2021 年度　卒　業　論　文

危険を回避し動的に変化する目的地への移動を実現するキャラクター AI に関する研究

指導教員：渡辺 大地 教授

メディア学部 ゲームサイエンスプロジェクト
学籍番号　M0118219
平川 孟弥

2022 年 2 月

2021 年度　　卒　業　論　文　概　要

論文題目

危険を回避し動的に変化する目的地への移動を実現するキャラクター AI に関する研究

メディア学部 **学籍番号：M0118219**	**氏名**	平川 孟弥	**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

ゲーム AI において, 行動がもたらす効用から自身の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. ゴールベース指標を用いるゲーム作品では目標の変更が行われることがある．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．先行手法として, 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にゴールベースの特性が失われないために, 離散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, 目標の達成を保障する手法がある. しかしゴールベース指標の目標が変更された場合の検証は行われていない.

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, ゴールベースの目的が変更された場合でも, 先行手法を用いて目的の達成を保障が可能であることの検証を目的とする．結果としてゴールベースの目的が変更された場合では, 目的の達成を保障できることを確認したが, マップの形状によっては目的の達成に時間がかかるということが分かった.

目 次

図 目 次

第 1 章

はじめに

1.1 背景と目的

近年のゲームにおいて, ゲーム AI に関する研究が盛んに行われている [1]. ゲーム AI とは, デジタルゲーム内の人工知能のことである. ゲーム AI の分野の一つとして, キャラクター AI というプレイヤーの仲間や敵などのキャラクターの頭脳となり制御する人工知能がある．キャラクター AI に関する研究は, 多く行われている．佐藤ら [2] は弾幕シューティングゲームにおいて, 経路探索をベースにした Influencemap による被弾危険度見積もりと探索の諸工夫による手法によって,AI の操作が人間らしい印象に貢献することを確かめた．入倉ら [3] はサッカーゲームにおいて AI が守備を行う際にボールを保持した敵の行動を推測させた．隅山ら [4] はパズルゲームの『ぷよぷよ』の人間のプレイデータから定石形を自動抽出する方法で人間のプレイを再現する AI の作成を目指した．張ら [5] はプレイログを利用して, プレイヤ行動の模倣によって自動的に AI キャラクタの行動ルールを生成した．秋山ら [6] は AI に強化学習を用いてボール保持行動を獲得させた．森ら [7] は環境に応じた行動を自動的に選択できるキャラクタ AI の開発を目指した．キャラクター AI のうちの 1 つとしてユーティリティベース AI がある [8]．ユーティリティは効用を意味する．ユーティリティベース AI とは，自分の選択できる行動の効用を評価値として, あらかじめ設定した評価基準を元に算出し, 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．佐藤 [9] は味方 NPC に勝利への貢献度を評価基準として AI の難易度調整の簡易化を目指した．野津 [10] らは将棋 AI の評価項

目自動抽出の高速化を行った. 『クロムハウンズ』というアクションシューティングゲームでは, 敵 AI は相手との距離に応じた武器の効果を評価基準としている [11]. 『The Sims』という人生シミュレーションゲームでは, 空腹度や快適度などの数値を評価基準にしている [12]. 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. 桃太郎電鉄 [13] やマリオパーティ [14], ビリオンロード [15] のようなゲームでは, 目的地に到達した時やゲーム内のイベントによってゴールの位置が変化することがある. ゴールベース指標に関する研究は, 多く行われている. 丸山ら [16] は複数の目的地を効率よく巡回する観光のためのシステムを提案した. 大内ら [17] はサッカーゲームにおいてボールを持っていないプレイヤーの適切な移動先を決定する手法を提案した.

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. ゴールベースと非ゴールベースを足し合わせた場合, 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が, 古川ら [18] は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でも, 敵を避け続けて目的地に辿り着くことを保障することで効用指標による推移律崩壊が生じた際の復帰を実現した. しかし目的地に向かう途中で目的地が変更された場合の検証は行われていない. 本研究の目的は, 古川ら [18] の手法が目的地に向かう途中で目的地が変更された場合においても有用であることを確認することである. 古川ら [18] の手法を用いて, 目的地が変更された場合の目的地到達時間と敵から受けた攻撃回数を計測した. 結果として, 多くの場合では目的地到達時間が短くなり, 敵から攻撃を受けた回数が減ることが確認できたが, マップの形状によっては目的地到達時間が長くなることが分かった.

1.2　論文構成

本論文は全 4 章で構成する.2 章では先行研究の手法について説明する. また,3 章では検証と評価を行い,4 章では本研究のまとめを示す.

第 2 章

本研究の基本となる先行研究

ここでは本研究の先行研究となる古川ら [18] の研究について述べる.

2.1 ゲーム内エージェントの前提条件

先行研究では扱うマップおよびエージェントの想定を以下のように定める.

- 移動可能なエージェントとして 1 体の「逃亡エージェント」と, 逃亡エージェントを追跡する複数の「追跡エージェント」がある.
- マップ内には「目的地」が 1 箇所設定されており, 逃亡エージェントは, 追跡エージェントを回避しつつ目的地に到達することを目指す.
- エージェントが移動可能な領域は離散グラフによって表現されるものとする.
- エージェントの移動は離散的なものではなく, エージェントごとに設定されている時間経過と共に設定速度に従って移動する.
- マップ内のあるノード i に対し, 位置を P_i と表す.
- あるノード i の隣接ノード集合を $N(i)$, $N(i)$ の要素数を $|N(i)|$ と表す.

2.2 手法概要

先行研究では, 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 P_i におけるコスト評価関数 $T(P_i)$ が定義できるものとする. ここで,$T(P_i)$ 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. 目的地への移動距離によるコスト関数

を $C_g(P_i)$ とおく．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,$C_g(P_i)$ はゴールベース指標と言える．また, 時刻 t における敵からの距離によるコスト関数を $C_u(P_i)$ とおく.$C_u(P_i)$ も $C_g(P_i)$ と同様にスカラー場として捉えることができる.$C_u(P_i)$ 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.

ここで，両方の指標を考慮する場合, 時刻 t におけるコスト評価関数 $T(P_i)$ を以下のようにすることが考えられる.

$$T(P_i) = k_g C_g(P_i) + k_u C_u(P_i) \tag{2.1}$$

ここで k_g,k_u は調整係数である.k_g,k_u の初期値はユーザーが任意の数値に設定して調整を行う.k_u の数値は動的に調整し,k_g の数値は固定値とした．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, 単峰性が保たれないため,$T(P_i)$ 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.

