THE SOPRANTS:
CONCEPTUAL AND TECHNICAL FRAMEWORK FOR A 3D INTERACTIVE VIDEO GAME

A Thesis

by

TATSUYA NAKAMURA

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2006

Major Subject: Visualization Sciences

THE SOPRANTS: CONCEPTUAL AND TECHNICAL FRAMEWORK FOR A 3D INTERACTIVE VIDEO GAME

A Thesis

by

TATSUYA NAKAMURA

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee,	Carol LaFayette
Committee Members,	Bradleigh Vinson
	Vinod Srinivasan
Head of Department,	Mardelle Shepley

August 2006

Major Subject: Visualization Sciences

ABSTRACT

The Soprants: Conceptual and Technical Framework for a 3D Interactive Video Game. (August 2006)

Tatsuya Nakamura, B.S., Kyoto University;

M.S., Kyoto University

Chair of Advisory Committee: Prof. Carol LaFayette

This thesis covers the design of an interactive 3D video game with certain unique features and demonstrates the design through a prototype implementation. Insect characters are modeled after human characters and set in a game story. The ants in the game behave similar to leaf-cutter ants. A 3D game environment based on a real ant colony nest is created and used for prototyping the game. Insect behavior based on behavior of real ants is implemented in an interactive 3D environment. The cinematic scenes and the trailer of the game are created to present the game story.

To my family

ACKNOWLEDGMENTS

I would like to thank my committee chair Prof. Carol LaFayette and Dr. Bradleigh Vinson for their encouragement and guidence. I would also like to thank Dr. Vinod Srinivasan who served on my committee for being prompt and supportive.

I cannot fully express my gratitude to my parents and family for blessing me with their love and support.

TABLE OF CONTENTS

CHAPTER Page

LIST OF FIGURES

FIGURE Page

CHAPTER I

INTRODUCTION

The aim of this research is to design an interactive 3D video game with certain unique features and to demonstrate the design through a prototype implementation. Three key features of game design in this research are:

1. A game story in which insect characters are modeled after human characters.
2. A 3D game environment based on a real ant colony nest.
3. Insect behavior in an interactive 3D environment based on behavior of real ants.

Insect characters are often seen in many kinds of interactive and non-interactive stories. Their appearances and behaviors are are well known and easily characterized. Ants are especially known for their characteristic group behavior. Some entomologists have proposed that an ant colony is best viewed as a "superorganism" because of their dominance as a group [1]. Ants are to be found everywhere and are categorized as social insects.

There are some interactive video games that explore the behavior of ants and other insects. They can be divided into three categories, where a single game title is selected to represent the games in that category:

1. Simulation game: SimAnt^{TM}(1991)
2. Strategy game: Empire of the Ants^{TM}(2001)
3. Platform game: A Bug's Life^{TM}(1998)

The journal model is *IEEE Transactions on Automatic Control.*

The first two games include types of ant behavior, which will be discussed in the next chapter. The goal of both games is to survive as an ant or a group of ants as long as possible in the rule system, which simulates real world interaction. The third is based on the movie of the same name. It requires the player to navigate a character through various puzzles using a player's wit and skill with the joystick, as with other general platform games, such as Nintendo's Super Mario Bros^{TM}(1985).

This research project uses the category of the adventure game to create a visual, interactive story. The game requires a player to navigate an ant character through an original story for the game, but does not require fast reflexes. The story is built upon a player's actions until the game is completed.

The game will provide an ant s experience in an interactive 3D environment through a visual narrative about family life within a colony. The leaf-cutter ant's unique behavior of growing fungus will be an interesting part of the game story. The game player will be easily involved in the game experience.

The game is about a conflict between two ant colonies. A player becomes an ant who is working as a spy. The ants in the game behave similar to leaf-cutter ants: they are cultivating a fungus inside their nest that is grown to feed the queen and her brood. The fungus grows on bits of leaves cut and brought into the nest by foraging workers. The ants form a family and follow a family code, just as a human family does.

The ant nest as a game environment will be based on collected data of a leaf-cutter ant colony. The game player explores the game environment as a leaf-cutter ant. The player's role as a spy in the ant's family is an important part of the game. Through the game experience the player learns about an ant family that is very similar to a human family.

A. Problem Statement

The purpose of this research is to create a conceptual and technical framework for developing a 3D interactive video game. The game story and navigation will be developed as the conceptual framework. The technical framework will involve:

1. The 3D modeling of an ant character
2. Creating a 3D interactive animation with the ant behavior
3. Prototyping the game with 3D game development tools

To create ant behavior patterns in a 3D interactive environment, two interactive graphics techniques will be used: a crowd controlling system and an autonomous agent system. A crowd controlling system will be used for managing the characters' movement. An autonomous agent system will be used for creating non-player characters in an interactive 3D game and an AI (artificial intelligence) solution is often used in its implementation.

CHAPTER II

PREVIOUS WORK

There have been many approaches to creating insect or animal behavior for both computer animated feature films and video games. In the film industry, there is high demand for animating hundreds of background characters efficiently and effectively. Some autonomous crowd systems have been developed to create computer generated feature films with 3D graphics technology.

Pixar used a proprietary crowd system for animating hundreds of ants in the feature film, "A Bug's Life" (1998) [2]. PDI/DreamWorks also developed its own in-house crowd systems that included a rule-based simulator to create crowds of ants in its first feature-length animation, "Antz" (1998) [3]. Both crowd systems are fully customized in their animation pipelines and also control the behavior of the ants animated with crowd simulators. See Fig. 1.

In the production of "A Bug's Life", a visual-effects team controlled the movement of the environment and crowds of ants with Pixar's proprietary crowd system. Animators created the performance for the characters and provided animation cycles for the crowds. The crowd simulator used procedures to mix and match bits of keyframe animation and rules to help guide the behavior of ants. With the simulator they could create a line of dancing ants in which each ant does something slightly different.

For the production of "Antz", PDI/DreamWorks created two types of crowd systems. One system blends a mixture of body types and motions and is used primarily for crowds of fewer than 50 ants. The second system gives animators less control and more automation and is used for larger crowds. The second crowd system used for "Antz" is a rule-based simulator. The behavioral simulation is loosely based on Craig

Reynolds‘ flocking system: basically the ants avoid obstacles, avoid each other, and have a goal they try to get to in the environment. The simulator can work with ants placed on a plane with height fields; this plane is then mapped onto the geometry of the environment. These two crowd systems provide flexibility for both small and large size crowds. See Fig. 1.

The most recent 3D animated feature film created with a crowd system is “Robots” (2005) [4]. Blue Sky Studio used its own layered crowd system for the film and its basic idea of the crowd system is to move characters on a ground plane while they execute animation cycles, with goal seeking and collision avoidance. The system was implemented as a plug-in to the Maya^{TM}software package. The system was a custom field that operates on a particle system, similar to gravity or turbulence. The plug -in was written in C++ using the Maya^{TM}API.

(a) (b)

Fig. 1. (a) “A Bug’s Life” (1998), (b) “Antz” (1998).

In the computer games industry, autonomous agents for interactive game environments have been widely used. Agents are used to control computer characters with AI techniques. Character behavior is one of the important aspects of gaming

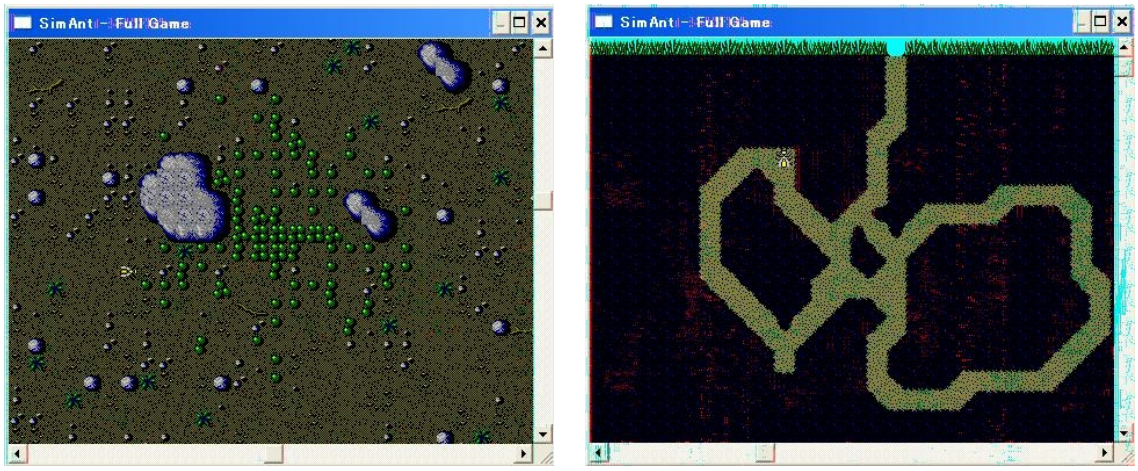
technology used to create a more realistic gaming experience. There are a couple of games that have implemented insect behavior in interactive game environments.

