Way to go - A framework for multi-level planning in games

Norman Jaklin, Wouter van Toll and **Roland Geraerts**

Utrecht University, Department of Information and Computing Sciences

Abstract

Path planning is one of the classical computational tasks in video games. Virtual characters need to autonomously find a path from their current position to a designated goal position. This is usually solved by running the A* algorithm on a grid or a navigation mesh. However, in many modern games, strictly following the resulting path is not sufficient. More levels of planning are necessary to efficiently simulate realistic and advanced behavior, and the underlying data structure should support those levels. In this paper, we discuss a five-level hierarchy of planning in games. Furthermore, we present a framework that provides solutions for the three center levels: global route planning, route following, and local planning. It uses an efficient and flexible navigation mesh based on the exact geometry of the environment. Our framework can be extended to solve advanced path planning problems in future games. When used as an interface for higher-level semantic planning systems, it provides a comprehensive set of techniques for game developers and path planning researchers.

Introduction - the different levels of planning

Similarly to domains such as graphics, animation, or physics simulation, the field of path planning in video games has increased in complexity over the last decades. This aspect of game AI has been studied extensively (Rabin 2002). It might seem that path planning problems have been solved by algorithms such as A* (Hart, Nilsson, and Raphael 1968). However, in modern games, path planning is still limited with respect to an ever-increasing demand for new features that enhance player immersion. In addition, even when solving allegedly simple tasks such as letting a character reach a goal position, some games still suffer from flaws due to approximated graph-based representations of the navigable space.

A* is a valuable method to find global shortest paths in any graph structure. In simple cases, one could use a rectilinear grid, which does not lead to any information loss if the game world consists of rectilinear tiles. A* on a basic graph also works well if the game does not require any advanced features such as visually convincing and smooth trajectories, clearance from obstacles, collision avoidance between characters, path planning in environments with multiple height levels, reacting to dynamic changes in the environment, dealing with characters of various sizes, or taking other environmental factors into account, e.g. different terrain types or crowd density information.

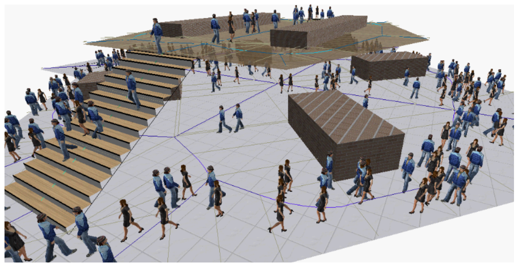
Figure 1: Autonomous virtual characters in a multi-layered 3D environment using the *Explicit Corridor Map* (ECM) (van Toll, Cook IV, and Geraerts 2011)

Modern games, however, do require such advanced features. Viewed from this perspective, computing a global shortest path from a start to a goal position is only one aspect in a multi-level hierarchy of planning systems. We propose five levels of planning that a modern video game might require. Figure 2 shows this five-level hierarchy.

At the top of the hierarchy, *high-level planning* (5) translates the desired *semantic* behavior of a character to *geometric* path planning problems. For example, the character could have an abstract task such as ‘steal a stash of gold’. This can be converted to a list of more concrete tasks, e.g. ‘enter the village, find character X, plunder its chest full of gold and leave the village without being seen’. Based on this plan, the character should compute an ordered list of goal positions. High-level planning is a research topic of its own, involving techniques such as *STRIPS* (Fikes and Nilsson 1971) and *Hierarchical Task Networks* (Kelly, Botea, and Koenig 2008).

Next, the *global route planning level* (4) uses the list of goal positions to compute geometric routes through the environment. This is where a classical method such as A* might be used, provided that the underlying graph structure does not yield any drawbacks with respect to the tasks at hand.