先行研究では,$T(P_i)$ の単峰性を保つため，盛り土関数 $C_f(P_i)$ と調整係数 k_f) により

$$T(P_i) = k_g C_g(P_i) + k_u C_u(P_i) + k_f C_f(P_i) \tag{2.2}$$

とし,$T(P_i)$ が単峰性となるように補正する.k_f は k_g,k_u 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, 固定値である．この $T(P_i)$ 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 離散ラプラシアンによる盛り土関数

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, 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離散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.

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

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. しかし, 先行研究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, 先行研究では以下の式によってラプラシアンを擬似的に算出するものとした.

$$\nabla^2(P_i) = \frac{1}{|N(i)|}\sum_{j\in N(i)}(T(P_i) - T(P_j)) \tag{2.4}$$

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. 各経路点の離散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 $C_f(P_i)$ 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た. 減衰を施す理由は, 第一に減衰しない場合にこの補正値はマップ全体で著しく増加してしまい,C_g や C_u の効果が次第に影響を持たなくなってしまうこと, 第二に既に各エージェントが移動した後の古い情報が誤判断を生じてしまうことを避けるためである. 減衰は以下の式によるものとした.

$$C_f^t(P_i) = (1-\epsilon)(C_f^{t-\Delta t}(P_i) + \nabla^2(P_i)) \tag{2.5}$$

ここで,$C_f^t(P_i)$ は時刻 t における $C_f(P_i)$ の値を示し, Δt はシーン更新時間間隔である. 本研究では,$\Delta t = \frac{1}{60}$ 秒として扱った. また,ϵ は正の微少量を意味し, 先行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 k_u を動的に変更するようにした. 追跡エージェントが複数いるような状況のとき, 逃亡エージェントが単純に追

跡エージェントを回避する行動を繰り返し続けると, 同じ場所を右往左往するような行動となってしまい, 結果的に複数の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ある程度停滞状態にある場合は, 状況打開のために追跡エージェント回避よりも目的地到達の優先度を上げることが, 結果的に有効となることが多い. この行動を実現するため, 逃亡エージェントの停滞の度合いによって係数 k_u を動的に調整するよう実装を行った.

先行研究では, 逃亡エージェントの時刻 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は, 一定量 H_t 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_t 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 P_t と一定時間 l 前の時点での位置 $P_t - l$ との距離を反映するものであり, 以下の式で算出した.

$$L_t = \frac{lV - |P^t - P^{t-l}|}{lV} \tag{2.6}$$

ここで V は逃亡エージェントの速さを表す. 結果として,L_t は上限を 1, 下限を 0 として, 大きいほど停滞した状態であることを意味する. 本研究では,$l = 10\Delta_t$ として算出した.

k_u 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 [19] とは,Lotfi Zadeh[20] のファジー論理が元となる, 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. ファジー推論は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. 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

$$k_u = U(1 - Z(H_t, L_t)) \tag{2.7}$$

を算出し, 調整係数 k_u の自動調整を行った. U は k_u の初期設定値であり,k_u の上限値を意味する.

2.5 先行研究での実装

$G(P)$ は，経路探索手法のダイクストラ法を用いて, 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 $U(P)$ 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

検証と評価

本研究では, ゴールベース指標の目標であるゴール地点を任意の位置への変更を実現と, マップの作成を行った. ゴールの変更は, 開始からユーザーが定めた時間の経過後に, ゴールの位置がユーザーが定めた位置に変更を行うようにした.

図 3.1〜3.28 は実行結果を示したものである. 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 1 に設定し, 閲覧性をよくするために 10 倍した数値を表記している. 体力が最大値の状態から開始する. 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なっている. 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. 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なり, 逃亡エージェントの軌跡を赤い線で描くようになっている. 表 3.1〜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.

3.1 検証結果

3.1.1 マップ 1 の検証結果

図 3.1, 図 3.2, 図 3.3 は離散ラプラシアンを用いずに $k_f = 0$ で, マップ 1 を動くエージェントの様子である. スタート地点の左下端からゴールである右上端に向かった. 開始から 40 秒後にゴールを変更し, 右下端に向かった. 図 3.1, 図 3.2 は図 3.3 の途中経過である. 一方, 図 3.4, 図 3.5, 図

3.6 は離散ラプラシアンを用いて $k_f = 0.3$ で動くエージェントの様子である. 図 3.4, 図 3.5, 図 3.6 と同じようにスタート地点の左下端からゴールである右上端に向かった. 開始から 40 秒後にゴールを変更し, 右下端に向かった. 図 3.4, 図 3.5 は図 3.6 の途中経過である.

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離散ラプラシアンを用いた場合の方が 3 回少なく, 到着にかかった時間は 9 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

離散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
離散ラプラシアンなし（$k_f = 0$）	82 秒	3 回
離散ラプラシアンあり（$k_f = 0.3$）	73 秒	0 回

