The Quest for the Perfect Perfect-Maze

Paul Hyunjin Kim
Department of Computer Science and Engineering
The Ohio State University
Columbus, OH, USA
kim.3983@osu.edu

Roger Crawfis
Department of Computer Science and Engineering
The Ohio State University
Columbus, OH, USA
crawfis.3@osu.edu

***Abstract*— In this paper, the quest for the perfect perfect-maze is performed over the search space of perfect mazes using an approach of search-based procedural content generation. Perfect maze construction is rather random with little to no control of the final product. We propose a search-based framework based on attributes or metrics of a constructed maze to provide a foundation for evaluation functions (fitness functions). Since the meaning of "perfect" is subjective and different for every designer, we allow designers to construct their own evaluation function to generate the best maze. We have also analyzed each metric's space on an exhaustive enumeration of small-sized mazes to determine allowable, and perhaps desirable, ranges for each metric. Using these metrics, an evaluation function is constructed to search for the "best" maze.**

Keywords*— *Procedural content generation; Search-based procedural content generation; Maze evaluation; Game level design.

I. INTRODUCTION

In the field of a procedural content generation, search-based procedural content generation looks for the best content for a game over a search space (defined by the generation parameters) by comparing scores rewarded by an evaluation function. The best content is the content with the highest score. Search-based procedural content generation results in a good content in general, but it is hard to design an evaluation function if our desired property for a content includes the subjective preference such as the "fun" factor of a content. The word "fun" is subjective and has a different meaning for each person. For example, in the case of a maze, some like a maze with many decisions, and others like one with lots of turns. Thus, defining one specific evaluation function for a subjective factor is not a trivial task.

Therefore, we provide guidelines, such as metric information, and a format for an evaluation function that allows designers to easily set up an evaluation function for their own subjective preference. Since our domain for this paper is a perfect maze, explained in section III, we define metrics over a maze that can be manipulated by designers and measure a metrics' space by first order statistics to prevent designers from having unallowable values for metrics. A format in which desired metrics can be embedded in an evaluation function is introduced and a method for constructing an evaluation function with desired values of metrics is provided.

For demonstration, we define desired properties of our best maze with several evaluation functions chosen for a use-case of a 2D pencil & paper maze and a use-case of a 3D FPS maze. We show results that satisfy these desired properties of a maze.

The rest of the paper is organized as follows. In section II, related works are introduced. The structure and components of a perfect maze and algorithms for generating a perfect maze are explained in section III. The format of an evaluation function is provided in section IV. The search space which is the domain of our experiment is explained in section V. In section VI, we survey and describe many metrics on mazes. Specific investigation of the search space is performed in section VII. Then, the results are introduced in section VIII. Finally, conclusion and future work are offered in section IX.

II. RELATED WORKS

Search-based procedural content generation is one approach of procedural content generation. It evaluates each candidate with a designed evaluation function and accepts the content with the highest evaluation. Many search-optimization techniques have been explored, including simulated annealing, Markov chains, and genetic search algorithms. Please refer to [11] for a more detailed definition and survey for search-based procedural content generation.

Our domain for searching is focused on mazes, in particular, what are known as perfect mazes in which there are no loops and no inaccessible areas. In Christopher Berg's website [1], there is an explanation about how to draw a fun maze by hand and key points to consider when making difficult mazes. Also, types of mazes, algorithms for maze construction and mathematical components are analyzed in [2], [7] and [9]. Fortin [9] also performed analysis of human maze solving for randomized mazes and predefined mazes. These works greatly inspired ours. We will define types of mazes and algorithms in Section III.

For maze construction, there has been recent research related to computer-aided maze construction. Xu.J and Kaplan [3] introduced a method for maze construction based upon a vorticity structure. They also concentrated on the aesthetic aspect of the maze construction [4].

978-1-4673-7921-2/15/$31.00 ©2015 IEEE

For an evaluation of a puzzle, the difficulty of SUDOKU was evaluated by using a computational model [5] and a designed difficulty metric [6]. However, there is no research related to the evaluation of a maze with designer-preference. In this paper, we will define metrics on a maze and introduce how to design an evaluation function with defined metrics for searching for the best maze over the search space.

III. PERFECT MAZE

In this section, we will formally define a perfect maze, and its structure and components. Also, algorithms for generating a perfect maze are introduced.

