

遺伝的アルゴリズムと人間らしいAIプレイヤーを用いた 2Dシューティングゲームのステージ生成

吉田 友太^{1,a)} 池田 心^{1,b)}

概要：グラディウスなどに代表される2Dシューティングゲーム(STG)は、プレイヤーが自機を操作して敵や弾を回避しながら進めるゲームであり、その面白さや難易度は、敵の配置や攻撃パターンから構成されたステージに大きく依存する。それらは通常、“初見で攻略することは困難だが、繰り返し同じステージをプレイすることで攻略法が見つかることができる”ように、人手でデザインされる。一方で、毎回見たことのない新しいステージをプレイしたいと考えるプレイヤー層も存在し、ランダムなステージ生成にも需要がある。しかし、完全にランダムに生成してしまうと、難易度や面白さの観点から、人間がプレイするには不適当なステージばかりが生成されてしまう。そこで本研究では、それらに配慮したステージの自動生成システムの構築を目指した。生成手法は様々存在するが、我々は、遺伝的アルゴリズムによる最適化に、AIプレイヤーによる評価を組み合わせた生成検査法を採用した。評価関数としては、AIプレイヤーのクリア状況や、緊迫度を図るために無行動率などが一定範囲に入っているかのペナルティなども加えた。また、多様性の抑制のため、敵の群れをステージの最小要素とした。さらに、人間らしい挙動を行うAIプレイヤーを用いることで、「AIプレイヤーには簡単だが、人間には難しい」といった難易度の誤推定の防止を図った。得られたステージは、被験者実験によって、工夫を行わないものよりも難易度・面白さともに良い評価を得た。

キーワード：2Dシューティングゲーム、コンテンツ生成、遺伝的アルゴリズム、人間らしいAI

1. はじめに

近年、ゲームにおける人工知能の研究は盛んに行われており、中でもゲームAIプレイヤーは人間プレイヤーを凌駕するほどの強さに到達している。そのため、強いAIプレイヤーを作成する研究から、“人を楽しませるAI”の研究などへと目標が移行し始めている。後者の一分野として、Procedural Content Generation[1](PCG)と呼ばれる、最適化や機械学習などのアルゴリズムを用いて“コンテンツ”を自動で生成する技術がある。ゲームにおけるコンテンツはグラフィックやストーリーなど様々だが、敵や障害物の配置を司る“ステージ”もPCGの重要な対象である。本研究では、特に日本で人気のある2Dシューティングゲーム(STG)におけるステージ生成に着目する。

STGは、プレイヤーが敵や弾を回避しつつ撃破していくゲームであり、その面白さや難易度はステージ(敵の配置や攻撃パターン)に大きく依存する。多くのSTGのステージは、“初見で攻略することは困難だが、繰り返し同じ

ステージをプレイすることで攻略法を見つけられる”ように、職人芸により設計されたものが固定数だけ用意されている。一方で、毎回見たことのない新しいステージをプレイしたいと考えているプレイヤー層も存在する。そのため、ランダムなステージを生成することには一定の価値がある。とは言え、完全にランダムに生成してしまうと、難し過ぎて攻略のできないものや、易し過ぎてどうプレイしても攻略できてしまうもの、敵の出現の推移にメリハリがなく、終始暇ないし忙しい面白くないものなど、人間がプレイするには不適当なステージが生成されてしまう。

そこで本研究では、人間プレイヤーにとって面白く適切な難易度のステージを生成するシステムを目指す。PCGの手法は様々にあるが、テストAIプレイヤーによる評価と遺伝的アルゴリズム[2]による最適化を組み合わせた生成検査法を用いる。このようなアプローチでは、テストAIプレイヤーが人間離れした挙動を行うようだと、最適化されたステージが人間プレイヤーにとって難し過ぎるものになってしまう。そこで、テストAIプレイヤーに“人間らしさ”をもたらす3つの工夫を行う。最適化した結果は被験者実験によって評価を行う。

¹ 北陸先端科学技術大学院大学
Japan Advanced Institute of Science and Technology
^{a)} yuta.yoshida@jaist.ac.jp
^{b)} kokolo@jaist.ac.jp

2. 関連研究

2.1 ステージ生成

生成対象となるコンテンツとして，“ステージ”はしばしば取り上げられる。例えば，2009 年から 2012 年までゲームの著名な国際会議 IEEE CIG(Computer Intelligence on Games)では Mario AI Competition という競技会を行っており，「ステージを早くクリアできるマリオ AI」「人間らしく見えるマリオ AI」のほかに，「人間プレイヤーのレベルに合わせたステージ生成」を競技として行っていた。マリオのステージ生成には，GAN を用いたもの [3] や遺伝的アルゴリズムを用いたもの [4] など様々なものが提案されている。

