修士論文

遺伝的アルゴリズムを用いた 2D シューティングゲームのステージ生成

1610201　吉田 友太

主指導教員　池田 心
審査委員主査　池田 心
審査委員　飯田 弘之
白井 清昭
長谷川 忍

北陸先端科学技術大学院大学
先端科学技術研究科
（情報科学）

令和 2 年 2 月

概要

近年，ゲームにおける人工知能の研究は盛んに行われている．中でもコンピュータゲームプレイヤ（以下，AI プレイヤ）の強さに関する研究では，ハードウェアやアルゴリズムの進歩も相まって目覚ましい成果が挙げられており，多くのゲームにおいて AI プレイヤは人間プレイヤと同等ないしそれを凌駕するほどの強さに到達している．そのため，ゲームにおける人工知能分野において，今後は特に AI プレイヤの強さ以外に着目した研究の重要性が増すと考える．着目する性質には，“ 人を楽しませる ” や “ 人に教える ” などが挙げられる．本研究ではその内の，人を楽しませることに着目し，AI プレイヤを用いた Procedural Content Generation に関する研究を行う．

Procedural Content Generation とは，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のことであり，大規模なゲーム作成の省力化などの目的で注目されている．ゲームにおけるコンテンツには，グラフィック・音楽・ダンジョンの構造など様々なゲームの構成要素が含まれるが，本研究では 2D シューティングゲーム（以下，STG）における “ ステージ ”（敵や障害物の配置）に着目する．

ゼビウスなどに代表される STG は，プレイヤが自機を操作して敵とその攻撃や障害物を回避しながら遠距離武器により敵の撃破を行うゲームであり，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している．また，従来の 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ることができる ” ように，人手によりデザインされたものが固定数だけ用意されていることが多い．このようなステージは，ゲームデザイナーの趣向や創意工夫が凝らされた職人芸の賜物でありプレイしていて楽しいものとなっている．しかし一方で，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層も存在すると考えられる．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と考えられる．

そこで本研究では，“ 人間プレイヤにとってのステージの難易度や楽しさはどう推定できるのか ” 及び “ そのためのテスト AI プレイヤはどのような特徴を備えていなければならないか ” を解明することを目的とする．その上で，人間プレイヤにとっ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生成システムを構築することを目標とする．

まず，望ましいステージの特徴について考察を行い，“ 難易度 ”，“ 緊張感 ”，“ 多様性 ” に関する 3 種に大別した上で，これらを部分的な場合と全体的な場合，ミクロとマクロの 2 視点から細分化し，計 6 つの意味での望ましい特徴があると仮定した．生成手法には，ランダムに初期化したステージを AI プレイヤが検査し，

その評価値を最大化するようにステージの調整を行う遺伝的アルゴリズムを用いた．検査に用いるテスト AI プレイヤが人間離れしていた場合，生成物の難易度推定に支障を来す恐れがあるため，テスト AI プレイヤには“人間らしさ”が必要だと考えた．そのため，テスト AI プレイヤには，“連続フレーム先読み・同一行動”，“敵や画面端との位置関係を考慮して探索”，“探索時の当たり判定を大きくすることで安全回避”を行う AI プレイヤを採用した．

ステージはミクロな多様性制御のために，出現時間や位置に多少のずれのある敵集団，“群れ”を最小要素とした．この群れを遺伝的アルゴリズムにおける遺伝子として扱うこととした．生成したステージの評価には，テスト AI プレイヤのプレイ結果を入力とするヒューリスティックな評価関数を作成し，その評価値を用いた．評価関数には，難易度や緊張感制御のため，クリア状況や区間毎の敵や弾の数，プレイヤの無行動率などについての適切な範囲を定める項目を段階的に追加し，最適化を行った．結果，大別した 7 個の項目（2 項目を細分化すると合計 23 個の項目）の全てにおいて，その値が定めた適切な範囲の中に納まるものが得られた．そのため，作成した評価関数が目標とするステージの生成に成功したと言える．

実験は，自作の単純化した STG のプラットフォーム上で行った．被験者実験と，AI プレイヤによる最適化の両方に対応できるように，リアルタイムの入出力・表示装置と，画面表示なしでの高速評価モードを備えたものとした．

最後に，本手法の評価のため，簡単な被験者実験を行った．本手法により生成したステージや単純にランダム生成したステージなどをそれぞれ 2 個ずつ，計 8 個のステージを同じ設定の AI プレイヤがプレイしているところを，被験者 7 人に見てもらい，面白さと難しさを 1〜5 の 5 段階で評価してもらった．その結果，単純にランダム生成したものの平均点が面白さ 2.43・難しさ 5.00 であったのに対し，本手法による生成ステージの平均点は，面白さ 3.86・難しさ 3.08 となり，面白さ・難しさ共に，本手法による有効性を確認することができた．

Abstract

Research in artificial intelligence (AI) for games has been popular in past decades. Among the research, creating strong computer game players, or AI players, has achieved remarkable results. The achievements were demonstrated by AI players' superhuman levels of plays in many games. Thus, we expected the importance to increase for other research in game AIs than creating strong players. More specifically, this research focused on entertaining human players and studied procedural content generation (PCG) with the use of AI players.

PCG can be used to create game content massively and reduce the costs of making games by applying optimization or machine learning algorithms. Thus, it attracted attention from both academia and industry. The term "game content" covers various components in games such as graphics, music, and dungeons. In this research, we targeted at "stages" (i.e., the arrangements of enemies and obstacles) for a "shoot 'em up" game (STG).

In classical STGs such as Xevious, players control their spacecraft to attack enemies by ranged weapons while avoiding enemies' attacks and obstacles. The interestingness and the difficulty of stages crucially depend on the arrangements of enemies and their patterns of attacks. Usually, STG stages are elaborated by human experts and thus demonstrate their preferences and creativity. The design can be considered as a kind of art; however, the numbers of stages that can be made are limited. As a result, players may get bored easily since they can find good strategies to play after practice even for the stages thought difficult at first glance. Playing in new stages every time, which involves randomness, is desired by some players and is also important for entertainment. Even so, just randomly generating stages is impractical since many improper stages may be created. By improper stages, some examples are those too difficult for players to clear, those too easy that players can clear no matter how they play, and those without various patterns for enemies' actions but keep players busy all the time.

This research aimed to clarify "how to estimate the difficulty and the enjoyment of STG stages to human players" and "what features should test AI players have to achieve this." The goal was to generate stages that have proper difficulty such that human players can enjoy playing.

First, to evaluate STG stages, we proposed six features that were composed of three factors and two viewpoints. The three factors were difficulty, tenseness, and diversity. The two viewpoints were macro and micro, which represented an overall evaluation of a stage and evaluations of sections of a stage, respectively. We then applied a genetic algorithm (GA) to optimize randomly initialized stages so that the evaluations from test AI players were maximized. To make the generated stages suitable for human players, the test AI players should be "human-like." For

this purpose, the test AI players performed the same actions for several continuous frames, did tree searches with considering the positional relations to enemies and borders of the screen, and tried to avoid possible enemies and bullets by enlarging the ranges for collision judgment.

To control the micro level of diversity of STG stages, we grouped enemies so that those in the same group slightly differed in the positions and the appearance time. Groups were set as the minimal composition of enemies and were represented by genes in GA. The generated stages were evaluated by test AI players, where the play results were inputted into a heuristic fitness function, and the values were used as evaluations. Furthermore, to control difficulty and tenseness, we defined appropriate ranges for some additional criteria, such as indicators of how the players cleared the stages, the numbers of enemies and bullets in each section, and the ratios of time that the players had no actions. For comparison, we increasingly included the criteria for the fitness function. From the results, for all of the seven proposed criteria (further divided into 23 by segmenting two of the criteria), the values fell into the defined ranges. The results demonstrated our success in generating stages where the designed fitness function was properly reflected.

The experiments were conducted on a simplified STG platform made by ourselves. To enable both doing subject experiments and speeding up AI players' plays, we prepared two modes, where one had real-time inputs, outputs, and screen display while the other had no screen display for high-speed evaluations.

Finally, to evaluate our approach, a simple subject experiment was conducted. We asked seven human subjects to watch videos where an AI player played in eight different stages. The stages were either randomly generated or generated by our approach. We displayed two each of the stages from different approaches and asked human subjects to rate the interestingness and the difficulty of each stage in five-grade evaluation (1, 2, 3, 4, and 5). From the results, randomly generated stages received interestingness of 2.43 and difficulty of 5.00 on average. In contrast, stages generated by our approach received interestingness of 3.86 and difficulty of 3.08 on average. The experiment confirmed the effectiveness of our approach.

目 次

図 目 次

表 目 次

第1章　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いる．中でもコンピュータゲームプレイヤ（以下，AI プレイヤ）の強さに関する研究では，ハードウェアやアルゴリズムの進歩も相まって目覚ましい成果が挙げられている．例えば，ボードゲームにおいては，チェスで 1997 年に IBM 社の Deep Blue が，囲碁で 2017 年に Google DeepMind 社の AlphaGo が，それぞれの世界チャンピオンである Garry Kasparov, 柯潔に勝利している [1, 2]．また，より複雑なリアルタイムストラテジーゲームである StarCraft II においては，2019 年に Google DeepMind 社の AlphaStar がゲーム内ランキングの上位 0.2%に入った [3]．このように，多くのゲームにおいて AI プレイヤは人間プレイヤと同等ないしそれを凌駕するほどの強さに到達している．そのため，ゲームにおける人工知能分野において，今後は特に AI プレイヤの強さ以外に着目した研究の重要性が増すと考える．着目する性質には，“ 人を楽しませる ”や“ 人に教える ”などが挙げられる．本研究ではその内の，人を楽しませることに着目し，AI プレイヤを用いた Procedural Content Generation[4]（以下，PCG）に関する研究を行う．

PCG とは，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のことである．主にゲームにおいて用いられており，ゲーム情報学研究の一分野となっている．ゲームにおけるコンテンツには，グラフィックなどのゲームの素材，ダンジョンなどのゲーム内におけるデータの形状，難易度などのゲームのデザインまで様々なゲームの構成要素のことが含まれる．これらのコンテンツは基本的にゲーム製作者の手によって作成されている．そのため，PCG を利用しコンテンツを生成することには，人手による作成コストの削減や人手では限界のあった大量生成などの意義がある．

ゲームにおけるコンテンツの一つには“ ステージ ”と呼ばれるゲームプレイの区切りとなる構成要素も存在する．ステージには敵や障害物が配置されており，プレイヤのゴール到達やボスの撃破，全敵の退場などの条件を満たすことでクリアとなる [5]．ステージも他のコンテンツと同様に PCG の研究対象となっているが，ゲームにおける歴史ある 1 ジャンルで人気のある 2D シューティングゲーム（以下，STG）におけるステージ生成に関する研究はほとんど行われていない．そのため，STG におけるステージ生成について研究することには価値があると考える．

STG とは，プレイヤが自機を操作して敵とその攻撃や障害物を回避しながら遠距離武器により敵の撃破を行うゲームのことで，代表的なタイトルとして，ゼビウス（ナムコ，1984, 売上約 127 万本），グラディウス（コナミ，1986, 売上約 100

万本）などが挙げられる [6, 7, 8]．STG において，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している．また，従来の 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ることができる”ように，人手によりデザインされたものが固定数だけ用意されていることが多い．このようなステージは，ゲームデザイナーの趣向や創意工夫が凝らされた職人芸の賜物でありプレイしていて楽しいものとなっている．しかし一方で，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層も存在すると考えられる．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と考えられる．