Maxis' "SimAnt™"(1995) simulates ant behavior based on real ant biology. In the game, the player experiences life as an ant. The player digs tunnels and collects food to hatch eggs in the nest. The player can also recruit other ants to form an army to fight with opponents. The ants in the game have a simple class system which includes workers, soldiers, and the queen ant. Will Wright, the creator of "SimAnt™," suggests that the educational objective is to teach players about the emergent behavior of multi-cellular organisms like ant colonies [5]. See Fig. 2.

The game provides two different kinds of views: one is a side view of underground ant colonies, the other is an overhead view of a yard where the ants establish their colonies. The player establishes a black ant colony in a small patch of yard. The computer establishes a competing red ant colony in the same patch. The ultimate aim of the game is to spread throughout the yard by producing young queens and battling against the red ants. The player has direct control of a single ant at a time and may switch control to a different ant at any time. The player's ant may influence the behavior of other black ants by leaving pheromone trails to destinations such as food and enemy ant colonies.

Strategy First's "Empire of the Ants™" (2001) [6] is a real-time strategy game based on the science fiction book *Empire of the Ants* (1991) authored by Bernard Werber [7]. The story behind "Empire of the Ants™" comes straight from Werber's novel. The Western Empire of Russet Ants is trying to expand its territory while protecting its borders from hostile invaders.

Players assume the role of an ant commander who must oversee the management and expansion of a colony of ants. The game includes a series of missions with a variety of objectives that range from collecting food to fighting an enemy. The



(a) (b)

Fig. 2. Screen shots from "SimAnt^{TM}." (a) An ant is digging tunnels in the nest. (b) Ants are collecting food out of the nest.

concept and design of the game are very similar to those of other real-time strategy combat games. Many of the ants' function in the game like classic real-time combat units.

The game provides a realistic 3D game environment in both appearance and behavior, with many different kinds of insects. Maps include a variety of woodland and rural environments, all seen from a movable, top-down, 3D perspective. In addition, the program employs an underground viewpoint that allows players to examine a series of tunnels and chambers from the inside of ant hills, hives, and nests. See Fig. 3.

Disney Interactive's "A Bug's Life^{TM}" (1998) is based on the Pixar movie of the same name. A player becomes the ant "Flik" and experiences his adventure in the bugs' world [8]. Many other insect characters from the movie also show up in the game and help or hinder a player depending on their roles.

(a) (b)

Fig. 3. Screen shots from "Empire of the Ants™." (a) The player is exploring the 3D game environment around the nest. (b) Its opening cutscene shows a scene inside the nest.

The game environment is fully generated with 3D graphics and a player can walk through the world from Flik's point of view. Although the graphics resolution is not as high as that of the movie, the game presents the fascinating bugs' world with interactivity.

CHAPTER III

METHODOLOGY

There are three stages in my research procedure: the first stage is developing the story; all characters and environments are designed for prototyping the game in this stage. An important part of the story is character development. The characters for this game are modeled after the human characters in the HBO drama series, "The Sopranos." The second stage is developing the interactive game demo, including 3D ant nest modeling, pheromone based foraging, prototyping, and "cutscenes." A cutscene is "a film in a game" and an important part in the storytelling of an interactive game. A cutscene is either a film/video or a 3D animation [9]. There are two different types of animated cutscenes: one is a fully pre-rendered CGI movie clip and the other is a real-time 3D graphics clip rendered by a game engine.

The third is developing the game trailer, which includes screen captures of the interactive demo and edited cutscenes.

A. Story Development and the Game Design

1. The Ants' Family and Their Business

A mafia family story is developed as a background theme. In the story, two ant colonies become two mafia groups and the game player is a spy for one of the families. The goal of the game is to observe ants in the opposing family and to collect information about their activity.

The game has three different phases and the player is to complete a different mission in each phase:

The First stage The player explores the underground nest and installs wires to

transmit the ants activity to a location outside of the nest. The purpose of this mission is to learn about the structure of the nest and to meet with some ants who lead their family business.

The Second stage The player works as a soldier ant and helps forage in order to observe enemy activity, which is reported to an agent. The purpose of this mission is to collect information about growing fungus and hatching eggs. The game field in this stage is located between the nest and the tree where ants collect leaves to grow fungus. The highlight of the second mission is to meet with the boss of the nest - the queen ant.

The player detects the red ants' secret activity similar to the drug dealings in the human society which is one of the mafia activities in "The Sopranos". The red ants grow the mushroom in their fungus and sell it to the other ants as a drug. The player finds that the red ant mafia has a secret route to bring their mushroom through this mission. They are a drug-trafficking criminal organization, which causes many mushroom addicts in the black ant society.

The Third stage After the second mission is complete, one of the under-bosses of the black ant family is accidentally killed. They suspect that someone outside the family has killed their under-boss, so they prepare for war against the red ant family. The purpose of the third mission is to collect information about the army of the red ants before the war begins. At the end of this mission, the war between the black ants and the red ants occurs. The result of the war depends on the activity of the player throughout the three missions.

HBO's TV drama series "The Sopranos" (1999) is referenced for character development. "The Sopranos" is the story of a mafia boss who cares for both his immediate

family and his mafia family. See Fig. 4. The game story is modeled along the plot lines of "The Sopranos": the ants have their own code and they have to follow their code to do business. Each ant has a role to play to be a member of the family.

Fig. 4. "The Sopranos" is the story of a mafia boss and his family.

2. Game Navigation

Fig. 5 shows the flow chart of the entire game. The game starts at Game Start and the player ends with Game Clear if he or she goes through the entire story without failing. Between Game Start and Game Clear, the player experiences the three interactive parts (the first, the second and the third stage) and the four intermission cutscenes.

3. Game User Interface

The game screen displays the game objects and the environment in an interactive game sequence. The game screen also provides the game user interface, which allows the player to interact with the game objects. The game user interface includes the following functions:

Dialog selection The player has multiple choices of dialog when he or she encounters a game character. The dialog selection involves clicking the text to choose

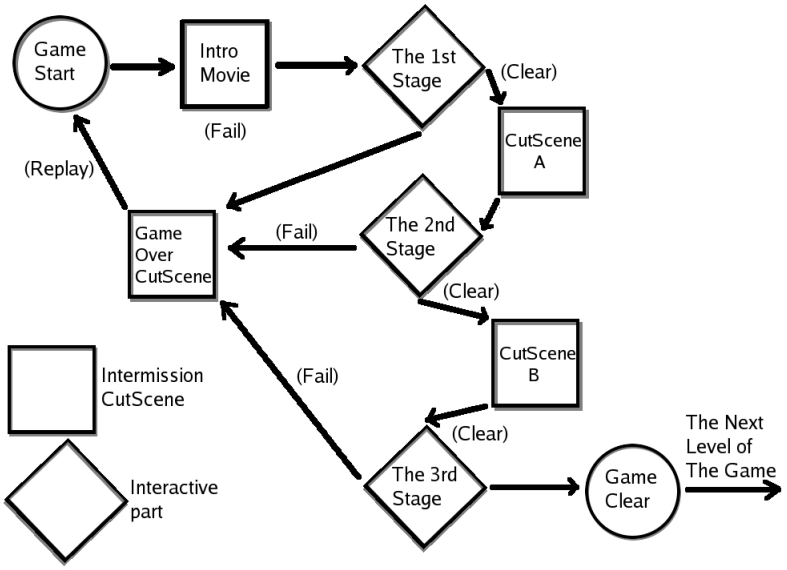


Fig. 5. The game flow chart.

the player's dialog. The dialog list displayed during the selection changes, depending on the player's situation. The player's choice of dialog creates a different narrative path.

Information The user interface displays detailed information about a game object or character when the player examines it. The information is provided via text which describes the object and/or its high-resolution model.

The game user interface is displayed in an overlay onto the 3D graphics in the game screen.

4. Cutscenes

The game includes four different cutscenes as shown in Fig. 5:

Intro Movie This cutscene describes various episodes about one or more characters in the game. One of the episodes is selected randomly from 20 different episodes at the beginning of the game. The player learns about the background story of the characters each time he or she begins the game.

Gameover CutScene This cutscene plays when the player fails his or her mission. The cutscene is about his or her death.

Intermission CutSceneA This cutscene transitions between the first mission and the second mission. In this cutscene, The player is overhearing a conversation at a meeting of the red ant mafia, using a transmitter which he or she installs during the first stage.

Intermission CutSceneB This cutscene transitions between the second mission and the third mission. This cutscene is about the accidental death of the red ant under-boss.

For prototyping the game, Gameover CutScene and Intermission CutScenes are created. These are pre-rendered 3D animation clips with game resolution characters and environment models are created and rendered with Maya™.

5. Game Trailer

In this stage, the game trailer is created with captured screen shots and cutscenes. A game trailer is created to present research results in the form of a short (60 - 90 second) sequence, similar to a music video. The trailer shows the results of 3D modeling of characters and environments developed for the game and gives an idea of the interactive game play. The soundtrack for the trailer is outsourced and created by a composer.

