Vol.2016-EC-41 No.16
Vol.2016-EC-41 No.16
2016/8/6

成績評価のためのプログラミングゲームの設計と実践

水口 充[†1]

概要：著者らは Java 言語の基礎を扱う演習授業において，受講生の成績評価のために実施するプログラミング課題として，受講生の作成したプログラム同士を対戦させるプログラミングゲーム形式の課題を実施してきた．この課題において，受講生は与えられた親クラスを継承して，利用可能なメソッドやオブジェクトを使った思考ルーチンを実装した自分のクラスを作成する．これまでの実践により，学習者の幅広い理解度に対応可能かつ十分に自由度の高いゲームの設計指針を得ることができた．本稿では，この指針に基づき可能な限りシンプルなゲーム設計を行い，課題として実施した結果について報告する．

Design and practice of a programming game for grading skills

MITSURU MINAKUCHI[†1]

1. はじめに

演習系のプログラミング授業において，受講生の理解度を評価するためにプログラミング課題が実施される．筆記試験はプログラミングに関する知識を問うのに適している一方で，アルゴリズムの立案，デバッグ，テストといった実践的な能力を問うにはプログラム開発を課す必要があるからである．

しかし，プログラミング課題においては課題内容の設定が難しい．例えば，自由にプログラムを作成させると，受講生はどのようなプログラムを作成すればよいか困惑するであろうし，評価基準を設定することも難しい．さらに，プログラムの剽窃をチェックする手間が大変になる．逆に，詳細な仕様を与えてプログラミングさせる内容とすると，正解が存在することになるので，受講生間で似通ったプログラムになりやすい．すると，評価が画一的になりがちである．よって，プログラミング課題としては，理解度を測りたいポイントを盛り込みつつ，プログラミングの自由度が適度にある内容とすることが望ましい．

我々はプログラミングゲームを利用することでこの問題に対応してきた[1]．プログラミングゲームとは，与えられたルールのもとに思考ルーチンを実装したプログラム同士を戦わせるゲームである[2]．この実践を通じてプログラミングゲーム課題を成績評価に使用することが可能であることを確認したが，プログラミングゲームのルール設計自体は試行錯誤的に行っており，どのようにすれば労力を掛けずに課題としての要件を満たす設計ができるかは明らかにできていなかった．

†1 京都産業大学
Kyoto Sangyo University

そこで本稿では，可能な限りシンプルとしたプログラミングゲーム課題について実践結果を報告し，設計指針について議論する．

2. 課題としてのプログラミングゲームの設計指針

既報[1]では，プログラミングゲームを適切に設計することによって，モチベーションを維持しながらデバッグ・テストを含めたソフトウェア開発を経験させつつ，自由度が高く，受講生の幅広い理解度に対応可能な課題を作成することが可能であることを示した．さらにその後の 2 年間を含めた 2011〜2014 年度の実践によって，以下のような課題向けプログラミングゲームの設計指針を得ることができた．

(1) ゲームの目的と手段を明確にする

多くの受講生はボードゲームなどの思考型ゲームに不慣れであり，勝利条件やそれを達成するための手段が複雑であると何をプログラムすればよいか分からなくなると予想される．課題向けプログラミングゲームとしては面白いゲームを作成するよりも成績評価に適したゲームを作成することが優先されるので，やや単純すぎる程度の複雑さで適切と言える．これは，人間にとっては実行が容易な手段でも，初学者が意図通りにプログラミングすることは難しいことも想定している．

(2) 取り得る作戦を段階的に想定する

受講生のプログラミングスキルの幅が広い場合，低いレベルのスキルでも最低限のプログラムを作成できる一方で，高いスキルの学生はより高度なプログラムを作成できることが望ましい．さらに受講生の課題に対するモチベーションの維持，およびプログラムの評価の観点では，高度なスキルを使って時間を掛けて作成したプログラムほどゲームでは強いこ