STG については，あまり海外でよく遊ばれるジャンルでないことも影響してか，多くの研究があるわけではない。自動生成した STG のステージに類するコンテンツの評価手法の研究として，長による研究がある [5]。長は，ランダム生成した弾幕と呼ばれる，大量の敵の弾を発射するコンテンツの難度推定手法として，AI プレイヤーにプレイさせた結果を用いて推定する手法を提案しており，実験の結果，人間プレイヤーのプレイ結果に基づいて付与された正解難度を AI プレイヤーによってある程度推定できることを示している。推定できていないものの一部に関して，長は，AI プレイヤーが局所的な回避行動を取ってしまっていることにより生じたもので，その解決にはより大局的な判断を行う AI プレイヤーが必要であると述べている。

2.2 人間らしい AI プレイヤー

人間らしいリアルタイムゲームの AI プレイヤーの作成にもまた，様々なアプローチがある。藤井らは，「知覚のノイズ」「行動や認知の遅れ」「操作疲れ」など，人間プレイヤーであれば誰でも避けることのできない物理的な制約を，Q 学習に組み込むことにより，人間プレイヤーと遜色ない「人間らしさ」を持つマリオエージェントの作成に成功している [6]。STG においては，弾幕を人間がどのように認識しているかモデル化して，人間らしいエージェントを行う平井らの試みがある [7]。また，佐藤らは，人間の「大域的に安全そうな場所を目指す傾向」「細かく操作を変更しない傾向」などを取り込むことで人間らしい STG-AI プレイヤーの作成を試みている [8]。

3. アプローチ

本節では，まず，2D シューティングゲーム (STG) における望ましいステージの特徴についての考察を述べる。次に，そのような特徴を備えたステージを生成するための提案手法として，テスト AI プレイヤーを用いた生成検査法について述べる。

3.1 望ましいステージの特徴に関する考察

望ましいステージの特徴について考える前に，まずは望ましくないものについて考えてみると，以下のような特徴のステージが望ましくないと考えられる。

- どうプレイしてもクリアできない，難し過ぎる
- どうプレイしてもクリアできてしまい，易し過ぎる
- 終始動く必要があり，繁忙で，緊迫状態が連続している
- 終始動く必要がなく，退屈で，弛緩状態が連続している
- 繰り返しが多く，ワンパターンで，多様性に乏しい
- 繰り返しが少なく，まるで規則性がなく，雑然としている

これらの望ましくない特徴を反転させたものが望ましい特徴であると考えられる。しかし，ただ反転させたものは，総合的な特徴であるため，ステージを区間ごとに分けた際の部分的な特徴 (ミクロ) とステージの全体的な特徴 (マクロ) に細分化して考える必要がある。そのため，特徴を 3 つに分類した上で，ミクロとマクロの 2 視点に細分化すると以下ようになる。

- 難易度
 - ミクロ：難しい区間と易しい区間が混在
 - マクロ：被弾はするものの，クリア可能な，ほど良い難易度
- 緊張感
 - ミクロ：緊迫した区間と弛緩した区間が混在
 - マクロ：緩急はあるが，総合すると，ほど良い緊張感
- 多様性
 - ミクロ：似通った要素で構成された統一感のある区間が多様に混在
 - マクロ：統一感のある多様性により，総合すると，ほど良い多様性

本研究では，これらを備えたステージが望ましいものであると仮定して，敵のパターンを調整・配置することにより，そのようなステージの生成を目指す。

3.2 テスト AI プレイヤーを用いた生成検査法

提案する生成検査システムの全体像は以下の図 1 に示す。

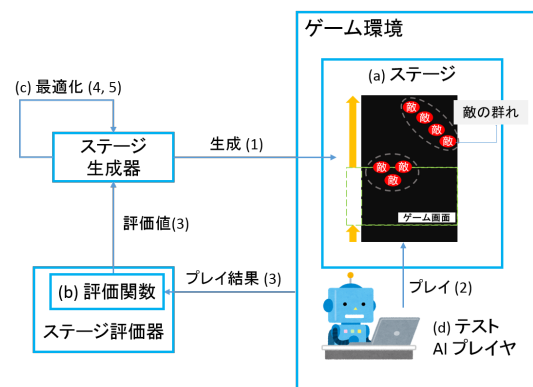


図 1 生成検査システムの全体像

システムは以下のような流れとなっている。

- (1) ステージ生成器がランダムに初期化したパラメータ列からステージを生成する
- (2) 生成したステージをテスト AI プレイヤにプレイさせる
- (3) (2) のプレイ結果（残り体力数や自機の行動履歴など）をステージ評価器内の評価関数に入力し、ステージの評価値を算出する
- (4) (3) の評価値を元に遺伝的アルゴリズムを用いて、次世代のステージを生成する
- (5) 一定世代数に到達するまでは、(2) に戻り、最適化を試みる

システムには以下のような手法や工夫を導入する。

(a) ステージ

ステージを構成する最小単位の要素は、単純に考えれば、各敵の出現・移動・攻撃パターンである。しかし、1つの敵ごとにランダムな配置や決定をしていては、3.1 節で述べたような「まるで規則性のない雑然としたステージ」が出来てしまいやすい。そこで、ミクロな統一感が生じやすいような工夫を加える。具体的には、出現に関する値（出現座標や時間）をずらした複数の敵パターンを、“群れ型パターン”と定義する。この群れ型パターンを最小要素とする、群れ型パターンの集合を“群れ型ステージ”と定義する。

