Generative Methods for Guard and Camera Placement in Stealth Games

Qihan Xu and **Jonathan Tremblay** and **Clark Verbrugge**
School of Computer Science
McGill University
Montréal, Québec, Canada
qihan.xu@mail.mcgill.ca, jtremblay@cs.mcgill.ca, clump@cs.mcgill.ca

Abstract

Enemy observers, such as cameras and guards, are common elements that provide challenge to many stealth and combat games. Defining the exact placement and movement of such entities, however, is a non-trivial process, requiring a designer balance level-difficulty, coverage, and representation of realistic behaviours. In this work we explore systems for procedurally generating both camera and guard placement in a stealth game context. For the former we use an approach based on weakening theoretical results for optimal camera placement, while for the latter we perform automatic roadmap construction, generating more specific patrol behaviours through a grammar-based technique. We evaluate both approaches with a non-trivial implementation in Unity3D, and apply quantitative metrics to demonstrate how different parametrizations can be used to control level difficulty without sacrificing believability.

Introduction

A stealth problem in a game requires a player to move between locations, undetected by entities such as cameras or AI-controlled guards. Defining the location, routes, and other parametrization of enemy agents, however, is a non-trivial task, made especially complex by the need to ensure a solution exists, provides an appropriate challenge to the player, and that agent behaviours appear suitably realistic for the game context.

We investigate two methods for procedural placement of enemy agents, considering first camera placement and then the construction of mobile guard routes. For the former, solutions to camera (static guard) placement can be related to the well known "art gallery" problem in computation geometry (O'Rourke 1987), which seek to optimally place cameras to ensure complete coverage of a polygonal space. Weakening these theoretical results by both limiting individual camera coverage and by reducing the number of cameras then provides a parametrized method for ensuring an incomplete coverage, and thus a potentially solvable level design.

Mobile guards introduce the additional complexities of planning reasonable movement routes that tour a space, along with specific behaviours guards may use to inspect the space at different points. For this we first build a simplified roadmap based on a reduced medial skeleton (Blum 1967); this enables us to select appropriate end-points and paths between them that traverse the space without appearing random. Additional behavioural complexity is then introduced through a grammar-based model that partitions the route into different kinds of movement and scanning segments.

Both approaches have trivial reductions that can ensure solutions exist. For more interesting results we use a heuristic, Monte-Carlo approach. An initial, arbitrarily complex arrangement of enemy agents is first constructed, and then verified for solution existence and level complexity through a search-based analysis technique. We demonstrate this process on two game level designs, one synthetic and one based on a commercial game, exploring different configurations to give some sense of appropriate parametrizations for our techniques. Specific contributions of this work consist of:

1. A heuristic approach to camera placement based on weakening a solution to the well known "art gallery problem" for simple polygons.
2. The design of a flexible, grammar-based method for defining roadmap-based guard patrol routes.
3. Application of quantitative metrics that demonstrate how different parametrizations affect the existence of level solutions and player perception of difficulty.

Background & Related Work

Stealth problems can be loosely defined as tasks wherein the player must move from a start to a goal position, while avoiding enemy Non-Player Character (NPC) Fields of View (FoV). This problem is presented to players in many combat games, and is central to games in the stealth genre, such as *Mark of the Ninja* or the *Metal Gear Solid* series.

Designing such a game or level involves careful orchestration of enemy positions and behaviours, obstacles, and other positive or negative factors that affect stealth (light, shadow, sound, etc.) (Smith 2006) so as to present an interesting but feasible challenge to the player. This requires solving problems in geometric visibility along with procedural generation.

Within our exploration of stealth games, the design space is limited to placing geometries (occlusions) and two flavours of NPC: guards and cameras, both of which have

limited FoV. Guards refers to entities moving along a repeating, pre-determined path (patrol route). Cameras do not move, although they can rotate, sweeping a given subset of the level. The quality of a stealth level is typically measured through playtesting, although recent work has suggested that objective, quantitative measures can also approximate perceived risk (Tremblay, Torres, and Verbrugge 2014).

