Vol.2021-CG-184 No.27
Vol.2021-DCC-29 No.27
Vol.2021-CVIM-227 No.27
2021/11/6

平面充填に基づくドア迷路自動生成アルゴリズム

郭　清蓮　　茶谷卓実

Abstract: A maze is a game to start from one location to reach another by passing through an intricate path. In 2001, a door-maze game was proposed in which multiple rooms connected by doors. Each room has a certain amount of space and is surrounded by walls or doors. Such door-maze is expected useful for automatically generating the environment of exploration games.　However, conventional maze generation algorithms cannot automatically generate door-mazes. Therefore, we started our research to aim at automatic generation of door-mazes. Our originally developed algorithm has two remarkable features: one is the algorithms for room division based on plane filling, and another is a door placement algorithm to insert doors and form maze paths. With our original algorithms, a variety of door mazes can be automatically generated. In this paper will describe our research in detail.

Keywords: Maze, Door maze, Algorithms, Room division, Maze path, Polyhedron tiling

1. はじめに

迷路とは複雑に入り組んだ道を通り抜けて目的地を目指すゲームであり、スタートからゴールまで基本的に一本道であり、それ以外は袋小路である。路には自動生成のアルゴリズムが複数存在し、代表的なものとして棒倒し法、穴掘り法、壁伸ばし法、クラスカル法[5, 8, 11]などが挙げられる。

2001 年に嘉来氏より複数の空間をドアでつなぐ「ドア迷路」が提案された[6]。絵画的な迷路[4, 7]が魅力的になっている時代だったため、当時ドア迷路がそれほど注目されなかった。図 1-(b)が示すように、「ドア迷路」の特徴といえば、ゲーム空間がある程度の大きさを持つ「部屋」で充填される。まず、「部屋」の間に「壁」と「ドア」がある。また、「壁」が薄く面積がないため、迷路全体の面積=全ての部屋の面積の総和である。更に、各部屋には 1 つ以上の「ドア」がある。ドア迷路のメリットといえば、各部屋にある程度のスペースがあるため、アイテムやオブジェクトを配置することが可能である。また、ドアを開けない限りは隣の部屋の状況が予想できないことから、ドア迷路を 3 次元的に展開すれば、探検ゲームに必要とされる環境が自動的に生成できる。

既存なゲームにはドア迷路に近いものが多くある。例えば、2001 年に任天堂が発売した「ルイージマンション」が挙げられる[9]。行方不明になったマリオを探すために、ルイージがオバケ掃除機「オバキューム」を使いながらオバケの出る入り組んだ屋敷を探索する。各部屋内にオブジェクトが置かれている、様々な仕掛けが用意されてある。自動生成できるようになれば、難易度の異なる部屋配置が簡単に作れる。また、2019 年に城間かずき氏が提案した「シャドウコリドー」[10]にも、同様な回廊と部屋が入り組んだゲーム環境がある。同じ難易度でもプレイヤーを飽きさせない複数の回廊と部屋の配置がある。

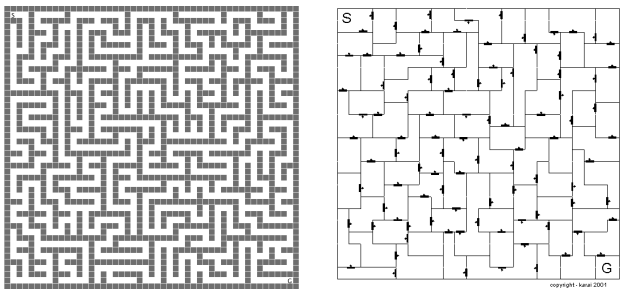


図 1. 通常の迷路(a)と嘉来氏の「ドア迷路」[6](b)

しかし、現実問題として、従来の迷路生成アルゴリズムではドア迷路を自動的に生成できない。棒倒し法、穴掘り法、壁伸ばし法、クラスタリング法ではドア迷路の様々な大きさの部屋に対応できない。嘉来氏が 1 枚の設計図だけ提案したが[6]、コンピュータによる自動生成アルゴリズムまで考えなかった。「ドア迷路」の特徴から分かるように、難易度が制御できる、かつ同じ難易度でも毎回異なる部屋とドアの配置ができることが、自動生成アルゴリズムに期待される結果である。