(b) 評価関数

ステージがほど良い難易度・緊張感であるかを検査するためにヒューリスティックな評価関数を作成する。その入力には、ステージ情報を含む、AI プレイヤによるプレイ結果を用いる。評価には、プレイ結果の中の、難易度や緊張感や緊張感に関係するであろう統計量を用いる。例えば、区間毎の敵数や弾数、クリアに関わる統計量、無行動に関わる統計量などである。少し具体的には、各統計量が、統計量ごとに予め定めた理想とする値域からどれだけ離れているかの逸脱値を計算し、その逸脱値に対して負のペナルティを与えることで、その緊張感や難易度が適しているかどうかの評価を試みる。

(c) 最適化

最適化には、生物の進化に関する法則や理論に着想を得た最適化アルゴリズムである、遺伝的アルゴリズム (GA) [2] を用いる。GA は、対象データの一要素を遺伝子とみなし、遺伝子进行操作（交叉・変異）することで新たなデータを生成し、改善を試みる手法である。本研究では、対象データであるステージの構成要素、敵の群れのパターンを遺伝子とみなし、それを操作することでステージの改善を試みる。

(d) テスト AI プレイヤ

テスト AI プレイヤが人間離れした挙動を行うようだ

と、人間プレイヤーにとっては難し過ぎるステージを易しいものだと誤判定してしまうなど、ステージの難易度推定に支障を来す恐れがある。そのため、人間らしい工夫を盛り込む必要があると考える。

また、各部詳細に関しては第 5 節にて述べる。

4. ゲーム環境

提案システムで用いるためのゲーム環境として、図 2 のような 2D シューティングゲーム (STG) のプラットフォームを作成した。プラットフォームは研究用であるため、そのゲーム性と構成要素はシンプルであるべきだと考え、構成要素は 4 種類のオブジェクト、自機・自弾・敵機・敵弾に限定した。それにより、そのゲーム性を「自機の行動による敵機の攻撃の回避」と「自弾の発射による敵機の撃墜」の 2 点にのみ注力するものとした。ゲームの終了条件は、ゴール到達によるゲームクリアと、自機の死亡によるゲームオーバーである。

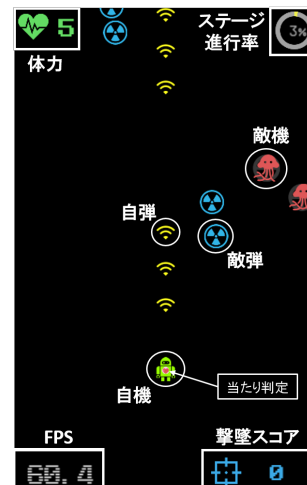


図 2 STG プラットフォーム

5. アプローチの各部詳細

本節では、3.1 節にて考察した 2D シューティングゲーム (STG) における望ましいステージを生成するための手法として、3.2 節にて提案した、遺伝的アルゴリズムと人間らしい AI プレイヤによる生成検査法の各部詳細について述べる。

5.1 ステージデータ構造・表現

本研究で使用する STG において、ステージは敵の配置と挙動のみを設定するものとする。加えて、本研究におけるステージは次節で述べる遺伝的アルゴリズム (GA) [2] により最適化を行う対象である。そのため、ステージを遺伝子の集合として扱えるよう、ステージのデータ構造は「敵の配置や挙動、種類に関する複数のパラメータ」であ

る“パターン”の配列とした。GAで扱う際には、このパターンが1つの遺伝子として扱われる。

まず初めに、単純なものとして「1体の敵のみの配置や挙動、種類に関する複数のパラメータ」を遺伝子とするステージを考え、これを“単純型ステージ”と定義することとした。この単純型ステージに、3.1節で述べた望ましい特徴の1種であるミクロな多様性（統一感）を制御するための工夫として、出現時間や位置にのみ多少のずれのある敵の集団、“群れ”を導入した。群れを次の2種類に分類することとした。

- 編隊型：複数の敵が、ある基準位置の周辺に、同時に出現するもの
- 連鎖型：複数の敵が、ある基準位置・時間に対して一定間隔のずれを生じさせながら出現するもの

この2種類の群れを表現する「群れの配置、挙動、種類に関する複数のパラメータ」を遺伝子とするステージを“群れ型ステージ”と定義した。群れ一つにつき1~9体の敵が内包され、移動や攻撃のパラメータはそれぞれ共通のものが設定される。

5.2 遺伝的アルゴリズム

遺伝的アルゴリズム（Genetic Algorithm, GA）[2]は、生物の進化を参考にした確率的最適化アルゴリズムである。本研究では、3.2節及び5.1節で述べたように、ステージを“個体”、群れに関するパラメータを“遺伝子”テストプレイヤによる評価値をそのステージの“適応度”、としてGAを行う。テストプレイヤの実装は5.3節、適応度の計算は5.4節に詳述するとして、本節ではそれ以外の部分について述べる。

