies [6] show that although the diversity among dance thresholds improve nectar gathering, it is not critical as one could imagine. A simulation of bees possessing identical thresholds yielded almost as good results.

The final previously mentioned factor that enters into whether a bee should dance or not, is the time it takes for the bee to unload its nectar. In a bee hive, a large number of workers act as food storers that accept the nectar incoming

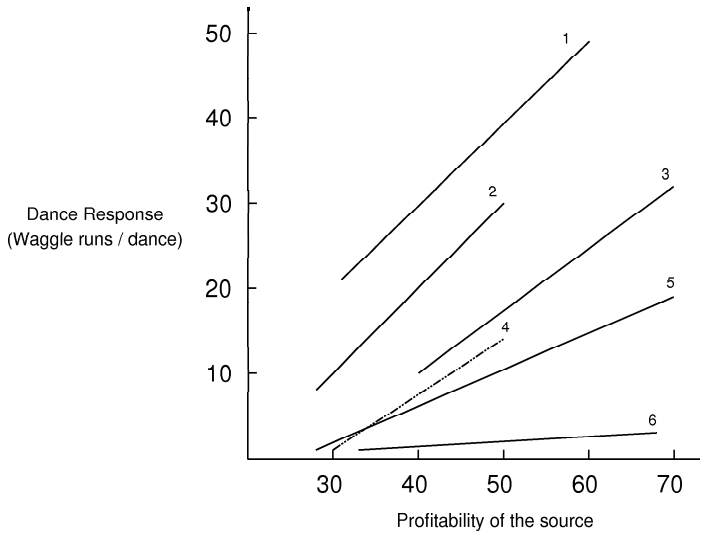

Figure 2: Six individual bee dance thresholds. Profitability based on the concetration of sugar solution provided by the feeder the bees foraged at. Redrawn from Seeleys book [24].

bees carry. These bees then store the nectar for later use and resume their duty of offloading incoming bees. Occasionally these bees will also wander outside the hive to offload incoming bees earlier. The time from arrival to the offloading of nectar affects whether or not a bee advertises the source it came from. If the bee cannot find an available food-storer bee fast enough, it will decide not to perform a waggle dance. However, in some instances the bee might perform a tremble dance which is intended to recruit more food-storer bees. This is a vital part of the collaborative effort in a real hive, but will not be discussed further since it is beyond the scope of this project. Receiver bees are only indirectly simulated in this project.

3.2.3 Nectar gathering

Once sources of nectar have been found and bees start dancing on the dancefloor, all that remains for the bees to gather the nectar. It is not uncommon for newly recruited foragers not to find the source they are looking for on the first try. Especially if there isn't a strong scent connected to it. In this case the bee will usually return to the hive and observe another dance related to the same source. After which it will attempt another foraging trip. Depending on the distance from the hive to the source, it may take an average bee 5 individual trips to find the advertised source. It is important to note, that there is no guarantee, that a bee will observe a dance for the same source twice in a row. According to Seeley [24] a bee may alternate between trying to find several sources before finding and committing to one.

Studies show that the profitability of a source affects the speed at which bees forage. More lucerative sources will cause the bees to work faster and dance more vigeriously. This behaviour improves the positive feedback loops,

already created by lucerative sources, even more.

A final note regarding the foraging behavior of bees. Research [24] shows that bees may not forage according to the same criteria during the whole year. Foraging behaviour is highly dependant on the nectar intake/supply in the hive as well as the available sources. While the previously mentioned behavior and characteristics are all based on valid research, they are still only a part of what is involved in the foraging behavior of bees.

4 Previous/Related Work

As previously mentioned, bees have been the focus of several studies after the original discovery and decoding of their dance language by Karl Von Frisch [26]. Several of these studies have focussed on building models with which to simulate, test and/or verify bee behavior and theories. In general, these models can be seperated into two categories: agent based models, or models based on differential equations. These equation based models are often referred to as mathematical models, although the agent based models are mathematical aswell. For the sake of conformity, I will refer to models based on differential equations as mathematical models from this point on.

In the next two sections I mention existing models and articles related to bee foraging which I have researched for this project.

Apart from bee foraging behavior, this project presents work related to Artificial intelligence. Artificial intelligence is a wide and deep field with extensive work in almost every conceivable direction. However, I have not found a single project implementing any type of decision making AI on a GPU. Instead I have researched other projects containing work that resembles some of the functionality I need. These projects are described in Section 4.3.

4.1 Mathematical Models

The mathematical models presented in this section have not influenced the project as much as the agent based models. However, mathematical models still serve this project in three different important aspects: First, the models provide sound proof that it is possible to produce results identical to empirical tests, via a set of formulas. Second, different variables (such as flight speed, bee carry capacity) used in these mathematical formulas are often directly transferrable to an agent based simulation. Finally, the mathematical models shed light on relations between different aspects in a honey bee hive, for example, how the colony is expected to react if one were to remove half of their nectar supply.

Below is a list of published mathematical models concerning bee behavior which I have examined while undertaking this project.

- **Forage/Nectar selection model [14] by Seeley et al.** - The earliest mathematical model I have reviewed. Seeley et al. present a model which focuses exclusively on the behavior and nectar selection of foragers. Based on empirically gathered research, a model of nectar source selection is created and tested against the data it was based upon.

  Other agent-based models compare their results with this article, which I also intend to do. The relevant details regarding the experiment they

performed are described in Section 5.2. The model presented by Seeley et al. supports the hypotheses and data that it was built upon by yielding results corresponding to observed behavior.

- **Forager allocation [16] by John J. Bartholdi, III et al.** - A mathematical model presented by Bartholdi et al. that also examines the bee nectar allocation pattern as well as its effectiveness. The model is not a hypothesis about how the allocation process works, but a quantitative description of the allocation process inspired by various empirical studies.

  The model presents a large collection of differential equations and variables that together analyse existing empirical data.

- **Hidden cost of information in collective foraging [5] by François-Xavier Dechaume-Moncharmont et al.** - François-Xavier Dechaume-Moncharmont et al. present a mathematical model that challenges the assumption that new recruits in insect colonies always increase a colony's net energy gain.

  The study emphasizes the importance of time constraints and the dynamic behavior involved in foraging behavior, not just steady states.

- **What makes a honey bee scout? [13] by Madeleine Beekman et al.** - As the title for the study suggests, Madeleine Beekman et al. research which circumstances lead bees to either scout independently or become a recruit. In the words of the authors it's aim is

  > [...] to determine the likely mechanism that regulates scouting in honeybee colonies.

  Via a mathematical model, Madeleine Beekman et al. partially conclude that bees can use the availablility of recruitment dances to decide to become a scout or a recruit.

4.2 Agent based Models

A few agent based simulations exist, which simulate the same behaviour that this project will focus on. Some of them also include elements that are beyond the scope of this project, for example food storers (receiver bees) or a more complex enviroment. One such agent based simulation is "HoFoReSim" whose author I have been in contact with during the development of this project.

The existing agent based simulations present a solid basis from which to create a version, that can be executed on a GPU.

I have closely examined the models mentioned below in order to gain insight into existing methods involved when simulating bees. In the remaining part of this document I will elaborate, where appropriate, if conclusions draw directly upon knowledge produced by one of these models.

- **Colony/Forager behavior and communication [4] by David J.T. Sumpter & D.S. Broomhead** - Sumpter and Broomhead present a

formal specification of bee behaviour through the use of a process algebra called "Calculus of Communicating Systems". The result is a formal specification of behaviour which is independent of implementation.

The specification focusses soley on bees that forage, which includes activities such as scouting and dancing.

- **Collective foraging via individual behavior rules [9] by Han de Vries & Jacobus C. Biesmeijer** - The goal of this article is

  > [...] to obtain a set of rules that is necessary and sufficient for the generation of the collective foraging behaviour observed in real bees.

  Vries and Biesmeijer present a set of rules and variables which together with a scenario directly from an empirical study [14] tests itself against the result from the same study.

  The simulation produces results that are very similar to the empirical study, simulating a total of 250 bees.

- **Exploration and exploitation of food sources [10] by Jacobus C. Biesmeijer & Han de Vries** - While neither a mathematical model nor directly an agent based simulation, this article deserves a mention because it examines several existing state-diagrams relating to bee behavior. In this article, Biesmeijer and Vries cover varying terminology across existing studies and point out the need for a revision of the scout-recruit concept.

  I intend to base my own bee model on the research and knowledge presented by this article.

- **HoFoReSim [8, 6, 7] by Ronald Thenius & Thomas Schmickl & Karl Crailsheim** - An agent based simulation [8] originally developed by Schmickl and Crailsheim. But in subsequent articles [6, 7] also developed by Thenius. The first published version of the simulation [8] only simulated foragers as actual agents. The simulation produced results very similar to empirically observed data [14]. In contrast to the previously mentioned agent based simulation [9], this simulation also focusses on the energy expenditure of bees while foraging.

  The simulation has also been used to test various hypothesies related to the individual dance threshold each bee posses [6].

  Recently, the model was extended to include the simulation of food-storer (or recieved bees) and related activities [7]. This currently makes it the most comprehensive simulation and arguably the most reliable.

  The simulation presented in the latest article [7] simulated a total of 1000 bees.

4.3 GPU executed AI

As previously mentioned, I have been unable to find projects that use the GPU to run AI. However, there are two GPU related projects I have researched while creating my own framework to simulate bee behavior.

- **BOIDS - Flock Behavior Based on Influence Maps [15] by Jonas Flensbak** - Flensbak presents a new flocking procedure based on influence maps. The implementation can utilize either the CPU or GPU for the creation of influence maps that the agents react to. The results show that the GPU achieves a better performance in many cases compared to the CPU.

  Proof that performing the exact same task on the GPU, as on the CPU, can yield superior results. I have been in contact with the author of this project and discussed the techniques he used to utilize the GPU. Although the negative results from preliminary results forced me to abandon influence maps in this project, possible improvements regarding this technique are still presented in Section 8.4.

- **Building a Million Particle System [17] by Lutz Latta** - An implementation of a GPU based particle system, capable of rendering over 1 million particles interacting with arbitrary geometry.

  Similar to AI, particles react to their enviroment based on a number of conditions and variables. Although the implementation in general uses out-of-date methods, I have taken a close look at their program structure in order to avoid possible pitfalls when creating my own framework for GPU program execution.

5 Bee-havior and World Model

This section will explain the behaviour model used to simulate bees on the GPU. The model is described using a top down approach, beginning with the big picture and finishing with specific details. As previously mentioned, sources of inspiration for the model will be mentioned where appropriate. This chapter will not mention any implementational considerations made when creating the behavior model as these are explained in Section 6. However, the model presented in this section is still constrained by what can be implemented on a GPU. Direct consequences regarding this matter are mentioned where necessary.

The model used to define bee behavior in this project will be a type of finite state machine, similar to the behavior models presented by both Vries and Biesmeijer [10] as well as Schmickl and Crailsheim [8]. Both models have produced data similar to the empirical data gathered by Seeley et al. from 1991 [14], proving that they are adequate in modelling the behavior of bees. At least as far as the empirical data is concerned.

Biesmeijer and Vries compare in their article [10], different procedures and results on individual behavior of social insect foragers from individual studies. They try to find a common terminology and present a refined concept comprised of seven mutually exclusive behavioral categories as well as the transitions between them. A redrawn, but otherwise unaltered version of their concept, can be seen in Figure 3. I have used the behavioral catagories they present as a basis for the finite state machine used to simulate bees in this project. The models presented in this project are simpler, with less states and transitions, as well as one or two new transitions. For example, the behavioral catagories shown in Figure 3 do not have any transition for a scout returning to the hive

without locating a source. Something which is possible in all the agent based simulations I have researched.