そこで、私たちの目標はドア迷路の自動生成アルゴリズムに設定された[2]。ドア迷路の自動生成は、平面充填に基づく「部屋分けのアルゴリズム」と部屋の間に迷路が形成させる「ドア設置アルゴリズム」の 2 ステップで実現できた。まず、部屋分けアルゴリズムは、長方形あるいは正多角形の部屋で平面を充填する。それから、設置アルゴリズムは、従来のクラスカル法[5, 8, 11]を拡張し、複雑な隣接状況において

Vol.2021-CG-184 No.27
Vol.2021-DCC-29 No.27
Vol.2021-CVIM-227 No.27
2021/11/6

も、部屋間にドアをランダムに開け、最終的にスタートから目的地までの一本道を作る。この 2 つの独自のアルゴリズムによって、迷路に必要とされる複雑性と汎用性のある結果が得られた。本論文は我々の研究について述べる。

2. 部屋分割アルゴリズム

まず長方形による部屋分割アルゴリズムについて説明する。ここでは、迷路空間を大きさの異なる長方形の「部屋」で隙間なく充填される。また、長方形の大きさと形にランダム性を持たせながら部屋を作ることが必要である。図 2 は、独自に開発された長方形部屋分けのアルゴリズムを示す。具体的に以下の処理で実現される。

① 四角形の親領域においてランダムに、**N** 個の点を配置する(赤い点)。
② 親領域を互いに垂直する 3 本の直線で 4 つの子領域に分割する。子領域も四角形の形になるが、それぞれの大きさは異なる。3 本の直線の位置はランダムに決まるため、毎回子領域分割の結果が変化する。
③ 各子領域に対して、そこにある赤い点を数え、それから細分割するかどうか決める。もし、赤い点の数が一定値 m 個より多いなら、この子領域を親領域と設定し直し、②から再帰的に分割を繰り返す；もし、赤い点の数が m 個より少ないなら、この子領域を細分割しない、一つの部屋として設定する。
④ 全ての子領域が部屋になると、処理を停止する。

また、パラメータ N と m を変えることによって、部屋分割の密度を変えることができる。図 2 はそれぞれ N=500、m=5 と N=250、m=20 の場合の部屋分け例である。N=500、m=20 のケースに比べ、部屋分割の密度が違うことが分かる。これによって、迷路の複雑さをコントロールできる。

部屋分割の過程は、四分木構造によってコンピュータに記述される。図 3 で示すように同じ色の直線(黄色、緑、水色)がそれぞれ異なる木構造の階層を表す。各部屋は四分木の葉ノードになる。また、分割完了した後、四分木に従い、順に各部屋にインデックス番号が割り振られる。

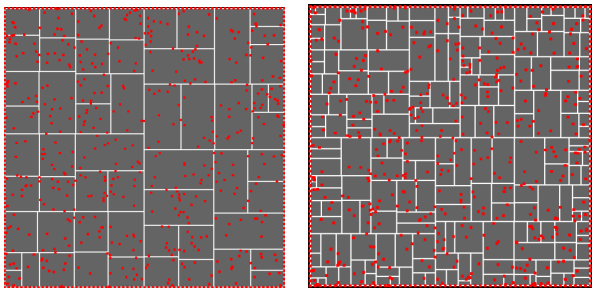


図 2　長方形による部屋分割アルゴリズム

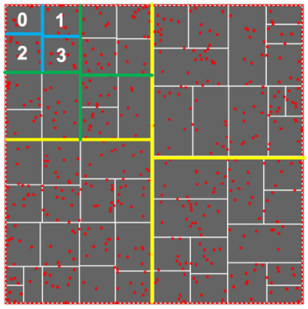


図 3　4 分木で記述する部屋分割と部屋のインデックス番号の振り方

3. ドア設置アルゴリズム

まず、部屋に関連する情報を保存するクラス *Room* を定義する。*Room* クラスには部屋の情報を以下の 6 個の変数で表し、設置アルゴリズムの実施に利用される。

- 現在の部屋のインデックス番号：in_num
- 現在の部屋が属すクラスタの番号：clu_num
- 部屋の 4 頂点座標：v[]
- 部屋の中心点の座標：c
- 隣人リスト（隣接する部屋のインデックス番号のリスト）：nei_i
- 通路リスト（ドアで接続された部屋のインデックス番号のリスト）：door_i