そこで本研究では,「人間プレイヤにとってのステージの難易度や楽しさはどう推定できるのか」及び「そのためのテスト AI プレイヤはどのような特徴を備えていなければならないか」を解明することを目的とする．その上で，人間プレイヤにとっ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生成システムを構築することを目標とする．PCG の手法は様々にあるが，生成したステージの難易度判定にはテスト AI プレイヤを用いた生成検査法が有効であると考え，これを採用する．テスト AI プレイヤに生成ステージをプレイさせ，その結果をステージが良い特徴を有しているかを判断するヒューリスティックな評価関数に入力し，その出力値を用いて遺伝的アルゴリズム [9] により最適化を行うことで目標とするステージの生成を試みる．

このような生成検査型のステージ生成においては，生成ステージをプレイするテスト AI プレイヤが人間離れした挙動を行うようだと，その結果により最適化されたステージが人間プレイヤにとって難し過ぎるものとなってしまうことが考えられる．そのような事態を防ぐため，テスト AI プレイヤには“人間らしさ”が備わっていることが望ましいと考える．テスト AI プレイヤには人間らしさに関する工夫を盛り込む．また，生成したステージの評価のため，テスト AI プレイヤによるプレイを被験者に見てもらう被験者実験を行う．その結果を単純にランダム生成したステージの結果と比較することで，本手法の価値の有無を確認する．

本章に続き，第 2 章では，ステージ生成や人間らしい AI プレイヤなどに関する関連研究を紹介する．第 3 章では，STG における望ましいステージの特徴を考察した上で，そのような特徴を持ったステージを生成するための提案手法について述べる．第 4 章では，本研究のために作成したプラットフォームについて述べる．第 5 章では，ステージ表現型や遺伝的アルゴリズム，AI プレイヤ評価関数，などのアプローチに関する設計や実装について述べる．第 6 章では，実施したステージ生成実験及び，生成ステージ評価のための被験者実験について述べる． 最後に，第 7 章で本研究を総括し，今後の展望や課題を述べる．

第2章　関連研究

本研究は，2D シューティングゲーム (STG) における面白いステージの自動生成を行うことを目標としている．そのために，ステージを生成し，テスト用のコンピュータゲームプレイヤ（AI プレイヤ）が評価し，評価値が最大となるように遺伝的アルゴリズムによってステージパラメータを最適化するというアプローチを取る．本章では，このような研究の枠組みおよび構成要素について，関連研究を簡単に紹介する．

2.1　Procedural Content Generation

第 1 章で述べたように，Procedural Content Generation（PCG）は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”を自動生成する技術のことである．コンテンツの指す範囲は広くゲーム以外にも用いられるが，特にゲームにおいて PCG は急激に発展し，また利用されるようになっている [4, 10, 11]．一般的なビデオゲームに限定してもまだコンテンツの指す範囲は広く，映像や音声，文章，ストーリーなどから，ステージやマップ，パズル，敵や味方キャラクタのパラメータなど，様々である．

対象だけでなく，PCG の手法そのものもまた様々である．頻繁に使われるものとしては以下のようなものがある．

- 生成検査法：
  何らかのアルゴリズムによりコンテンツを生成し，それに対して何らかの方法で評価を行い，良いものを選択する，ないし，悪いものを排除する．
- 最適化：
  生成検査法を進め，評価が高くなるようにコンテンツを進化させる．
- 機械学習：
  既存のコンテンツを訓練データとして，それに似た新しいコンテンツを生成する．特に最近は，この目的に適した，GAN(Generative Adversarial Networks) が使われることも多い [12].

これらにはそれぞれ一長一短がある．単純な生成検査法では本当に良い解を見つけることが困難であったり，かといって最適化を遺伝的アルゴリズムで行えば大きな時間がかかることも多い．また，コンテンツの評価も自動で行わないといけないため，それが難しい場合があることも課題である．機械学習はオンラインでは高速にコンテンツを生成できるが，そのためには訓練データが大量に必要になることが多いという点がボトルネックとなっている．生成されるコンテンツが，訓練データに依存してしまうことも懸念点の一つである．

2.2 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 [12] のほか，Markov chain を用いたもの [13]，LSTM-RNN (Long short-term memory recurrent neural network) を用いたもの [14]，遺伝的アルゴリズムを用いたもの [15] など様々なものが提案されている．

マリオなどの横スクロールアクションゲーム以外にも，ステージ生成の試みは多い．例えば，レースゲームのコース（トラック）を遺伝的アルゴリズムを用いて行う試み [16, 17] がある．ここでは，人間プレイヤをモデル化した AI によるテストプレイヤを実装し，このテストプレイヤの評価値が最大となるように，コースのパラメータが最適化されている．この研究は，「遺伝的アルゴリズムを用いている」「生成対象がステージである」「人間に近いテストプレイヤを用いている」という点で，本研究と強く関連している．

ロールプレイングゲーム (RPG) の，ダンジョンの構成が対象となることもある．RPG のダンジョンはゲームデザイナが工夫して用意することも多いが，自動生成することができればユーザに新しい楽しみを与えることができる．一方で，ダンジョン前半の行動が後半の状況に影響を与えることも多いことから，そのバランス調整はレースゲームやマリオなどと比べても難しいものである．Nam らは，ここに強化学習のアイデアを取り入れた．これまでのダンジョンを状態，次の 1 マス分のダンジョンのパラメータを行動，そのダンジョンのテストプレイヤによる評価値を報酬としてダンジョン生成そのものをマルコフ決定過程における強化学習にモデル化したものである [18]．

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 [19]．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 プレイヤが必要であると述べている．

2.3 人間らしいAIプレイヤ

少し前までは，AIプレイヤの研究と言えば，強いものを作ることが目的となることが多かった．しかし，AlphaGo[2] やDQN[20] の登場以降は，強さはもう十分で，人間を楽しませる研究 [21] や，それに関連して人間らしいAIプレイヤを作る研究 [22] が注目を集めるようになっている．

人間らしいAIプレイヤの意義や必要性は広範にわたるが，ここではSTGやマリオなどのリアルタイムゲームにおけるステージ生成に限って考えてみる．もし，生成検査法の際のテストAIプレイヤが，“1ドット単位で敵の弾を避け，敵の出現を1フレーム後に発見する”ようなものだったとしたらどうだろう．そのような人間離れした能力がないとクリアできないようなステージを，「これはクリアできる」と判定して，人間プレイヤに提供してしまうかもしれない．そこで，Togeliusらはこれを防ぐために，人間プレイヤをモデル化してテストAIプレイヤとして用いている [16, 17].

人間らしいリアルタイムゲームのAIプレイヤの作成にもまた，さまざまなアプローチがある．Silaらは感情を持ったように見えるマリオAIをA*アルゴリズムの変形により作成している [23] が，テストAIプレイヤとして用いる際にはこのような精神面よりは，物理面に着目すべきかもしれない．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 [22]．STGについては，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作成する平井らの試みがある [24]．また，佐藤らによる，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プレイヤを作成する試みもある [25]．本論文のテストAIプレイヤにはこのアイディアを盛り込んでいる．

第3章　アプローチ

本章では，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関しての考察を述べる．次に，ステージの生成手法および評価手法に関しての検討を述べる．それらを踏まえた上で，望ましい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

一般的な STG において，ステージは「自動的な画面のスクロールや時間の経過に合わせて敵が出現し，敵が移動や攻撃を行うことで，プレイヤの妨害を行うもの」となっている．このことから，ステージを構成する共通要素は「敵の（出現・移動・攻撃の制御に関する）パターン」であり，このパターンの集合が STG のステージの難しさや楽しさを司っていると考える．

望ましいステージ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，難し過ぎるステージ
- どうプレイしてもクリアできてしまう，易し過ぎるステージ
- 終始動く必要があり，繁忙な，緊迫状態が連続したステージ
- 終始動く必要がなく，退屈な，弛緩状態が連続したステージ
- 繰り返しが多く，ワンパターンで多様性に乏しいステージ
- 繰り返しが少なく，まるで規則性のない雑然としたステージ

これらの望ましくないステージから特徴を抽出および分類すると，以下の 3 つにまとめることができる．

- 難易度：高過ぎる・低過ぎる
- 緊張感：あり過ぎる・なさ過ぎる
- 多様性：あり過ぎる・なさ過ぎる

よって，これらを極端にならないように整えた以下の特徴が，望ましいステージの特徴と言える．

- 難易度：高過ぎず・低過ぎず，ほど良い
- 緊張感：あり過ぎず・なさ過ぎず，ほど良い
- 多様性：あり過ぎず・なさ過ぎず，ほど良い

しかし，これら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．例えば，緊張感について言えば，ステージの開始から終了に至るまでの数分間において常に同じレベルの緊張度（例えば敵や弾の数）が継続することが望ましいわけではない．多くのSTGの人手で作られたステージがそうであるように，まずは様子見程度の敵集団，続いてやや本格的な敵集団，それを凌いだ後に一度緩い区間があってボス集団，などとメリハリがあることが望ましい場合が多いと考える．このようなことを考慮した上で，特徴をミクロとマクロに細分化しまとめると以下のようになる．

- 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　ステージ生成手法および評価手法についての検討

コンテンツ生成の手法は様々あり，2.1 節では代表的な手法として，生成検査法と最適化，機械学習の 3 手法を紹介した．本節では，それらの手法に関してもう少し掘り下げて行くことで，本研究で採用する生成手法についての検討を行う．

機械学習を用いたコンテンツ生成手法は，ここ数年の深層学習アルゴリズムの急激な発展に伴い活発に研究がなされている．このタイプの手法の利点は，学習データとして既存のコンテンツを用意できさえすれば，学習データに近しい多様なコンテンツを大量に生成可能となる点である．一方で，大きな問題点が 2 つある．

まず 1 つは，学習用データが大量に必要となることである．例えば，Generative Adversarial Networks[26] を用いて 3D シューティングゲーム（first-person shooting）におけるステージを自動生成する研究 [27] では，満足いくものを生成するために 1000 個もの訓練データによって学習を行っている．このような分量の訓練データは，既存のゲームであっても集めるのは容易ではなく，新規ゲームにおいては尚のこと困難となる．既存の同ジャンルの別のゲームから集めて流用する手も考えられるが，その場合には，訓練データないし生成コンテンツを対象とするゲームに合わせて加工する手間が生じてしまう．その際には，どのように加工するかについても検討する必要がある．2 つ目の問題点は，生成できるコンテンツの性質が用意したデータの分布に依存してしまうため，性質の制御が困難となることである．例えば，用意したステージデータに難易度の偏りがあるようだと，生成されるステージにもその傾向が反映されるため，偏った難易度のステージばかりが生成されてしまう．

機械学習による生成法に対して最適化による生成法は，データ構造を工夫する必要はあれど生成のためにデータを用意する必要はないため，データを集めるのが困難なゲームにおいて有効である．また，一試行内での最終的な生成物に性質の偏りはあれど，評価手法を変えて複数回最適化を行うことで多様な性質のコンテンツを得ることが期待できるため，難易度などの性質を制御したい場合には有効である．

最適化は，生成検査法 [28] と呼ばれる，「何かしらの手段でコンテンツを生成し，生成したコンテンツの良さを評価し，良い評価のコンテンツを選択する」手法を発展させたものである．そのため，生成したコンテンツの良さを評価する手段を用意する必要がある．評価手法は主に次の三種類がある．

