An Enhancement of HPA\* Pathfinding Algorithm Applied on Nightmares

Mark Daryl D. Caguia, Christian Neil Anthony S. Wico

Computer Science Department, Pamantasan ng Lungsod ng Maynila

caguiadaryl@gmail.com, wico.christian@gmail.com

**Abstract**

The popularity of video games in recent years has made an impact to society on what the media can do to people. But video games are not fun and games as they make it appear to be, problems persist as early as the media was created. One problem is how the AI (Artificial Intelligence) find the path from start to goal. The study is to help figure out a way to solve this problem through the enhancement of the pathfinding algorithm called HPA\*. A game was developed by the researchers to identify and solve the problems found on the pathfinding algorithm and applied an enhancement on the algorithm to solve the problems. Through this process, it was found that the algorithm is still lacking the ability to find a path when encountering dynamic, and non-rectangular obstacles. This was resolved by applying navmesh for scanning the obstacles the unit may encounter. Pathfinding is important to a game because just from its performance, the player’s experience from playing the game will change.

1. **Introduction**

Real Time Strategy is a sub-genre of strategy video games. Real time strategy games require the player to manage the units given to them under some certain scenarios, the game usually tasks the player to take control of a unit (civilization, city, group, etc.) and must improve it until the end of the game. In an RTS, the players must adapt to the environment, location, and condition of the unit they control and must devise strategies to keep its survival. With how RTS are usually played, the games under it shine the most when played by multiple players due to its competitive nature. That said, playing RTS games alone does not mean that the experience deteriorates.

When playing alone, players must deal with opposing units that are controlled by Bots or AI (Artificial Intelligence). These bots are programmed to deter the players of their progress by attacking and depriving them of resources. This gives the player a different yet still entertaining experience when playing RTS games.

Pathfinding algorithms are used to give the AIs a certain distinction and understanding on their surroundings. It lets them choose a path that will reach the destination with the least distance covered.

HPA\* consists of a build algorithm and a search algorithm. The build algorithm defines the hierarchy through a series of graphs, where each graph abstracts a higher resolution graph. After the hierarchy is prepared, the search algorithm finds a path at the highest level, and refines it into a series of segment paths along the lowest level. The utility of HPA\* is that a great deal of computation can be done in the preprocessing stage making the actual pathfinding task much faster. When a path is requested between locations a and b, all that is needed is to temporarily connect them to the pre-calculated graph by making small Dijkstra searches on the original cost raster within the blocks containing a and b, then calculate a path between them on the graph using A\*.

