2018 年度　卒　業　論　文

2 次元波動方程式の伝搬作用による NPC 行動制御に関する研究

指導教員：渡辺　大地 准教授

メディア学部 ゲームサイエンスプロジェクト
学籍番号　M0115266
西川　孝亮

2019 年 2 月

2018 年度　　卒　業　論　文　概　要

論文題目

2 次元波動方程式の伝搬作用による NPC 行動制御に関する研究

メディア学部 学籍番号：M0115266	氏名	西川　孝亮	指導教員	渡辺　大地 准教授

キーワード	波動方程式、NPC 行動制御、ゲーム、 人工知能、経路探索

近年コンピューターゲームの発展により、ゲーム AI が盛んに研究されている。FPS などのゲームでは、ノンプレイヤーキャラクター (以下 NPC と呼称する) を目的対象まで移動する必要がある。NPC とは、プレイヤーが操作しないで動作するキャラクターのことである。しかし、基本的に目的対象は動的に移動するため、NPC はリアルタイムで目的対象を知る必要がある。そこで、本研究では 2 次元波動方程式を用いて、NPC を目的対象に移動する手法を提案する。まず目的対象で波を発生し、その波が NPC に到達した際に波が NPC に対してどの方向から来たかを計測する。波は障害物を迂回するため、NPC と目的対象の間に障害物があっても波は NPC にたどり着く。波が当たった NPC は波が来た方向に進行する。波は障害物にあたると反射するため、反射した波が遅れて NPC にあたった際に NPC が目的対象を誤認識してしまう問題がある。そこで、波が NPC にあたった段階で波を消し、再度目的対象から波を発生する。実験結果から NPC を目的対象まで移動することができた。障害物が NPC と目的対象の間にある場合でも、NPC が障害物を迂回し目的対象へ到達することを確認できた。しかし、本手法では NPC が目的対象から離れるほど障害物を迂回する際に大回りになってしまう問題があった。

目 次

図 目 次

第 1 章

はじめに

1.1　本研究目的と背景

近年人工知能 [1][2] が発展してきている。三宅 [3][4] は、ディジタルゲームはこの 10 年でハードウェアの急速な進歩により AI との融合を推進し、エージェント型 AI は本来ユーザーのキャラクターの役割の代理をさせることができると述べた。梶原ら [5] は、ゲームはそれ自身の興味深さに加えて、ルールが明確であり、勝ち負けがはっきりしているため、性能差を評価しやすい特徴があげられ、人工知能の有用性を示すうえで優れた題材だと述べている。2017 年には将棋の電王戦 [6] で、その年度の名人戦タイトル保持者が将棋 AI と 2 番勝負をし 2 局ともタイトル保持者が破れている。また、Max ら [7] は id Software が開発した QUEKE[8] のキャプチャーザフラッグという敵を倒しながら旗を奪い合うゲームモードにおいて、人工知能が人間を超えたと述べた。他にも様々なゲームでノンプレイヤーキャラクター (以下 NPC と呼称する) に人工知能が利用されている。NPC とは、プレイヤーが操作しないで動作するキャラクターのことである。キルゾーン [9] という FPS ゲームでの NPC の制御や、ストラテジーゲームなどでの NPC の制御 [10][11] などが行われている。FPS ゲームにおいて、NPC の行動を制御する方法として、マップ上に等間隔にポイントを敷き詰め、そのポイントの性質を分析・評価し NPC を移動する手法 [12] があ

る。他に、シューティングゲームにおいて敵の弾をよけるルートを経路探索を用いて求める手法[13][14] もある。経路探索の関連研究として門馬 [15] による、マップグラフ構築の処理を軽減する手法がある。これらに共通しているのは NPC を目的対象まで移動することである。目的対象までは基本的に障害物があり、また目的対象が動的に変化することもある。三宅 [12] の方法はあらかじめ各ポイントに様々な情報を蓄積しなければならないため、マップが広大になるにつれ蓄積する情報も膨大なものになる。経路探索を用いる手法では、マップが複雑化するほど処理が重くなる。必要領域すべてを探索せずに一定まで探索し、その時点での最適解を求める手法もある。しかし、一定までしか探索しないため精度が落ちてしまい、場合により望む結果が出ない場合がある。またコンピューターゲームで利用する経路探索の手法の一つに A*アルゴリズム [16] がある。A*アルゴリズムは NPC から目的対象までにある各地点間の時間や距離などのコストを計算し、コストマップを生成する。そして、コストマップを基に最適な経路を計算する。しかし、問題点として、マップ全体のコストマップを参照するため処理が重く、また目的対象が移動するとコストマップの再構築が必要になる。

そこで本研究では、2 次元波動方程式 [17][18] を用いて NPC に目的対象を認識させる手法を提案する。まず目的対象で波を発生する。波は障害物を迂回するため、障害物に隙間なく囲まれた空間以外の空間にいきわたる。波の伝搬しない空間に目的対象への移動が必要な NPC もしくは目的対象そのものがあることは、NPC が目的対象へ移動できないことを示しているため波の伝搬しない空間は考慮しないものとする。そして波が NPC に到達した際に波が NPC に対してどの方向から来たかを計測する。目的対象から発した波が当たった NPC は波が来た方向へと移動する。波は障害物にあたると反射するため、反射した波が遅れて NPC にあたった際に NPC が目的対象を誤認識してしまう問題がある。そこで、波がプレイヤーにあたった段階で波を消し、再度目的対象から波を発する。また、本論文で提案する手法において、マップを新しくするごとに必要な事前準備は、マップ上の障害物の部分に、波を伝えないようにするだけである。そのため事前準

備が少なくなる利点がある。検証結果として、本研究では NPC を波の発生地点である目的対象へと移動することができた。

1.2　論文構成

本論文は全 4 章からなる。第 1 章は研究背景と目的について述べる。第 2 章は研究手法について述べる。第 3 章では検証内容とその結果について述べる。最後の第 4 章ではまとめと考察を述べる。

第 2 章

研究手法

2.1　波の伝搬

波の計算には 2 次元波動方程式 [17][18] を使用する。2 次元波動方程式は xy 平面上において位置および時刻 t によって表されるスカラー関数 $z = z(x, y, t)$ において、以下の式によって表される。

$$\frac{\partial^2 z}{\partial t^2} = c^2 \left(\frac{\partial^2 z}{\partial x^2} + \frac{\partial^2 z}{\partial y^2} \right). \tag{2.1}$$

ここで c は波の進む速さである。式 (2.1) の左辺を考えると偏微分の定義から式 (2.2) を得る。

$$\frac{\partial z}{\partial t} = \lim_{\Delta t \to +0} \frac{z(x, y, t + \Delta t) - z(x, y, t)}{\Delta t}. \tag{2.2}$$

また $z(x, y, t)$ が t で 2 階偏微分可能と仮定すると、$z(x, y, t)$ の変化量は、t のプラス方向とマイナス方向で同一値とみなせる。従って式 (2.3) も正しいということになる。

$$\frac{\partial z}{\partial t} = \lim_{\Delta t \to +0} \frac{z(x, y, t + \Delta t) - z(x, y, t)}{\Delta t} = \lim_{\Delta t \to -0} \frac{z(x, y, t) - z(x, y, t - \Delta t)}{\Delta t}. \tag{2.3}$$

Δt を十分小さな値と考えると式 (2.4) を得る。

$$\frac{\partial z}{\partial t} \approx \frac{z(x, y, t) - z(x, y, t - \Delta t)}{\Delta t}. \tag{2.4}$$

式 (2.4) と同様にして 2 階微分を近似すると式 (2.5) とできる。

$$\frac{\partial^2 z}{\partial t^2} \approx \frac{\frac{z(x,y,t+\Delta t)-z(x,y,t)}{\Delta t} - \frac{z(x,y,t)-z(x,y,t-\Delta t)}{\Delta t}}{\Delta t} = \frac{z(x,y,t+\Delta t) - 2z(x,y,t) + z(x,y,t-\Delta t)}{(\Delta t)^2}. \tag{2.5}$$

次に 2 次元波動方程式 (2.1) の右辺を式 (2.5) と同じように近似式を得ると

$$\frac{\partial^2 z}{\partial x^2} = \lim_{\Delta h \to +0} \frac{z(x+\Delta h,y,t) - z(x-\Delta h,y,t)}{(\Delta h)^2} \approx \frac{z(x+\Delta h,y,t) - 2z(x,y,t) + z(x-\Delta h,y,t)}{(\Delta h)^2} \tag{2.6}$$