The three lower levels update the character in every step of the simulation. On the *route following level* (3), the global routes are being traversed. Depending on the application, this can be either a strict and simple following routine, or an advanced method that creates visually convincing tra-

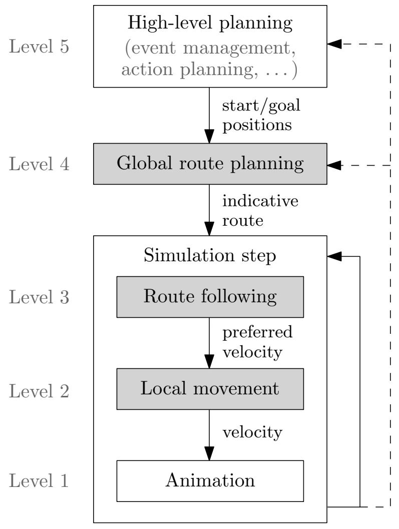

Figure 2: The five-level hierarchy of planning in games.

jectories while take other parameters such as terrain types into account. On the *local movement level* (2), the character might temporarily deviate from its global route to resolve local collision avoidance with other characters or to react to dynamic changes in the environment. Finally, the *animation level* (1) handles the actual animation down to the skeleton representation of the character model.

This planning process is not purely serial: events in the lower levels may cause a character to reconsider its global plans. For instance, if a part of the environment turns out to be too crowded, a character may choose to take a detour.

In this paper, we present an efficient and flexible framework for levels 4, 3, and 2, i.e. the levels that concern geometric path planning. We deliberately treat the high-level planning phase as a black box, and we argue that our framework can be plugged into any game AI system that follows the suggested hierarchy.

The Explicit Corridor Map framework

The core of our framework is a navigation mesh called the *Explicit Corridor Map* (ECM) (Geraerts 2010). We assume that the environment consists of polygonal obstacles. The ECM is based on the *medial axis*, which is the set of all points that have at least two distinct closest obstacle points in the environment. The medial axis is closely related to the Voronoi diagram, which is a fundamental data structure in the field of computational geometry (de Berg et al. 2000).

Figure 3 shows an example. The medial axis can be seen as a special type of waypoint graph in which all edges run through the middle of the free (or traversable) space. For each vertex of this graph (shown as big black discs), there are either at least three different nearest obstacle points, or the vertex is placed in a non-convex corner of an obstacle. An edge of the medial axis consists of a sequence of line segments and parabolic arcs, depending on the type of obstacles to the left and right. For a 2D environment with n obstacle vertices, the medial axis has $O(n)$ complexity and

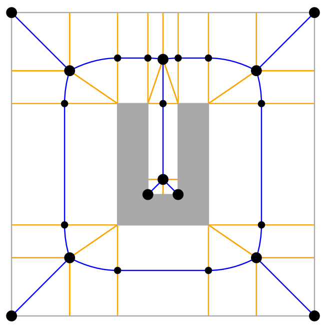

Figure 3: A 2D environment with obstacles (shown in gray). Its ECM is the medial axis (blue) annotated with closest-obstacle information (orange) at a selection of points. This subdivides the traversable space into polygonal regions.

can be constructed in $O(n \log n)$ time. Alternatively, one can use graphics hardware to robustly approximate the structure (Hoff III et al. 1999).

The ECM is an *annotated medial axis*: it stores the left and right closest obstacle points for each edge section. This partitions the environment into a set of polygonal walkable regions. Recently, we have extended the medial axis and the ECM to multi-layered 3D environments (van Toll, Cook IV, and Geraerts 2011). An example of a crowd in a multi-layered ECM is shown in Figure 1.

The ECM has many features that make it well-suited for our framework. All space is represented with respect to the exact geometry of the environment. This resolves the issues that are inherent to approximated representations such as grids or waypoint graphs. Furthermore, the ECM is space-efficient and supports time-efficient extraction of global paths with any desired amount of clearance from obstacles. It therefore supports characters with arbitrary sizes. The ECM is well-defined for both 2D and multi-layered 3D environments. In addition, we have shown that the ECM can be efficiently updated in response to insertions and deletions of obstacles (van Toll, Cook IV, and Geraerts 2012a). Finally, the concept of the ECM is general enough to allow for many extensions and advanced planning methods that build upon it, as will be illustrated in the next section.

