6Z-1

複数の玉を扱う携帯型迷路ゲームの自動生成

浅賀 一樹　　藤代 一成　　大野 義夫
慶應義塾大学 理工学部 情報工学科

1 背景と目的

芸術・デザインの分野において長年受け継がれてきた作品として，迷路・迷宮がある．それらの迷路・迷宮は，その構造・性質によって 6 種類に分けることができる [1]．特に「単純接続迷路 (Simply-connected maze)」とよばれる「環状構造をもたない迷路」を，コンピュータを用いて自動的に生成するアルゴリズムに関する研究が近年多数行われている (例えば [2])．

迷路を自動的に生成する利点の 1 つは，ユーザがいくつかのパラメータを設定するだけで，簡単に迷路を生成することができる点である．これによって，ユーザは迷路生成に関する深い知識無しでも，複雑な構造をもつ迷路を簡単に生成し，利用することができる．

そこで本研究では，最小極大木をランダムに生成することによる迷路生成アルゴリズムを発展させ，迷路上を複数の玉が同時に動くという複雑な迷路ゲームを自動的に生成する方法を提案する．

本手法によって，ユーザが玉の個数・迷路の大きさなどのパラメータを設定するだけで，自動的に迷路が生成される．また本研究では，実装環境に Android 端末を使用し，傾きセンサやタッチパネルを用いてユーザに対しより直感的な操作を可能にする．

2 アプローチ

2.1 パラメータ設定

玉の個数 (1〜4 個)，迷路の大きさ (「小」「中」「大」の 3 種類) をユーザがそれぞれ任意に設定することで，迷路が自動的に生成される．

4 個の玉はそれぞれ色分けされており，対応するゴールの色も色分けするか，すべて黒に統一するか，ユーザは任意に選ぶことができる．迷路の大きさに関しては，大きさ「大」の場合に限って，表示可能な最大サイズが変わるようにした．

2.2 迷路生成

迷路生成の基礎は，初めに格子 (grid) を用意し，格子の構成要素であるセル (cell) の壁を 1 つずつ取り除いて道を作っていくことである．提案方法では，格子におけるセルをグラフの頂点，セルの壁をグラフの辺，壁の壊れやすさを辺の重みにそれぞれ対応させることによって，迷路生成の問題をグラフの問題に帰着させている．迷路の格子構造をグラフ構造として扱うことで，後に行う迷路の領域分けやゴール設定などの操作を行いやすくしている．

ここでは環状構造をもたない迷路を生成するために，グラフの最小極大木を生成するアルゴリズムであるクラスカル法 [3] を利用している．

提案方法では，複数の玉が通る経路をそれぞれ別の領域に分けることによって，同時に複数の玉を扱うことを可能としている．領域分けは，その境界が目で見てすぐにわからないようにするため，格子の 4 隅に存在するセルを初期セルとして選び，そのセルから 4 近傍にあるセルを領域候補として記憶した後，その領域候補から 1 セルずつランダムに選んで同様の操作を繰り返すという方法をとる．領域分けが行われる様子を図 1 に示す．

領域分けが行われた後，領域の境界をあらかじめ迷路の壁として確定し，それ以外の壁を消去することで複数の経路を一度に生成することができる．

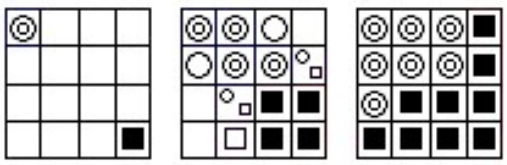

図 1　領域分けが行われる様子．左から初期状態，途中経過，最終状態．◎および■の領域がすでに選ばれているセル，○および□の領域が次に選ばれる候補のセルである．

2.3 スタート＆ゴールの設定

スタートは，各領域内に存在する行き止まりのセルからランダムに選んだ 1 つのセルに設定する．

提案方法では，スタートとゴールの距離が極端に短い場合は，簡単すぎる迷路として不適切としている．そのためゴールは，設定したスタート地点のセルから幅優先探索を用いて最も遠いセルを探し出し，そのセルをゴール地点として設定する．

2.4 玉の移動

玉の移動は，Android 端末の傾きセンサを利用する．端末の傾きが大きいほど，それぞれの玉にかかる加速度は大きくなる．

提案方法では，玉画像を表示し，移動・衝突処理を行った後に再び玉画像を表示するまでの一連の動作を 1 ステップとし，そのステップを何度も繰り返し行うことで玉の移動を表現している．

Automatic generation of maze games with two or more balls
Kazuki Asaga　　Issei Fujishiro　　Yoshio Ohno
Department of Information and Computer Science, Keio University

2.5 壁と玉の衝突

提案方法では，1 ステップ後の移動地点が壁と重なる位置になってしまう場合に，玉の位置を一番近くにある壁と接する位置に修正し，衝突係数などを考慮したうえで適切な速度に変更することで，壁と玉の衝突を再現している．