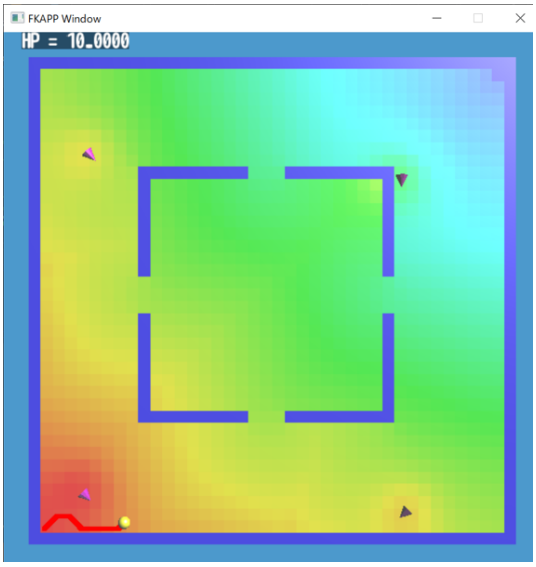

図 3.1　$k_f = 0$ で動く様子

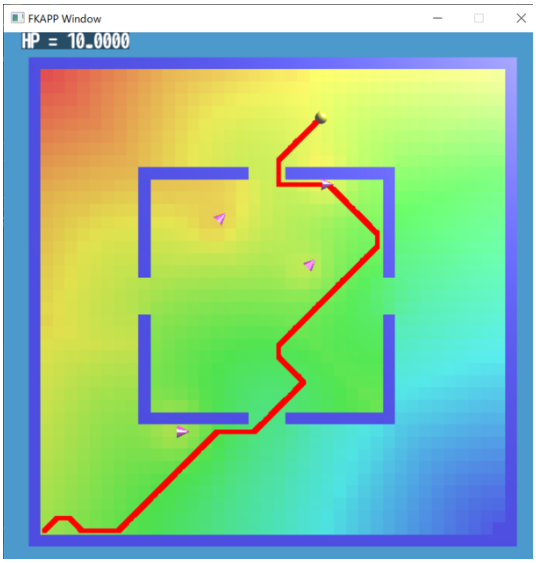

図 3.2　$k_f = 0.3$ で動く様子

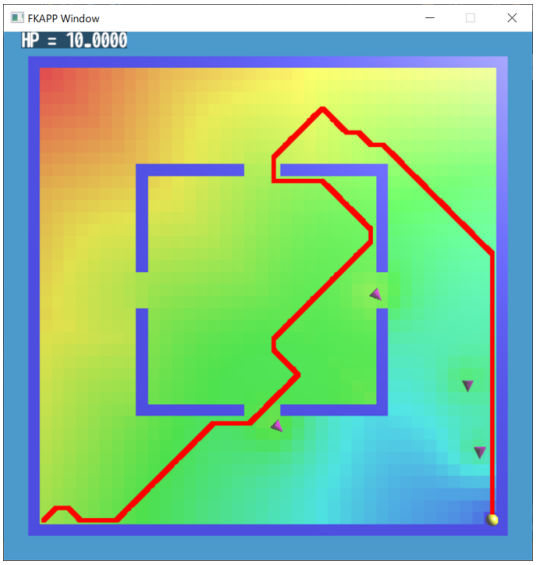

図 3.3　$k_f = 0.3$ で動く様子

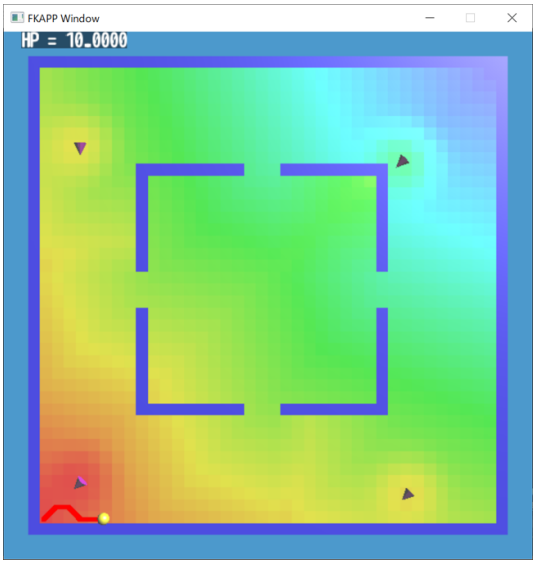

図 3.4　$k_f = 0$ で動く様子

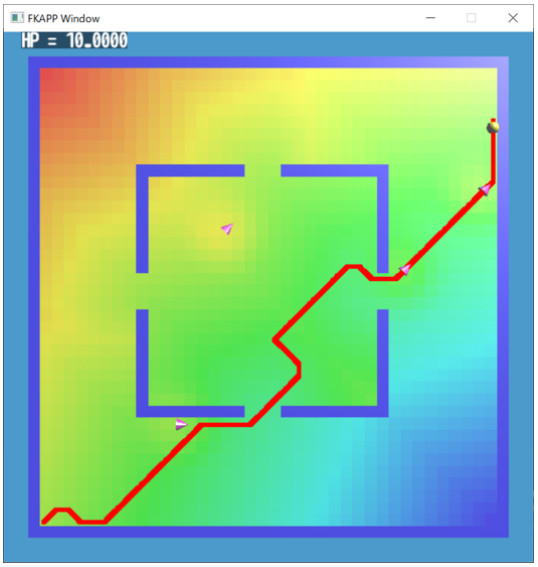

図 3.5　$k_f = 0.3$ で動く様子

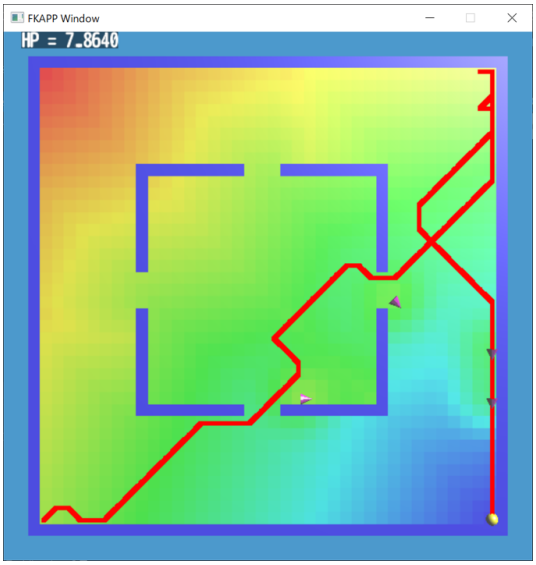

図 3.6　$k_f = 0.3$ で動く様子

3.1.2　マップ 2 の検証結果

図 3.7, 図 3.8, 図 3.9 は離散ラプラシアンを用いずに $k_f = 0$ で, マップ 3 を動くエージェントの様子である. スタート地点の左下端からゴールである右上端に向かった. 開始から 40 秒後にゴールを変更し, 右下端に向かった. 図 3.7, 図 3.8 は図 3.9 の途中経過である. 一方, 図 3.10, 図 3.11, 図 3.12 は離散ラプラシアンを用いて $k_f = 0.3$ で動くエージェントの様子である. 図 3.10, 図 3.11, 図 3.12 と同じようにスタート地点の左下端からゴールである右上端に向かった. 開始から 40 秒後にゴールを変更し, 右下端に向かった. 図 3.10, 図 3.11 は図 3.12 の途中経過である.