1. 直接評価

   概要：

   コンテンツの生データを入力とするヒューリスティックな評価関数により，その良さを評価する．

   利点：

   仕組みは最も単純で，評価コストが少ない．

欠点：
コンテンツの生データに依存したヒューリスティックな知見が必要となる．また，生データからは直接読み取れない性質の良さを評価できない．

2. インタラクティブ評価

概要：
コンテンツを人間が目視や実プレイなどにより評価する．

利点：
評価者にもよるが，人間に適した評価が可能である．特にその評価者にとって良いものができる．

欠点：
評価コストが膨大となる．

3. シミュレーション評価

概要：
テスト AI プレイヤによるプレイ結果を入力とするヒューリスティックな評価関数により，その良さを評価する．

利点：
評価コストはインタラクティブ評価よりも大幅ダウンを見込める（直接評価よりは大きい）．また，コンテンツの生データからは直接読み取れない性質の良さを評価できる．例えば，ステージにおいては，クリア状況から難易度，プレイヤの行動率から緊張度，などを評価することが可能であると考える．

欠点：
AI プレイヤの出来に依存する（AI プレイヤを作るのが難しい場合は適さない）．また，AI プレイヤを作成するコストも別途必要となる．

2.2 節で述べた長の研究 [19] では，ランダム生成した弾幕と呼ばれるステージの一構成要素（敵の攻撃パターン）の難易度推定に，AI プレイヤによる評価が有効であることが示されている．このことから，敵の出現や移動のパターンも含めたステージそのものの難易度などの推定においても，AI プレイヤによる評価が有効であると考える．

以上を踏まえ，本研究では，最適化による生成とテスト AI プレイヤによる評価を組み合わせた生成検査法を提案する．

3.3　テスト AI プレイヤ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 3.1 のようにする．

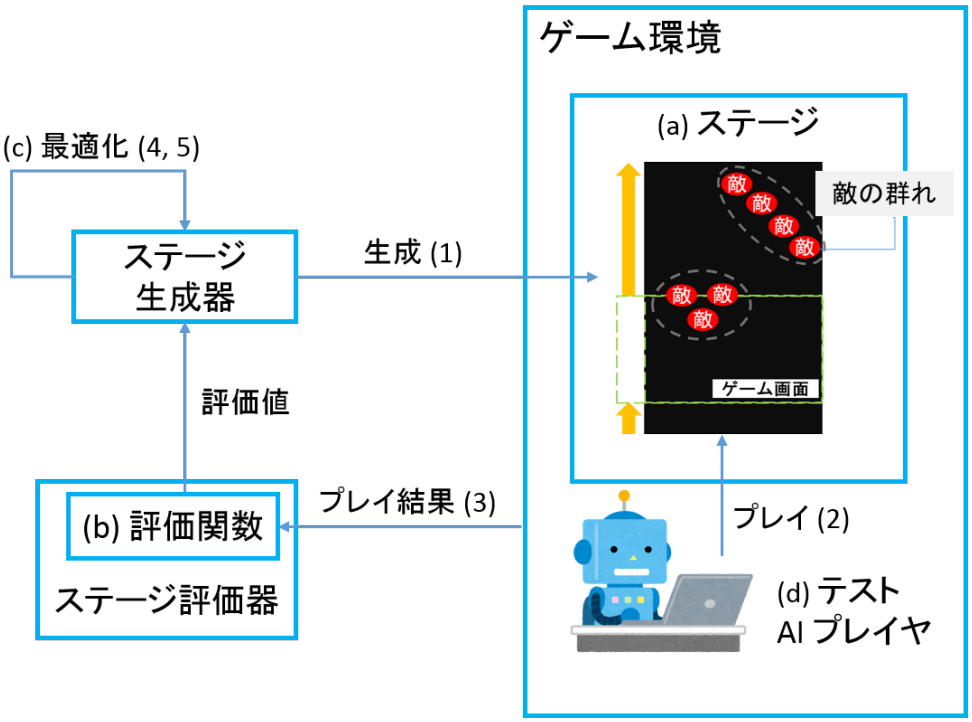


図 3.1: 生成検査システムの全体像

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する
4. 2-3 が一定の回数行われていれば終了する．そうでなければ 5 に進む．
5. 3 から出力される評価値を元に生成器内でステージパラメータを遺伝的アルゴリズムを用いて最適化し，次世代ステージを生成し，step2 に戻る．

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，普通に考えれば，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，プレイ結果の内の難易度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（遺伝の法則，自然選択説）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）[9] を用いる．これは，

- どのように対象データを操作すれば上手く評価値を上げることができるのかについての知識が必要ない，メタヒューリスティクなアルゴリズムである
- 2.2 節で紹介した関係研究において用いられており，その有効性が示されている
- 評価関数が明示的に与えられず，勾配情報が得られない場合でも用いることができる

ためである．

GA は，対象データの一要素を遺伝子とみなし，遺伝子を操作（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，敵や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，人間らしさの工夫を盛り込む必要があると考える．

また，各種詳細・設計・実装に関しては第 5 章にて述べる．

第4章　作成したプラットフォーム

前章で述べたアプローチの実装を行うため研究環境として，シンプルな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．本章では，プラットフォーム作成にあたり意識した点とその各種仕様について述べる．

4.1　意識した点

プラットフォームは研究用であるため，シミュレーションの実行やAIプレイヤ開発の観点から，次のような点を意識し作成を行った．

- ゲーム性・構成要素をシンプルに
  多くの構成要素を盛り込みゲーム性を複雑にする行為は，研究ではなくゲームデザインの範疇であると考える．また，その行為によりAIプレイヤの思考設計コストの増大も懸念される．一方で，あまりに構成要素を削りすぎると，本来のSTGにあるはずの重要な面白さや難しさまで対象から外してしまうことになる．そのため，重要なもののみを選んで入れる必要がある．
  本研究では，一般的なSTGに採用されている範囲攻撃・緊急回避手段などである“ボム”や自機強化オブジェクトである“アイテム”，壁・扉・岩など自機を妨げるオブジェクトである“障害物”，障害物を制御する“ギミック”，強大な敵オブジェクト“ボス”は採用しない．その一方で，STGに関する既存研究[25]では採用されていなかった自分の攻撃については，将来の利用を見据えて可能なように実装することにする．つまり，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，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．
- 人間のプレイや目視を可能に
  本研究では，被験者にステージを実際にプレイしてもらったり，他者のプレイを見てもらう実験を想定している．そのため，ゲームを“リアルタイムで”操作・表示できるような入出力機構を持たせた．
- AIプレイヤによる高速なプレイ実行を可能に
  本研究では，ステージを遺伝的アルゴリズムなどの生成検査法で最適化することを想定している．そのためには，多くのステージを作り，AIプレイヤにプレイさせ，評価する必要がある．もしこの1プレイにリアルタイムの

時間（1 分前後）がかかってしまうとすると，最適化全体では莫大な時間となってしまう．そこで，画面表示等を行わず，高速に状態遷移と評価値出力が行えるようにする機能を持たせた．

- ゲームの再現を可能に
  シミュレーションによるステージの生成後，生成したステージのプレイ結果を目視などで確認できる必要がある．そのため，同じ設定の AI エージェントと同じステージをゲームに与えた際に，同じ結果を返すよう，ゲームループ自体はランダム性を持たない決定的な処理とした．
- ゲーム実行中のパフォーマンスを最適化
  AI プレイヤの思考時間は通常の 1 フレームの時間（ゲーム画面及びゲーム内の状態を更新する間隔），約 16.67 ミリ秒に収まることが望ましい．そのため，ゲーム中に AI プレイヤの思考時間を少しでも長く確保することを試み，コストのかかる動的なメモリ確保はゲーム開始前に済ませてしまうことで，ゲーム実行中のメモリ確保・解放の発生を極力抑え，パフォーマンスの最適化を図った．

各種の詳細な実装は次節にて述べる．

4.2 各種仕様

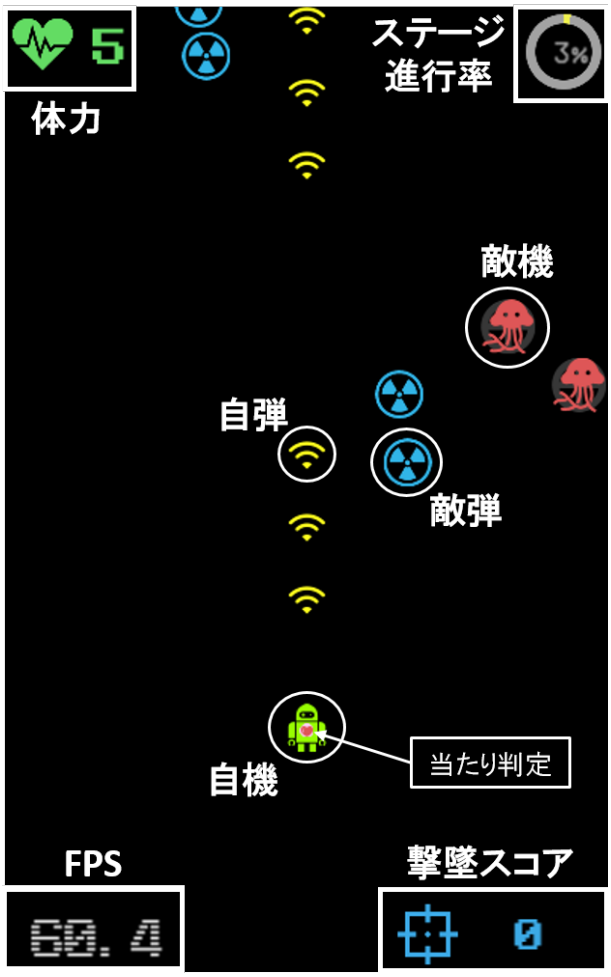


図 4.1: ゲーム画面

概要

図 4.1 のような縦型 STG を作成した．画面の縦横比は 16:10 で，デフォルトのサイズは縦 1280pixel × 横 800pixel に設定されているが，640 × 400 の半減サイズで起動することも可能である．STG は，画面が実際にスクロールしてゲームが進行するスクロールタイプのものと，画面がスクロールせずに時間経過によってオブジェクトを出現させゲームを進行させる疑似スクロールタイプのものとがある．本研究で作成したのは後者の疑似スクロールタイプである．また，1 秒間のフレーム数（ゲーム画面及びゲーム内の状態を更新する回数）は，一般的な設定の 60FPS（Frames Per Second）とした．

オブジェクト

ゲームに登場するオブジェクトはプレイヤの操作する自機，自機から発射さ

れる弾（自弾），プレイヤを妨害してくる敵機，敵機から発射される弾（敵弾）の 4 種類である．

ゲーム終了条件

ゲームの終了条件は以下の 2 点である．

- ゲームオーバー
  ライフ（体力値）が 0 になることによる自機の死亡
- ゲームクリア
  全ての敵が登場済み，かつ画面上に敵機や敵弾が存在しない，かつ自機が生存

プレイヤは自機を操作し，ゲームクリアを目指す．

自機の初期状態とその遷移

自機は画面中央やや下の位置にいる状態でゲームが開始される．ゲーム開始時の自機の初期ライフは 5 で，初期状態は「生存状態」である．自機中央の当たり判定部に敵機や敵弾の当たり判定が接触する（被弾する）と，ライフが 1 減少する．同時に被弾した場合でも減少する値は 1 である．この時の残りライフの値によって，次のように状態が遷移する．

- ライフが 0 :「生存状態」から「死亡状態」に遷移（ゲームオーバー）
- ライフが 1 以上 :「生存状態」から「無敵状態」に遷移

「無敵状態」では，当たり判定が無効となり被弾しなくなるが，2 秒後にはまた「生存状態」に遷移する．

自機の可能な行動

自機の取りうる可能な行動は，その場での待機（無移動）と上下左右斜め移動（8 方向への移動）に，自弾の発射（射撃）の有無を組み合わせた，18 種類である．プレイヤはこれらの行動を示す行動値をゲームに入力し，自機を操作する．画面上全ての範囲に移動することができるが，画面外に出ることはできない．