また，衝突処理を行った後の玉の位置がその玉に対応するゴール地点と重なった場合，その玉は「ゴール済」と判定され，玉の画像をゴール後の意匠に変更後，それ以降その場に止まったまま動かないようにする．

2.6 その他の機能

提案方法では，同じ設定で迷路を再生成する・初めから設定をやり直す・生成した迷路を初めからプレイし直すなど，迷路の任意再生成・再プレイも可能となっている．

また，ユーザがプレイ中に一時休止したい場合などのために，一時停止機能を設けている．

3 実装と実験

3.1 実装

本研究では，実装環境として Android 端末を使用し，開発言語として Java を用いてプログラミングを行った．また，実機デバッグ用の端末として Android OS 2.2 を搭載する「DoCoMo Garaxy S」を使用した．

図 2 および図 3 に，実際に生成されたゲームのキャプチャ画像 (ゲームプレイ中の画面) を示す．

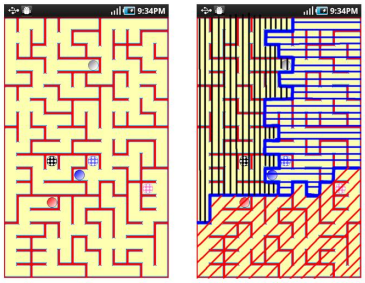

図 2 生成された迷路ゲーム．左の迷路は右図のようにそれぞれ領域分けされている．

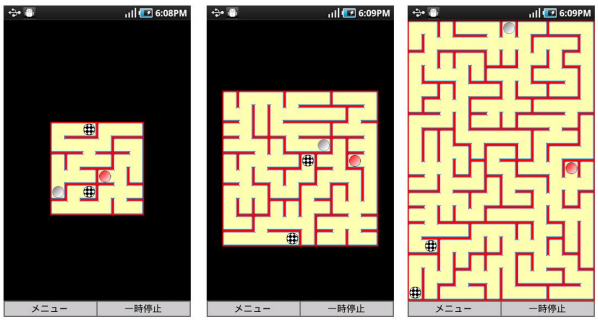

図 3 生成された迷路ゲーム．玉の個数はいずれも 2 個で，左から大きさ「小」・「中」・「大」の場合．大きさ「大」のサイズは横 12 セル×縦 18 セルである．

3.2 評価実験

生成された迷路を，「面白さ」という観点から客観的に評価することは難しい．そこで今回は，ゲームクリア時にスタートからゴールするまでにかかった時間を表示させ，その時間の長短によってその迷路の「難しさ」を推定するという方法で評価を行った．

スタートからゴールまでの距離が長く，途中で曲がる回数が多く，それぞれの玉のスタートからゴールまでの進行方向が異なる場合に，クリア時間が長くなる傾向があった．また，図 4 に見られるような経路が多数存在する場合など，特定の迷路構造によってもクリア時間が長くなる場合があることがわかっている．

これらの要素をそれぞれ適切に重みづけすることによって，迷路の「難しさ」を定量的に表すことが可能であると考えられる．

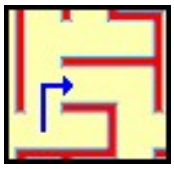

図 4 クリア時間が長くなる迷路構造の例．矢印の方向に進む場合，曲がりたい地点で上手く玉を止める必要があるため，スムーズに進ませるにはある程度の慣れが必要となる．

4 まとめと今後の課題

本稿では，既存のアルゴリズムを応用し，Android 端末上で複数の玉を同時に扱う迷路ゲームを自動生成する方法を提案した．また，迷路の「難しさ」にスタートからゴールまでの距離や，途中で曲がる回数，各玉の進行方向の関係，特定の迷路構造などさまざまな要素が関わっていることを示した．

現段階では，玉ごとに経路を完全に分けることで複数の玉を扱うことを可能にしているが，玉同士の衝突を実現することによって単一経路をもった迷路，複数の玉を同じゴールへ導く迷路など，より多くの種類の迷路を生成することが可能になると考えられる．

また，玉と壁との衝突判定を簡略化するために格子を正方形格子だけで考えており，円形格子などを用いて様々な形状をもつ迷路などへの本研究の応用などが，今後の課題として挙げられる．

参考文献

[1] Berg, C. : Amazeing Art, http://www.amazeingart.com.

[2] Xu, J., Kaplan, C.S. : “Image-Guided Maze Construction,” ACM Transactions on Graphics, Vol. 26, No. 3, Article 29, July 2007.

[3] 石畑 清：アルゴリズムとデータ構造，岩波書店，2007 年，pp. 280-290.

[4] 布留川 英一：Android 2.1 プログラミングバイブル，ソシム株式会社，2010 年.