交叉オペレータは、以下のように行うこととした。

- 一点交叉：ランダムに決めた1つの群れから後ろの各群れを交叉させる
- 二点交叉：ランダムに決めた2つの群れの間の各群れを交叉させる
- 一様交叉：約50%の確率で選択した各群れを交叉させる

群れ同士の交叉は、“群れそのもの”を交換することで行うこととした。すなわち、例えば8体で構成された親の群れから、5体だけが子に引き継がれるといったことはなく、元の群れの全パラメータ（出現時間や出現座標、挙動、速度など）を受け継いで、群れの組み合わせだけが多様性変わって子が生成される。

突然変異オペレータは、ステージ x が持つ m 個の群れから、約 $0.05 \times m$ 個の群れを取り出して、完全に再初期化することで行う。例えば、40個の群れを持つ個体から、38個はそのままに、2個を完全に再初期化する。交叉オペレータ・突然変異オペレータともに、群れを崩さないようにした。

世代交代モデルとしては、MGG（minmarl generation gap）[9]+best2 と呼ばれる単純なモデルを用いた。以下に、本研究で用いたパラメータとともに、その手順を示す。

- (1) n 個の個体をランダムに生成する。各個体は、最小10個、最大40個の群れを持つように初期化する。
- (2) 全個体を、テストAIプレイヤによって評価する。
- (3) 個体群の中から、ランダムに異なる2つの個体（親） $p1, p2$ を選ぶ。
- (4) $p1, p2$ から、10個の子個体 $\{c1, c2, \dots, c10\}$ を、交叉オペレータによって生成する。交叉オペレータは、一点交叉1回、二点交叉2回、一様交叉を2回行うこととした。1回の交叉オペレータで、子個体は2つ生成される。
- (5) それぞれの子個体に対して、10%の確率で突然変異オペレータを施す。
- (6) 全ての子個体を評価する。
- (7) 親と子の集合 $\{p1, p2, c1, c2, \dots, c10\}$ から、最も評価値の高かったもの2つを選んで、 $p1, p2$ の代わりに個体群に戻す。これを1世代と呼ぶ。
- (8) 一定世代に達したらGAを終了する。そうでなければ(3)に戻る。

このモデルは、実装が簡単で、解の質が確率的に劣化することがない利点があり、しばしば用いられる。

5.3 テスト用AIプレイヤ

遺伝的アルゴリズムによるステージの最適化には、目的関数となるステージ評価値を定める必要があり、それにはステージを「ある程度人間らしく」自動プレイするAIが必要になる。AIプレイヤに人間らしく振舞わせる研究はさまざまな方法がありえるが、本研究では佐藤らのアイデア[8]、「安全そうな場所を目指す」「細かく操作を変更しない」といった方法を援用する。

5.3.1 基本的なアルゴリズム

人間らしさのための工夫は後述するとして、まずは基本的な流れを説明する。AIプレイヤは毎フレーム、自機の状態や周囲の情報を観測し、移動方向を出力する。本研究では弾を撃たない設定としている。基本的なアルゴリズムは、未来 n フレーム先までの「全ての取りうる行動の組み合わせ」について、将来を予測し、最も安全そうな将来に導く最初の1フレーム分の行動を選ぶ、というものである。

STGでは多くの場合画面外から画面内に敵が登場するため、敵や弾の回避に余裕がある場合は画面中央に位置しておくのが安全でもあり、また人間らしくもある。これらを手続き的にまとめると、概要としては以下の手順で行動を定める。

- (1) 毎フレーム、観測可能な状態を与えられる。
- (2) もし画面に敵や弾がない、あるいは遠い弾のみ存在

するのであれば、将来の安全のために、画面中央に向かう。

- (3) 弾のみが存在するがそれが自機より遠いのであれば、同様に画面中央に向かう。
- (4) 敵が存在するがそれが遠い場合は、敵が弾を射出する可能性があるため、その場にとどまる。
- (5) それ以外の比較的危険な状況では、 n フレーム先まで将来予測をして、回避のための行動を取る。

5.3.2 人間らしい行動のための工夫

前節の基本アルゴリズムはかなり短い未来を正確に予測して次の行動を選ぶものであり、敵の弾を危険が迫ってからぎりぎり回避したり、弾が多ければ1フレーム単位で細かく操作して回避するような挙動が見られる。これは人間プレイヤーの挙動とはかなり異なる [8]。人間プレイヤーは、正確で素早い操作が苦手である代わりに、大局的な判断によって余裕を持った回避を行うことが多い。そこで本研究では、人間らしい挙動ひいては適切なテストプレイを実現するために、佐藤らの手法を参考に以下の3つのオプションを適用する。

【オプション1：ざっとした先読みと行動】

人間には、1フレーム単位で行動を小刻みに変えることはできない。そこで、10フレームの間、同じ行動を取り続けるオプションを設けた。同時に、未来予測も、10フレーム×深さ2まで行うこととした。これにより「正確ではないが、より長期的で大局的な」制御を狙った。

【オプション2：画面の端に近づき過ぎない】

人間プレイヤーは、敵や弾など危険なものからは距離を取り、同時に回避可能なスペースの少ない画面端も危険であると認識する傾向にある。そこで、20フレーム先までを探索した際に「生き残ったかどうか」ではなく、「敵・敵の弾・画面端から遠いか」を考慮にいれ、最も安全なルートを選ぶオプションを設けた。