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. 逃亡エージェントが攻撃を受けた回数は, 離散ラプラシアンを用いない場合の方が用いる場合よりも用いた場合の方が 1 回多く, 到着にかかった時間は 9 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

離散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
離散ラプラシアンなし（$k_f = 0$）	89 秒	1 回
離散ラプラシアンあり（$k_f = 0.3$）	80 秒	2 回

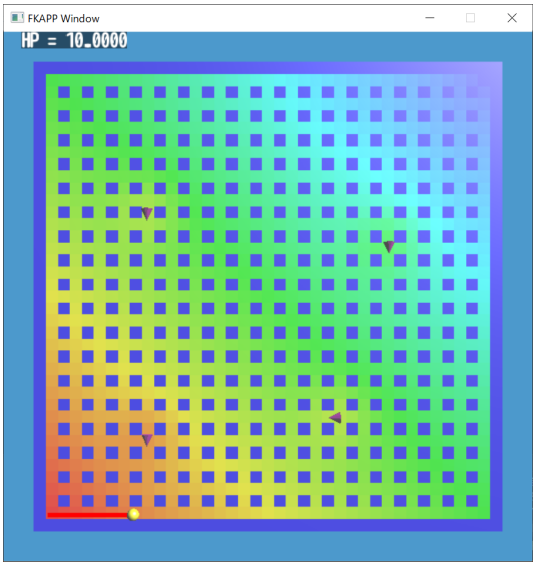

図 3.7　$k_f = 0$ で動く様子

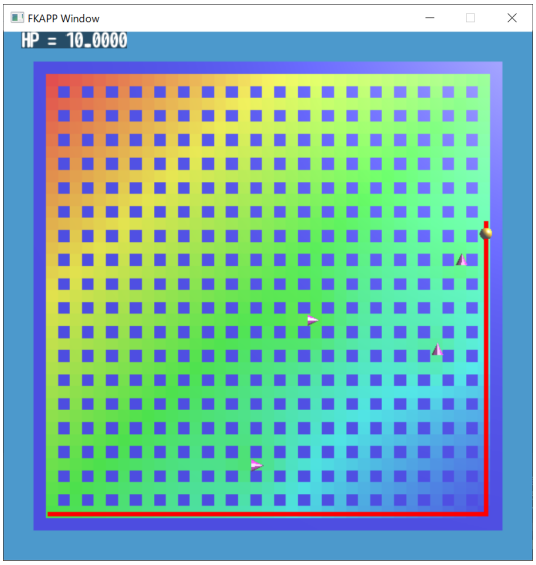

図 3.8　$k_f = 0.3$ で動く様子

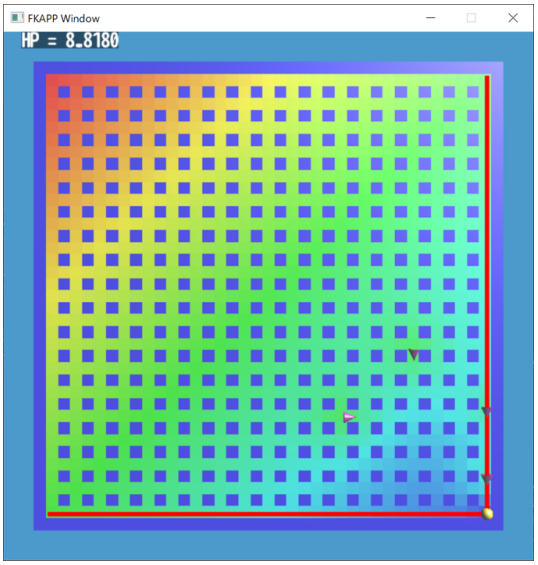

図 3.9　$k_f = 0.3$ で動く様子

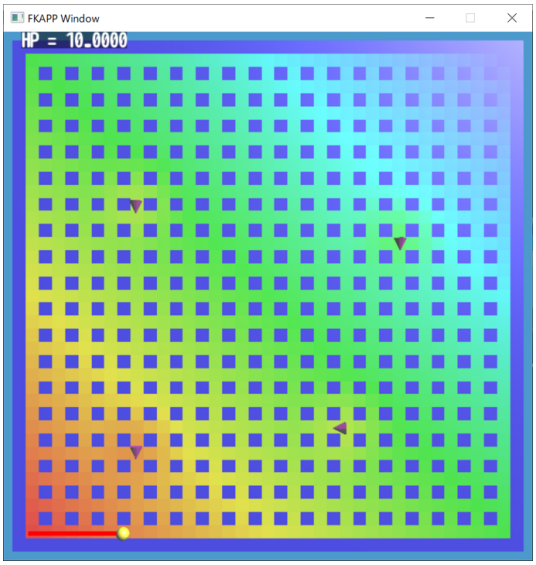

図 3.10　$k_f = 0$ で動く様子

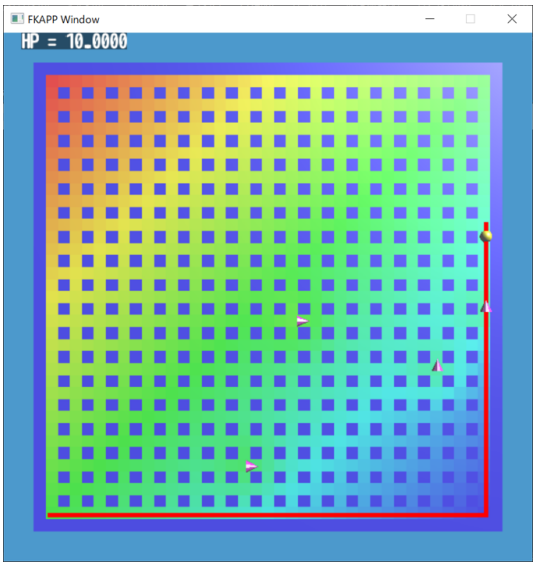

図 3.11　$k_f = 0.3$ で動く様子

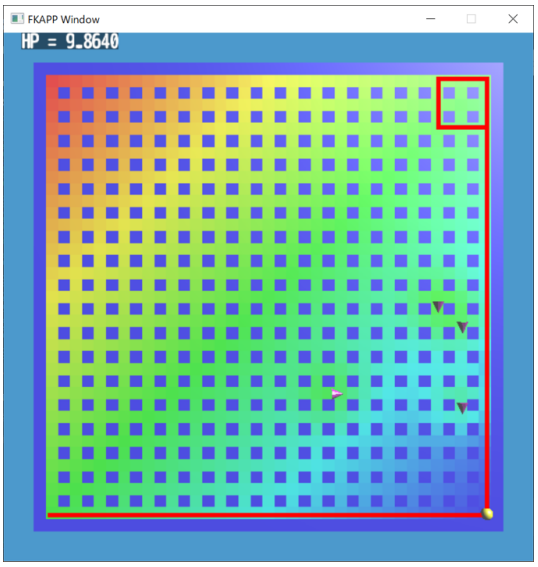

図 3.12　$k_f = 0.3$ で動く様子

3.1.3　マップ 3 の検証結果

図 3.13, 図 3.14, 図 3.15 は離散ラプラシアンを用いずに $k_f = 0$ で, マップ 3 を動くエージェントの様子である. スタート地点の左下端からゴールである右上端に向かった. 開始から 40 秒後にゴールを変更し, 右下端に向かった. 図 3.13, 図 3.14 は図 3.15 の途中経過である. 一方, 図 3.16,