となり、同様に、

$$\frac{\partial^2 z}{\partial y^2} \approx \frac{z(x,y+\Delta h,t) - 2z(x,y,t) + z(x,y-\Delta h,t)}{(\Delta h)^2} \tag{2.7}$$

となる。結果として、

$$c^2\left(\frac{\partial^2 z}{\partial x^2} + \frac{\partial^2 z}{\partial y^2}\right) \approx c^2\left(\frac{z(x+\Delta h,y,t) + z(x-\Delta h,y,t) + z(x,y+\Delta h,t) + z(x,y-\Delta h,t) - 4z(x,y,t)}{(\Delta h)^2}\right) \tag{2.8}$$

となる。式 (2.1) に式 (2.5) と式 (2.8) を代入すると、

$$\begin{aligned}&\frac{z(x,y,t+\Delta t) - 2z(x,y,t) + z(x,y,t-\Delta t)}{(\Delta t)^2}\\ &= c^2\left(\frac{z(x+\Delta h,y,t) + z(x-\Delta h,y,t) + z(x,y+\Delta h,t) + z(x,y-\Delta h,t) - 4z(x,y,t)}{(\Delta h)^2}\right).\end{aligned} \tag{2.9}$$

式 (2.9) をプログラムで記述しやすい形に変形する。格子状に並んだ面 $z_{i,j}$ に対して、格子点間の距離を Δh と考える。また z の時刻 k における値を $z^k_{i,j}$ として表し、$z^{k+1}_{i,j}$ と $z^k_{i,j}$ の間の時間差を Δt とする。これを式 (2.9) に当てはめると

$$\frac{z^{k+1}_{i,j} - 2z^k_{i,j} + z^{k-1}_{i,j}}{(\Delta t)^2} = c^2\left(\frac{z^k_{i+1,j} + z^k_{i-1,j} + z^k_{i,j+1} + z^k_{i,j-1} - 4z^k_{i,j}}{(\Delta h)^2}\right) \tag{2.10}$$

となる。この式 (2.10) を $z^{k+1}_{i,j}$ について解くことで、

$$z^{k+1}_{i,j} = \frac{c^2(\Delta t)^2}{(\Delta h)^2}(z^k_{i+1,j} + z^k_{i-1,j} + z^k_{i,j+1} + z^k_{i,j-1}) + \left(2 - \frac{4c^2(\Delta t)^2}{(\Delta h)^2}\right) z^k_{i,j} - z^{k-1}_{i,j} \tag{2.11}$$

を得る。しかし式 (2.11) はメッシュ端では適用できない。メッシュ端の計算では別の式が必要となり、隣接点が 3 点の x 成分が大きい方向の端では、

$$z^{k+1}_{i,j} = \frac{c^2(\Delta t)^2}{(\Delta h)^2}(z^k_{i-1,j} + z^k_{i,j+1} + z^k_{i,j-1}) + \left(2 - \frac{3c^2(\Delta t)^2}{(\Delta h)^2}\right) z^k_{i,j} - z^{k-1}_{i,j} \tag{2.12}$$

となり、ほかの隣接点が 3 点のメッシュ端部分でも $(z^k_{i-1,j} + z^k_{i,j+1} + z^k_{i,j-1})$ の部分を適当な式にすればよい。隣接点が 2 点の場合も同様に

$$z^{k+1}_{i,j} = \frac{c^2(\Delta t)^2}{(\Delta h)^2}(z^k_{i-1,j} + z^k_{i,j-1}) + \left(2 - \frac{2c^2(\Delta t)^2}{(\Delta h)^2}\right) z^k_{i,j} - z^{k-1}_{i,j} \tag{2.13}$$

とすればよい。隣接点が 3 点の時と同様に $z^k_{i-1,j} + z^k_{i,j-1}$ の部分を適当な式にすればよい。

2.2 NPC の行動原理

本研究において、NPC は波から得る情報を用いて目的対象へと移動する。最初 NPC はマップ上にランダムに生成する。マップは xy 平面上にあり、4 つの頂点で構成した面のメッシュである。マップ上の隣接する各頂点間の距離は 1 とし xy 平面上に等間隔に設置している。この時 NPC は目的対象や障害物、マップの大きさなどの情報は一切取得していない。NPC は生成してから波が衝突するまでは何もせずに待機する。その後目的対象から発生した波が NPC に衝突する。NPC に衝突した波がどの方向から来たかを計測する。図 2.1 は NPC が存在する周辺の頂点を表したものである。図 2.1 の黒丸が NPC である。

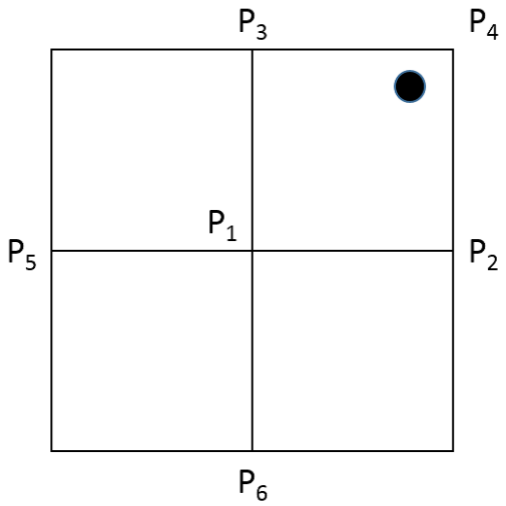

図 2.1　NPC と周囲の頂点

P_1 から P_2 へのベクトルをベクトル $\mathbf{a}$、P_1 から P_3 へのベクトルをベクトル $\mathbf{b}$、P_1 から P_5 へ

のベクトルをベクトル $\mathbf{c}$、P_1 から P_6 へのベクトルをベクトル $\mathbf{d}$ とする。NPC に衝突した波がどの方向から来たかを計測するには、NPC から見てどの方向が波の値が大きいかを計測すればよい。どの方向が波の値が大きいかを計測するには、波の値を求める関数を $f(x,y)$ とし NPC の座標を (x,y) としたときに x 軸方向と y 軸方向の単位ベクトルを $\mathbf{x}$ と $\mathbf{y}$ とすると式 (2.14)[19] で求められる。時刻 t における波の値は頂点の座標ごとにすでに求めてあるため、ここで波の値を求める関数は $f(x,y)$ となる。

$$\nabla f(x,y) = \frac{\partial f}{\partial x}\mathbf{x} + \frac{\partial f}{\partial y}\mathbf{y}. \tag{2.14}$$

頂点 P_1 の座標を (x_1,y_1) とするとき、$\nabla f(x_1,y_1)$ で求められるベクトルは、波の値 z を高さとしたときにベクトル $\mathbf{a}$ とベクトル $\mathbf{b}$ の外積と等しい。NPC がちょうど頂点上に存在すれば $\nabla(x_1,y_1)$ で求めたベクトルを使用すればいいが、NPC が頂点上に存在しない場合は補間する必要がある。補間する方法として、バイリニア補間 [20] を使用する。バイリニア補間とは、NPC と周囲の各頂点との距離に応じて重み付けをし補間する方法である。

NPC に衝突した波がどの方向から来たかを計測する際に、まず $\mathbf{a}$ と $\mathbf{b}$ の外積を求める。P_1 の z 座標を z_1、P_2 の z 座標を z_2、P_3 の z 座標を z_3 とすると、隣接する頂点の距離は 1 であるため、

$$\mathbf{a} = (1, 0, z_2 - z_1), \tag{2.15}$$
$$\mathbf{b} = (0, 1, z_3 - z_1) \tag{2.16}$$

となる。外積の定義から $\mathbf{a} = (a_1, a_2, a_3)$、$\mathbf{b} = (b_1, b_2, b_3)$ であるとき

$$\mathbf{a} \times \mathbf{b} = \begin{pmatrix} a_2b_3 - a_3b_2 \\ a_3b_1 - a_1b_3 \\ a_1b_2 - a_2b_1 \end{pmatrix} \tag{2.17}$$

である。式 (2.17) に式 (2.15)、式 (2.16) を代入すると、

$$\mathbf{a} \times \mathbf{b} = \begin{pmatrix} z_1 - z_2 \\ z_1 - z_3 \\ 1 \end{pmatrix} \tag{2.18}$$

となる。同様に $\mathbf{b}$ と $\mathbf{c}$、$\mathbf{c}$ と $\mathbf{d}$、$\mathbf{d}$ と $\mathbf{a}$ のそれぞれの外積を求め、求めた 4 つの外積のベクトル平均をベクトル $\mathbf{P}_1$ とする。同様の方法で $\mathbf{P}_2$、$\mathbf{P}_3$、$\mathbf{P}_4$ を求める。