Vol.2016-EC-41 No.16
Vol.2016-EC-41 No.16
2016/8/6

とが望ましい．これを実現するために，プログラミングスキルに応じた難易度の作戦を数パタン想定したルール設計とするとよい．

我々が実践してきた Java の基本科目であれば，簡単な条件分岐，フィールド，配列などのデータ構造，クラスの利用がプログラミングスキルの各レベルに応じている．これらのスキルを使用して作成可能なプログラムとしてはレベル順に，固定的なパタンで行動する，簡単な条件で行動を選択する，盤面の状態から行動を選択する，相手の行動を予測した上で自分の行動を選択する，といったパタンを確立してきた．

加えて，サンプルプログラムやヒントを提示する事によって，想定した戦略に誘導することもできる．これはゲームに不慣れな受講生に対しても必要な配慮である．

(3) **プログラムで使用可能な機能を絞る**

プログラミングゲームでは，対戦プログラムが行動を選択するための関数（メソッド）や，行動を決定するための情報としてゲームの状態を知るための関数や変数（フィールドやオブジェクト）が必要になる．これらの，対戦プログラムから使用可能な機能が多すぎると，それらを使いこなすことがプログラム作成の目的になってしまいがちである．本来評価したいのはアルゴリズムの立案と実装であるので，機能は極力絞った方がよい．

なお，これまでの実践では取得可能な情報を不完全にする，結果の判定にランダムな要素を利用するなど，不確実性を導入することで機能を絞ることに成功してきた．今回は不確実性を使用せずに複雑なゲームを作成する方針とした．

3. ロボバトラーV の設計

2015 年度秋学期に開講したプログラミング演習科目にて実施する最終課題としてプログラミングゲーム「ロボバトラーV」を設計した．当該科目はコンピュータ理工学部 2 年次配当の必修科目であり，Java 言語を通じてオブジェクト指向プログラミングの基本を扱う内容であった．2011〜2014 年度にプログラミングゲーム課題を実践したのも同じ科目である．

以下，ロボバトラーV のルール設計の過程を紹介する．最終的なルールは付録 A.1 の通りである．また，ゲームのスクリーンショットを図 1 に示す．適宜参照されたい．

まず，モチーフとなるゲームとしてスプラトゥーン™[3]に着目した．ゲーム設計に当たっては主題を設定しておくことが重要であるが，既存のゲームを

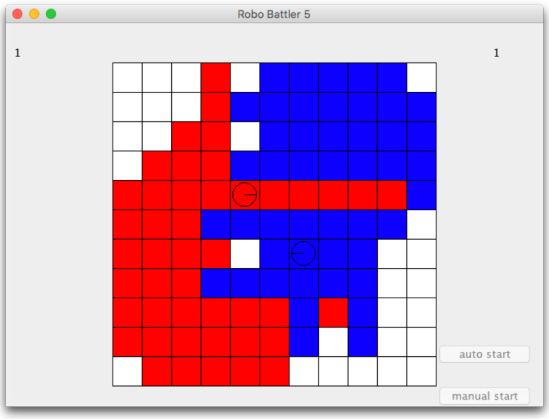


図 1. ロボバトラーV のゲーム場面．円は各プレイヤー，赤と青のマスはそれぞれのプレイヤーの陣地を表している．

Figure 1. Screenshot of RobobattlerV. The circles indicate the players. The red and blue squares are the possessed area by the players.

