Computers & Graphics 59 (2016) 68–78

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Technical Section

Hierarchical path-finding for Navigation Meshes (*HNA**)[☆]

Nuria Pelechano*, Carlos Fuentes

Universitat Politècnica de Catalunya, Spain

A R T I C L E I N F O

Article history:
Received 27 October 2015
Received in revised form
18 May 2016
Accepted 20 May 2016
Available online 30 May 2016

Keywords:
Path-finding
Hierarchical representations
Navigation meshes

A B S T R A C T

Path-finding can become an important bottleneck as both the size of the virtual environments and the number of agents navigating them increase. It is important to develop techniques that can be efficiently applied to any environment independently of its abstract representation. In this paper we present a hierarchical NavMesh representation to speed up path-finding. Hierarchical path-finding (*HPA**) has been successfully applied to regular grids, but there is a need to extend the benefits of this method to polygonal navigation meshes. As opposed to regular grids, navigation meshes offer representations with higher accuracy regarding the underlying geometry, while containing a smaller number of cells. Therefore, we present a bottom-up method to create a hierarchical representation based on a multilevel k-way partitioning algorithm (*MLkP*), annotated with sub-paths that can be accessed online by our Hierarchical NavMesh Path-finding algorithm (*HNA**). The algorithm benefits from searching in graphs with a much smaller number of cells, thus performing up to 7.7 times faster than traditional A* over the initial NavMesh. We present results of *HNA** over a variety of scenarios and discuss the benefits of the algorithm together with areas for improvement.

1. Introduction

Most video games are required to simulate thousands or millions of agents who interact and navigate in a 3D world and show capabilities such as chasing, seeking or intercepting other agents. Path-finding provides characters with the ability to navigate autonomously in a virtual environment. The most well known path-finding algorithm is A*, which explores the nodes of a graph while balancing the accumulated cost with a heuristic to find an optimal path quickly. Throughout the years many algorithms have been proposed to further speed up the basic A* algorithm, but the cost of these algorithms is still strongly dependent on the size of the graph. Hierarchical path-finding aims to reduce the number of nodes that need to be explored when computing paths in large terrains. The reduction in the number of nodes for higher levels of the hierarchy significantly decreases the execution time and memory footprint when calculating paths.

Current hierarchical techniques may result in unbalanced abstractions. For example, top-down hierarchies are created by splitting the environment into large square clusters, where all the clusters contain the exact same number of lower level grid cells. The main disadvantages of such constructions are that the resulting higher level of the hierarchy may have an uneven number of edges between nodes and also an uneven number of walkable cells (since there may be some clusters with a large percentage of the grid cells being occupied by obstacles).

Navigation meshes represented by polygons provide closer representation of the geometry with a lower number of cells than regular grids. Since having a smaller number of cells can greatly accelerate path-finding, it is therefore necessary to extend the concept of hierarchical path-finding to a more general representation of navigation meshes with polygon based cells. Moreover it would also be beneficial to have a hierarchical representation with a balanced number of polygons per node and portals between nodes.

In this paper we present a new hierarchical path-finding solution for large 3D environments represented with polygonal navigation meshes. The presented solution works with navigation meshes where cells are convex polygons, and thus it also includes triangular representations. Our hierarchical graph representation is based on a multilevel k-way partitioning algorithm annotated with sub-path information. Our method presents a flexible approach in terms of both the number of levels used in the hierarchy and the number of polygons to merge between levels of the hierarchy. We evaluate the gains in performance when using our hierarchical path-finding, and discuss the trade-offs between the number of merged polygons and the number of levels employed for the search. We present a number of benchmarks that can help

[☆]This article was recommended for publication by R. Boulic.
* Corresponding author.
E-mail address: npelechano@cs.upc.edu (N. Pelechano).

http://dx.doi.org/10.1016/j.cag.2016.05.023
0097-8493/© 2016 Elsevier Ltd. All rights reserved.

during the parameter fitting process to achieve the best speedups, as well as a quantitative analysis of the bounds on sub-optimality of the paths found with *HNA**. We also present an evaluation of the bottleneck that appears for certain configurations when inserting the start and goal positions in the hierarchical representation.

2. Related work

A large amount of work to speed up path-finding focuses on enhancing the A* algorithm to reduce the computational time needed to calculate a path. This comes at the cost of finding sub-optimal paths or allowing a certain degree of error when searching for the optimal path and then allows the algorithm to repair those errors in future searches that are interleaved with the execution.

The well known A* algorithm [1] is a robust and simple to implement method with strict guarantees on optimality and completeness of solution. The A* algorithm uses a heuristic to restrict the number of states that must be evaluated before finding the true optimal path and it guarantees to expand an equal number or fewer states than any other algorithm using the same heuristic. However A* can be very time consuming for large scenarios. Anytime Planning algorithms find the best suboptimal plan and iteratively improve this plan while reusing previous plan efforts. One of the most popular A* is called Anytime Repairing A* (ARA*) [2]. It performs a series of repeated weighted A* searches while iteratively decreasing a loose bound (ε). It iteratively improves the solution by reducing ε and reusing previous plan efforts to accelerate subsequent searches. However ARA* solutions are no longer guaranteed to be optimal.