Abstract solutions to related problems have been explored in more theoretical contexts. Placing cameras (with infinite FoV) in a given n-vertex polygonal geometry is well known as Klee's 1973 "art gallery" problem, and there have been many different algorithms and lower bounds demonstrated for variations of this problem (O'Rourke 1987). More complex formulations also exist; Erdem *et al.*, for example, presented a realistic expression of the art gallery problem where guards do not have 2π rotational and infinitely long view (Erdem and Sclaroff 2006). Using a discrete (grid-based) world they optimally place cameras that can ensure every point is observed with period less than t. For this they reduced the problem to a *Set Coverage Problem*, solving it using a special case of integer programming. Cohen-Or *et al.* give a general survey of other, real-life application of camera network placement (Cohen-Or et al. 2003). In the context of games, a combined approach to guard/camera placement has been previously explored by Xu *et al.* (Xu, Tremblay, and Verbrugge 2014). They presented a space-decomposition approach that enabled them to locate cameras with (heuristically) maximized coverage, and to define simple, straight-line guard patrol paths. Our work here generates significantly more complex guard routes and behaviours, and provides an approach to camera placement that can both guarantee good coverage properties, and has less stochastic variance in scaling down the difficulty.

Our camera/guard placement approach is intended to help in procedurally generating stealth levels. Procedural Content Generation (PCG) has been used for a wide variety of tasks in games, level generation (*Spelunky*), weapon generation (*Borderlands*), vegetation filling (*Skyrim*), *etc.,* and continues to be a popular research topic. Shi and Crawfis, for instance, presented a design tool that tries to optimize distribution of objects within a level, based on different properties of player paths, such as the minimum-damage cover, longest path, and standard deviation of cover points (Shi and Crawfis 2013). In our work we enable flexible, high-level structuring of the content generation through use of a grammar-based rewrite system, generating increasing complexity by applying rewrite rules. Grammar-based approaches have been used for a number of other PCG applications, including vegetation (Togelius, Shaker, and Dormans 2014), abstract structure for platformer levels (Smith et al. 2009), narratives (Kybartas and Verbrugge 2014), and even as a general tool that creates whole story, missions and levels (Dormans 2010).

Strategies for Camera and Guard Placement

Although 3D elements are sometimes used in stealth games, the stealth pathing problem is most typically a 2D one, and thus can be addressed in terms of its underlying floor-plan. Below we describe our two main approaches to enemy agent placement, beginning with camera placement, and followed by mobile guard placement.

Camera Placement

Our approach to camera placement builds on known algorithms for achieving a minimal camera arrangement shown to be theoretically optimal, and which we subsequently weaken. In this way we can avoid excessive, unrealistic overlap in coverage, as well as be sure that both hard and easy solution end-points exist.

We start the placement by following Fisk's 1978 algorithm for a minimal covering of a simple polygon, using $\lfloor \frac{n}{3} \rfloor$ cameras for a polygon with n vertices (O'Rourke 1987). This algorithm begins with a triangulation of the polygon interior, the vertices of which are then 3-coloured, ensuring no adjacent vertices share a colour. Each triangle must then use all 3 colours, and since triangles are convex, associating one of the 3 colours with camera placement results in a complete coverage of the polygon interior.

The presence of obstacles in our floor-plan, however, means that we are actually working on a polygon with holes, and so Fisk's approach for simple polygons does not apply. More complex algorithms exist that address this concern (Hoffmann, Kaufmann, and Kriegel 1991), but since a theoretically minimal solution is not actually required, we can take a simpler approach. For this we continue to make use of a triangulation, using the well known technique of reducing a polygon with holes to a simple polygon through a network of (conceptually) infinitely thin channels that connect all obstacles to the exterior, effectively converting the holes (obstacles) into part of the polygon exterior. Figure 1 shows an example, where thick (blue) edges indicate the channels. In this approach a 3-colouring is still possible, with the drawback that since each channel now represents two exterior edges we have 2 overlapping vertices at the vertices of each channel, and so end up with a slightly less optimal camera placement using up to $\lfloor \frac{n+2h}{3} \rfloor$ cameras for an n-vertex polygon with h holes.

