コンピュータプレイヤー同士の対戦を通した プログラミングコンテストのパターンランゲージ

坂本 一憲　　大橋 昭　　志水 理哉
高橋 周平　　村上 真一
鷲崎 弘宜　　深澤 良彰

早稲田大学
基幹理工学研究科　情報理工学専攻

近年，プログラミングコンテスト，特に，題材となるゲームソフトウェア上でコンピュータが操作するプレイヤーの思考ルーチンのプログラムを作成して競い合う形式のコンテストが，教育向けの枠組みとして注目を浴びている．一方，こうしたゲームソフトウェアはユーザが直接プレイするゲームソフトウェアやエンタープライズシステムとは異なり，開発手法や技術的な知見が体系化されておらず，ノウハウが共有されずに開発される傾向にある．本論文では，我々が主体となって開催した 3 つのプログラミングコンテストの開発と運営を通して得られた知見をゲームソフトウェア開発者の視点とプログラミングコンテスト運営者の視点の両方からパターンを提案する．さらに，それらのパターンを整理してパターンランゲージを提案する．

A Pattern Language for Programming Contest through Fight between Computer Players

Kazunori Sakamoto, Akira Ohashi, Masaya Shimizu,
Syuhei Takahashi, Shinichi Murakami, Hironori Washizaki
and Yoshiaki Fukazawa

Waseda University
Dept. Computer Science

1. はじめに

コンピュータ向けゲームソフトウェア（以降，ゲーム）の普及が進み，ゲーム産業が国内における重要な産業の 1 つとして認められている．こうした中，ゲームにおいて，コンピュータが操作するプレイヤーの思考ルーチンのプログラム（以降，AI プログラム） を作成して競い合うコンテスト（以降，AI コンテスト） が活発に開催されている．AI コンテストはプログラミングの面白さを伝え，また，学習機会を創出できるため，有用な学習機材として教育界から注目を浴びている．

AI コンテストの例として，国際大学対抗プログラミングコンテスト（International Collegiate Programming Contest；ICPC）[1] と情報処理学会創立 50 周年記念事業のコンピュータ将棋プロジェクト特別対局 [2] が挙げられる．ICPC は ACM（Association for Computing Machinery）が主催する，大学生を対象にした世界的規模のプログラミングコンテストである．1976 年度から開催

され，2010 年度で第 35 回目を迎える．2009 年度大会では，全世界 82 カ国から 1,931 大学，約 22,000 人が世界各国で開催された地区予選に参加している．さらに，ICPC 内で併設されるコンテストの JavaChallenge では，Java 言語を用いてゲームの AI プログラムを作成して競い合う．また，コンピュータ将棋プロジェクト特別対局は，情報処理学会が開催したコンピュータ将棋の「あから 2010」と女流棋士が対戦するコンテストである．このように，AI プログラム同士の対戦のみならず，AI プログラムと人間が対戦する場合もある．

AI コンテストでは，AI プログラム同士が対戦するために，題材となるゲームが必要となる．題材となるゲームの例として，将棋やチェスなど古くから遊ばれているボードゲームに加えて，IBM 社が開発した Robocode [3] や CodeRally [4] などが有名である．Robocode と CodeRally はユーザがロボットを制御する AI プログラムを作成して対戦するゲームである．

我々は，2009 年度に開催された ICPC のアジア地区予選内の併設コンテストである JavaChallenge [5]，2010 年度に開催された ICPC の国内予選内の併設コンテストである JavaChallenge [6]，そして，2010 年度に開催された楽天テクノロジーカンファレンス[7] の 1 セッションである早稲田楽天プログラミングコンテスト[8] の 3 回に渡る AI コンテストの運営と，題材となる 3 種類のゲームの開発を経験した．本論文では，上述した経験から得られた知見をいくつかのパターンで形式化して，パターンランゲージに向けてパターンをまとめあげる．

2. パターンランゲージの全体像

本論文で取り上げるコンピュータプレイヤー同士の対戦を通したプログラミングコンテストのパターンランゲージは，題材となるゲームのゲームデザインパターンと AI コンテストの運営パターンの 2 種類に分類される．これらのパターンのパターンマップを図 1 で示す．なお，ゲームデザインパターンとは，ゲームデザインのパターンであり，デザインパターンと異なる．