B. Ants' Behavior Creation

To create 3D interactive animation with the ant behavior in the game, ants' foraging is simulated. A simplified ants' foraging behavior can be described as follows:

1. Ants go out of their nest and find food.

2. Ants carry food and go back to the nest.

There are several ways to find the way home for real ants in the natural world. The method of navigating depends on the situation in which an ant may find itself. There is always a back-up method should one fail. One of them is by scent or pheromone trail; however, scent trails do not last very long as the substance eventually evaporates. Ants may also make use of the shape of the land or landmarks, the position of the sun in the sky, or the shape of the tree canopy above the nest [10]. It is known that the worker ants of *Atta texana*, whose trail marking substance contains both volatile and nonvolatile components, readily followed trails made with synthesized substances as shown in Fig. 6 [11].

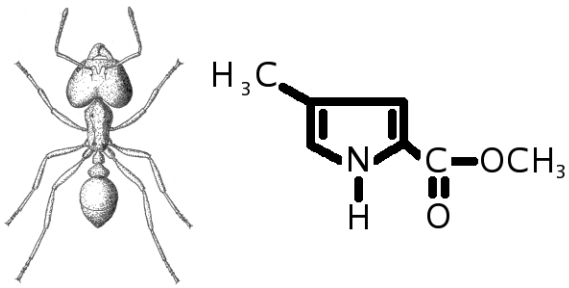


Fig. 6. Left: the worker ant of *Atta texana*. Right: its trail pheromone.

To create ant foraging behavior in the game, two different methods are used for finding the way to food and the nest. One is a famous path finding algorithm called

A* (a-star). The other is by pheromone trail. The ants that lead other ants to either food, leaves, or the nest are animated easily with the former method. The other ants are animated with the latter method and they form a crowd of foraging ants. Each ant character in the crowd is an autonomous agent which interacts with a 3D game environment. To simplify the simulation of its behavior, an agent takes a 2D square tile-based game field mapped to a 3D game environment and moves in any of four directions: front, back, left and right.

1. A*

The A* algorithm is widely used as a general path finding algorithm for agents, especially for non-player characters in many interactive games. The algorithm finds the shortest path between two positions on a map, if one exists, and does so relatively quickly. It doesn't blindly search for a path, but instead assesses the best direction to explore, sometimes backtracking to try alternative means [12].

To implement the A* algorithm, the following objects are defined and used:

1. A map is the space that A* uses to find a path between two positions. In the game, the map is a 2D tile-based game field which represents the game environments where the ants are foraging.

2. Nodes are structures that represent positions on the map. The position represented by a node corresponds to the grid in the map. The nodes store information critical to the A* algorithm as well as position information. Two or more nodes can correspond to the same position on the map.

3. The distance is used to determine the "suitability" of the node explored and is calculated with the number of tiles in the shortest path between two nodes when the A* is applied to a 2D tile-based map.

4. The cost is associated with each node and it represents whatever it is that the path is supposed to minimize - typically, distance traveled, time of traversal, or fuel consumed. In the game, the distance traveled in a 2D map is mainly used. The time of traversal or fuel consumed could be an alternative cost when 3D geometry that ants are trailing is taken into account.

A* keeps track of two lists of nodes, called Open and Closed, for unexamined and examined nodes, respectively [13]. At the start, Closed is empty, and Open has only the starting node (the agent in its current position). In each iteration, the algorithm removes the most promising node from Open for examination. If the node is not a goal, the neighboring locations are sorted:

1. If they're new, they're placed in Open;
2. If they're already in Open, information about those locations is updated, if this is a cheaper path to them;
3. If they're already in Closed, they are ignored, since they've already been examined.

If the Open list becomes empty before the goal is found, it means there is no path to the goal from the starting location.

2. Pheromone Trail

Pheromone-based ant foraging is implemented as an agent behavior with a Finite-State Machine. A Finite-State Machine is a rule-based system in which a finite number of "states" are connected in a directed graph by "transitions" between states.

The agent has three states. See Fig. 7. The first state is looking around to find the pheromone. The second state is smelling three adjacent squares (the one is in

the direction in which it is heading and the other two are its left and right side) and stepping into the square which includes the pheromone it recognizes. The third state is bringing the food to the nest.

In the first state, an agent walks randomly to find any pheromone around it. In this state, an agent does not exude a pheromone itself. In the second state, an agent moves into its new position exuding pheromone to lead another ant along the same path. In the third state, an agent tries to return to the nest [14].

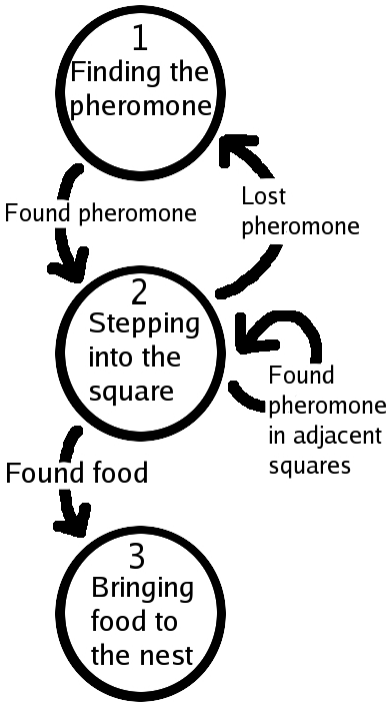


Fig. 7. The three states and transitions of the agent in its Finite-State Machine.

C. Prototyping

Rapid prototyping of the game ensures that techniques are usable. The game development tool for prototyping should have the following capabilities:

- The tool must present the game look and feel in a relatively complex 3D environment.
- The tool must serve as a game engine which enables characters to interact with a player in real-time.
- The tool must provide an easy user interface to build the game in the short term, as well as a programmable logic structure to implement various kinds of algorithms used in the game.

After looking at some of the available options, including full-fledged game engines, Blender™was selected as the prototyping tool. It is a free, open source software designed as a 3D modeling, animation and rendering tool which includes a real-time game engine for interactive 3D content [15]. The game engine supports collision detection and dynamics simulation which are necessary to create a 3D gaming environment. Blender™provides instant game play without preprocessing or compiling any 3D object data because it has an internal 3D object database which is directly accessed from the game engine.

1. Game Engine

A game engine is the core software component of a video game. The most common element that a game engine provides is graphics rendering facilities (2D or 3D) [16]. It might also handle additional tasks such as game AI and collision detection between game objects, among other things.

The architecture and interface of a game engine mostly depends on what it is customized for. For instance, some of the game engines currently available are customized for creating 3D first person shooters. The game engines utilize hardware accelerated graphics rendering. They use game objects created with external 3D graphics software to build and render the 3D interactive environment.

Blender^{TM}has its own game engine. The architecture of the Blender^{TM}Game Engine (BGE) is similar to "behavior-based architecture" in robotics. In behavior-based architecture, the robotics agent design is decomposed into behavior modules such as wall-following and obstacle avoidance. Each behavioral module accesses sensor inputs it needs and sends its own signals to actuators, reducing the need for a central processing unit and thus improving response time [17]. Sensors are components of a robot that enable it to gather information about their environment. An actuator is a device that converts software commands into physical motion. Fig. 8 shows an example of a behavior-based architecture, in which the four behavior modules independently take in input from the sensors and then send commands to the actuators.

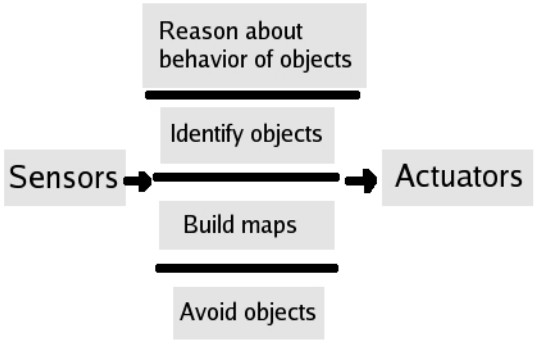


Fig. 8. A behavior-based architecture example.

In the BGE, Game Logic is the run-time module executed by the game engine and

is designed based on the behavior-based architecture. In the Blender^{TM}3D interactive environment, each game object becomes an agent which works based on user-defined modules. Fig. 9 shows a simple example of a Game Logic for an agent, which turns either left or right if something is in its heading direction. The controller, which is in between the sensor and the actuator, works as a behavior module in the behavior-based architecture and is implemented by a simple arithmetic/logical expression or by a Python script, which is written in the object-oriented programming language called Python, which is interpreted and run on the BGE.

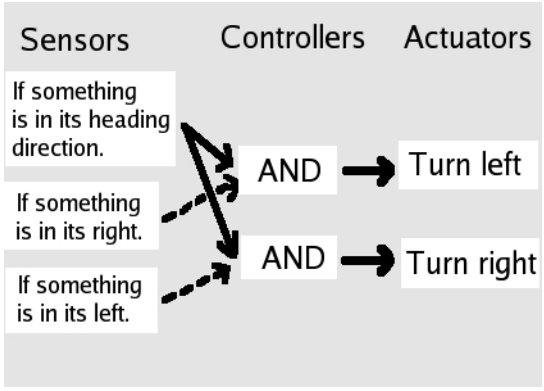


Fig. 9. Game Logic example.