射撃行動を行うと，数フレーム間隔で 3 発の自弾が自動で y 軸上方向に発射される．この自弾の当たり判定が，敵機の当たり判定に接触すると，敵機のライフが 1 減少する．ライフが 0 になった敵機は消滅する．

敵機・敵弾

敵機と敵弾は，一時停止時を除き，等速直線移動を行う．そのため，その進路の予測は一時停止時の前後以外では容易である．

また，敵機は生成時に以下のようなパラメータにより初期化される．

- 出現フレーム数
- 出現座標
- 移動速度
- 敵の種族：3 種で，見た目やサイズ，体力などがそれぞれ異なる．
- 移動目標座標
  この座標と出現座標との差分ベクトルを正規化したものが，初動の進行方向ベクトルとなる．
- 移動の種類
  図 4.2 の 6 種で，出現座標と移動目標座標（とその y 軸対称座標），それらから計算される各進行方向ベクトルにより制御される．
- 射撃の種類：6 種（無射撃，1 方向，2 方向，3 方向，2 弾 1 方向，十字）
- 照準の種類
  6 種（敵の進行方向に撃つ，下向きに撃つ，全方位に回転撃ちをする，毎回自機位置を狙って撃つ，一度自機位置を狙ってその位置にずっと撃つ，自機の未来位置を予測してそこに撃つ）
- 連射シーケンスの回数
- 連射シーケンスの間隔フレーム数
- 連射シーケンス中の射撃回数
- 連射シーケンス中の射撃間隔フレーム数
- 弾の種類：3 種で，見た目とサイズがそれぞれ異なる．
- 弾の速度

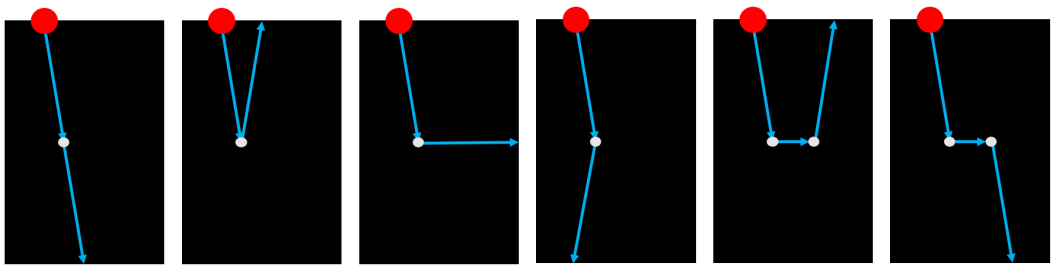


図 4.2: 敵機移動法：6 種

ステージ

本ゲームには，アイテムや障害物，ギミックなどは登場しない．すなわち，前項の敵の配置や挙動こそがステージの本体となる．ステージをどのようなデータ構造として表現するかは，最適化手段と関連して本研究にとって重要な部分である．その詳細は 5.1 節で述べる．

プレイログ

ゲームプレイ後に出力されるプレイログデータには，以下のような情報が記録されている．

- ステージ情報
- ゲーム結果情報
  クリアしたかどうか，残りライフ数，生存時間，獲得スコアなど
- プレイ計測情報
  自機行動履歴，敵や弾に近づいてしまった回数，各フレームにおける画面上の敵数や弾数の履歴など

5.4 節で述べるステージ評価関数にはこのプレイログが入力される．

第5章　アプローチの設計・実装

本章では，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 節と 3.3 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設計や実装などの詳細を述べる．具体的には，ステージデータをどのような構造・表現にしたかや，どの種類の遺伝的アルゴリズムを選択したか，AI プレイヤの行動選択アルゴリズムにはどのような工夫を盛り込んだか，評価関数はどのような設計にしたかについて順に述べる．

5.1　ステージデータ構造・表現

4.2 節でも述べたように，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である．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）[9] により最適化を行う対象である．そのため，ステージは遺伝子の集合として扱えるデータ構造である必要がある．また，生成したステージは確認や再評価のためにファイル入出力が可能であるべきであり，出力されたステージファイルは人間が見た際に理解しやすい構造であることが望ましい．

以上を踏まえ，本研究では，ステージのデータ構造を「敵の配置や挙動に関する複数のパラメータ」の配列とした．この「敵に関する複数のパラメータ」は“パターン”と呼称することとし，GA で扱う際には，パターンが 1 つの遺伝子として扱われる．また，この構造は二次元配列として表現が可能なため，csv ファイル形式による入出力も行えるようにした．パターンの表現については次項から順に述べる．

5.1.1　単純型

まず初めに単純なものとして，以下のような「1 体の敵のみの配置や挙動に関する複数のパラメータ」であるパターンを遺伝子とするステージを考えた．

- 出現フレーム数：[0, max]
- 出現座標 X：[0, 800]

- 敵の種族番号：[0, 2]
- 移動速度比率：[0.40, 0.70]
- 移動目標座標 X：[0, 800]
- 移動種類番号：[0, 5]
- 射撃種類番号：[0, 5]
- 照準種類番号：[0, 7]
- 連射シーケンスの回数：[1, 3]
- 連射シーケンス中の射撃回数：[1, 10]
- 連射シーケンス中の射撃間隔フレーム数：[10, 30]
- 弾の種類番号：[0, 2]
- 弾の速度比率：[1.20, 1.40]

これらのパラメータを“単純型パターン”とし，単純型パターンの集合からなるステージを“単純型ステージ”と定義することとした．

単純型パターンの 1 つの配列は，ゲーム開始時に 4.2 節で述べた 1 つの敵のパラメータに変換される．殆どはそのまま用いることになるが，一部計算を伴うものもある．

- 移動速度や弾の速度は比率をパラメータとして持つため，実際の値の計算を行う．
  - 移動速度 = 自機速度×移動速度比率
  - 弾の速度 = 移動速度×弾の速度比率
- 出現座標は，X 座標のみをパラメータとして持ち，Y 座標は 1280 に固定した．
- 移動目標座標も，X 座標のみをパラメータとして持ち，Y 座標は 640 に固定した．
- 連射間隔は 40 に固定し，パラメータ化しなかった．

この単純型パターン・ステージに，次項のような工夫を加えた．

5.1.2 群れ型

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）を制御するために，前項で述べた単純型ステージに，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．導入にあたり，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 種類の群れをパターンとして表現するために，以下のような「群れの種類や配置に関する複数のパラメータ」を単純型パターンに加え，これを“群れ型パターン”と定義した．合わせて，群れ型パターンからなるステージを“群れ型ステージ”と定義した．

- 群れモード：[0, 1]
  0 なら編隊型，1 なら連鎖型となる．
- 敵ビット表現：[000001, 111111]
  この値によって，群れを構成する敵の数とその敵の配置が決定される．その処理は，群れモードの値によって異なる．
  群れには，基準時間や基準座標，基準移動目標位置が，出現フレーム数や出現座標，移動目標座標によりそれぞれ設定される．群れを構成する各敵のパラメータは，出現フレーム数や出現座標，移動目標座標を除き共通のパラメータが使用される．

  編隊型

  各ビット位置の番号を 543210 とした時，各ビットの配置は以下のように変換する（3’, 4’, 5’ には 3, 4, 5 と同じ値を入れる）．

  5’ 2 5
  4’ 1 4
  3’ 0 3

  この時，ビットが立っている位置に敵がいるものとする．そのため，敵の数は，0 から 2 までのビット数に，3 から 5 までのビット数の 2 倍を加えたものとなり，その値域は [1,9] である．例えば，敵ビット表現が 010101 であれば，その際の敵の配置は以下のようになり，敵の数は 4 となる．

  _ e _
  e _ e
  _ e _

  ビット位置が 0 の位置を基準座標とし，群れの出現座標を設定する．各敵の座標間隔は，敵の種族ごとに異なる．各敵の座標は，基準座標と移動目標座標のなす角度により回転したものが設定される．その他の出現フレーム数などは，共通のものを設定する．これにより，同じパターンの敵が編隊を組んだような出現の仕方をするよう制御する．

連鎖型

立っているビット数が敵の数となる．そのため，敵の数の値域は [1,6] となる．また，各敵のビット位置は考慮しない．
群れ全体の基準時間は出現フレーム数，基準座標は出現座標（と移動目標座標）に設定する．群れの各敵を生成する際には，先頭の敵には基準時間・座標をそのまま，2 体目以降の敵には，基準時間・座標に対して一定間隔でずれ時間やずれ座標を順に加算したものを設定する．それ以外のパラメータは共通のものを設定し，それぞれ生成する．それにより，同じパターンの敵が連鎖して出現するよう制御される．ずれ時間や座標は，以下の出現間隔ずれフレーム数や出現間隔ずれ方向ビット表現により計算される．

- 敵数：[1, 9]（敵ビット表現から算出）

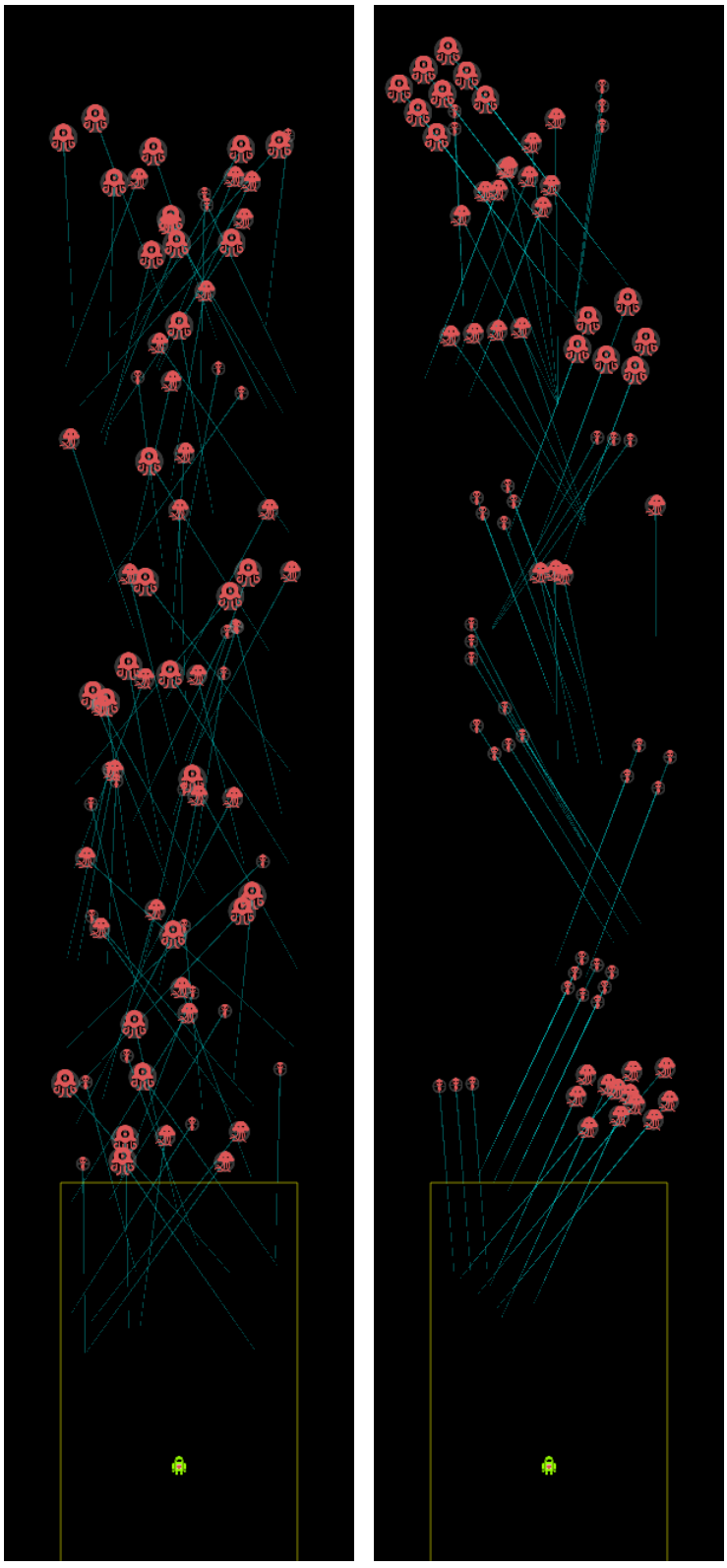
- 出現間隔ずれフレーム数：[0, 12]
  連鎖型においてのみ使用する．先頭の次の敵から順に，この間隔ずつ出現フレーム数にずれを生じさせる．
- 出現間隔ずれ方向ビット表現：[01, 11]
  連鎖型においてのみ使用する．先頭の次の敵から順に，01 なら横方向，10 なら上方向，11 なら斜め方向に，出現座標や移動目標座標にずれを生じさせる．ずれの値は敵の種族ごとに異なる．
- 単純型パターンのパラメータ
  出現フレーム数や出現座標 X，敵の種族番号など 13 種．

