人工知能の授業のための対戦型ボードゲームプログラミングフレームワークの実装

Implementation of Player vs. Player Board Game Programming Framework for Class of Artificial Intelligence

丸山 一貴 †
MARUYAMA, Kazutaka

要旨

情報学科の 3 年次配当科目であるインテリジェントシステムでは、人工知能研究の古典的な内容であるヒューリスティック探索への理解促進と、プログラミング関連科目との連携を図る目的で、対戦型ボードゲームであるバトルシップ（海戦ゲーム）のプレイヤーをプログラムで実装する課題を出題している。本論文ではこの課題について述べると共に、C と Java で実装可能なプレイヤー開発フレームワークの設計と実装、授業での運用方法について報告する。情報学科の多くの 3 年生が持つプログラミング能力と、開発環境の利用経験を考慮する必要があることも述べる。

1 はじめに

情報学科の 2014 年度から 2018 年度入学生カリキュラムでは、コンピュータ科学コースの 3 年前期選択科目としてインテリジェントシステムを配当しており、人工知能に関する内容を扱う科目と設計されている。筆者がこれを担当するにあたり、長らく休講中であった科目の開講となることから、どのような内容を扱うべきか検討した。

近年の情勢から、機械学習の中でも特にニューラルネットワークを利用した画像等の認識技術や囲碁等のプレイヤー実装に対する注目度が高く、学生の興味もそれらに向いていることが想定された。機械学習を含む最新技術を中心に構成することも考えられたが、例えば機械学習の一手法である SVM（Support Vector Machine）の理解に必要な数学的な素養を持つ学生は本学科にはほとんどいないのが現実であり、機械学習に関する幅広い内容を理解させるのは困難である。また、Python を用いて Deep Learning を学ぶための書籍 [1] や scikit-learn [*1] といったライブラリを活用して、演習形式を主とすることも考えられたが、これらは独学も可能であり、演習や実験として設計された科目で扱う方がより適切であると判断した。

3 年生にとってはむしろ、人工知能研究の古典的な内容から現在までの発展を正しく理解して、基礎固めをする方が重要であると筆者は考えた。古典的な分野では探索や局面のモデル化といった内容を扱うが、これらは 2 年後期までのプログラミング科目や、アルゴリズムとデータ構造といった科目との連

† 明星大学 情報学部
School of Information Science, Meisei University

[*1] https://scikit-learn.org/

携も図れることから、学習内容の定着という観点でも適切である。

以上の考察から各回の講義内容を定めていったが、板書を中心とした講義だけでは出席や受講態度の悪化が懸念されたため、古典的な探索の内容を終えたところで、対戦型ボードゲームであるバトルシップ（海戦ゲーム）のプレイヤーをプログラムで実装する課題を出題することとした[*2]。類似の取り組みは先行研究に多いが、人工知能の授業で取り入れていること、C と Java という 2 つの言語で扱えるよう設計した点に特徴がある。

本論文の目的は、インテリジェントシステムの授業内課題である対戦型ボードゲームについて述べると共に、C と Java で実装可能なプレイヤー開発フレームワークの設計と実装、授業での運用方法について報告することである。本論文の構成は、第 2 章で関連研究について述べ、第 3 章で出題したプログラミング課題について説明する。第 4 章では課題で使用するプログラミングフレームワークの設計と実装について述べ、第 5 章でまとめを述べる。

2 関連研究

水口は、Java を用いたプログラミングの授業において、対戦型ゲームのプレイヤー開発という課題を導入している [2]。このゲームはターン制の陣取りゲームであり、担当授業のために新規に開発したものであり、以下の 5 つの要件を満たすよう設計した。

1. オブジェクト指向プログラミングの理解を深めさせる
2. 開発工程におけるデバッグとテストも体験させる
3. 授業に対する学生のモチベーションを維持する
4. 学生の幅広い理解度に対応可能にする
5. 類似の提出物が現れにくいようプログラムの自由度を高める

また、水口はこのゲームの設計指針について、プレイヤーが取り得る作戦を徐々に発展させられることや、フレームワークが提供する機能を限定的にするといった観点が重要であると述べている [3]。これらのうち、要件 3 と 4、ゲームの設計指針は筆者が目指したものに近いが、課題に取り組む期間は大きく異なっている。水口 [2] は週 2 回の授業で 6 週間であるのに対し、本論文で述べるインテリジェントシステムでは週 1 回の授業で 2 週間である。こうした違いは要件 1 と 2 に対応する、授業そのものの目的の違いに起因している。