図 3.17, 図 3.18 は離散ラプラシアンを用いて $k_f = 0.3$ で動くエージェントの様子である. 図 3.16, 図 3.17, 図 3.18 と同じようにスタート地点の左下端からゴールである右上端に向かった. 開始から 40 秒後にゴールを変更し, 右下端に向かった. 図 3.16, 図 3.17 は図 3.18 の途中経過である.

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. 逃亡エージェントが攻撃を受けた回数は離散ラプラシアンを用いた場合の方が用いない場合よりも, 用いた場合の方が攻撃を受けた回数は 1 回多く, 到着にかかった時間は 17 秒長いという結果になった.

表 3.3　途中経過時点での経過時間と攻撃を受けた回数

離散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
離散ラプラシアンなし（$k_f = 0$）	64 秒	0 回
離散ラプラシアンあり（$k_f = 0.3$）	81 秒	1 回

図 3.13　$k_f = 0$ で動く様子

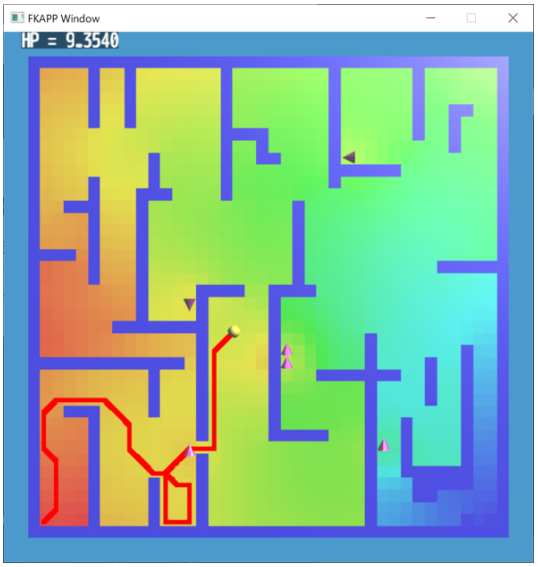

図 3.14　$k_f = 0.3$ で動く様子

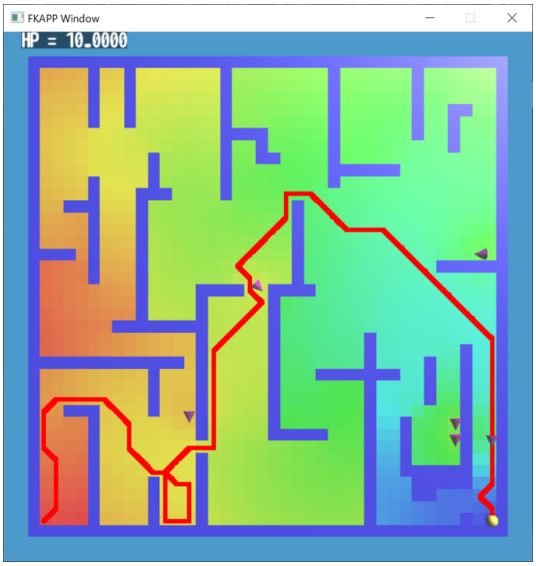

図 3.15　$k_f = 0.3$ で動く様子

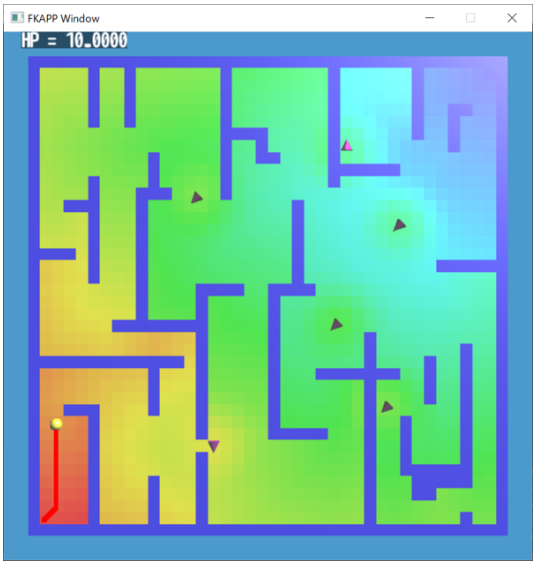

図 3.16　$k_f = 0$ で動く様子

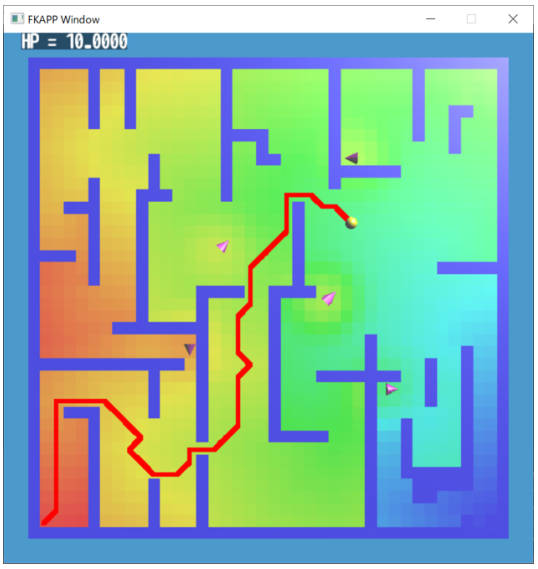

図 3.17　$k_f = 0.3$ で動く様子

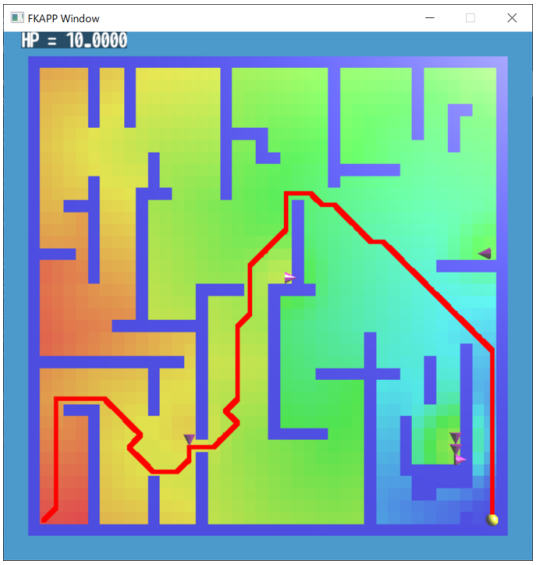

図 3.18　$k_f = 0.3$ で動く様子

図 3.19, 図 3.20, 図 3.21 は離散ラプラシアンを用いずに $k_f = 0$ で, マップ 3 を動くエージェントの様子である. スタート地点の左下端からゴールである右下端に向かった. 開始から 40 秒後にゴールを変更し, 右上端に向かった. 図 3.19, 図 3.20 は図 3.21 の途中経過である. 一方, 図 3.22, 図 3.23, 図 3.24 は離散ラプラシアンを用いて $k_f = 0.3$ で動くエージェントの様子である. 図 3.22, 図 3.23, 図 3.24 と同じようにスタート地点の左下端からゴールである右下端に向かった. 開始から 40 秒後にゴールを変更し, 右上端に向かった. 図 3.22, 図 3.23 は図 3.24 の途中経過である.

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. 逃亡エージェントが攻撃を受けた回数は, 離散ラプラシアンを用いない場合の方が用いる場合よりも用いた場合の方が 1 回多く, 到着にかかった時間は 28 秒短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

離散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
離散ラプラシアンなし（$k_f = 0$）	63 秒	7 回
離散ラプラシアンあり（$k_f = 0.3$）	91 秒	8 回

