修士論文　　2020 年度

推移律を保証する効用指標による ゲーム AI に関する研究

東京工科大学大学院

バイオ・情報メディア研究科

メディアサイエンス専攻

古川 真帆

修士論文　　2020 年度

推移律を保証する効用指標による ゲーム AI に関する研究

指導教員　渡辺 大地 教授

東京工科大学大学院

バイオ・情報メディア研究科

メディアサイエンス専攻

古川 真帆

論 文 の 要 旨

論文題目	推移律を保証する効用指標による ゲーム AI に関する研究
執筆者氏名	古川 真帆
指導教員	渡辺 大地 教授
キーワード	ゲーム AI，ユーティリティベース AI，ゴールベース AI， 評価関数，ラプラシアン，ファジー推論

[**要旨**]

ゲーム AI において，現在の状態に対し何かしらの評価関数により行動を選択する手法をユーティリティベース AI と呼ぶ．ユーティリティベース AI の中には，目的地への到達などの最終的な目標達成を保証する評価基準を持つゴールベース指標と，局所的な評価のみにより行動を決定し，最終的な目標達成を保証しない非ゴールベース指標がある．ユーティリティベース AI はスカラー値による評価関数であることから，複数の評価指標を足し合わせて同時に考慮することが可能という特長がある．しかしながら，ゴールベースと非ゴールベースを足し合わせて両方の考慮を行った場合，ゴールベースの特性が失われて目的達成の保証ができなくなるという問題がある．

本研究は，ゴールベースと非ゴールベースを両方同時に利用しても，最終的な目標をできるだけ迅速に達成する行動の実現を目的とする．本研究では，疑似ラプラシアンを用いた盛り土関数を使って推移律を保ち最終的な目標達成を保証する手法を提案する．例として，ゴールベースを目的地到着，非ゴールベースを追跡エージェントからの回避行動とし，追跡エージェントを回避しつつ目的地到達を迅速に行うという AI を盛り土関数を用いて実現した．また，盛り土関数の調整係数は AI の体力や停滞度の数値を使用しファジー推論を用いて動的に変更するようにした．

A b s t r a c t

Title	A Study on Game AI with Utility Indicators to Guarantee the Transitive Law
Author	Maho Furukawa
Advisor	Taichi Watanabe
Key Words	Game AI, Utility-based AI, Drop Goal-based AI, Evaluation function, Laplacian, fuzzy inference

[**summary**]

In game AI, the method of selecting an action for the current state by some evaluation function is called utility-based AI. There are two types of utility-based AI: goal-based indicators with evaluation criteria that guarantee the achievement of final goals, such as reaching the destination, and non-goal-based indicators that determine actions by local evaluation only and do not guarantee the achievement of the final goal. Utility-based AI is an evaluation function with scalar values, and thus it has the advantage that multiple evaluation indices can be added together and considered simultaneously. However, if both goal-based and non-goal-based considerations are added together, there is a problem that the characteristics of the goal base are lost and the achievement of the goal cannot be guaranteed.

This study aims to achieve behaviors that achieve the final goal as quickly as possible, even when utilizing both goal-based and non-goal-based simultaneously. In this study, we propose a method to maintain the transition law and guarantee the achievement of the final goal by using a fill function with a pseudo-Laplacian. As an example, we used a fill function to realize an AI in which the goal base is the arrival at the destination and the non-goal base is the avoidance action from the tracking agent, and the AI is to reach the destination quickly while avoiding the tracking agent. The adjustment coefficients for the fill function were changed dynamically using fuzzy reasoning with the AI's fitness and stagnation values.

目 次

図 目 次

第 1 章

はじめに

1.1 研究背景と目的

近年のゲームにおいて，ゲーム AI という高度な技術が重要視されている [1][2]．ゲーム AI とは，ゲーム内の人工的に作り出された知能のことである．その 1 つに，キャラクター AI というプレイヤーが操作しない仲間や敵などのキャラクターに搭載する人工知能があり，『FINAL FANTASY XV』や『ドラゴンクエスト X』など多くのゲーム作品で用いられている．キャラクター AI に関する研究は数多く行われており，アクションや格闘，弾幕シューティング，パズル，人狼，将棋，囲碁など幅広く研究が行われている [3][4][5][6][7][8]．例えば，藤井ら [9] はアクションゲーム「Infinite Mario Bros.」において，人間の生物学的制約を課した機械学習によって人間らしい振る舞いを自律的に獲得する AI を作成した．星野ら [10] は模倣学習の手法を用いて COM の行動パターンを試合ごとに拡張し，成長する格闘ゲームキャラクターを作成した．渡辺ら [11] は，波動方程式を用いた追跡アルゴリズムを提案している．

キャラクター AI は多様な手法が用いられるが，そのうちの 1 つとしてユーティリティベース AI がある．ユーティリティベース AI とは，現在の状態に対し何かしらの評価基準により評価値を算出し，より良い評価となる行動を選択する手法である．例えば，危険度を評価基準とし，敵と近いほど危険であるとした場合，敵との距離が評価値となる．これらの評価基準や評価値は，AI の制作者が自由に設定することができるものである．図 1.1 は，ユーティリティベース AI のイメージ図である．『ぎゅわんぶらあ自己中心派』という麻雀ゲームでは，AI がどの手を選ぶかは評価関数によって決定されており，『The Sims』という人生シミュレーションゲームでは，感情や生理を評価基準としている [12]．評価関数を用いたゲームに関する研究は多くあり，仲道ら [13] はコンピュータ将棋で評価値が指し手の互角に近いほど好ましいとなるように変換を行い，ユーザーの棋力に合わせる手法について述べている．また，大森ら [14] は将棋の攻めと受けという概念について棋譜の分類と評価関数の学習を行い，棋風を反映した評価関数が得られる手法につい

て述べている．

ユーティリティベース AI の中には，目的地への到達などの最終的な目標達成を保証する評価基準を持つゴールベース指標と，局所的な評価のみにより行動を決定し，最終的な目標達成を保証しない非ゴールベース指標がある．ゴールベースは目的地（ゴール）到達を最終的な目標とした場合，スタートからゴールまで順に辿っていけば必ずゴールに辿り着くという推移律が保たれているという特性がある．『F.E.A.R』『Deus Ex: Human Revolution』『Middle-earth: Shadow of Mordor』『サカつく DS』など，多くのゲーム作品でゴールベースが用いられている [15][16]．また，Studiawan ら [17] はゴールベース AI の一種である GOAP 手法をリアルタイムストラテジーゲームに適用する手法について述べている．

一方，ユーティリティベース AI はスカラー値による評価関数であることから，複数の評価指標を足し合わせて同時に考慮することが可能という特長がある．しかしながら，ゴールベースと非ゴールベースを足し合わせて両方の考慮を行った場合，ゴールベースの推移律が崩れ目標達成の保証ができなくなるという問題がある．例えば，目的地に向かって経路探索をしているエージェントが周囲を敵に囲まれて袋小路の状態になった場合，敵を避け続けて目的地に辿り着かなくなる可能性が考えられる．

また，効用指標による推移律崩壊が生じた際に復帰するという研究は見当たらなかった．

そこで，本研究ではゴールベースと非ゴールベースを両方同時に利用しても，最終的な目標をできるだけ迅速に達成する行動の実現を目的とし，疑似ラプラシアンを用いた盛り土関数を使って推移律を保ち，最終的な目標達成を保証する手法を提案する．具体的には，ゴールベースを目的地到着，非ゴールベースを追跡エージェントからの回避行動とし，追跡エージェントを回避しつつ目的地到達を迅速に行うという AI を盛り土関数を用いて実現した．

本研究の初期段階の成果を NICOGRAPH2020 にて口頭発表を行った [18]．これに対し調整係数の動的変更機能を追加し，AI による行動がより状況に応じて適切となるよう拡張を行った．ま

た，その動的変更が有効となるようにシミュレーション用ゲームルールの拡張を行い，逃亡エージェントの体力値の設定や追跡エージェントによる攻撃を追加した．

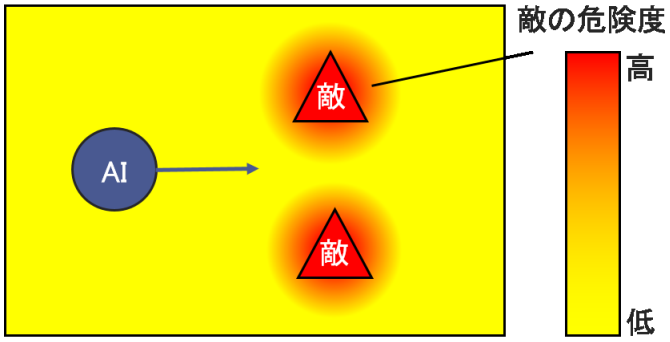

図 1.1　ユーティリティベース AI のイメージ

1.2　論文構成

本論文は全 4 章で構成する．2 章ではゴールベースと非ゴールベースについて説明し，本研究の提案手法について説明する．また，3 章では評価と考察を行い，4 章では本研究のまとめを示す．

第 2 章

提案手法

2.1 ゴールベース指標と非ゴールベース指標

本研究では，ユーティリティベース AI で用いる評価関数指標を「ゴールベース指標」と「非ゴールベース指標」の 2 種類に分類する．

「ゴールベース指標」とは，評価関数全体で最小値以外の極小値が存在しない指標のことである．ゴールベース指標では，評価関数が減少する方向に入力パラメータを変更していくと，必ず最小値に到達できることが保証される．このような指標では，評価関数が最小となるようなパラメータは最急降下法 [19] 等を用いることで容易に求めることができる特性がある．経路探索においては，各経路点からゴールまで到達する距離を「コスト値」と設定する手法が多く，このコスト値を指標に用いると，ゴールに近いほどコスト値は下がるため，経路探索コスト指標はゴールベース指標と言うことができる．ゴールベース指標による行動遷移では，最適行動を取った場合の評価値は常に単調減少となる．これは，状態遷移を $\{S_0, S_1, \ldots\}$, 評価関数を $G(S_i)$ と表したとき，$i < j$ なら常に $G(S_i) > G(S_j)$ という推移律を満たすことを意味する．

一方，「非ゴールベース指標」は「ゴールベース指標」を満たさない指標のことである．例えば，出発地から障害物を避けつつ目的地を目指すような状況のとき，ゴールから現在値までの直線距離を指標としたとき，迂回をしなければ目的地へ到達できないような状況では，ゴールへの過程で指標が増加してしまうことがありえる．このような指標は「非ゴールベース指標」と言える．

前述したように，ユーティリティベース AI では，複数の指標を足し合わせることが可能であるが，そのときにゴールベース指標と非ゴールベース指標の両方を足してしまった場合，推移律が崩れてしまうため，推移律を前提としたアルゴリズムは一切利用できなくなるという問題が発生する．例えば，経路探索によって求めた目的地への最適経路をゴールベース指標とし，追跡エージェントとの距離を非ゴールベース指標とした場合，これらを足し合わせると評価関数の局所的極小値が発生してしまい，目的地への到達が保証されなくなることがある．図 2.1 は，逃亡エー

ジェントが追跡エージェントに囲まれ，目的地への到達が困難な状況を表したものである．

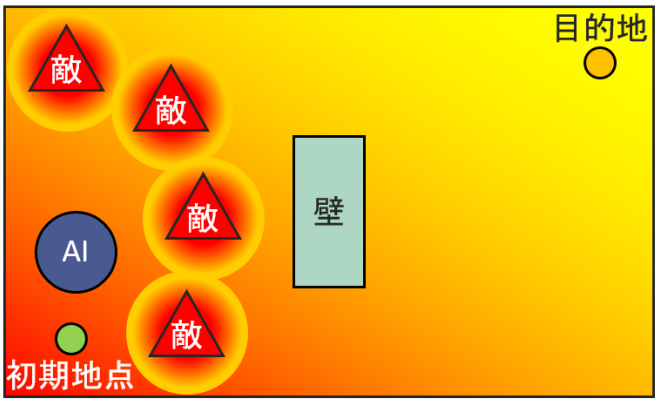

図 2.1　目的地への到達が困難な状況

本手法は，このような状況において，補正関数を用いて融合指標に対して推移律を復帰し，局所的極小値から脱却して真の最小値状態（目的地）に到達することを目的とする．

2.2　手法概要

ゲーム中でエージェントが移動できる領域において，位置 $\mathbf{P}$ におけるコスト評価関数 $T(\mathbf{P})$ が定義できるものとする．ここで，$T(\mathbf{P})$ はエージェントの「望ましさ」をスカラー値で表したものであり，小さいほど望ましいものと定義する．例えば，エージェントに目的地があり，目的地に近いほど望ましいのであれば，目的地に近づくほど $T(\mathbf{P})$ は小さい値となる．また，同様に敵との距離が離れている方が望ましい場合，敵に近い位置では $T(\mathbf{P})$ の値は増大することになる．

ここでは，目的地への移動距離によるコスト関数を $G(\mathbf{P})$ とおく．図 2.2 は，$G(\mathbf{P})$ をスカラー場としてコスト値に応じて着色した様子である．目的地への移動距離によるコスト値は，目的地に近づく方向に単調減少するため，$G(\mathbf{P})$ はゴールベース指標と言える．

また，敵からの距離によるコスト関数を $U(\mathbf{P})$ とおく．$U(\mathbf{P})$ も $G(\mathbf{P})$ と同様にスカラー場として捉えることができる．図 2.3 は，$U(\mathbf{P})$ のスカラー場を表している．$U(\mathbf{P})$ は，複数の敵が存在するとスカラー場として局所的な極値が存在するため，非ゴールベース指標と言える．