2. Character Behavior Implementation

The ant agent is modeled with Maya^{TM}software and exported to Blender^{TM}as a 3D polygon model. It is rigged with blender armature system and animated in the BGE. The armature system is used for deforming a character model to animate it with keyframes. The armature is an object comprising several interconnected bones and works like a skeleton in a living creature. To implement the ant character behavior with Game Logic, the following two steps are used:

1. Character animation with user inputs
2. Character navigation with A*

a. Character Animation with User Inputs

The ant character walk cycle is created with the keyframe animation of bones in the ant's armature and driven by user inputs through keys with the ant character's Game Logic. The following modules are used in the Game Logic:

Keyboard sensor : scans user inputs through keys and passes key information to controllers. The game player control the ant's movement with four arrow keys: up, down, left, and right corresponding to going forward, turning back, turning left, and turning right respectively.

Python controller : calculates the force to apply to the character when it goes forward, interpolates the angle of the character's direction when it turns, and selects an appropriate keyframe in its walk cycle animation based on the distance it has moved from the last frame.

Force actuator : applies force to the character to move it in the direction it is going.

Armature actuator : sets the character's frame to all the bones in its armature to repeat its walkcycle animation.

b. Character Navigation with A*

Some ants in the game are autonomous agents which go back and forth on the game map between their start positions (usually their nest) and their end positions (often either food or a leaf to carry to the nest). The same walk cycle animation as used

for the user-controlled ant is used for the agent ants. However, their direction of movement was determined with A* algorithms rather than user input keys.

To navigate an agent ant character with A*, three steps are processed by the Python script in its Game Logic:

1. Obtaining the game map information: the map consists of 2D grids and each grid contains the information about the cost to pass over the grid. For instance, the cost of the grid covered with mud is higher than that covered with sand. The simplest map for the game consists of 2D grids which contain the values either 0 or 1, which correspond to open space or obstacle, respectively.

2. Finding the corresponding grid to the agent's position.

3. Run A*: the algorithm stops after it finds the grid that the agent steps into before it searches the entire map and finds the complete shortest path to its goal.

4. Move toward the grid where the agent steps into based on the result of the previous step.

The Game Logic repeats the above steps each time the agent moves so that it can perform A* correctly even if the map or goal information is changed dynamically.

3. Pheromone Trail Implementation

To create the crowd of ants, the pheromone trail method is implemented on the BGE. In the Game Logic, the agent's state transition is managed with a state property which specifies the agent's current state and different modules are used in different states as follows:

1. **Random walk**: the agent's direction is determined with a random number.

2. **Smelling the pheromone and stepping into its grid**: the three radar sensors are used to find pheromone objects in the grid cells to the front, left, and right of the agent's current position. A Python controller drives force and armature actuators to move and animate ants based on the inputs from the sensors.

3. **Bringing food to the nest**: the A* algorithm is used to direct the agent to the goal.

A pheromone object is an invisible object which is produced and put on the game map when agents are in the second state.

D. 3D Ant's Nest Model

The 3D nest model is based on the shape of a real ant colony which is adapted to make it suitable for use in a gaming environment. Creating the model with GPR (Ground Penetrating Radar) scanning technology provides a believable structure for nest chambers and tunnels.

Ground Penetrating Radar can see beneath the soil. High frequency radar pulses are sent from a surface antenna into the ground. The elapsed time between when the pulse is transmitted and when it is received after being reflected from buried materials or sediment and soil is measured. The antennas are moved along the surface, following transects of a grid [18]. The result of GPR scanning is processed with customized software and converted into 3D volumetric data which consists of many 3D grids.

To create the 3D surface model, visualization of the volumetric data is done in the following process:

1. Display the volumetric data of the 3D grids.

2. Construct surface from the volumetric data.

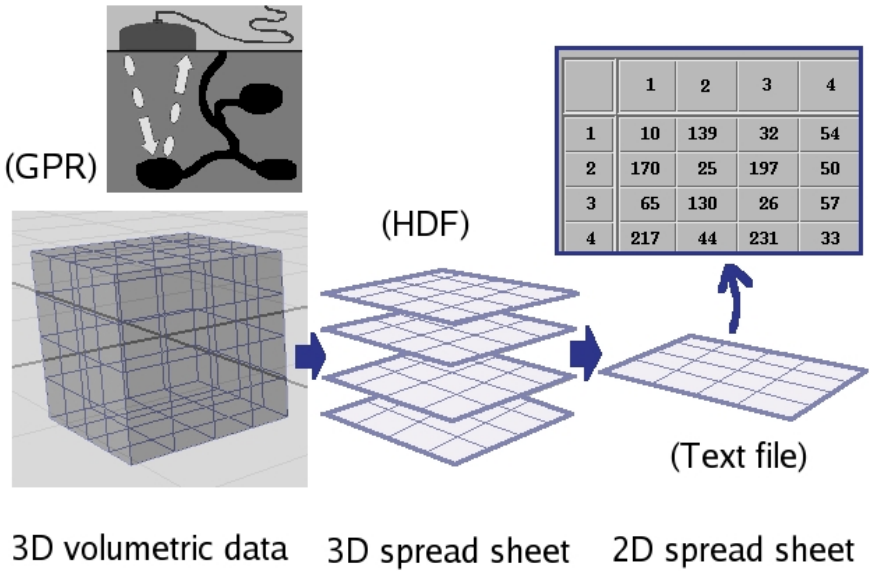
3. Composite the surface models to create ant's nest model.

Maya™is selected as a 3D visualization tool for the process. Maya™provides an extremely powerful and flexible user interface for 3D virtual environment creation, and many game companies use Maya™as their primary common platform for authoring 3D data. To translate the 3D nest model, Maya™works perfectly as a powerful modeling tool.

1. Display the Volumetric Data of the 3D Grids

The 3D volumetric data is provided in Hierarchical Data Format or HDF. HDF is an extensible, binary, public domain file format specification for storing data and images. The volumetric data used here is in the form of a 3D spreadsheet. The 3D spreadsheet is separated into 100 2D spreadsheets with NCSA HDFView, which was developed for browsing and editing general HDF data, and is available for free. Each 2D spreadsheet in the data corresponds to one slice of the volumetric data and is provided as a text file, which has 80 rows and 80 columns. Each value in its cell represents a measure of the density in the corresponding grid contained in the volumetric data See Fig. 10.

The text files are read and converted into a 3D array with Maya™Embedded Language or MEL. MEL is a script language interpreted and run on Maya™. It is generally used for creating models and animations procedurally. To display the volumetric data, the MEL script slices through the grid in the 3D array data, creates planes of polygons, and renders these with appropriate colors that represent density values in the data. The result shows the structure for the ant nest as well as different types of soil that surround the nest. See Fig. 11.



	1	2	3	4
1	10	139	32	54
2	170	25	197	50
3	65	130	26	57
4	217	44	231	33

Fig. 10. Convert 3D volumetric data to text files.

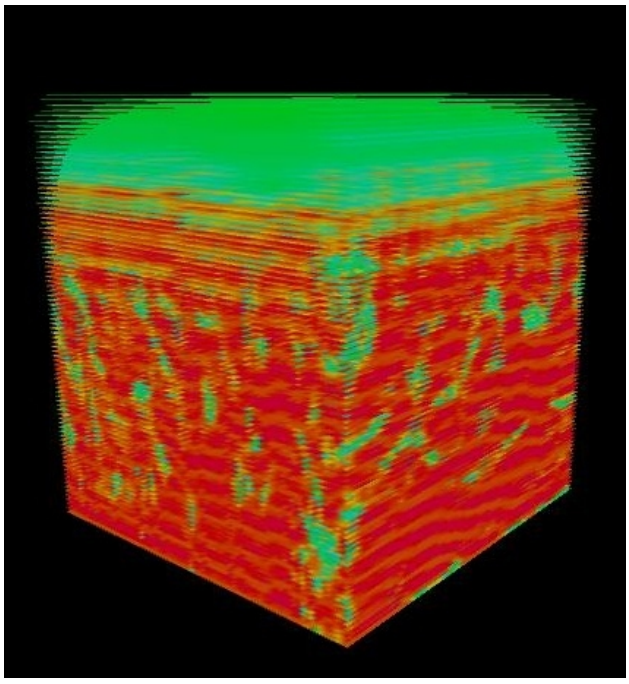


Fig. 11. 3D volumetric data of ant colony nest.

2. Constructing Surface from the Volumetric Data

The surface construction process involves the creation of a surface model which consists of 3D volume data. To select the desired surface to construct, a density value in the volume data is specified. The surface model consists of polygons so that graphics hardware can draw the model as a gaming environment in real-time.

The algorithm called "marching cubes" creates a polygonal surface representation of an isosurface of a 3D scalar field. An isosurface is a 3D analog of a contour line. It represents a surface of constant value (e.g., pressure, temperature, velocity, density) within a volume of space. The algorithm processes a 3D scalar field in scan-line order and calculates triangle vertices using linear interpolation. The detail in images produced from the generated surface models is the result of maintaining the inter-slice connectivity, surface data, and gradient information present in the original 3D data.