Finally, as these approaches assume full, 360° camera view, and infinite range, we can change camera parameters in order to permit more and easier solutions. We experiment with reductions in the camera FoV angle and distance, while imposing fixed limitations on rotation speed/patterns. Other factors may also be considered, such as focus, light sensitivity, or more complex, 3D FoV and movements.

Guard Placement

Guard placement introduces a number of additional complexities to defining enemy agents. First, guards are expected to be mobile, and thus we need not only an initial position, but a patrol route to follow as well. Second, guards are usually expected to make occasional observations as they patrol, sometimes looking around rather than just staring forward as they move.

Theoretical bases for good coverage with such guards can be developed, either by extending camera visibility results to consider mobile guards following interior paths, within the context of a *weak* visibility model (Ghosh 2007),

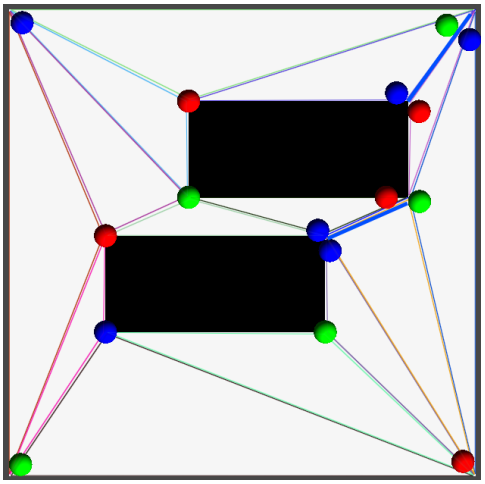

Figure 1: Triangulating and colouring a polygon with holes. The blue edges represent channels, consisting of two parallel edges, and chosen so as to form a spanning tree that connects all obstacles to the exterior.

or through exploration algorithms more commonly used in robotics (Obermeyer, Ganguli, and Bullo 2011). The resulting algorithms, however, tend to have high computational complexity, or generate movement patterns that do not seem suitable for modelling guards in games—weak visibility solutions tend to favour (unrealistic) wall-hugging, while optimal exploration strategies result in minimized, hard to predict movement that does not match player expectation for guards on routine patrol.

Our approach is thus based on two techniques that can easily and efficiently ensure an appropriate result for guard simulation, while still providing ample, flexible control for difficulty adjustment. We perform this in two stages, first constructing a basic patrol route, and then introducing intra-route activities to add further complexity and interest.

Patrol routes. Guards should move in a repetitive fashion through some subset of the level geometry, concentrating on large areas, and avoiding obstacle features. For this we generate a roadmap as a reduced and straightened subset of the medial skeleton. We expand all edges, flooding outward from obstacles and inward from the boundary. When different obstacle or boundary/obstacle wavefronts meet we generate roadmap edges. We use a discrete representation for this, and so we perform a final smoothing step to remove zig-zags and other discrete artifacts, and also reduce the number of nodes in the roadmap. The end result is something close to a simplified straight skeleton (Aichholzer et al. 1995), but absent edges that would connect to polygon vertices, or which would otherwise normally be generated by interaction of wavefronts from different edges of the same polygon (obstacle or boundary). This approach avoids the potential for guard paths that would walk to or from corners and concentrates guard behaviour on touring around obstacles. Figure 2 shows an example of a basic roadmap, and figure 3 shows the result after smoothing and node reduction.