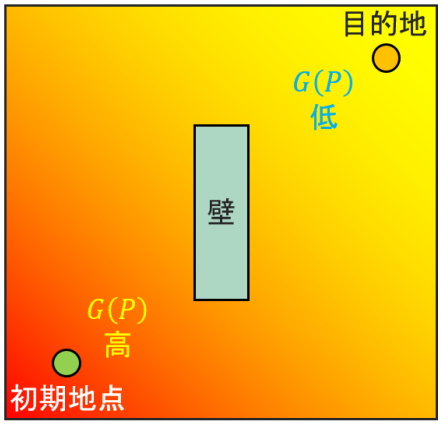

図 2.2　$G(\mathbf{P})$ のスカラー場

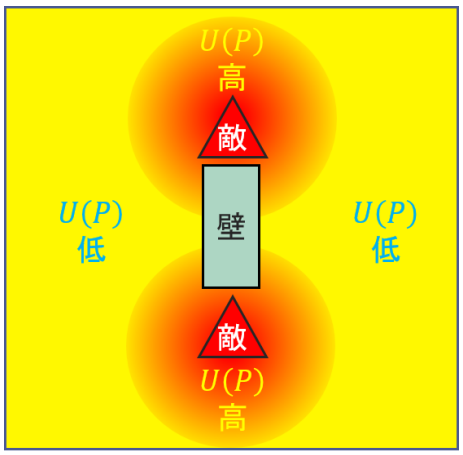

図 2.3　$U(\mathbf{P})$ のスカラー場

ここで，両方の指標を考慮する場合，コスト評価関数 $T(\mathbf{P})$ を以下のようにすることが考えられる．

$$T(\mathbf{P}) = \alpha G(\mathbf{P}) + \beta U(\mathbf{P}) \tag{2.1}$$

ここで α, β は調整係数とする．α, β は，ユーザーが任意の数値に設定して調整するものとし，現状は固定値である．

この評価関数は両方の指標を元にコストが構成されるが，前述したように推移律が保たれないため，$T(\mathbf{P})$ は「非ゴールベース」となってしまい，目的地到達が保証されなくなる．

本手法は，$T(\mathbf{P})$ の推移律を保つため，盛り土関数 $C(\mathbf{P})$ と調整係数 γ により

$$T(\mathbf{P}) = \alpha G(\mathbf{P}) + \beta U(\mathbf{P}) + \gamma C(\mathbf{P}) \tag{2.2}$$

とし，$T(\mathbf{P})$ の推移律を保つものである．γ は α, β と同様に，ユーザーが任意の数値に設定して調整するものとし，現状は固定値である．$C(\mathbf{P})$ については次節で説明する．この $T(\mathbf{P})$ を用いることで，エージェントは目的地への到達を保証できる．

2.3 疑似ラプラシアンによる盛り土関数

これまで述べてきたように，非ゴールベース指標となった評価関数の場合，最終的な目的値への到達が保証されないという問題がある．その理由は，コスト値が減少する方向に移動を続けた際，目的地とは異なる局所的極値に至ってしまい，目的地への到達経路が不明となるからである．

この状況を避けるためには，局所的極値のコストを増加することで，エージェントが局所的極値に向かうことを回避することが望ましい．このコスト増加を盛り土関数を用いて実現し，追加するコスト値は疑似ラプラシアンを用いて算出するものとした．

ラプラシアン ∇^2 は，本来は連続ベクトル場 ϕ において以下の式によって定義されるスカラー値である [20].

$$\nabla^2 \phi = \frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} + \frac{\partial^2 \phi}{\partial z^2} \tag{2.3}$$

この値は，流れ場においては流体が増加する箇所で正，減少する箇所で負の値が出るものである．ラプラシアンは，画像処理や 3DCG など様々な分野で広く使われている [21][22][23][24]．また，既に十分研究がなされており，実数の四則演算のみで算出が可能で計算時間が高速であるという利点がある．これは数学的に証明されており，本手法適用で期待できると考えた．しかしながら，本手法で扱っているエージェントの移動空間はグラフネットワークによる離散空間であり，

各経路点においてラプラシアンを算出することはできない.

そこで，本手法では以下の式によって (2.3) 式によるラプラシアンを擬似的に算出するものとした.

$$\nabla^2 \approx \sum_{j \in N(i)} (T(\mathbf{P}_i) - T(\mathbf{P}_j)) \tag{2.4}$$

ここで，i は算出対象となる経路点，$N(i)$ は i に隣接する経路点の集合，$\mathbf{P}_i$ は i の位置ベクトルである．疑似ラプラシアンの模式図を図 2.4 に示す．例えば，コスト値が 40 の場所は，周囲のコスト値が 50 の場所から差分である 10 をもらい，30 の場所に 10 を渡すため，結果的に疑似ラプラシアン値は 0 となる．

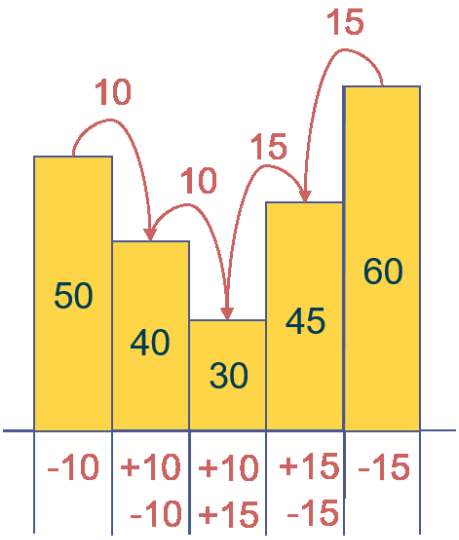

図 2.4　疑似ラプラシアンの概要図

各経路点は，盛り土関数用の補正値を保持するものとし，初期値を 0 とする．各経路点の疑似ラプラシアン値を算出し，一定の設定値を超えた場合に現在の補正値に追加する．その補正値を返値する関数を盛り土関数 $C(\mathbf{P})$ とした.

なお，補正値は時間経過により減衰するものとし，その減衰の割合は調整できるものとして，以下の式で表す.

$$C_t(\mathbf{P}) = (1-\epsilon)\left(C_{t-\Delta t}(\mathbf{P}) + \nabla^2(\mathbf{P})\right) \tag{2.5}$$

ここで，t は現在時刻，Δt はシーン更新時間である．また，ϵ は正の微少量を意味し，本研究における実装では 0.01 を用いた．

2.4 調整係数の動的変更

逃亡エージェントに体力（HP）を付与し，体力が少なくなるほど目的地への到達より追跡エージェントを避ける方を優先するといった，逃亡エージェントの状態による調整係数 β を動的に変更するようにした．

追尾しつつ攻撃してくる追跡エージェントを避けながら目的地を目指すというゲームにおいて，一般的にプレイヤーの行動として自身の体力が少なくなると追跡エージェントの攻撃を受けないために追跡エージェントを避ける行動を優先することが考えられる．また，同じ場所にいると追跡エージェントに囲まれる可能性が高くなることが想定される．そのため，逃亡エージェントがどのくらい追跡エージェントを避けるか左右する調整係数 β に体力や停滞度が影響すると考えた．

体力は実行開始時は最大値となっており，追跡エージェントのある一定範囲内にいると攻撃を受けて減るようにした．また，体力の自動回復を設けて徐々に最大値になるように設定した．一方，停滞度は逃亡エージェントが現在と一定時間前でどれくらい動いたか距離で算出した．

β を動的に変更する手法は，これら 2 つの数値を使用しファジー推論を用いた．ファジー推論 [25] とは，Lotfi Zadeh[26] が 1965 年に考え出したファジー論理を元に，Mamdani ら [27] が 1975 年に提案したものが源流の理論で，コンピュータが人間が問題を解決するのと同じ方法で言葉や言葉によるルールに関する推論が行える手段である [28][29]．ファジー推論は列車，空調設備，ロボットの制御など身の回りの様々な制御システムに利用されており，曖昧な入力に対する応答がスムーズであるという利点がある．この利点はゲームの動作制御や意思決定に応用できる

ため，調整係数 β の算出方法にこの手法を選択した．

本研究では，調整係数 β に逃亡エージェントの体力や停滞度が影響すると考え，時刻 t の時点での体力値 H_t を以下の法則になるよう実装した．

- 初期値は上限値とする．
- 追跡エージェントと接触した場合，一定量 H が減少する．ただし，接触時から一定時間は接触しても減少しない (つまり無敵状態になる) ものとした．
- 追跡エージェントと接触していない状態で，かつ H が上限値でない場合は一定量回復する．

時刻 t の時点での停滞度 L_t は，逃亡エージェントの現在位置と一定時間 f 前の時点での位置との距離とし，以下の式で算出した．

$$L_t = f \cdot S - |\mathbf{P}_t - \mathbf{P}_{t-f\Delta t}| \,. \tag{2.6}$$

ここで S は逃亡エージェントの速さを表し，$f \cdot S$ は L_t が取り得る最大値を意味することになる．結果として，L_t は大きいほど停滞した状態であることを意味する．

β を動的に変更する手法は，先ほど述べたようにこれら 2 つの数値を使用しファジー推論を用いた．基本的な方針としては，「体力値が大きい場合」と「停滞度が大きい場合」は追跡エージェントからの回避を軽視し，目的地到達を優先するというものである．重心法によるファジー推論関数 Z により

$$\beta = B - Z(H_t, L_t) \tag{2.7}$$

を算出し，調整係数 β の自動調整を実現した．B は β の初期設定値であり，β の上限値を意味する．

具体的な計算方法については，付録 A で記述する．

2.5 本研究における実装

$G(\mathbf{P})$ は，経路探索手法のダイクストラ法 [30] を用いて，ゴールベースの経路部分を実現した．ダイクストラ法とは，最初にスタートとゴールを設定し，ゴールから隣り合うコスト値を 1 ずつ減らしスタートまで決定していく最短経路手法である．また，非ゴールベースを表す $U(\mathbf{P})$ は，それぞれの追跡エージェントの周囲に危険度コストを配置し，追跡エージェントに近いほどコストを高くして最短経路のコストに加えた．

以上により算出されたコスト値に従って，コストが低い方へ逃亡エージェントは動いていくようにした．

第 3 章

評価

図 3.1〜3.64 は実行結果を示したものである．左下が初期地点，右上が目的地となっており，マップの青い部分はエージェントが通れない場所となっている．黄色の球体が逃亡エージェント，紫色の円錐が追跡エージェントとなっており，逃亡エージェントは追跡エージェントに近づくほど赤色になるように設定している．逃亡エージェントに体力を付与して最大値を 10 に設定し，体力が最大値の状態から実行が開始する．さらに，体力の自動回復を設けて徐々に最大値になるようにした．また，逃亡エージェントが追跡エージェントのある一定範囲内に近づくと，体力が 1 減るという追跡エージェントによる近接攻撃を再現した．そして，コストを可視化するためにマップをコストが高いほど赤く，低いほど青くなるようにし，逃亡エージェントの軌跡を赤い線で描くようにした．

3.1 自動追跡エージェントによる評価実験

3.1.1 通れない場所がないマップ

図 3.1，3.2，3.3，3.4 は疑似ラプラシアンを用いずに $\gamma = 0$ で，通れない場所がないマップを動くエージェントの様子である．図 3.1 は初期段階の様子，図 3.2 は追跡エージェントの包囲から脱却した様子，図 3.3 は 2 回目の追跡エージェントの包囲から脱却した様子，図 3.4 は目的地に到着したときの様子である．図 3.2，3.3 のように逃亡エージェントが追跡エージェントに囲まれた場合，追跡エージェントを横切って目的地へ向かった．

一方，図 3.5，3.6，3.7，3.8 は疑似ラプラシアンを用いて $\gamma = 0.3$ で動くエージェントの様子である．図 3.5 は初期段階の様子，図 3.6 は追跡エージェントの包囲から脱却した様子，図 3.7 は 2 回目の追跡エージェントの包囲から脱却した様子，図 3.8 は目的地に到着したときの様子である．疑似ラプラシアンを用いない場合と同様に，図 3.6，3.7 のように逃亡エージェントが追跡エージェントに囲まれた場合，追跡エージェントを横切って目的地に向かった．

表 3.1 は目的地到着までの経過時間と逃亡エージェントが攻撃を受けた回数を記録したもので

ある．逃亡エージェントが攻撃を受けた回数は疑似ラプラシアンの有無で差は見受けられなかったが，疑似ラプラシアンを用いた場合の方が用いない場合よりも到着にかかった時間は 2.8 秒短いという結果になった．

表 3.1　目的地到着までの経過時間と攻撃を受けた回数（通れない場所がないマップ）

疑似ラプラシアンの有無	目的地到着までの経過時間	攻撃を受けた回数
疑似ラプラシアンなし（$\gamma = 0$）	18.8 秒	1 回
疑似ラプラシアンあり（$\gamma = 0.3$）	16.0 秒	1 回