群れ型パターンは，ゲーム開始前に，内包する敵数と等しい数の 4.2 節で述べた敵の初期化用パラメータに変換される．基本的には単純型と同様の変換が行われるが，編隊型の群れにおいては出現座標と移動目標座標に，連鎖型の群れにおいては，出現フレーム数と出現座標や移動目標座標に，上記のパラメータから計算されるずれ値がそれぞれ加算されることで変換が行われる．

単純型と群れ型ステージの比較のために，同じ敵数（80 体）を指定してランダムに生成したものから，各敵の出現時系列に合うよう画面上部から上方向へと敵を配置した図を，図 5.1 に示す．

赤いオブジェクトが敵で，青い線が各敵の初動の進行ルートを表す．このように，単純型と違い，群れ型ステージにはミクロな統一感が生じることを確認した．

次節で述べる遺伝的アルゴリズムでは，本項の群れ型ステージを遺伝子配列として扱い，その最適化を行う．



(a) 単純型　　(b) 群れ型

図 5.1: ランダム生成ステージ

5.2 遺伝的アルゴリズム

5.2.1 一般論

遺伝的アルゴリズム（Genetic Algorithm, GA）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って，ベンチマーク問題から実応用までさまざまな領域で優れた性能を発揮している．最適化アルゴリズムには，大まかにいって「必ず最適解を見つけられるが，時間がかかるもの」と「最適解を保証しないが，高速で満足な解を得ることを目的としたもの」がある．分枝限定法は前者の代表，やきなまし法（Simulated Annealing）や GA は後者の代表である．さらに，勾配法などと違って評価関数の明示的な定義やその微分可能性が不要であることも，応用領域で頻繁に用いられる理由の一つである．

遺伝的アルゴリズムの構成要素は，大きく分けて以下のものである．

- 個体：一匹の生物をイメージしている，最適化したい解のパラメータ x．だいたいの場合，配列（ベクトル）で表す．
- 適応度：その生物が環境にどの程度適応しているか，つまり優れているかを表す．最適化したい解の評価値 $f(x)$ またはそれを変換したものである．
- 交叉オペレータ：2 つの個体から，新しい 1 つの個体を生み出す，確率的操作．$(x_1, x_2) \mapsto x_c$.
- 突然変異オペレータ：1 つの個体から，新しい 1 つの似た個体を複製する，確率的操作．$x \mapsto x'$.
- 世代交代モデル：個体の集合（群）が世代を重ねて進化する手続き．交叉や突然変異で新しい個体が作られ，適応度によって何らかの基準で選択・淘汰が行われる．

それぞれ，どのように定めるかは GA の実施者によって決められ，またそれによって満足できる解が得られるかどうかも変わってくる．

適応度が適切に定められなければ，GA そのものは「良い適応度の解を見つけました」と終了しても，それが現実世界で好ましいものであるとは限らなくなる．一方，交叉オペレータや突然変異オペレータが不適切だと，少し進化した解からより進化した解が得られる確率が減ってしまい，進化が停滞する．そこで，これらのオペレータでは，元の解の良いところをあまり崩さないようにする，という配慮が必要である．

5.2.2　本研究での遺伝的アルゴリズム

本研究では，3.3 節及び 5.1 節で述べたように，ステージパラメータを“個体”，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．

ステージパラメータ（個体）の内容については 5.1.1 節および 5.1.2 節に記述した通りであり，単純型であれば 1 つの敵が 13 次元ベクトルで表せる．群れ型であれば，それに 5 次元が加わって 18 次元ベクトルである．一つのステージは多数の群れから構成されるため，これを最大 m 個とするならば，$18m$ 次元ベクトルが一つの個体ということになる．

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に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」を崩さないというのが我々の工夫である．もちろん，例えば「群れの中で，進行方向（出現座標と移動目標座標）を少しだけ変える」ような突然変異も有効かもしれないが，そのような比較までは行っていない．

世代交代モデルとしては，MGG（minmarl generation gap）[29]+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, ..., 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, ..., 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 節でテストプレイヤの実装，5.4 節で評価関数の実装を述べたあと，6 章でこの GA の結果を示す．

5.3 テスト用 AI プレイヤ

本研究の主たる部分は遺伝的アルゴリズムによるステージの最適化であり，それには目的関数となるステージの評価値を定める必要があり，そのためにはステージを自動プレイする AI プレイヤが必要になる．3 章で述べたように，その AI プレイヤがある程度「人間らしく」プレイしてくれないと，ステージの難しさなどを適切に評価できない．AI プレイヤを人間らしく振舞わせる研究は奥が深くさまざまな方法がありえるが，本研究では 2.3 節で紹介した佐藤らのアイデア [25]，すなわち「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」を再現するための方法を援用することにする．

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 5.3.2 節で述べるとして，まずは基本的な構造について説明する．AI プレイヤが利用することのできる入力情報は，通常の人間プレイヤと同じもの，たとえば以下のものである．

- 自機の座標，状態（無敵かどうか），残りライフ
- 画面内にある敵のパラメータのうち，推測可能なもの（出現位置，種類，速度など）
- 画面内にある敵の弾の情報
- ゲーム開始からの経過時間

このうちいくつかは実際には利用されない．重要なこととして，ゲーム本体と AI プレイヤは同じコンピュータ・同じプログラムで動いているとはいえ,「人間プレイヤには分からないはずのデータ」例えば未出現の敵のパラメータや，出現している敵の今後のルート（図 4.2 参照）などは，AI プレイヤには利用させない．

これらの情報を用いて，AI プレイヤは，各フレームごとに，つまり比較的短時間で自分の行動を決めることになる．本研究ではまず自機は弾を撃たない設定にしたため，行動は上下左右斜め＋待機の 9 通りあることになる．

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」つまり 9^n 通りについて，将来を予測し，最も望ましい将来に導くような最初の 1 フレーム分の行動を選ぶ，というものである．2 人ゲームの基本アルゴリズムが minmax 法であるのに比べ，これは自分の行動だけを決めればよいのでいわば max 法である．このようなアルゴリズムでは，自分の行動を決めたとしてその場合の未来が予測できることが前提となる．STG の場合，敵が予想外の動きをしたり，突然敵が出現したり，突然敵が弾を発射するなどの理由によって予測が正確には行えない．しかし多くの人間プレイヤは，そのようなことがあるかもしれないと思って危険そうな場所からは離れるなどの工夫によって，予測の不正確性に対応している．

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予測の不正確性に対応して安全でもあり，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のであれば，将来の安全のために，画面中央に向かう行動を選択する．
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自機より遠い場合は，敵が弾を射出する可能性があるので，その場にとどまる行動を選択する．
5. それ以外の場合，すなわち敵や弾が自機の近くにある場合は，将来予測をして，回避のための行動を取る．具体的には，9^n 通りの行動の組み合わせからなる回避ルートを計算し，どの最初の行動が最も安全な状態に導くのかを調べてそれを選択する．

ルートの安全さについては，最も単純には,「その最初の行動を選んだ場合に，n フレーム先まで生き残れるルートが何通りあるか」を用いることとした．実際には $n = 3$ としたが，この場合，9 通りの最初の行動それぞれについて 81 通りのルートがあることになる．例えば，行動 A だと 81 通りすべてで生存が可能だが，行動 B だと 20 通りのルートでは被弾するようなケースならば，行動 A のほうがより安全な手として選ばれる．

5.3.2　人間らしい行動のための工夫

5.3.1 節で提案したアルゴリズムを用いた AI プレイヤは，かなり短い未来を正確に予測して次の行動を選ぶものになる．実際に挙動を見てみると，敵の弾をかなり危険が迫ってからぎりぎりで回避したり，弾が多ければ 1 フレーム単位で細かく操作してそれを避けるような挙動が見られた．ただ，佐藤らも指摘しているように，これは人間プレイヤの挙動やプレイスタイルとはかなり異なる [25]．人間プレイヤは，自機を正確にコントロールしたり敵の弾との距離を正確に見切ったりすることが簡単にはできない一方で，将来ありうる危険を予測して安全そうな方面への退避したり，敵や弾との距離を余裕を持って回避したりといった行動は頻繁に見られる．また，2.2 節で紹介した長の研究 [19] においても，AI プレイヤをコンテンツの難易度推定に用いる際には，AI プレイヤには大局的な判断を行わせる必要があることが述べられている．

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 AI プレイヤに持たせることにした．

【オプション 1：ざっとした先読みやざっとした行動を行う】
人間はよほどの上級者を除いては，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．このようにすることで細かな制御ができなくなるので，プレイヤにとっては大きな制約となる．これに加えて，未来予測も 10 フレーム×深さ 2 まで行うこととした．もしナイーブに 20 フレーム先まで探索しようとすれば 9^{20} のノードが必要になり現実的ではないが，10 フレーム先を 9 ノード，20 フレーム先を 81 ノード探索するのであれば探索コストは大きくない．ただし，5 フレーム先や 13 フレーム先も同じ行動を取り続けたものとして，敵との当たり判定だけは行っている．こうすることにより，「正確ではないが，より長期的で大局的な」人間らしい制御を行うことを狙った．
【オプション 2：敵・敵の弾・画面の端からの距離を考慮に入れる】
5.3.1 で説明した短期的な生き残りを重視する方法 (a) では，見えている敵や弾との距離を取ろうとして自機が画面の端まで追い詰められ，そこで回避をし続ける挙動がしばしば見られる．これは人間の STG 初心者にもよくあることではあるが，少し上達していくと「回避行動を行いにくい画面端はむしろ危険なため，可能であれば回避スペースを取りやすい中央付近にいるほうが望ましい」ことを学んでいく．また，方法 (a) では探索により最も安全な状態が導き出した結果，敵の進行方向などを考慮して敵や弾からあまり離れない行動を選択することがある．通常，人間プレイヤは敵や弾といった危険なものからは距離を取るように回避をすることが多いため，このような行動は不自然に見える．そこで，20 フレーム先までを探索した際にただ「生き残ったかどうか」を調べるのではなくて，「敵・敵の弾・画