A. Structure of the Perfect Maze

The structure of a perfect maze can be thought of as a spanning tree of an underlying graph, typically a grid graph as shown in Fig. 1 referenced from [10]. A spanning tree of a grid is a connected undirected graph. Every node in a spanning tree is connected without cycles, and it is possible to reach one node from every other node. In a perfect maze, like a spanning tree, there are no unconnected areas. As such, the start and end points can be placed anywhere in a perfect maze with a guaranteed unique solution. There is a unique path from any node to any other node in the spanning tree. A perfect maze is those comprised of a spanning tree, a start location and an end location. Thus, for a single spanning tree there are numerous possible perfect mazes.

B. Components of a Perfect Maze

We define the components of a perfect maze as follows:

- ***Cell***: Each node on a maze is called a cell. For example, if the maze is defined on a 2D 8x8 grid, there are 64 cells in the maze.
- ***Start Point***: The start point is the cell where the player enters the maze. This is typically a boundary cell, but does not need to be. Note, if the cell is a corner, it does not matter which "wall" is the entrance.
- ***End Point***: The end point is the cell where the player finishes or exits the maze. This is typically a boundary cell, but does not need to be so. Note, if the cell is a corner it does not matter which "wall" is the exit.
- ***Solution Path***: The solution path is the unique passage which connects the start point to the end point. The goal of a maze is to find the solution path.
- ***Decision cell***: A cell in which there are more than two passages connected to it. There are two types of decision cells: junctions and cross roads.
    - ***Junction***: If there is a cell with three passages, we call it a junction. The player can enter from one cell and has to choose which of two other cells to travel into.
    - ***Cross Road***: A cross road is a cell connected with all four neighboring cells. The player can enter from one cell and has to choose which of three other cells to travel into.

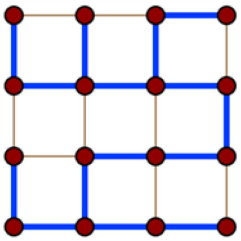

Fig. 1. Spanning tree in grid graph [10].

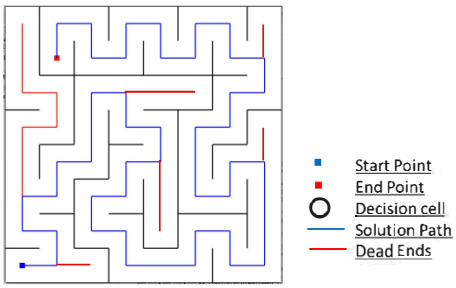

Fig. 2. Compoenets of Perfect Maze.

- ***Dead End***: A dead-end is a rooted tree with a set of passages having no exit. The root of a dead-end is a decision cell on the solution path. We define three types of dead-ends. A perfect maze may have several dead-ends.
    - ***Alcove***: This type of dead-end has no turns or decisions. It has a straight short path.
    - ***Forward Dead-End***: One of the contributions in this paper, this dead-end provides the illusion it is progressing towards the exit. It will be defined more clearly later.
    - ***Backward Dead-End***: This dead-end turns away from the solution path and away from the end point.

C. Forward dead-end

In this paper, we use directional property when we define forward dead-end. In a 2D pencil and paper maze, the location of the end point is shown for a player. If some dead-end looks like either forwarding to the end point or going straight from the solution path, we call it a forward dead-end. A backward dead-end have opposite directional property such as back-warding from the end point.

However, in a 3D FPS maze, since the location of the end point is unknown due to occluded vision, it is hard to know whether a dead-end is moving forward to the exit or not. We define a forward dead-end as a dead-end which has forward direction from current location and a backward dead-end as a dead-end which is neither a forward dead-end nor an alcove in a case of 3D FPS.

D. Algorithms for a Perfect Maze

There are several algorithms to generate a perfect maze as described in [2] and [9]. Each algorithm generates a maze with different properties in a statistical sense. The randomized Prim's algorithm has a tendency to produce a maze with many short dead-ends. The randomized recursive backtracker

algorithm creates mazes with long dead-ends, and little branching. There is virtually no control over the construction of the maze for most of these. A single random number is the only parameter used in the generation. Current workflow for a designer is to either build the maze by hand or repeatedly ask the generation algorithm to create a new maze until one with satisfactory characteristics is created.

IV. Defining the "Best"

In this paper, we are trying to provide a tool for a designer to find the best perfect maze, however it is a difficult task. What characteristics make one maze better than another? Can we measure these characteristics? If so, what is the best we can do? In Section VI, we provide measureable attributes that can be defined on a given maze.