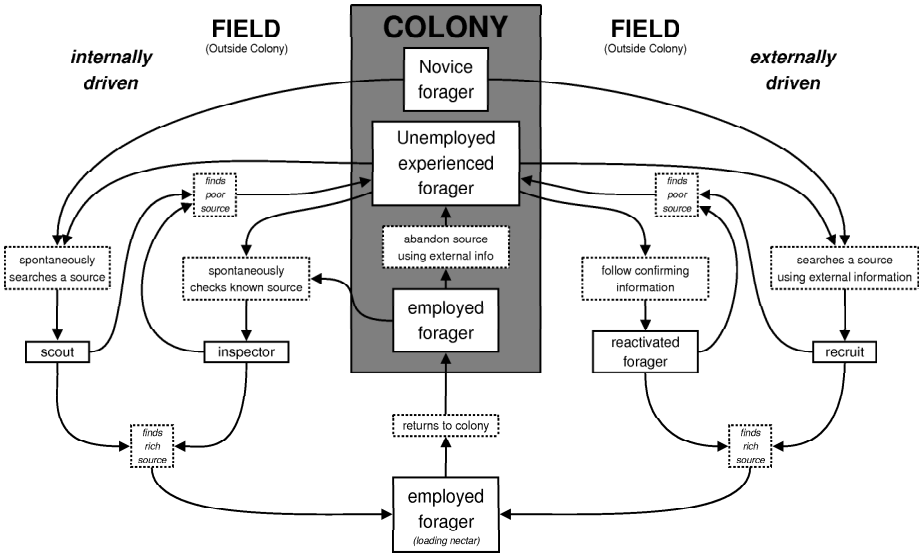

Figure 3: Behavioral control structure of a social insect forager. Shows the possible transitions between defined categories. The solid boxes represent a behavioral category. The dotted-lined boxes describe the information upon which a transition from one solid box to another is based. Left are the internally driven categories, right the externally driven ones. As the figure depicts, some of the behavioral categories are first entered once a forager leaves the hive.

The possible transitions between states resemble those presented by Schmickl and Crailsheim in their article [8]. They are as follow:

- **Fixed time delays** - For example, from foraging nectar to returning home, or scouting until giving up and flying home.
- **Fixed events** - Arriving at the hive will automatically change the state of every bee depending on the state they were in before arriving.
- **Fixed probabilities** - In many states, bees have a fixed probability to switch from their current state to another. For example from a non-foraging idle bee to a non-foraging dance following bee. This type of transition bears a strong resemblance to belief networks (or bayesian network) described in-depth by Norvig and Russell in their book [21].
- **Thresholds** - In many states external and internal stimuli influence every bee to determine a bee's transitional probability via a stimulus-response-system.

If a transition or required variable could not be extracted from the work presented in the articles mentioned in Section 4, I have used other available literature such as empirical studies of real bees. If none of these sources yielded the necessary information, I have chosen one that appeared to be likely.

The behavior the bees will exhibit in this project is known as reactive planning [30]. The bees are reactive agents and compute no more than the immediate action to be carried out, based only on the current context. They do not plan ahead, learn or use logic in order to determine the best course of action. However, according to the research available on bees, this is still enough for even real bees to cope in a dynamically changing enviroment while still maximizing their nectar gathering.

5.1 Bee State Machine and Transitions

Initially, it was my intention to base the implementation in this project directly upon an agent based model presented in another article [8]. In addition to simulating the foraging behavior of honey bees, this model also simulates their individual energy usage. An exact reproduction of the model is beyond the scope of this project. Instead approximations were implemented which significantly simplified the model and also eased the process of changing the model from being CPU to GPU based. For example, in the original implementation, dance-following bees would detect other bees within a close radius and select one when looking for dancing bees. In the simplified model, a dance-following bee chooses a bee at random until a dancing bee is found.

Preliminary tests of the approximated model did not yield anything close to the desired result. Instead, a probability model was constructed based directly upon the work presented in the empirical study [14] by Seeley et al. Once the probability model yielded results similar to those gathered in the experiment by Seeley et al., the model was expanded to give each bee more individuality and functionality.

The probabilities provided by Seeley et al. in their article [14] are based upon an experiment involving a nectar substitute. Suger water feeders to be exact. According to Seeley [24]:

> Sometimes sufficient control can be achieved with specially planted patches of flowers [...], but generally this is best accomplished by providing the bees with an artificial food source filled with sucrose solution.

Although bees prefer real blossoms, the sugar water feeders give off the scent of nectar and provide the sugar that bees would otherwise extract from nectar. For all intents and purposes, the suger water might as well be nectar as long as no real nectar sources exist near by. From this point on, sugar water feeders and nectar sources are are not distinguished and considered practically identical.

The probability and extended model are described in the subsequent chapers respectively. The results of each model compared to the empirical model is presented and discussed in Section 8.

5.1.1 Probability Model

The states and the transitions present in the probability model can be seen in Figure 4.

A number of states present in the implementation of the probability model has been left out of the figure. The missing states only serve one purpose, which

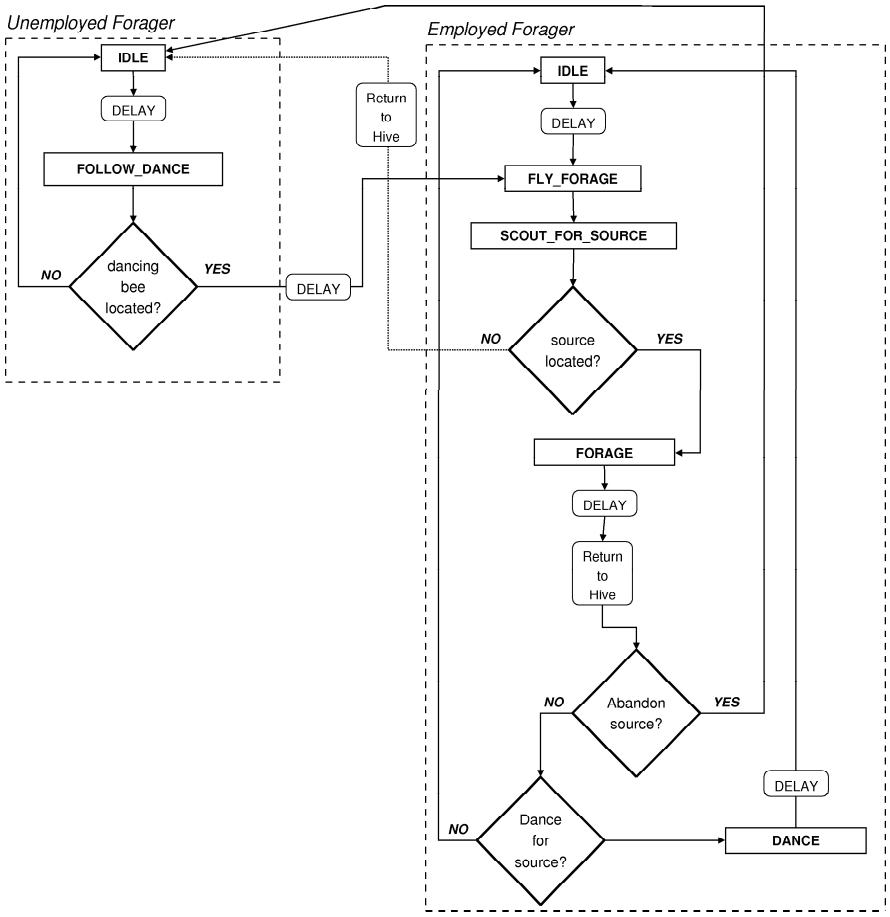

Figure 4: Probability model state diagram. Dashed boxes are a visual aid, representing an overall state that all the sub-states within can be considered to be a descendant of. The solid boxes represent states, the rounded boxes are actions, and the diamond shaped ones are branches.

is to introduce a delay in-between states. Their addition to the Figure would not introduce any new transitions or functionality.

Each delay in the figure is derived directly from the work presented by Seeley et al. in their model [14]. The exact values can be found in their article as well as in the table shown below:

Transition	Delay (in mins)	Source conc. (mol/l)
Arrival in Hive to Dance	1.0	2.50
Dancing to start of Foraging	1.5	2.50
Foraging to arrival at Hive	2.5	2.50
Following Dancer to start of Foraging	60.0	2.50
Arrival in Hive to Dance	3.0	0.75 (1.00)
Dancing to start of Foraging	2.0	0.75 (1.00)
Foraging to arrival at Hive	3.5	0.75 (1.00)
Following Dancer to start of Foraging	60.0	0.75 (1.00)

The idle state present both for the unemployed and employed forager in Figure 4 is technically redundant, because the bee does not perform any action of consequnce while being in it. Nevertheless it has been included because the unemployed forager is not in the dance following state when delayed, just like the employed forager is not dancing until it decides to forage.

The probability model is relatively simple and only contains four branches in its entire state machine. The decision making process involved in each branch is explained in detail below, including any deviation from the probabilities utilized by Seeley et al.:

- **Dancing bee Located?** - Research suggests that follower bees choose which dancing bee to follow at random. Seeley et al. estimates the probability of following a dancer from source A as $D_A/(D_A + D_B)$ and from source B as $D_B/(D_A + D_B)$, where D_i denotes the number of dancers, dancing for source i. In the model presented by Seeley et al., bees are in the dancing state until they start to forage at their known source. The time a bee takes from the beginning of their first waggle dance until they leave to forage is 90 second and 120 seconds for source of 2.50 mol/l and 0.75 mol/l respectively. However, bees do not dance for this full 90 or 120 second duration until they leave the hive. On average a bee danced 14 and 1 waggle dances for a high and low profitable source respectively. One waggle dance for 400-m target requires approximately 2.4 seconds [26] which yields a total dance time of 33.6 and 2.4 seconds for a high and low source respectively. Seeley et al. uses these time intervals to calculate a probability that counter balances the over sized amount of time a bee is in the dancing state.

  In my probability model, each foraging bee only enters the dancing state for the appropriate interval mentioned above (33.6 or 2.4), assuming the bee has decided to dance of course. Each follower bee picks a bee at random. If the selected bee is dancing, the follower bee is recruited to its source. This essentially gives each follower bee the same probability of spotting a dancer as in Seeley et al.'s model.

- **Source located?** - This branch is technically redundant, since every follower bee in Seeley et al.'s model eventually locates the source it learns about from another bee. On average, a bee in their model finds an advertised source after 60 minutes of searching.

  In the probability model, every bee is delayed for approximately 60 minutes once it has learned of a source of nectar. After said time has elapsed,

the bee is allowed to fly and forage at the known location. The source is therefore always found by a searching bee, but only after 60 minutes.

- **Abandon source?** - Identical to the behavior described by Seeley et al., a source of nectar with 0.75 mol/l (or 1.00 mol/l) is abandoned with a probability of 0.04 per foraging trip. A source of nectar with 2.50 mol/l sugar concentration is never abandoned.

  Each bee initially has a 0.00 probability rating of abandoning a source. After 5 trips to a low profitability source a bee would have a 0.20 chance of abandoning it.

- **Dance for source?** - If a bee doesn't abandon its source of nectar, there is a chance that it will perform a waggle-dance to advertise it. Keeping in line with the probabilities presented by Seeley et al., every bee dances when returning from a source of high profitability. Sources with low profitability only carry a 0.15 dance probability.

To summarize, the probabilities utilized from Seeley et al.'s article [14] in the probability model are presented in the table below:

Probability of...	Value
Locating a Source	1.0
Abandoning 2.50 mol/l Source (per trip)	0.0
Abandoning 0.75 mol/l Source (per trip)	0.04
Dancing for 2.50 mol/l Source	1.0
Dancing for 0.75 mol/l Source	0.15

The probability model produces results similar to the ones presented in Seeley et al.'s article [14]. The results are shown and its discrepancies discussed in Section 8.1.

5.1.2 Extended Model

The states and the transitions present in the extended model can be seen in Figure 5.

As with the figure of the probability model, a number of existing states have been omitted from the diagram. These states only serve to introduce a delay or set certain parameters. The consequences of these missing states are still represented in the available figure and no new transitions or functionality is missing due to these states being omitted.