面端からの距離の最小値」を計算し，それが最も大きくなるようなルートを選ぶこととした．

【オプション 3：安全めの回避】
STG やマリオなどのアクションゲームでは，自機・敵・弾などに「この物体はここからここまで存在する」ことを定義する，当たり判定と呼ばれる領域がある．通常は長方形や円形で近似され，長方形や円形同士が触れると，その 2 つの物体は接触したと判定され，敵と自機ならば，自機のライフが減ることになる．AI プレイヤは，1 ドット単位で自機や敵を認識できるので，1 ドット単位での回避も可能である．そのため，弾と弾の間の僅かな隙間を縫うようにして回避するなどの行動がしばしば見られる．しかし人間プレイヤはよほどの上級者でない限りこれは不可能であるため，ある程度のマージンをとって回避を行う．これは現実社会における車の運転などでも見られる自然な行動である．そこで，3 つめのオプションとして,「自分の当たり判定が 2 倍になったものとして探索を行う」というものを導入した．

挙動が自然かどうかを 5 段階評価してもらう簡単な被験者実験の結果からは，特にオプション 1 が極めて有効に働くことが分かった．オプション 1 を用いない場合の平均点が 1.29 だったのに対し，用いた場合は 4.21 であった．オプション 2 については統計的に有意な結果は得られず，オプション 3 についてはむしろ否定的な結果が得られてしまったが，実験を行った順序の関係で，6 章での実験は全てのオプションを用いたものになっている．

なお，オプション 3 での挙動が不自然に見えた理由としては，明らかに避けなくても良いような状態でも避けたことが挙げられる．これは当たり判定が倍というのが極端な設定だったことが原因かもしれない．

5.4 評価関数

GA でステージを最適化する際には，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) と (f) については，評価関数ではなく，5.1.2 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．図 5.1(a) が謂わば多様すぎるステージであるのに対し，図 5.1(b) は部分的に見ると群れには統一感があり，群れの集合であるステージにはほど良い多様性がある．もちろん，たまたま多様性のなさすぎるステージが生成されてしまうこともあるだろうが，そこは今回は考慮しないことにした．

(a)〜(d) については，各ステージ x ごとに，AI プレイヤによるテストプレイを行い，評価関数値 $f(x)$ を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．

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

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．

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機数であり，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：HiyariCount と同様に，当り判定の 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，(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,「画面上最大何体の敵が登場したか」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，休憩をおいて，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．

それぞれの範囲と，重みについては次の表 5.1 にまとめた．これらの値は筆者が試行錯誤によって適当に定めたものである．区間最大敵数と弾数は，初めは静かで簡単な状態から始まり，緩急が波のような道中を経て，最後に激しく難しい，所謂クライマックスな状態で終了するように意図してのことである．弾数の区間が敵数のものと異なる粗い分割を行っているのは，弾は敵が射出するものであるため，敵自体の出現数を細かく制御しさえすれば弾は細かく制御する必要はないのではないかと考えたためである．また，区間などを定めるにあたり，生成するステージの時間長（最後に登場する敵の出現時間）は 60 秒（3600 フレーム）とすることとした．

表 5.1: 各統計量の理想範囲と重み

統計量	理想範囲 最小値	最大値	重み
ライフ数	2	3	500
ヒヤリ数	10	30	10
ニアミス数	0	5	10
無行動率	0.4	0.6	50000
最大連続無行動数	60	150	5
区間最大敵数			
0000-0059[f]	0	0	5
0060-0599[f]	6	10	5
0600-0779[f]	0	2	5
0780-1379[f]	9	15	5
1380-1559[f]	0	2	5
1560-2159[f]	8	12	5
2160-2339[f]	0	2	5
2340-2939[f]	12	18	5
2940-3119[f]	3	7	5
3120-3419[f]	10	14	5
3420-4799[f]	16	24	5
区間最大弾数			
0000-0119[f]	00	00	3
0120-0779[f]	05	15	3
0780-1559[f]	15	25	3
1560-2339[f]	10	20	3
2340-3119[f]	20	30	3
3120-3419[f]	15	25	3
3420-4799[f]	30	50	3

第6章　ステージ生成実験・評価

本章ではまず，3章及び5章で述べた提案手法により実際にステージ生成実験を行ったため，その目的や設定，結果，考察について述べる．提案手法は，具体的には，ステージデータの1要素を敵の群れとし，それを人間らしい工夫を盛り込んだテストAIプレイヤとそのプレイ結果を入力とする評価関数により評価し，その評価値を用いて遺伝的アルゴリズムにより最適化を行う手法である．その後，生成したステージの評価のために行った簡単な被験者実験について述べる．

6.1　目的・概要

本章で述べる実験の目的は，3.1節で考察した以下の3種2視点の望ましいステージの特徴を持ったステージを生成することである．そのため，5.2節で述べた遺伝的アルゴリズムを用いて最適化することにより，その生成を試みた．

- 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

遺伝的アルゴリズムにおいて個体として扱うこととなるステージは，その最小要素を5.1節で提案した敵の群れとすることとした．これにより，多様性の制御を試みた．

遺伝的アルゴリズムでは，解の良さを評価する必要がある[9]．そのため，5.3節で提案した以下の3つの人間らしさに関する工夫を盛り込んだテストAIプレイヤによるプレイ結果を，5.4節で提案した評価関数に入力し，その評価値を計算することとした．

- 連続10フレーム先読み・同一行動

- 敵や弾，画面端との最小距離が最大となるような行動を探索
- 探索時は当たり判定サイズを 2 倍にしてシミュレーション

評価関数では，5.4 節で提案したようにテスト AI プレイヤによるプレイ結果から以下の統計量を計算し，それを用いてステージの評価値を算出することとした．具体的には，予め定めておいた各統計量の理想範囲からの逸脱値を 2 乗したものに重みをかけたものをペナルティとして合計評価値から減算し，算出することとした．その評価値を最適化に用いることで，難易度や緊張感の制御を試みた．

- ライフ数 LifeNum：テストプレイ後の残機数
- ヒヤリ数 HiyariCount：テストプレイ時に，敵や弾が自機の当り判定の 5 倍以内を通った回数
- ニアミス数 NearmissCount：同様，当り判定の 2 倍以内を通った回数
- 無行動率 NoMoveRatio：テストプレイ時に，停止を選んだ回数の割合
- 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
- 区間最大敵数 EnemyNum：ステージ全体のフレーム時刻を 11 分割し，それぞれの区間において,「画面上に最大何体の敵が登場したか」を表す 11 次元の統計量
- 区間最大敵数 BulletNum：EnemyNum とほぼ同様に，それぞれの区間において「画面上に最大いくつの弾が登場したか」を表す 7 次元の特徴量

6.2 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 試行行った．MGG-best2[29] 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．60 秒にした理由は，生成物の確認や評価をする際に，ステージが長くなるとその分だけ時間がかかってしまうため，その削減を意図してのことである．さりとて短すぎればストーリー性の感じられる緩急を持たせられないため，60 秒とした．代表的な STG, グラディウス [7] の 1 つめのステージでは，ボスが登場するまでの道中は約 2 分なため，それと比較しても短過ぎるということはない．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】
前項でも述べたように，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．

まず，無敵モードによる追加プレイを行うのはステージ全体を同じ条件で必ず評価するためである．通常のプレイでは，途中で自機が死亡してしまった際に残りの区間の敵・弾数を計測することができなくなってしまう．そのため，必ず計測できるように自機無敵モードによる追加プレイを行うこととした．

その際に無行動 AI プレイヤを用いるのは，AI の思考時間を減少させることで追加プレイ時の評価時間削減を意図してのことである．本研究では，まずは敵の撃墜を目指さないこととし，実験には射撃行動を行わない AI プレイヤを用いる．そのため，AI プレイヤの行動による敵や弾の出現数の推移への影響は少ないと考え，無行動 AI プレイヤによる追加プレイ評価を行うこととした．実際にいくつかのステージにおいて，探索 AI プレイヤ・通常モードと無行動 AI プレイヤ・自機無敵モードの両方で区間最大敵数や弾数の計測を行ってみたところ，その差はご

く僅かであった．もし将来的に射撃を行う AI プレイヤを用いて実験を行うのであれば，敵の撃墜によりその同時出現数に大きな差が出ることが考えられる．そのため，その際には無敵モード・同じ AI プレイヤによる追加プレイが必要になる．

6.3 結果

各試行の世代内評価値の推移を [-1000,0] の範囲で区切ったものを図 6.1 に示す．グラフ中の各凡例は，mean は世代内平均評価値，max は世代内最大評価値，min は世代内最小評価値，-var は世代内評価値の分散にマイナス 1 をかけたものを表す．

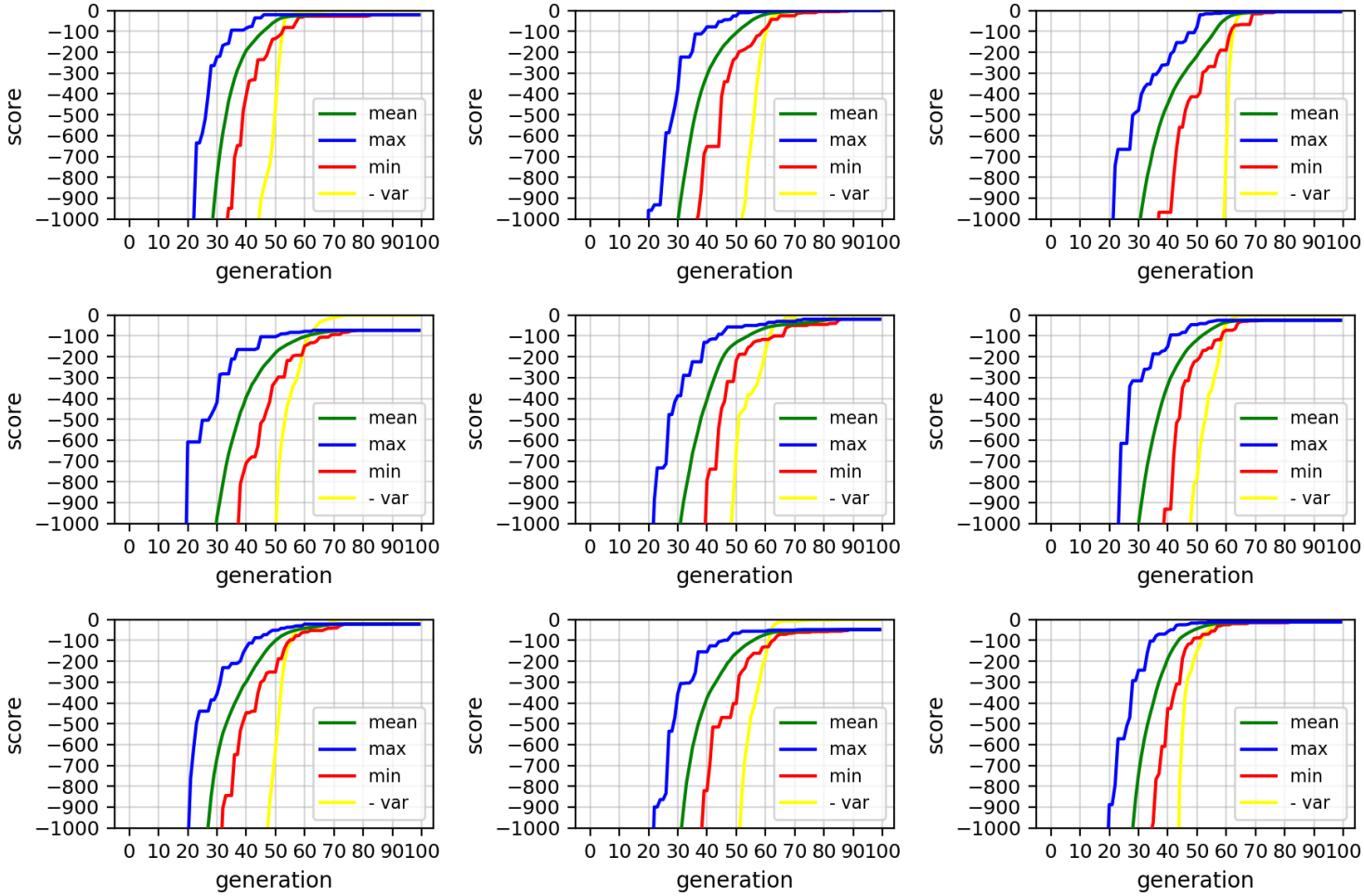


図 6.1: 世代内評価値の推移：平均・最大・最小・-分散

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．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 6.2 に示す．

