EAS 499 Senior Capstone
Project Proposal Specification – 1/22/12
Coordinated by Max Mintz

Artificial Intelligence for Monsters in Tower Defense Games (AIMTD)

Federico Nusymowicz
fedenusy@gmail.com

Advisor: Kostas Daniilidis
kostas@cis.upenn.edu

University of Pennsylvania

ABSTRACT

Tower defense game developers typically use waypoints to predefine monster attack routes. Unfortunately, these scripted attack patterns translate into predictable game experiences with low replay values. The TD genre would be far richer if monsters could choose the optimal path to follow, elect to wait for reinforcements, or decide to sacrifice themselves for their teammates.

My project will consist of a light open-sourced library for TD monster intelligence written in Ruby. Developers will input the map layout along with the monsters' movement speed, intelligence level, health, and other characteristics. AIMTD will then return the monster's optimal next move in real time.

The goal of the project is to lay a foundation that will help developers create the next generation of TD games. AIMTD will need to be well documented and easily extensible so that it can grow to process more advanced monster behaviors and to handle more complex inputs.

1. INTRODUCTION

1.1 Tower Defense

Tower defense (TD) is a subtype of real-time strategy games. The TD player's goal is to prevent groups of monsters from crossing a map by building offensive towers that shoot projectiles.

TDs gained vast popularity over the last 5 years. Kingdom Rush, for example, accumulated over 17 million plays over just a few months. TDs became so popular because they are

- Simple enough for any casual gamer to pick up and play.
- Engaging enough to entertain any 'hardcore' gamer for hours.
- Light enough for a single indie developer to program without much difficulty.

1.2 Game Mechanics

In TDs, monsters (aka "creeps") appear in waves at specific places on the map ("spawn points"). Monster waves then travel through the map in order to reach a vulnerable objective.

The player's mission is to protect the objective from monster attack. To do so, the player must spend limited resources to build shooting towers that will destroy creeps before they reach the objective. Towers have a limited shooting range however, so the player must strategize carefully. Different towers also vary in their build and upgrade costs, attack speeds, and damage rates.

The player earns resources by killing monsters. Resources can in turn be used to build more powerful towers or to upgrade existing ones.

Like towers, different monster types have unique characteristics, including health, defense rating, and movement speed. Monsters tend to become more powerful and numerous as the level progresses.

Levels finish once the player defeats all monster waves. See Figures 1, 2, and 3 at the end of this document for an illustration of a typical TD.

1.3 Current State of TD Monster Intelligence

TD monster intelligence is typically based on one of two very simple rules. Monsters either a) follow a scripted path or b) choose the shortest route towards the objective. While both behaviors are simple enough to program, their simplicity harms the TD game experience in a couple of ways.

First, simple behaviors make creeps easier to kill as the level progresses. As soon as players recognize monster attack patterns they can exploit their predictability for each successive wave. This

translates into shallow winning strategies and rigid game experiences.

Secondly, simple monster behaviors harm the TD's replay value. Players who complete a level can simply recycle old winning strategies to defeat the level once more. Therefore, once a player defeats a TD, there's usually no reason to revisit the game ever again.

1.4 Motivation

My roommate and I began developing an indie multiplayer TD last semester. We wanted to create a dynamic and ever-changing TD world with randomized maps, so monster intelligence (AIMTD) quickly became one of our project's key features. AIMTD's ultimate goal is to force players to think on their feet by requiring them to adopt a wider variety of winning strategies.

My hope is that making AIMTD an open-source project will spark a move towards more dynamic TDs from the game developer community.

1.5 Project Features and Functionality

Developers should be able to easily input

- A grid-based representation of the TD map.
- Tower info: location, shooting range, and damage rate.
- Monster info: health, speed, and defense rating.
- Wave info: monster deployment order linked to each creep in the wave.

Monsters should then have the ability to continuously poll AIMTD to determine their optimal next move. AIMTD should base its decision on the shortest path to the objective, a likelihood-of-survival estimate for the given path, and monster group tactics that the creep wave can follow to maximize damage.

Developers should also have the ability to optionally determine

- Monster intelligence level (ie, how likely the monster is to follow the optimal strategy).
- Predefined waypoints that a monster must traverse when crossing the map.
- Monster field of vision.
- Autonomous monster subgroups that act independently from the rest of the wave.

2. RELEVANT WORK

[Y08] Focuses on identifying the sources of player satisfaction from videogames. The author identifies challenge as a driving factor in player engagement and goes on to suggest real-time difficulty adjustments in order to enhance the gaming experience.

[ATAL11] Introduces the TD genre in depth and explores several ways TDs could benefit from computational intelligence. Suggestions for TD AI include intelligent map/game element generation, dynamic difficulty adjustments, player simulation AI, and AI for determining creep strength and deployment rates. The paper ends by proposing Infinite TD, a prototype TD game that would incorporate intelligent features. Note that this paper does not discuss the individual monster's intelligence with regards to pathfinding and teamwork but rather proposes a model for monster health and deployment rate adjustments based on the player's skill level.

[ALA09] Presents an implementation of computational intelligence in the context of coordinated spatial strategies for autonomous entities. In the case of this particular implementation, the autonomous entities are boats acting to beat an opposing team in a naval capture-the-flag game. The Influence Maps used in the paper are an applicable approach for developing optimal creep decisions when monsters try to reach their objective in a TD game.

[WM09] Deals with case-based reasoning for build orders in the real-time strategy game Wargus. Wargus players have to deal with limited vision due to the game's 'fog of war'; this paper presents a player AI implementation for strategic allocation of resources. Because of the player's limited vision, this solution would help when implementing limited creep vision in TD games.