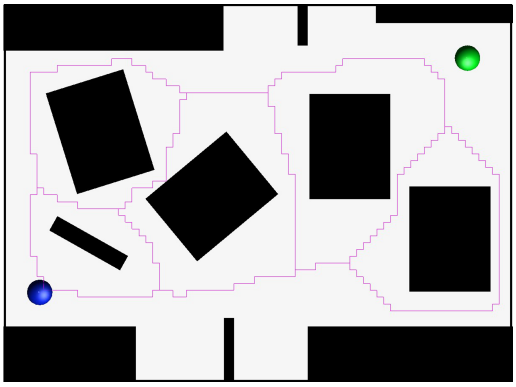

Figure 2: Test level with initial skeleton roadmap. The blue sphere (lower-left) indicates the player start position, and the green sphere (upper-right) the goal.

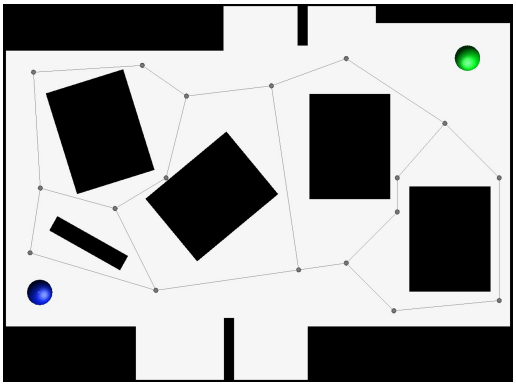

Figure 3: Test level with simplified roadmap.

Given a roadmap network, we then select patrol routes by randomly choosing starting and ending nodes within the network (different for each guard), and constructing a path between them through a simple depth-first search, biasing and randomizing the search to generate relatively long paths. Patrol routes are assumed to be cyclic, with the final position matching the starting position: given two distinct end-points, a and b, the route consists patrols from a to b and then back to a. Note that it is possible that $a = b$, in which case the route will consist of a random tour through the network that returns to a.

Intra-route activities. The basic route generation results in guard patrols that have reasonably good level coverage, but are overall somewhat robotic in behaviour. More human-like behaviours are created by breaking up the patrol movement, introducing points at which a guard stops and turns to scan the space in different fashions. For this we use the grammar-based construction, shown in figure 4.

However many roadmap edges are traversed, a patrol route is initially a single segment of continuous movement. The first few rules of this grammar break up continuous segments of movement into multiple segments; rule 1 rewrites a segment into two, not necessarily equal length segments separated by an activity (•), while rule 2 rewrites a segment

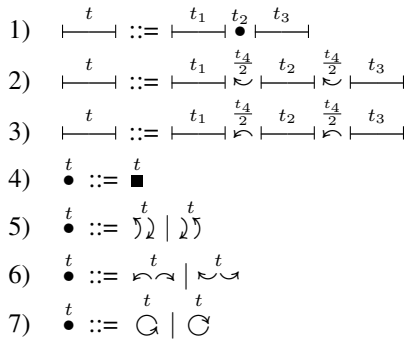

Figure 4: Guard patrol grammar.

into 3 sections with 180° turns between (rule 3 is symmetric, turning in the other direction), and so causes the guard to go back and forth over the same segment. Rule 4 rewrites the abstract activity designator into a pause (■), while the remaining rules rewrite an activity into different kinds of scanning: a 90° turn and return, left or right (rule 5), a 180° turn and return (rule 6), or a full 360° rotation (rule 7).

All rules have a time value associated with each segment or activity. This allows us to control the pace of individual actions; for example, in rule 1 the time taken to traverse the LHS segment t is broken up into time values t_1, t_2, and t_3 for the three parts of the rewritten segment. Control over patrol timing may thus be enforced ahead of time by forcing the sum of times in the RHS of each rule to match the time in the LHS, although we do not do this in our prototype.