Contributions to the planning hierarchy

Our framework comprises methods and techniques that provide efficient real-time solutions for the second, third and fourth levels of the hierarchy. Hence, it can be applied to *geometric* planning problems induced by a *semantic* high-level planner. We will now discuss the contributions in detail.

Global route planning

A global route planner should compute an *indicative route* from the character's start s to its goal position g. Formally, an indicative route can be any curve $\pi_{ind} : [0,1] \rightarrow \mathbb{R}^2$ through the free space of the environment. In practice, we implement such a route as a sequence of points connected by straight-line segments that do not intersect any static obstacles. The concept of using an indicative route for path

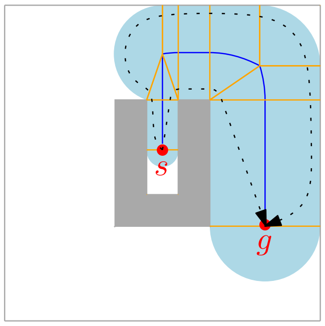

Figure 4: In the ECM, a path along the medial axis (blue) induces a corridor (light blue) due to its closest-obstacle annotations. Within the corridor, we can define any indicative route from s to g; two examples are shown in black.

planning has first been introduced in the *Indicative Route Method* (Karamouzas, Geraerts, and Overmars 2009).

There are various approaches to compute a global indicative route. For instance, we have implemented A* on the ECM to find shortest paths along the medial axis. This is generally more efficient than performing A* on a grid due to the sparseness of the ECM structure. Furthermore, with the clearance information stored in the ECM, characters of all sizes can use the same graph without having to inflate the obstacles in a preprocessing step.

The optimal route through the ECM does not have to be the *shortest*; optimality can also be based on other criteria. For instance, we have shown how to map *crowd density* information onto the regions induced by the ECM (van Toll, Cook IV, and Geraerts 2012b). By using density information in the A* algorithm, characters can prefer paths with little expected delay. In practice, they will plan detours around congested areas, and the crowd will automatically spread over multiple routes of different homotopy classes.

Performing A* on the ECM always yields a *corridor*, which is a sequence of medial axis edges plus a description of the surrounding free space. A corridor represents a subset of the free space in which valid indicative routes that belong to the same homotopy class are contained. We can therefore create various indicative routes from s to g, e.g. a route that stays on the left or right side of the corridor, or the shortest route in the corridor with a preferred amount of clearance to obstacles (Geraerts 2010). Figure 4 illustrates this concept.

We have also created a method to find *stealthy global paths* with limited exposure to other characters, i.e. a path that lets the character stay unseen by other moving characters as much as possible. To this end, we computed *visibility information* on the GPU and mapped it onto an extended version of the ECM (Schager and Geraerts 2010).

Only recently, we added various *terrain types* to our virtual environments. Those can be used to ensure that a character plans its global route based on a set of individual terrain preferences. For example, a pedestrian might prefer to walk on the sidewalk while avoiding roads, puddles or muddy terrain. We refer the reader to Figure 6 for an example.

Lastly, we have developed a planning approach based

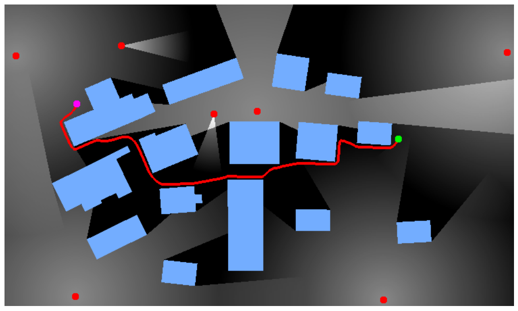

Figure 5: We have used an extended ECM to plan stealthy paths based on visibility information.

on linear programming (Karamouzas, Geraerts, and van der Stappen 2012). It coordinates an entire crowd consisting of one or more independent groups of characters. The method efficiently computes the most promising paths in both time and space and yields an optimal distribution of the groups members over these paths. Thus, the characters' average traveling time is minimized. The computed space-time plan is then combined with an agent-based steering method to handle collisions and generate the final motions of the characters. The method runs at interactive rates and is able to solve complex planning problems involving one or multiple groups in gaming or crowd simulation applications.