次に求めた $\mathbf{P}_1$、$\mathbf{P}_2$、$\mathbf{P}_3$、$\mathbf{P}_4$ のベクトルに NPC からの距離に応じて重み付けを行い補間するバイリニア補間を行う。

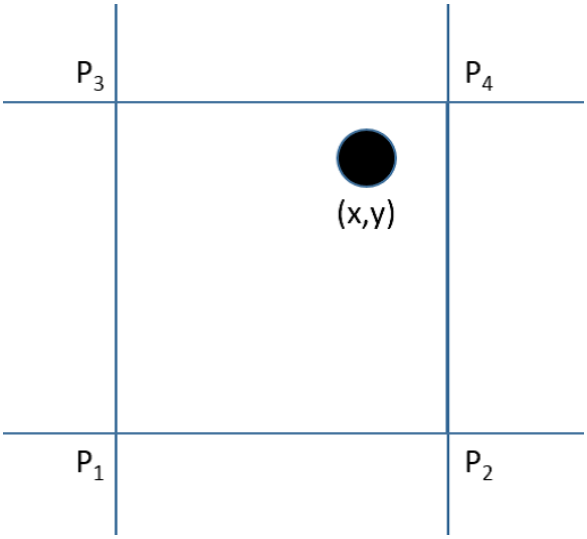

図 2.2　バイリニア補間

図 2.2 は、NPC が存在する周辺の頂点を表したものである。図 2.2 の黒丸は NPC を表しており、NPC の座標を (x, y)、頂点 P_1 の座標を (x_1, y_1)、頂点 P_2 の座標を (x_2, y_2) とする。まず、NPC の座標から各頂点の重みを計算する。図 2.3 は、NPC、頂点 P_1、頂点 P_2 それぞれの座標の x 成分の位置関係を表した図である。

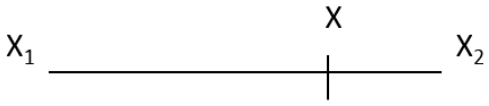

図 2.3　バイリニア補間の重み付け

ここで、x 軸方向の重み t_x を以下の式 (2.19) で求める。

$$t_x = \frac{x - x_1}{x_2 - x_1}. \tag{2.19}$$

図 2.3 に、式 (2.19) を適応すると、隣接する頂点の距離は 1 であるため、

$$t_x = x - x_1 \tag{2.20}$$

を得る。同様にして t_y も求める。求めた t_x、t_y と式 (2.18) で求めた $\mathbf{P}_1$、$\mathbf{P}_2$、$\mathbf{P}_3$、$\mathbf{P}_4$ を使用し補間したベクトルを求める。まず、t_x を用いて $\mathbf{P}_1$ と $\mathbf{P}_2$ について線形補間 [20] し、求めたベクトルを $\mathbf{V}_1$ とする。同様に、$\mathbf{P}_3$ と $\mathbf{P}_4$ について線形補間したベクトルを $\mathbf{V}_2$ とする。次に、t_y を用いて $\mathbf{V}_1$ と $\mathbf{V}_2$ について線形補間し求めたベクトルを $\mathbf{V}$ とする。

$$\begin{aligned} \mathbf{V}_1 &= (1-t_x)\mathbf{P}_1 + t_x\mathbf{P}_2, \\ \mathbf{V}_2 &= (1-t_x)\mathbf{P}_3 + t_x\mathbf{P}_4, \\ \mathbf{V} &= (1-t_y)\mathbf{V}_1 + t_y\mathbf{V}_2. \end{aligned} \tag{2.21}$$

式 (2.21) で求めた $\mathbf{V}$ の成分を (v_1, v_2, v_3) とする。波を発生する際、発生源の z 座標をプラスにするため、波の先端の座標について式 (2.21) と同様に $\mathbf{V}$ を求めた場合、$\mathbf{V}$ の xy 成分を取り出したベクトルは波の発生源と反対を向く。そのため $\mathbf{V}$ ベクトルが変化した瞬間に $\mathbf{V}$ ベクトルを記録し、$\mathbf{V}$ の xy 成分の符号を反転させれば波の来た方向を計測できる。NPC は計測した波が来た方向へ移動する。ここで障害物に反射しなかった波を直進波、障害物に反射した波を反射波と呼称する。ただし、直進波が障害物を回り込んだ場合は直進波とするものとする。直進波の先端が NPC に衝突した際、直進波は最短距離で NPC に衝突するのに対し、反射波は障害物に反射した後に NPC の方向に進んでいくため直進波より伝搬距離が長くなる。そのため、反射波が直進派の先端に影響することはない。直進波と反射波を図 2.4 であらわす。

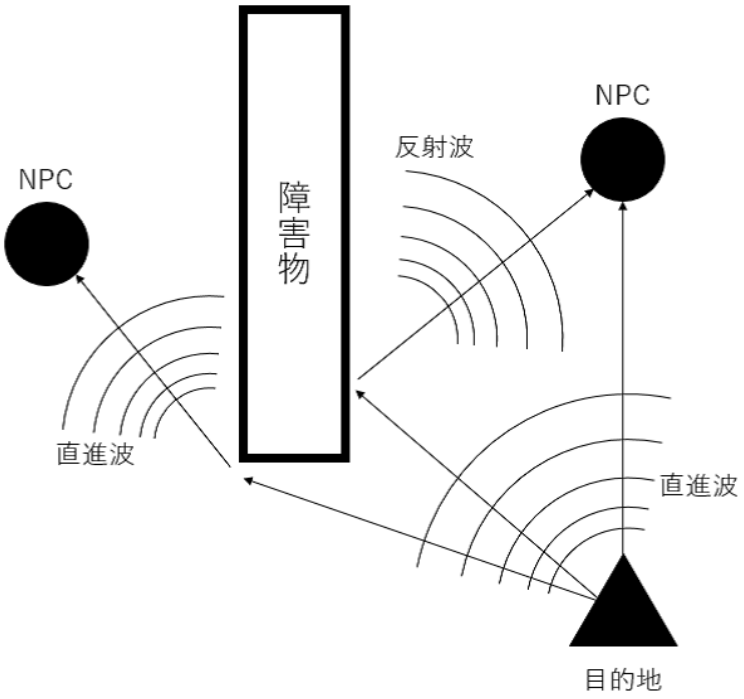

図 2.4　直進波と反射波

直進波の先端は障害波の影響を受けないが、直進波の先端以外はその限りではない。そのため、NPC に波が衝突した瞬間にマップ上の波をすべて初期化する。その後再度波を発生することで、NPC へと到達する波が障害波の影響を受けないようにする。移動している NPC に波が衝突したら再度計測し、波が来た方向へと移動する。これを繰り返すことで目的対象へ移動する。

2.3　システム概要

本研究で波が伝搬するマップでは、頂点を xy 平面上に等間隔に設置している。また、障害物として波の伝わらない部分を作成した。波はマップ上を伝搬する。また波は NPC の影響を受けない。図 2.5 が作成したマップの全体像である。