The algorithm is implemented with C++ language in a customized program. The program works in the following steps:

1. Read the text files produced from the original HDF file and convert them into 3D array data.

2. Create the surface of a specified density value with the "marching cubes" algorithm.

3. Save the surface models as a set of triangles in Alias Object file format.

The program is executed as many times as the number of different density values in the volume data and results in a set of surface model data which covers whole values in the volume data.

3. Compositing the Surface Models

To create the surface model which represents the structure for nest chambers and tunnels, appropriate density values are selected based on the 3D volumetric data displayed in the first process. The surface composition process combines the surface models with selected density values. Maya™imports the surface model produced from the customized program in the previous process and displays it in a separate layer depending on its associated density value.

Imagery of the surface model created from volumetric data was first used in an interactive installation entitled *atta* [19]. See Fig. 12. The surface model is a large set of triangles and is manually crafted for use in a 3D game environment with Maya™. A portion of the entire model is used in demonstrating the game environment included here.

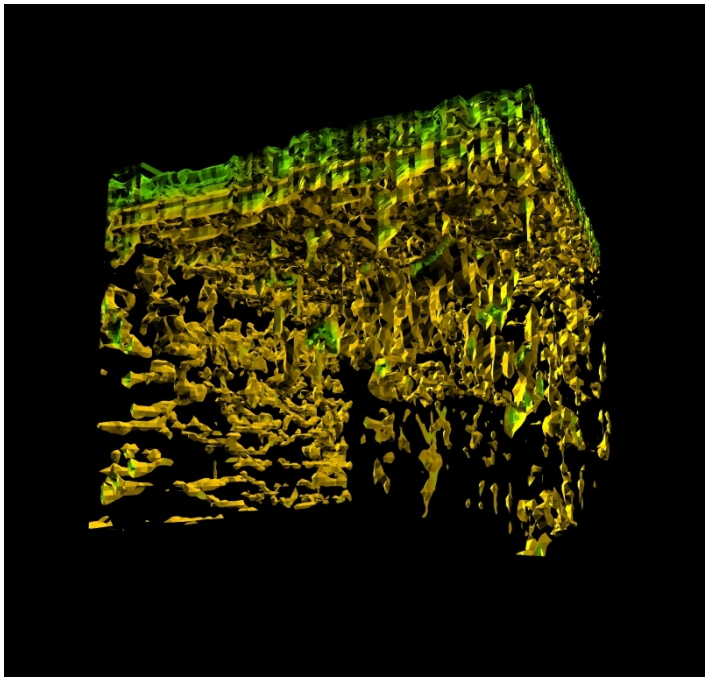


Fig. 12. The surface model created from the volumetric data.

CHAPTER IV

RESULTS

The results of the research described in this thesis include the following:

1. Ants behavior visualization with the Game Engine
2. The ant's walk through system in the 3D nest
3. Two short animation clips for the cutscenes
4. Game trailer

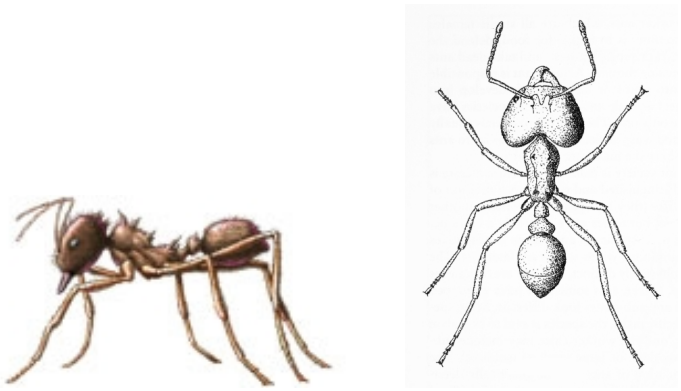
The first result shows the pheromone trail implementation in a 3D interactive environment to be used in both interactive and non-interactive part of the game. The second result is a sample game navigation of the first stage in the game with the ant's nest model. The third result is cinematic sequence inserted between the interactive parts of the game. The fourth result is the preview of the complete game.

A. Ants Visualization with the Game Engine

1. 3D Modeling and Animation

a. Ant Modeling

The 3D ant model shows the appearance of *Atta texana*. It is modeled with triangular and square polygons. Fig. 13 shows the reference picture of *Atta texana* worker and soldier ants. From the reference, the model is characterized with a big head, some spikes on its back, and leg joints. Fig. 14 shows the final model in three diffrent views.



(a) (b)

Fig. 13. The reference picture of *Atta texana*: (a) a worker and (b) a soldier.

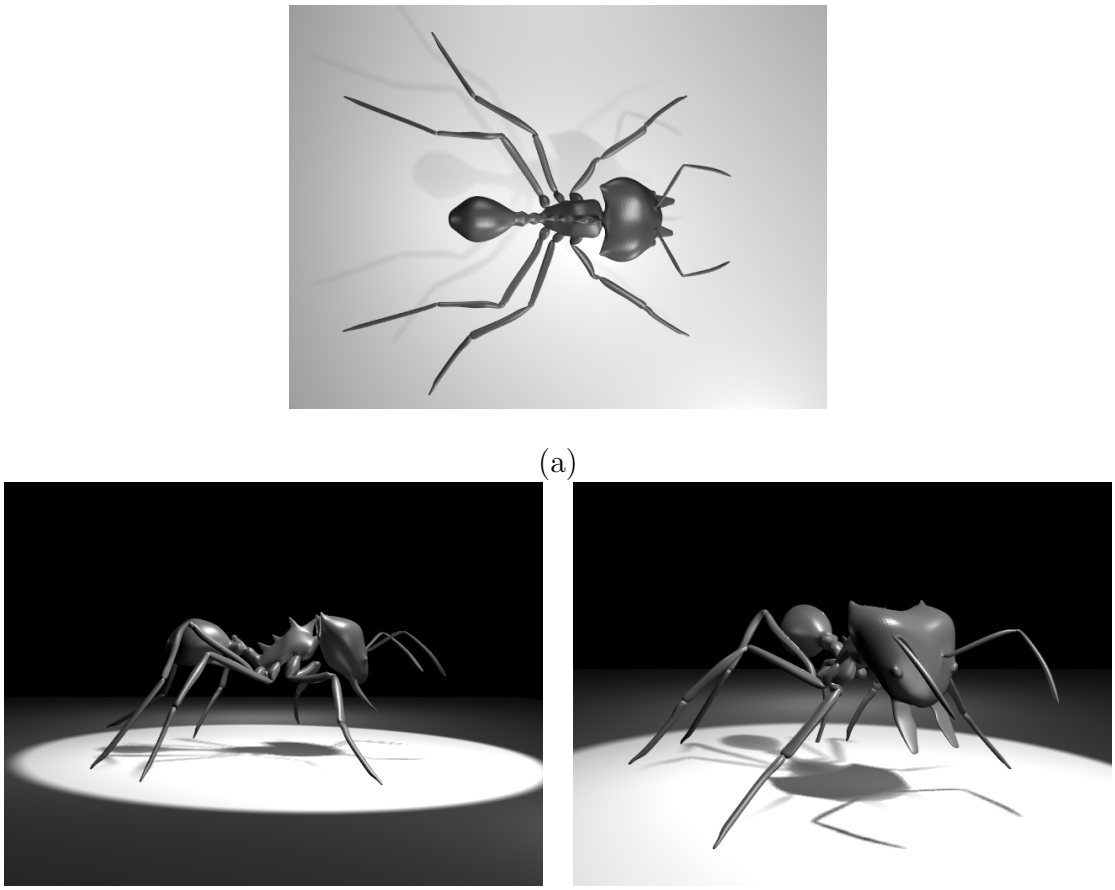
b. Ant Walk Cycle

The 3D ant model is rigged with the Blender armature system and animated on the BGE with Game Logic. Fig. 15 shows the bones of the ant model. The ant walk cycle is keyframed with a procedural approach using Python scripting, which automates the keyframes of the six legs characters' walk cycle [20].

To animate the ant model walk cycle keyframes, two Python scripts are written and run on Game Logic. The scripts are extensions of the two scripts "Foot Lock" and "Orientation" used in "The Official Blender Gamekit" [21]:

Foot lock Python script The ant character moves with a force actuator which produces a linear motion of the character on the BGE. The keyframe of the walk cycle is selected in each frame according to the distance the ant moves from the former frame. The Foot lock Python script calculates the distance the ant moves between frames and selects the keyframe from the walk cycle in each frame.

Orientation Python script The ant character walks in one of eight different direc-



(a)

(b) (c)

Fig. 14. The final 3D ant model: (a) top view, (b) side view, and (c) front view.