参照すれば手間が軽減される．プログラミングゲーム課題においては純粋に新しいゲームを作成することが目的ではないので，手頃なゲームを参考にすればよい．さらに，流行などを考慮して受講生の多くが興味を持ちそうな題材であればなお良いであろう．

因みに，著者はスプラトゥーン™を実際にプレイしたことはまったく無く，商品紹介サイトや各種記事を通じて概要を知っている程度であった．つまり，参考にするゲームについては詳細を知っている必要はなく，面白さのポイントを適当に流用できれば十分と言える．

筆者の理解では，スプラトゥーン™は次のようなゲームである：基本的には Third-person shooter (TPS)，すなわち三人称視点で互いに撃ち合うゲームであるが，武器（ブキ）は相手を撃つだけでなくゲームフィールドを塗ることができる．制限時間終了時に，より多くのフィールドを自分の色で塗っているプレイヤーが勝ちである．撃たれたプレイヤーはスタート地点に戻される．つまり，相手を撃つことはフィールドを塗るために有効な手段の一つであるが，これ自体が目的ではない．囲碁に似た戦略性のあるゲームである．スプラトゥーン™はリアルタイムゲームであるが，ロボバトラーV ではターン制，すなわち手番が交互に来て，自分の手番で行動を選択するルールとすることにした．

陣取りタイプのゲームは目的が明確であり，状態を視覚化しやすい．手段についても，直接塗る（陣地を獲得する）か，相手を攻撃して撃退する，のい

ずれかで明確である．一方で，プログラムが効率的に振る舞うためには，ゲームフィールドの状態を分析し，相手の行動を予測する必要がある．このように，2 章で挙げた「(1)ゲームの目的と手段を明確にする」を満たすことができると考えた．

一方，「(2)採りうる戦略を段階的に設定する」と「(3)プログラムで使用可能な機能を絞る」を実現するには工夫を要した．

まず，ゲームフィールドを 11×11 の格子に単純化して表現した．データ構造としては 2 次元配列を使って，それぞれのマス目の所有者をなし／自分／相手の 3 状態に対応する値で表現すればよいので，ある程度のプログラミングスキルを有していれば十分扱える程度の難度とすることができる．ゲームフィールドの大きさは 1 ゲームの長さやゲームの局面の数を想定して多すぎず少なすぎずとなるように直感的に決定したが，概ね妥当で後から調整する必要は無かった．

次に，プレイヤーの表現を検討した．ロボバトラーV では，プレイヤーはプログラムが操作して動かすゲーム中のキャラクタである．ゲームフィールドは格子で表現したので，プレイヤーはいずれかのマス目に位置する．プレイヤーの向きについては，上下左右の 4 方向とするか，斜めを加えた 8 方向とするかが考えられる．過去に実施した初代ロボバトラーの経験では 8 方向の移動は配慮すべき要因が多くなりすぎて作成するプログラムが難しくなる．そこでゲームの複雑さとしては単純すぎることが予想されたが，今回は 4 方向とした．この場合，相手を攻撃することが難しくなることも予想された．この点についてはテストプレイで検証し調整することにした．

ところで，スプラトゥーン™の特徴の一つに多様なブキがある．ブキは攻撃範囲や射撃速度等に変化をつけることでプレイヤー毎に有効な作戦を変える，すなわちゲームのバリエーションを広げる効果がある．ロボバトラーV においては，ブキが 1 種類しかないと固定的な行動パタンを 1 つ作るだけで適度に強いプログラムが作成できてしまう危惧があり，逆に多すぎると作戦を立てにくくなると考えた．そこで, 相手を遠くから攻撃できるタイプ(シューター)，広く塗りつぶしやすいタイプ（ローラー），それらの中間的なタイプ（スロッシャー）の 3 種類のブキを用意することにした．具体的な攻撃範囲は付録のルールに示したとおりである．一般的なゲームではプレイヤーが使用するブキを選ぶのが普通であるが，課題の観点では 1 種類のブキだけに対応するプログラムを作ればよいのは不都合である．そこで，使用するブキは対戦プログラムからは選択できず，ランダムに指定されることにした．ただし，選択の偏りによってゲームバランスが悪くならないように，同じブキでない場合はブキを入れ替えて連戦するようにした．