D* Lite [3] performs A* to generate an initial solution and repairs its previous solution to accommodate world changes by reusing as much of its previous search efforts as possible. D* can correct “mistakes” without re-planning from scratch, but requires more memory. Anytime Dynamic A* (AD*) [4] combines the properties of D* and ARA* to provide a planning solution that meets strict time constraints. It efficiently updates its solutions to accommodate dynamic changes in the environment.

DBA* algorithm [5] combines the memory-efficient sector abstraction developed for [6] and the path database used by [7] in order to improve space complexity and optimality. Huang [8] presented a path planning method for coherent and persistent groups in arbitrarily complex navigation mesh environments. The group is modeled as a deformable and splittable area preserving shape. The efficiency of the group search is determined by three factors: path length, deformation minimization, and spitting minimization.

Hierarchical graph representations have also been used for visualization purposes of large data sets [9,10]. The goal in these applications is to offer an overview first, and then be able to zoom and filter to offer details on demand.

Planning via hierarchical representation has been used to improve performance in problem solving for a long time [11]. Holte et. al. [12] introduced hierachical A* to search in an abstract space and use the solution to guide search in the original space. There has also been work on abstraction based on bottom-up approaches for general graphs [13,14] but without considering balancing the number of nodes or minimizing the edge-cut. Sturtevant and Jansen [15] extended the theoretical work slightly and provided examples of a number of different abstraction types over graphs. In this work graphs are created from 2D grid-like structures by setting a node for each walkable cell. Bulitko et al. [16] showed that the quality of paths can decrease exponentially with each level of abstraction. Sturtevant and Geisberger [17] studied the combination of abstraction and contraction hierarchies to speed up path-finding. Abstraction uses a top-down approach creating a 16×16 overlay across the lower level regular grid. Contraction builds a higher level graph using the concept of importance of nodes, which requires priorities for the nodes to be set correctly as they will affect the contraction algorithm.

Hierarchical representations have been used over 2D grid representations [18]. In [19] an adaptive subdivision of the environment is proposed with efficient indexing, updating, and neighbor-finding operations on the GPU which reduces the memory requirements. Another similar method based on *HPA**, but taking into account the size of the agents and terrain traversal capabilities, is Hierarchical Annotated A* (HAA*) [20]. It presents an extension of *HPA** which allow multi-size agents to efficiently plan high quality paths in heterogeneous-terrain environments. Another interesting implementation is *DT-HPA** [21] which uses a decision tree to create a hierarchical subdivision.

Jorgensen presented an automatic structuring method based on a hierarchy that separated buildings into floors linked by stairs and represents floors as rooms linked by doorsteps [22]. This method has a strict hierarchy and does not scale to large outdoors environments such as the ones often presented in video games. Zlatanova [23] presented a framework of space subdivision exclusively for indoor navigation, by identifying rooms and corridors and including semantical information.

There are other approaches that focus on allowing agents to be more environment-aware [24]. In this work planning is based on an Anytime Dynamic A*, and it is carried out satisfying multiple special constraints imposed on the path, such as: stay behind a building, walk along walls or avoid the line of sight of other agents. In [25] a multi-domain anytime dynamic planning framework is presented which can efficiently work across multiple domains by using plans in one domain to accelerate and focus searches in more complex domains. It explores different domain relationships including the use of way-points and tunnels. The different domains use only two representations in terms of spacial subdivision, a 2D grid, and a triangular mesh.

Hierarchical representations have been used to calculate agents moving between two points at different levels of complexity [26,27]; from finding a route to animating 3D characters. They have also been used to combine high level path-finding with low level local motion [28]. When using triangular representations, it is possible to optimize the data structures and built in features such as clearance that can greatly improve performance during path-finding [29,30]. But it is not straight forward to extend this implementation to polygonal meshes (i.e. it would not be enough with a simple triangulation of the polygons). There has been a recent technical report extending *HPA** to triangular representations [31].

As most of the abstract representations for large 3D complex environments employ polygon based representations (e.g: NEOGEN [32], Recast [33], or navmeshes built from the medial axis [34]), it is thus necessary to extend the concept of hierarchical path-finding for general representations of navigation meshes. Polygonal meshes have certain features and characteristics that must be taken into account when evaluating the most suitable hierarchical abstraction to be used.

3. Framework

Our framework consists of a pre-processing phase where the hierarchy is created, and an adapted version of the basic A* algorithm to perform searches online in this hierarchical representation.

The pre-process phase starts with a polygonal navigation mesh that represents an abstract partition of the 3D world. This first navigation mesh is considered to be the lowest level in a