Our approach is to define an evaluation function as equation (1), which we use in searching for a desired maze over the search space.

$$D = \sum_{i=1}^{n} \left(\frac{w_i}{W} * b_i \right) \quad (1)$$

In equation (1), $b_i = \left(1 - \frac{|a_i - t_i|}{\max(a_i, t_i)}\right)$, $W = \sum_{i=1}^{n} w_i$, a_i is a metric value and t_i is a target or desired value of a metric as specified by the designer. As a_i becomes closer to t_i, b_i increases in value. Since W is the summation of w_i, equation (1) is a weighted average of metrics. Our search is thus for the maze which maximizes value D in our evaluation function. Given desired measurable properties of a maze, we do not know each metric's space. A designer should not be allowed to set desired values of a metric larger than the maximum value of a metric. Even if we know a metric's space, how many samples are needed to generate a desired maze? As we said in section III.*C*, different generation algorithms create mazes with different properties.

V. The Search Space

Our algorithm proceeds by repeatedly generating a spanning tree (a sample). For each sample in the search space, we generate a spanning tree in a given grid as the structure of a maze and set the start point and the end point on any nodes of the spanning tree as shown in Fig. 3. From a given spanning tree we can quickly generate all possible perfect mazes (there are n choose 2 possible start/end pairs, where n is the number of cells). The designer can restrict the start to the left edge and the end to the right edge or specific locations.

In order to have all possible mazes within a fixed grid structure, we need to enumerate all possible spanning trees. Theoretically, there are $e^{1.37N^2}$ possible spanning trees in an *N*x*N* grid. Enumerating this is a hard problem and a subject of further research. Indeed, even intelligently sampling the space is a difficult undertaking as the algorithms are only controlled by a random seed, several of which may produce the same spanning tree.

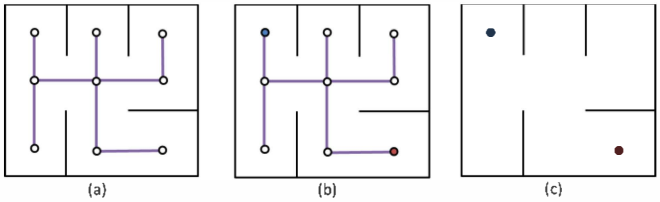

Fig. 3. (a) The structure of the spanning tree (purple line) with nodes (white circle) in 3x3 grids. (b) Set the start point (blue circle) and the end point (red circle) on the structure. (c) The generated sample, a perfect maze, with the start point and the end point.

VI. Metrics

In this section, we define measurable metrics on the spanning tree, the solution path and the set of dead-ends. Then, association between metrics will be investigated.

A. Definitions of Metrics

1) Metrics per spanning tree

- *CE* = Number of cells.
- *E* = Number of edges.
- *J* = Percentage of junctions
- *CR* = Percentage of crossroads

For an *N*x*N* grid structure, *CE* has a value of N^2, and *E* has a value of N^2-1 as a spanning tree is connected and has no cycles. *J* is a percentage of a number of junctions divided by *CE*. Similarly, *CR* is the percentage of a number of crossroads divided by *CE*. These metrics can be computed once per spanning tree.

2) Metrics per solution path given start and end

- *DE*= Percentage of dead-ends.
- *AC* =Percentage of alcoves.
- *FDE* = Percentage of forward dead-ends.
- *BDE* = Percentage of backward dead-ends.
- T_s = Percentage of turns on the solution path.
- J_s = Percentage of junctions on the solution path.
- CR_s = Percentage of cross roads on the solution path.
- D_s = Percentage of decisions on the solution path.
- L_s = Relative length of the solution path.

These metrics need to be computed for each (start/end) pair for each spanning tree. Metrics 1-8 are divided by the length of the solution path which is the number of cells on the solution path. Thus, each metric value is a percentage over the solution path. *Ls* is percentage of length of the solution path divided by *CE*. The quantity, *DE=AC+FDE+BDE*, indicates the relative number of dead-ends on the solution path. For each of these the following metrics are defined.

3) Metrics per each dead-end

- T_d, T_{fd}, T_{bd} = Percentage of turns.
- J_d, J_{fd}, J_{bd} = Percentage of junctions.