Applying these rules to the initial guard path, following different rule selection strategies and limits on rewriting depth gives control over the complexity of the patrol behaviours. In the next section we explore the relative impact of both route selection and activity choice/density.

Experiments & Results

In order to validate our approach we examined the effects of camera and guard placement on multiple game-levels, including a non-trivial test level as well as a mock-up of the first level from Metal Gear Solid (MGS), the Dock. We measured the impact on game challenge through quantitative metrics to determine how we can relate varying parametrization to different measures of player difficulty. Below we describe our metrics, followed by results from camera placement and then guard patrol paths.

Metrics

To measure the difficulty of our stealth levels, we make use of two different metrics, one that measures relative feasibility, and another that measures the perception of risk.

We determine the degree of feasibility by computing the success rate of a heuristic path exploration technique. Path-finding techniques are able to solve stealth levels, at least with deterministic enemy behaviours, by modeling the game state as a geometric space extruded over time. A path that reaches the goal position, while continually advancing in time and avoiding all enemy FoV at each point in time is then a solution to the stealth level. We compute such paths using a Rapidly Exploring Random Tree (RRT) search (Morgan and Branicky 2004) rather than a more traditional A* or other deterministic approaches, since the RRT paths may fail, tend to have a significant, sub-optimal randomness to the result, and so better approximate a variety of human gameplay behaviours. We thus use the *success ratio* of RRT searches as a proxy for overall level difficulty.

Other metrics have also been proposed and evaluated for measuring properties of stealth levels. Tremblay *et al.* describe 3 different path quality metrics that attempt to measure the player sense of risk or danger (Tremblay, Torres, and Verbrugge 2014). Through comparison with results from a human study they argue that a conceptually simple path-distance measure, considering the relative distance from a player to enemies at each point best approximates human perception. We thus apply this metric in order to determine the degree of risk our levels present to players. Note that this metric is inverted, and higher values in the distance metric indicate riskier paths that tend to be closer to enemy agents.

Both metrics are applied to game levels modeled in a non-trivial Unity3D game development framework, and so consider realistic player movements, including physical and visual constraints typical of modern first or third-person 3D environments.

Camera Analysis

Our algorithmic basis for camera placement requires we place cameras at particular points in the space. It is of course possible to introduce variation into that placement as a means of perturbing difficulty, but randomizing camera placement easily loses the intended quality guarantees on visibility coverage of the algorithm. We thus used a more overtly scalable approach, modifying the angle of camera FoV as well as the distance (range) cameras can see: with full 360° FoV and and infinite range we have 100% coverage, whereas we can be certain the level is solvable if the range is sufficiently reduced.

For this experiment we used the Dock level of MGS, as shown in figure 8. The level was filled by 22 cameras, each with identical FoV and range, sweeping back and forth through its entire potential viewing area. Figure 5 shows the result of a parameter sweep considering different combinations of FoV angles and ranges over the level: FoV angle changes from 1 degree to 14 degrees in unit increments, while FoV distance ranges from 1 to 5 with a step of 0.5—the level is approximately 40×60, and a range of about 3 is sufficient to block a corridor. For each combination we performed 20 RRT searches, as reasonable balance between being able to assess success rate and the total duration of these experiments. Note that we do not show the path-distance metric here, as the camera positions do not change.

From this data we can see that the design space allows for reasonable levels of difficulty by modifying either or both of angle and range, albeit within a fairly narrow band of variability. Camera view distance has a stronger impact than angle, which makes sense, as it is easy for a camera with a relatively long range to block an entire passage and make the level unsolvable, even with a very narrow angle of view.