尾崎ら [4] は、Java プログラミングの演習科目で扱う題材として、五五という五目並べに類似したボートゲームのプレイヤー開発を採用している。知識のある受講者が有利にならないよう、あまり知られていないゲームを選択している。学生は、局面をヒューリスティックな知識で評価して、勝てるアルゴリズムを実装することになる。インテリジェントシステムではプレイヤー開発の作り込みにあまり比重を置けないため、比較的シンプルなゲームとして商品化もされているやや著名なゲームを選択したが、既知である学生はほとんどいないのが実態である。また、ヒューリスティックな知識を用いて有効な手を選択するよう実装するという点は共通である。尾崎らの課題も 5 週間をかけて取り組むものであり、インテリジェントシステムの期間とは大きく異なっている。

富永ら [5] は、ACM ICPC [*3]のようなプログラミングコンテスト形式による初級 C プログラミングの演習課題の自動採点フレームワークを実装している。学生が課題の解答となるプログラムをオンライ

[*2] この課題と他の要件により、インテリジェントシステムは PC 演習室を使用している。

[*3] ACM International Collegiate Programming Contest。`https://icpc.baylor.edu/`。

ンジャッジサーバにアップロードすると、予め用意された入力データと出力データの組により、自動的に正誤判定が行われる。教室型は 60 分前後の制限時間で、宿題型は 1〜2 週間の期間で実施するとしている。取り組む期間という観点では近いが、インテリジェントシステムでは明確な正解があるプログラムを作成するわけではないという点が異なる。一方で、インテリジェントシステムでも同様に自動的な対戦システムが実現できれば、学生にとって有効である。これについては第 5 章で述べる。

3 バトルシップ

本章では、インテリジェントシステムで採用したプログラミングの課題について求められる要件を定義した上で、実際に採用した対戦型ボードゲームのバトルシップについて概要を紹介する。また、このプログラミング課題を授業中にどのように運営したかについて報告する。

3.1 プログラミング課題としての要件

インテリジェントシステムの授業では古典的な探索の話題から始めており、状態空間法によるモデル化と深さ・幅優先探索、ヒューリスティック探索、ゲーム木の探索という順序で進めていく。これらの内容は、2 年後期までのプログラミング科目（全て必修）で習得した内容で十分に実現可能なものであり、アルゴリズムと具体例を通じて丁寧に説明している。ここまでで概ね学期の半分程度が経過しており、座学中心の授業のペースを一度リセットして、学生がこれまで学んだ内容を考え直すきっかけとすることを主な目的の 1 つとして、プログラミング課題を取り扱う。

主な目的のもう 1 つは、過去に学んだ内容の活用や、科目間の連携を学生に意識させることである。2 年後期までのプログラミング科目では、学生はプログラミング言語の持つ機能を習得することが中心となるため、言語の機能やプログラミング技術をどのように活かすべきかを理解できていないことが多い。また、与えられた課題をクリアすることがプログラミングの授業の目的となっている学生も多く見受けられる [6]。人工知能研究はそもそも、人間の知的活動をコンピュータとプログラムによって代行させることに端を発していることから、プログラミングとは切っても切れない関係にあり、プログラミング科目の応用例の 1 つとして提示することは重要である。また、探索で扱う木構造はアルゴリズムとデータ構造の基本的な題材であり、さらに他の科目群との関連性を理解させることにも繋がると考えている。

一方で、授業の後半では知識の表現や推論、機械学習の紹介も行うことから、プログラミング課題にかけられる授業時間は 2 回程度という制約がある。課題に求められる要件は、以下の 4 つに整理される。

課題の要件 1　ヒューリスティックな手法が単純である
課題の要件 2　手法をより複雑化する余地がある
課題の要件 3　自身が選択した行動の結果を活かす余地がある
課題の要件 4　プログラミングが苦手な学生でも取り組める

要件 1 と 4 は主に時間の制約に起因している。この課題に多くの時間を割けないことから、ボードゲームの経験が浅い学生でも初歩的な戦略を思いつきやすく、また、プログラミングが苦手な学生でも最低限の処理を書けることが求められる。後者については、単純な戦略を実装したサンプルプレイヤーのソースコードを準備することでも補っている。