- CR_d, CR_{fd}, CR_{bd} = Percentage of cross roads.
- D_d, D_{fd}, D_{bd} = Percentage of decisions.
- L_d, L_{fd}, L_{bd}, L_a = percentage of size of a dead-end

(d: dead-end, fd: forward dead-end, bd: backward dead-end, a: alcove)

Like the metrics in section VI.*A.2*, each metric except L_* is a percentage divided by corresponding size of the dead-end. Each L_* is a relative size of the corresponding dead-end (i.e., it is divided by *CE*).

B. Association between Metrics

We describe a relationship between metrics defined in section VI.*A* over a maze.

- $L_s = 1 - \sum L_d$
- $DE = AC + FDE + BDE$
- $D_{\#} = CR_{\#} + J_{\#}$ $\quad where\ \# \in \{s, d, fd, bd, a\}$
- $\sum D_d = \sum D_{fd} + \sum D_{bd}$
- $\sum T_d = \sum T_{fd} + \sum T_{bd}$
- $\sum L_d = \sum L_{fd} + \sum L_{bd}$
- $D_s \leq DE$
- $DE = J_s + 2 * CR_s$

$\sum Ld$ means the total size of all dead-ends. Since there are relations between metrics, care is needed when specifying desired properties. For example, if there is solution path with *Ls* = 50%, we cannot desire 60% as the size of all dead-ends on a maze because it violates the relation $L_s = 1 - \sum L_d$.

VII. Analysis of Metrics

A. Sampling the Space

For sampling the space, we have several algorithms to generate a perfect maze as a sample as described in [2] and [9]. Since each algorithm creates a perfect maze with different properties, we generate mazes from either the randomized recursive backtracker or the randomized Prim's algorithm and gather those samples in one search space so that the search space have various samples with different properties.

Since we use a random seed as a parameter for a generator of a sample, the generation process has low controllability, and it is hard to guarantee that there are no duplicates in our samples. Thus, we generate a huge amount of samples to lower the probability of having duplicates on a search space.

10,240,000 spanning trees are generated on an 8x8 grid by each maze generation algorithm with random seeds. We restrict the start point to anywhere on the left edge and the end point to the right edge, giving eight choices for each location for a given spanning tree. Hence, 64 perfect mazes are created for each spanning tree. A total 1,310,720,000 mazes were generated and used as samples in the subsequent analysis.

B. Analysis of Metrics based on Our Samples

We analyze each metric based on our generated samples except metrics *CE* and *E* in section VI.*A.1*. Because both *CE* and *E* are constants for given grid structure, analysis is not needed for these metrics. Each metric's space is represented by a minimum value, a maximum value, an average value and a standard deviation in table I. A designer can choose a metric and set desired value for it within the allowable range as described in table I.

TABLE I. METRIC'S SPACE BY *MAX*, *MIN*, *AVG* AND *STDEV*.

	Max	***Min***	***Avg***	***Stdev***
J	40.63%	0%	15.94%	7.12%
CR	17.19%	0%	2.55%	2.97%
DE	76.67%	0%	18.99%	8.25%
AC	76.67%	0%	19.86%	16.37%
FDE	13.79%	0%	1.62%	1.3%
BDE	65.38%	0%	13.08%	10.85%
T_s	81.69%	0%	52.98%	10.68%
J_s	75.86%	0%	17.92%	15.24%
CR_s	8%	0%	0.22%	0.51%
D_s	76.67%	0%	19.86%	16.37%
L_s	70.5%	12.5%	17.89%	11.47%
T_d	86.67%	0%	18.92%	20.07%
J_d	43.75%	0%	6.63%	9.79%
CR_d	6.25%	0%	0.22%	0. 64%
D_d	43.75%	0%	6.63%	9.79%
L_d	93.75%	0%	3.86%	8.49%
T_{fd}	86.67%	0%	41.48%	15.87%
J_{fd}	37.5%	0%	10.87%	8.9%
CR_{fd}	5.36%	0%	0.38%	0.7%
D_{fd}	38.1%	0%	12.4%	10.22%
L_{fd}	92.5%	0%	9.34%	12.23%
T_{bd}	86.67%	0%	35.14%	13.23%
J_{bd}	43.75%	0%	11.73%	10.42%
CR_{bd}	0.625%	0%	0.39%	0.8%
D_{bd}	43.75%	0%	13.3%	11.52%
L_{bd}	93.75%	0%	6.41%	10.51%
L_a	2.75%	0%	0.56%	0.147%

If the *Avg* value is close to the *Min* value with a small *Stdev* for some metric, it means that we have a low probability of finding a maze where the value of the metric is close to the maximum. Minimum value for every metric is 0% except *Ls* which is 12.5%. Minimum value of *Ls* depends on the width of the grid which in this case is $\frac{1}{\sqrt{C}}$. Most metrics have low *Avg* values close to *Min* with a small *Stdev* value.