最後にプレイヤーの採り得る行動を設定した．通常の TPS では，自由に移動する，向きを変える，武器を使う（射撃する），弾を補充する，といった行動が基本であり，その他多様な行動がルールに応じて設定される．ロボバトラーV の場合，前述のようにプレイヤーの向きは 4 方向のみとしたので移動に関しては自由度が制約されるが，それでも隣接 1 マスの前後左右の移動に加え方向転換の有無，その場での方向転換を含めると 10 種類が必要になるし，2 マス以上の移動も許すとさらにバリエーションは増える．過去の実践経験からは，これは多すぎて受講生の手に負えないことが予想された．そこで，まず極限まで機能を絞って，移動については 1 マス前進する，左右に 90 度回転する，の 3 種類のみとした．

さらに攻撃については，タイミングを図って射撃するのが TPS の基本であるが，これは弾数に制限がある，連続的に射撃できない，といったルール上の制約があって成り立つ要素である．このルールはゲーム性を高める上では有効であるが，今回は制作期間の短さを考慮して敢えて外すことにした．そうすると射撃する，弾を補充するといった行動は意味が無くなるので，毎回自動的に射撃する，という設定にした．

以上のルール設計では，1 度の行動ターンで採り得る行動の種類は，「何もしない」を含めると 4 種類のみとなる．

ここまでのルール案を元に，紙上でゲーム進行をシミュレートした．その結果，シューターとスロッシャーはジグザグに進み，ローラーは盤上を周回するような動き方が，効率的に塗りつぶして陣地を獲得する動きのパタンであることがわかった．これらの動きは，状態を記憶する変数と簡単な条件判定のプログラムで実現できる．これは 2 章の(2)で述べた「固定的なパタンで行動する」に相当する．相手プレイヤーを狙って攻撃するのは難しいが，偶発的に攻撃できる位置に来たときに攻撃することは容易である．これをプログラムで実現するには，自分と相手の位置，および使用しているブキの攻撃範囲を参照し，攻撃可能か判定する必要がある．これは「簡単な条件で行動を選択する」に相当する．さらに戦略的に行動するには，空白地よりも相手が塗ったマ

ス目を奪いに行く，相手に攻撃されないように行動する，相手を攻撃しやすいように行動する，ブキの相性を考慮する，などの工夫が考えられるが，いずれもプログラムの難度は上がる．

ここまでのルール設計で概ね 2 章で述べた 3 項目を達成してプログラミング課題として適切な難度であると判断し，ゲームプログラムを作成し，サンプルの対戦プログラムを試作した．サンプルプログラムとしては，受講生がテンプレート的に使用できるようにランダムに行動を選択するプログラムと，簡単な条件で行動を選択する程度のプログラムを作成した．後者は 4 種類の行動のうち，相手を攻撃できる行動を選択，攻撃できない場合は獲得できるマス目が最大となる高度を選択するプログラムである．これは固定パタンを実装していないが，挙動としてはゲーム序盤では固定パタン的に振る舞い，中盤以降は状態に応じて行動を変化させる．初級から中級レベルの作戦を示唆するには適切なサンプルとなった．またこの試作を通じて，課題遂行に要するプログラミングスキルのレベルが適切であることを確認できた．

ただし，固定パタンの存在を示すと，類似したプログラムを提出する受講生が増える懸念がある．そこで，1 ゲームのターン数を 100 に設定した．これはジグザグに進む固定パタンの場合，対角まで到達しても自分の手番が十数ターン余る数である．このような残りターンを設けることで初級レベルのプログラムにも工夫の余地を残すことができた．

ところで，これまでのプログラミングゲームの設計では，不完全情報や偶然の要素を導入することでルールを単純に維持しつつプログラムで配慮するべき要素を増やすようにしたが，ロボバトラーV では不確実要因はブキの選択のみとした．毎回の行動は 4 択であるが全体では 4^{100} の局面となり適度に複雑で，強いアルゴリズムが勝ち易いと予想した．完全情報ゲームとするとミニマックス探索などを使用したプログラムが提出されることが予想される．ロボバトラーV のルールでは評価関数の作成も容易であるが，本演習のレベルとしてはこのようなプログラムを作成するスキルは十分に高いと言える．