要件 2 は、第 2 章で述べた先行研究 [2] の要件 4 に対応しており、より高度な戦略を思いついた学生や、実装する余力がある学生のモチベーションを保つために必要である。

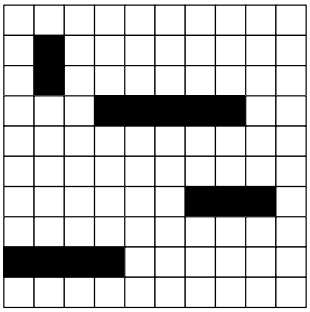

図 1　バトルシップの初期状態

要件 3 は一定のゲーム性を持たせることに関係しており、この要素がなければ戦略を考案する意義が得られない。本来、人工知能に関する課題であるならば、リバーシや将棋といった一般的な対戦型ボードゲームと同様に「自身と相手が選択した行動の結果」とすべきである。しかしながら、この課題にかけられる期間が短いこと、多くの学生のプログラミング能力ではあまり多くの要素は扱いきれないと推測されることから、敢えて簡略化した。

3.2　ゲームの概要

前節で述べた要件を満たすものとして、インテリジェントシステムでは対戦型ボードゲームとして商品化もされている、バトルシップを取り上げることとした。バトルシップは海戦ゲームとも呼ばれ、若干のバリエーションはあるものの、インテリジェントシステムではほぼ共通した仕様の中から以下のように定めた。

- 2 人のプレイヤーが交互に相手へ攻撃を行う、ターン制の対戦型ゲーム
- プレイヤーは 10×10 のマス目からなる領域を保有し、相手の領域は見えない
- プレイヤーはゲーム開始前に、自らの領域に 4 隻の艦艇を自由に配置する
- 艦艇は直線の形状であり、2 マスから 5 マスまでの長さを持つものが 1 隻ずつ存在する（図 1 は艦艇を配置した例で、図中の黒が艦艇の配置されたマス目）
- 艦艇は縦方向または横方向にしか配置できず、重ねることはできない
- プレイヤーは 1 ターンに 1 マスだけ相手の領域を攻撃でき、艦艇に命中したか否かが分かる
- 先に相手の艦艇が持つマス目を全て（つまり 14 マス）命中させた方が勝ち

艦艇が縦方向または横方向に配置されることから、仮に命中する箇所を発見した場合、そこから上下左右の隣接するマス目を攻撃する戦略は、単純かつ誰でも思いつくものである。命中箇所が見つからない場合は、相手の領域内を斜めに攻撃する方が、縦方向または横方向にしらみつぶしに攻撃するよりも有効であるという着想も得やすい。さらに、命中箇所の長さを数えることで、未発見の艦艇の長さを推定することができ、攻撃していない箇所のうち有望な場所を絞り込める可能性がある[*4]。こうした点は、第 3.1 節で述べた各要件を満たしていると言える。

[*4] カウンティングと呼ばれる手法に対応する。

3.3 授業における運営

インテリジェントシステムの授業運営では、バトルシップの課題は概ね中間試験の前後に取り上げることになる。授業時間としては 2 回分をこれに充てる。その前後 2 回を合わせた合計 4 回分の授業は以下のように配分する。

0 回目 通常授業の最後にバトルシップというゲームそのものと、Web で公開されている体験可能な実装を紹介し、次回までにどのような戦略が有効かを検討してくるよう指示する。その際、後でプログラムに落とし込むことを意識して、手順（アルゴリズム）を整理するよう伝える。

1 回目 プレイヤーを実装するために必要なフレームワークとその API 仕様（第 4 章で述べる）について説明すると共に、開発環境への組み込み方や、ごく初歩的で工夫のほとんどないサンプルプレイヤーのソースコードについて、スライドを用いて筆者と TA から説明する。フレームワークと API 仕様については A4 版の説明資料を PDF で用意してあり、要所のみ説明して、時間が不足した分は各自が確認することとしている。

学生には、考えてきた戦略を、フレームワークの枠組みの中でプログラムに落とし込む作業を次回までに進めておくよう指示する。

2 回目 この回は自習の時間であり、学生それぞれがプレイヤー実装の作業を進める。教員と TA 1〜2 名が待機して、個別の質問に対応する[*5]。ただし、多くの学生はほとんど手が付いていないか、1 回目に示したサンプルプレイヤーをそのまま取り込んだだけになっていることが多い。第 3.2 節で述べた最も単純な戦略を実現するために、どのようなデータ構造や制御構造を用いるべきかの相談も多い。インテリジェントシステムはプログラミングの授業ではないので、戦略に関するヒントはほとんど与えないが、コーディングについては積極的にアドバイスをすることとしている。