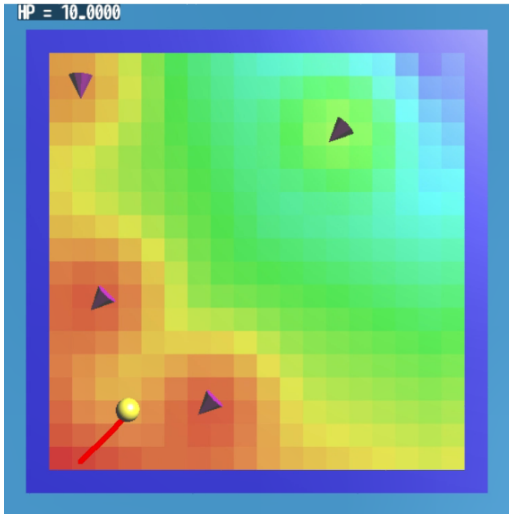

図 3.1　$\gamma = 0$ で動く様子（初期段階）

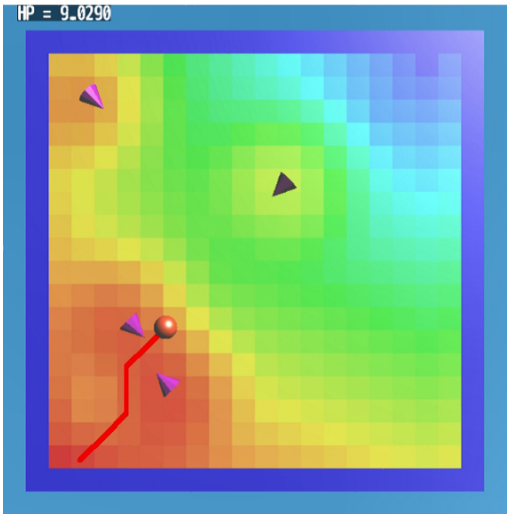

図 3.2　$\gamma = 0$ で動く様子（追跡エージェントの包囲から脱却）

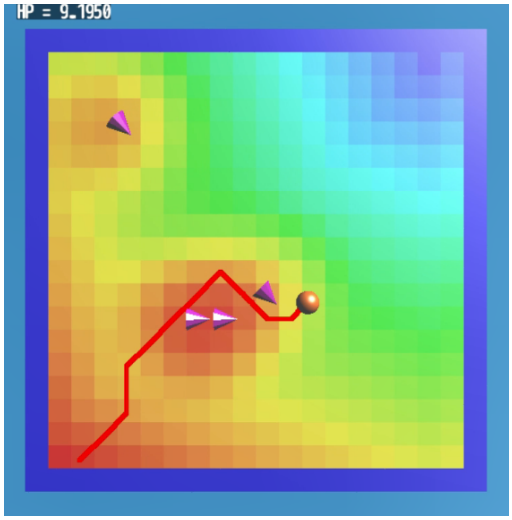

図 3.3　$\gamma = 0$ で動く様子（2 回目の追跡エージェントの包囲からの脱却）

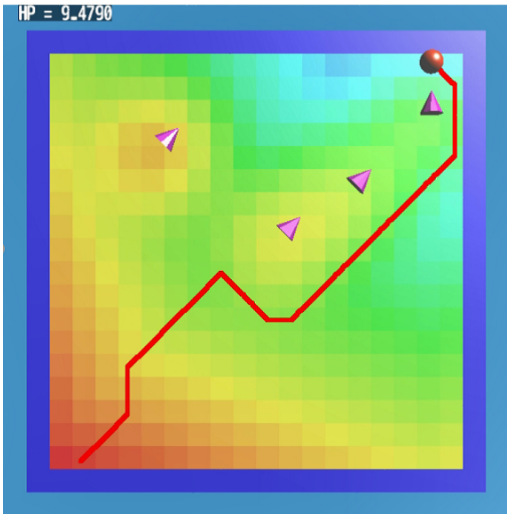

図 3.4　$\gamma = 0$ で動く様子（目的地到着）

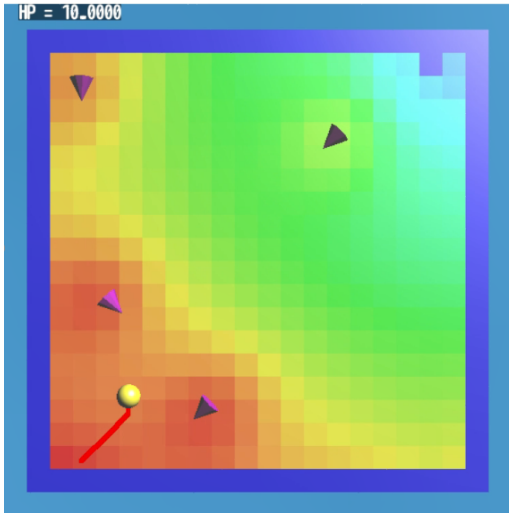

図 3.5　$\gamma = 0.3$ で動く様子（初期段階）

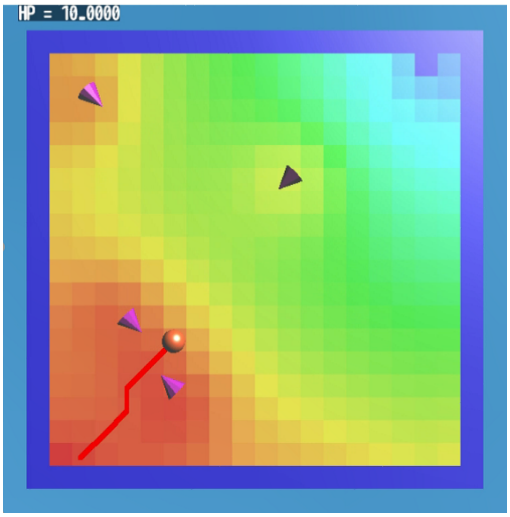

図 3.6　$\gamma = 0.3$ で動く様子（追跡エージェントに包囲された状態）

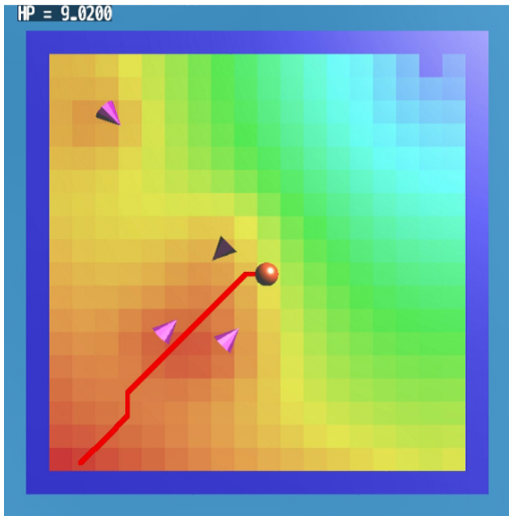

図 3.7　$\gamma = 0.3$ で動く様子（追跡エージェントの包囲から脱却）

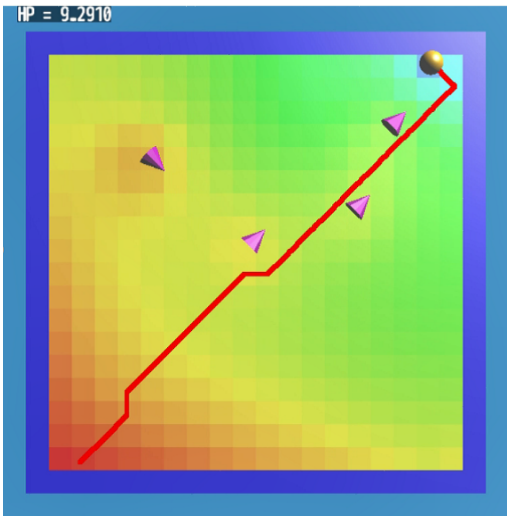

図 3.8　$\gamma = 0.3$ で動く様子（目的地に向かう様子）

3.1.2　通れない場所が 2 箇所あるマップ

図 3.9，3.10，3.11，3.12 は疑似ラプラシアンを用いずに $\gamma = 0$ で，通れない場所が 2 箇所あるマップを動くエージェントの様子である．図 3.9 は初期段階の様子，図 3.10 は追跡エージェントの包囲から脱却した様子，図 3.11 は 2 回目の追跡エージェントの包囲から脱却した様子，図 3.12 は目的地に到着したときの様子である．通れない場所がないマップと同様に，図 3.10，3.11 のように逃亡エージェントが追跡エージェントに囲まれた場合，追跡エージェントを横切って目的地へと向かった．

一方，図 3.13，3.14，3.15，3.16 は疑似ラプラシアンを用いて $\gamma = 0.3$ で動くエージェントの様子である．図 3.13 は初期段階の様子，図 3.14 は追跡エージェントの包囲から脱却した様子，図 3.15 は 2 回目の追跡エージェントの包囲から脱却した様子，図 3.16 は目的地に到着したときの様子である．疑似ラプラシアンを用いない場合と同様に，図 3.14，3.15 のように逃亡エージェントが追跡エージェントに囲まれた場合，追跡エージェントを横切って目的地に向かった．

表 3.2 は目的地到着までの経過時間と逃亡エージェントが攻撃を受けた回数を記録したもので

ある．通れない場所がないマップと同様に，逃亡エージェントが攻撃を受けた回数は疑似ラプラシアンの有無で差は見受けられなかったが，疑似ラプラシアンを用いた場合の方が用いない場合よりも到着にかかった時間は 2.6 秒短いという結果になった．

表 3.2　目的地到着までの経過時間と攻撃を受けた回数（通れない場所 2 箇所あるマップ）

疑似ラプラシアンの有無	目的地到着までの経過時間	攻撃を受けた回数
疑似ラプラシアンなし（$\gamma = 0$）	18.7 秒	1 回
疑似ラプラシアンあり（$\gamma = 0.3$）	16.1 秒	1 回