With the exception of the more complex overall unemployed forager state, the extended model looks similar to the probability model. However, the transitions in the extended model are based upon the current state of the simulation and not on a fixed probability as in the previous model, with the exception of the "abandon source?" branch.

The branches and resulting transitions are described in detail below:

- **Dancing bee Located?** - In this model, each dancing bee is created with a preset dancing threshold as described in Section 3.2.2. The possible thresholds are calculated from empirical data gathered by Seeley and redrawn in Figure 2. In the real world, the bees would compare their

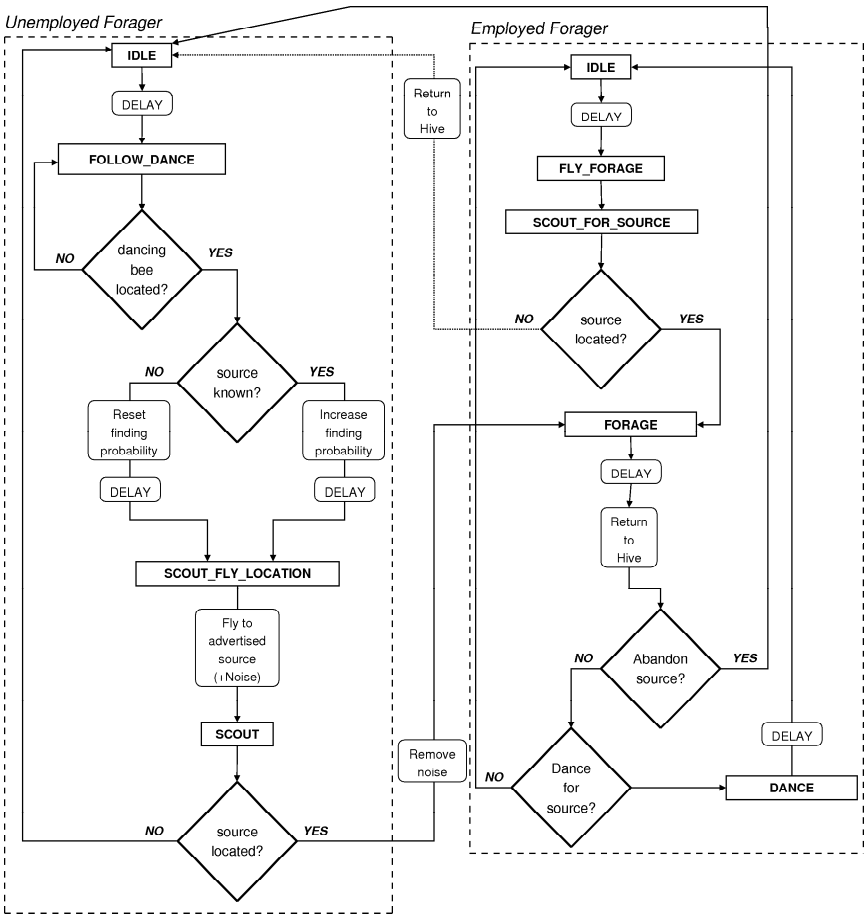

Figure 5: Extended model state diagram. Dashed boxes are a visual aid, representing an overall state that all the sub-states within can be considered to be a descendant of. The solid boxes represent states, the rounded boxes are actions, and the diamond shaped ones are branches.

own perceived profitability to the threshold and perform the appropriate amount of waggle runs. Studies suggest that the bees judge the profitability of a source according to energetic efficiency. Both sources of nectar that the bees will forage are an equal 400 meters away from the hive (described in Section 5.2). The only percievable difference between the sources for the bees in this simulation is the sugar concentration they provide. The profitability of the source perceived by the bees is therefore directly derived from the profitability of the source they are foraging. I have adjusted this perceived profitability to yield approximately the same dance times as observed by Seeley et al. in their article [14].

In the probability model, each follower bee would at random choose a bee and always find the source after its one hour delayed foraging flight. If the chosen bee wasn't dancing, the follower bee would be delayed and

then try to choose another bee. Initially, the bees in the extended model used the same following procedure. In contrast to the probability model however, a bee would not necessarily find an advertised source on its first foraging flight. This lead to a large group of bees finding one dancing bee, performing a foraging flight, failing to find the source, and finally unable to find another dancing bee for a long time. The dance following procedure has therefore been altered for each bee to indefinitely search until it finds another dancing bee. It is important to note that this indefinite search does not actually occur since the GPU will eventually stop a shader programs execution once it has passed an instruction execution limit.

However, the change in search behavior is enough to ensure that most follower bees find a dancing bee when they search for one.

- **Source known?** - Vries and Biesmeijer present a technique in their article [9], with which they introduce random noise in a bees transmission for a source of nectar. They state that most recruits search within about 200 meters from the intended source for a target located at 450 meters from the hive.

  Similar to their model and real life bees, every bee in the extended model that learns of an unknown advertised source is provided with a slightly distorted location. Every observed dance includes a random number between 0 and 200 meters of noise in both dimensions.

  If however, the advertised source is known, the follower bee reduces the previously induced noise by half. Making the finding of the source more likely with every perceived dance. When the scouting bee finally locates the source, all the noise associated with its location is removed.

- **Unemployed Forager: Source located?** - Each bee scouts for a little over three minutes, attempting to find the advertised location. If the source is not found, the bee returns to its idle state in the hive. If the source is located, the bee becomes a recruited forager and forages at the nectar source for the amount of time observed by Seeley et al. in their article [14].

- **Employed Forager: Source located?** - As in the probability model, this branch is still technically redundant. Although transitions exist for a foraging bee not to find its known source, the bee is directed to be nearly on top of it before searching for a source.

  Given the possible inaccuracies on the GPU, if the forager bee does not find its known source of nectar, it becomes an unemployed forager.

- **Abandon source?** - Identical to the behavior in the probability model. Each successful visit to a known nectar source will potentially affect the bees probability of abandonment.

- **Dance for source?** - Whether or not a bee should dance for a known source is dependent upon two things in the extended model:

  First the time it takes for a bee to deliver its nectar load to a recieved bee is determined. This is a combination of a random value between 0 and 8 minutes multiplied by the amount of bees actively engaged in foraging.

The more bees that are foraging, the more likely a longer delivery time. This value is then used as the signal strength in a sigmoid function [31] presented by Schmickl and Crailsheim in their model [8]. The sigmoid function is as follows:

$$P(s) = \frac{s^n}{s^n + \Theta^n} \tag{2}$$

Schmickl and Crailsheim obtained empirically comparable results with $n = 2.5$ and $\Theta = 10$. I intend to use the same values in the extended model.

Research has shown that bees do not work with the same vigor and speed when collecting nectar from a low profit source. As a result, a bee foraging a low nectar source will not find a reciever bee with the same speed, as a bee foraging a high profit source. Because of this, I have decided to keep the low probability of dancing for a low source as described in the probability model.

5.2 Simulated World Scenario

As previously mentioned, the goal of this project is to produce results similar to empirical data collected and described in an article [14] by Seeley et al. from 1991.

Seeley et al. observe a colony of approximately 4000 bees placed in-between 2 nectar sources (sugar feeders), each distanced 400 meters north and south respectively, from the hive. The observation takes place over the course of two days in two 8 hour segments. Each segment commencing at 8:00 and ending at 16:00. This project will only attempt to simulate and compare results with the first 8 hour segment.

Prior to the experiment, Seeley et al. have trained 12 bees to feed at the north source, and 15 bees to feed at the south source, providing the colony with knowledge of each source's existence. During the course of the experiment, the two nectar sources yield either a high or low concentration of sugar (nectar). At the beginning of the first day, the north source yields a low concentration of 1.00 mol/l and the south source yields a high concentration of 2.50 mol/l. At noon (12:00) the concentrations are (almost) switched. The north feeder provides a 2.50 mol/l solution after noon, and the south feeder provides a 0.75 mol/l solution.

Based on the experiment performed by Seeley et al., the simulated world is relatively simple. It can only contain two types of elements. A bee hive and nectar sources. Only one bee hive can exist in the world, although there can be a potentially unlimited amount of nectar sources. The world is two dimensional. The bee hive and nectar sources can be placed anywhere within a 512 by 512 grid except on top of each other. An example of a possible scenario is presented in Figure 6.

The grey pixels represent the bee hive, and the colored pixels represent different sources of nectar. Apart from those two elements types of elements, the rest of the world is drawn black, representing empty space. Each pixel corresponds to 2.7m * 2.7m in the real world.

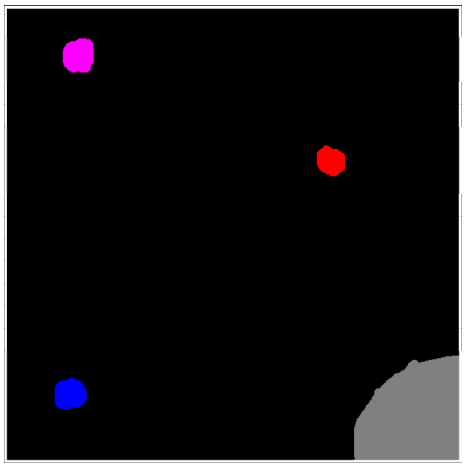

Figure 6: An example of a simulated bee scenario.

The nectar sources do not change location, but can change concentration, as in the experiment performed by Seeley et al.

5.3 Bee perception and scouting

Section 3.1 and 3.2.1 describe the perception and scouting of the real honey bees respectively. This section will explain how the simulated bees perceive and scout the simulated world they exist in.

Originally it was my intention to simulate both spontaneous scouting as well as scouting for known sources. However, the bees in this simulation only scout for known sources, never spontaneously for unknown ones. The next section on Model limitations will explain why this has little consequence and why this functionality was not included.

The world in which the bees are simulated is a very simple two dimentional scenario as described in Section 5.2. Due to limitations imposed by the GPU, the simulated bees perception of the world is very narrow. The bees are only aware of the simulated scenario and not any of the other bees. The only time at which one bee may perceive another bee is when one of them is dancing, and the other is trying to follow said dance. But even then, only the following bee is aware of the other, and not vice versa.

As described in Section 3.1, the bees use a large variety of methods in order to orient themselves in the real world.

In this project, there is only one interesting instance where a bee must orient it self in the world. This is when a bee travels to an unknown location via instructions, for example, when a bee tries to find a source advertised through a dance it has observed. A simulated dancing bee gives exact coordinates of the nectar source it is advertising to the following bee. This is obviously not how bees communicate in the real world, but an adequate approximation. The fact that a real bee performs an actual dance which is then interpreted is irrelevant. However, research has shown that real bees do not always arrive at their intended destination, and a single bee might perform 5 scouting flights

for an advertised source of nectar before actually arriving at the source. This will be simulated in the extended model (as well as indirectly in the probability model) by intentionally distorting the coordinates each bee recieves. This only applies for nectar sources and not for the hive, which every bee is assumed to find without fail.

When trying to locate an advertised source in the extended model I intend to implement two destinct abilities related to searching, as described below:

- **Narrow sight** - The bee is able to perceive what is directly beneath it. If the bee is directly above a source of nectar, it will be aware of it. However, it is directly next to it, it will not see the nectar.
- **Extended sight** - The bee can perceive what is approximately 13.5 meters in front of the bee as well as approximately 13.5 meters to the left and right of it. Flying close to a source of nectar will therefore alert the bee to its presence.

  Vries and Biesmeijer attribute every bee a perceptual distance of 25 meters in their agent based model [9]. Since the nectar sources in the simulated scenario have an approximate diameter of 13.5 meters, the bees perceptual distance of 13.5 meters even out to approxmately the same as Vries and Biesmeijer use.

The bees will search while randomly adjusting their direction. The abilities described above will aid the scouting bees in turning in the right direction or stopping once they have located the nectar. Since the a bee always locates an advertised source in the probability model, only the narrow sight ability is used in it. Both scouting abilities will be used in the extended model.