図 4 は、ドア設置前と設置後、部屋クラス *Room* の各変数値の変化を示す。紫色の文字に注目してほしいが、最初、clu_num と in_num と同じ値であるが、ドア設置されたことによって、最終的に全て最も小さいクラスタ番号 clu_num＝0 に変わる。また、部屋の door_i にはドアが設置された隣部屋のインテックス番号 in_num が格納される、最後にクラスタリングによって、ドアを通して移動できる全ての部屋のインデックス番号が door_i リストに保存される。

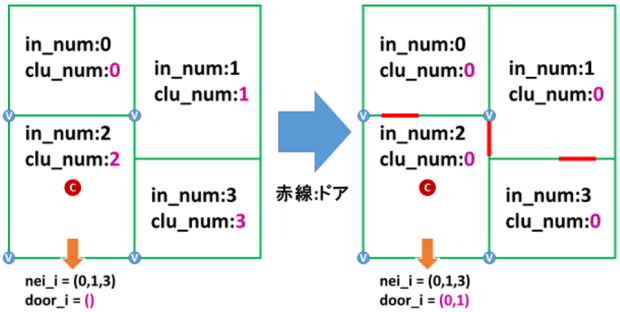


図 4　クラス Room に保存する部屋に関する情報の 1 例

これから、ドア設置アルゴリズムをプロセス順に説明する。まず、各部屋にクラスタ番号を設定する。図 5 には、黒

Vol.2021-CG-184 No.27
Vol.2021-DCC-29 No.27
Vol.2021-CVIM-227 No.27
2021/11/6

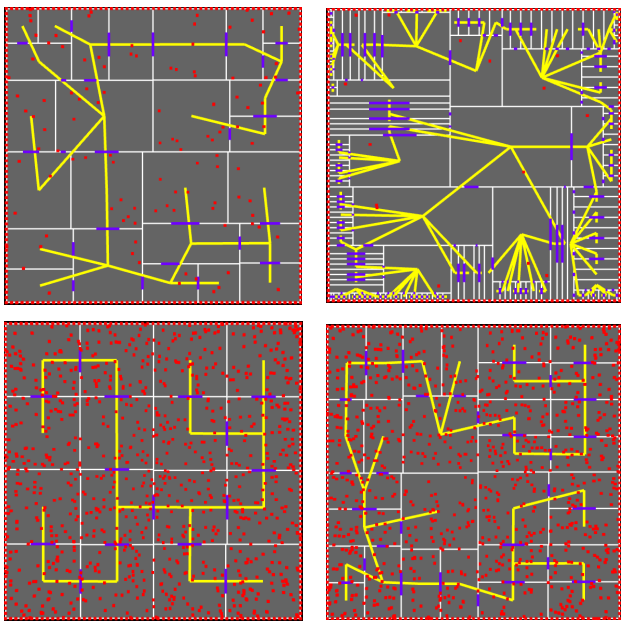
い数字がインデックス番号、赤い数字がクラスタ番号を示す。クラスタ番号の初期値は部屋のインデックス番号と同じ。

次に、各部屋に対して選択可能な隣人リスト choosable_list を作る。選択可能な隣人とは、隣接するが異なるクラスタ番号を持つ隣人のことである。例えば、部屋 10 の場合、選択可能な隣人リストは (3, 5, 6, 8, 11, 12)である。プロセスが進むと同時に、各部屋の隣人リストが短くなり、選択可能な隣人が少なくなる。

更に、ランダムに一つの部屋 A を選ぶ。部屋 A の choosable_list から 1 つ隣接する部屋 B を選び、A と B の間にドアを設置する。例えば、部屋 10 を選び、10 の choosable_list からランダムに 6 を選択し、部屋 10 と 6 の間に 1 それから、A と B 二つの部屋のクラスタ番号を比較し、より小さい番号の方に統一する。同時に、この二つの部屋と通じるすべての部屋とクラスタ番号と統一する。例えば、部屋 6 と 10 は小さい方の 6 に統一した後、部屋 5 と 6 の間にもドアが設置されたため、部屋 5、6、10 は全部より小さいクラスタ番号 5 に統一する。

以上のプロセスを繰り返し、全ての部屋のクラスタ番号が 0 になり、また、すべての部屋の選択可能な隣人リストが空になるまで続く。

ドア設置アルゴリズムは、既存の迷路生成クラスカル法を発展させたものである。両者の違いと言えば、まず、隣接する部屋数が挙げられる。既存クラスタリング法は、上下左右の 4 近傍に隣接するものが配置と仮定している。ドア迷路の場合、隣接する部屋の数と配置がより複雑になっている。隣接情報を独自のクラス Room で管理することによって、複雑な隣接関係にも確実に対応するように拡張した。