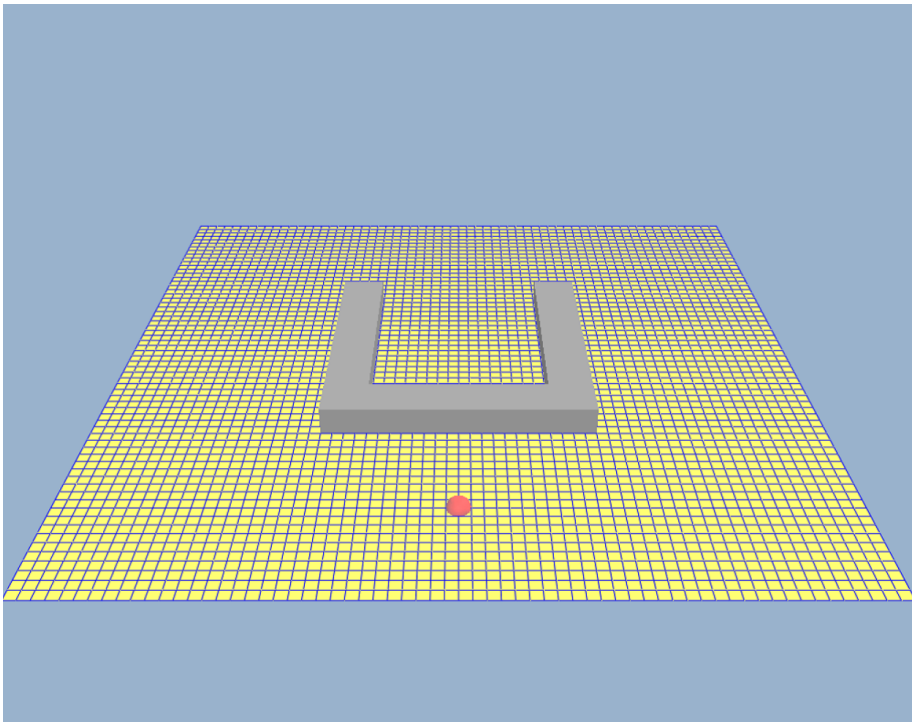

図 2.5　マップ

また NPC は見やすいように球体のモデルを設定しているが、NPC に波が衝突する際の判定はモデルの表面ではなく、モデルの中心座標の xy 成分で行う。また、NPC は xy 平面上のみを移動し、z 軸方向つまりは上下の移動はしないよう設定する。波の障害物への反射や回り込みの影響を防ぐため障害物を除いたマップを使用する。まずは NPC の座標の法線ベクトルを計測する。NPC の座標の法線ベクトルを可視化するため、NPC の座標の法線ベクトルの向きを表した棒状のオブジェクトを NPC に設置した。マップ上に NPC と NPC の座標の法線ベクトルをあらわした図が図 2.6 である。

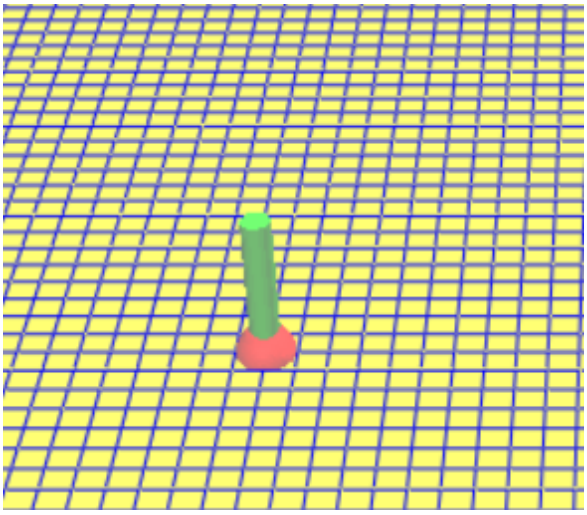

図 2.6　NPC 座標の法線ベクトル

NPC と波の発生位置を表した図が図 2.7 である。

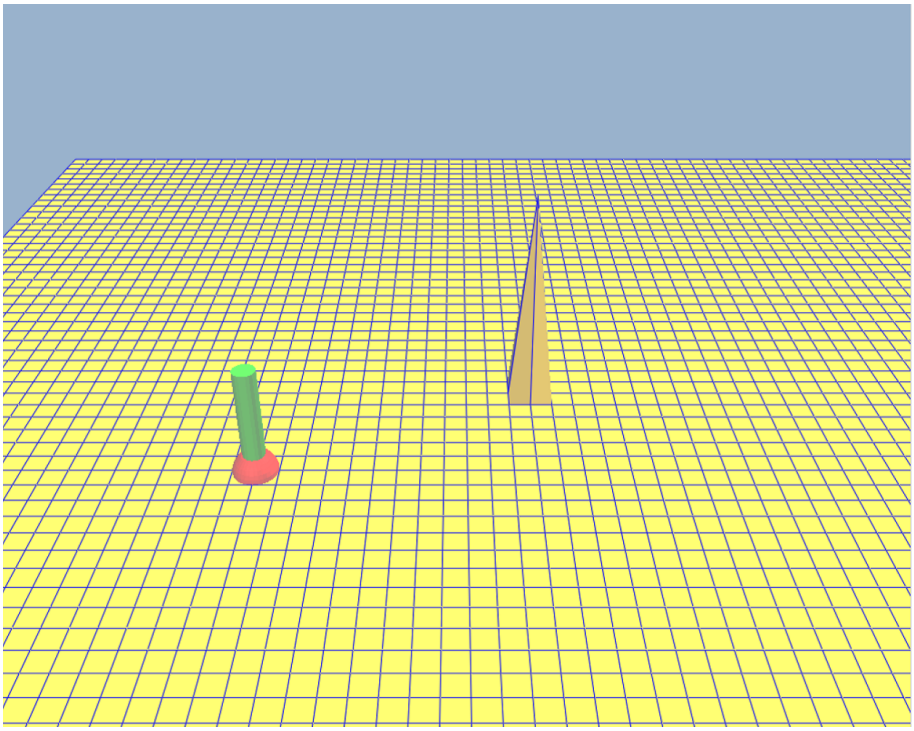

図 2.7　NPC と波の発生位置

ここで、波の発生位置を向いて山になっている波の部分を山側、谷になっている波の部分を谷側と呼称する。つまり、NPC が山側の波に衝突している間は、NPC の存在する座標の波の z 座標

は大きくなっていき、逆に NPC が谷側の波に衝突している間は、NPC の存在する座標の波の z 座標は小さくなっていく。xz 平面において、山側の波と谷側の波についてあらわした図が図 2.8 である。

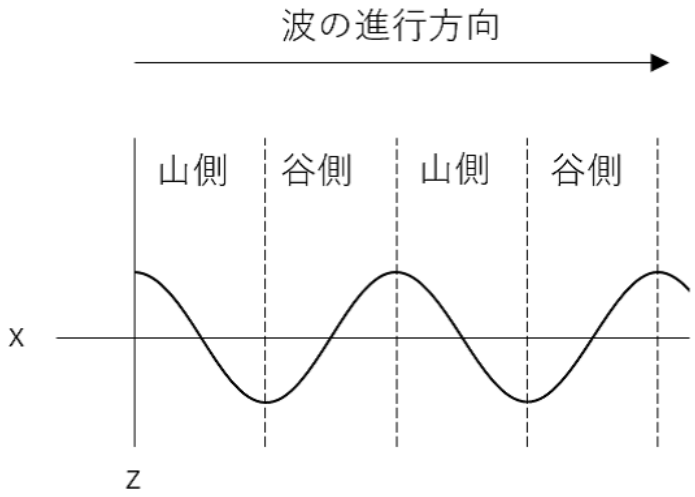

図 2.8　NPC と波の発生位置

NPC が波の山側と衝突している間は、NPC の座標の法線ベクトルの xy 成分は波の発生位置の反対方向を向く。また、NPC が波の谷側と衝突している間は、NPC の座標の法線ベクトルの xy 成分は波の発生位置の方向を向く。波がマップ上を伝搬し、NPC が波の山側に衝突している様子を表した図が図 2.9 である。