In addition to passing coordinates inbetween bees when dancing, I originally intended to implement an alternate solution using influence maps. Instead of giving the dance-following bee exact coordinates of the nectar source, each dancing bee would project a circle on an influence map, relative to the source they are foraging. Once each bee has participated by adding its known source to the influence map, the combinated circles can then be used to lead newly recruited foraging bees in the right direction. The spot on the influcence map with the highest value will be nectar source advertised by the most bees. However, preliminary testing has shown that the implementation suffers a huge perforamance drop, from approximately 450 to 20 fps, when blending multiple quads. A number of optimizations were applied without any significant gains. This alternative solution will not be further considered in the project, except in Section 8.4 where possible improvements or alterations are discussed.

5.4 Model limitations

Apart from the simplification of both the enviroment and bee behavior when compared to the real world, the models presented in this project have a number of other relevant limitations.

Below is a list detailing elements relevant for the proper simulation of bees in a realistic enviroment which are not in this project, as well as possible consequences of their absence:

- **Spontaneous scouts** - In the real world, bees which spontaneously scout are vital to the success and survival of a bee hive. New sources must constantly be located and exploited in order to keep up with the nectar demand of the hive.

  Within the scenario which this project simulates the scouts are almost completely redundant. There are only two sources available for the hive to exploit and a select number of bees already possess knowledge of their existance prior to the experiment. In two comparable agent based models, the spontaneous scouts are arguably an equally small factor. Schmickel et al. present in their article [8] a maximum probability of 1 to 10000 for a scout to appear in every simulated step. This probability decreases (becomes more unlikely) as more bees are recruited. Vries & Biesmeijer do not simulated sponaneous scouts at all in their model and state in their article [9] that:

  > "If their function is to detect good sources not yet known to the colony, then their behaviour will not alter the colony's collective forager pattern given the experimental design used in this model, since the only two sources available are the ones that are already known to the colony. Moreover, the chance that a scout would find one of these two sources by random search would be very small, so the effect of including scouts in the model would only be that the number of unemployed bees inside the nest is somewhat diminished."

  Therefore I have chosen not to simulate spontaneous scouts in order to avoid introducing unnecessary complexity. Additionally, any change perceived due to the different search abilities presented in the previous section will be directly related to bees scouting for a known source which I believe can have a large impact.

- **Other colonies** - Bee's rarely exist in a closed enviroment and are often forced to compete with other colonies - even steal resources from them or have their own stolen.

  Conversely, the bees simulated in this project are modelled after bees observed in an arguably closed enviroment. Therefore the lack of other colonies does not bear any consequences for the results. In addition, bees only "steal" nectar from other colonies when there is a severe nectar shortage, which is not the case in this experiment.

- **Dynamic environment** - The real world is an ever-changing dynamic enviroment to which the bees adapt. This project only simulates static nectar sources which richness may vary.

  This limitation is not significant in regards to this project as the two sugar feeders remain static, but their richness changes once during the experiment. However, proper simulation of bees in an actual enviroment requires the creation and removal of new nectar sources at many different locations.

- **Foraging behavior** - Apart from nectar (sugar), honey bees also forage both pollen and water. Seeley notes in his book [24] that the foraging

behavior of bees are likely to change during an entire year, depending on which resources are available and how much nectar/pollen/water the hive has in storage. Research has also shown that the weather and temperature has a large impact on the bees foraging behavior, none of which is simulated in this project.

The foraging behavior of the bees simulated in this project is based on a specific set of empirically gathered data from bees observed at Cranberry Lake Biological Station in northern New York State on a clear day in the middle of June. The behavior simulated in this project should resemble bees observed in any comparable enviroment. Any deviation in temperature, surroundings, availability of water/pollen may alter the foraging behavior of real bees.

- **Receiver bees** - As described in Section 3.2, a bee colony is comprised of several thousand bees each performing one of many activities. Only a few of these activities are simulated in this project. One activity not simulated, is the recieving and storing of nectar, which is closely related to the foraging behavior of the bees. The receiver bees are only indirectly simulated by letting every returning forager wait a random amount of time before delivering its nectar in the probability model. The extended model includes a more realistic depiction by using the total amount of active foragers as a factor in how long a bee waits to deliver its nectar.

  The waiting time each bee experiences is based on the empirically gathered data and comparable simulations. Therefore the result should resemble that of a simulation which includes the receiver bees.

- **Life Cycle** - Real bees spend two thirds of their life on other activities than foraging for food. Many of which also impact the foraging behavior of bees in the hive.

  There are two important aspects worth considering regarding the absence of the life cycle in this simulation: First, bees performing non-foraging activities may directly and/or indirectly influence the actions of foraging bees. For example, upon arriving at the nest a foraging bee will attempt to locate a receiver bee to deliver its load. The amount of bees present at the hive entrance, such as bees patrolling, ventilating the nest or cleaning cells etc. may delay the foraging bee in finding a receiver bee. This in turn will affect whether or not the foraging bee decides to dance once it has delivered its nectar. The exact implications of not properly simulating these activities is arguably impossible to predict. However, other projects decribed in Section 4 have produced results close to actual bees, without simulating these non-foraging activities. Simulating the individual behavior and activity for each bee is apparently not necessary when trying to reproduce bee foraging behavior.
  The second aspect worth considering is that bees who forage may undertake a new activity and at that point take their knowledge regarding a profitable with them "off the dancefloor". In the experiment [14] conducted by Seeley et al., 4000 individual bees were labelled over a period of 2 days. Within the next six days the population of the colony decreased to approximately 3450 bees and after another 10 days it had increased to 4200 bees. Seeley et al. observed on that date that all of the bees

exhibiting foraging behavior were still labelled. This suggests that even though the bees undertake varying activity in their lifetime, there is still a significant amount of consistency. Not a single new forager was observed after initial labelling 16 days ago.

- **Bee Memory** - Research has shown that bees remember nectar sources that they have, but are not actively visiting. The experiments performed by Seeley et al. extends over the course of two days. On the second day, the recruitment rate of bees on the poor source is very high when compared to the recruitment rate of bees on the poor source on the first day. This suggests that many foragers who visited the source during the course of the first day returned in the hopes of still finding a profitable source.

  I believe that this limitation has minimal consequences for the validity of the results my simulation produces. This project only simulates forager bees within the course of a continous eight hour session. Within this timeframe, the effects of the bees memory is arguably negligable.

6 Simulating Bee Foraging on a GPU

Existing models simulating the foraging behavior of honey bees were presented in Section 4. I have reviewed and used some of the models as inspiration for the implementation of honey bee foraging in this project. However, since this project is executed on a GPU, as opposed to the CPU, the basic structure of the program is unlike previous implementations.

Implementing general purpose code on the GPU ranges from challenging to practically impossible, depending on what needs to be accomplished. For example, collision detection in between moving bees on the GeForce 7600 GT [28] GPU is practically impossible due to the sheer number of texture look-ups required. Consequently, collision detection among other things will not be simulated. Apart from the limitations discussed in the previous section, Section 8.4 will detail further improvements not implemented in the project.

In the the next two sections I will give a brief description of how GPUs generally differ from CPUs and what limitations are imposed upon GPU general purpose programming. The sections can be skipped if the reader already has knowledge regarding these topics as nothing further is mentioned with specific relation to this project.

The remaining subsections in this section describe how the behavior model described in previous sections is executed on the GPU, certain parts of the program structure as well as specific techniques used in order to achieve CPU similar functionality, on a GPU.

6.1 From CPU to GPU

A CPU is generally a sequential execution processor. Given a set of data and a set of instructions to execute for each piece of data, a CPU will sequentially perform all instructions for each piece of data before proceeding to the next. Modern CPUs contain a large amount of advanced logic to determine which instructions are independent of one another and can actually perform instructions

out of order, while still arriving at the same result. The purpose of this is to maximize efficency by avoiding CPU stalling, which is when the CPU is delayed due to waiting for a result.

Within the last few years, CPUs containing multiple cores have been released to the average consumer. Essentially, this means that such a CPU is able to execute seperate sequential code simultaniously on two seperate pieces of data.

GPUs, which are generally stream processors, are able to process a set of instructions on several pieces of data at once, similar to a CPU with multiple cores. However, the stream processor sacrifices some flexibility in its model in order to allow easier and faster execution. Stream processors are built exclusively for accepting a set of data, executing a set of instructions for the entire set and returning the result.

It is important to note that a stream processor would not perform better (probably worse) than a sequential processor, with a data set where each members result depended on the result from the previous member. Simultainous execution of instructions on several pieces of data requires that they are independent from eachother.

6.2 GPU limitations

Even though modern day GPUs have advanced considerably since their original creation, there are still a number of significant limitations to them. Especially when considering general purpose programming. While the following limitations mainly apply to the GPU utilized in this specific project (GeForce 7600 GT [28]), some still apply to the newer generation. Where possible I will specify advances that have effectively removed or minimized the limitation.

The following are noteworthy limitations when using the GeForce 7600 GT to preform general purpose programming on its GPU:

- **Instruction limitation** - Programs executed on the CPU have limitless potential with regards the amount of instructions being processed. This is unfortunately not true on the GPU. While the shader model 3.0 [29] standard (with which the GeForce 7600 GT is compliant) specifies that both vertex and pixel shaders must allow for a minimum of 512 instruction slots per program, the upper limit of instructions executed is around 65,000 for both the pixel shader and the vertex shader. While 65,000 instructions executed initially seems like a high boundary, this can quickly be exceeded by loops iterating through big arrays and/or textures.

  It should be noted that this limitation can potentially be sidestepped by storing temporary calculations after an incomplete run. These calculations could then be continued and finished in a succeeding program. However, there is undoubtedly a performance penalty involved by using multiple incomplete runs compared to a single execution run. Additionally - only certain types of calculations allow for this "multiple run" style on a GPU.

  In the newer shader model 4.0 [29], these restrictions have been further relaxed, by allowing more instructions to execute. In some instances the restructions have been completely removed.

- **Calculation inaccuracy** - Previous experience with creating general purpose programs on GPUs has revealed that certain calculation inaccuracies

exist on presumably all GPUs. This comes as no surprise as GPUs generally favor speed over accuracy in order to deliver rendered images as fast as possible. While this may not impact results in any significant manner, it is important to realize that simulations preformed on a GPU are bound to be more inaccurate than if they were preformed on a CPU.

If the success of a simulation depends on highly accurate results, a simulation performed on the GPU would almost certainly be considered useless.

While newer GPU architechtures may improve the accuracy, the primary goal of a GPU has always been to deliver as large an amount of calculatory power as fast as possible.

- **Recursive functions** - All the current, including the latest series of geforce cards [28] with support for the newer shader model 4.0 [29], do not support recursive functions.

  However, it is possible to achieve similar results using other programming techniques.

- **Pointers** - Pointers are not supported in the shader model 3.0 standard and at the time of writing, I have yet to confirm their support in the latest shader model.

  The lack of pointer support restricts the use of advanced data-structures on the GPU. While some data-structures that traditionally rely on pointer can be implemented using alternative techniques, the level of abstraction is lowered. The programmer is forced to interact directly with the data-structure, rather than a simplified interface.

- **Output** - Shaders executed on a GPU are very limited in their output capabilities. With shader model 3.0, the vertex and fragment shader can only alter data delivered via the in-data stream from the CPU. Both shaders are incapable of creating any new data to pass on, either from the vertex to the fragment shader, or from the fragment shader to the CPU. The only data the GPU can deliver to the CPU for further processing is the actual image it renders. Depending on the graphics card, this can be a high resolution image consisting of 4 floating point values per pixel. Essentially, this means that per executed renderpass, the GPU can provide the CPU with 4 floating point values per pixel rendered.

  In the latest shader model 4.0, a new type of shader has been introduced. A geometry shader which can create further vertices to be processed in the following renderpass. At the time of writing I have been unable to determine whether or not the latest graphics cards allow access to more data during executing than the image written to a buffer.

6.3 Bee State-machine execution

6.3.1 Bees as fragments

As mentioned earlier, the GPU is a stream processor and is optimized for the simultainous execution of instructions on several seperate pieces of data. This data is either comprised of vertices or fragements (pixels). Any general computation on a GPU must therefore be performed on the vertex or pixel shaders

(Shader Model 3 [29]). The data used in the computation must therefore be provided the in format of vertices or pixels.