【オプション3：ぎりぎりでの回避を抑制】

AIプレイヤーは1ドット単位で自機や敵を認識できるので、弾と弾とのわずかな隙間を縫う回避が頻繁に見られる。これは普通の人間プレイヤーとは異なるので、その抑制のため「自機の当たり判定サイズが2倍になったものとして探索する」オプションを設けた。

5.4 評価関数

GAでステージを最適化するには、どんなにアルゴリズムが優れていて最適解が出せようとも、その評価関数が「現実に良いものをきちんと高評価できる」ように設計されていなければならない。3.1節で議論したように、望ましいステージには、以下の6つの要素が求められると考える。

- (a) 難しい区間と易しい区間が混在している。
- (b) 全体として、被弾はするがクリアは可能である。

- (c) 緊迫した区間と弛緩した区間が混在している。
- (d) 全体として、ほど良い緊張感がある。
- (e) 部分的には統一感がある。
- (f) 全体的には多様である。

このうち、(e)については、評価関数ではなくて、5.1節で述べた「群れ」のアイデアによって接近することにした。(f)については今回は考慮しなかった。

(a)~(d)については、各ステージ x ごとに、AIプレイヤーによるテストプレイを行い、評価関数値 $f(x)$ を式(1)のように計算することにした。ステージまたはテストプレイの結果から、いくつかの統計量 $\{s_i\}$ が計算できる。このそれぞれは (a)~(d) のいずれかに関連する統計量であって、「この統計量 s_i は、範囲 $[min_i, max_i]$ に入ってるべきである」という理想範囲を指定される。この理想範囲から逸脱している場合、その逸脱値の2乗に重み w_i をかけた値が、評価関数値から減算される。例えば、重みが2のある統計量が10から20の範囲であるべきなのに7であったならば、その部分での評価値は-18となる。

$$f(x) = \sum_{i=0}^n \begin{cases} -w_i(s_i - max_i)^2 & (s_i > max_i) \\ -w_i(s_i - min_i)^2 & (s_i < min_i) \\ 0 & (otherwise) \end{cases} \quad (1)$$

以下に、統計量のリストを記述する。

- ライフ数 LifeNum：テストプレイ後の残り体力数であり、要求(b)に関連する。これが大きすぎれば全体に簡単すぎ、小さすぎれば全体に難しすぎる。
- ヒヤリ数 HiyariCount：テストプレイ時に、敵または弾が、自機の当たり判定の5倍以内（自機の見かけの大きさと同程度の範囲内）を通った回数、つまり「ヒヤリとした回数」であり、(d)や(b)に関連する。
- ニアミス数 NearmissCount：同様、当たり判定の2倍以内を通った回数。被弾直前レベルのニアミスを起こした回数であって、同様(d)や(b)に関連する。
- 無行動率 NoMoveRatio：テストプレイ時に、停止を選んだ回数の割合。主に(d)に関連する。動かなくてよいということは危険がない状態であるということである。これが大きすぎれば全体に退屈であり、小さすぎれば慌ただしすぎるステージである。
- 最大連続無行動数 MaxNoMoveCount：テストプレイ時に、連続して停止していた最大フレーム数。主に(c)や(d)に関連する。NoMoveRatioだけでは、「最初70%の時間停止していたが、最後30%はずっと動いていた」といった極端なものを低く評価しにくい。そのため、適度な休憩が含まれているかどうかをこの統計量で測る。

- 区間最大敵数 EnemyNum : 実際には 11 次元の統計量である。全ステージのフレーム時刻を 11 分割した際に、そのそれぞれの区間において、「画面上最大何匹の敵が登場したか」を表すものである。(a) や (c) や (f) に関連している。具体的には、“最初は様子見のような敵の群れ、続いて本格的な群れ、休憩を置いて、最も激しい群れが来る”といった、ストーリー性やメリハリのあるステージを作るための統計量である。
- 区間最大敵数 BulletNum : 実際には 7 次元の特徴量である。EnemyNum とほぼ同様の意味を持つ、それぞれの区間において「画面上最大いくつの弾が登場したか」を表すものである。EnemyNum だけでは弾数をコントロールできないので導入したものである。

6. ステージ生成実験・評価

本節では、まず、3 節及び 5 節で述べた提案システムにより、望ましい特徴を備えたステージの生成を目的として、実際にステージ生成実験を行ったため、設定と結果、考察について述べる。その後、生成したステージの評価のため行った簡単な被験者実験について述べる。

6.1 設定

本実験は、以下のような設定で行った。

【遺伝的アルゴリズム】

世代数は 100, 1 世代あたりの個体数は 400 で、9 試行を行った。MGG-best2[9] における 2 つの親個体から生成される子個体数は 10 個としたため、1 試行あたりの評価回数は約 $100 \times 400 \div 2 \times 10 = 200000$ 回となる。