学生には次回の授業開始までにプレイヤーと説明書を明星 LMS [*6]に提出することを指示する。説明書では、実装した戦略と、実装したかったができなかった戦略を簡単にレポートさせる。また、提出期限までの間に実装で困った場合は明星 LMS の掲示板で質問したり、プログラミング個別指導室で実習指導員に相談したりするように伝える。

3 回目 この回から通常授業に戻り、板書を中心とした座学による人工知能の授業を再開する。TA は提出物をダウンロードして、提出者同士の総当たり戦を行い、勝率で順位を付けた結果を明星 LMS で公開する。公開後、上位 2 名程度には実装した戦略を簡単に教壇で紹介してもらい、戦績の上位者には成績集計時に加点することを伝えて本課題を終了とする。

受講者の増加により、最近は授業時間内に総当たり戦が終了しない。その場合、戦略の紹介はさらに翌週の冒頭で行う。

第 3.1 節で述べた通り、本課題の目的はプログラミングとの関連性や重要性を理解させることにある。プレイヤーの説明書は検討した戦略を評価するために重要であるが、プレイヤーの強弱はプログラミング能力に依存する部分が大きいため、戦績の評価は小幅な加点にとどめている。

[*5] 直近の 2019 年度は 90 名近い履修者がいるため、TA 2 名ではやや不足していた。

[*6] 明星大学情報科学研究センターが提供する LMS（Learning Management System）。

4 フレームワークの設計と実装

本章では、第 3 章で述べたバトルシップのプレイヤー開発と、一対一での対戦を実現するフレームワークについて、その概要を述べる。フレームワークはクライアントとサーバで構成されており、学生がクライアント側のプログラムを実装して動作させると、TCP/IP でサーバと通信し、相手との対戦を行う。対戦の途中経過と結果は標準出力およびログファイルに書き出すこととし、ログを可視化する簡易ツールも提供した。

4.1 フレームワークの要件

バトルシップのフレームワークを実装する上で、以下の要件を満たすよう開発した。

フレームワークの要件 1 学生がプレイヤーを開発する際、ターンごとの動作の開発に専念できること
フレームワークの要件 2 前回の攻撃結果や、敵の領域の状況を確認する手段があること
フレームワークの要件 3 学生は C と Java のいずれでも開発でき、言語に依存せず対戦できること
フレームワークの要件 4 不正な着手をなるべく許容すること

学生は主に turn() というメソッド（関数）を実装し、これをフレームワーク内からコールバックとして呼び出す。turn() 内では、このターンで打つべき手を決定し、返り値として返す設計とした（要件 1)。その際、前回の攻撃結果や敵の領域の状況を知るための API 等を利用することで、適切な手を選択するための情報を入手できる（要件 2)。

インテリジェントシステムは他コース履修を受け入れており、コンピュータ科学コースの学生は Java を、ソフトウェア技術コースの学生は C を主に学んでいることから、2 つの言語のいずれでも開発できる必要があった。そこで、プレイヤーのプログラムはクライアント、ゲームロジックを実装したプログラムはサーバとして位置づけ、クライアントとサーバは TCP/IP で通信するよう設計した（要件 3)。

第 3.1 節で述べた課題の要件 4 に挙げた通り、プログラミングが苦手な学生が諦めずに取り組めるよう、不正と思える着手を許容してエラーとはしないことに配慮した（要件 4)。バトルシップにおける不正な着手として考えられる動作は 2 つあり、1 つは既に攻撃済みの箇所を攻撃すること、もう 1 つは領域外を攻撃することである。1 つ目については、1 回目の攻撃のみ有効とし、2 回目の攻撃は全て「命中せず」（その座標に相手の艦艇はなかった）とみなす[*7]ことで、プログラム進行上のエラーを起こさないこととした。