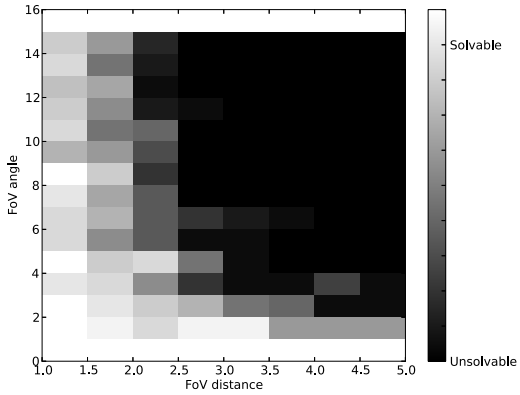

Figure 5: RRT success ratio from 0% to 100% for different camera FoV angles and ranges.

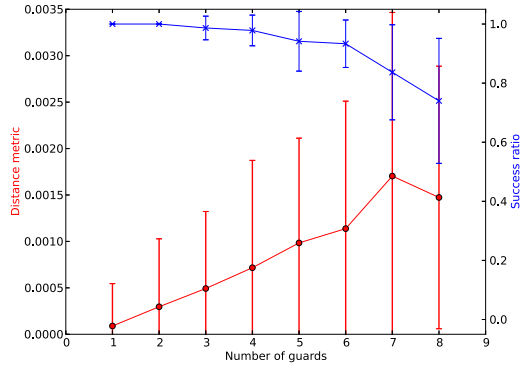

Figure 6: Success ratio (×'s, blue) and distance metric (○'s, red) *vs.* number of guards in the test level.

Guard Analysis

Camera placement offers clear and easy flexibility in difficulty adjustment, but mobile, human-like guards provide more player interest and increase the potential design space. As such, however, the space is too large to consider exhaustively, and so we analyze different features separately.

We begin by exploring the impact of number of guards on level quality. For this we use the test level shown in figure 3. After computing the simplified roadmap we assign guards different, random patrol routes, but without any intra-route activities. Figure 6 shows the results for both the RRT success rate and the path-distance metrics. For each number of guards, we applied 20 RRT searches to each of 30 unique choices of guard paths, giving us an RRT rate /20 and distance metric values on up to 600 situations (we do not include failed solutions in the distance metric).

The randomized path selection results in a large variance, but as expected, increasing the number of guards in an otherwise fixed level increases the relative coverage of space, and so makes it harder to find a path that traverses the level unseen from start to finish. Unlike cameras, however, the cutoff for infeasibility is not nearly as sharp: a moving guard may retard player progress, but by virtue of moving rarely tends to permanently block off an entire corridor, and even with 8 guards (which is quite dense) the RRT is still easily able to find solutions. In this sense the greater slope of the distance metric is perhaps a better representation of game difficulty for players—with more guards, player paths necessarily come closer to guards, and thus closer to being seen.

Grammar Influence

The choice of and amount of application of grammar rules potentially affects the level as well. Experimentally, however, we found that even fairly dense application of the grammar rules does not have a significant impact on RRT success—whether solutions exist is driven primarily by the degree of coverage induced by the number of guards and their paths, and while interrupting a guard path to scan around and so forth is visually interesting, it has limited impact on whether a level is actually solvable or not.

This does not mean, however, that the grammar has no useful effect on solution paths. As guards spend more time scanning around, they temporarily block possible player paths, and/or require players skirt around them. We thus expect that by increasing and changing guard activities we can influence a player's experience in terms of perceived risk in terms of the amount of time a player spends in close proximity to guards, and this is reflected in the distance metric.

Figure 7 shows the distance metric calculations for different kinds and amounts of rewriting, for 4 guards on our test level. We apply our grammar rules to introduce each of the 3 kinds of rotations, the zigzag repetition of a segment (rules 2 and 3 in figure 4), and a random choice of rewrite rule. For each of these kinds of rewriting, we considered rewriting to different recursive depths, where higher depth means more intra-route activities.