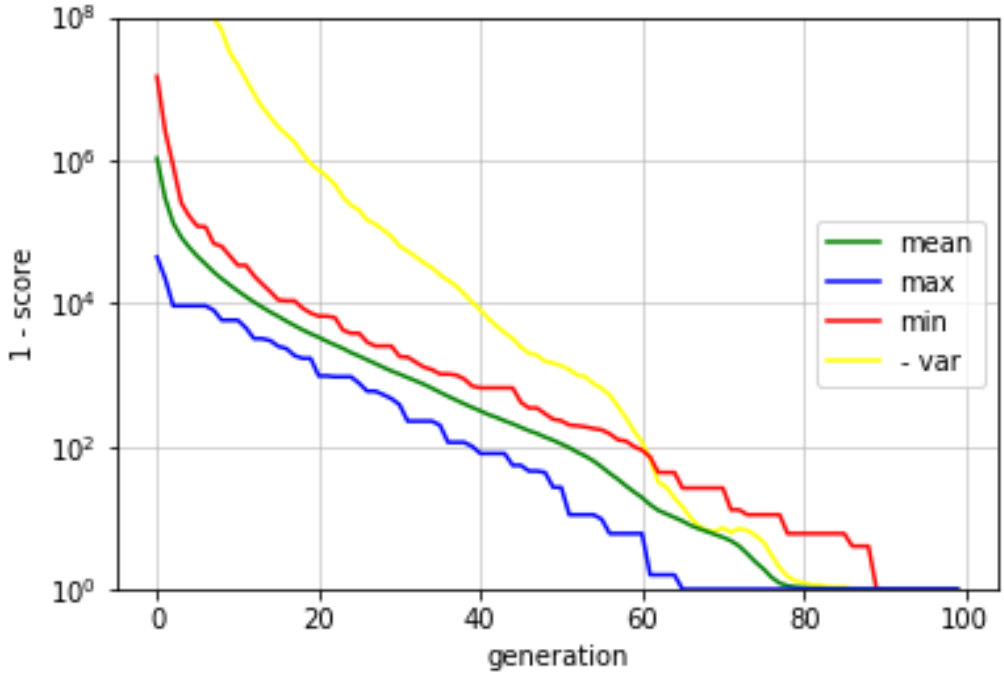


図 6.2: 世代内評価値のログスケールでの推移の 1 例：平均・最大・最小・-分散

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．

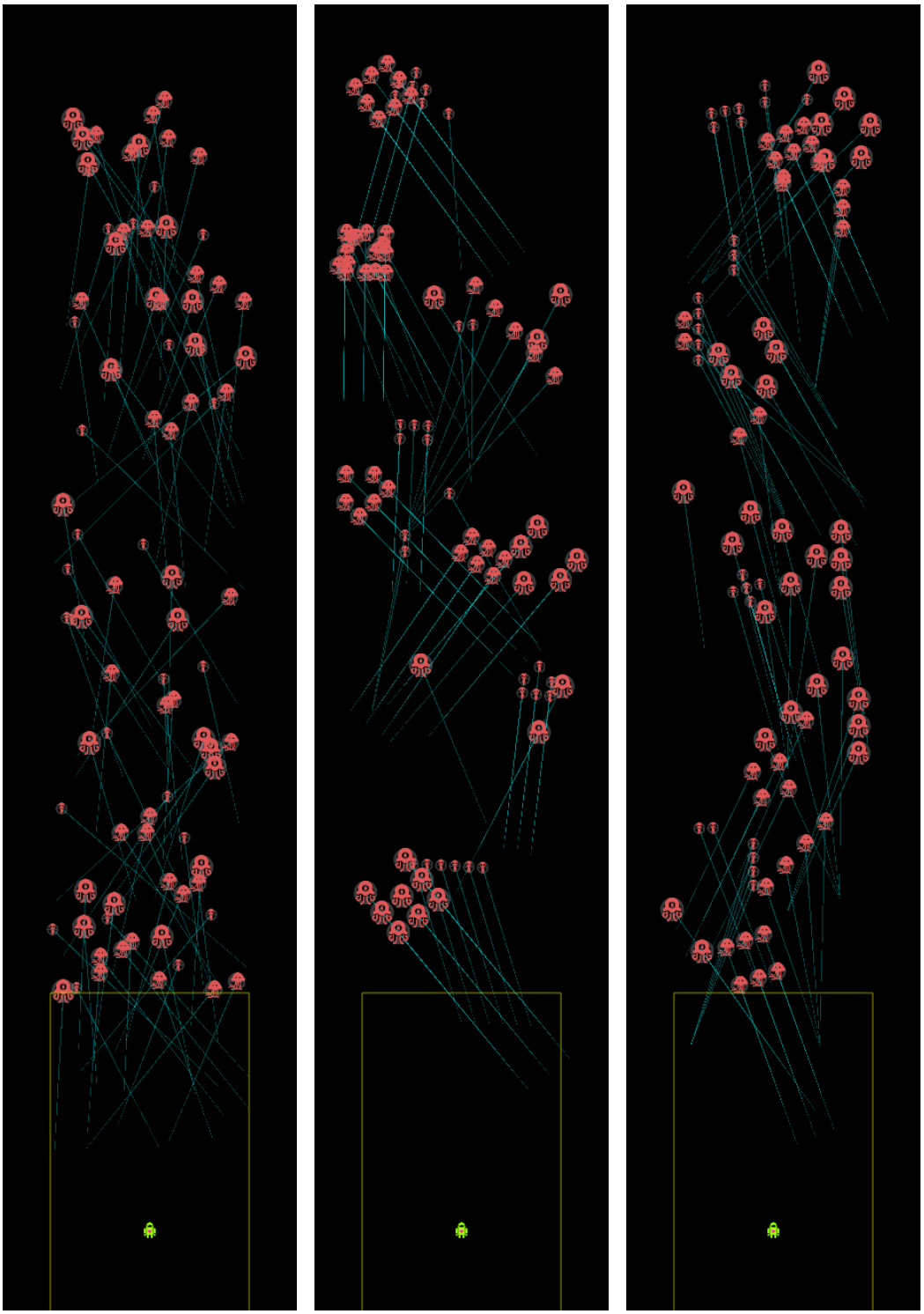
9 試行内最高評価値におけるその評価値の内訳を表 6.1 に，最低評価値における内訳を表 6.2 にそれぞれ示した．最高評価値のものは統計量が理想範囲内に全て収まっていることが確認できる．また，最低評価値のものには敵数や弾数に逸脱値がいくつか見られるが，それも 3 項目計 7 個と僅かであり，理想範囲内に収まってこそいないものの，内訳自体からは十分満足できる結果であると言える．最高評価値ステージの全体像は図 6.3 内にて示した．

表 6.1: 評価値の内訳：9 試行内最高評価値ステージ

統計量	理想範囲 最小値	最大値	実際値	逸脱値	部分評価値
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	150	0	0
無行動率	0.4	0.6	0.597	0	0
区間最大敵数					
0000-0059[f]	0	0	0	0	0
0060-0599[f]	6	10	8	0	0
0600-0779[f]	0	2	2	0	0
0780-1379[f]	9	15	10	0	0
1380-1559[f]	0	2	2	0	0
1560-2159[f]	8	12	11	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	12	0	0
3420-4799[f]	16	24	16	0	0
区間最大弾数					
0000-0119[f]	00	00	00	0	0
0120-0779[f]	05	15	11	0	0
0780-1559[f]	15	25	22	0	0
1560-2339[f]	10	20	18	0	0
2340-3119[f]	20	30	26	0	0
3120-3419[f]	15	25	24	0	0
3420-4799[f]	30	50	47	0	0
合計評価値					0

表 6.2: 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 最小値	最大値	実際値	逸脱値	部分評価値
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



(a) ランダム生成単純型　(b) ランダム生成群れ型　(c) 提案手法群れ型（最高評価）

図 6.3: ランダム生成ステージと 9 試行内最高評価値ステージの全体像

6.4 考察

図 6.3 に，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．

まずは図 6.3 の各全体像を見比べていく．(a) のランダム生成した単純型ステージがミクロ・マクロ共に多様であり統一感が全く感じられない一方で，(b),(c) の群れ型ステージにはミクロな統一感と・マクロな多様性が生じていることがわかる．マクロな多様性に関しては運頼みになってしまっているが，少なくともミクロな多様性の制御には群れの導入が有効であることがわかる．

また，図 6.3 から緊張感に関してもわかることがある．(a) のランダム生成単純型ステージからは，メリハリがあるのかないのか定かではないぼんやりとした印象を受ける．これはランダムに生成しているからというだけでなく，各敵の進行方向がバラバラなことにより生じていると思われる．一方で，(b),(c) の群れ型では，進行方向が整っており各敵も固まっているため，メリハリが感じられる．

(b) と (c) を見比べると，ランダム生成群れ型の方がメリハリがあるように見えるかもしれないが，実際は (b) は不適切なものである可能性が高い．全体像には 1 フレームを 1pixel で換算して構築したものを縮小して表示しているため，ランダム生成群れステージにおける空白地帯は見た目以上のフレーム間隔である．例えば，(b) における 2 つの群れ出現後の初めの空白地帯は約 550 フレーム，9 秒間近くもの間隔となっている．もちろん，出現した敵がどれだけ画面に留まるかはパラメータによって決まるため，時系列を距離換算した全体像を見ただけではわからない．しかし，幅が広いほど実際の空白時間も長くなりやすいため，不適切なものになりやすい．(c) の提案手法による生成ステージにはそのような長い空白はないが細かい空白があり，終始忙しいままではなく隙間隙間に余裕のあるような，適切なメリハリのあるものとなっている．次は，実際のプレイ中の特徴に関して述べて行く．

【ランダム生成単純ステージ：図 6.2(a)】
生成する敵の数にもよるが，ランダム単純ステージは多くの場合，敵の出現に規則性がないため，その移動方向を予測できない．また，そのような状況で全方位（四方八方）から弾が飛んでくるため，人間にはその軌道を読み切ることはまず不可能であり，クリアするのは困難な不適切な難易度のものとなる．また，そのような場合，終始忙しくなってしまい，緊張感にメリハリを感じることもできない．

【ランダム生成群れ型ステージ：図 6.2(b)】
ランダム群れ型ステージは，敵の出現に規則性があるため，単純型ステージに比べ，移動方向や弾の軌道の予測が格段に容易になる．また，敵を群れ単位で配置す

る関係でメリハリが生じやすい．一方で，ランダム単純に比べて，敵があまりに何も出てこ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発生しやすくもなる．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，一体の敵が大量の弾を吐き出す場合，絨毯のような多数の弾，まさに弾幕により自機が押し潰されるかのごとく逃げ場がなくなり，被弾に陥るケースがよく見られる．また反対に，プレイヤに対して何も干渉せずに去っていく敵や弾もしばしば見られる．例えば，何も撃たずに上方向に出戻りしていく敵や，プレイヤと関係ない位置に弾を発射していくような敵などがそれにあたる．このように，メリハリはあるが，あまりにもその落差の大きい理不尽・退屈な箇所が発生しやすい問題があるように感じた．