なお，ロボバトラーV を作成・実践した後に判明したのであるが，ゲーム的には SamurAI Coding[4] に似た点も多い．事前に参照したことは無かったの偶然似てしまったのであるが．SamurAI Coding は競技用ゲームであるため複雑なルール設計となっているが，ロボバトラーV は初学者のスキル評価に使用することを目的としているためごく単純化している点が大きな違いである．

4. 実践

受講生には課題の指示として，ゲームのルール，ゲーム本体の jar ファイル，API ドキュメント，ランダムに行動を選択する対戦プログラムのサンプルソースコード，簡単な条件で行動を選択するサンプル対戦プログラム（class ファイルで提供），コンパイル・実行およびプログラム作成のヒントのドキュメントを配布した．課題の提出締切は，課題内容を説明してから 25 日後に設定した．

142 本の課題プログラムが提出され，うち 2 本はプログラムの体裁を成していなかった．残り 140 本のうち，7 本はサンプルプログラムを少しだけ改変した程度の，基本的にランダムに行動を選択するタイプであった（ランダム型）．74 本は固定パタンを実装したが（固定パタン型），次の行動を決定するために座標と向きを使うもの，ターン数を使うもの，フラグを使うもの，など実装方法は様々であった．また，多くはブキがシューターとスロッシャーの場合はジグザグに進むパタンを実装していたが，予期したとおり対角に達した後の行動は考慮しないものから別パタンを作るものまで多岐に渡っていた．27 本は更に敵を攻撃する等の条件に応じた行動選択を実装していた（条件判定型）．28 本は盤面の状態から最適と思われる行動を判断するタイプの実装であった（評価関数型）．そのうち 3 本は先の手まで読む探索型の実装であった．2 本は機械学習に挑戦したプログラムであった．

140 本のプログラムで 1 次から 3 次予選および決勝リーグからなる大会を実施した．それぞれの段階で約半数が敗退した．それぞれのリーグに勝ち進んだプログラムの本数の，思考アルゴリズムの種類毎の内訳を表 1 に示す．実装の不具合や対戦相手の偏りなどの要因による例外はあるが，基本的に上位のアルゴリズムのプログラムほど強い傾向となった．機械学習型については学習にかける時間が不足したため 1 次予選で敗退する結果となったようである．

表 1. 大会の段階および思考アルゴリズムごとのプログラムの本数
Table 1. Numbers of programs for the stages of the championship and the types of thinking algorithms

	ランダム	固定パタン	条件判定	評価関数	探索	機械学習
1 次予選	2	34	17	7	1	2
2 次予選	2	26	4	4	0	0
3 次予選	0	9	4	5	0	0
決勝	0	5	2	9	2	0

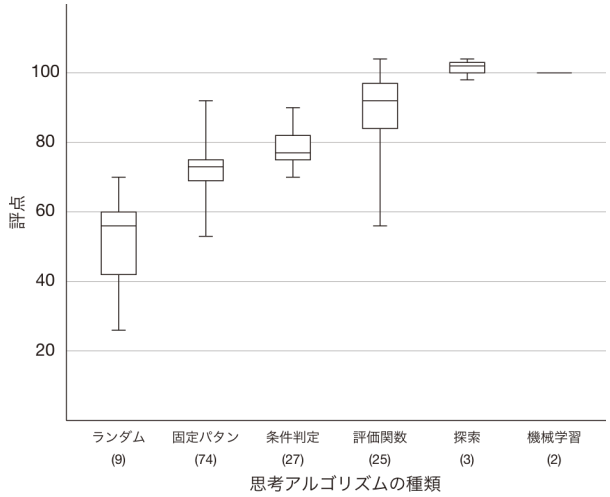


図 2. 思考アルゴリズムの種類ごとの評点の分布
Figure 2. Distributaions of scores for the types of thinking algorithms.