【ステージ】

生成するステージの時間長は 60 秒、3600 フレームとし、3600 フレーム経過時に必ず最後の敵が出現するものを生成するようにした。また、5.2 節でも述べたが、初期ステージの群れ数は [10,40] の範囲内でランダムに生成することとした。群れ 1 つにつき敵は [1,9] 体なため、ステージの敵数の値域は [10,360] となる。これは、評価関数で用いる区間最大敵数の合計数を考慮し決定した。

【テスト AI プレイヤ】

前節でも述べたように、評価関数で用いる各統計量の計算には、基本的には、3 つの人間らしい工夫のためのオプションを盛り込んだテスト AI プレイヤによるプレイ結果を用いた。ただし、区間最大敵数・弾数の計測に関してのみ、自機無敵モード・無行動 AI プレイヤによる追加プレイの結果を用いることとした。すなわち、1 ステージをテスト通常モード・AI プレイヤと、無敵モード・無行動 AI プレイヤにより 2 回プレイさせ、その結果を用いてステージを評価することとした。

6.2 結果

実験は、CPU: Intel Core i7-4799, RAM: 8GB の PC で行い、1 試行の平均処理時間は 2:55:16[h:mm:ss] であった。実験の結果、9 試行中 1 試行で最高評価値 0 への収束、すなわち、用いた評価関数により定められた最適なステージの生成を確認した。これは大別 7 種、細別 23 個の統計量を全て理想の範囲に収めることが出来たことを示す。また、各試行の世代内最大評価値の平均値は -25 だった。これは、敵数の逸脱であれば 2 個程度、弾数の逸脱であれば 3 個程度のずれでしかないことを示す。また、最高評価値 0 を確認した試行を 1 例として、世代内評価値 $score$ を $1 - score$ に変換し、その推移を $[10^0, 10^8]$ の範囲でログスケール表示したものを図 3 に示す。

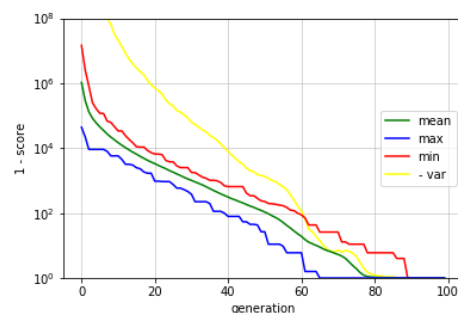


図 3 世代内評価値の推移例：平均・最大・最小・分散

このように、0 世代目が最大 10^5 弱、最小 10^7 強ほどだったものが、60 世代を超えた辺りで最高値 10^0 が発見され、90 世代手前には 10^0 に完全に収束していることがわかる。

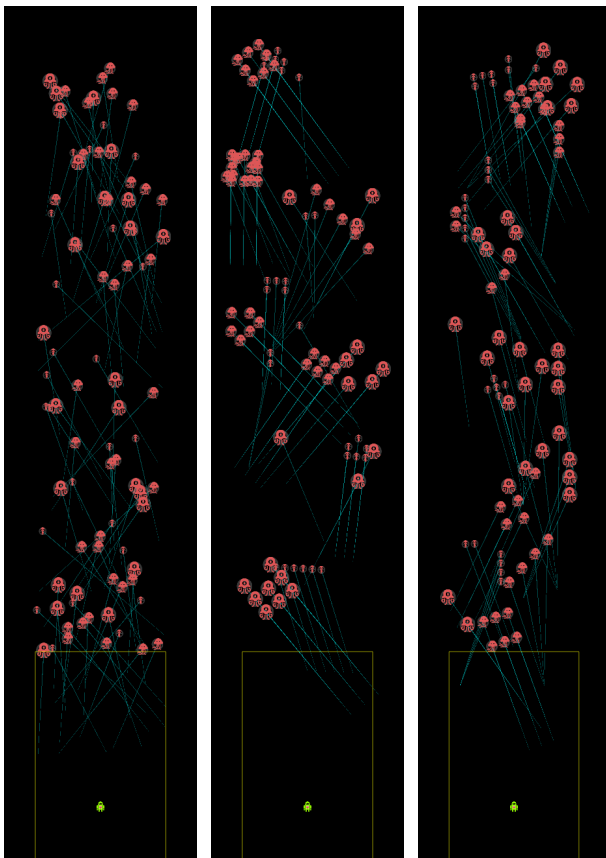
表 1 評価値の内訳：9 試行内最低評価値ステージ

統計量	理想範囲				部分評価値
	最小値	最大値	実際値	逸脱値	
ライフ数	2	3	3	0	0
ヒヤリ数	10	30	17	0	0
ニアミス数	0	5	3	0	0
最大連続無行動数	60	150	130	0	0
無行動率	0.4	0.6	0.579	0	0
区間最大敵数					
0000-0059[f]	0	0	2	2	-20
0060-0599[f]	6	10	7	0	0
0600-0779[f]	0	2	3	1	-5
0780-1379[f]	9	15	9	0	0
1380-1559[f]	0	2	2	0	0
1560-2159[f]	8	12	8	0	0
2160-2339[f]	0	2	2	0	0
2340-2939[f]	12	18	12	0	0
2940-3119[f]	3	7	6	0	0
3120-3419[f]	10	14	13	0	0
3420-4799[f]	16	24	19	0	0
区間最大弾数					
0000-0119[f]	0	0	4	4	-48
0120-0779[f]	05	15	15	0	0
0780-1559[f]	15	25	17	0	0
1560-2339[f]	10	20	15	0	0
2340-3119[f]	20	30	28	0	0
3120-3419[f]	15	25	24	0	0
3420-4799[f]	30	50	42	0	0
合計評価値					-73