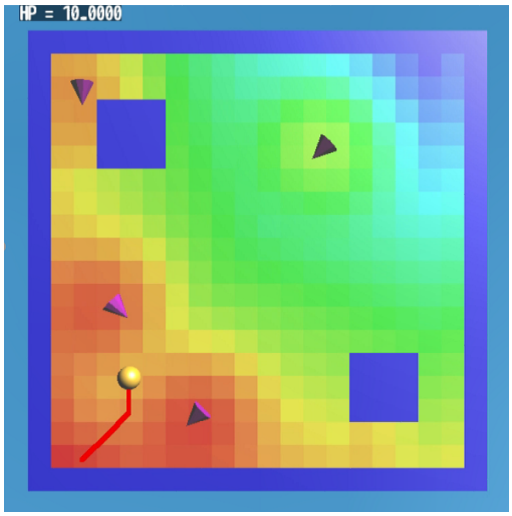

図 3.9　$\gamma = 0$ で動く様子（初期段階）

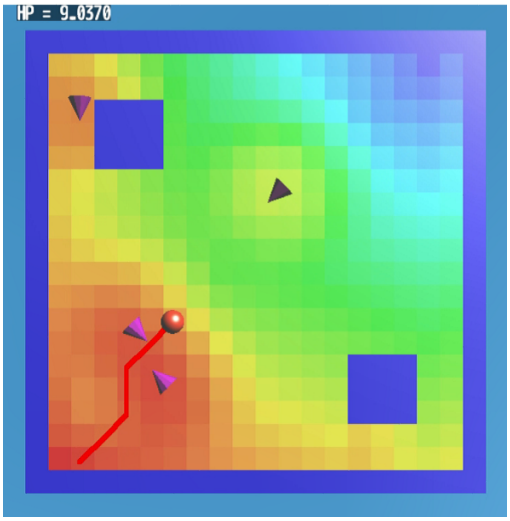

図 3.10　$\gamma = 0$ で動く様子（追跡エージェントの包囲から脱却）

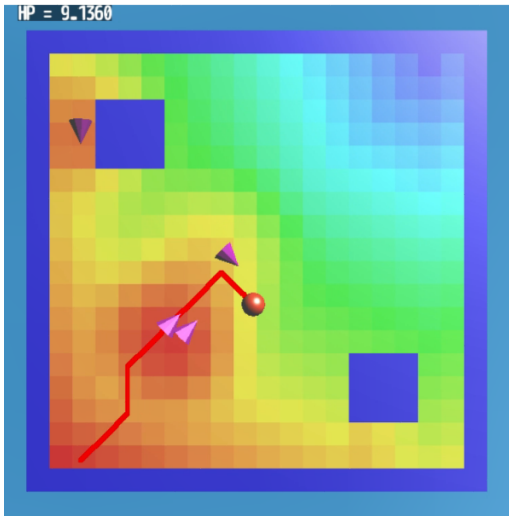

図 3.11　$\gamma = 0$ で動く様子（2 回目の追跡エージェントの包囲からの脱却）

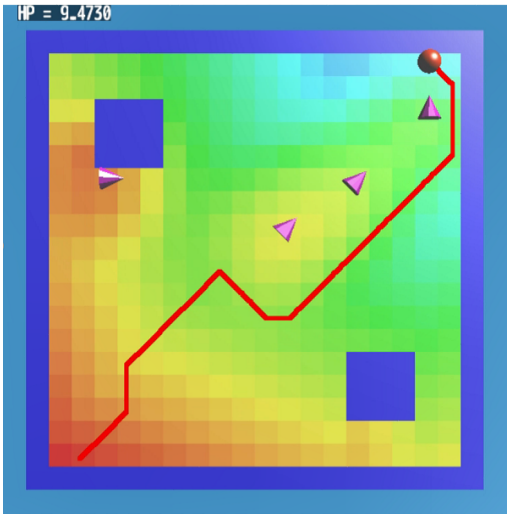

図 3.12　$\gamma = 0$ で動く様子（目的地到着）

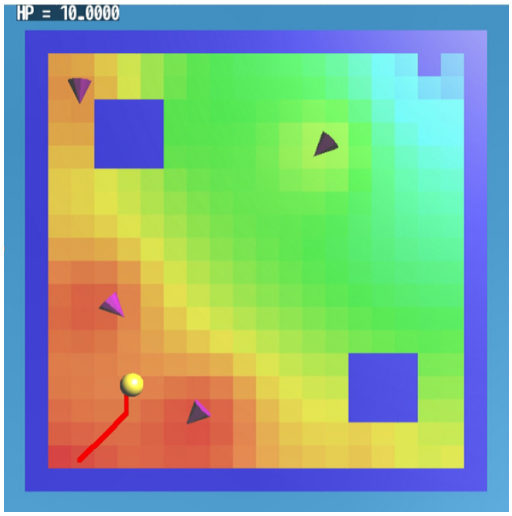

図 3.13　$\gamma = 0.3$ で動く様子（初期段階）

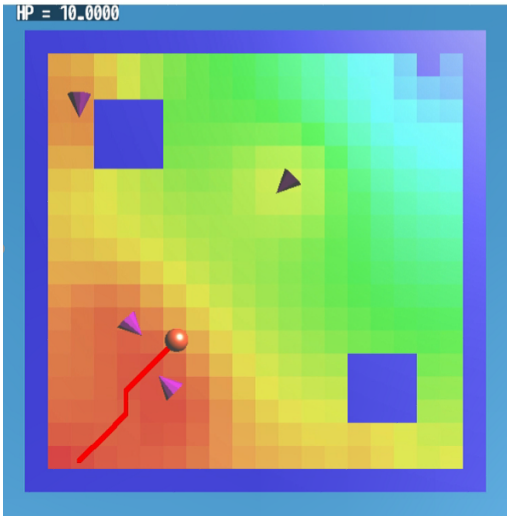

図 3.14　$\gamma = 0.3$ で動く様子（追跡エージェントの包囲から脱却）

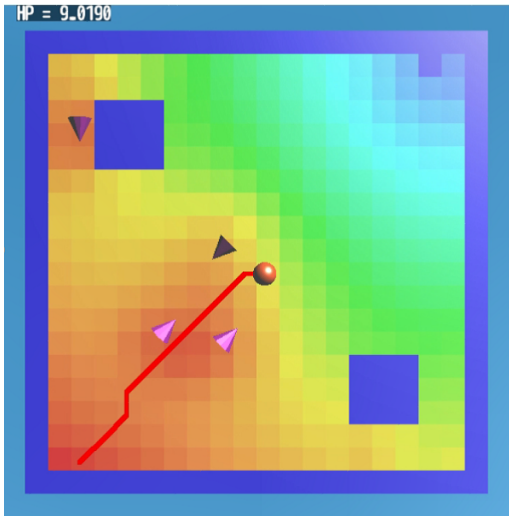

図 3.15　$\gamma = 0.3$ で動く様子（2 回目の追跡エージェントの包囲からの脱却）

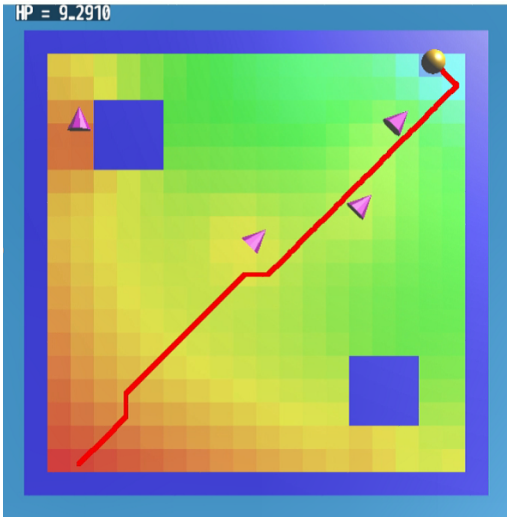

図 3.16　$\gamma = 0.3$ で動く様子（目的地到着）

3.1.3　通れない場所が複数あるマップ

図 3.17，3.18，3.19，3.20，3.21，3.22 は疑似ラプラシアンを用いずに $\gamma = 0$ で，通れない場所が複数箇所あるマップを動くエージェントの様子である．図 3.17 は初期段階の様子，図 3.18 は追跡エージェントに包囲された様子，図 3.19 は追跡エージェントの包囲から脱却した様子，図 3.20 は 2 回目の追跡エージェントに包囲された様子，図 3.21 は 2 回目の追跡エージェントの包囲から脱却した様子，図 3.22 は目的地到着の様子である．図 3.18，3.20 のように追跡エージェントに囲まれた場合，この場合も先ほどの通れない場所が少ないマップと同様に図 3.19，3.21 のように追跡エージェントを横切って目的地に向かった．

一方，図 3.23，3.24，3.25，3.26，3.27，3.28 は疑似ラプラシアンを用いて $\gamma = 0.3$ で動くエージェントの様子である．図 3.23 は初期段階の様子，図 3.24 は追跡エージェントに包囲された様子，図 3.25 は追跡エージェントの包囲から脱却した様子，図 3.26 は 2 回目の追跡エージェントに包囲された様子，図 3.39 は 2 回目の追跡エージェントの包囲から脱却した様子，図 3.28 は目的地到着の様子である．図 3.24，3.26 のように追跡エージェントに囲まれた場合，疑似ラプラシアン

を用いない場合と同様に図 3.25，3.27 のように追跡エージェントを横切って目的地に向かった．

表 3.3 は目的地到着までの経過時間と逃亡エージェントが攻撃を受けた回数を記録したものである．疑似ラプラシアンを用いた場合の方が用いない場合よりも到着にかかった時間は 3.5 秒短いという結果になったが，攻撃を受けた回数は 2 回多いという結果になった．このことから，逃亡エージェントが自身の体力よりも目的地到着を優先したことが分かった．

表 3.3　目的地到着までの経過時間と攻撃を受けた回数（通れない場所が複数あるマップ）

疑似ラプラシアンの有無	目的地到着までの経過時間	攻撃を受けた回数
疑似ラプラシアンなし（$\gamma = 0$）	22.4 秒	4 回
疑似ラプラシアンあり（$\gamma = 0.3$）	18.9 秒	6 回