2 つ目を許容するには、前回の攻撃が不正着手だったと区別できるように再度 turn() を呼び出して、やり直しをさせることが考えられる[*8]。この方法は別の科目で扱う同様の対戦型ボードゲームで実装した経験があるが、学生のプログラムが決定的な動作をする（乱数の要素を含まない）場合には、やり直しによってフレームワーク側と turn() の間で無限ループとなってしまう（図 2)。フレームワーク側で無限ループの対策をすることは可能だが、やり直しの仕組みとその失敗対策という説明を追加することで、プログラミングが苦手な学生にとっては取り組みのハードルがかえって高まってしまう。従って、領域外の攻撃については許容せず、直ちにエラー（対戦は敗北）とすることとした。

[*7] その座標に敵の艦艇があったとしても、後述する API で攻撃結果を確認すると、2 回目の攻撃は命中しなかったという返り値が得られる。学生がプログラムで命中回数を計数する場合を考慮して、命中の結果は 1 つのマス目で 1 回しか発生しない。

[*8] 例えば、やり直しか否かを判断できるような引数を turn() に追加する方法がある。

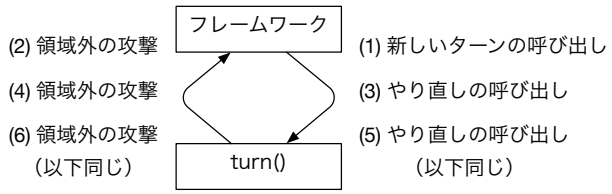

図 2　フレームワークと turn() 間の無限ループ

表 1　フレームワークの概要

名称	記述言語　ファイル数（クラス数）	行数
common パッケージ	Java　9	325
server パッケージ	Java　3	395
client パッケージ	Java　6	496
C 版クライアント	C　2	398

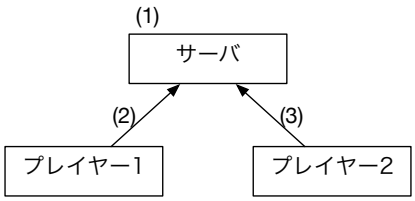

図 3　クライアントとサーバの接続状況

4.2　フレームワークの概要

フレームワークは (1) サーバ、(2)Java 版クライアント、(3)C 版クライアントからなる。フレームワークを構成するプログラムの概要を表 1 に示す。サーバは Java で記述して server と命名したパッケージに、Java 版クライアントは client パッケージに、両者で共用するクラスは common パッケージに集約している。C 版クライアントはこれらとは独立したプログラムとなっており、サーバとの通信を含む Java 版クライアントの機能の大半が実装されている。なお、表 1 に示した C 版クライアントの行数は、学生が実装すべき関数を除いている。

クライアントサーバ間にはテキストベースのプロトコルを定義し、ゲームの進行に必要な情報をやり取りする。対戦を行うには、最初にサーバを起動して、次いで 2 つのプレイヤープログラムを起動するとそれぞれがサーバに接続する。サーバは 2 つ目のプレイヤープログラムが接続されると、1 つ目のプレイヤーを先手として自動的に対戦を始める。図 3 はこれらの関係を示したものであり、図中の数字は起動と接続の順序を表している。

4.3 Java 版クライアント

先行研究 [2][4] と同様に、プレイヤーのプログラムを開発する場合はフレームワークに用意されている Player クラスを継承し、必要なメソッドをオーバーライドする方式を採用している。Java 版クライアントでは、以下 2 つのメソッドを実装する。これらはフレームワーク内からコールバックとして呼び出される。

newGame() 対戦が始まる際に 1 回だけ呼ばれる。フィールドの初期化に利用できるが、実装しなくてもよい。

turn() 領域のどこを攻撃するか、返り値として座標を返す。

turn() 内では、フレームワーク側の API を呼び出すことで、以下の 4 つの機能を利用することができる。

- 敵艦艇の残数（攻撃すべき残りのマス目の数）を得る
- 前回の攻撃結果が命中か否かを得る
- 敵の領域のマス目それぞれが以下のいずれかを得る（現時点で得られる完全な情報を得る）
  - 攻撃して命中した
  - 攻撃して命中しなかった
  - まだ攻撃していない
- 敵の領域を標準出力に表示する

Java 版クライアントを起動する際は、起動済みのサーバのホスト名とポート番号を指定する。起動時のオプションで、自身の艦艇の配置を指定することもできる。テキストベースの設定ファイルを用意することで指定可能だが、ファイルを作成しなければフレームワーク側がランダムに配置する[*9]。