9 試行内最低評価値における評価値の内訳を表 1 に示した。最低評価値のものには、敵数や弾数に逸脱値がいくつか見られるが、それも 3 項目計 7 個と僅かであり、理想範囲内に収まってこそいないものの、内訳自体からは十分満足できる結果であると言える。

6.3 考察

図 4 に、提案手法により生成した 9 試行内最高評価値ステージと、最高評価値ステージの敵数 (88 体) に合わせてランダム生成した単純型、群れ型ステージそれぞれの全体像を示した。本節ではそれぞれの手法についての比較を行いつつ、本実験による生成結果についての考察を述べる。



(a) ランダム単純型 (b) ランダム群れ型 (c) 提案手法群れ型

図 4 ランダム生成ステージと提案手法最高評価値ステージの全体像

敵の数にもよるが、単純型ステージ (a) はミクロ・マクロ共に雑然としており統一感が感じられない。また、敵の出現の推移にメリハリがなく、規則性なく出現する敵から弾を浴びせ続けられるような状況が見られた。このように、人間プレイヤーにとっては忙しすぎ、クリア困難なものとなってしまった。

一方、群れ型ステージ (b) はミクロな統一感が生じていることがわかる。また、メリハリが生じやすく、敵の出現に規則性があるため、単純型 (a) に比べ敵や敵弾の動きの予測が容易になっている。一方で、敵が 8 秒間も出てこな

かったり、自機に干渉して来ない疎な区間や、大量の敵が固まって出てくる密な区間が多く見られた。特に後者は、同じパラメータを持つ敵が同時に出現するため、一体の敵が多数の弾を吐き出す場合、大量の弾により逃げ場がなくなり被弾するケースがよく見られた。つまり、メリハリはあるが、その落差が大きく、理不尽・退屈な箇所が発生しやすい問題がある。

最後に、提案手法ステージ (c) を実際に見ると、敵や敵弾が出て来すぎることはなく、終始忙しいままではなく適度に余裕時間が与えられるものとなっていた。また、敵の弾も無駄なものは減り、危機的状況は少ないものの、適度な緊張感はあるものとなっていた。

以上のように、まず、ステージに群れを導入することでステージのミクロな多様性を制御し、AI プレイヤによるプレイ統計量 (ライフ数、ヒヤリ数、無行動率等) と区間最大敵数・弾数を一定範囲内に収めることにより、難易度や緊張感のある程度制御できたと考える。

6.4 被験者実験：生成ステージ評価

生成したステージが適切な面白さや難しさになっているかどうかの評価のため、簡単な被験者実験を行った。まず、以下の 4 種類の生成ステージをそれぞれ 2 つずつ用意した。

- (1) ランダム生成した単純型ステージ：敵数は (3),(4) の平均値付近の値を指定。RndSimple.
- (2) ランダム生成した群れ型ステージ：敵数は (3),(4) の平均値付近の値を指定。RndSwarm.
- (3) 限られた統計量のみを考慮した評価関数により生成した群れ型ステージ：LifeNum, EnemyNum, BulletNum のみ。最適化実験を行い、最高評価値 0 のものを無作為に 2 つ選択。LEBSwarm.
- (4) 全ての統計量を考慮した評価関数により生成した群れ型ステージ：前節までで述べた、提案手法による実験の結果生成されたものから、最高評価値 0 のものと、次点の -5 のものを選択。AllSwarm.

用意した各ステージは、生成実験で用いたものと同じ設定の AI プレイヤにランダムな順でプレイさせた。その様子を被験者 7 人に同時に見てもらい、各プレイ終了後に、ステージが面白そうかや、難しそうかを 5 段階 (1-5) で絶対評価してもらった。また面白さと難しさに関する回答項目はそれぞれ、面白くなさそう (1 点) / あまり面白くなさそう / どちらとも言えない / 少し面白そう / 面白そう (5 点), および、簡単そう (1 点) / 少し簡単そう / どちらとも言えない / 少し難しそう / 難しそう (5 点) とした。

本研究において目標とするステージは、「プレイしていて面白く感じられる程よい難易度のステージ」である。そのため、上記の回答が、面白さに関しては 5 に近いほど良い評価となり、難しさに関しては極端でない値、つまり 2-4 の間で 3 に近いほど良い値となる。

被験者は7人だが、伝達ミスにより、難しさに関しては6人分の結果しか得ることが出来なかった。次の表2に手法ごとの回答結果を示す。

表 2 ステージ評価実験結果：面白さと難しさの手法毎の平均

	(1)RndSimple	(2)RndSwarm	(3)LEBSwarm	(4)AllSwarm
面白さ	2.43	2.36	1.79	3.86
難しさ	5.00	4.25	2.08	3.08