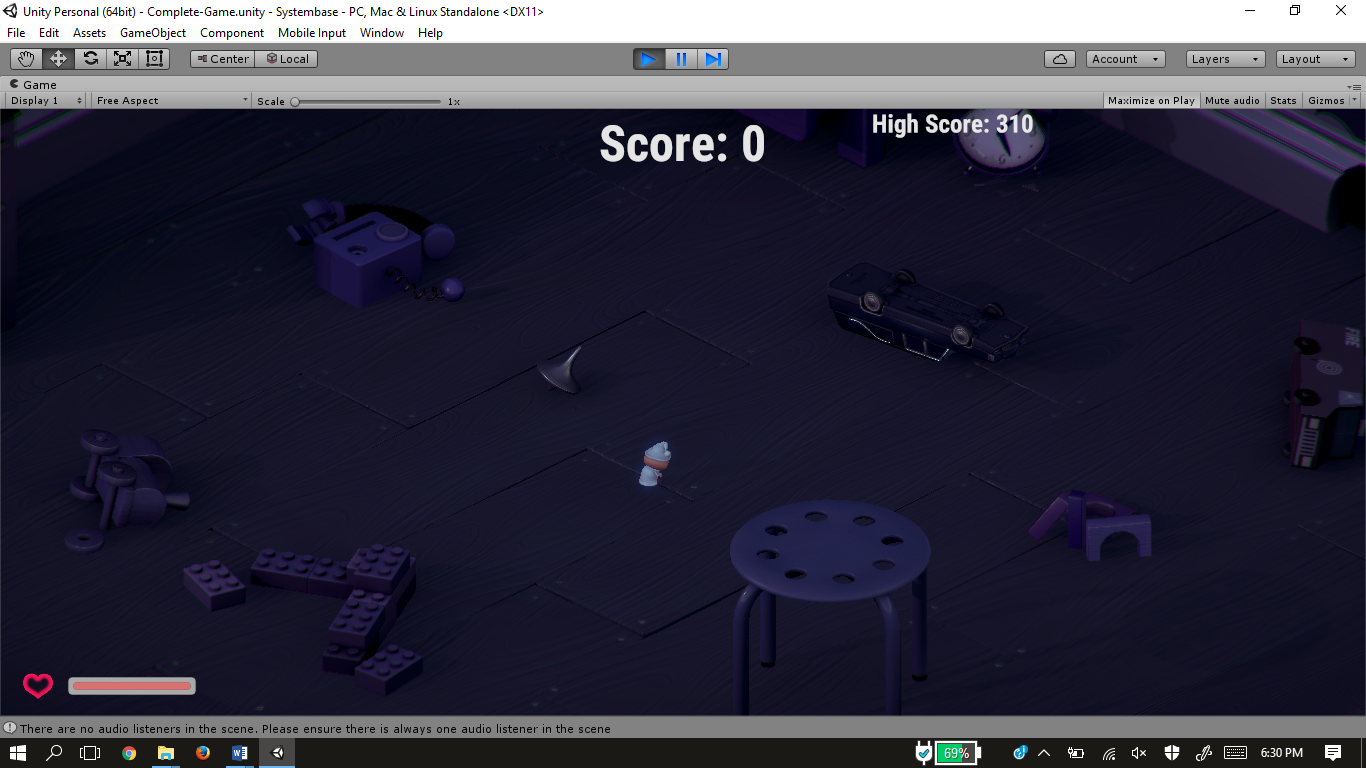
**2. Proposed Method**

In this section, we will be discussing about the optimality of the proposed pathfinding algorithm when it comes to finding a path when encountering a dynamic obstacle and/or non-rectangular obstacle. The HPA\* is an algorithm that focuses on grid-based maps, that it scans the whole map in a pre-processing manner. We applied navmesh on the scanned map by the algorithm to enhance its capability to scan different obstacles.

2.1 Navmesh

Navigational Meshes or NavMeshes, in contrast to HPA\*’s rectangular clusters, use interconnected triangles to form a mesh which represent traversable terrain. The reason why NavMeshes are more effective (though less efficient) in scanning for obstacles is because triangles can form much more polygons as composed to rectangles, which produce rather “pixelized” clusters. The primary drawback here is that because you need more triangles to cover an area as opposed to rectangles wherein each cluster can cover twice the area of a single rectangle.

NavMeshes have two sub-processes running within them: The Global Navigation process and the Local Avoidance process. The Global Navigation scans the entire map (static) and checks for stationary obstacles, carving holes into the generated navmesh as they encounter them. These “holes” tell the pathfinder that this part of the map is non-traversible. The other process, Local Avoidance, works only during the actual pathfinding. As a path is being calculated, the local avoidance searches for changes in the map which the global navigation may have missed since it only scans static obstacles. The difference between them is that the local avoidance only scans the path currently being traversed, in contrast to the global navigation which scans the entire map. Another is that the global avoidance only scans at the beginning of the map generation process and would not be re-executed until explicitly invoked by the user. A reason why is because navmesh generation takes up a significant amount of resources. Hence, global navmeshes are generated before actual gameplay begins. The local avoidance, on the other hand, scans repeatedly for as long as the agent (the unit currently guided by the pathfinder) has not reached the destination. The reason why is that local avoidance is made to search for dynamic obstacles, meaning they periodically change their states.