The GPU (GeForce 7600 GT) used to simulate bees in this project has a total of 12 pixel shaders and 5 vertice shaders. Apart from the fact that the pixel shaders offer the largest amount of computational power, they also produce output which can be read back to the CPU. The bees will therefore be simulated using the fragmet shaders. Each bee will be represented by a single fragment upon which different calculations are performed depending on the cg program being executed. Matching one bee to one fragment is arguably the simplest method to utilize the GPU in this project.

The simplest technique for the GPU to perform a necessary calculation on every bee, is to render a quad the size of required number of bees and perform a paralel projection thereof. For example, to perform calculations for 100 bees, one needs to draw a 10 by 10 sized quad represented by 100 pixels on-screen (or in a buffer).

While it is technically possible to simulate a bee as several fragments instead of one, it would undoubtedly increase the complexity of the program.

6.3.2 State-Machine Cg Pipeline

Reynolds presents in his article [23] a solution for autonomous agent navigation in a life-like and improvisational manner. As reynolds does not define these terms explicitly, I assume that he means fluid navigation. In the terminology provided by Reynolds, the bees simulated in this project can be described as situated, embodied, reactive, virtual agents. In other words, the bees exist in a world with other bees, represented as an embodied agent, driven by stimulus and entirely non-real (virtual).

Reynolds suggests dividing up behaviour of an autonomous agent into three seperate layers: action selection, steering, and locomotion. Although this layered approach to agent behavior does not directly transfer to the implementation of bees in this project, it does have certain similarities. As decribed in Section 6.2, the GPU has certain output limitations. These limitations accomodate a division of labor among multiple programs running on the GPU. The simulation of bees is broken into four seperate programs. One controls the actual state of the bees, which could be compared to the action selection layer. Another determines the position of the bee, which covers both the steering and locomotion layer. The final two program sets the color and probability values for each individual bee respectively.

Figure 7 shows a simplified version of the rendering pipeline implemented in order to simulate the bees.

In every simulated step each bee fragment has calculations performed upon it by four individual programs. Because the functionality is divided among four seperate cg programs, some overhead occasionally occurs in order for the simulation to advance properly. For example, the `beeState.cg` program is the only cg program capable of changing the bees state. At some points, it is necessary for a bee to perform certain calcalculations and store these in conjunction with a transition between states. In this case it is necessary to introduce an intermediate state which informs the responsable program, makes it perform the calculations, and then immediately switch to the intended state.

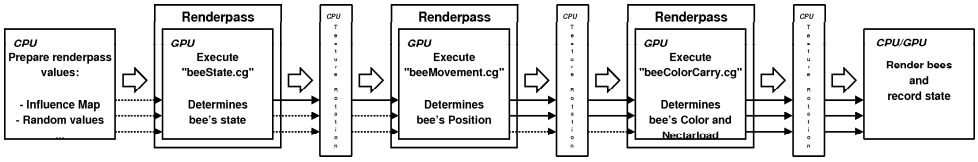

Figure 7: State machine computational pipeline. The slim rectangles represent a "texture rotation" performed on the CPU. The three solid and dashed lines from and to each renderpass represent a texture. A dashed line represents an unupdated texture from the previous simulation step. Each texture is eventually updated (and becomes a solid line) once all cg programs have been executed. The fourth program has been left out of the diagram because it makes the figure more complicated without adding anything of relevance.

6.4 Persistent GPU State Management

Almost every model or simulation has some kind of persistant state related to it, even if only for the possibility of temporarily stopping and starting the model at a specific point in time. Similarly, the bees in this simulation require an amount of persistant data related to them. Among this information is for example each bees position and current heading or the knowledge it carries regarding a source of nectar.

While the GeForce 7600 GT carries a large (256 MB) portion of memory on-board, it cannot be accessed in the usual manner as with a CPU. It's not possible to create customized (or even standardized) data containers and reserve the memory for such a purpose. The memory is strictly for texture use and must be accessed as such.

Fortunately, due to the fact that each bee is treated as a fragment on a quad, the same quad can be texturized with textures that contain information disguised as colors, but utilized as for example bee position or heading. As described in Section 6.2, depending on the graphics card the textures can contain a varying amount of information per pixel. Most modern graphic cards (including the one utilized in this project) can use 4 floating point numbers to represent a single pixel, which is the case in this implementation.

Because of the how the architechture is optimized for graphics rendering (and was never concieved for general purpose computation), it is not possible to read/write to a single texture in a render pass. Instead, the "ping pong" technique must be employed. This technique entails using the output of a given rendering pass as input in the next one. Essentially this means that every given texture used to store bee-state information will technically be represented by two textures. One texture in which to read the current information, and one with which to record the changed (new) information. The references to the two textures are then swapped following every rendering pass.

Figure 8 shows an example of a very simple Cg program performing an addition on a collection of numbers. The numbers are placed in a texture and the Cg Program is then executed with it as input. The results are placed in an alternate texture which is then used as input in the next renderpass. The result is a carried over persistant state.

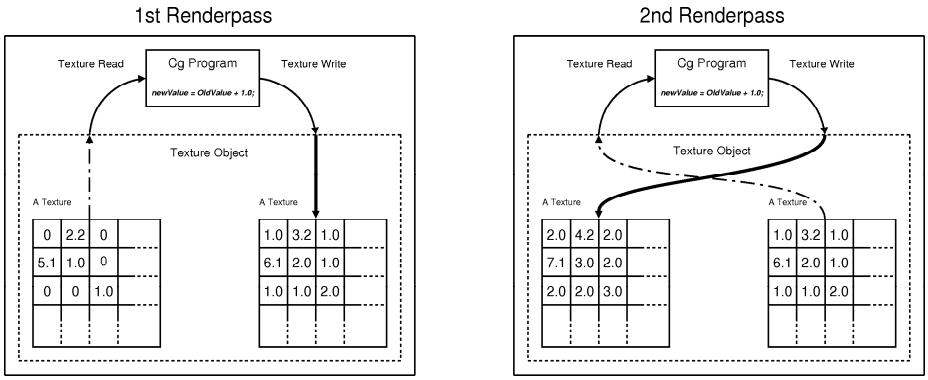

Figure 8: Texture ping-pong technique using a simple cg program. Each renderpass adds 1.0 to the exiting numbers in a texture.

6.5 Bee rendering

During the development of this project, the most recent techniques have been applied when communicating in between the CPU and the GPU. Thanks to modern programming interfaces, these modern techniques are rarely more difficult and in most instances much easier to implement and use. One exception exists, which is the rendering of the bees on screen. Initially, this was easiest to implement on the CPU in order to closely inspect each value and debug the simulation. In total, the project can display the bees using four different techinques:

- Display nothing.
- Display bees as points on the CPU.
- Display bees as triangles on the CPU.
- Display bees as triangles on the GPU (Vertex Shader).

Simulation runs testing each of the four display models (with the probability model) are shown in the table below:

Display Method	Nr. of Bees	Sec. Elapsed	Average FPS	Max FPS
No Bees	121	45	1269.47	1499.02
CPU Points	121	55	1049.24	1152.71
CPU Triangles	121	70	815.28	839.41
GPU Triangles	121	56	1037.51	1119.09
No Bees	10000	200	287.56	340.89
CPU Points	10000	206	279.64	339.57
CPU Triangles	10000	346	166.30	186.02
GPU Triangles	10000	245	235.12	284.73

The results clearly show that nothing is faster than rendering nothing. It is interesting to note that even using the vertex shader to render the bees as

triangles on the GPU cannot compete with the simple point rendering on the CPU. Perhaps due to the necessary mesh creation on the CPU.

Also of interest is the fact that the increase in simulated bees is not proportionate to the time the simulation takes. The difference in number of bees is approximately 180 fold, where as the simulation time is a little over quadrupled. Probably because the simulated bees aren't the bottleneck in the simulated run with 121 bees.

7 Program Description

The program which simulates bee foraging behavior is written in standard C++ programming language [25] and uses OpenGL [3], as well as GLUT [22] and GLEW [18]. The implementation has been developed using Visual Studio 2005 Professional [20]. The framework has been written from the ground up apart from the aforementioned API's and two classes. The program uses Berghen's basic XML parser for C++ [2] and a slightly modified version of Walsh's BMP Loader [27].

Figure 9 shows an UML class diagram of the bee simulator. The figure shows classes on the CPU side and cg programs on the GPU side of the implementation. The diagram is a simplified representation of the actual program in order to make it more readable. The exact attributes and functions of every class can be seen in header file for each class included on the cd-rom.

The `main` function (included in the `AIGPU.cpp` file) sets up OpenGL, GLUT and necessary runtime variables before creating a `BeeSystem` object. The `BeeSystem` object first creates an instance of the `BeeSystemConfig` which provides `BeeSystem` with all configuration values provided by the XML parser from the file specified. The `BeeSystemConfig` uses references to the relevant runtime variables in `BeeSystem` and updates these directly. The `BeeSystemConfig` object is destroyed soon after it is created by leaving scope.

Next `BeeSystem` creates a single of instance of the `BeeWorld`, `RandomTexGen`, and `BeeReporter` classes. `BeeWorld` loads and processes a specified BMP file. Once processed, the file is loaded into a texture and ready for use in the simulation. The `RandomTexGen` continously provides the simulation with random values in a texture. The `BeeReporter` class keeps track of which bees are currently foraging which nectar source. After creating the aforementioned objects, the `BeeSystem` loads every necessary Cg program and creates a `cgrwTexture` for each Cg program.

For every fragment shader used to simulate the foraging behavior of bees (which in this case is four), the `BeeSystem` binds the necessary texture and cg program, and renders a quad to an off-screen buffer.

Note that on the source code included on the cd-rom, each model in this project has its own set of cg programs. The filenames ending with "2.cg" belong to the probability model, where as the filenames with "3.cg" endings belong to the extended model.