ゲームデザインパターンは，ゲームの形式やルールを決定する際に利用されるパターンである．本論文では以下のパターンについて取り上げる．

- クライマックスパターン
- 擬似リアルタイムパターン
- 仕切り直しパターン
- 縮退盤面パターン
- 戦略と戦術パターン
- 3 プレイヤー以上同時対戦パターン
- 三すくみパターン

AI コンテストの運営パターンは，AI コンテストを運営する際に利用されるパターンである．本論文では以下のパターンについて取り上げる．

- 予選と本戦パターン
- フィードバック期間パターン
- API 固定パターン

本論文で利用するパターンテンプレートの項目は以下のとおりである：Context, Problem, Forces, Solution, Consequences, Our Uses, Known Uses, Related Patterns．パターンテンプレートは久保の研究 [9] によるものをベースに Examples を Our Uses へと置き換えて利用している．これは，本論文が我々の開発経験を元に執筆しているため、Examples という項目を取り上げる場合、内容がほとんど我々の開発経験に基づくものになってしまうためである。したがって、Known Uses との対比を兼ねて、Our Uses という形で、我々の開発経験の説明の項目も追加した。

なお，本論文で利用する用語として，観客とは AI コンテストの試合を観戦する人のことを指す．

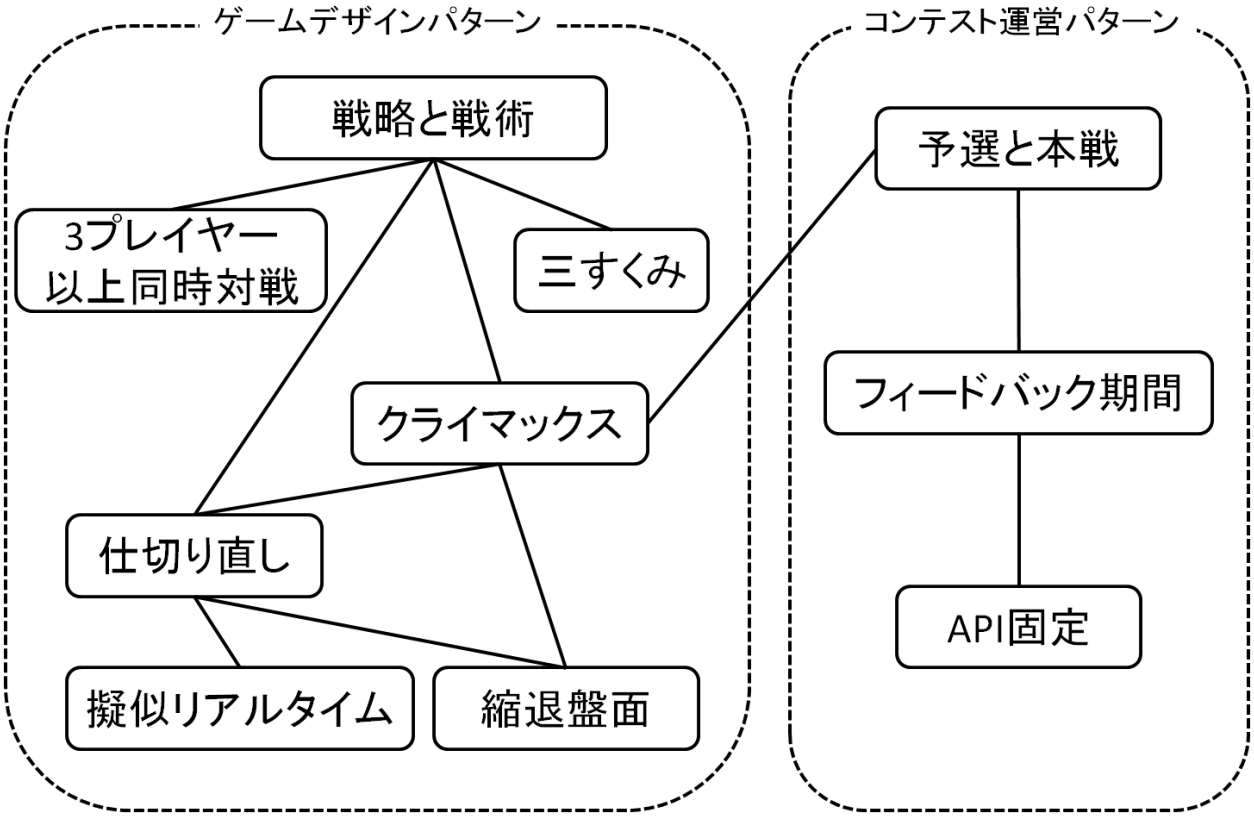


図 1. パターンマップ

3. ゲームデザインパターン

3.1. クライマックスパターン

Context

観客が試合を観戦する際，同じゲームを題材とした試合を何度も観戦する．特に，会場放映する場合，観客は自分のペースで試合を観戦できないため，自分のペースで観戦するよりも退屈しやすい傾向にある．