【提案手法により生成した群れ型ステージ：図 6.2(c)】
提案手法による生成ステージはこれらの問題が解決されていた．まず，区間最大敵数・弾数を考慮したことで全体の難易度が抑制された上で，緊張感にメリハリが発生していた．ライフ数を考慮したことで，死亡する程ではないがある程度被弾もするよう難易度の制御を行うことができていた．また，ヒヤリ数やニアミス数を導入したことによってか，危機的な状況や，敵や弾の間を縫って移動しなければならないような状況の発生が抑制されているように感じた．また，最大連続無行動時間を考慮したことで暇すぎる区間の発生が防止されていた．具体的には，自機位置を狙って射撃を行う敵や，画面下半分にいることの多い自機に接近しやすくなる，中央から下に対して移動する敵がよく見られたように感じた．その上で，無行動率が考慮されているため，終始忙しくなることも防止されていた．

緊張感や難易度の制御に貢献したと思われるこれらの統計量（ライフ数，ヒヤリ数，ニアミス数，無行動率，最大連続無行動数）は，人間らしさの工夫を盛り込んだテスト AI プレイヤによるプレイ結果から計算されたものである．そのため，その人間らしさの工夫がステージの難易度や緊張感の制御に貢献した可能性もあると考える．

初級者である筆者が実際にプレイしてみたところ，初プレイでクリアできるものもいくつかあり，クリアできなかったものでも，次回以降にはクリアできるように思える理不尽ではないものであると感じた．

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できた．また，人間らしい工夫を盛り込んだ AI プレイヤによるプレイ結果（ライフ数，ヒヤリ数，ニアミス数，無行動率，最大連続無行動数）と区間最大敵数・弾数により評価値を計算する評価関数によって，難易度や緊張感も制御できたと考える．

6.5　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を評価するため，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．LEBSwarm.
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．
(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．AllSwarm.
最高評価値 0 のものと，次点の-5 のものを選択．

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．面白さと難しさに関する回答項目はそれぞれ以下の通りである．

【面白さ】

1. 面白くなさそう
2. あまり面白くなさそう
3. どちらとも言えない
4. 少し面白そう
5. 面白そう

【難しさ】

1. 簡単そう
2. 少し簡単そう
3. どちらとも言えない
4. 少し難しそう
5. 難しそう

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 6.3 に手法ごとの回答結果を示す．

表 6.3: 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評価である．また，平均面白さは 2.43 と 2.36 となり，残念ながら群れを導入したことによって有意な差は得られなかった．これは，群れ型に対し，退屈・理不尽の激しい落差により評価が低く与えられてしまった可能性が考えられる．また，単純型は 6.4 節で述べたように全方位からの弾の軌道にさらされ，人間にはクリア不可能な難易度になっている．前述の通り，その平均難しさは 5.00 と，被験者全員から難しいと判断されているが，プレイしている AI プレイヤにはある程度回避が可能であったことでその理不尽さが正確には伝わらず，面白さの評価があまり下がらなかったのではないかと考える．また，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．そのため，単純型の割合を増やしつつ，実際に被験者にプレイしてもらうステージ評価実験を行う必要があると感じた．

また，ライフ数と敵数，弾数のみを考慮して生成したもの (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．また，その平均難しさは 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．また，その面白さに関しては，マン・ホイットニーの U 検定を行った結果 $p = 0.0067$ となり，提案手法により生成したあるステージが，ランダムに生成した単純型のあるステージよりも統計的に有意に面白いと判定された．これにより本手法の有効性を確認することができた．

第7章　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ステージは敵の集団を 1 要素として扱うこととした．加えて，テスト AI プレイヤによる検査と遺伝的アルゴリズムを用いた生成を行うことにより，望ましいステージの生成を目指した．テスト AI プレイヤによりステージの検査を行う際には，そのプレイ結果から得られる統計量を評価することで望ましい特徴の制御を試みた．また，検査に用いるテスト AI プレイヤには人間らしさに関する特徴が必要だと考え，そのような工夫を盛り込むこととした．

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でき，本手法の有効性を確認することができた．

今後の展望としては，評価関数にて評価する統計量に，敵の挙動の種類の多様性に関するものを用いることにより，マクロな視点での多様性の制御を試みたい．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．

本研究を通して，2D シューティングゲームにおけるステージの人間プレイヤにとっての適切な難易度や面白さを評価した上で，そのようなステージの生成が可能であることを確認することができた．また，人間らしさの工夫を入れてない AI プレイヤによる生成実験とその結果を用いた比較を行えていないため断言は出来ないが，評価に用いるテスト AI プレイヤに人間らしさの工夫を盛り込むことにより，良いステージを生成できる可能性があることを示すことができた．これらは他のゲームコンテンツの評価と生成にも利用可能な知見であるため，重要な貢献であると考える．

謝辞

主指導教員である池田心准教授と副指導教員である飯田弘之教授には，本研究を進めるにあたりご指導ご鞭撻をいただきました．深く感謝いたします．特に，池田准教授は研究や大学院生活で躓きがちな自分を温かくかつ辛抱強く教え導いてくださりました．自分とこの研究がここまで辿り着くことができたのは，ひとえに池田准教授のおかげといっても過言ではありません．重ねて，深く深く感謝いたします．

HSUEH Chu-Hsuan 助教授に深く感謝いたします．HSUEH 助教授には，本論文執筆にあたり題目や概要の英訳において多大なご協力をいただき，自分の拙い英語力では到底辿り着けない完成度にまで持っていくことができました．また，その内容についても的確な助言をいただきました．

博士後期課程の TEMSIRIRIRKKUL Sila さんにはお世話になりました．TEMSIRIRIRKKUL さんには，本研究を進めるにあたり度重なる助言をいただきました．また，遊びや食事などにもよく誘っていただき，適宜息を抜くことができました．

博士後期課程のナム サンギュさんに感謝いたします．ナムさんは PCG の研究に関して造詣が深く，本論文執筆にあたり関連研究に関して助言いただきました．

共に修士論文に取り組んだ池田研究室の 5 人のメンバー，池田裕太郎さん，森長剛志さん，石井岳史さん，及川大志さん，原口海さんに感謝いたします．皆さんと共に進められたことは大いに励みになりました．

先に修了して行った池田研究室 OB で同期の高橋竜太郎さん，高橋一幸さん，LIANG Yubin さんにはお世話になりました．共に過ごした研究室生活では元気をいただけました．

その他，池田研究室の皆さんには被験者実験にご協力いただくなど，お世話になりました．感謝の意を表します．

保健管理センターの公認心理師である中村美知恵先生に深く感謝いたします．中村先生は，学生生活や研究活動に関して何度も相談に乗ってくださり，落ち込みがちな自分を温かく励ましてくださりました．

最後に，ここまで学生を続けるにあたり温かく見守り支えてくださった家族に，心よりの感謝をいたします．

参考文献

[1] Murray Campbell, A.Joseph Hoane, and Feng hsiung Hsu. Deep Blue. *Artificial Intelligence*, Vol. 134, No. 1, pp. 57–83, (2002).

[2] David Silver, Aja Huang, Christopher Maddison, Arthur Guez, Laurent Sifre, George Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, Vol. 529, pp. 484–489, (2016).

[3] O. Vinyals, I. Babuschkin, W.M. Czarnecki, and et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, Vol. 575, pp. 350–354, (2019).

[4] Mark Hendrikx, Sebastiaan Meijer, Joeri Velden, and Alexandru Iosup. Procedural content generation for games: A survey. *ACM Transactions on Multimedia Computing, Communications, and Applications*. Vol. 9, (2013).

[5] ステージ (コンピュータゲーム) – Wikipedia. [`https://ja.wikipedia.org/wiki/ステージ_(コンピュータゲーム)`]. （アクセス：2020/02/01）.

[6] ゼビウス – Wikipedia. [`https://ja.wikipedia.org/wiki/ゼビウス`]. （アクセス：2020/01/16）.

[7] グラディウス (ゲーム) – Wikipedia. [`https://ja.wikipedia.org/wiki/グラディウス_(ゲーム)`]. （アクセス：2020/01/16）.

[8] 歴代ファミリーコンピュータソフト売上ランキング Top50【PRiVATE LiFE】歴代ランキング. [`https://entamedata.web.fc2.com/hobby/game_rank_fc.html`]. （アクセス：2020/01/16）.

[9] 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).

[10] J. Togelius, G. N. Yannakakis, K. O. Stanley, and C. Browne. Search-Based Procedural Content Generation: A Taxonomy and Survey. *IEEE Transactions on Computational Intelligence and AI in Games*, Vol. 3, No. 3, pp. 172–186, (2011).

[11] B. De Kegel and M. Haahr. Procedural Puzzle Generation: A Survey. *IEEE Transactions on Games*, (2019).

[12] Vanessa Volz, Jacob Schrum, Jialin Liu, Simon M. Lucas, Adam M. Smith, and Sebastian Risi. Evolving mario levels in the latent space of a deep convolutional generative adversarial network. In *GECCO '18: Proceedings of the Genetic and Evolutionary Computation Conference*, p. 221–228, (2018).

[13] S. Snodgrass and S. Ontañón. Learning to Generate Video Game Maps Using Markov Models. *IEEE Transactions on Computational Intelligence and AI in Games*, Vol. 9, No. 4, pp. 410–422, (2017).

[14] Adam Summerville and Michael Mateas. Super Mario as a String: Platformer Level Generation Via LSTMs. In *DiGRA/FDG '16 - Proceedings of the First International Joint Conference of DiGRA and FDG*, (2016).

[15] S. Dahlskog and J. Togelius. A multi-level level generator. In *2014 IEEE Conference on Computational Intelligence and Games*, pp. 1–8, (2014).

[16] J. Togelius, R. De Nardi, and S. M. Lucas. Towards automatic personalised content creation for racing games. In *2007 IEEE Symposium on Computational Intelligence and Games*, pp. 252–259, (2007).

[17] D. Loiacono, L. Cardamone, and P. L. Lanzi. Automatic Track Generation for High-End Racing Games Using Evolutionary Computation. *IEEE Transactions on Computational Intelligence and AI in Games*, Vol. 3, No. 3, pp. 245–259, (2011).

[18] ナムサンギュ, 池田心. 強化学習を用いたターン制 RPG のステージ自動生成. ゲームプログラミングワークショップ 2018 論文集, No. 2018, pp. 160–167, (2018).

[19] 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. デジタルゲーム学研究, Vol. 5, No. 1, pp. 51–56, (2011).

[20] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis

Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, Vol. 518, No. 7540, pp. 529–533, (2015).

[21] 池田心. 楽しませる囲碁・将棋プログラミング. オペレーションズ・リサーチ : 経営の科学, Vol. 58, No. 3, pp. 167–173, (2013).

[22] 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).

[23] Sila Temsiririrkkul, Luong Huu Phuc, and Kokolo Ikeda. Production of Emotion-based Behaviors for a Human-like Computer Player. In *GAMEON'2016*, pp. 49–53, (2016).

[24] 平井弘一, Grimbergen Reijer. 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).

[25] 佐藤直之, Temsiririrkkul Sila, Huu Phuc Luong, 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, No. 2016, pp. 57–64, (2016).

[26] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27*, pp. 2672–2680. (2014).

[27] Edoardo Giacomello, Pier Luca Lanzi, and Daniele Loiacono. DOOM Level Generation using Generative Adversarial Networks. *2018 IEEE Games, Entertainment, Media Conference*, pp. 316–323, (2018).

[28] 小高知宏. はじめての AI プログラミング　C 言語で作る人工知能と人工無能 . オーム社, (2006).

[29] Hiroaki Satoh. Minimal Generation Gap Model for GAs Considering Both Exploration and Explotation. In *Proceedings of IIZUKA '96*, pp. 494–497, (1996).