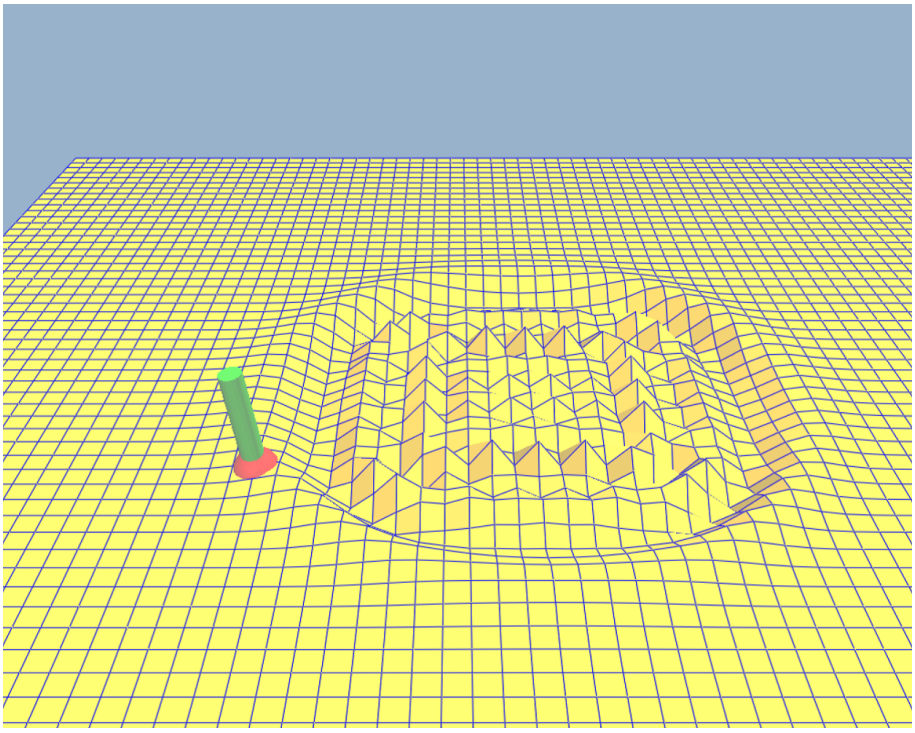

図 2.9　山側の波の衝突

波の山側が NPC を通り抜け、NPC が波の谷側に衝突している様子を表した図が図 2.10 である。

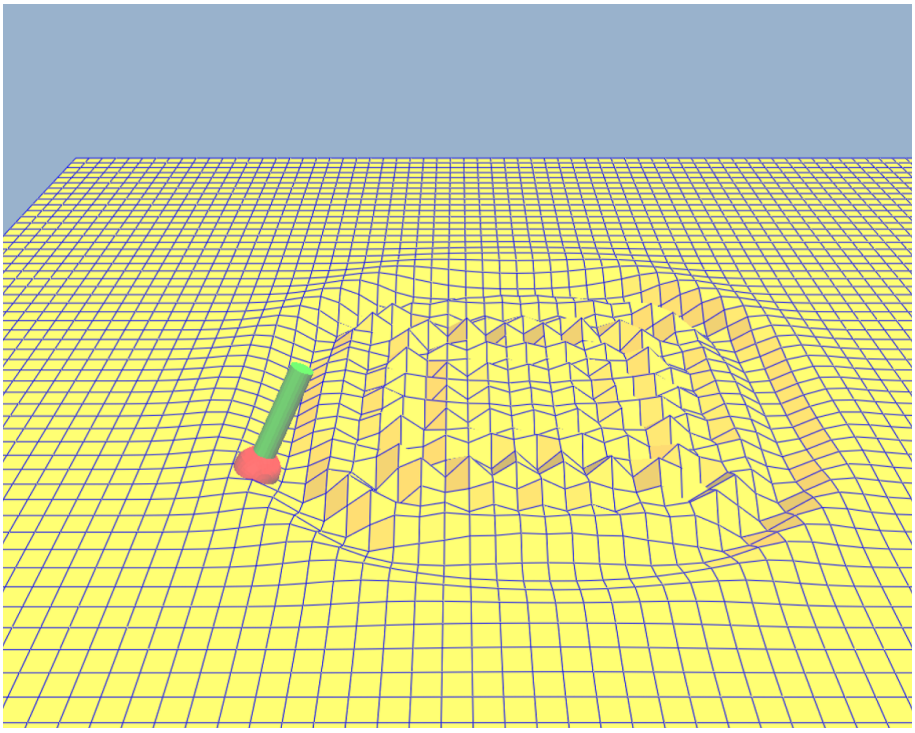

図 2.10　谷側の波の衝突

波を発生する際、z 座標をプラス方向に設定するため、波の先端は常に山側である。波の先端の座標は常に山側であるため、NPC は波が衝突した瞬間の法線ベクトルの xy 平面上において反対に進めばよい。NPC の座標の法線ベクトルは 3 次元ベクトルであるため、xy 平面上しか移動できない NPC の進行方向としては、そのままでは使用することができない。そのため、NPC の座標の法線ベクトルから z 軸方向の成分を 0 とした 2 次元ベクトルを進行方向とした。進行方向は NPC が波に衝突している間変化し続けるものではなく、波が NPC に衝突した瞬間のみ計測する。

次に、波がマップ上の障害物を回り込み NPC に衝突した場合である。波を障害物を回り込ませてから NPC に衝突するようにするため、NPC と目的対象の間に障害物を設置し、NPC の初期位置と波の発生位置を示した図が図 2.11 である。

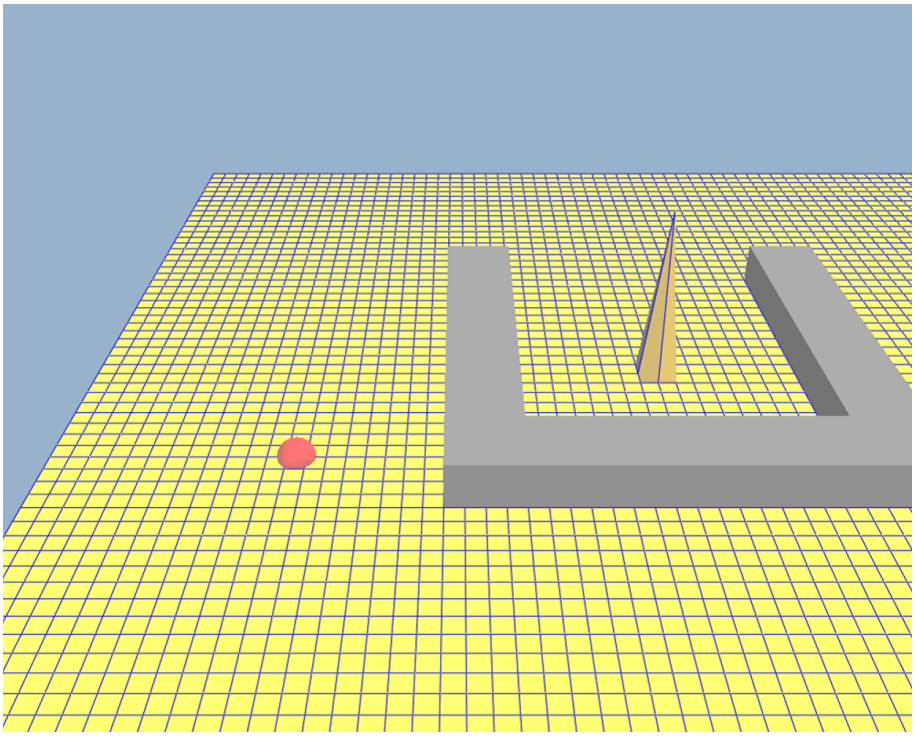

図 2.11　NPC の初期位置と目的対象

波は障害物を回り込んでから NPC に当たるため、波が NPC に衝突した瞬間の NPC の進行方向は目的対象ではなく、波が障害物を回り込んだ場所を向く。ここで、進行方向を視認するため NPC の進行方向を表した棒状のオブジェクトを NPC に設置した。図 2.12 は波が障害物を回り込む場面を表した図である。図 2.13 は波が NPC に衝突した際の NPC の進行方向を表した図である。

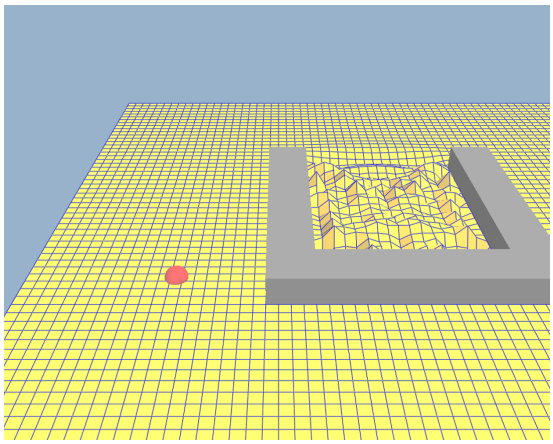

図 2.12　障害物を回り込んでいく波

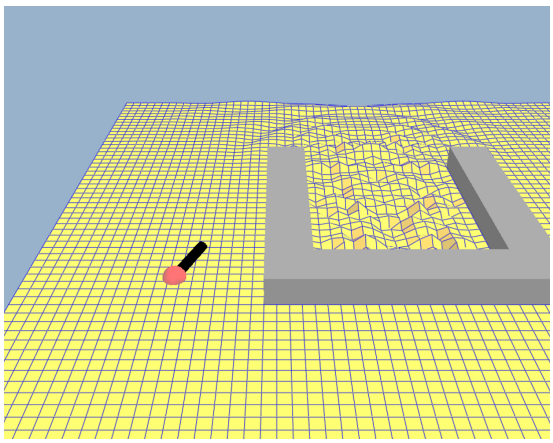

図 2.13　障害物がある場合の進む方向

図 2.13 の NPC から生えている棒状のオブジェクトは NPC が進む方向を示したものである。波が障害物を回り込んで NPC に衝突した場合は、NPC の進行方向は目的対象を向くのではなく、波が障害物を最後に回り込んだ場所を向く。NPC の進行方向を計測したら、NPC を進行方向へと移動し、波をリセットする。その後、再度目的対象から波を発生する。NPC は波が衝突するまで、計測した進行方向へ移動し続ける。移動している NPC に再度波が衝突したら進行方向を計測しなおし、NPC を新たに計測した進行方向に移動する。波の発生から NPC への波の衝突、進行方向の計測、波のリセットを繰り返すことで、NPC を目的対象まで移動する。