The result of this experiment shows a high degree of variance, but nevertheless separates behaviours into 3 main categories with statistical significance ($p < 0.05$). The distance metric value mainly increases due to use of 180° rotations and the zigzag movement, while others have a lesser impact, and all are different from a baseline of no rewriting. We attribute the stronger influence of 180° rotations and zigzag movements to the way these rules alter the ability of a player to follow a guard: if a guard rotates 90° or does a full 360°, it is relatively easy to follow a guard from a distance or get past them as their point of view rotates. With 180° rotations and zigzagging, however, guards end up looking backwards for some time, and a solution path nearby will need to follow the guard movements closely for longer in order to avoid detection.

MGS Comparison

In this experiment we show that our grammar-based approach can achieve a similar distribution of guard behaviours to a manual design in a commercial game. For this we examine the first level of the Metal Gear Solid, as shown in figure 8. This level has just 2 guards, and a fairly stable 100% success rate under RRT search. Of course this would be easy to make more difficult by adding more guards, but this relatively easy level design is also mitigated by varied guard behaviour that provides interest and perceptual challenge to

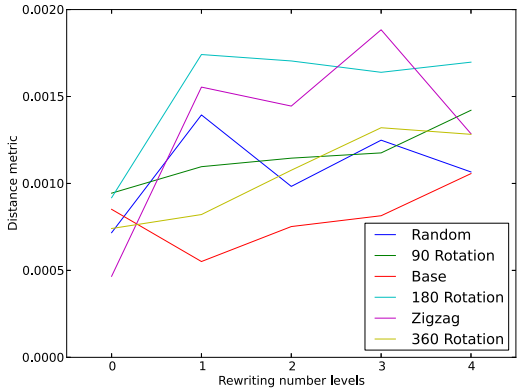

Figure 7: Average distance metric values on the test level with different kinds of rule choice and depth of rewriting.

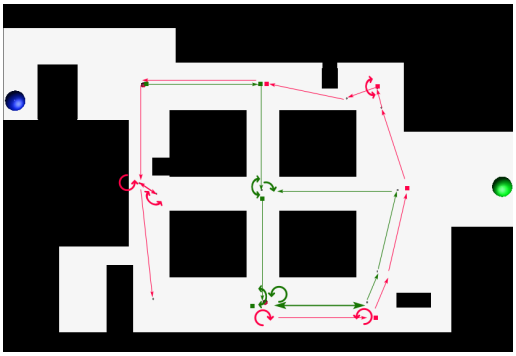

Figure 8: The "dock" level from Metal Gear Solid, with two guards route overlay produced by the grammar.

the player as their first experience in the game.

In figure 9 we analyze the original guard movements and show metric data for the distance metric based on 500 successful paths found by RRT searches. Overlaid with this we show the distributions of the same metric data produced by applying our grammar-based approach to the same basic guard routes. We used a rewrite depth of 2, and biased our rule selection to favour zigzag rewrites over segment splits, and pauses over rotations. This produced the 2 paths presented in figure 8, The red guard patrols the outside of the level, whereas the green guard follows a more complex route within the level. Double-ended arrows represent zigzag behaviour, the rest follows the notation presented in figure 4. Given the randomization inherent in our grammar application and RRT choices the distributions do not match perfectly, but the data-sets do not show a statistically significant difference. This suggests we can algorithmically produce a desired style of guard behaviours, although more extensive experimentation is required to establish the relative impact of the different factors involved.

Conclusion & Future Work

Stealth games or levels are popular with players for posing interesting, puzzle-like challenges different from the typical combat-intensive scenarios found in many modern games.

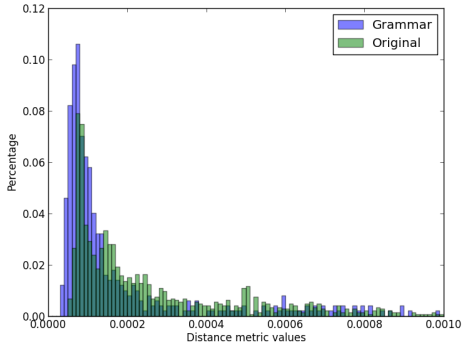

Figure 9: Histogram of distance metric values for MGS level.