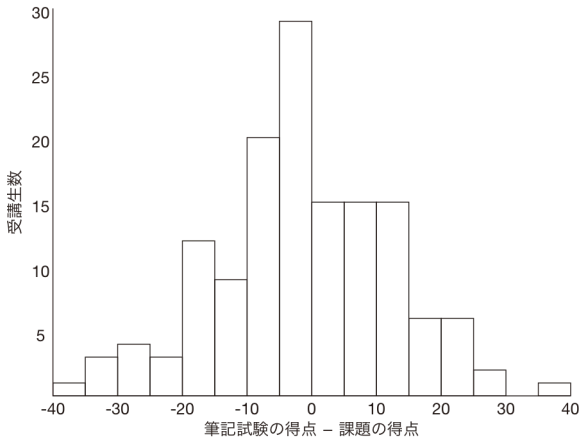


図 3. 筆記試験とプログラミング課題の得点差の人数分布
Figure 3. Distribution of the difference between the paper test and the programming assignment.

図 2 はアルゴリズムの種類毎の，プログラムの評点の分布である．評点はプログラムの強さ自体とは無関係に，プログラミング技術，アイデア，オリジナリティ，完成度の観点で評価した．計算の都合で 104 点満点とした。上位のアルゴリズムを実装したプログラムほど評価が高い傾向を示している．

図 3 は，別途実施した筆記試験の得点と，本プログラミング課題の評点との差の人数分布である．筆記試験はオブジェクト指向やクラスライブラリ，および基本的な文法やアルゴリズムに関する知識を問う内容であるため，試行錯誤を含めた実技であるプログラミング課題の評価とは必ずしも一致するものではないが，概ね一致していると言える．

5. 考察

実践結果より予め想定したとおり，思考アルゴリズムの種類に応じてより強く，より評点の高いプログラムが作成されたことが確認できた．これは，受講生にとってより強くするために高度なプログラミングに挑戦するモチベーションを与える上で役に立つ．また，評価する教員にとってもある程度の評価指針となるので有用であるが，例外的なプログラムがあることに注意する必要があるだろう．よく考えられていて十分に試行錯誤した完成度の高いプログラムは下位のアルゴリズムであっても強いプログラムとなる．あるいは，アイデアは高位のアルゴリズムでも不具合を含んでいたり，テストが不十分であれば弱くなる．

幅広い強さの対戦プログラムが作成され，特に勝ち残ったプログラムは安定して強かったことから，ルール自体はシンプルに保ちつつ適度に複雑なゲームが設計できたと言える．このようにできた要因としては，1 ゲームのターン数とゲーム盤の広さが挙げられる．

ターン数については 3 章の設計で述べたとおり，局面数を増やすことで複雑化できる．完全情報ゲームの場合，1 手あたりの可能手が少なく目的が明確なゲームでは先読みが容易であるが，今回実践した演習科目の受講生のレベルではトップ層のみが実装可能であったため特に問題にならなかった．受講生

のレベルが高い場合はルールに不確実性を導入して対策するのがよいであろう．

ゲーム盤の広さは，広いほど局面数が増え複雑化するが，ゲームにおいて配慮するべき制約として重要な要素であるので，ルールとの兼ね合いで適度に設計することが重要となる．例えばロボバトラーVでは，盤外となる攻撃範囲は無駄になるようにし，シューターとローラーの攻撃範囲が盤外に差し掛かりやすいような広さとした．これにより，行動を選択するための配慮要因を増やすことができる．例えば，多少攻撃範囲を犠牲にしても先の行動を見越して移動するといった選択を考慮する必要が生じる，などである．このようなジレンマを仕込んでおくことはゲーム性を高める上で重要であるが，課題として使用するプログラミングゲームにおいてはどの程度の複雑な処理になり，受講生のスキルで扱うことが可能かを想定しておく必要がある．