Java での開発はコンピュータ科学コースの学生を想定しており、当該コースの 2 年後期のプログラミング必修科目では Eclipse 上で Java 開発を行っているため、インテリジェントシステムでも Eclipse での開発を推奨している。フレームワークの JAR ファイルをビルドパスに追加する手順は、第 3.3 節で述べた 1 回目の授業で資料を示しながら説明し、サンプルプレイヤーを動作させるところまで授業内に作業することで開発環境を整備している。

4.4 C 版クライアント

C 版クライアントで実装すべき関数や、turn() 内で利用可能な API は、Java 版クライアントと同様である。なお、C 版クライアントでは自身の艦艇をランダム配置する機能を省略している[*10]。デバッグの観点では固定配置の方が望ましいことも、ランダム配置機能を省略した理由の 1 つである。

Java 版クライアントとの大きな違いは、プログラムのファイルを敢えて分割せず、ヘッダーファイルと C ソースファイルを 1 つずつという構成にしたことである。つまり、C 版クライアントを実装する学生は、フレームワークのコードを含む単一の C ソースファイルを編集する。この構成はソフトウェア開発の観点ではもちろん望ましくない。学生が誤ってフレームワーク側のコードを変更してしまう可

[*9] 総当たり戦を行う際はランダム配置を使用している。

[*10] サーバ側には艦艇がルール通りの数だけ、重なることなく配置されているか検証する機能があり、Java 版クライアントではそのコードを共用することで容易に実装できたので追加したという経緯がある。

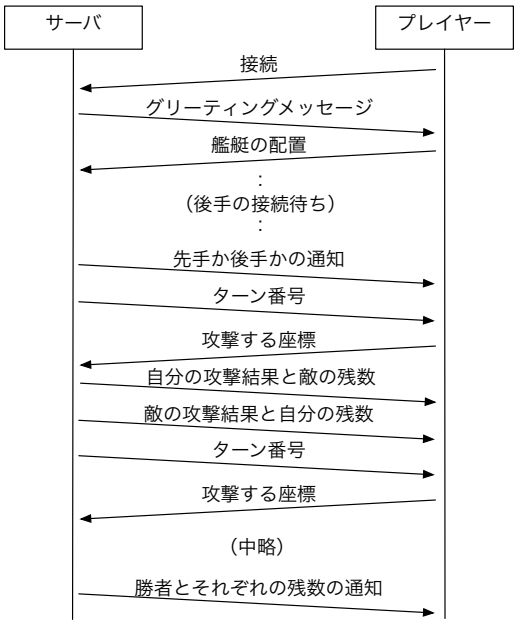

図 4　クライアントとサーバの通信の概要（プレイヤーが先手の場合）

能性があるので、本来は複数のファイルに分け、フレームワーク側のコードはオブジェクトファイルとして提供すべきである。しかしながら、分割コンパイルを身に着けていない学生や、make のようなビルドツールを使いこなせない（もしくは知らない）学生も一定数は存在すると考え、敢えて単一ファイルにまとめた。結果として、開発環境を整えるためのアナウンスや作業は不要となっている。学生には C ソースファイルの指定した範囲以外を編集しないよう伝えており、フレームワーク側のコードを誤って破壊したことに起因する問題を相談された実績は 1 件のみである。

4.5　サーバとプロトコル

サーバの動作と、クライアントとの通信の概要を図 4 に示す。サーバは、クライアントが接続してくるとグリーティングメッセージを送信し、プロトコルバージョンを確認する。次いで、クライアントから艦艇の配置が送られてくるので、艦艇の数とそれぞれの長さ、配置が重なっていないかを検証する。不適切な場合はそのクライアントを敗北として終了する。検証に成功し、2 つのプレイヤーが揃うと、各プレイヤーに先手か後手かを通知する。先手から着手の内容（攻撃する座標）が送られてくるのを待ち、その座標と攻撃結果を後手に送信する。続いて後手から着手の内容が送られてくるのを待つ、というように進行し、勝敗が決すると各プレイヤーに勝者とそれぞれの残数（攻撃されなかった艦艇のマス目の数）を通知して終了する。