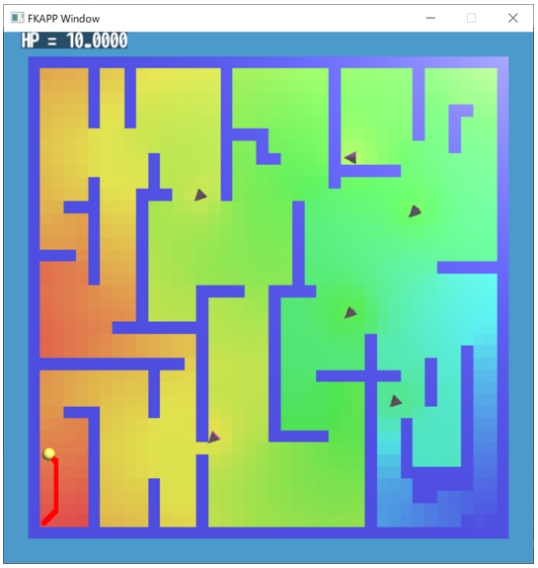

図 3.19　$k_f = 0$ で動く様子

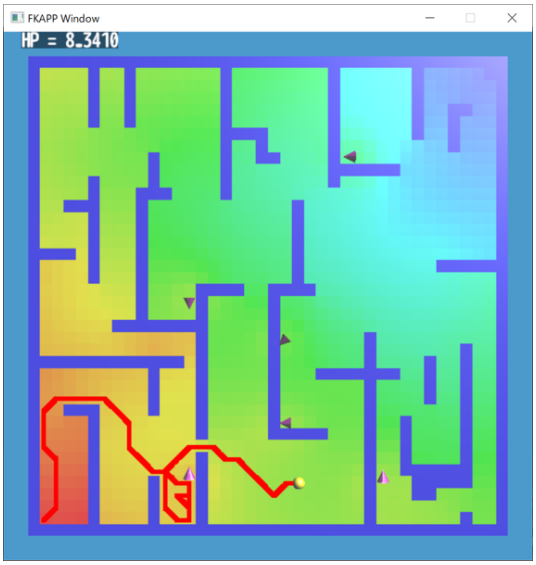

図 3.20　$k_f = 0.3$ で動く様子

図 3.21　$k_f = 0.3$ で動く様子

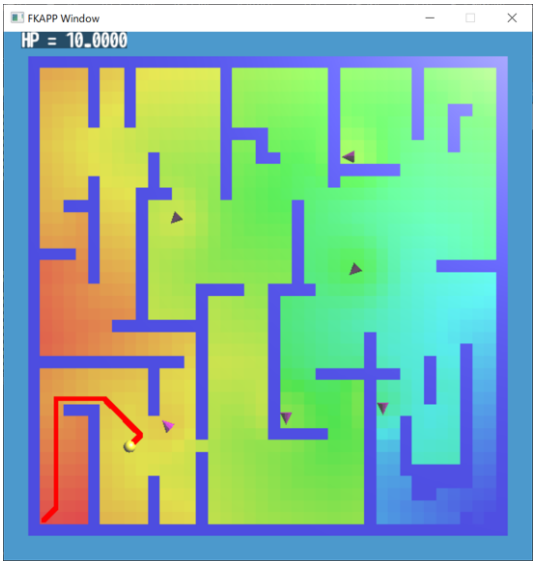

図 3.22　$k_f = 0$ で動く様子

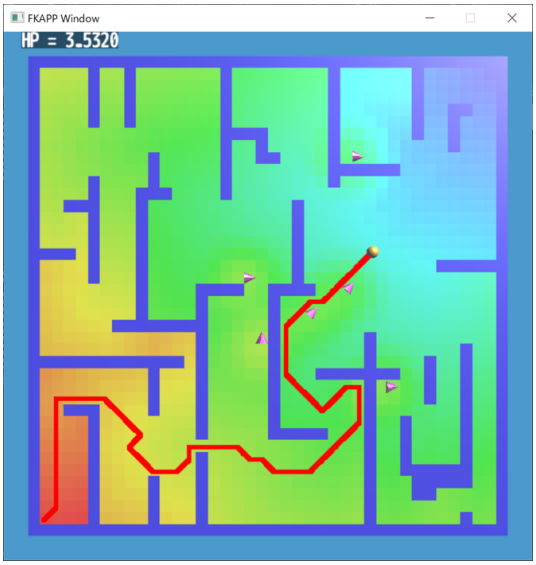

図 3.23　$k_f = 0.3$ で動く様子

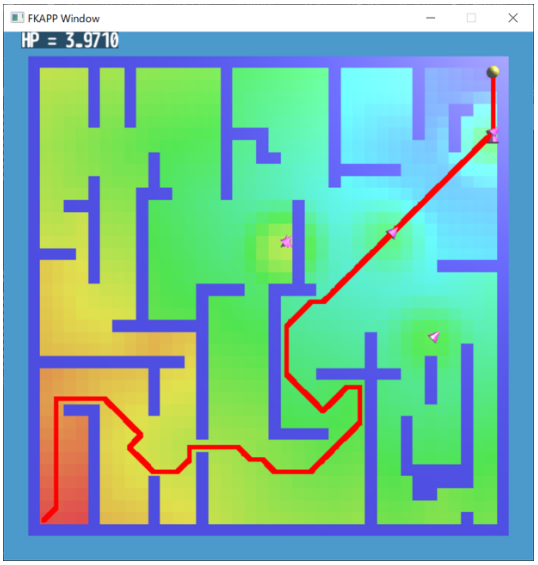

図 3.24　$k_f = 0.3$ で動く様子

マップ 3 ではゴールを変更しない場合での検証も行う. 図 3.25 は離散ラプラシアンを用いずに $k_f = 0$ で, マップ 3 を動くエージェントの様子である. ゴールである右上端に向かった. 一方, 図 3.26 は離散ラプラシアンを用いて $k_f = 0.3$ で動くエージェントの様子である. ゴールである右上端に向かった.

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．逃亡エージェントが攻撃を受けた回数は，離散ラプラシアンを用いない場合の方が用いる場合よりも用いた場合の方が 1 回少なく，到着にかかった時間は 17 秒短いという結果になった．

表 3.5　目的地到着までの経過時間と攻撃を受けた回数