[HC04] Presents the concept of dynamic difficulty adjustment (DDA) for videogames. The article focuses on Hamlet, a DDA system based on monster AI difficulty tweaks. Most DDA concepts presented will be useful for determining different levels of monster AI strength in AIMTD.

[H10] This MIT OpenCourseWare class focuses on Artificial Intelligence. Topics include goal trees, rule-based systems, basic search, optimal search, and games. Several more advanced topics follow. I expect to progress through this course as AIMTD evolves in complexity.

[H92] This book is the companion to [H10] above. Chapter 6 is of special interest as it discusses AI in games. Chapter 5 discusses optimal search (including A* search), which will prove highly relevant to real-time monster pathfinding.

3. PROJECT PROPOSAL

3.1 Anticipated Approach

The first step will consist of designing the testing environment. I will create a simple TD sandbox using Gosu, a Ruby library for 2D game development. The testing environment should allow users to

- Add/delete paths that monsters can traverse.
- Add/delete various types of towers.
- Deploy various types of monsters.
- Deploy waves of monsters.

After the TD sandbox is complete I will lay the groundwork for AIMTD. Developers will need to input the map layout (including tower status) into the AIMTD library for the algorithm to return creep movement decisions. Since I want to make the library as simple as possible for game developers to implement, I will present AIMTD as a well-documented Ruby gem.

Once AIMTD's groundwork is set up I will start writing the monster AI. First I will add methods for waypoint support and shortest-path A* search. Then I will move on to more complex behaviors, including optimal-survival-likelihood paths and monster group tactics.

I also intend to add support for varying monster intelligence levels. The most intelligent monsters (as designated by the developer) will be the most likely to adopt complex behaviors. This means developers won't have to delve into the intricacies of the AIMTD library; they can simply define how intelligent they want their creeps to become, and AIMTD will handle the rest.

If time permits I would also like to implement

- Limited monster fields of vision.
- Creep subgroups that act independently from the rest of the wave.
- Support for other types of map representations.
- Handling of other monster attributes (eg elemental resistances).
- Handling of other tower attributes (eg elemental damage).
- Dynamic AI difficulty adjustments.
- Other more advanced monster AI.

3.2 Evaluation Criteria

Highly intelligent monsters should reach their objective most often. Thus, for identical map/tower layouts and monster waves (but varying intelligence levels), more of the creeps with higher intelligence should reach the objective. The project should also be evaluated on ease of implementation and clarity of the documentation, although these two are more subjective criteria.

3.3 Timeline

See Figure 4 at the end of this document for a proposed timeline.

The AIMTD final deliverable will consist of a well-documented Ruby gem. The AIMTD gem will contain a TD sandbox for testing the various AI behaviors. Behaviors will include but not be limited to

- Waypoint specifications
- Shortest-path A* search
- Optimum-survival-likelihood decisions for the individual monster
- Monster group tactics for maximizing damage
- Intelligence levels for the individual monsters

References

[Y08] G. N. Yannakakis. "How to Model and Augment Player Satisfaction: A Review." Proceedings of the 1st Workshop on Child, Computer and Interaction. Chania, Crete: ACM Press, October 2008.

[ATAL11] P. Avery, J. Togelius, E. Alistar, and R.P. van Leeuwen. "Computational Intelligence and Tower Defence Games." Proceedings of the 2011 IEEE Congress on Evolutionary Computation (CEC). IEEE Press, 2011.

[ALA09] P. M. Avery, S. J. Louis, and B. Avery. "Evolving coordinated spatial tactics for autonomous agents using influence maps." Proceedings of the 2009 IEEE Symposium on Computational Intelligence in Games. IEEE Press, 2009.

[WM09] B. G. Weber and M. Mateas. "Case-based reasoning for build order in real-time strategy games." AIIDE, 2009.

[HC04] R. Hunicke and V. Chapman. "AI for dynamic difficulty adjustment in games." Proceedings of Artificial Intelligence and Interactive Digital Entertainment, 2004.

[H10] P.H. Winston. *6.034 Artificial Intelligence, Fall 2010*. (Massachusetts Institute of Technology: MIT OpenCourseWare), http://ocw.mit.edu (Accessed 13 Jan, 2012). License: Creative Commons BY-NC-SA.

[H92] P.H. Winston. *Artificial Intelligence (Third Edition)*. Boston, MA: Addison Wesley, 1992.

Figure 1. An empty TD map. Skulls at the bottom represent monster spawn points; blue flags at the top represent the vulnerable objective creeps must not reach. The white flags alongside the road are locations where towers can be built. (Credit: Kingdom Rush)

Figure 2. About to build an Archer Tower, which costs 70 resource units (gold in this case). The green area represents the tower's attack range. (Credit: Kingdom Rush)

Figure 3. Archer and Mage Towers shooting at the first wave of monsters. (Credit: Kingdom Rush)

	1/29	2/4	2/11	2/18	2/25	3/3	3/10	3/17	3/24	3/31	4/7	Beyond
Progress through [H10] and [H92]												
Complete background reading												
Build test environment												
AIMTD input handling & documentation												
Waypoint support and A* search												
Optimum survival likelihood path												
Monster group tactics												
Monster intelligence levels												
Cleanup and optimization												
Monster field of vision (if time)												
Autonomous monster subgroups (if time)												
Handling other monster attributes (if time)												
Dynamic difficulty adjustments (if time)												
Implementing more advanced AI (if time)												

Figure 4. Gantt chart of proposed timeline.