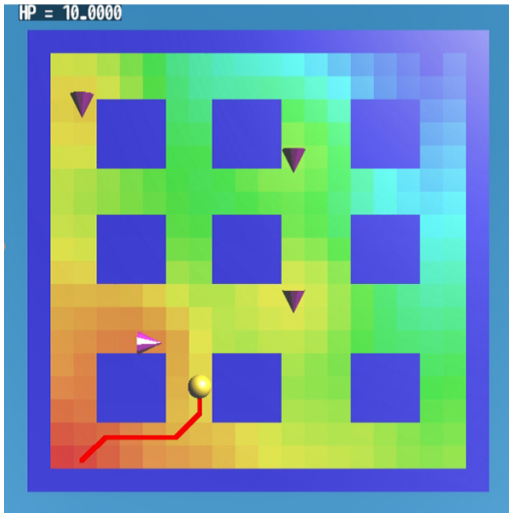

図 3.17　$\gamma = 0$ で動く様子（初期段階）

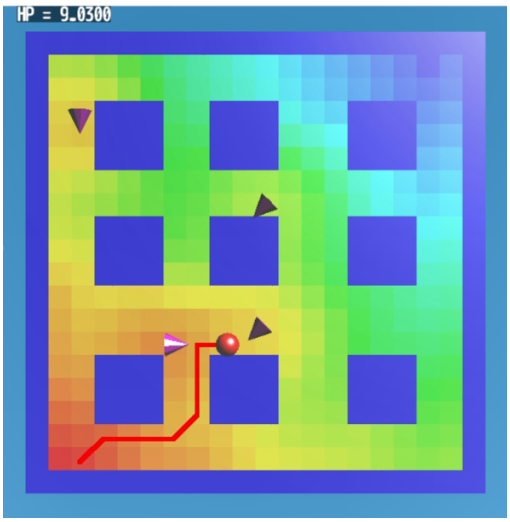

図 3.18　$\gamma = 0$ で動く様子（追跡エージェントに包囲された状態）

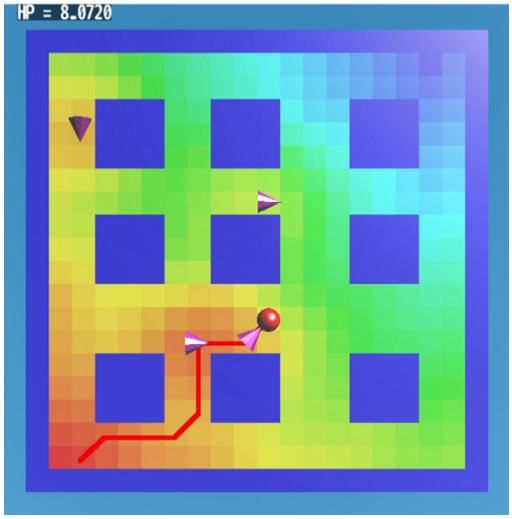

図 3.19　$\gamma = 0$ で動く様子（追跡エージェントの包囲から脱却）

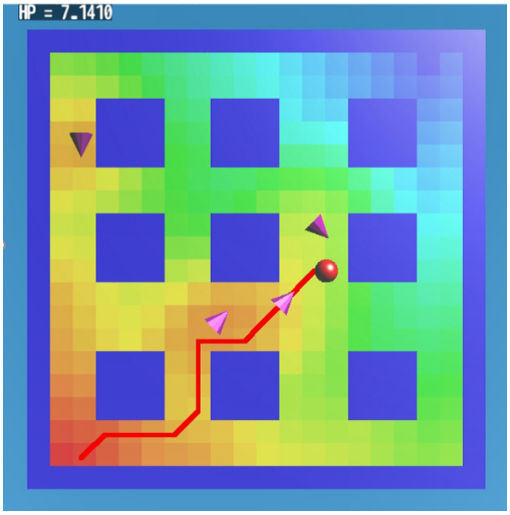

図 3.20　$\gamma = 0$ で動く様子（2 回目の追跡エージェントに包囲された状態）

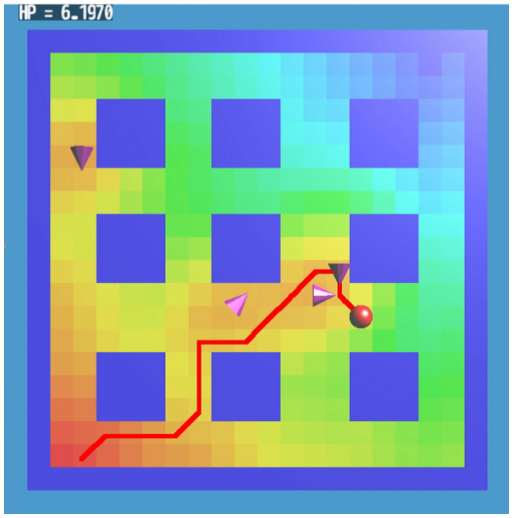

図 3.21　$\gamma = 0$ で動く様子（2 回目の追跡エージェントの包囲からの脱却）

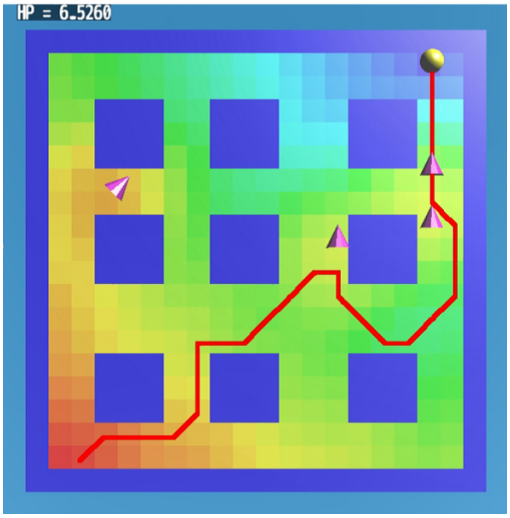

図 3.22　$\gamma = 0$ で動く様子（目的地到着）

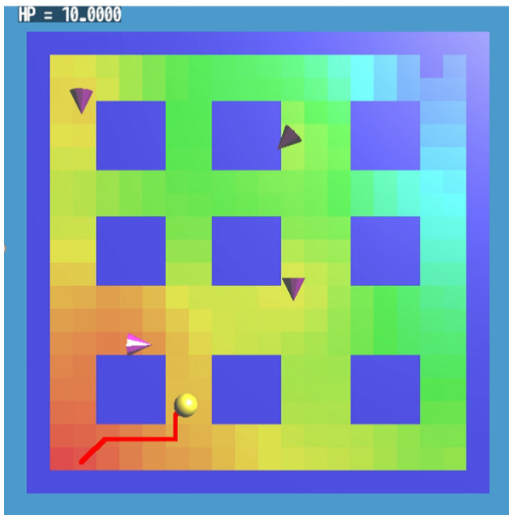

図 3.23　$\gamma = 0.3$ で動く様子（初期段階）

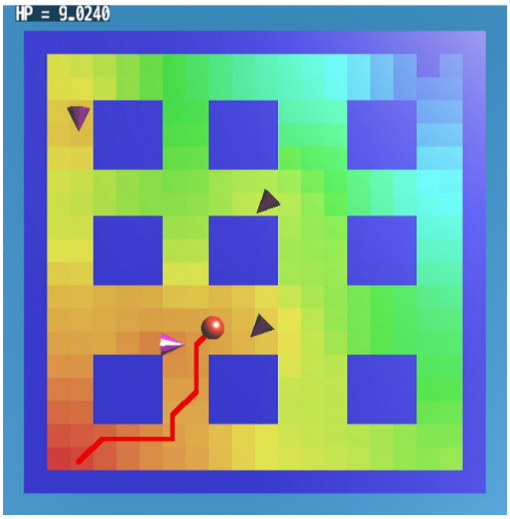

図 3.24　$\gamma = 0.3$ で動く様子（追跡エージェントに包囲された状態）

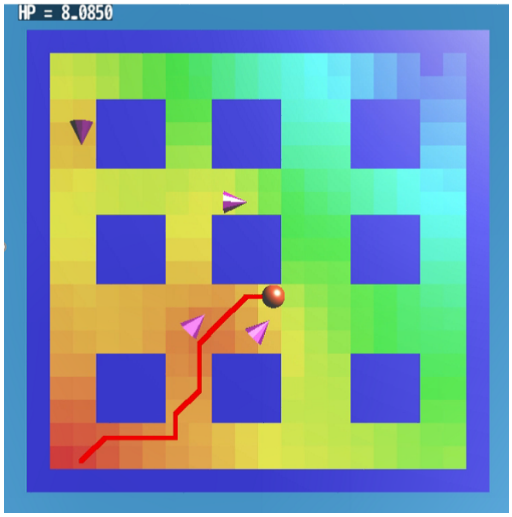

図 3.25　$\gamma = 0.3$ で動く様子（追跡エージェントの包囲から脱却）

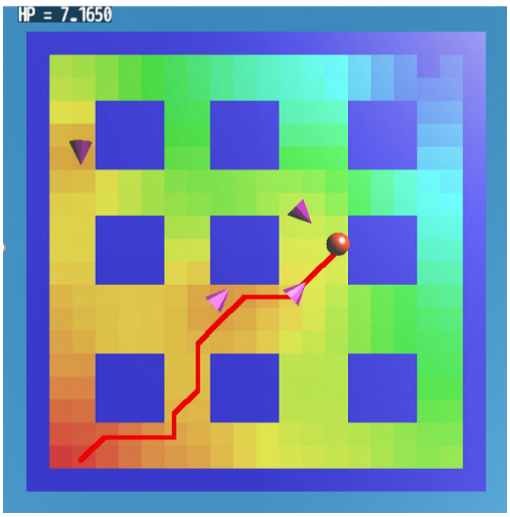

図 3.26　$\gamma = 0.3$ で動く様子（2 回目の追跡エージェントに包囲された状態）

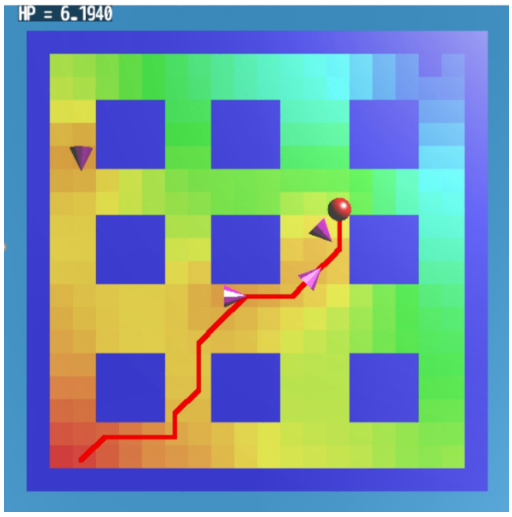

図 3.27　$\gamma = 0.3$ で動く様子（2 回目の追跡エージェントの包囲からの脱却）

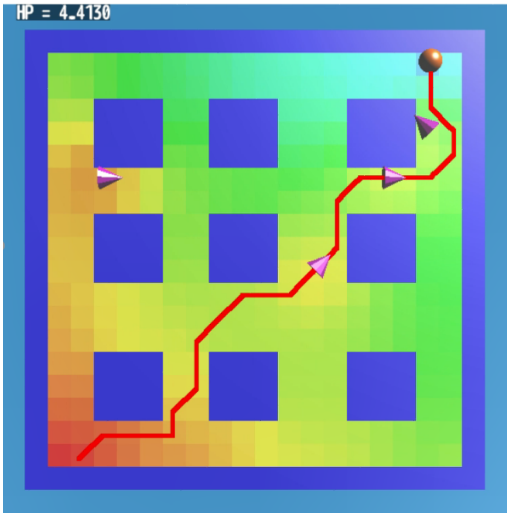

図 3.28　$\gamma = 0.3$ で動く様子（目的地到着）

3.1.4　通れない場所が細長いマップ

図 3.29，3.30，3.31，3.32，3.33，3.34 は疑似ラプラシアンを用いずに $\gamma = 0$ で，通れない場所が細長いマップを動くエージェントの様子である．図 3.29 は追跡エージェントが近くにいるときの様子，図 3.30 は追跡エージェントを横切ったときの様子，図 3.31 は追跡エージェントに包囲された様子，図 3.32 は追跡エージェントに追い込まれた様子，図 3.33 は追跡エージェントの包囲から脱却した様子，図 3.34 は目的地到着の様子である．図 3.31，3.32 のように追跡エージェントに囲まれた場合，追跡エージェントを避けるあまり袋小路の状態に陥ってしまったが，図 3.33 のようにある段階で追跡エージェントの攻撃を受けつつも横切って目的地に向かった．

一方，図 3.35，3.36，3.37，3.38，3.39，3.40 は疑似ラプラシアンを用いて $\gamma = 0.3$ で動くエージェントの様子である．図 3.35 は追跡エージェントが近くにいるときの様子，図 3.36 は追跡エージェントを横切ったときの様子，図 3.37 は追跡エージェントに包囲された様子，図 3.38 は追跡エージェントの包囲から脱却した様子，図 3.39 は 2 回目の追跡エージェントに包囲された様子，図 3.40 は 2 回目の追跡エージェントの包囲からの脱却と目的地到着の様子である．図 3.37，3.39

のように追跡エージェントに囲まれた場合，追跡エージェントを避けるあまり最初は袋小路の状態に陥ってしまったが，図 3.38，3.40 のように早い段階で追跡エージェントの攻撃を受けつつも横切って目的地に向かった．

表 3.4 は目的地到着までの経過時間と逃亡エージェントが攻撃を受けた回数を記録したものである．疑似ラプラシアンを用いた場合の方が用いない場合よりも到着にかかった時間は 1.8 秒短いという結果になり，攻撃を受けた回数も 1 回少ないという結果になった．

表 3.4　目的地到着までの経過時間と攻撃を受けた回数（通れない場所が細長いマップ）

疑似ラプラシアンの有無	目的地到着までの経過時間	攻撃を受けた回数
疑似ラプラシアンなし（$\gamma = 0$）	34.5 秒	7 回
疑似ラプラシアンあり（$\gamma = 0.3$）	32.7 秒	6 回