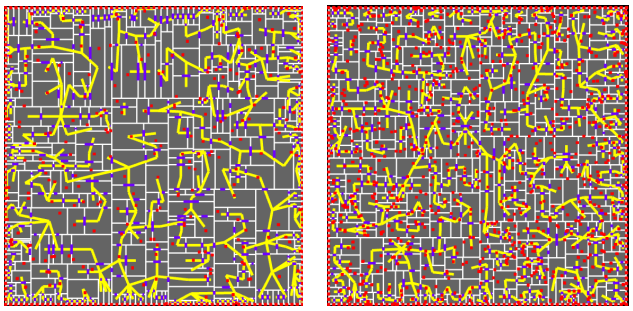


図６　長方形部屋分割アルゴリズムとドア設置アルゴリズムで自動生成した様々なドア迷路

図６に示すように、長方形部屋分割アルゴリズムとドア設定アルゴリズムを使い様々なドア迷路を生成することができる。変数(N, m)が異なる結果であり、黄色の線はドアを通した部屋同士のつながりを表す。我々のアルゴリズムの頑丈性が様々な実験を通して立証できた。例えば、図 3 は変数 N と m を異なる数値に設定し、得られたドア迷路の結果を示す。部屋の数と大きさが変化しても、確実にスタートからゴールまでの一本道、迷わせるための複数の小道が生成されている。また、ドアを通して、全ての部屋まで辿り着けることが確保できる。

4. 正多角形部屋分割とドア迷路

正多角形による平面充填とアルキメデスの平面充填形の幾何学[1,3]を部屋迷路に応用し、対称性を持つ多様なパターンで部屋分割実現できる。全部で以下の 11 種類の分割パターンがある。ここでは、正多角形の一辺の長さ a の値を制御し、部屋の大きさを変えることができる。そして、部屋の数と迷路の複雑さを制御することができる。部屋数が増えれば増えるほど迷路が複雑かつ挑戦的になる。また、正多角形の部屋分割は迷路の形に対称性と美しさを与え、ゲームに限らず、人々を楽しむ参加型プロジェクションマッピングにも利用できる。

図 7 は、正多角形の部屋分割による 11 種類ドア迷路を示す。具体的には、正 3 角形、正 4 角形、正 6 角形による部屋分け(3 種類)、正 3 角形と正 6 角形による部屋分け(2 種類)、正 3 角形と正 4 角形による部屋分け(2 種類)、正 3 角形と正 12 角形による部屋分け(1 種類)、正 4 角形と正 8 角形による部屋分け(1 種類)、正 3 角形と正 4 角形と正 6 角形による部屋分け(1 種類)、正 4 角形と正 6 角形と正 12 角形による部屋分け(1 種類)などがある。

正多角形の部屋は、長方形の部屋と同様なクラス *Room* で管理する。ドア配置アルゴリズムは長方形の時と同様に実行する。しかし、部屋の隣接関係の判定に関してはより複雑な判定をしなければならない。また、各多角形部屋の頂点の座標、頂点数、辺数などの情報を加えて記述しなければならない。