第 3 章

検証と考察

検証用のシステムは、Visual Studio2017 を使用し、ライブラリは FK[21] を使用して作成した。検証を行う環境について以下の表に表す。

表 3.1　検証を行った環境

OS	Windows 10
CPU	Intel(R) Core(TM) i5-7400 CPU @ 3.00GHz
GPU	GeForce GTX 1060 3GB

マップは初期状態として 2 次元平面の正方形である。マップは xy 平面上にあり、4 つの頂点で構成した 32×32 の合計 1024 面のメッシュである。頂点は xy 平面上に等間隔に設置している。また、本手法において 2 次元波動方程式を用いたマップの可視化は必要ないため処理時間の計測はマップを可視化してない状態で行った。処理時間を、Frame Per Second（以下 FPS と呼称する）という単位で計測したところ、60FPS 前後出ていた。FPS とは 1 秒あたりに何回描写を行ったかを指すものである。マップ上の隣接する各頂点間の距離は 1 とし、特に注釈がない場合 NPC の移動速度は 1 フレームに 0.05 ずつ進む。実測値として NPC がマップ左端から右端まで進むのに約 21 秒かかった。

3.1　基本的な動作

まず、基本的な動作を検証した。NPC と目的対象の間に障害物を設置し、NPC が障害物を回り込み目的対象まで移動する場合である。NPC の回り込みを確認するため、目的対象の位置は凹状の障害物に囲まれた地点とし、NPC は凹上の障害物の外側左下に設置した。マップ上に NPC の初期位置と障害物、目的対象を表したものが図 3.1 である。

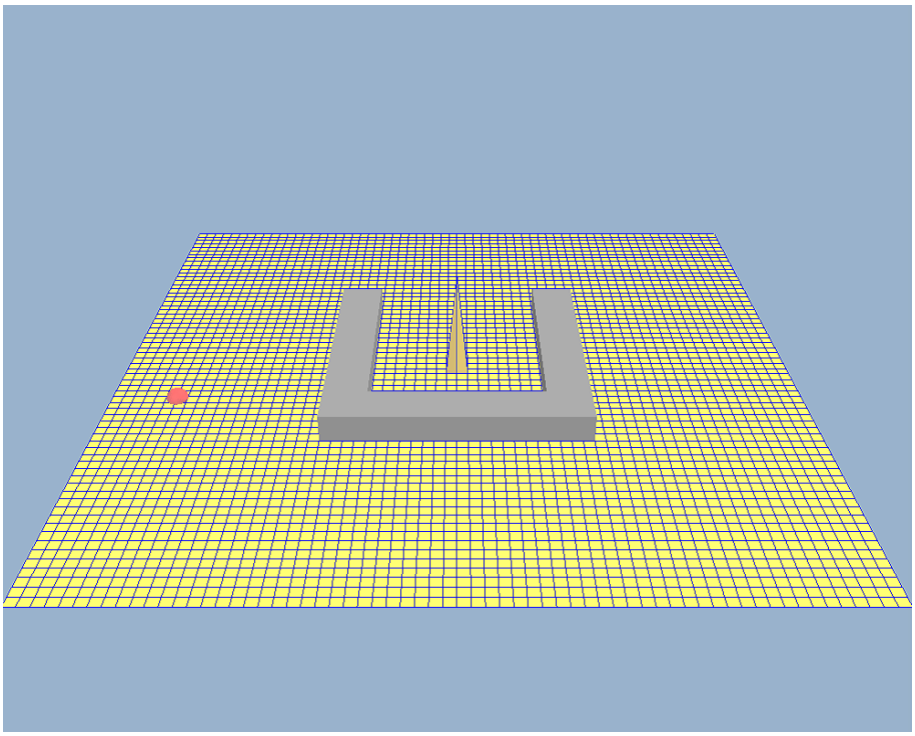

図 3.1　NPC の初期位置と目的対象

まず、目的対象で波を発生する。NPC は波が衝突した瞬間に進行方向を計算し、波の来た方向へと進行する。これにより NPC が障害物を回り込んでから目的対象へと進行することを確認した。図 3.2、図 3.3、図 3.4、図 3.5 は NPC に波が衝突してから、NPC が障害物を回り込み目的対象に到達するまでを表したものである。

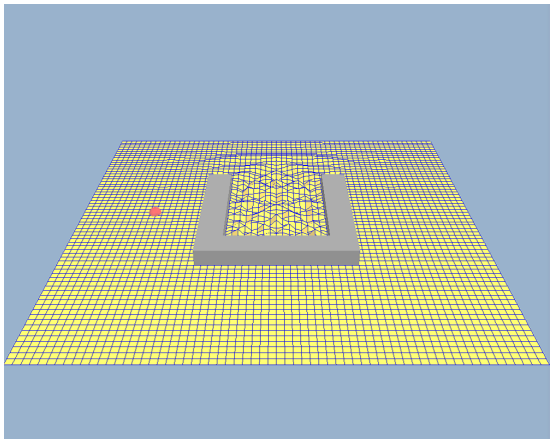

図 3.2　障害物の回り込み 1

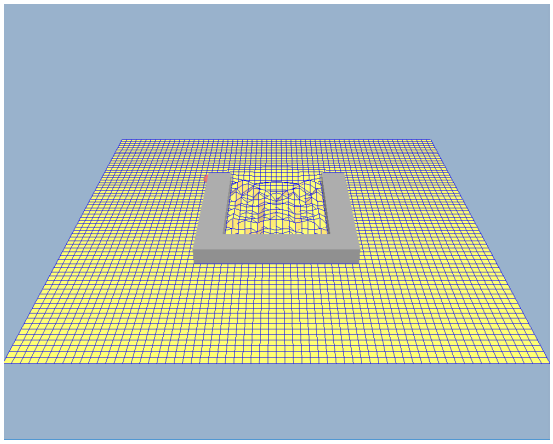

図 3.3　障害物の回り込み 2

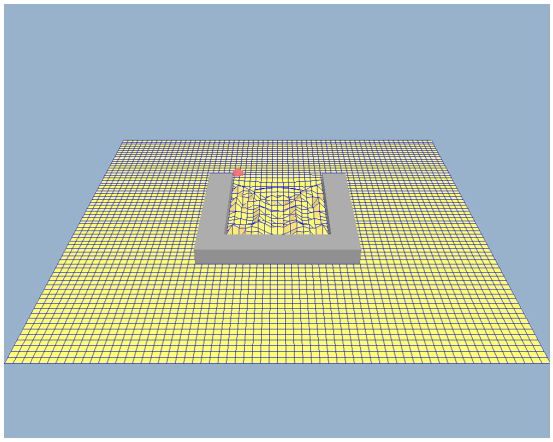

図 3.4　障害物の回り込み 3

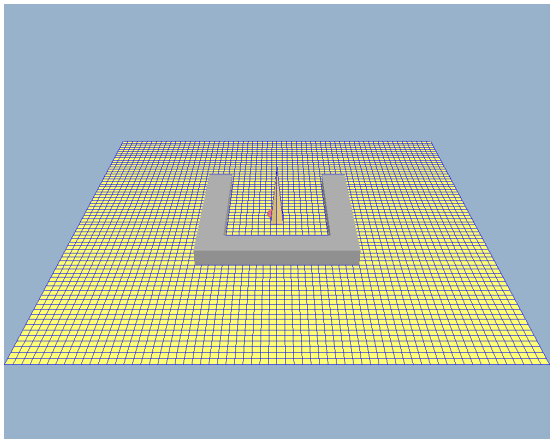

図 3.5　障害物の回り込み 4

目的対象から NPC の間に障害物があり、目的対象から発生する波が NPC に到達するまでの時間が長い場合、NPC が障害物を回り込む際に大回りになってしまう場面があった。NPC を障害物の後方に設置し、図 3.1 の時よりも目的対象で発生する波が NPC に衝突するまでの時間を長くした。図 3.6 は NPC の初期地点と目的対象を表している。