These statistics are over our one billion mazes, not necessarily the entire set of mazes. There is a small set of mazes with maximum properties of specific metrics. A Hilbert curve would have a maximum number of turns with no decisions; hence its solution path percentage would be one. Likewise, a Caterpillar graph would have a maximum number of decisions on a solution path, where each dead-end had a

length of one cell. Thus, its length of a solution path equals to number of alcove with length one.

VIII. RESULTS

In this section, we demonstrate for designers how to design an evaluation function. We define our desired best perfect maze for a 2D pencil & paper maze and a 3D first person-shooter (FPS), choose metrics and decide a desired target value and weight value for each metric to embed our desired properties of a maze into an evaluation function. Then, an evaluation function is constructed based on metrics' information. For the search space, 22,400,000 mazes of 20x20 grids are generated. Mazes with the worst, median and the best evaluation are selected over the search space using the designed evaluation function. Searching for the best maze takes 3,063,132 seconds with the machine of 3.40 GHz Inter Core i7-4770 processor. The maze with the highest score is investigated to see if it has desired properties compared to other mazes with middle and the lowest score.

A. 2D maze

For desired properties of the best maze in a field of 2D maze, we would like to have the solution path on half of a maze and forward dead-ends on the rest. On the solution path, we want to have turns on 40% of the solution path and decisions on 2% of the solution path. For forward dead ends, in order to making a player not to escape from each forward dead-end easily, it should be complicated with 40% turns and 1% decisions on each forward dead-end on average.

We choose metrics from table I and set target values t_i and weight values w_i for each metric as shown in table II. High w_i means that we would like to force corresponding desirable property on a maze more than other properties. Since we want total size of forward dead-ends over a maze to be half a maze, $\sum L_{fd}$ is specified. Based on above desired properties, we design an evaluation function for our best maze as shown in equation (2). b_1, b_2 and b_3 are related to metrics of the solution path (L_s, T_s and D_s), and b_4, b_5 and b_6 are related to metrics of forward dead-ends (L_{fd}, T_{fd} and D_{fd}). In b_5 and b_6, we have average value of metrics over forward dead-ends in μ.

$$
\begin{aligned}
D &= \sum_{i=1}^{6}\left(\frac{w_i}{W} * b_i\right) \\
b_1 &= \left(1 - \frac{|L_s - t_1|}{\max(L_s, t_1)}\right) \\
b_2 &= \left(1 - \frac{|T_s - t_2|}{\max(T_s, t_2)}\right) \\
b_3 &= \left(1 - \frac{|D_s - t_3|}{\max(a_i, t_3)}\right) \\
b_4 &= \left(1 - \frac{|\sum L_{fd} - t_4|}{\max(\sum L_{fd}, t_4)}\right) \\
b_5 &= \left(1 - \frac{|\mu - t_5|}{\max(\mu, t_5)}\right), \ where\ \mu = \frac{\sum_{m=1}^{FDE}(T_{fd})_m}{FDE} \\
b_6 &= \left(1 - \frac{|\mu - t_6|}{\max(\mu, t_6)}\right), \ where\ \mu = \frac{\sum_{m=1}^{FDE}(D_{fd})_m}{FDE}
\end{aligned}
\tag{2}
$$

TABLE II. DESIRED PROPERTIES FOR 2D MAZE.

	t_i = target value	$\frac{wi}{W}$ = weight value
L_s	50%	0.22
T_s	40%	0.17
D_s	2%	0.11
$\sum L_{fd}$	50%	0.17
$Avg(T_{fd})$	40%	0.22
$Avg(D_{fd})$	1%	0.11

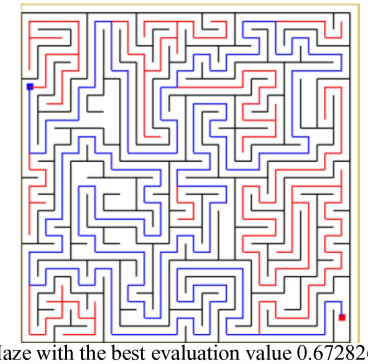

Fig. 5. Maze with the best evaluation value 0.672826.