A creative puzzle challenge, however, requires a creative puzzle design, ensuring solutions exist, but having appealing and appropriate complexity. The techniques we propose here for procedural generation help with this design challenge by providing distinct mechanisms for scaling difficulty in camera placement, generating guard routes and improving player experience through variety in guard behaviours.

There are many facets of stealth game behaviours we have not covered, and which we intend to explore in future work, such as further, more dynamic properties of enemy behaviours (speed, non-continuous observation), as well as the full gamut of stealth-related game features, such as sound and footprints. We are also interested in extending the analysis further to account for combat scenarios, and to consider gameplay in other game genres.

Acknowledgements

This research was supported by the Fonds de recherche du Québec - Nature et technologies, and the Natural Sciences and Engineering Research Council of Canada.

References

Aichholzer, O.; Aurenhammer, F.; Alberts, D.; and Gärtner, B. 1995. A novel type of skeleton for polygons. *Journal of Universal Computer Science* 1(12):752–761.

Blum, H. 1967. A transformation for extracting new descriptors of shape. In *Models for the perception of speech and visual form*. MIT Press. 362–380.

Cohen-Or, D.; Chrysanthou, Y. L.; Silva, C. T.; and Durand, F. 2003. A survey of visibility for walkthrough applications. *IEEE Transactions on Visualization and Computer Graphics* 9(3):412–431.

Dormans, J. 2010. Adventures in level design: Generating missions and spaces for action adventure games. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, 1:1–1:8.

Erdem, U. M., and Sclaroff, S. 2006. Automated camera layout to satisfy task-specific and floor plan-specific coverage requirements. *Computer Vision and Image Understanding* 103(3):156 – 169.

Ghosh, S. K. 2007. *Visibility Algorithms in the Plane*. Cambridge University Press.

Hoffmann, F.; Kaufmann, M.; and Kriegel, K. 1991. The art gallery theorem for polygons with holes. In *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science*, 39–48.

Kybartas, B., and Verbrugge, C. 2014. Analysis of ReGEN as a graph-rewriting system for quest generation. *IEEE Transactions on Computational Intelligence and AI in Games* 6(2):228–242.

Morgan, S., and Branicky, M. 2004. Sampling-based planning for discrete spaces. In *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 2, 1938–1945.

Obermeyer, K. J.; Ganguli, A.; and Bullo, F. 2011. Multiagent deployment for visibility coverage in polygonal environments with holes. *International Journal of Robust and Nonlinear Control* 21(12):1467–1492.

O'Rourke, J. 1987. *Art Gallery Theorems and Algorithms*. Oxford University Press.

Shi, Y., and Crawfis, R. 2013. Optimal cover placement against static enemy positions. In *Proceedings of the 8th International Conference on Foundations of Digital Games*, 109–116.

Smith, G.; Treanor, M.; Whitehead, J.; and Mateas, M. 2009. Rhythm-based level generation for 2D platformers. In *Proceedings of the 4th International Conference on Foundations of Digital Games*, 175–182.

Smith, R. 2006. Level-building for stealth gameplay. Presentation at the Game Developers Conference. http://www.roningamedeveloper.com/Materials/RandySmith_GDC_2006.ppt.

Togelius, J.; Shaker, N.; and Dormans, J. 2014. Grammars and L-systems with applications to vegetation and levels. In Shaker, N.; Togelius, J.; and Nelson, M. J., eds., *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*. Springer.

Tremblay, J.; Torres, P. A.; and Verbrugge, C. 2014. Measuring risk in stealth games. In *FDG'14: Proceedings of the 9th International Conference on Foundations of Digital Games*.

Xu, Q.; Tremblay, J.; and Verbrugge, C. 2014. Procedural guard placement for stealth games. In *Fifth Workshop on Procedural Content Generation in Games (PCG 2014)*.