離散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
離散ラプラシアンなし（$k_f = 0$）	58 秒	0 回
離散ラプラシアンあり（$k_f = 0.3$）	75 秒	1 回

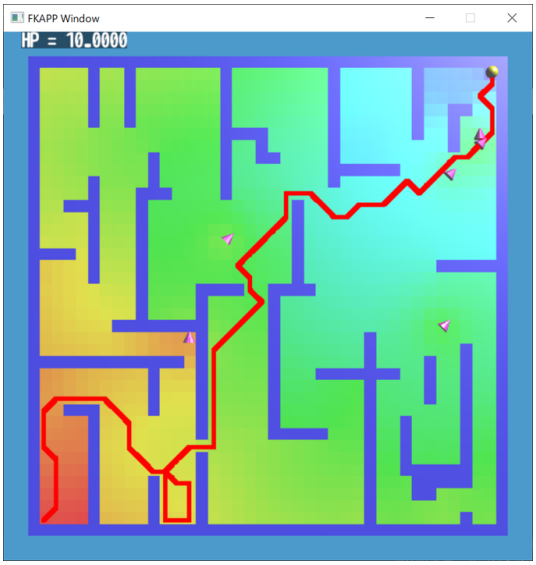

図 3.25　$k_f = 0$ で動く様子

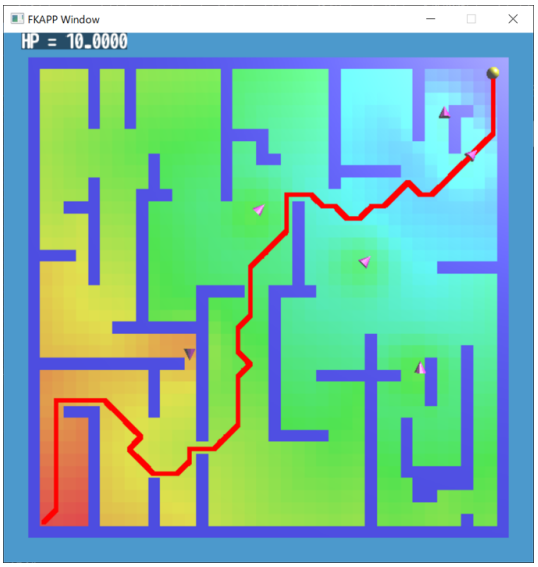

図 3.26　$k_f = 0.3$ で動く様子

図 3.27 は離散ラプラシアンを用いずに $k_f = 0$ で, マップ 3 を動くエージェントの様子である. ゴールである右下端に向かった. 一方, 図 3.28 は離散ラプラシアンを用いて $k_f = 0.3$ で動くエージェントの様子である. ゴールである右下端に向かった.

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. 逃亡エージェントが攻撃を受けた回数は, 離散ラプラシアンを用いない場合の方が用いる場合よりも用いた場合の方が 4 回多く, 到着にかかった時間は 17 秒長いという結果になった.

表 3.6　目的地到着までの経過時間と攻撃を受けた回数（障害物のあるマップ）

離散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
離散ラプラシアンなし（$k_f = 0$）	91 秒	2 回
離散ラプラシアンあり（$k_f = 0.3$）	114 秒	6 回