Vol.2021-CG-184 No.27
Vol.2021-DCC-29 No.27
Vol.2021-CVIM-227 No.27
2021/11/6

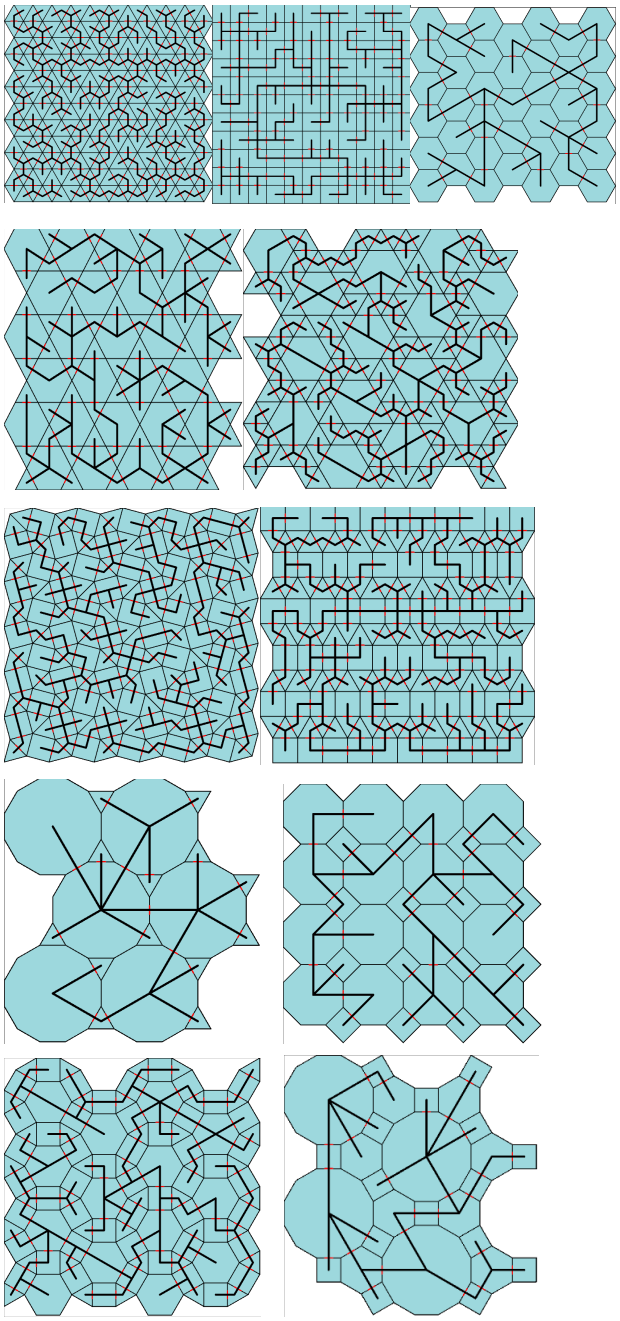


図 7　11 種類の正多角形部屋分割ドア迷路

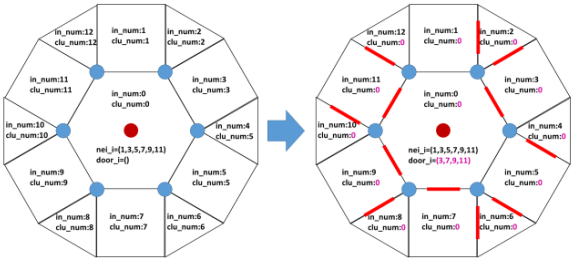


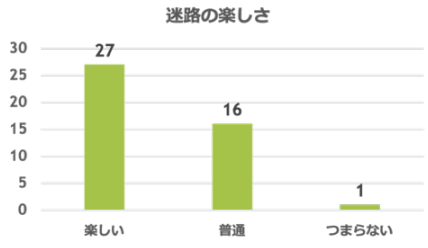
図 8　三種類の正多角形における部屋分割のクラス *Room* の具体例

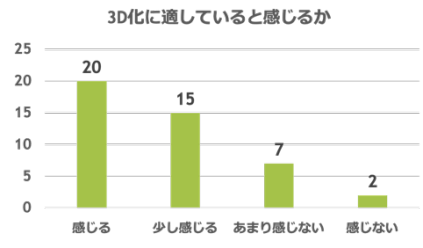
更に、隣接する部屋を探すとき、個々の辺に対して、共通の辺を持つ隣の部屋を探し、隣接する部屋のリストを作成しなければならない。このリストがあれば、長方形と同様にドア配置アルゴリズムで「行ける」部屋を決められる。図 8 は、正六角形と正三角形と正四角形による部屋分割を例にして説明する。中心にある正六角形に注目してほしい。正六角形に隣接するのは正 4 角形であるが、正 3 角形は隣接していない。ドアを配置するために、共通な頂点 2 つ以上を持つ正多角形同士に限る。

5.　結果検証

アルゴリズムの頑丈性を検証するために、沢山の長方形や正多角形のドア迷路を作った。変数 N や m や a の値をランダムに変化させながら、12 種類それぞれ 100 パターンの迷路を自動生成した。変数の範囲として、N は[4, 10000]、m は[1, N/4]、a は[10, 100]だった。結果として、変数 N と m または a が変化しても、迷路が破綻することなく確実にスタートからゴールまでの一本道と迷わせるための複数の小道が生成できた。また孤立する部屋は存在せず、ドアを通して、全ての部屋に行くことができた。