ランダム生成した単純型 (1) と群れ型 (2) の平均難しさは、5.00 と 4.25 であり、これらは難し過ぎることを示す。単純型は 6.3 節で述べたように全方位からの弾の軌道にさらされ、人間にはクリアが困難な難易度になっているため、この評価となったと考えられる。一方で、平均面白さは、2.43 と 2.36 となり、群れを導入したことのみでは面白さの改善ができていないことがわかる。群れ型には、6.3 節で述べたような退屈・理不尽の激しい落差が生じやすく、そのことがあまり面白くないとの評価に繋がったのではないかと考える。あまり両者に差がない原因としては、見てもらったものの中では単純型の割合が少なかったことで、物珍しさのようなものが働いた可能性も考えられる。

また、ライフ数と敵数、弾数のみを考慮して生成したものの (3) は、平均面白さ 1.79 となり、ランダム単純型 (1)・群れ型 (2) の両方 (2.43, 2.36) に対して、面白さの点で劣ってしまっていた。平均難しさは 2.08 と、ランダム群れ型の 4.25 から大きく下がっており、これは簡単すぎる、適切な難易度でないことを示す。このことから、ライフ数や敵数・弾数を制御したことで、ランダム群れ型における理不尽な箇所の抑制には成功した一方で、退屈な箇所が目立つようになってしまったのではないかと考える。そのように考えると、理不尽さよりも退屈さの方が、人間プレイヤーにとっての面白さの評価に対して負の影響を与えやすいと考えることもできるかもしれない。

最後に、提案手法 (4) は、平均面白さと難しさが 3.86, 3.08 となり、ランダム単純型 (1)・群れ型 (2) に対し、面白さ、難易度共に良い結果が得られた。この難しさ 3.08 という評価は、実際にプレイした結果ではなく、プレイを見た印象ではあるが、中程度の適切な難易度であることを示している。これにより、本手法の有効性を確認することができた。

7. おわりに

本研究では、2D シューティングゲームにおいて、プレイしやすい難易度かつプレイしていて楽しいステージの評価手法および生成手法の提案を行った。まず、望ましいステージの特徴を考察し、そのような特徴を持ったステージを生成するために、遺伝的アルゴリズムによるステージの生成と、人間らしい工夫を盛り込んだ AI プレイヤによる検査を行う、ステージ生成検査システムを構築した。

生成実験の結果、評価関数によって定めた望ましいステージの生成を確認できた。最後に、生成ステージの評価を目的として行った被験者実験により、ランダムに生成したステージよりも有意な結果を得ることが出来、本手法の有効性を確認することが出来た。ただし、AI プレイヤに盛り込んだ人間らしさの工夫によるステージ生成への有効性は検証出来ていない。

今後の展望としては、人間らしさの工夫なしの AI プレイヤによる生成実験を行い、工夫有りの場合の結果と比較することで、人間らしさの有効性の検証を行いたい。また、AI プレイヤに射撃行動を行わせることで、敵機の撃墜を目指す上での、難易度や面白さが適切なステージの生成を試みたい。

参考文献

- [1] Hendrikx, M., Meijer, S., Velden, J. and Iosup, A.: Procedural Content Generation for Games: A Survey, *ACM Transactions on Multimedia Computing, Communications, and Applications*, Vol. 9 (2013).
- [2] 北野宏明: 遺伝的アルゴリズム, 人工知能学会誌, Vol. 7, No. 1, pp. 26-37 (1992).
- [3] Volz, V., Schrum, J., Liu, J., Lucas, S. M., Smith, A. M. and Risi, S.: Evolving mario levels in the latent space of a deep convolutional generative adversarial network, *GECCO '18: Proceedings of the Genetic and Evolutionary Computation Conference*, p. 221-228 (2018).
- [4] Dahlskog, S. and Togelius, J.: A multi-level level generator, *2014 IEEE Conference on Computational Intelligence and Games*, pp. 1-8 (2014).
- [5] 長 健太: 弾幕生成エンジンを用いた AI プレイヤーによる自動生成コンテンツ評価手法の提案 (特集 インディーズゲーム), デジタルゲーム学研究, Vol. 5, No. 1, pp. 51-56 (2011).
- [6] 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ゲームエージェントの「人間らしい」振舞いの自動獲得, 情報処理学会論文誌, Vol. 55, No. 7, pp. 1655-1664 (2014).
- [7] 平井弘一, Reijer, G.: 弾幕の認識に人間の視覚特性を取り入れたシューティングゲーム AI の研究, ゲームプログラミングワークショップ 2016 論文集, No. 2016, pp. 158-161 (2016).
- [8] 佐藤直之, Sila, T., Luong, H. P., 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, Vol. 2016, pp. 57-64 (2016).
- [9] Satoh, H.: Minimal Generation Gap Model for GAs Considering Both Exploration and Exploitation, *Proceedings of IIZUKA '96*, pp. 494-497 (1996).