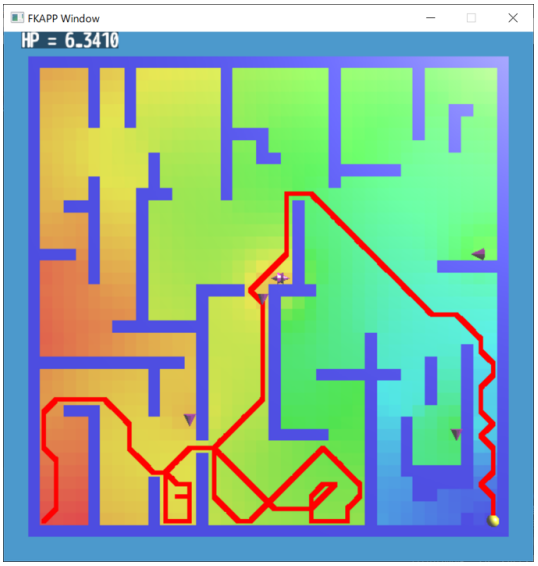

図 3.27　$k_f = 0$ で動く様子

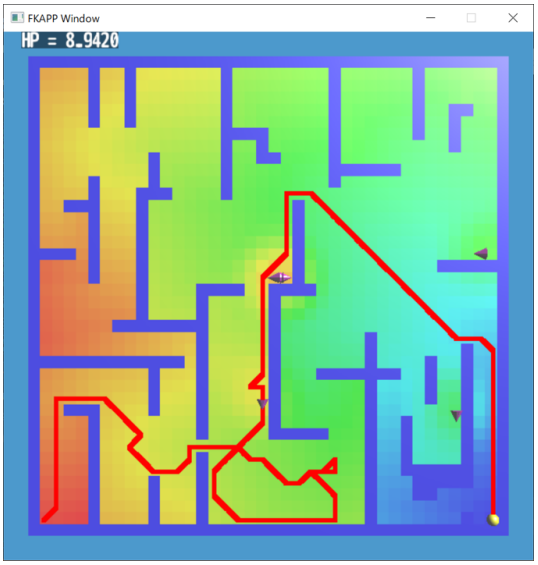

図 3.28　$k_f = 0.3$ で動く様子

3.2　考察

検証した結果, マップ 1, マップ 2 では疑似ラプラシアンを用いた場合の方が用いなかった場合よりも目的地到着までの経過時間が短く, 攻撃を受けた回数が少なくなることが分かった. マップ 3 では目的地までの到達時間が長く, 攻撃を受けた回数は多くなった. しかし, マップ 3 のゴールの変更を行わない場合での検証でも目的地までの到達時間が長く, 攻撃を受けた回数は多くなったため目的地までの到達時間が長く, 攻撃を受けた回数は多くなった原因はゴールの変更を行ったことではなくマップ 3 の形状と敵の配置であると考えられる.

第 4 章

まとめ

本研究は, ユーティリティベース AI のゴールベースと非ゴールベースを足し合わせた場合に, ゴールベースの特性である推移律を失わずに最終的な目標達成を保証する手法においてゴールベースの目標が途中で変更された場合にも手法の適用が有効であることを検証した. 本研究では, 作成した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ゴールが変更された場合でも目的地への到達時間が短くなることが確認できた. しかし, マップによっては目的地への到達時間が長くなる場合もあったが, 同じマップでゴールの変更を行わずに検証した結果, 目的地への到達時間が長くなることが分かり, ゴールの変更ではなくマップの形状や敵の配置が原因であると考えられる. 今後は目的地への到達時間が長くなる原因について調査し, 目的地への到達時間を短くする方法を検討していきたい.

謝辞

本論文の作成にあたり, ご指導いただきました先生方, 多くのアドバイスをしてくださった先輩方に心より感謝いたします.

渡辺先生には多くの力添えをいたただきました. 研究のテーマが定まらない時や, 研究の進捗が芳しくない時も, 多くのアドバイスをいただき, 研究をここまで進めることができました. 心より感謝いたします.

阿部先生にも大変お世話になりました. 自分の研究に足りない点や改善すべき点など, 研究内容に対して多くの助言をいただきました. 心より感謝いたします.

研究室の先輩方には多くの助力をいただきました. 研究の助言や論文の添削をしていただき, 自分では気付けなかった課題を知ることが出来ました. 心より感謝いたします.

中間発表では学部生の方や先生方からの多くの質問やアドバイスをいただきました. 自分にはない視点からの意見を聞くことができ, とても参考になりました. 心より感謝いたします.

研究を進めていく上でうまく行かないことが沢山ありました. それでも研究をやり遂げる事が出来たのは, 多くの方々に助けていただいたおかげです. 本当にありがとうございました.

参考文献

[1] 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.

[2] 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.

[3] 入倉雅春, 五十嵐治一, 山岸準, 山岸拓海. RoboCup サッカーシミュレーションリーグ 2D における守備力の向上. 情報処理学会研究報告, Vol. 2019-GI-41, No. 17, pp. 1–5, 2019.

[4] 隅山淳一郎, 張輝陽, 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.

[5] 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.

[6] 秋山英久, 岡山智彦, 中島智晴. 強化学習を用いたサッカーエージェントのボール保持行動獲得. 人工知能学会 AI チャレンジ研究会 (web), Vol. 35, pp. B201–1.

[7] 森寅嘉, 角薫. 状況に応じた動作を制御するキャラクタ AI. Vol. 2018-EC-50, No. 33, pp. 1–7, 2018.

[8] 三宅陽一郎. ゲーム AI 技術入門. 技術評論社, 2019.

[9] 佐藤悠. 協調ユーティリティを用いたゲーム AI の調整の簡易化. 第 75 回全国大会講演論文集, Vol. 2013, No. 1, pp. 105–106, 2013.

[10] 野津裕太, 後藤嵩幸, 橋本剛. 将棋 AI における評価項目自動抽出法の高速化手法. ゲームプログラミングワークショップ 2016 論文集, Vol. 2016, pp. 123–128, 2016.

[11] 三宅陽一郎. クロムハウンズにおける人工知能開発から見るゲーム AI の展望. https:

//www.slideshare.net/youichiromiyake/ss-250062606. 参照: 2021.12.25.

[12] まなべ, 小山大輔. 面白さの評価関数は作れるか？麻雀対局中の嗜好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. 参照: 2021.12.25.

[13] KONAMI. 桃太郎電鉄　公式サイト. https://www.konami.com/games/momotetsu/. 参照:2022.1.19.

[14] 任天堂. マリオパーティ　スーパースターズ. https://www.nintendo.co.jp/switch/az82a/. 参照: 2022.1.19.

[15] バンダイナムコエンターテインメント. ビリオンロード. https://br.bn-ent.net/. 参照:2022.1.19.

[16] 丸山敦史, 柴田直樹, 村田佳洋, 安本慶一, 伊藤実. 観光スケジュール作成支援とスケジュールに沿った経路案内を行うパーソナルナビゲーションシステム. 情報処理学会論文誌, Vol. 45, No. 12, pp. 2678–2687, 2004.

[17] 大内斉, 五十嵐治一. 曲面評価関数を用いたサッカーエージェントの移動先決定. *The 21st Game Programming Workshop 2016*, 2016.

[18] 古川真帆, 阿部雅樹, 渡辺大地. 長期的目標達成を考慮したユーティリティーベース AI に関する研究. 芸術科学会論文誌, Vol. 20, No. 2, pp. 139–148, 2021.

[19] 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.

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