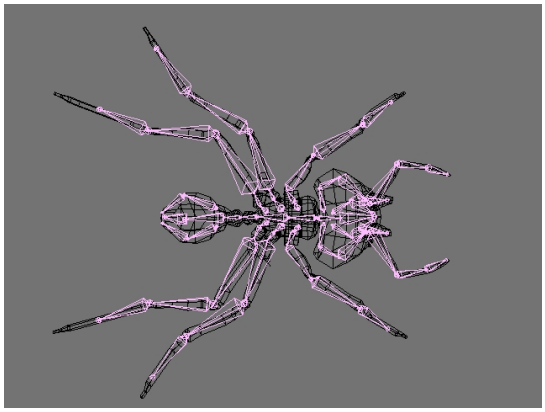


Fig. 15. The 3D ant model rigged with the bones.

tions: North, Northeast, East, Southeast, South, Southwest, West, and Northwest. See Fig. 16. The character changes its orientation according to the direction it is heading. Each time it changes direction, it turns within a range of 45 to 180 degrees. The Python script changes its orientation at a certain degree and smoothes interpolation between frames. Fig. 16 shows the interpolations for two in-between frames from Northeast (NE) to East (E). The ant orients to one-third of the angle between NE and E in the first frame, then to the two-thirds of the angle in the second frame, and finally to E in the third frame.

2. Pheromone Trail

a. Leaves, Pheromones, and the Game Field

Other than ants, the following game objects exists in the game field for pheromone trail implementation, and Game Logic manages each object while the Game Engine is running:

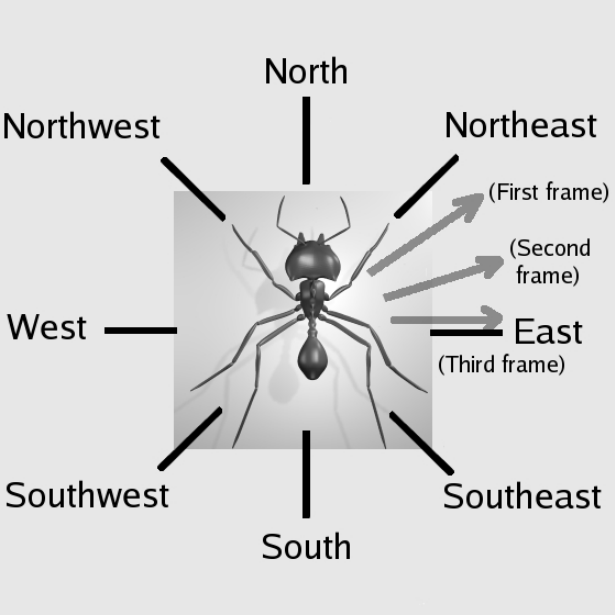


Fig. 16. Eight directions of the ant's orientation and its interpolation.

Leaves The ant goes back to the nest to deliver a leaf after locating it. If the ant touches a leaf at certain times, the leaf is gone. Game Logic counts each time the ant touches a leaf a certain number of times and removes the leaf when the count exceeds a predefined number.

Pheromone The ant exudes a pheromone trail while bringing leaf to the nest. The pheromone disappears after a certain duration.

The Game Field The game field is the 3D environment where the ants are foraging. The game field is a square plane without obstacles except for borderlines in the four directions: north, east, west, and south. The ant turns back if it encounters a borderline in front of it.

b. Collision Avoidance

The ant character animated in the BGE obeys its physics engine, which detects collisions among characters and resolves them. If two ants try to approach each other after collisions are resolved, another collision happens and they are stuck in one place. To avoid this, each ant has a "radar" which scans a defined heading area, and it turns back when its radar detects other ants on a borderline.

c. Random Walk

Random walk cycles occur in two cases:

1. The ant avoids collisions with other ants while bringing leaves to the nest.
2. The ant is looking for a pheromone trail.

In the former case, the ant still tries to carry a leaf to the nest after it avoids collision (i.e. no obstacle is detected ahead of it). In the latter case, the ant walks randomly to find a pheromone trail and the ant's behavior is similar to a real ant. A real ant gets confused if the trail is blurred and cannot find their way to the food supply. Often, the finder does not go home at all, but simply circles around the food it has discovered [22].

To create this behavior, the following steps are applied to a random walk:

Step A The ant draws a circle from the grid at which it begins its random walk. The ant goes forward four grids and then turns right. The ant repeats four times and comes back to the grid where it started this step.

Step B The ant draws another circle in the opposite direction. The ant turns left in this step.

Step C The ant moves toward the direction either a) where the leaf exists, or b) its nest in a defined amount of time. If the ant found a leaf before, the case a) is applied in this step. Otherwise, b) is applied to find the pheromones near the nest.

Step D After repeating process A to C two times, it goes back to the nest and repeats its random walk. Two times repeating helps the ant find pheromone around the leaf discovered before or nest.

Step A and B cause the ant to seek a pheromone trail or a leaf near the place where its trail became lost. Step C and D cause the ant to approach either a leaf or the nest to find the pheromone trail in that location. At any time the ant finds pheromone in neighbor grids, it trails the pheromone.

3. Configurations and Results

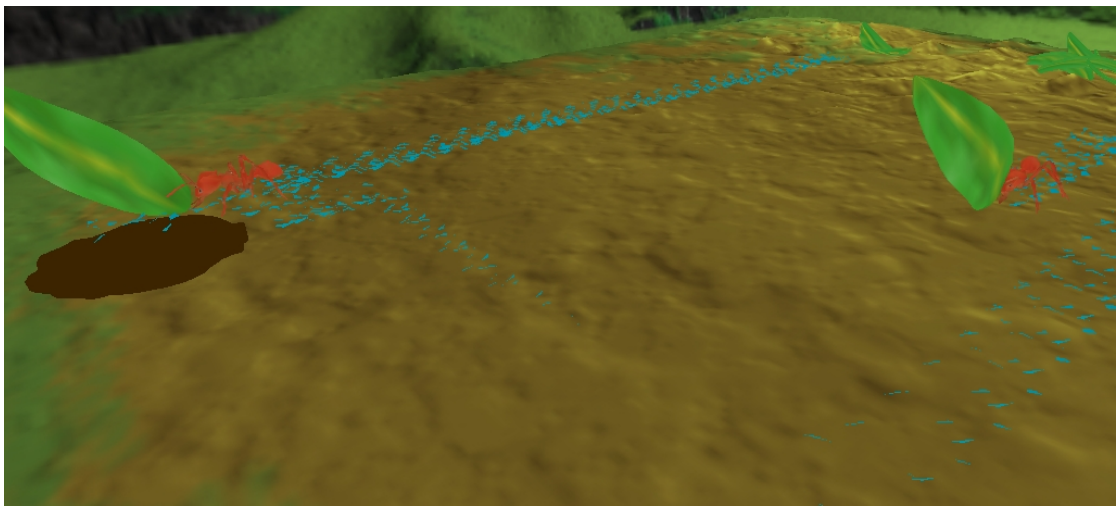
Two configurations of the game objects are tested:

1. The 20 x 20 grid game field with two ants and five leaves.
2. The 30 x 30 grid game field with three ants and twenty leaves.

The duration of pheromone existing in the field is the same in both configurations. Foraging simulation in the game field resulting from these two configurations are:

In the first configuration, all leaves are gone after one and a half minutes. In the second configuration, all leaves are gone after seven minutes.

The Blender game engine for this simulation was executed on the Blender 2.41 that is run on Linux workstation with an Intel®Pentium®4 CPU 3.00GHz speed and 1G-byte memory. See Fig. 17.



(a)

(b)

(c)

Fig. 17. (a) Two ants bring leaves to their nest. The dots on the field represent their pheromone. (b) One ant reached to their nest. (c) The other an ant is walking near the pheromone.

B. The Ant's Walk Through System in the 3D Nest

1. Game Character and Environment Settings

The following character and environment are designed for the game prototyping of the first stage:

1. The 3D model of a non-player character
2. The office of the "BUG BING!"

The player ant's character used in the walk through system is the 3D ant model. The ant model and animation created for the pheromone trail implementation are used for the system. The non-player characters have two different appearances; one is a real ant style and the other is a half-ant-half-human style. When the player encounters them on his or her way to the nest, they are shown in the real ant style. When the player communicates with them, they are in the other style as shown in Fig. 18.

"BUG BING!" is the place where the mafia ants have meetings. It is named after the place called "BADA BING!" which is one of the most popular places in "The Sopranos." A player meets with other mafia ants in its office. See Fig. 19.

A player visits the back office and speaks with the mafia ants. In this prototype, the player does not have many choices for the conversation. Instead, the player installs a transmitter somewhere in the office. A table, a TV set and a fish toy are in the office. See Fig. 20.

2. Game Navigation

The player ant character explores the 3D nest and visits the "BUG BING!" The walk through system in the 3D nest is developed with the Blender Game Engine (BGE)

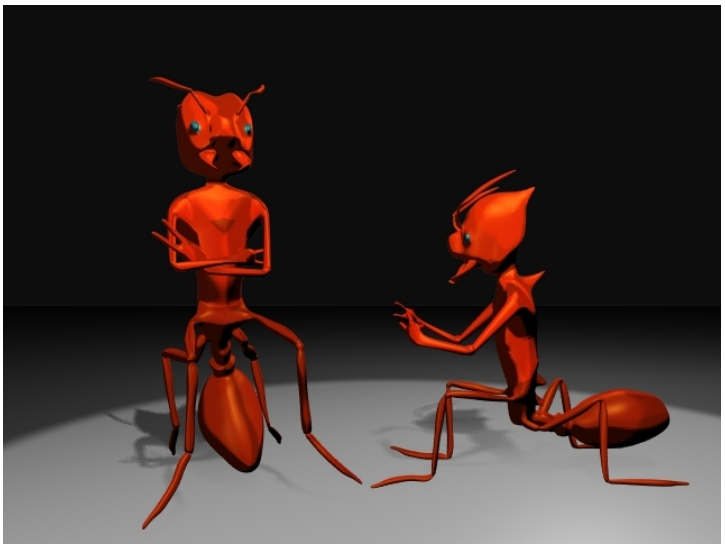


Fig. 18. The 3D model of a non player character.