図 3.29　$\gamma = 0$ で動く様子（追跡エージェントが近くにいる状態）

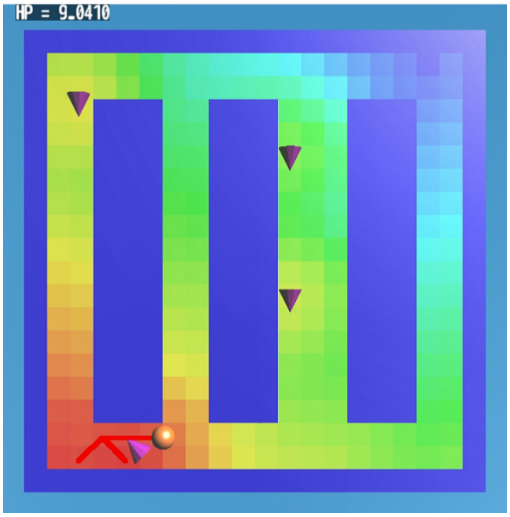

図 3.30　$\gamma = 0$ で動く様子（追跡エージェントを横切る様子）

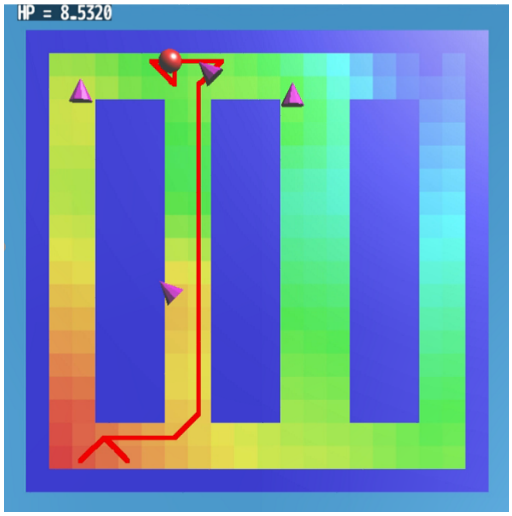

図 3.31　$\gamma = 0$ で動く様子（追跡エージェントに包囲された状態）

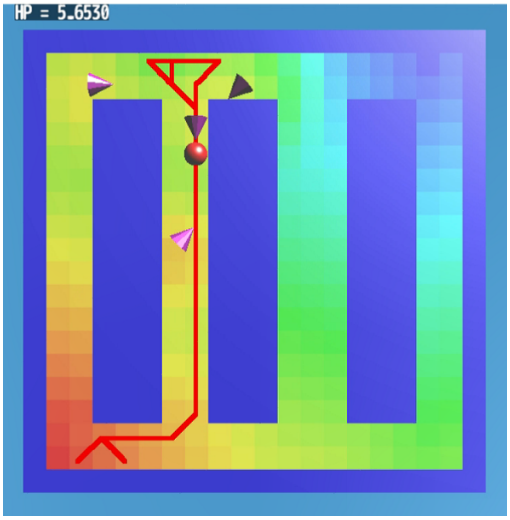

図 3.32　$\gamma = 0$ で動く様子（追跡エージェントに追い込まれた状態）

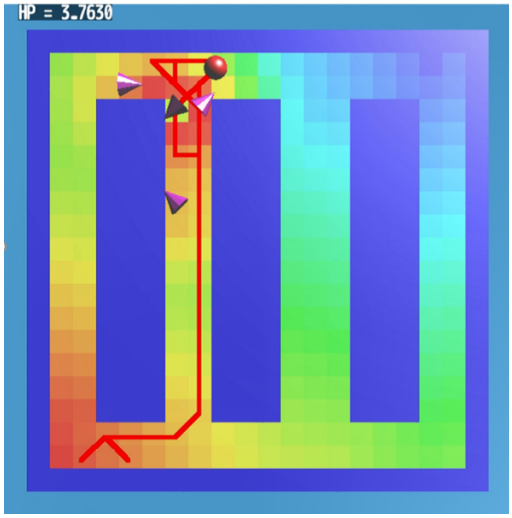

図 3.33　$\gamma = 0$ で動く様子（追跡エージェントの包囲から脱却）

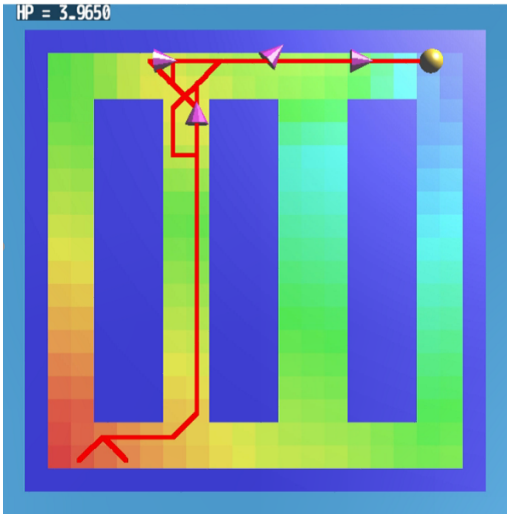

図 3.34　$\gamma = 0$ で動く様子（目的地到着）

図 3.35　$\gamma = 0.3$ で動く様子（追跡エージェントが近くにいる状態）

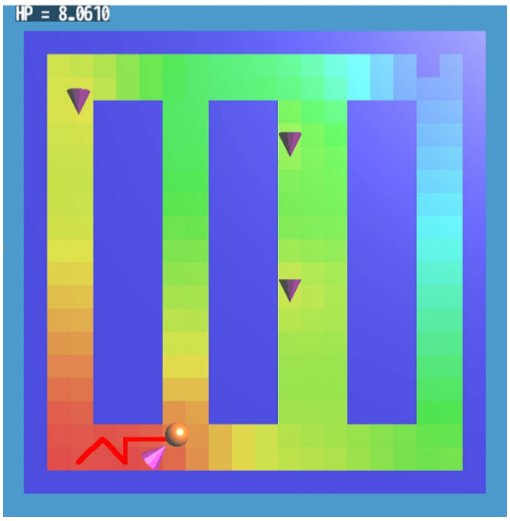

図 3.36　$\gamma = 0.3$ で動く様子（追跡エージェントを横切る様子）

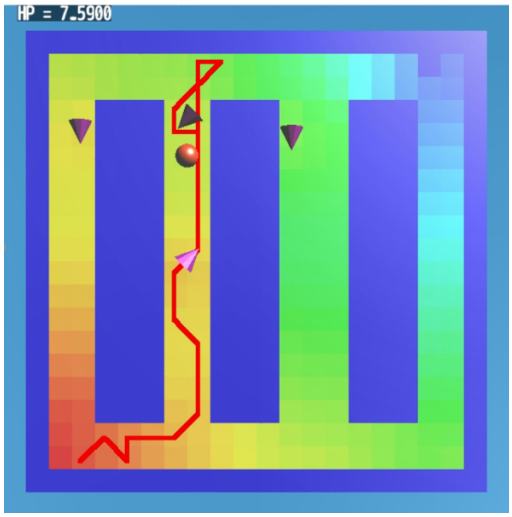

図 3.37　$\gamma = 0.3$ で動く様子（追跡エージェントに包囲された状態）

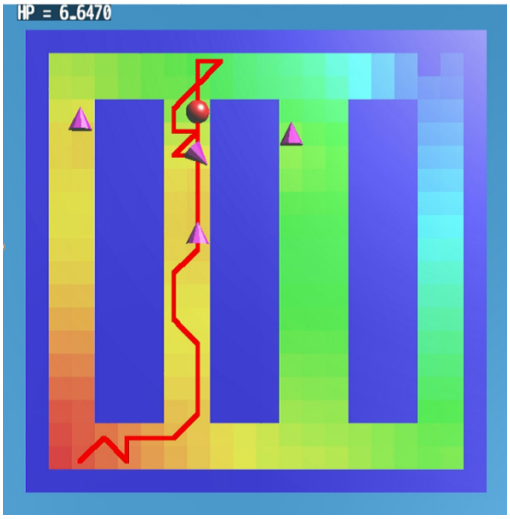

図 3.38　$\gamma = 0.3$ で動く様子（追跡エージェントの包囲から脱却）

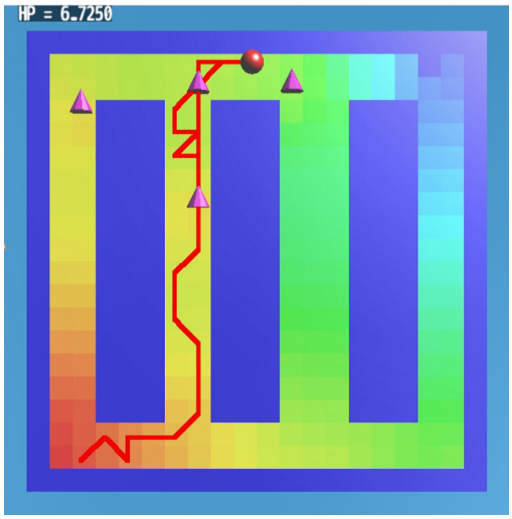

図 3.39　$\gamma = 0.3$ で動く様子（2 回目の追跡エージェントに包囲された状態）

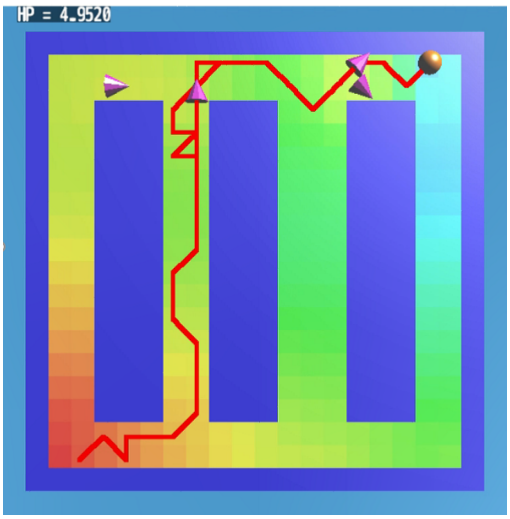

図 3.40　$\gamma = 0.3$ で動く様子（2 回目の追跡エージェントの包囲からの脱却，目的地到着）

3.1.5　複雑なマップ

図 3.41，3.42，3.43，3.44，3.45，3.46 は疑似ラプラシアンを用いずに $\gamma = 0$ で，複雑なマップを動くエージェントの様子である．図 3.41 は追跡エージェントを横切ったときの様子，図 3.42 は追跡エージェントに包囲された様子，図 3.43 は追跡エージェントに追い込まれた様子，図 3.44 は目的地到着が困難な様子，図 3.45 は初期地点付近まで戻った様子，図 3.46 は目的地到着が達成できない様子である．図 3.42，3.43 のように追跡エージェントに囲まれた場合，追跡エージェントを避けるあまり袋小路の状態に陥ってしまって，図 3.45 のように初期地点付近まで戻ってしまった．また，図 3.46 のように同じ場所を右往左往して目的地到着が達成できなかった．

一方，図 3.47，3.48，3.49，3.50 は疑似ラプラシアンを用いて $\gamma = 0.3$ で動くエージェントの様子である．図 3.47 は追跡エージェントを横切ったときの様子，図 3.48 は追跡エージェントに包囲された様子，図 3.49 は追跡エージェントの包囲から脱却した様子，図 3.50 は目的地に到着した様子である．図 3.48 のように追跡エージェントに囲まれた場合少し右往左往したものの，図 3.49 のように早い段階で追跡エージェントの攻撃を受けつつも横切って目的地に向かった．

表 3.5 は目的地到着までの経過時間と逃亡エージェントが攻撃を受けた回数を記録したものである．疑似ラプラシアンを用いない場合は目的地に到着できなかったが，用いた場合は到着できて経過時間も短かった．また，疑似ラプラシアンを用いた場合の方が用いない場合よりも攻撃を受けた回数が 5 回少なかった．

表 3.5　目的地到着までの経過時間と攻撃を受けた回数（複雑なマップ）

疑似ラプラシアンの有無	目的地到着までの経過時間	攻撃を受けた回数
疑似ラプラシアンなし（$\gamma = 0$）	目的地到着できず	9 回
疑似ラプラシアンあり（$\gamma = 0.3$）	29.6 秒	4 回

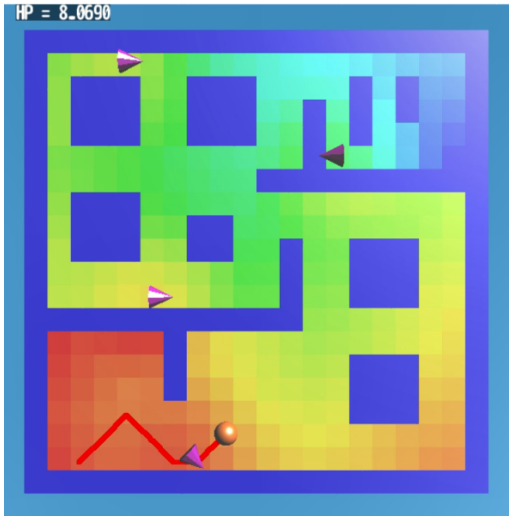

図 3.41　$\gamma = 0$ で動く様子（追跡エージェントを横切る様子）

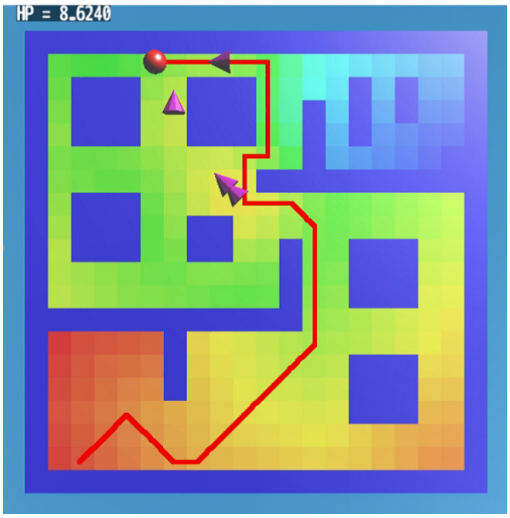

図 3.42　$\gamma = 0$ で動く様子（追跡エージェントに包囲された状態）

図 3.43　$\gamma = 0$ で動く様子（追跡エージェントに追い込まれた状態）

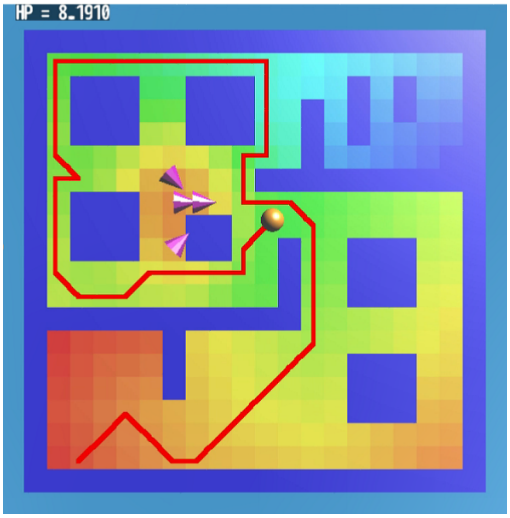

図 3.44　$\gamma = 0$ で動く様子（目的地到着が困難な状態）

図 3.45　$\gamma = 0$ で動く様子（初期地点付近まで戻った様子）

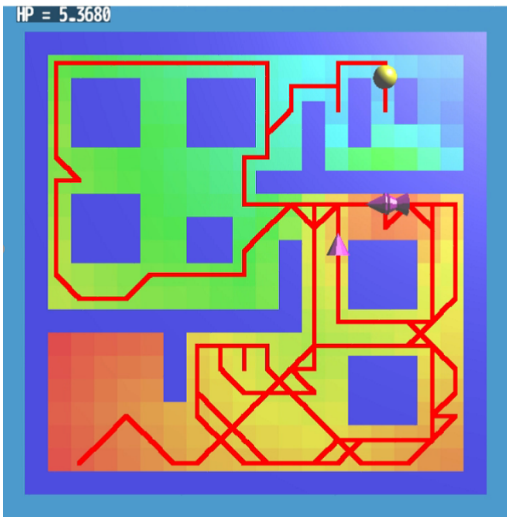

図 3.46　$\gamma = 0$ で動く様子（目的地に到着できない様子）

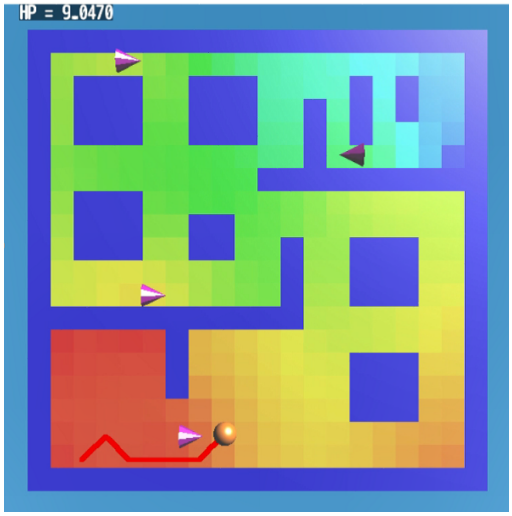

図 3.47　$\gamma = 0.3$ で動く様子（追跡エージェントを横切る様子）

図 3.48　$\gamma = 0.3$ で動く様子（追跡エージェントに包囲された状態）

図 3.49　$\gamma = 0.3$ で動く様子（追跡エージェントの包囲から脱却）

図 3.50　$\gamma = 0.3$ で動く様子（目的地到着）

3.2　プレイヤー操作による評価実験

追跡エージェント 1 体を自分で操作できるようにして黄色の円錐にし，複雑なマップで検証を行った．

図 3.51，3.52，3.53，3.54，3.55，3.56 は疑似ラプラシアンを用いずに $\gamma = 0$ で複雑なマップ

を動くエージェントの様子である．図 3.51 は追跡エージェントに包囲された様子，図 3.52 は追跡エージェントの包囲から脱却した様子，図 3.53 は初期地点付近まで追い込まれた様子，図 3.54 は追い込まれた状態から脱却した様子，図 3.55 は追跡エージェントに追い込まれた様子，図 3.56 は目的地到着が達成できない様子である．図 3.51，3.53 のように追跡エージェントに包囲された場合，図 3.52，3.54 のように右往左往はするもののある段階で追跡エージェントを横切って目的地に向かおうとしたが，図 3.53 のように初期地点付近まで戻ってしまった．また，図 3.55 のように何度も追跡エージェントに追い込まれてしまい，図 3.56 のように再び初期地点付近まで戻ってしまい，目的地到着は達成できなかった．

図 3.57，3.58，3.59，3.60，3.61，3.62，3.63，3.64 は疑似ラプラシアンを用いて $\gamma = 0.3$ で複雑なマップを動くエージェントの様子である．図 3.57 は追跡エージェントに包囲された様子，図 3.58 は追跡エージェントの包囲から脱却した様子，図 3.59 は追跡エージェントに追い込まれた様子，図 3.60 は初期地点付近まで追い込まれた様子，図 3.61 は追い込まれた状態から脱却した様子，図 3.62 は 2 回目の追跡エージェントに包囲された様子，図 3.63 は 2 回目の追跡エージェントの包囲から脱却した様子，図 3.64 は目的地に到着した様子である．図 3.57，3.60，3.62 のように追跡エージェントに包囲された場合，疑似ラプラシアンを用いなかった場合と同様に，図 3.58，3.61，3.63 のように同じ場所で右往左往はするものの，ある段階で追跡エージェントを横切り目的地に向かった．また，疑似ラプラシアンを用いなかった場合と同様に図 3.60 のように初期地点付近まで戻ってしまった．しかし，図 3.64 のように時間はかかったものの目的地に到着した．

表 3.6 は目的地到着までの経過時間と逃亡エージェントが攻撃を受けた回数を記録したものである．疑似ラプラシアンを用いない場合は目的地に到着できなかったが，用いた場合は時間はかかったものの到着できた．また，疑似ラプラシアンを用いた場合の方が用いない場合よりも攻撃を受けた回数が 2 回少なかった．以上の結果から，疑似ラプラシアンを用いた方が用いない場合よりも包囲状態からの脱却が迅速で，判断が早いという印象を持った．