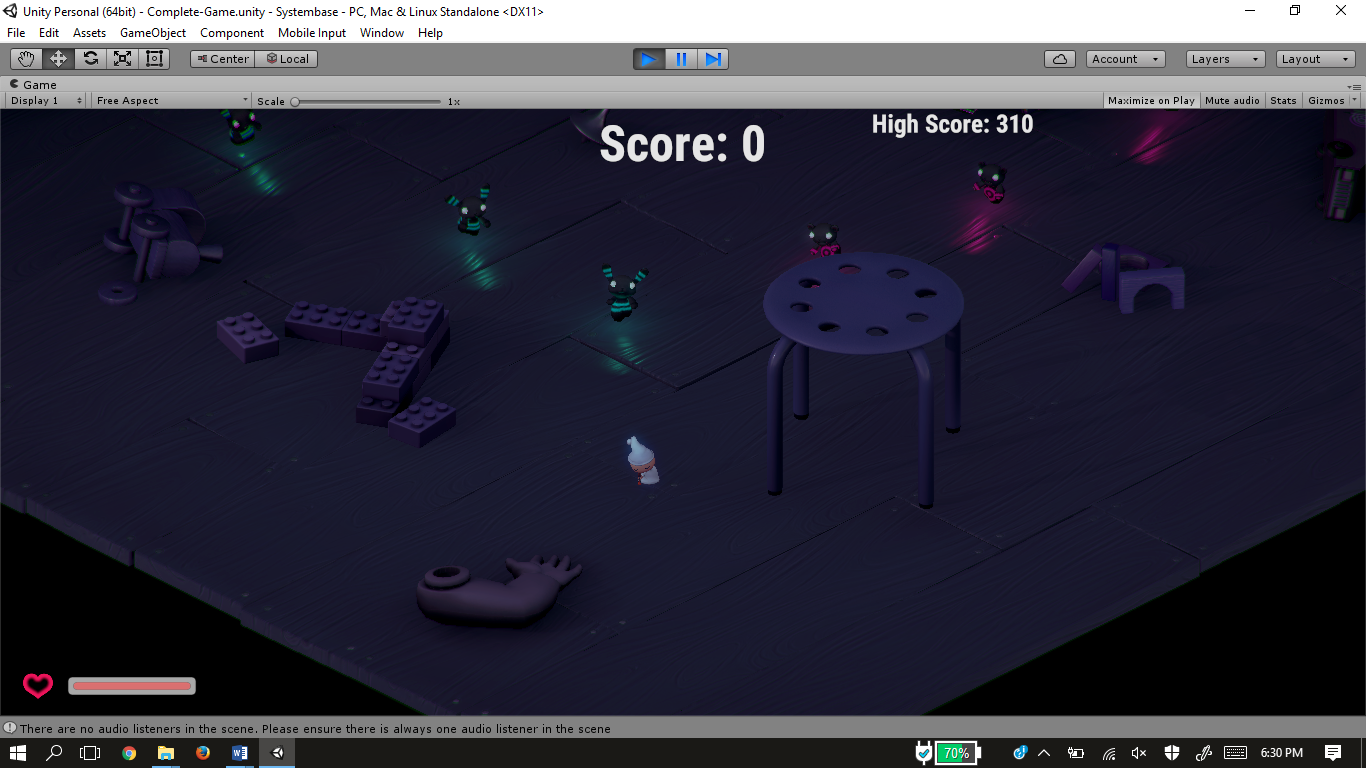


**Figure 1.** Interface of the game as it starts.

**3. Experimental Results**

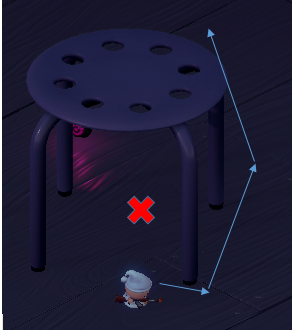
In order to evaluate the algorithms performance, we have subjected the system in to various experiments on how the algorithm will interact to different obstacles. In order to compare the HPA\* and the Enhanced algorithm, the system was run with the application of both algorithms. Each run shows the difference between the two algorithms about the performance of the system.