ロボバトラーVでは，予め想定した作戦パタンに相当するプログラムが大半を占めたが，一部想定外なプログラムも見受けられた．例えば機械学習を使用したプログラムは，過去のプログラミングゲーム課題では見受けられなかったので想定していなかったが，昨今のディープラーニングブームを鑑みると想定しておきべきだったかもしれない．また，条件判定型以上のタイプにおける詳細な作戦は自由度が高いため想定し切れていなかったが，実際に多様なアイデアが提出プログラムから見受けられた．例えば，対戦型シューティングゲームにおけるリスポーンキル[1]というテクニックを意図的に実装したプログラムが数件見受けられた．

6. まとめ

成績評価に使用するプログラミングゲーム形式の課題について，これまでに得られた設計指針に基づいて可能な限り簡潔でありながら成績評価を行うために十分な程度の複雑さを持つゲームルールを設計し，実践した．実践結果より，想定通りに設計され，プログラミングスキルを概ね妥当に評価できたことが確認された．

謝辞　当該授業を共同で運営した玉田春昭氏およびTA諸氏に感謝いたします．本研究の一部はJSPS科研費15K00510の助成を受けたものです．

1 相手を倒した際に復活する地点で待ち構えて，復活した瞬間に再度攻撃して倒し相手をピン留めする作戦．対戦プレイではルール違反ではないものの不適切な行為として忌み嫌われている．

参考文献

[1] 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. 情報教育シンポジウム2013論文集, 2013(2), pp. 233–240.
[2] Programming Games Wiki. http://programminggames.org, (参照2016-07-09).
[3] スプラトゥーン. https://www.nintendo.co.jp/wiiu/agmj/, (参照2016-07-09).
[4] SamurAI Coding. http://samuraicoding.info/index-jp.html, (参照2016-07-09).

付録

付録 A.1 ロボバトラーV ゲームルール

1. 概要

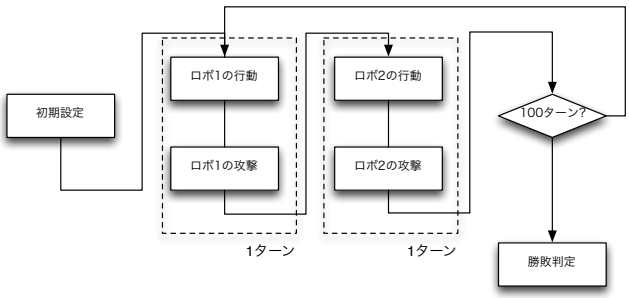
ロボバトラーVは，2体のロボプログラム（以下，単にロボと呼ぶ）が対戦する陣取りゲームです．ゲーム開始時に，双方のロボはスタート位置に配置され3種類のブキのいずれかを指定されます．その後，双方のロボは交互に移動します．移動後，ブキの種類に応じたマス目を自分の陣地として塗りつぶします．また，塗りつぶしたマスに相手のロボがいた場合は，相手のロボはスタート位置に戻されます．100ターン終了したときに陣地のマス目の多い方が勝ち（同数の場合は引き分け）です．このゲームを7戦し，勝ち数の多い方が総合勝利です．

プレイヤーは自分のロボを，Roboクラスを継承して作る必要があります．

Roboクラスは1ターンごとに呼ばれるactionメソッドを抽象メソッドとして定義しているので，プレイヤーは少なくともactionメソッドを実装しなければなりません． また，ゲーム開始時に呼ばれるinitGameメソッドをオーバーライドして実装することもできます．