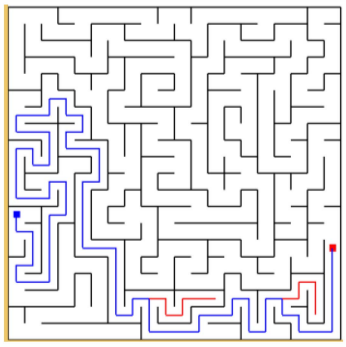

Fig. 4. Maze with median evaluation value 0.347566.

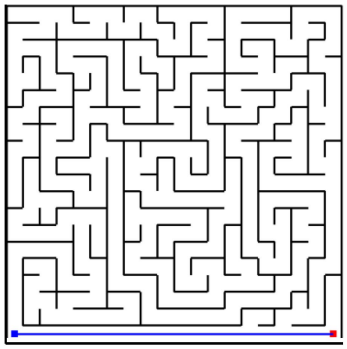

Fig. 6. Maze with the lowest evaluation value 0.022.

In Figs 4, 5 and 6, we can see mazes with the best, median and the worst evaluation respectively. The blue line indicates the solution path, and the red line indicates forward dead-ends. Also, tables III, IV and V shows how well the resulting mazes satisfy our desired properties numerically by showing the error measured by difference between the actual value and the target value for each metric. The maze with the best score has lower error values for most of the metrics compared to the other mazes' one. However, D_{fd} in table III has the five times bigger

value than the desired value of D_{fd} in table II. Because our search space does not include all possible mazes, it is hard to get a maze that satisfies every desired property. For our desired properties, the size of the solution path as a half portion of the maze and the size of forward dead ends as the rest was wanted. As you can see in Fig. 4, this desired property is visually satisfied on the maze with the best evaluation.

TABLE III. ACTUAL VALUES FOR DESIRED PROPERTIES FOR 2D MAZE IN FIG.4.

	Actual value	Error
L_s	51.5%	1.5%
T_s	60.2%	20.2%
D_s	3.9%	1.9%
$\sum L_{fd}$	43%	7%
$Avg(T_{fd})$	49.7%	9.7%
$Avg(D_{fd})$	6.5%	5.5%

TABLE IV. ACTUAL VALUES FOR DESIRED PROPERTIES FOR 2D MAZE IN FIG.5.

	Actual value	Error
L_s	21%	29%
T_s	59.5%	19.5%
D_s	2.38%	0.38%
$\sum L_{fd}$	3.25%	46.75%
$Avg(T_{fd})$	53.57%	13.57%
$Avg(D_{fd})$	0%	1%

TABLE V. ACTUAL VALUES FOR DESIRED PROPERTIES FOR 2D MAZE IN FIG.6.

	Actual value	Error
L_s	5%	45%
T_s	0%	40%
D_s	0%	2%
$\sum L_{fd}$	0%	50%
$Avg(T_{fd})$	0%	40%
$Avg(D_{fd})$	0%	1%

B. 3D First Person-Shooter (FPS)

In a 3D FPS maze, we do not know where the end point is from current location because of occluded vision. Thus, when we search the best maze in a field of a 3D FPS, we define a forward dead-end as a dead-end which is going forward from current location and a backward dead-end as a dead-end which is neither a forward dead-end nor an alcove. In Fig. 6, if a passage is a dead-end and not an alcove, a passage with forward direction will be a forward dead-end, and a passage with non-forward direction will be backward dead-end.

Here, we define desired properties of our best maze in a 3D FPS. For the solution path, we would like 60% as a length of the solution path. For dead ends, if a maze has only alcoves as dead-ends it will be very easy and not challenging. Thus, it is desirable for a size of dead-ends except alcoves to have 10% over a maze. Since we don't want a complicated solution path, we hope to have 30% turns and 1% decisions including junctions and crossroads. For each dead-end, 10% turns and 10% decisions desired.