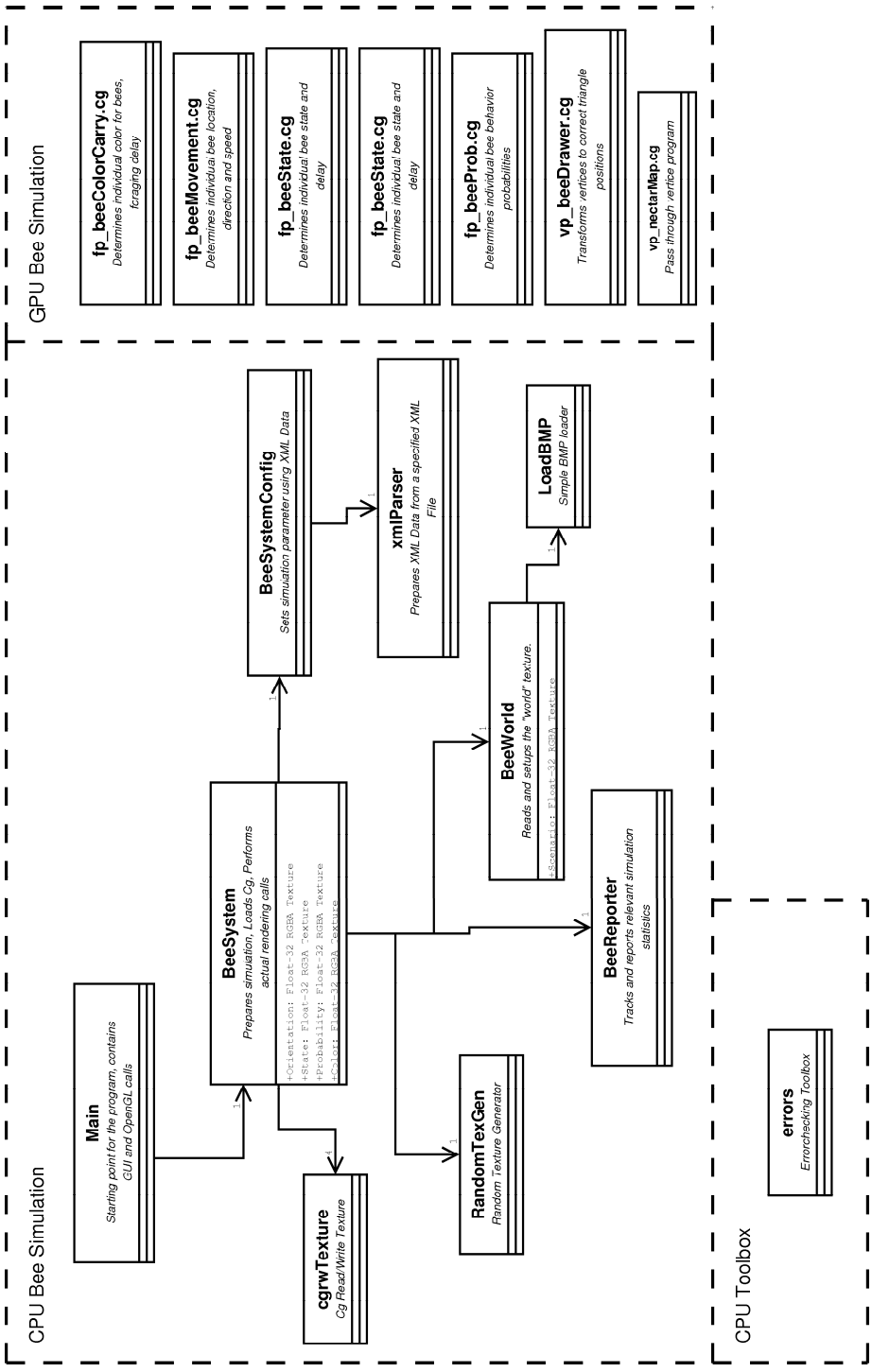

Figure 9: An UML class diagram of the bee simulator implementation. The diagram only shows the associations in between existing classes as well as a few select attributes. Cg programs included in the diagram are denoted by the ending ".cg". Cg programs are not classes, but a vital part of the program structure, and therefore included in the diagram.

8 Results and discussion

In the following sub-sections results (and comparisons) of the implemented bee foraging simulator are presented and discussed. As described in Section 5.2, Seeley et al. performed an experiment with bees foraging at two seperate nectar sources. Each located approximately 400 meters to the south or north of the hive, respectively. In the figures presented in this section, the bees foraging at the north source are abbreviated with "NF" (north foragers), and those foraging at the south source with "SF" (south foragers).

In addition to the results, Section 8.4 details possible improvements and future work with regard to the project.

8.1 Probability Model

This section presents the results of the probability model compared to Seeley et al.'s own model results and real empirically observed data in Figure 10 and 11 respectively. The probability model is based upon the probabilities presented in Seeley et al.'s own article [14]. The state diagram for the model is shown in Figure 4. Section 5.1.1 contains tables with the exact delays and probabilities used in the model.

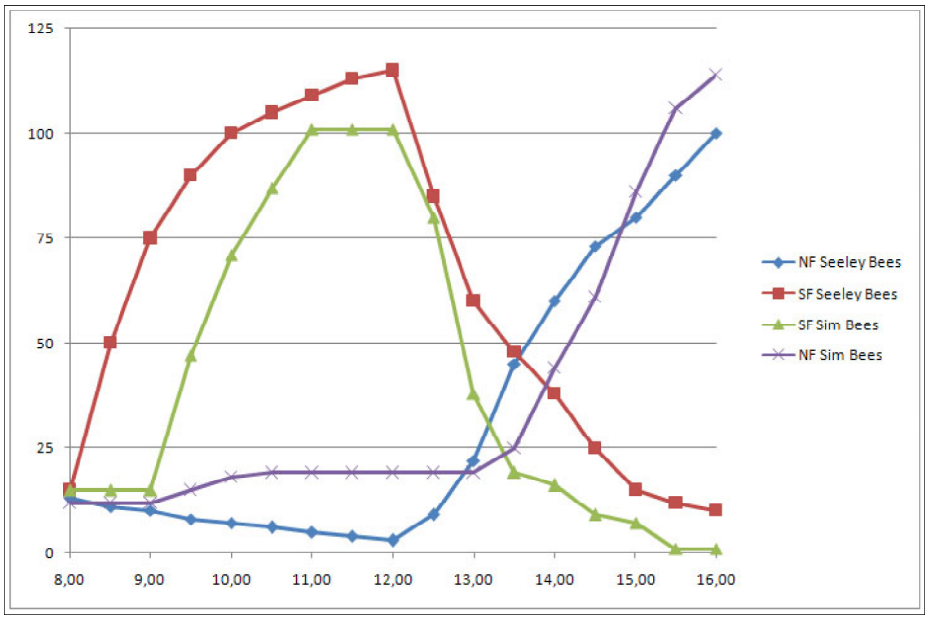

Figure 10: First simulated run of the probability model with the results of Seeley et al.'s model presented in their article [14]. The figure shows the amount of foragers recruited and foraging at either the north source (abbreviated as NF) or south source (abbreviated as SF), for both models. This run took 70 seconds to complete.

Seeley et al. state in their article that they obtained the results shown in Figure 10 with 101 foragers. I assume that they are referring to the amount of

foragers added to the 12 and 15 trained to feed at the north and south source respectively. This means they tested their model with a total of 128 foragers. The test runs I performed included 121 foragers, which is the closest number to 128 my implementation can simulate. The cause of this and a solution is described in Section 8.4.

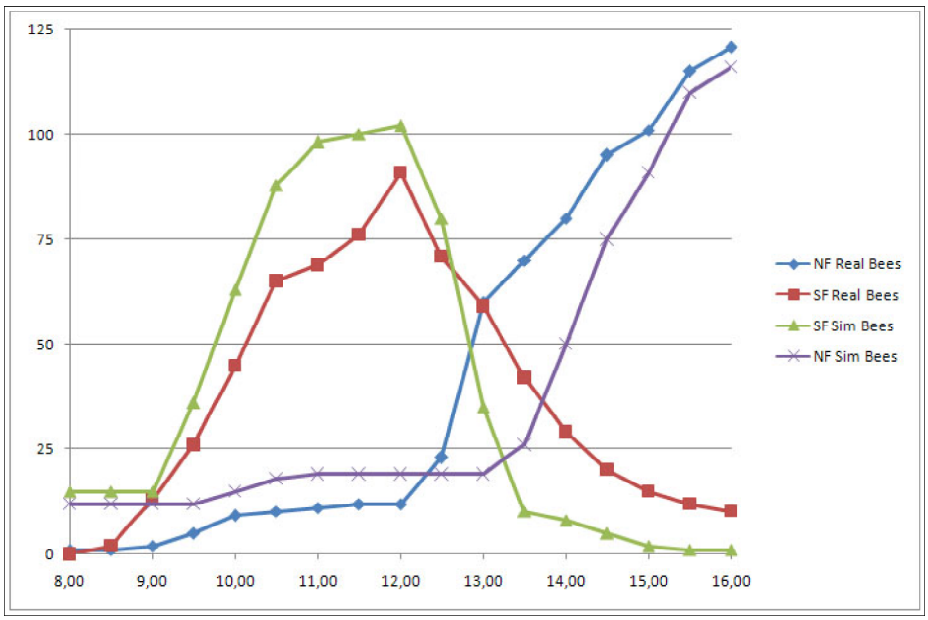

Figure 11: Second simulated run of the probability model with the results of Seeley et al.'s observed bees presented in their article [14]. The figure shows the amount of foragers recruited and foraging at either the north source (abbreviated as NF) or south source (abbreviated as SF), for both the model and observed data. Note that the pre-trained bees are not counted in Seeley et al.'s data. This run took 70 seconds to complete.

The results from the probability model and Seeley et al.'s model (Figure 10) and the empirically gathered data (Figure 11), are comparable with a few notable differences:

- **Nectar find delay** - As described in Section 5.1.1, the probability model includes a delay of one hour for every bee observing a dancing bee and trying to find the advertised source of nectar. These delayed bees cannot be influenced by other bees or perform any action while being delayed. The result is that new forager bees are first visible after 9:00.
- **Late increase of NF** - Compared to Seeley et al.'s model, the foragers in the probability model the forages at the north source do not increase in size dramatically before 13:00. This is related to the one hour delay described above. Any bees of the switching from the now poor south source to the now lucerative north nectar source will experience the one hour delay. Hence, forager increase at the north source is approximately one hour delayed compared to Seeley et al.'s model.

- **No decrease of NF** - The reason for this descrepancy is either due to no foragers abandoning the poor north nectar source, or too many forager bees being attracted to the source, replacing the ones that do abandon it.

  Seeley et al. describe the observed average recruitment rate per nectar profitability in their article [14]. This probability has only indirectly been implemented in the probability model due to the exact bee-dance times presented in the same article. The actual implementation of this recruitment limit would involve the CPU more than is intended in this project. I believe that the lack of indirect implementation is the cause of the difference between Seeley et al.'s model and the probability model.

  Conversely, compared to the empirically observed data, the simulation is a closer match as the real bees also increase in size at the north source.

- **Complete abandonment of source by SF bees** - Once the concentration changes at the south source, the SF bees are quick to abandon it. In both simulated runs, the SF bees are reduced to zero which is not the case with the real bees.

  In the first run shown in Figure 10 one could be lead to believe that the reduction of the SF bees to zero is due to them never gaining enough foragers before and until 12:00.

  However, in the second run presented in Figure 11 the real SF bees never reach the same size as the simulated bees yet still have bees foraging at the south source at 16:00. I believe the reason for this discrepancy is either due to the bees being able to visit the source too often, thereby increasing their abandonment tendancy too much, or because the bees which have been trained to feed at the source have an extra strong affinity for it. Perhaps they are only willing to forage at a new location once the source has completely ceased to provide nectar or when the hive is in dire need of it.

The test runs performed with the probability model showed only small variations to the extent between the two test runs shown in Figure 10 and Figure 11.

8.2 Extended Model

In this section the results of two test runs with the narrow and extended vision in the extended model is presented and compared to the empirical data collected by Seeley et al. in their experiment [14]. Finally, two test runs using a regular sized colony are presented and compared.

The extended model is a modified version of the probability model with bee behavior directly related to the current state of the simulation and not based on pure probabilities. The state diagram for the model is shown in Figure 5.

8.2.1 Extended Vision

The colony used in the experiment presented by Seeley et al. was comprised of approximately 4200 bees. Research has shown that in a typical colony, approximately one quarter will be involved in food collection. Therefore the simulation runs presented below are simulations involving exactly 1024 forager bees.

Figure 12 and 13 show the results of the extended model with extended vision. The results are comparable with Seeley et al.'s observed data with some notable differences:

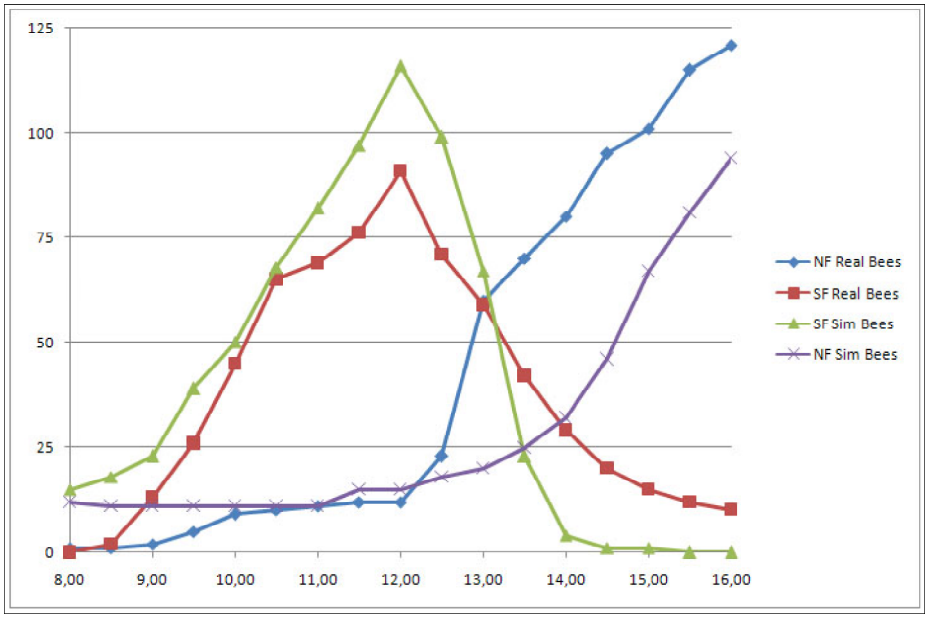

Figure 12: First simulated run of the extended model (extended vision) with the results of Seeley et al.'s observed bees presented in their article [14]. The figure shows the amount of foragers recruited and foraging at either the north source (abbreviated as NF) or south source (abbreviated as SF), for both the model and observed data. This run took 131 seconds to complete.