**Input**

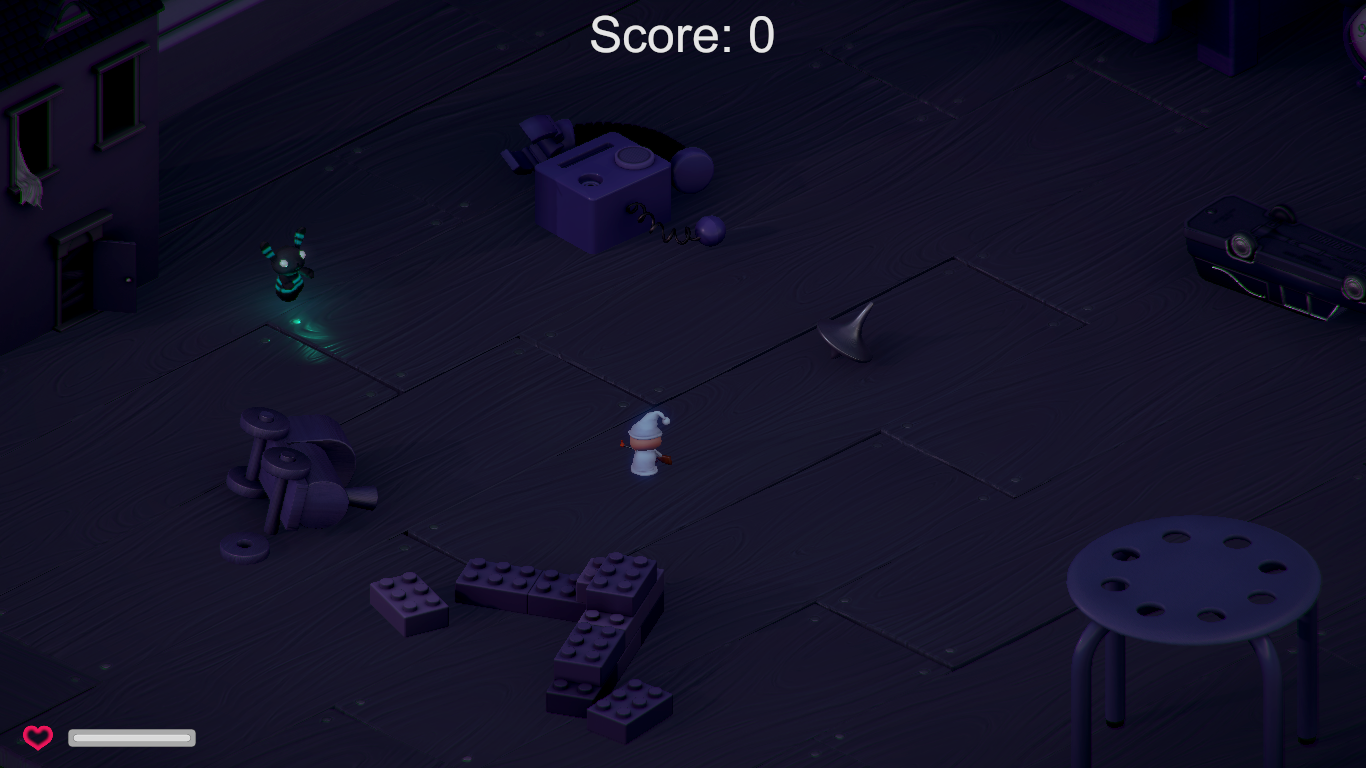


**Figure 2.** Interface including dynamic obstacle, non-rectangular obstacle, and the unit itself.

The figure above shows the various obstacles that proved to be a problem for the existing HPA\* algorithm.

**Existing**

**Figure 3.** Unit is unable to find the path between the obstacle.

****

**Figure 4.** Unit stops on its path whenever it encounters a dynamic obstacle.

**Enhanced****

**Figure 5.** Unit is now able to find and traverse the path now found by the existing HPA\* algorithm

**Figure 6.** Unit moves continuously towards its destination even when encountering dynamic obstacles

The results show the difference of the existing and proposed algorithm. The unit interacts with obstacles differently on which algorithm is applied. The existing algorithm **(Figures 3 and 4)** show that the algorithm has limitations when encountering different kinds of obstacles. This was resolved by the enhanced algorithm, where the algorithm is now able to interact properly **(Figures 5 and 6)** with these obstacles.

**4. Conclusion**

Since HPA\* is a “preprocessing” pathfinding algorithm, it is only logical to conclude that it will have problems trying to detect elements that it encounters beyond the preprocessing period. As such, we have tackled its issues head-on, employing other pathfinding technologies. We have discovered that to cover the “static” loophole in the algorithm, we needed to employ a fix that covers the same holes in another algorithm. This is how we found NavMesh. Since navmesh can cope with dynamic terrain so well, we decided it’s a good candidate to combine and fuse with HPA\*.

Using the efficiency of preprocessing pioneered by the HPA\* Algorithm to deal with the static part of the map in conjunction with NavMesh’s local avoidance to detect changes in the path and evade obstacles led to our enhancement of the HPA\* Algorithm.

**References**

L. van Elswijk, (2013) “Hierarchical Pathfinding Theta\*”

J. Vernette, (2012) “A Survey of Pathfinding Algorithms Employing Automatic Hierarchical Abstraction”, University of Windsor, Ontario, Canada

R. Engman, (2014) “HPA\* Used with a Triangulation-Based Graph”, Blekinge Institue of Technology, Karlskrona, Sweden

B. Anguelov, (2011) “Video Game Pathfinding and Improvement to Discrete Search on Grid-based Maps”, University of Pretoria, Pretoria

A. Strand-Holm Vinther, M. Strand-Holm Vinther, (2015) “Pathfinding in Two-dimensional Worlds”, Aarhus University, Aarhus, Denmark

A. Noori, F. Moradi, (2015) “Simulation and Comparison of Efficieny in Pathfinding Algorithms in Games”, Department of Computer at Technical and Vocational University, Tehran, Iran.

A. Kring, A. Champandard, N. Samarin, (2012) “DHPA\* and SHPA\*: Efficient Hierachical Pathfinding in Dynamic and Static Game Worlds”

Z. Bhathena, (2012) “Near Optimal Hierachical Pathfinding using Triangulations”, University of Texas, Austin, Texas