このように，疑似ラプラシアンを用いることで追跡エージェントに包囲される状態から迅速に脱却し，最終的な目標を達成することを確認できた．

表 3.6　目的地到着までの経過時間と攻撃を受けた回数

疑似ラプラシアンの有無	目的地到着までの経過時間	攻撃を受けた回数
疑似ラプラシアンなし（$\gamma = 0$）	目的地到着できず	21 回
疑似ラプラシアンあり（$\gamma = 0.3$）	119.7 秒	19 回

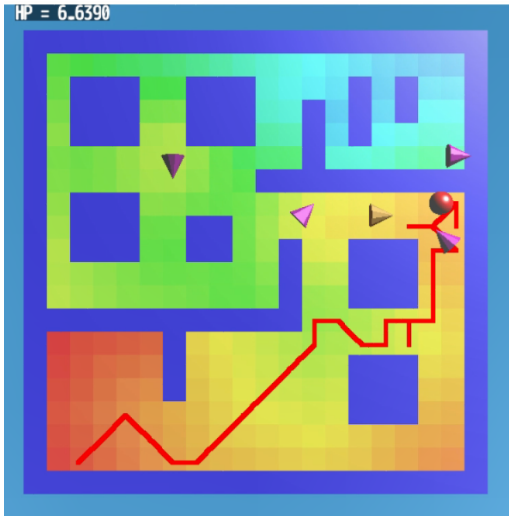

図 3.51　$\gamma = 0$ で動く様子（追跡エージェントに包囲された状態）

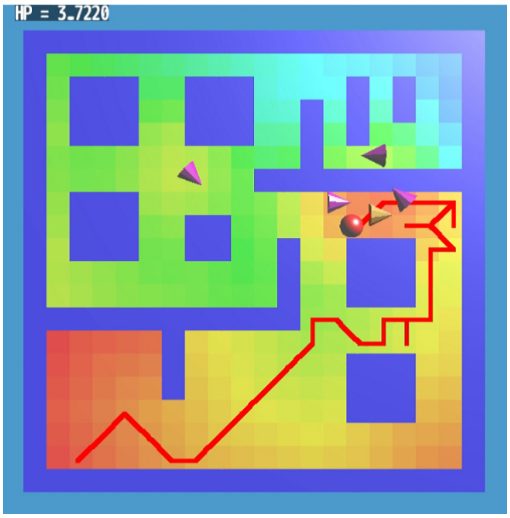

図 3.52　$\gamma = 0$ で動く様子（追跡エージェントの包囲から脱却）

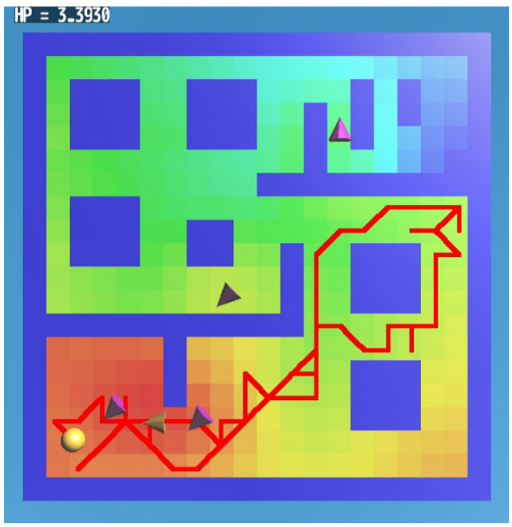

図 3.53　$\gamma = 0$ で動く様子（初期地点付近まで追い込まれた状態）

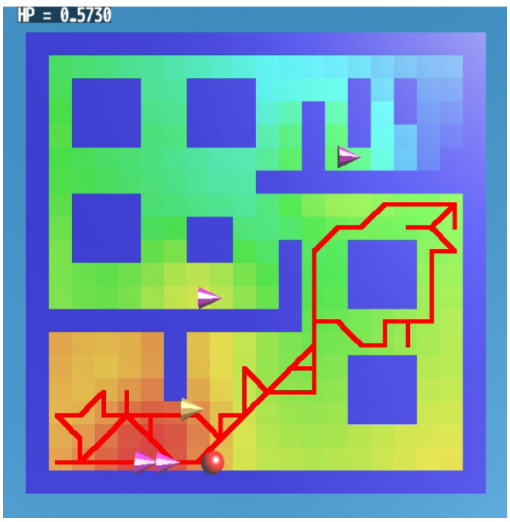

図 3.54　$\gamma = 0$ で動く様子（追い込まれた状態から脱却）

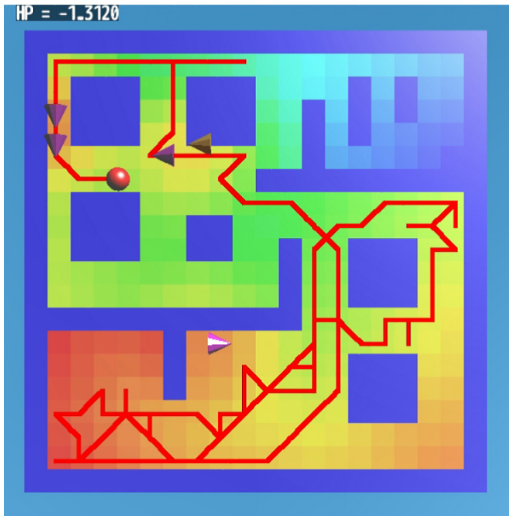

図 3.55　$\gamma = 0$ で動く様子（追跡エージェントに追い込まれた状態）

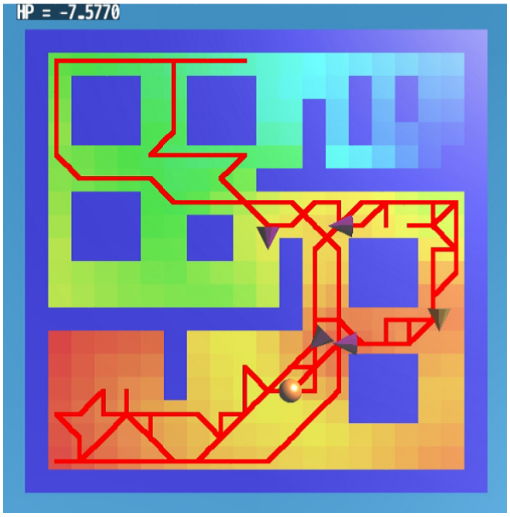

図 3.56　$\gamma = 0$ で動く様子（目的地に到着できない様子）

図 3.57　$\gamma = 0.3$ で動く様子（追跡エージェントに包囲された状態）

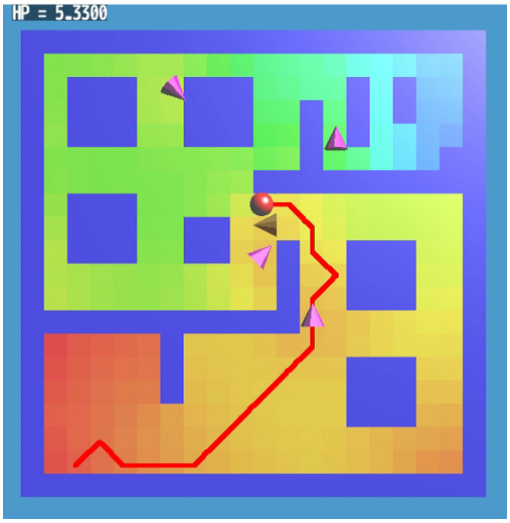

図 3.58　$\gamma = 0.3$ で動く様子（追跡エージェントの包囲からの脱却）

図 3.59　$\gamma = 0.3$ で動く様子（追跡エージェントに追い込まれた状態）

図 3.60　$\gamma = 0.3$ で動く様子（初期地点付近まで追い込まれた状態）

図 3.61　$\gamma = 0.3$ で動く様子（追い込まれた状態からの脱却）

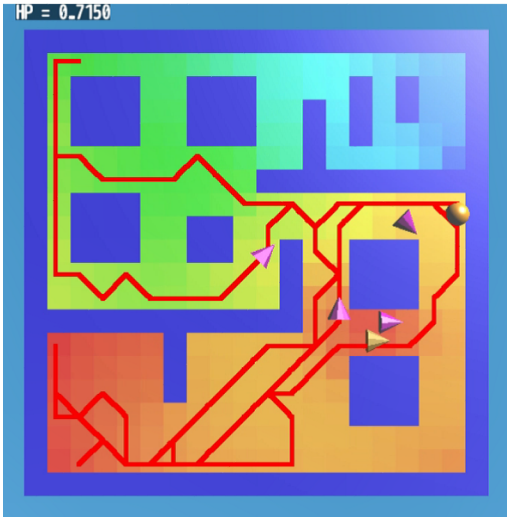

図 3.62　$\gamma = 0.3$ で動く様子（2 回目の追跡エージェントに包囲された状態）

図 3.63　$\gamma = 0.3$ で動く様子（2 回目の追跡エージェントの包囲からの脱却）

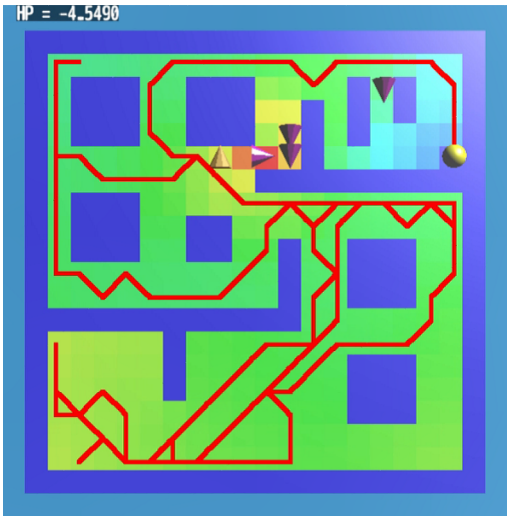

図 3.64　$\gamma = 0.3$ で動く様子（目的地到着）

第 4 章

まとめ

本研究は，ユーティリティベース AI の中のゴールベースと非ゴールベースを足し合わせて両方の考慮を行った場合，ゴールベースの特性である推移律が失われて最終的な目標達成が保証できなくなるという問題に着目した．本研究では，疑似ラプラシアンを用いた盛り土関数を使って推移律を保ち，最終的な目標達成を保証する手法を提案した．また，盛り土関数の調整係数は AI の体力や停滞度の数値を使用しファジー推論を用いて動的に変更するようにした．実装方法は，ゴールベース指標にダイクストラ法によるコストマップを使用し，非ゴールベース指標に追跡エージェントからの危険度コストを使用した．以上により算出されたコスト値に従って，コストが低い方へ AI は動いていくようにした．通れない場所が 1 箇所もないマップや複雑なマップなど複数のマップで検証した結果，疑似ラプラシアンを用いることで局所的状況に対応しつつ最終的な目標をできるだけ迅速に達成することが確認できた．

しかし，追跡エージェントの間を通った方がより迅速に最終的な目標を達成できるがそれをしない場面や，同じ場所で右往左往してばかりで体力が多いのにも関わらず追跡エージェントの包囲からなかなか脱却しない場面が見受けられた．そこで，盛り土関数の適用場所や調整係数の算出方法を改善することで，追跡エージェントを避けつつより迅速で効率的に最終的な目標を達成する AI を作成できるのではないかと考える．

謝辞

本研究を進めるにあたり，多くのアドバイスや指導をしてくださり，相談に乗ってくれた渡辺先生，阿部先生には大変お世話になりました．話を聞いてくれたり気にかけてくださり，多くの時間ご一緒させていただき助けていただきました．本当に心より感謝いたします．

また，合同ミーティングで発表を聞いていただき，アドバイスや指導をしてくださった柿本先生，三上先生にも大変お世話になりました．時には私の研究に対する考えもあって白熱した議論をすることもありました．先生方にはご迷惑をおかけしてしまいましたが，意見を交わすことが好きな私にとってはとても有意義な時間でした．心より感謝いたします．

さらに，本研究は嬉しいことに複数の学会で発表させていただきました．拙い発表ではあったと思いますが，私の発表を真剣に聞いてくださり貴重な意見やアドバイスをくださった多くの先生方や企業の方々に深く感謝いたします．時には，発表後の私に駆け寄ってくださり，「あなたの研究はすごいよ」と褒めてくださった方もいました．そのようなことを言ってもらえるとは思っておらず，研究をしていく中でそのお言葉に何度も救われてきました．ありがとうございました．皆様にいただいた意見やアドバイス，お言葉がなかったら私の研究は出来上がっていません．心より感謝いたします．

最後に，一緒に学会に参加して切磋琢磨したり，相談や話を聞いてくれたり応援してくれるなど辛い時期を支えてくれた友人達や後輩，両親に深く感謝いたします．

研究をしていく中で悩んだり，モチベーションが上がらずもうやめたいと思う日々もありました．しかし，このように最後まで研究をやり遂げたのは私を支えてくれた全ての方々のおかげです．感謝してもしきれません．

本当にありがとうございました．

参考文献

[1] 三宅陽一郎. ゲーム AI 技術入門. 技術評論社, 2019. ISBN: 978-429710828-1.

[2] 三宅陽一郎. ディジタルゲームにおける人工知能技術の応用の現在. 人工知能学会論文誌, Vol. 30, No. 1, pp. 45 – 64, 2015.