- **Higher number of SF Bees** - In both simulations using the extended model shown in Figure 12 and 13, the number of simulated SF bees is generally higher all through the simulated run when compared to the real bees. This can at least partially be explained by the fact that Seeley et al. do not include the bees with pre-experiment knowledge of the two sources in their statistic.

  I have decided to include these my statistic because even the foragers with existing knowledge of the sources sometimes decided to abandon their source.

- **Complete abandonment of source by SF bees** - As with the probability model, all the SF bees eventually abandon their source. I expect the cause for this to be the same as with the probability model.

- **Slow increase of NF bees** - The simulated NF bees increase slowly in both runs when compared to the real NF bees.

  This could be due to the SF bees not abandoning their source fast enough, which seems unlikely because they abandon it much quicker in the simulation than in the real world. I believe it is related to the simulated bees

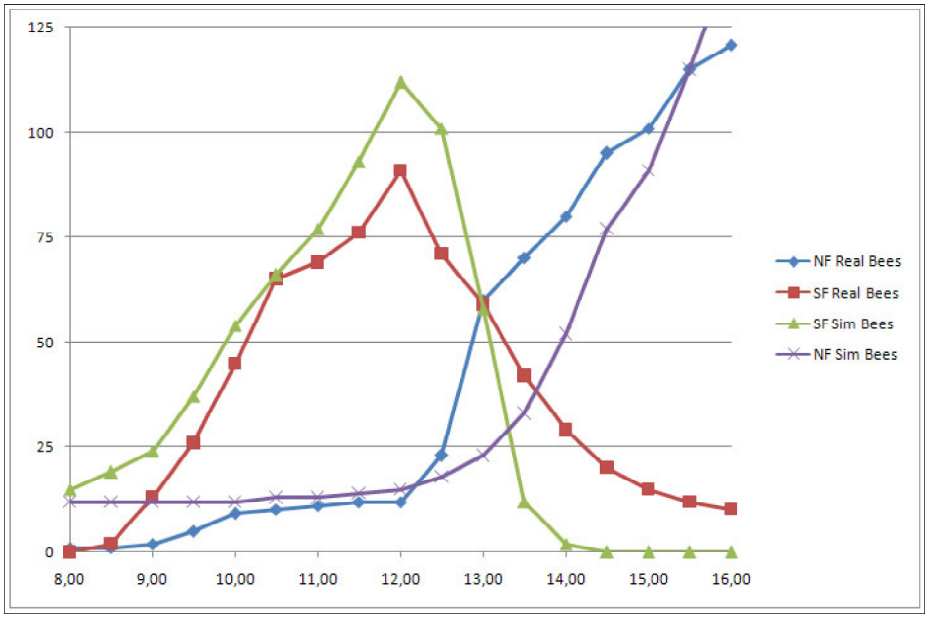

Figure 13: Second simulated run of the extended model (extended vision) with the results of Seeley et al.'s observed bees presented in their article [14]. The figure shows the amount of foragers recruited and foraging at either the north source (abbreviated as NF) or south source (abbreviated as SF), for both the model and observed data. This run took 129 seconds to complete.

difficulty in finding the north source. The NF bees have in all likelyhood searched for the north source before ending up as recruits for the south source. As previously stated, bees will make an average of 5 trips before finding an advertised source and will in many cases search for a different source every time. The real bees will gain knowledge of the north source before commiting to it later. The simulated bees on the other hand will search for the north source as if they've never searched for it before. Hence, they will have a more difficult time locating it even once it becomes more profitable after 12:00.

- **Low early increase of NF bees** - In contrast to the probability model and the real bees, the simulated bees in the extended model do not increase in size at the north source of nectar.

  The reason for this is twofold. First, because the extended model uses both a sigmoid function and the probabilities presented by Seeley et al. in their article [14] to calculate whether a bee should dance or not, the chances of dancing are lower for the low quality source. As a direct consequence, there is less chance for a following bee to locate a dance advertising that source.

  Second, a bee will not commit to a source until it has found and foraged at it. There is a large chance that a bee will follow several bees before finding its first source to forage at. Following several dances only increases the

chance of a dance following bee to follow a bee dancing for the profitable source of nectar, which is the case for the south source before 12:00.

As with the probability model, the extended model with extended vision showed no variance of particular interest.

8.2.2 Narrow Vision

The result of a single simulated run of the extended model with narrow vision is show in Figure 14.

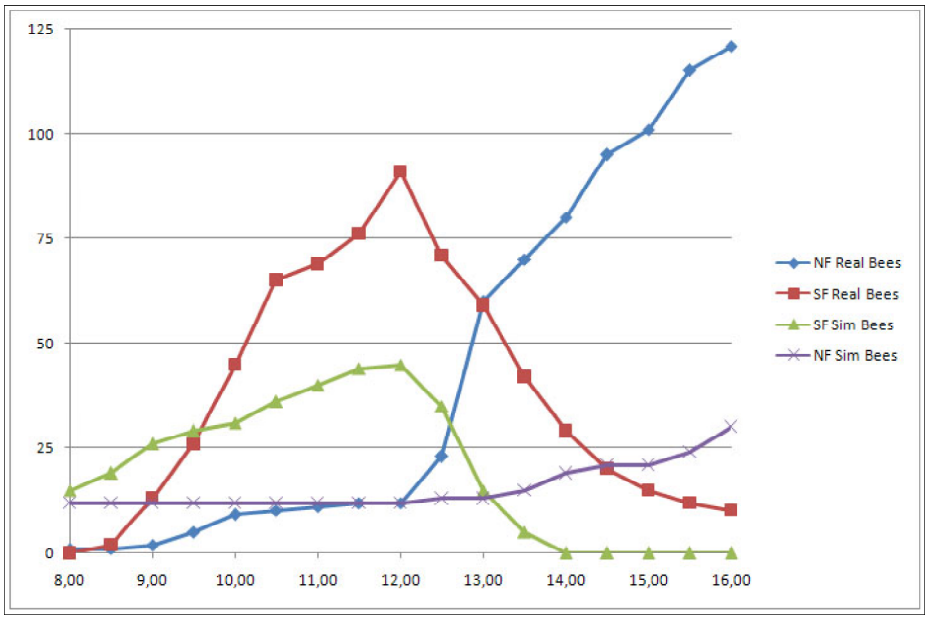

Figure 14: A simulated run of the extended model (narrow vision) with the results of Seeley et al.'s observed bees presented in their article [14]. The figure shows the amount of foragers recruited and foraging at either the north source (abbreviated as NF) or south source (abbreviated as SF), for both the model and observed data. This run took 135 seconds to complete.

Not surprisingly, the bees have a much harder time finding the advertised sources and as a result the amount of foragers on both sources increase much slower than in previous test runs. The reason for the even slower increase of NF bees after the 12:00 hour mark is probably due to a specific amount of foragers already committed to the south source and first abandoning it after a number of trips (once the abandonment probability is high enough). The rate of abandonment is probably not high enough for the NF bees to reach the critical mass needed in order for them increase to the number of foragers foraging at the south source at 12:00. As with the previous models, this one did not show exceptional variance.

8.2.3 Regular sized colony

An average sized colony is described by Seeley as consisting of 20,000 bees. As stated previously, approximately one quarter will be involved in food collection. Figure 15 shows a simulated run of the extended model with extended vision with 5041 foraging bees.

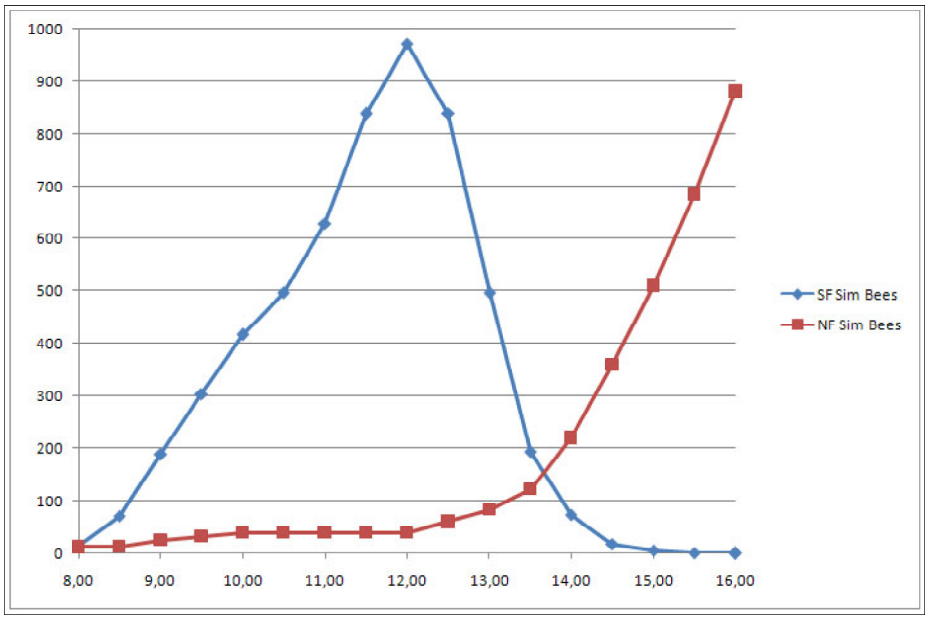

Figure 15: A simulated run of the extended model (narrow vision) using an average sized colony number of foragers. The figure shows the amount of foragers recruited and foraging at either the north source (abbreviated as NF) or south source (abbreviated as SF), for the model.

The results show no particular differences when scaled down and compared to the previous simulated runs using the extended model. Considering that every aspect of the simulator scales up when the amount of forager bees increase this comes as no surprise.

8.3 Performance comparison

The previous sections provided the results of the two models compared to the results presented by Seeley et al.'s model and their empirically observed data. In this section, the speed of the two models will be compared with eachother with and without select functionality that impacts performance.

The table below shows a standard run (using the GPU triangle display) for each of the existing models and their variants:

Model Version	Nr. of Bees	Sec. Elapsed	Average FPS	Max FPS
Probability	1024	72	789.93	902.55
Extended (Nar. Vision)	1024	127	452.19	574.38
Extended (Ext. Vision)	1024	143	400.95	504.92

Clearly there is a performance difference between the two models. I believe the main cause for this is not the more realistic calculations, but the dance following algorithm. In the probability model, a dance following bee would search for a dancing bee once every simulation step. In the extended model, a dance following bee searches for a dancing bee until one is found or until the GPU stops further execution. Changing the dance following algorithm to resemble that of the probability model yields the following performance in the extended model:

Model Version	Nr. of Bees	Sec. Elapsed	Average FPS	Max FPS
Extended (Nar. Vision)	1024	84	688.32	709.65
Extended (Ext. Vision)	1024	114	508.20	555.67

A clear difference in performance. It is interesting to note that the extended vision which consists of an extra five texture look ups has such a huge performance impact. An approximate difference of 180 frames per second.