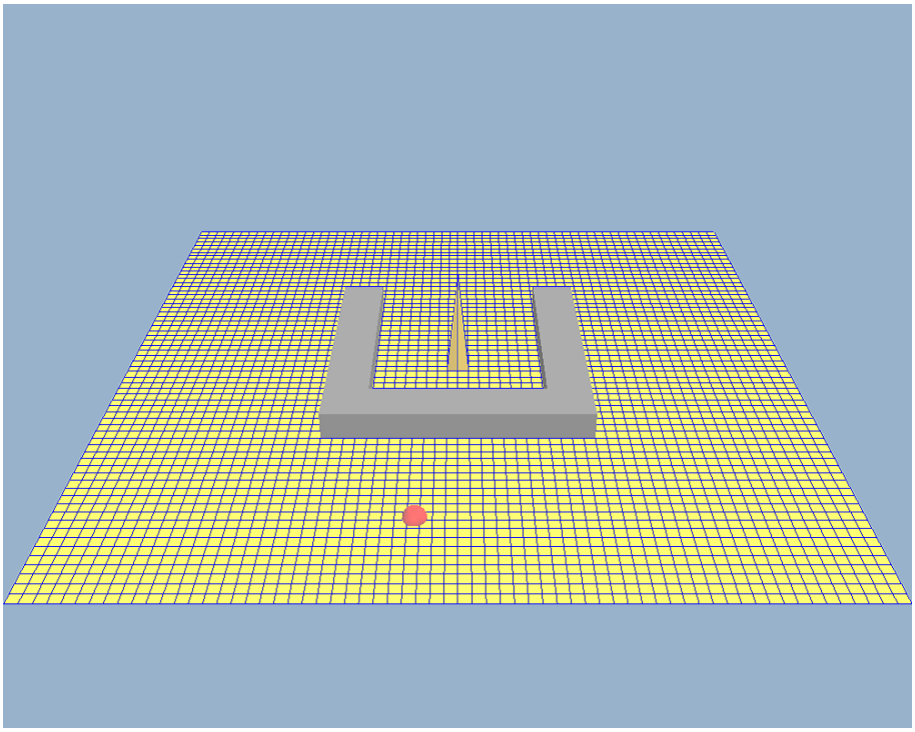

図 3.6　大回りでの回り込み 1

波は障害物の左前方から左後方を回り込み NPC に到達する。NPC は波が最後に回り込んだ部分である障害物左後方に向け進行した。ここで、NPC は障害物左後方に差し掛かっても直進を続けたことで、障害物から離れ大きく迂回するルートになった後目的対象へ到達したのが確認できた。図 3.7、図 3.8 は NPC が障害物の左後方を回り込む際に大回りになってしまった場面である。

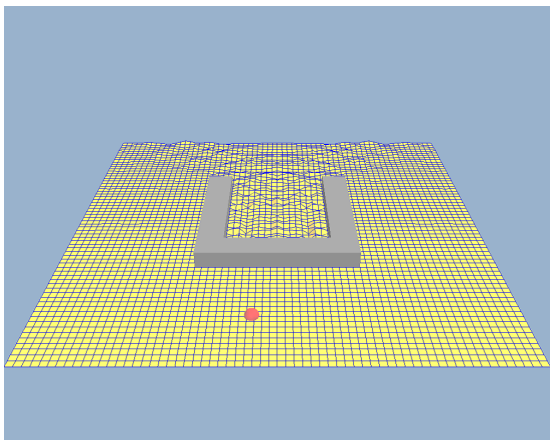

図 3.7　大回りでの回り込み 2

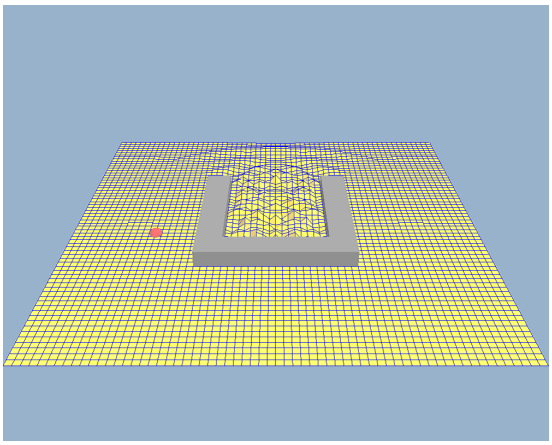

図 3.8　大回りでの回り込み 3

3.2　検証結果

目的対象から発生した波が複数回障害物を回り込み、NPC に衝突する場合を検証した。図 3.9 は検証に使用したマップを表したもので、初期設定として中央やや左側に NPC を配置し、右下に目的対象を設置した。検証により、NPC を目的対象まで移動させることができた。図 3.10 は NPC が目的対象へと到達した場面である。

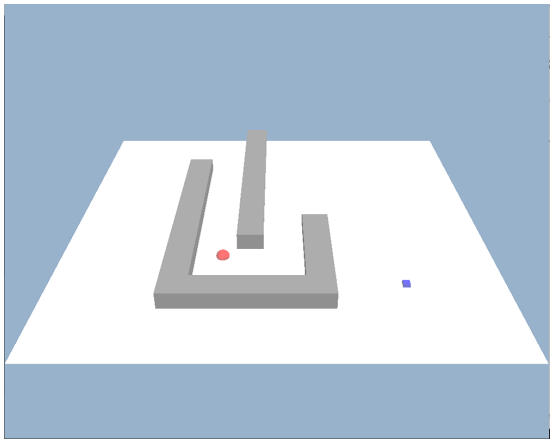

図 3.9　複数回の迂回 1

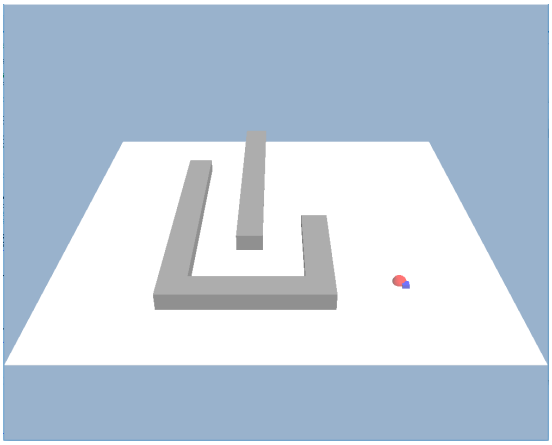

図 3.10　複数回の迂回 2

最初の波の発生から、NPC が目的対象へと到達するのに 17 秒 866 かかり 59.88FPS だった。また、図 3.11 に NPC のたどった軌跡を表した。目的対象から遠い地点を NPC が回り込むとき、目的対象から近い部分と比べ大回りになってしまう結果となった。目的対象が移動する場合の検証をした。図 3.12 は図 3.11 での目的対象を左側にいづさせたものである。目的対象が移動しても、NPC は目的対象へと移動するのが確認できた。

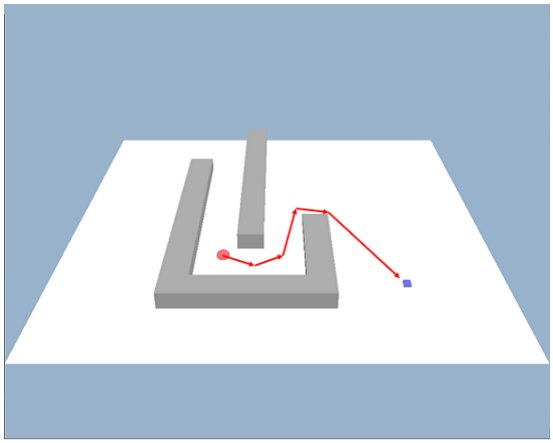

図 3.11　複数回の迂回 3

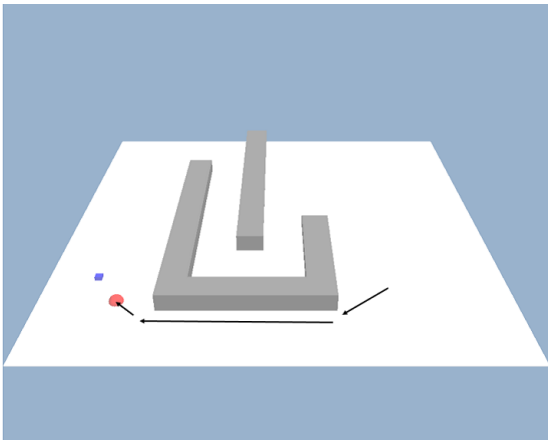

図 3.12　目的対象の移動

また、NPC の速度を 0.075 にして検証した結果、NPC が目的対象へと到達するのに 13 秒 435 かかり、59.86FPS だった。NPC の速度が 0.05 の時の結果より早く目的対象へ到達したが、NPC の速度が 0.05 の時のよりも大回りに迂回する結果になった。