対戦が開始した後で、サーバがエラーによりプレイヤーを敗北させる場合は以下の 3 つである。

1. 領域外を攻撃した場合
2. TCP が切断された場合
3. 所定の時間内にメッセージが送られてこなかった場合（タイムアウト）

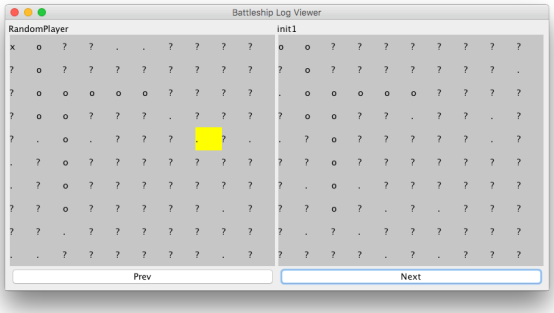

図 5　ログ可視化ツールの実行例

1 つ目については第 4.1 節で述べた。残る 2 つは通常、通信に起因するエラーを考慮する条件であるが、学生のプログラムの異常動作を検出する効果もある。2 つ目は学生のプログラムが異常終了した場合を、3 つ目は学生のプログラム内で無限ループが発生した場合をカバーすることができる。タイムアウトの時間は、同一ホストでの実行であれば 1 秒程度で十分だが、学生同士がネットワーク越しに対戦する場合を考慮して、より長く 5 秒と設定している。

対戦中のゲーム進行を詳細に確認できるよう、サーバは標準出力とは別にテキストベースのログファイルを出力する。目視で理解できるようなフォーマットになっているが、簡易的な可視化ツールを実装して、ゲームの進行を一手ずつ進めたり戻したりしながら確認できるようにした（図 5）。

5　まとめ

本論文では、情報学科の 3 年次配当科目であるインテリジェントシステムで運用しているプログラミング課題「バトルシップ」について紹介すると共に、プレイヤーの開発と対戦のためのフレームワークの設計と実装について述べた。受講生の多くは情報学科の 3 年生であり、2 年後期までのプログラミング科目の理解度を想定して、プログラミングを苦手とする学生への配慮、C における単一ファイルによる開発、複数の言語への対応といった工夫を行ってきたことを述べた。学生の提出物の内容を含む、課題運営の全体的な評価は今後別に述べることとする。

バトルシップは実質的に 2 週間という短期間での運営を行っているが、第 2 章で述べた先行研究のようにより長期に運営できる場合には、学生がプレイヤーを段階的に強化することも想定される。花川ら [7] は学生がプレイヤーを継続的にアップロードして、それらを自動的に対戦して戦績を公開すると共に、提出物が発展する過程を可視化してフィードバックすることを提案している。学生のモチベーションを向上させる上で効果的であると言え、インテリジェントシステムでも検討の余地があると考えている。

謝辞

TA としてインテリジェントシステムの授業運営に協力してくれた、小椋恵太氏と比留川翔哉氏に感謝する。

参考文献

[1] 斎藤康毅. ゼロから作る Deep Learning – Python で学ぶディープラーニングの理論と実装. オライリージャパン, 2016.

[2] 水口充. Java 言語演習科目における対戦型ゲーム課題の設計と実践. 情報処理学会 情報教育シンポジウム 2013 論文集, pp. 233–240, 2013.

[3] 水口充. 成績評価のためのプログラミングゲームの設計と実践. 情報処理学会研究報告 ゲーム情報学, Vol. 2016-GI-36, No. 16, pp. 1–7, 2016.

[4] 尾崎浩和, 富永浩之, 林敏浩, 山崎敏範. ボードゲームの戦略プログラミングを題材とした Java 演習の支援システムの開発. 情報処理学会研究報告 コンピュータと教育, Vol. 2006, No. 108(2006-CE-086), pp. 1–8, 2006.

[5] 富永浩之, 倉田英和, 林敏浩, 安藤一秋, 垂水浩幸. コンテスト形式による初級 C プログラミングの演習支援. 情報処理学会研究報告 コンピュータと教育, Vol. 2008, No. 42(2008-CE-094), pp. 49–56, 2008.

[6] 吉岡紫, 丸山一貴. 個別指導室におけるプログラミング必修科目の学修支援. 明星大学研究紀要. 情報学部, Vol. 27, pp. 7–17, 2020.

[7] 花川直己, 富永浩之. ボードゲーム戦略を題材とした Java 演習の大会中の提出コードの行数と勝点度による個人進捗の視覚化. 情報処理学会 情報教育シンポジウム 2016 論文集, pp. 151–156, 2016.