つまり，プレイヤーはロボの思考ルーチンをプログラミングすることになります．

2. ゲームの流れ

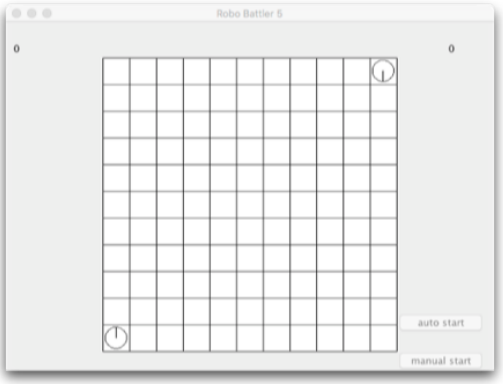


3. バトルフィールド（盤）

ロボバトラーVでは，ロボは縦横11マスのバトルフィールド（盤）で戦います．双方のロボは円＋ロボの向きを表す線で表示されます．赤と青色のマス

Vol.2016-EC-41 No.16
Vol.2016-EC-41 No.16
2016/8/6

は双方の陣地，白はどちらにも属さないマスです．ゲーム開始時には，ロボは左下のマスに上向きと，右上のマスに下向きで配置されます．バトルフィールドの座標および向きの扱いについてはプログラミング説明書を見てください．



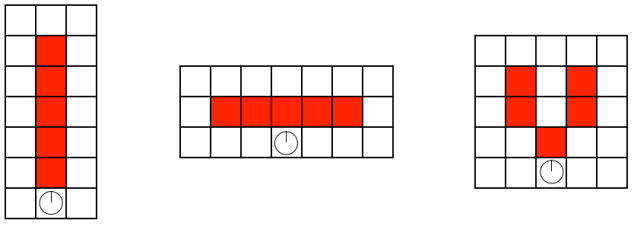
4. ブキ

4.1 ブキは自分の陣地を得，相手を攻撃する道具です．攻撃範囲の異なる 3 種類のブキがあります．

4.1.1 シューター shooter 前方をまっすぐ攻撃するブキ

4.1.2 ローラー roller 横に広く攻撃するブキ

4.1.3 スロッシャー slosher やや広範囲に攻撃するブキ



4.2 ブキによる攻撃（攻撃範囲のマスを自分の陣地にし，範囲内に相手のロボがいた場合は相手のロボをスタート位置に戻す）は，各ターンの移動後自動的に行われます．

4.3 使用するブキはゲーム開始時に 3 種類のうちのいずれかが自動的に指定されます．ロボが選択することはできません．

4.4 ブキの選択は，1 ゲーム目は双方ランダムに選択します．以降のゲームでは直前のゲームで同じブキを選択したかあるいは互いにブキを入れ替えて選択した場合はランダムに選択，そうでない場合は直前のブキを互いに入れ替えて選択します．

5. 行動

5.1 ロボは自分のターンで，ロボが向いている方向に 1 マス前進する，ロボが向いている方向を左右いずれかに 90 度回転する，のいずれかを 1 つの行動を行うことができます．あるいは，いずれの行動も選択しないことも可能です．

5.2 1 ターンに選択できる行動は 1 つだけです．複数の行動を行おうとした場合は，最初に行った行動のみが実行されます．

5.3 バトルフィールド外に出ることはできません．前進した結果場外になる場合は，その前進は無効になります．

5.4 双方のロボが同じマス目に入ってもかまいません．

6. 勝利条件

6.1 100 ターン終了時に陣地のマス目の多い方が勝ち（同数の場合は引き分け）です．ただし 100 ターン以前でも，プログラムの不備によりエラーを発生，あるいは無限ループに陥るなどで，続行が不可能と判定された場合は不具合を発生したほうが負けとなります．

6.2 1 回の対戦は 7 本勝負です．勝ちゲーム数の多い方が総合勝利となります．

6.3 先攻・後攻はゲームごとに交互に入れ替わります．ただし，最終ゲームの先攻は抽選で決定します．

6.4 どちらかのロボが不具合を起こして試合続行不可能にならない限り，ゲームは連続して行われます．このため，前のゲームで得た情報を保持しておき利用することは可能です．