The result of the global planning level serves as input to the *route following* level, which we will discuss next.

Route following

In this level, an indicative route π_{ind} is given. The character is supposed to follow the route, but it is allowed to deviate from it. Our framework is built to switch between different path following methods. Our methods use the concept of an *attraction point* p_{att} that lies on π_{ind} to generate smooth paths. In each step of the simulation, the character picks a new p_{att}, which directly leads to a *preferred velocity* v_{pref} for the character in the current step. The direction of v_{pref} is the vector from the character's current position to p_{att}; its magnitude is the character's preferred speed.

We implemented two different path following methods that use attraction points. Firstly, the *Indicative Route Method* (Karamouzas, Geraerts, and Overmars 2009) uses the clearance information provided by the ECM. It defines p_{att} as the last point along π_{ind} that intersects the character's clearance disk. When more free space is available, p_{att} lies farther along the route and larger parts of π_{ind} are skipped, i.e. the amount of smoothing increases.

Secondly, we have introduced a more general path following method named *MIRAN* (Jaklin, Cook IV, and Geraerts 2013 to appear), in which the user can control the character's look-ahead distance and its eagerness to take shortcuts. Furthermore, *MIRAN* lets characters plan their paths with respect to their individual terrain preferences. In other words, the amount of smoothing and route shortening depends on the local terrain costs for that particular character. Figure 6 shows an example of this method.

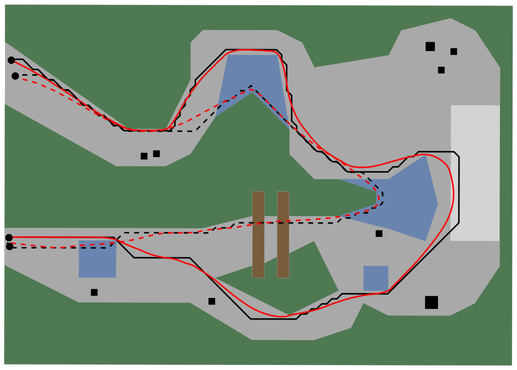

Figure 6: A path (gray) in a forest (green) with obstacle trees (black), puddles (blue), fallen trees (brown) and a spot with a panoramic view (light gray). Two characters (adult and child) follow automatically computed indicative routes (solid and dashed black). The smoothed paths (solid red for the adult, dashed red for the child) are computed with our MIRAN method. Both the indicative routes and the paths are based on the characters' terrain preferences.

The *local movement* level is the last remaining one before the actual *animation* on the skeleton level is handled. We will now discuss in what way our framework covers it.

Local movement

In a virtual crowd, the characters may need to adjust their velocities to avoid collisions with other characters. The task of the *local movement* level in our framework is to compute an actual velocity v for each character, based on its preferred velocity v_{pref} and other crowd members in its vicinity.

Many solutions for this *collision avoidance* problem are available. Early algorithms defined repulsive forces between characters (Helbing and Molnár 1995). Modern methods prevent future collisions based on the perceived velocities of other characters, while deviating from v_{pref} as little as possible. Our framework includes one such approach (Karamouzas and Overmars 2010), but it can support any other velocity-based algorithm, such as the popular RVO library (van den Berg, Lin, and Manocha 2008). Note that collision detection for *static* obstacles is trivial in our framework, because the ECM explicitly stores the nearest obstacle for any point in the free space.