次に、NPC の数を 2 つに増やして検証した。本手法は波が NPC に衝突した瞬間に波を初期化するため、NPC が二つある場合常に目的対象に近い NPC にしか波が当たらない。そのため、マップをもう一つ生成し、新たに生成したマップの波を目的対象から遠い NPC に対応させることで、2 つの NPC を移動した。図 3.9 と同じ配置で目的対象への到達時間を確認したところ 17 秒 871 であった。次に、目的対象の数を 2 つに増やして検証した。図 3.9 の右上に目的対象を一つ追加した。図 3.13 は検証に使用したマップを表したもので、初期設定として中央やや左側に NPC を配置し、右下と右上に目的対象を設置した。図 3.14 は NPC が目的対象へと到達した場面である。最初の波の発生から、NPC が目的対象へと到達するのに 17 秒 871 かかった。結果として、NPC からより遠い位置にある目的対象へ向かうことなく、NPC に近い位置にある目的対象へと行くことが確認できた。

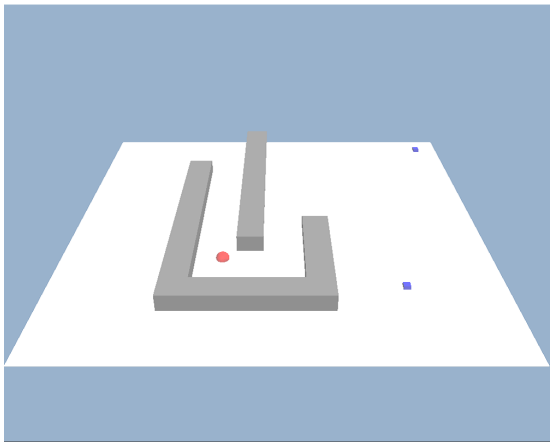

図 3.13　複数の目的対象 1

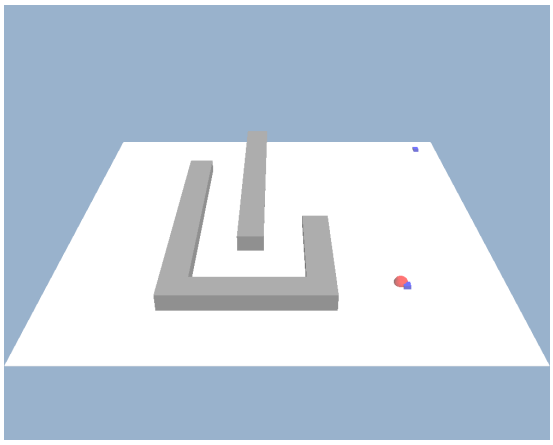

図 3.14　複数の目的対象 2

3.3　考察

本手法で、NPC を目的対象へと到達することができた。障害物があった場合も、障害物を回り込み、NPC を目的対象へと移動することができた。また、目的対象が移動した際も処理負荷を大きく増大することなく NPC を目的対象へ移動できた。しかし、NPC と目的対象との距離が遠くなるにつれて、障害物を迂回する際に大回りになる問題がある。これは、NPC と目的対象との距離が遠いため、NPC に波が当たる間隔が減るためだと推察する。また、NPC が障害物付近に存在する場合 NPC が正しい方向へ移動しない問題もある。これは、障害物に反射した波が直進波に影響したためだと考察できる。

第 4 章

まとめ

本研究では、波の伝搬作用を利用することで NPC を目的対象まで移動する手法を提案した。この手法は、マップをランダム生成するようなマップ情報を事前取得できないような環境でも使用できる。

検証の結果、障害物がなく、NPC から目的対象まで直進できる場合は NPC は目的対象へと進行することが確認できた。また、NPC から目的対象まで障害物があり目的対象まで直線できない場合は、NPC は障害物を迂回し目的対象まで向かうことが確認できた。しかし、NPC から目的対象までに障害物があり、NPC から目的対象までの距離が離れている場合、NPC が障害物を迂回する際大回りになってしまう問題があった。

今後の展望として、検証の結果わかった NPC から目的対象までの距離が離れると、NPC が障害物を回り込む際に大回りになる問題を改良する必要がある。NPC と目的対象までの距離が長くなるほど無駄な移動が増えるため、さらに検討する必要がある。

謝辞

本論文を執筆するにあたり、ご指導いただきました渡辺先生、阿部先生に深く感謝いたします。また、協力していただいた研究室のメンバーに感謝します。

参考文献

[1] 人工知能って何？https://www.ai-gakkai.or.jp/whatsai/. 参照:2018.12.26.

[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. 人工知能学会誌, Vol. 23, No. 1, pp. 44–51, 2008.

[3] 三宅陽一郎ほか. ディジタルゲームにおける人工知能技術の応用の現在 (特集エンターテイメントにおける AI). 人工知能, Vol. 30, No. 1, pp. 45–64, 2015.

[4] 三宅陽一郎. 人工知能が拓くオンラインゲームの可能性. *AOGC2007*, 2007.

[5] 梶原健吾, 鳥海不二夫, 稲葉通将, 大澤博隆, 片上大輔, 篠田孝祐, 松原仁, 狩野芳伸. 人狼知能大会における統計分析と SVM を用いた人狼推定を行うエージェントの設計. 人工知能学会全国大会論文集, Vol. JSAI2016, pp. 2F41–2F41, 2016.

[6] @DWANGO Co,. Ltd. 第 2 期 電王戦—ニコニコ動画. http://denou.jp/2017/. 参照:2018.12.26.

[7] Max Jaderberg, Wojciech M. Czarnecki, Iain Dunning, Luke Marris, Guy Lever, Antonio Garcia Castaneda, Charles Beattie, Neil C. Rabinowitz, Ari S. Morcos, Avraham Ruderman, Nicolas Sonnerat, Tim Green, Louise Deason, Joel Z. Leibo, David Silver, Demis Hassabis, Koray Kavukcuoglu, and Thore Graepel. Human-level performance in first-person multiplayer games with population-based deep reinforcement learning.

arXiv:1807.01281, 2018.

[8] id Software. https://www.idsoftware.com/en-us/. 参照:2018.12.26.

[9] KILLZONE(キルゾーン) ポータルサイト. https://www.jp.playstation.com/scej/title/killzone/. 参照:2018.12.26.

[10] 佐藤直之, 藤木翼, 心池田. 戦術的ターン制ストラテジーゲームにおける AI 構成のための諸課題とそのアプローチ. 情報処理学会論文誌, Vol. 57, No. 11, pp. 2337–2353, 2016.

[11] 加藤千裕, 三輪誠, 鶴岡慶雅, 近山隆ほか. ターン制ストラテジーゲームにおける戦術決定のための UCT 探索とその効率化. 情報処理学会ゲームプログラミングワークショップ 2013 論文集, pp. 138–145, 2013.

[12] 三宅 陽一郎 . Killzone における NPC の動的な制御. https://www.slideshare.net/youichiromiyake/killzonenpc. 参照:2018.12.17.

[13] 桑谷拓哉, 橋本剛. 熟練プレイヤーレベルを目指す弾幕シューティング AI の開発. 情報科学技術フォーラム, Vol. 12, No. 2, pp. 383–384, 2013.

[14] 佐藤直之, 池田心ほか. 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. 情報処理学会ゲームプログラミングワークショップ 2016 論文集, Vol. 2016, pp. 57–64, 2016.

[15] 門馬翔. エージェント型 AI の経路探索におけるグラフ構築の処理軽減手法に関する研究. 学部卒業論文, 東京工科大学, 2014.

[16] Smed Jouni, Hakonen Harri, 加藤諒, 中本浩. コンピュータゲームのアルゴリズム&ネットワーキング. ボーンデジタル, 2007.

[17] E.Lengyel, 狩野智英. ゲームプログラミングのための 3D グラフィックス数学. ボーンデジタル, 2002.

[18] 小菅晃治. 剛体による波の発生・反射を考慮した水面のリアルタイムシミュレーション. 学部

卒業論文, 東京工科大学, 2004.

[19] 宮本智之, 植之原裕行. スタンダード 工学系のベクトル解析. 講談社サイエンティフィク, 2014.

[20] 画像のリサイズを実装する (バイリニア編). https://hexadrive.jp/hexablog/program/26905/. 参照:2018.1.19.

[21] Fine Kernel ToolKit System. https://gamescience.jp/FK/. 参照:2018.12.26.