Fig. 19. The non-player characters in “BUG BING!”.

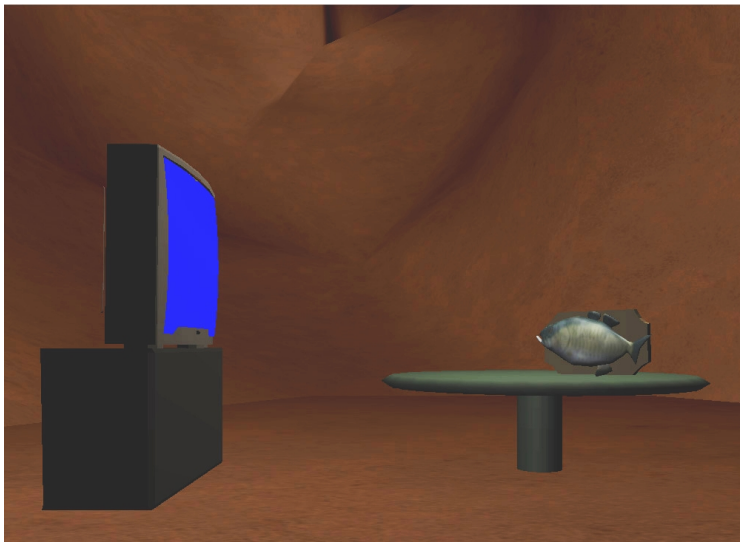


Fig. 20. The office of the "BUG BING!"

and the player ant walks along a pre-determined path with a pre-keyframed camera animation for the first stage prototype.

The terrain used for the system is a part of the entire nest model as shown in Fig. 21.

If no one is in the office, the player installs a transmitter. After the player installs the transmitter, the agent shows the result of his or her mission. The best place to install the transmitter is under the table in the office. See Fig. 22.

C. Cutscenes

Two intermission cutscenes are created for the game prototype: one is the intermission between the first and the second stage. The other is between the second and the third stage. The first intermission cutscene shows that the player character listening to the conversation in the back office of the "BUG BING!" through the transmitter which is supposed to be installed by the player in the first stage. The 3D model of the

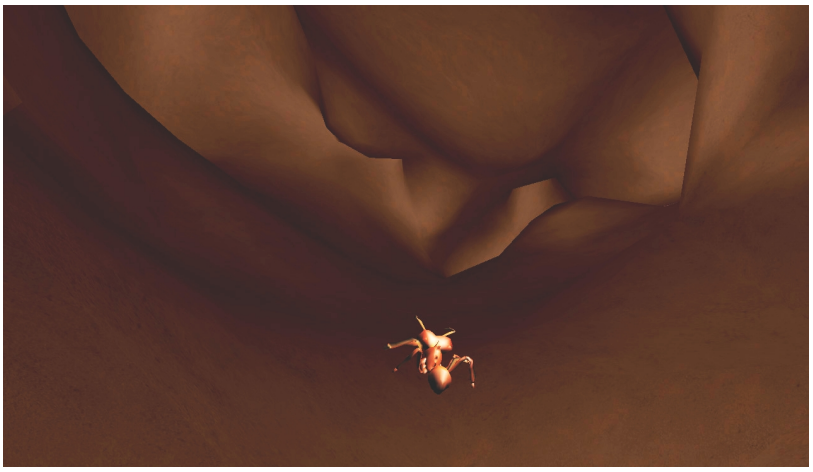


Fig. 21. The terrain used for the walk through system.

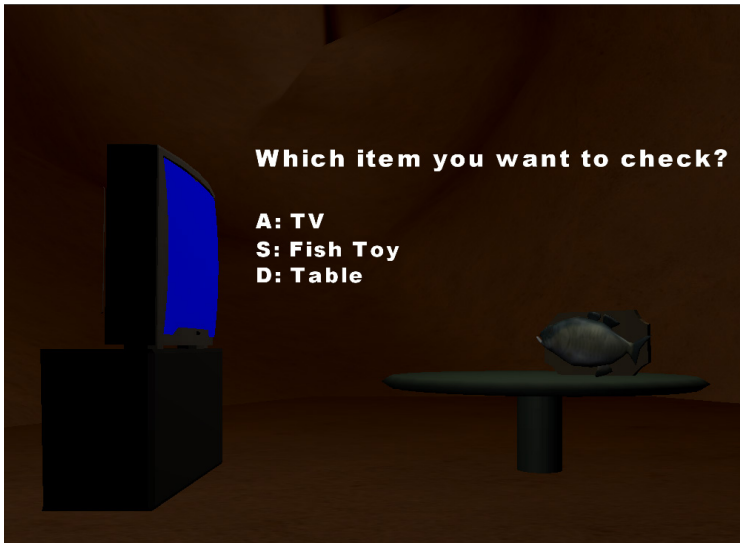


Fig. 22. The system shows an overlay message to a player.

characters and environment are rendered with Blender™and Maya™. See Fig. 23. The cutscenes are edited with the soundtrack which includes the actual voices from the TV show.

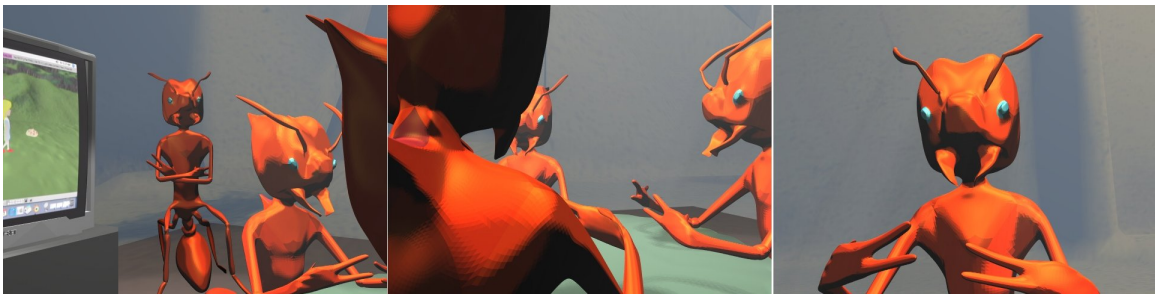


Fig. 23. The intermission cutscene for the second stage.

The second intermission cutscene describes the black ant under boss's accidental death which causes the war between the two ant families in the third stage. The black ants are crushed with a crumb of a cookie and they suspect that their underboss is killed by the red ants. See Fig. 24.

The foraging ant characters are animated with the pheromone trail behavior implemented on the BGE. The result of the simulation is converted to keyframes and used to animate the ants for the cutscene. See Fig. 25.

D. Game Trailer

The game trailer starts with the second intermission cutscene. The second sequence is the capture from the ant's nest walk through. The third sequence is the player installing the transmitter in the "BUG BING!" followed by the first intermission cutscene. The last sequence shows the black ants outside the nest. See Fig. 26. The trailer ends with the game title logo.

(a)

(b)

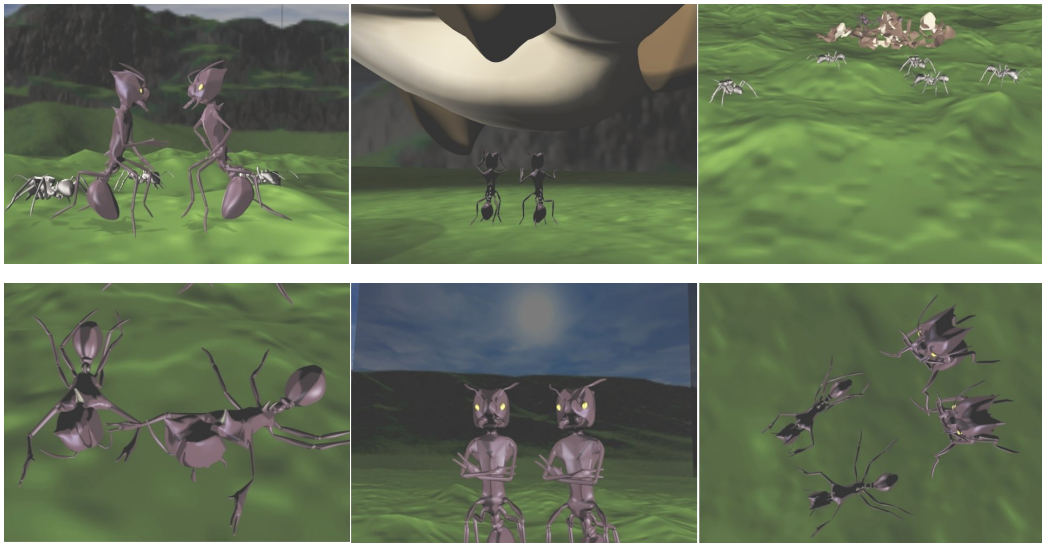


Fig. 24. The intermission cutscene for the third stage.