Conclusion and future work

We have given an overview of our framework that covers the three center levels of a five-level planning hierarchy. Those levels comprise the *geometric* aspects of planning in games and can be combined with higher-level planning systems.

Our framework is general enough to support various future extensions and improvements. One possible extension is to take characters of different heights into account. For instance, big vehicles may not fit through small tunnels that regular characters can use. Another extension could be visibility-based planning, e.g. letting characters re-plan their paths based on the dynamic changes they can perceive visually. Finally, we could extend the *MIRAN* method so that it does not only affect the *global planning* and *path following* levels of the hierarchy, but also the *local movement* level, e.g. by including terrain-based collision avoidance.

As we have shown, our framework is flexible and enables future extensions. We therefore believe that it provides a comprehensive set of techniques for game developers and path planning researchers.

Acknowledgements

This research has been supported by the COMMIT project (http://www.commit-nl.nl/) and the COMMANDS project in cooperation with the European Design Center (http://www.edc.nl/) and the Netherlands Forensic Institute (http://www.forensicinstitute.nl/). The authors are part of the Institute of Information and Computing Sciences, Utrecht University, 3584 CC Utrecht, the Netherlands.

References

de Berg, M.; van Kreveld, M.; Overmars, M.; and Schwarzkopf, O. 2000. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, second edition.

Fikes, R., and Nilsson, N. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence* 2(3/4):189–208.

Geraerts, R. 2010. Planning short paths with clearance using Explicit Corridors. In *IEEE International Conference on Robotics and Automation*, 1997–2004.

Hart, P.; Nilsson, N.; and Raphael, B. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics* 4(2):100–107.

Helbing, D., and Molnár, P. 1995. Social force model for pedestrian dynamics. *Physical Review E* 51(5):4282–4286.

Hoff III, K.; Culver, T.; Keyser, J.; Lin, M.; and Manocha, D. 1999. Fast computation of generalized Voronoi diagrams using graphics hardware. In *International Conference on Computer Graphics and Interactive Techniques*, 277–286.

Jaklin, N.; Cook IV, A.; and Geraerts, R. 2013 (to appear). Real-time Path Planning in Heterogeneous Environments. *Computer Animation and Virtual Worlds*.

Karamouzas, I., and Overmars, M. 2010. Simulating human collision avoidance using a velocity-based approach. In *Workshop on Virtual Reality Interactions and Physical Simulations*, 125–134.

Karamouzas, I.; Geraerts, R.; and Overmars, M. 2009. Indicative routes for path planning and crowd simulation. In *International Conference on Foundations of Digital Games*, 113–120.

Karamouzas, I.; Geraerts, R.; and van der Stappen, A. 2012. Space-time group motion planning. In *Workshop on the Algorithmic Foundations of Robotics*, 227–243.

Kelly, J.; Botea, A.; and Koenig, S. 2008. Offline planning with Hierarchical Task Networks in video games. In *Artificial Intelligence and Interactive Digital Entertainment Conference*, 60–65.

Rabin, S. 2002. *AI Game Programming Wisdom*. Charles River Media.

Schager, E., and Geraerts, R. 2010. Stealth-based path planning in corridor maps. In *Computer Animation and Social Agents*.

van den Berg, J.; Lin, M.; and Manocha, D. 2008. Reciprocal Velocity Obstacles for real-time multi-agent navigation. In *IEEE International Conference on Robotics and Automation*, 1928–1935.

van Toll, W.; Cook IV, A.; and Geraerts, R. 2011. Navigation meshes for realistic multi-layered environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 3526–3532.

van Toll, W.; Cook IV, A.; and Geraerts, R. 2012a. A navigation mesh for dynamic environments. *Computer Animation and Virtual Worlds* 23(6):536–546.

van Toll, W.; Cook IV, A.; and Geraerts, R. 2012b. Real-time density-based crowd simulation. *Computer Animation and Virtual Worlds* 23(1):59–69.