It is also clear that even with the same bee dance following algorithm, the extended model (with narrow vision) still requires more time than the probability model. The extended model performs a few additional texture look ups for bee individual variables. The individual threshold values for each bee for example. I believe the additional texture look ups are the main cause of the performance difference between the two models.

8.4 Future exploration and improvements

This section will detail future improvements/functionality as well as future work that could be performed in conjunction with this project.

Where appropriate, I will specify what impact the inclusion of this feature might have for the simulation.

- **Further experiments** - With the models presented in this project, there are certain experiments that are of particular interest when compared to real bees as well as other models.

  In comparison to other existing models, the ones presented in this project are arguably less complex (behavior wise) and probably less reliable. However, a single test run (without any display) of a colony of average size (5041 foragers) takes 6 minutes and 43 seconds to complete. I believe that the work presented in this project can be used as preliminary test-bed for new behavioral additions to a set of behavioural rules.

  The following set of scenarios are difficult to perform in the real world and are therefore of additional intrest. Be aware that some of the mentioned scenarios would require an alternate setup (nectar locations) or additional rules not implemented in this project.

- **Scouting behavior** - As mentioned several times during this project, the behavior of the spontaneous scout has at the time of writing not been properly documented. Although considering some of the recently published research [11] I have no doubt this will happen sooner than later. Nevertheless, various spontanous scouting algorithms could be simulated in this project and compared with empirical data from a bee colony placed at a new location with no previous knowledge of the surrounding sources.
- **Forager group size** - Varying the forager group size could provide interesting results in relation to the scouting behavior. A small bee colony placed in unknown surroundings may not be able to locate necessary provisions fast enough when compared to a large hive.

  Given that the simulation included additional rules for collision avoidance, the forager size would probably have a big impact on the foraging, dancing and delivery time. All of these factors would be interesting to take a close look at while varying the colony size.
- **Impact of feedback removal** - Prohibiting real live bees from providing the hive with either positive or negative feedback requires nothing less than the removal of the bees providing the feedback. However, using the simulated bees, it is possible to simply remove their ability to communicate discoveries to each other, to see how well the hive can manage without its recruiting capability.
- **Identical Vs. Different dancing thresholds** - Thenius et al. recently conducted research [6] regarding the individual dance thresholds of bees. The compared the performance of a hive populated by bees with identical thresholds to the a hive populated by bees with differing thresholds.

  The same experiment could be performed using this simulator in order to compare the results and gain new insight into how the models differ and if the findings could apply to real bees.

- **Comparisons with more empirical data** - Almost every model I have researched has compared its results to the ones presented by Seeley et al. in their article [14]. Perhaps because it is the only/most comprehensive study available at the time of writing.

  Nevertheless, comparing the results presented in this project with further empirical data would help (in)validate my findings.

- **Further rules and behaviors** - The results generated by the simulator are not an exact match when compared to the empirically gathered data. Further work on the simulator to introduce more realistic behavior is required for a closer match to the gathered data.

- **Further tests to pinpoint performance differences** - In addition to the performance comparison in Section 8.3, it would be interesting to perform more precise tests in order to pin point what causes the biggest loss of performance while simulating AI on the GPU. Various online sources state that multiple texture look-ups has a large performance impact which I believe is true. Running two duplicate programs, one using data from a

texture look-up and the other using constant data would reveal how large of a negative impact this causes.

However, considering the speed at which the graphics industry moves at the time of writing the results provided by such a comparision may be redundant only a few months later.

- **Performance comparison with identical CPU model** - In order to calculate the exact gain by performing the simulation of bees on a GPU, an identical CPU model is needed.

  Creating such a model and running thorough tests will help estimate how much faster GPU versions of similar simulations can be expected to become.

- **Improved bee memory** - In the extended model presented in this project, a bee is only able to remember one single nectar source. If it decides to abandon its current one or observes a dance advertising a different source than the one it has been seaching for, it will forget all about its current source.

  Existing research [24] as well as the results presented in this section, leads me to believe that real remember at least more than one source that they have visited. Implementing this ability would yield more realistic results.

- **Improved random texture generator** - The current version of the random texture generator, generates a new random texture for every simulated step. An improved version optimized for speed would pre-generate a certain amount of random values in an array which could be iterated through with a window the size of the necessary texture. Although random numbers would be "passed down" from bee to bee, it would increase output immensely. Instead of iterating and generating random values through an entire array of the necessary texture size, the improved random texture generator would only need to move one pointer.

  Without any conclusive testing, the bottleneck seems to be the cg programs (on the GPU) and not any of the processing being performed on the CPU. Hence, this improvement would probably not yield any increase in performance.

- **Less hard coding** - Certain aspects of the simulator have been hard coded due to the time constraints involved with the project. For example, the number of bees with pre-simulation knowledge of a nectar source is static. A more general version would lead to easier testing of alternate scenarios, increasing the simulators robustness.

  However, lack of generality does not lessen the validity of the results presented in this article.

- **Number of simulated bees** - The current implementation of the bee foraging simulator requires the number of simulated bees to be the power of two of a whole number. For example, 100 simulated bees is a valid number, because it is derived from 10^2. 101 simulated bees is not valid since it is derived from 10.0498756^2, and 10.0498756 is not a whole number.

This limitation is caused by the square textures used to represent and simulate the bees. The problem can either be solved by using rectangular textures or by allowing a texture to contain pixels which are not utilized in the simulation. Because a cg program automatically executes the fragment shader for every drawn pixel, the rectangular texture solution would be preferable.

8.5 Summary

Assuming that this is the first project of its kind, attempting to simulate AI (albeit simple AI) on a GPU, I believe the project has proven that this is possible.

Two moderately different models of foraging behavior have been introduced in this project. One of them (the extended model) running under more realistic circumstances, and both producing results similar to empirically gathered bee foraging data.

The models are still lacking certain behaviour rules in order for them to produce as realistic results as other existing models. These additional behaviour rules will in general require more complex programming on the GPU than on the CPU. However, graphics hardware producers are eager for developers to use their GPUs for other purposes than just processing graphics. New developer tools are developed and released regularly for the available graphics cards. At the same time, the GPUs are becoming more powerful and versatile. As time progresses, I believe that programming GPUs will become easier and more accessible.

I have no doubt that we will see more projects similar to this one and other general purpose programming projects on the GPU, in the future. To conclude this project, I have included a screenshot of the simulated bees, which are shown in Figure 16.

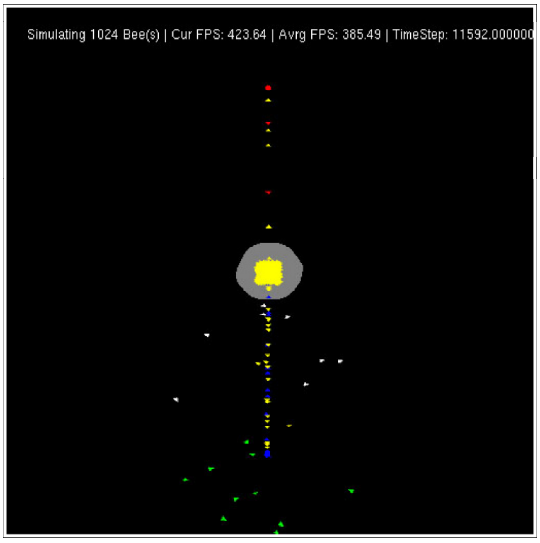

Figure 16: A screenshot of the bee foraging simulator using the extended model (extended sight).

References

[1] Scott Bauer. Photo of honey bee. USDA/Agricultural Research Services, Bugwood.org.

[2] Frank Vanden Berghen. Small, simple, cross-platform, free and fast c++ xml parser v2.23. `http://www.applied-mathematics.net/tools/xmlParser.html`, 2007.

[3] OpenGL Architechture Review Board. Opengl. `http://www.opengl.org/`.

[4] David J.T. Sumpter & D.S. Broomhead. Formalising the link between worker and society in honey bee colonies. *MABS98, LNAI*, 1534:95 110, 1998.

[5] François-Xavier Dechaume-Moncharmont et al. The hidden cost of information in collective foraging. *Proceedings of the Royal Society B*, 272:1689 – 1695, 2005.

[6] Ronald Thenius & Thomas Schmickl & Karl Crailsheim. The dance or work problem: Why do not all honeybees dance with maximum intensity. *CEEMAS 2005, LNAI 3690*, page 246 255, 2005.

[7] Ronald Thenius & Thomas Schmickl & Karl Crailsheim. Economic optimisation in honeybees: Adaptive behaviour of a superorganism. *SAB 2006, LNAI 4095*, page 725 737, 2006.

[8] T. Schmickl & K. Crailsheim. Costs of environmental fluctuations and benefits of dynamic decentralized foraging decisions in honey bees. *International Society for Adaptive Behavior*, 12(3-4):263 – 277, December 2004.

[9] Han de Vries & Jacobus C. Biesmeijer. Modelling collective foraging by means of individual behaviour rules in honey-bees. *Behavioral Ecology and Sociobiology*, 44:109 – 124, 1998.

[10] Jacobus Christiaan Biesmeijer & Han de Vries. Exploration and exploitation of food sources by social insect colonies: a revision of the scout-recruit concept. *Behavioral Ecology and Sociobiology*, 49:89 – 99, May 2001.

[11] Elizabeth A. Capaldi et al. Ontogeny of orientation flight in the honeybee revealed by harmonic radar. *nature*, 403:537 – 540, Febuary 2000.

[12] J. R. Riley et al. The flight paths of honeybees recruited by the waggle dance. *nature*, 435(7039):205 – 207, May 2005.

[13] Madeleine Beekman et al. What makes a honeybee scout? *Behavioral Ecology and Sociobiology*, 61:985 – 995, 2007.

[14] Thomas D. Seeley et al. Collective decision-making in honey bees: how colonies choose among nectar sources. *Behavioral Ecology and Sociobiology*, 28:277 – 290, 1991.

[15] Jonas Flensbak. Boids - flock behavior based on influence maps, 2007.

[16] III et al. John J. Bartholdi. The pattern and effectiveness of forager allocation among flower patches by honey bee colonies. *Journal of theoretical Biology*, 180:23 – 40, 1993.

[17] Lutz Latta. Building a million particle system. `http://2ld.de/gdc2004/`, 2004.

[18] Milan Ikits & Marcelo Magallon. The opengl extension wrangler library (glew). `http://glew.sourceforge.net/`.

[19] Axel Michelsen. *Honningbiens Dansesprog: Signaler og Samfundsliv.* Danmarks Biavlerforening, 1992.

[20] Microsoft. Visual studio 2005 professional. `http://msdn2.microsoft.com/en-us/vstudio/aa718668.aspx`, 2005.

[21] Stuart Russell & Peter Norvig. *Artificial Intelligence: A Modern Approach (2nd Edition).* Prentice Hall, 2002.

[22] Mark Kilgard (ported by Nate Robins). The opengl utility toolkit (glut). `http://www.xmission.com/~nate/glut.html`.

[23] C. W. Reynolds. Steering behaviors for autonomous characters. *Proceedings of Game Developers Conference 1999*, pages 763–782, 1999.

[24] Thomas D. Seeley. *The Wisdom of the Hive.* Harvard University Press, 1995.

[25] Bjarne Stroustrup. The c++ programming language. `http://www.research.att.com/~bs/C++.html`.

[26] Karl von Frisch. *The Dance Language and Orientation of Bees.* Harvard Univ. Press, Cambridge, Massachusetts, 1967.

[27] Karl Walsh. Bmp loader. `http://gpwiki.org/index.php/LoadBMPCpp`, 2004.

[28] WikiPedia. Geforce 7 series. `http://en.wikipedia.org/wiki/GeForce_7_Series`, 2007.

[29] WikiPedia. High level shader language. `http://en.wikipedia.org/wiki/High_Level_Shader_Language`, 2007.

[30] WikiPedia. Reactive planning. `http://en.wikipedia.org/wiki/Reactive_planning`, 2007.

[31] WikiPedia. Sigmoid function. `http://en.wikipedia.org/wiki/Sigmoid_function`, 2007.