また、アルゴリズムの有用性を検証する心理実験を行った。検証方法として、11 名の大学生に実際にドア迷路をプレイしてもらい、評価してもらった。参加者には用意された迷路 13 個から 4 個を選ばせ、プレイさせた。4 個の迷路にかかったプレイ時間とアンケートの 2 項目について答えさせた。アンケートには、「迷路の楽しさ」と「三次元化に適しているかどうか」の 2 つの質問があった。「迷路の楽しさ」に関しては、3 段階(楽しい、普通、つまらない)の評価、「三次元化に適しているかどうか」に 4 段階(感じる、少し感じる、あまり感じない、感じない)がある。





Vol.2021-CG-184 No.27
Vol.2021-DCC-29 No.27
Vol.2021-CVIM-227 No.27
2021/11/6

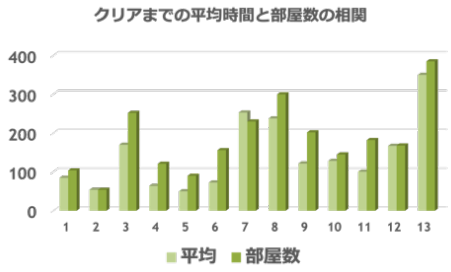


図 9 ドア迷路に関するアンケート結果

図 9 はアンケートの結果とプレイ時間をまとめたグラフである。「迷路の楽しさ」については過半数の参加者から「普通」以上の評価と、「三次元化に適しているか」に「少し適している」以上の評価がでた。また、迷路のプレイ時間に関しては、迷路にかかった時間と迷路の部屋数を比較したところ、少数のケースがあるが、大方比例していることが分かった。

6. おわりに

本論文は我々が独自に研究した「ドア迷路」自動生成アルゴリズムについて紹介した。平面充填に基づく「部屋分割アルゴリズム」と、従来の迷路生成クラスカル法を拡張した「ドア設置アルゴリズム」が研究の特徴である。多種多様な「ドア迷路」を自動的に生成できた、また、実験によりアルゴリズムの頑丈性と実用性を検証できた。これから、ドア迷路の自動生成アルゴリズムを 3 次元空間に拡張したい。また、ドア迷路は対戦ゲーム、アドベンチャーゲーム、探索ゲームなどのゲーム環境自動作成に応用されることを期待する。

参考文献

[1] Conway, John H.; Burgiel, Heidi; *Goodman-Strauss, Chaim* (April 18, 2008). "Chapter 21, Naming the Archimedean and Catalan polyhedral and tilings, Euclidean Plane Tessellations". The Symmetries of Things. CRC Press.

[2] 茶谷卓実、郭清蓮、屋敷型迷路自動生成アルゴリズム、第 82 回全国大会講演論文集、Vol. 2020, No. 1(2020-02-20)

[3] Grunbaum, Branko; Shaphard, G. C., Tilings and Patterns, W. H. Freeman and Company, pp.59, 96.

[4] 池田 心、絵画的迷路生成のある拡張，組合せゲーム・パズルミニプロジェクト第 5 回ミニ研究集会（2010）

[5] Joseph. B. Kruskal: On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. In: Proceedings of the American Mathematical Society, Vol 7, No. 1 (Feb, 1956), pp. 48–50

[6] Karai のドア迷路、(http://karai.secret.jp/door.html, 2021 年 9 月 8 日アクセス)。

[7] 望月士郎、浮き出し迷路、Vol.1, 学研（2006)。

[8] Michael T. Goodrich and Roberto Tamassia. Data Structures and Algorithms in Java, Fourth Edition. John Wiley & Sons, Inc., 2006. ISBN 0-471-73884-0. Section 13.7.1: Kruskal's Algorithm, pp.632.

[9] 任天堂、ルイージマンション：オバキュームの使い、(https://www.youtube.com/watch?v=IIa58vM19zc, 2021 年 9 月 8 日アクセス)。

[10] 城間一樹、Shadow Corridor、(2019)、(https://www.spaceonigirigames.com, 2021 年 9 月 8 日アクセス)。

[11] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to Algorithms, Second Edition. MIT Press and McGraw-Hill, 2001. ISBN 0-262-03293-7. Section 23.2: The algorithms of Kruskal and Prim, pp.567–574.