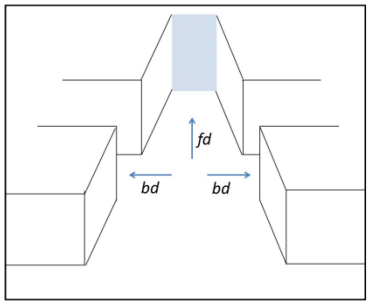

Fig. 7. fd: forward dead-end and bd: backward dead-end in 3D FPS maze.

Chosen metrics with a desired t_i and w_i is represented in table VI. Since we set desired metric value of size of dead-ends without alcoves, we calculate a value related to size of dead-ends as $\sum L_d$-$\sum L_a$ in table VI. An evaluation function for our best 3D FPS maze is represented in equation (3). It is similar to equation (2) except b_4, b_5 and b_6. In b_5 and b_6, we have average value of metrics over dead-ends except alcoves in μ.

$$
\begin{aligned}
D &= \sum_{i=1}^{6}\left(\frac{w_i}{W} * b_i\right) \\
b_1 &= \left(1 - \frac{|L_s - t_1|}{\max(L_s, t_1)}\right) \\
b_2 &= \left(1 - \frac{|T_s - t_2|}{\max(T_s, t_2)}\right) \\
b_3 &= \left(1 - \frac{|D_s - t_3|}{\max(a_i, t_3)}\right) \\
b_4 &= \left(1 - \frac{|\sum L_d - \sum L_a - t_4|}{\max(\sum L_d - \sum L_a, t_4)}\right) \\
b_5 &= \left(1 - \frac{|\mu - t_5|}{\max(\mu, t_5)}\right),\ where\ \mu = \frac{\sum_{m=1}^{DE-AC}(T_d)_m}{DE-AC} \\
b_6 &= \left(1 - \frac{|\mu - t_6|}{\max(\mu, t_6)}\right),\ where\ \mu = \frac{\sum_{m=1}^{DE-AC}(D_d)_m}{DE-AC}
\end{aligned}
\tag{3}
$$

TABLE VI. DESIRED PROPERTIES FOR 3D FPS.

	t_i = target value	$\frac{wi}{W}$ = weight value
L_s	60%	0.22
T_s	30%	0.16
D_s	1%	0.1
$\sum L_d$-$\sum L_a$	10%	0.22
$Avg(T_{fd})$	10%	0.14
$Avg(D_{fd})$	10%	0.16

The solution path is represented by the blue line, and dead-ends without alcoves are represented by the red line. In Fig. 8, we show the best maze with the best evaluation for 3D FPS which satisfies properties defined by us. Both mazes in figures 8 and 9 represents desired properties of the solution path well, but a maze in Fig. 9 does not represent the desired characteristic for dead-ends. In this maze, the total size of dead ends is much bigger than desired value 10% with the error 37%. In Fig. 10, no turns and decisions are on the solution path which means that desired properties are not satisfied at all. Overall low error values in table VII compared to error values in other tables also tells us that the maze in Fig. 8 is the best maze for us.

IX. CONCLUSION AND FUTURE WORKS

In this paper, we have provided an analysis of a perfect maze including metric's space and format of evaluation function to allow designers to generate a desired maze using their designed evaluation function.

There are no all possible mazes in our search space for now. If we enumerate all possible samples for a search space, we could generate all desired maze. However, because

978-1-4673-7921-2/15/$31.00 ©2015 IEEE

enumeration of mazes takes very long time, it would be hard to make the search space to consist with all possible mazes. Thus, we are going to explore the number of samples that are needed and which maze generation algorithm is particularly good to generate the specific desired maze.

Also, we have concentrated on the domain of perfect mazes; however, there are other kinds of mazes, like braid mazes. Because different kinds of mazes can have more than one solution path and cycles, more metrics need to be explored and added to the equation for an evaluation of a maze.

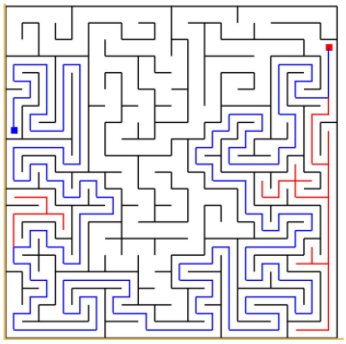

Fig. 8. Maze with the best evaluation value 0.47122.