(a)

(b)

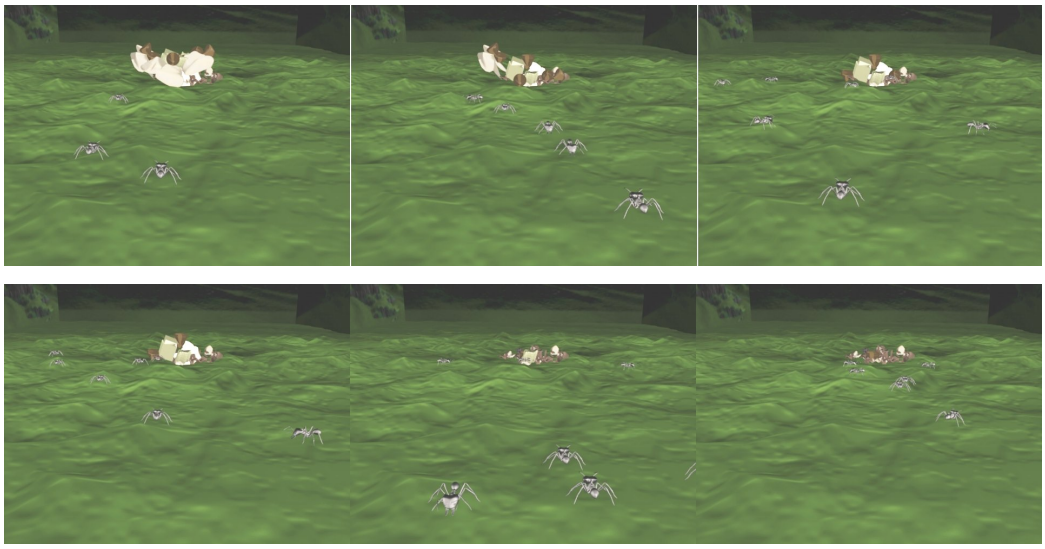


Fig. 25. The foraging ants are animated with the pheromone trail simulation.

Fig. 26. Four stills from the game trailer.

The soundtrack of the trailer includes those of two cutscenes as well as the title song of the TV show, "*Woke Up This Morning*" written and performed by A3.

CHAPTER V

CONCLUSION AND FUTURE WORK

A. Discussion of Results

The conceptual framework for this video game is as an example of an interactive story with insect characters. Characters are modeled after human characters in the popular TV series "The Sopranos," which also serves as an engaging plot structure for the game. Interesting leaf-cutter ant behavior, such as growing fungus, is introduced as the family business of organized crime.

The game is based on real world ants in terms of behavior and colony structure. A unique innovation is included in the game design: a 3D model from a GPR scan of a leaf-cutter ant colony. The model is created with the "marching cube" algorithm and crafted with Maya™to be used as a 3D environment. The algorithm produced the model, which consists of polygons. The model was easily converted to a 3D terrain in the game, and it was manually modified and textured to become one seamless surface model.

The technical framework for simulating an ant's pheromone trail is used to create a group of non-player ant characters whose behavior is similar to that of an existing colony. A crowd controlling system is successfully implemented on the Blender™Game Engine. Non-player ant characters are programmed as autonomous agents animated interactively with Game Logic and Python scripting.

The prototyping process of the game and its cutscenes with two 3D graphics tools is a useful experiment. Maya™is used to model 3D objects used in the game. Blender™is used to animate objects for both interactive sequences and cutscenes. The Blender™Game Engine is effectively used for rapid prototyping of game navi-

gation and the 3D environment layout. Cutscenes were also created with Blender™. Its rendering performance and quality are acceptable for the prototyping process.

B. Implications for Future Research/Creative Work

The following tasks are needed to complete the game:

1. An entire game navigation with all game characters designed.
2. All 3D objects and environment models.
3. All animations for interactive sequences and cutscenes.

The game characters are modeled after key characters in "The Sopranos" and their roles in the games are various depending on the entire game story; some of them are informers that help the player solve the game and others are enemies that try to trap the player. The player's tasks in the game involve the following steps:

1. Exploring inside or outside the nest.
2. Talking with ants and collecting information to complete the mission.
3. Speaking with the player's agent to complete the mission.

The game objects and environments are modeled and textured properly. They are used in both interactive sequences and cutscenes. Character animation, which includes lip-syncing for dialog, is a time consuming process. A character's rig could be reused among different character models and lip-syncing could be automated with voice recognition.

Implementation of the agents and crowd system, based on pheromone trail behavior, could be extended for animating more ants at once in a scene. The collision

avoidance algorithm could be improved so that more agent ants could be animated without losing the game engine's performance and interactivity.

The 3D nest model could be developed for use in other creative works. A 3D ant nest walk-through with a virtual reality system is one example. The ant model is also reusable for developing a walk-through nest environment. The game prototyping process itself could also be reused for general game creation. Performance and quality of consumer 3D graphics cards have improved dramatically. Graphics cards are becoming more common and prices are dropping. Blender^{TM}is supported by a robust developer community and is expected to become a more powerful 3D graphics platform in the future. One promising development of Blender^{TM}game creation is that of supporting networking games.

While gaming is becoming one of the most-consumed media forms today, new titles are still being developed by large companies. I hope my thesis project will be useful to those who wish to create video games independently.

REFERENCES

[1] B. Höllodobler and E. O. Wilson., *The Ants*. Cambridge MA: Belknap Press of Harvard University Press, 1990.

[2] B. Robertson, "Pixar has taken a unique approach to animating crowds of characters for its next feature film A Bug's Life," *Computer Graphics World*, Vol. 21, July 1, pp. 24-34, 1998.

[3] B. Robertson, "Faces and crowds," *Computer Graphics World*, Vol. 21, July 1, pp. 61-63, 1998.

[4] J. Patterson, "Implementing Autonomous Crowds In a Computer Generated Feature Film." M.S. thesis, Visualization Sciences, Texas A&M University, 2005.

[5] D. Hopkins, *Designing User Interfaces to Simulation Games*. [Online]. Available: http://www.donhopkins.com/drupal/node/9.

[6] Strategy First, *Empire of the Ants*. [Online]. Available: http://www.strategyfirst.com/en/games/redir/?iGameID=13.

[7] B. Werber, *Empire of the Ants*. New York: Bantam, 1991.

[8] R. Nelson, *A Bug's Life Review*. [Online]. Available: http://psx.ign.com/articles/160/160232p1.html.

[9] H. Hancock, "Better Game Design Through Cutscenes," *Gamasutra*. [Online]. Available: http://www.gamasutra.com/features/20020401/hancock_01.htm.

[10] R. North, *ANTS*. London: Whittet Books, 1996.

[11] J. H. Tumlinson, R. M. Silverstein, J. C. Moser, R. G. Brownlee, and J.M. Ruth, "Identification of the Trail Pheromone of a Leaf-cutting Ant, Atta texana," *Nature*, Vol. 234, December 10, pp 348-349, 1971.

[12] S. Ravin and M. Deloura, *AI Game Programming Wisdom*. Hingham MA: Charles River Media, 2002.

[13] M. Deloura, *Game Programming Gems*. Hingham MA: Charles River Media, 2001.

[14] S. Johnson, "A Pheromone Based Ant Foraging Simulation." M.S. thesis, Computer Sciences, University of Sheffield, 2001.

[15] Blender Foundation, *Blender*. [Online]. Available: http://www.blender.org/

[16] Wikimedia Foundation, *Game Engine*. [Online]. Available: http://en.wikipedia.org/wiki/Game_engine.

[17] T. Bryant, S. Engineer, and H. Hu, *Robotics*. [Online]. Available: http://www.mbhs.edu/˜ lpiper/Robotics03/.

[18] L. B. Conyers, *Ground-Penetrating Radar for Archaeology*. Lanham MD: AltaMira Press, 2004.

[19] C. LaFayette, *atta*. [Online]. Available: http://www-viz.tamu.edu/faculty/lurleen/main/attamain.htm.

[20] L. Wibaux, *Insect Procedural Walk Tutorial*. [Online]. Available: http://americanhistory.si.edu/muybridge/.

[21] T. Roosendaal and C. Wartmann, *The Official Blender GameKit: Interactive 3D for Artists*. San Francisco CA: No Starch Press. 2003.

[22] W. Goetsch, *the ants*. Ann Arbor MI: The University of Michigan Press, 1957.

VITA

Tatsuya Nakamura
Visualization Laboratory
A216 Langford Architecture Center
3137 TAMU
College Station, TX 77843-3137

Education

M.S. in Visualization Sciences	Texas A&M University, August 2006
M.S. in Information Science	Kyoto University (Japan), 1998
B.S. in Information Science	Kyoto University (Japan), 1996

Employment

Technical Artist Intern	Electronic Arts, June 2005 - August 2005
Research Assistant	Texas A&M University, September 2004 - December 2004