[3] 佐藤直之, Sila Temsiririrkkul, Luong Huu Phuc, 池田心. Influence map を用いた経路探索による人間らしい弾避けのシューティングゲーム AI プレイヤ. ゲームプログラミングワークショップ 2016 論文集, pp. 57 – 64, 2016.

[4] 平田佑也, 稲葉通将, 高橋健一, 鳥海不二夫, 大澤博隆, 片上大輔, 篠田孝祐. プレイログから獲得した行動選択確率を用いた人狼ゲームのシミュレーション. 人工知能学会全国大会（JSAI2015）論文集, pp. 1 – 4, 2015.

[5] 隅山淳一朗, 橋山智訓, 田野俊一. ぷよぷよにおける人間のプレイデータの特徴量抽出. 第 31 回ファジィシステムシンポジウム, pp. 2 – 4, 2015.

[6] 張輝陽, 星野准一. プレイヤ行動の模倣に基づく AI キャラクタ行動ルールの自動生成. 情報処理学会研究報告・ゲーム情報学, Vol. 31, pp. 1 – 4, 2014.

[7] 杵渕哲彦, 伊藤毅志. 手の流れを考慮して自然な手を選ぶ将棋 AI の試作. 情報処理学会研究報告・ゲーム情報学, Vol. 33, No. 12, pp. 1 – 8, 2015.

[8] Shi Yuan, Fan Tianwen, Li Wanxiang, 池田心. 深層学習囲碁プログラムを用いた場合の手加減に関する研究. 情報処理学会研究報告, Vol. 41, No. 9, pp. 1 – 8, 2019.

[9] 藤井叙人, 佐藤祐一, 若間弘典, 風井浩志, 片寄晴弘. 生物学的制約の導入によるビデオエージェントの「人間らしい」振舞いの自動獲得. 情報処理学会論文誌, Vol. 55, pp. 1655 – 1664, 2014.

[10] 星野准一, 田中彰人, 濱名克季. 模倣学習により成長する格闘ゲームキャラクタ. 情報処理学会論文誌, Vol. 49, pp. 2539 – 2548, 2008.

[11] 渡辺大地, 西川孝亮. 波動方程式による波伝播作用を利用した追跡行動アルゴリズム.

NICOGRAPH2019, pp. 91 – 98, 2019.

[12] まなべ, 小山太輔. 面白さの評価関数は作れるか？麻雀対局中の思考を真面目に再現したらゲーム ai になっていた – ゲームアーツ創設者宮路洋一氏が説く試行錯誤の大切さ、そして 80 年代【聞き手：三宅陽一郎】. http://news.denfaminicogamer.jp/interview/181024. 電ファミコンゲーマー, 2018, 参照：2020/08/05.

[13] 仲道隆史, 伊藤毅志. 人を楽しませる接待将棋システム. 第 28 回人工知能学会全国大会, pp. 1 – 2, 2014.

[14] 大森翔太朗, 金子知適. 将棋における棋譜から棋風を学習するための研究. 情報処理学会論文誌, Vol. 57, No. 11, pp. 2374 – 2381, 11 2016.

[15] Jeff Orkin. Three states and a plan:the AI of F.E.A.R. http://alumni.media.mit.edu/~jorkin/. GDC, 2006, 参照：2020/08/05.

[16] 安藤毅. 「サカつく」のサッカー試合 AI システム. https://cedil.cesa.or.jp/cedil_sessions/view/379. CEDEC, 2010, 参照：2020/11/24.

[17] Restuadi Studiawan, Mochamad Hariadi, and Surya Sumpeno. Tactical planning in space game using goal-oriented action planning. *JAREE (Journal on Advanced Research in Electrical Engineering)*, Vol. 2, pp. 5 – 9, 05 2018.

[18] 古川真帆, 阿部雅樹, 渡辺大地. ゲーム AI における評価関数による最終目標達成保証と局所問題回避の両立の実現. *NICOGRAPH2020*, pp. 87 – 92, 2020.

[19] 茨木俊秀. 最適化の数学. 共立出版, 2011. ISBN: 978-4320015654.

[20] 山崎昶. 法則の辞典. 朝倉書店, 2006. ISBN: 978-4254101973.

[21] Peter-Pike Sloan, Jan Kautz, and John Shyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Trans and SIGGRAPH*, Vol. 21, No. 3, pp. 527 – 536, 07 2002.

[22] Bruno Vallet and Bruno Lévy. Spectral geometry processing with manifold harmonics. *Computer Graphics Forum and Wiley*, Vol. 27, No. 2, pp. 251 – 260, 2008.

[23] Rhaleb Zayer, Christian Rössl, Zachi Karni, Hans ‐ PeterSeidel. Harmonic guidance for surface deformation. *Euro Graphics*, Vol. 24, No. 3, 2005.

[24] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. *ACM Trans and SIGGRAPH*, Vol. 22, No. 3, pp. 313 – 318, 2003.

[25] 廣田薫. ファジィ推論エキスパートシステムの現状と動向. 情報処理, Vol. 28, No. 8, pp. 1065 – 1074, 1987.

[26] L.A.Zadeh. Fuzzy sets. *Inf.Control*, Vol. 8, pp. 338 – 353, 1965.

[27] E.H. Mamdani and S. Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, Vol. 7, No. 1, pp. 1–13, 1975.

[28] David M. Bourg, GlennSeemann 著, 株式会社クイープ訳. ゲーム開発者のための AI 入門. オライリー・ジャパン, 2005. ISBN: 4-87311-216-8.

[29] MatBuckland 著, 松田晃一訳. 実例で学ぶゲーム AI プログラミング. オライリー・ジャパン, 2007. ISBN: 978-4-87311-339-8.

[30] B. コルテ, J. フィーゲン. 組合せ最適化. シュプリンガー・ジャパン, 2009. ISBN: 978-4431100218.

発表業績

口頭発表 (査読付)

1. 古川真帆, 阿部雅樹, 渡辺大地, バトルロイヤルゲームの練習モードにおける人間操作を模倣するキャラクター AI に関する研究, NICOGRAPH2019(ショートペーパー採録), 2019.
2. 古川真帆, 阿部雅樹, 渡辺大地, ゲーム AI における評価関数による最終目標達成保証と局所問題回避の両立の実現, NICOGRAPH2020(フルペーパー採録), 2020.

ポスター発表

1. 古川真帆, 渡辺大地, ガンシューティングゲームにおける正面対峙時の人間による操作の模倣行動に関する研究, NICOGRAPH2018, 2018.
2. 古川真帆, 阿部雅樹, 渡辺大地, ゲーム AI における長期的目標達成と局所的問題対処の両立の実現, 映像表現・芸術科学フォーラム 2020, 2020.

受賞歴

1. 第 19 回ビジュアル情報処理研究合宿 ポスター発表優秀賞, 論文題目「バトルロイヤルゲームの練習モードにおける人間操作を模倣するキャラクター AI に関する研究」, 2019.

付録 A

ファジー推論

ファジー推論とは，複数の入力からファジー値を算出する手段であり，最小最大法，代数積加算重心法，簡略化推論法など様々なものがある．本研究では，このうち代数積加算重心法を用いた．本章では，この手法の計算方法について述べる．なお，ここでは入力値を 2 つのスカラー値と想定する．

ファジー推論においては，複数のメンバーシップ関数を事前に規定する必要がある．典型的なメンバーシップ関数は，Bad(B)，Normal(N)，Good(G) の 3 つの関数を用いるものであり，それぞれ以下のような式となる．

$$B(x) = \begin{cases} -2x+1, & (0 \leqq x < \frac{1}{2}) \\ 0 & (\frac{1}{2} \leqq x \leqq 1) \end{cases} \tag{A.1}$$

$$N(x) = \begin{cases} 2x, & (0 \leqq x < \frac{1}{2}) \\ -2x+2 & (\frac{1}{2} \leqq x \leqq 1) \end{cases} \tag{A.2}$$

$$G(x) = \begin{cases} 0, & (0 \leqq x < \frac{1}{2}) \\ 2x-1 & (\frac{1}{2} \leqq x \leqq 1) \end{cases} \tag{A.3}$$

これらの関数をグラフとして表したものが図 A.1 である．

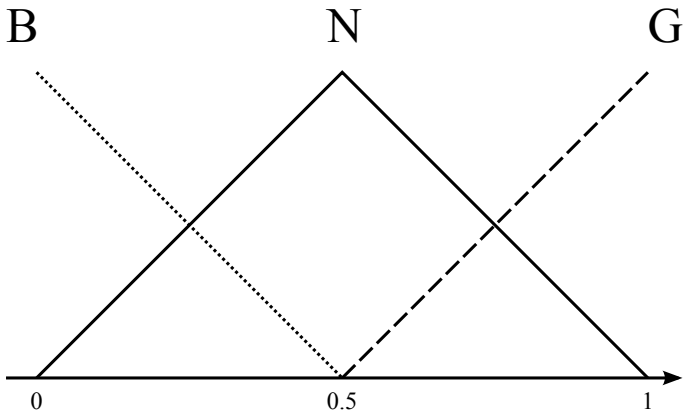

図 A.1　メンバーシップ関数

例えば，$x = 0.5$ のとき，$B(x) = 0, N(x) = 1, G(x) = 0$ となり，$x = 0.2$ のときは $B(x) = 0.6, N(x) = 0.4, G(x) = 0$ となる．

以降，入力値 a, b をそれぞれ 0.7 と 0.6 と想定して解説する．このとき，

$$\begin{cases} B(a) = 0 \\ N(a) = 0.6 \\ G(a) = 0.4 \end{cases} \tag{A.4}$$

$$\begin{cases} B(b) = 0 \\ N(b) = 0.8 \\ G(b) = 0.2 \end{cases} \tag{A.5}$$

と算出できる．

次に，各メンバーシップ関数における最小値を求める．これを B_m, N_m, G_m とすると，

$$\begin{cases} B_m = \min\left(B(a), B(b)\right) = 0 \\ N_m = \min\left(N(a), N(b)\right) = 0.6 \\ G_m = \min\left(G(a), G(b)\right) = 0.2 \end{cases} \tag{A.6}$$

となる．

次に，各メンバーシップ関数において先ほどの値 B_m, N_m, G_m より下側の部分の面積を求め総和を取る．つまり，

$$F(x) = \min\left(B(x), B_m\right) + \min\left(N(x), N_m\right) + \min\left(G(x), G_m\right) \tag{A.7}$$

として $F(x)$ を定義した上で，

$$S = \int_0^1 F(x)dx \tag{A.8}$$

を求める．図 A.2 は，上記領域を図示したものである．

最後に，全体のグラフの重心，つまり全面積を二等分する箇所を算出する．つまり，

$$\int_0^\alpha F(x)dx = \frac{S}{2} \tag{A.9}$$

となる α を求める．$(a, b) = (0.6, 0.7)$ のときに厳密解は $\frac{1}{5} + \frac{3}{20}\sqrt{6} \cong 0.5674$ である．図 A.3 に重心を求める模式図を示す．

実際の適用では，区分求積法などの近似値計算で問題ない．アルゴリズムは以下の通りである．

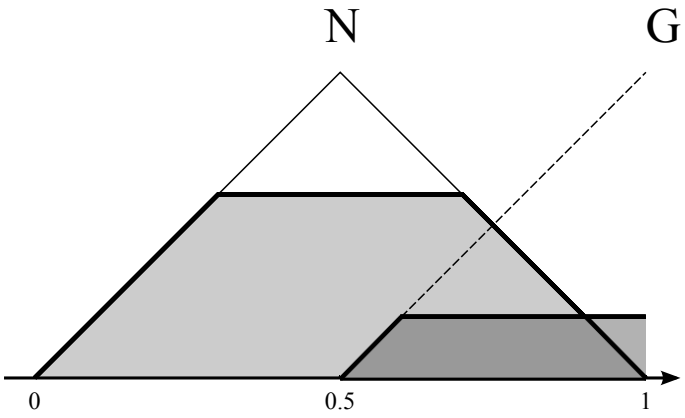

図 A.2　面積の算出

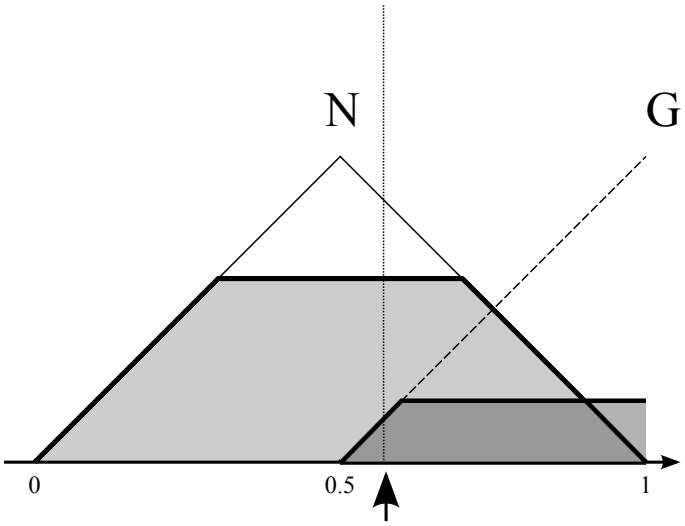

図 A.3　重心の算出

1. 各推論グラフの面積を求め，全部を合計する．
2. 合計値の半分を求める．
3. 値 0 から Δx ごとに順番に各推論グラフの面積を足していき，合計値の半分に達したら処理をやめる．