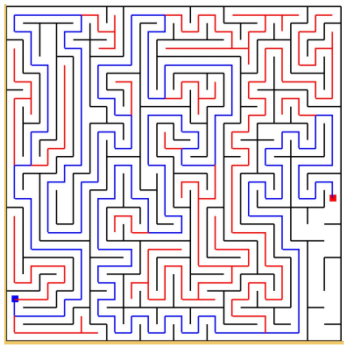

Fig. 9. Maze with the best evaluation value 0.24094.

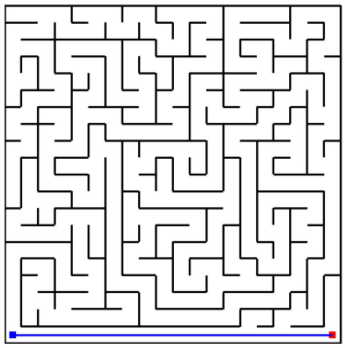

Fig. 10. Maze with the best evaluation value 0.010135.

TABLE VII. ACTUAL VALUES FOR DESIRED PROPERTIES FOR 3D FPS IN FIG. 8.

	Actual value	Error
L_s	45.75%	14.25%
T_s	65.57%	35.57%
D_s	1.09%	0.09%
$\sum L_d$-$\sum L_a$	10.75%	0.75%
$Avg(T_{fd})$	31.4%	21.4%
$Avg(D_{fd})$	11.9%	1.9%

TABLE VIII. ACTUAL VALUES FOR DESIRED PROPERTIES FOR 3D FPS MAZE IN FIG. 9.

	Actual value	Error
L_s	49.5%	10.5%
T_s	59.6%	29.6%
D_s	6.57%	5.57%
$\sum L_d$-$\sum L_a$	47%	37%
$Avg(T_{fd})$	44.69%	34.69%
$Avg(D_{fd})$	6.49%	3.51%

TABLE IX. ACTUAL VALUES FOR DESIRED PROPERTIES FOR 3D FPS MAZE IN FIG. 10.

	Actual value	Error
L_s	5%	55%
T_s	0%	30%
D_s	0%	1%
$\sum L_d$-$\sum L_a$	0%	10%
$Avg(T_{fd})$	0%	10%
$Avg(D_{fd})$	0%	10%

ACKNOWLEDGEMENTS

The author would like to thank the Evan DeLaubenfels, David Maung and Chloe Shi for their helpful suggestions on the topic.

REFERENCES

[1] BERG, C., 2005. Amazeing art. http://www.amazeingart.com.

[2] Pullen, W. D., 2005. Think labyrinth. http://www.astrolog.org/labyrnth/maze.htm

[3] Xu, J., AND KAPLAN, C. S. 2007. Vortex maze construction. Journal of MAthematics and the Arts 1, 1 (march), 7-20.

[4] Xu, J., AND KAPLAN, C. S. 2007. Image-guided maze construction. ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2007. Volume 26 Issue3, July 2007 Article No. 29.

[5] Pelanek, R. ´ 2011. Difficulty rating of sudoku puzzles by a computational model. In Proc. of Florida Artificial Intelligence Research Society Conference (FLAIRS 2011).

[6] Leone, A.; Mills, D.; and Vaswani, P. 2008. Sudoku: Bagging a difficulty metric and building up puzzles. Mathematical Contest in Modeling, University of Washington.

[7] Maze Works Website. http://www.mazeworks.com/mazegen/mazetut/

[8] Procedural Content Generation in Games – A textbook and an overview of current research. http://pcgbook.com

[9] Martin Foltin. 2011. Automated Maze Generation and Human Interaction [Diploma Thesis]. Masaryk University Faculty of Informatics.

[10] Wikipedia. Spanning tree. http://en.wikipedia.org/wiki/Spanning_tree

[11] Togelius, J., Yannakakis, G., Stanley, K. and Browne, C. 2011. Search-Based Procedural Content Generation: A Taxonomy and Survey. IEEE Transactions on Computational Intelligence and AI in Games, Vol. 3, No. 3, September 2011.

DOI=10.1145/1814256.1814257
http://doi.acm.org/10.1145/1814256.1814257.

[13] Dormans. J. Level design as model transformation: A strategy for automated content generation. In *Proceedings of the Procedural Content Generation Workshop 2011*, Bordeaux, France, 2011.

[14] Silva, P., Muller, P., Bidarra, R., and Coelho, A. (2013). Node-based shape grammar representation and editing. In Workshop Proceedings of the 8th International Conference on the Foundations of Digital Games. http://www.fdg2013.org/program/workshops/papers.html.