Problem

試合の流れが単調であると，どの試合も同じような内容に見えてしまい，観客が退屈する．

Forces

- 試合の流れが単調ではなく，観客が退屈しない．
- 会場放映する場合，観客が自分のペースで試合を観戦できなくて退屈しやすい．

Solution

試合の終了間際にルールを変化させたり，徐々に難易度を上げたり，外観や音楽を変化させたりする．

Consequences

試合が観客に予想の付かない劇的な内容になり，観客が各試合を退屈せずに楽しめる．一方，あまりに過度なルール変更を行うと，それまでの試合の過程の放映を無意味だと感じて，かえって退屈だと感じるため，適度な変更に留めた方が良い．

Our Uses

早稲田楽天プログラミングコンテスト 2010 では，プレイヤー同士で得点を奪い合うゼロサムゲームであった．試合終了間際に奪える得点の量が倍増する仕組みを導入して，会場を盛り上げることに成功した．

Known Uses

Robert らが作成したテトリス [10] の多くの改良版では，ゲームのタイムアップが近づくにつれ，ゲームの難易度が徐々に上昇する．

Related Patterns

- 仕切り直しパターン：観客を退屈させないという目的が同じである点に関連がある．
- 不可逆フィールドパターン：不可逆フィールドパターンを利用することで，本パターンで上げた問題を解決できる点に関連がある．
- 戦略と戦術スパターン：観客を退屈させないという目的が同じである点に関連がある．
- 予選と本戦パターン：本戦を会場で開催することによって，観客が自分のペースで試合を観戦できなくなる点に関連がある．

3.2. 擬似リアルタイムパターン

Context

題材となるゲームの開発期間に制約がある．また，AI コンテストの参加者は学生から社会人まで幅広い．さらに，観客はゲームのルールを熟知していない．

Problem

リアルタイムなゲームでは，プレイヤーが任意のタイミングで選択した手の結果を反映すため，ゲームと AI **プログラム**の両方が複雑になる．そのため，ゲームの開発コストが増大し品質低下に繋がる上，AI **プログラム**の開発コストが大きすぎて，参加者が減少する．一方で，カードゲームのようなターン制のゲームでは，ルールが分からないと観客が楽しめない場合が多い．

Forces

- 低コストで高品質なゲームを開発したい．
- 参加者が AI **プログラム**の開発に割り当てられる時間が少ない可能性がある．
- 試合の様子がグラフィカルであり，ルールを熟知していない観客も楽しめる．

Solution

ターン制を採用してプレイヤーが選択した手の結果を反映させるタイミングに制約を加える．また，ターンという単位時間のもとに擬似リアルタイムを実現する．

Consequences

ターン制を採用することでゲームと AI **プログラム**の複雑さを低減する．一方で，将棋や囲碁のように試合の様子に動きが少ない場合，ルールが分からないと顧客が試合を楽しめない．そこで，擬似リアルタイムを実現することで，試合の様子がグラフィカルで観客が楽しめるゲームとなる．なお，1 ターンの単位時間を長くし過ぎると観客からリアルタイムに見えなくなるため，1 ターンの単位時間を短くするための工夫が必要である．

Our Uses

JavaChallenge2009，2010 国内予選と早稲田楽天プログラミングコンテスト 2010 では，ターン制による擬似リアルタイムを採用して，ゲームと AI **プログラム**の複雑さを低減した．また，試合の様子をグラフィカルに表現して，観客を楽しめることに成功した．

Known Uses

マイクロプロース社のシヴィライゼーション[11] では，ターン制による擬似リアルタイムを採用したストラテジゲームである．

Related Patterns

- 仕切り直しパターン：ターン制の採用を前提としている点に関連がある．

3.3. 仕切り直しパターン

Context

ターン制を採用していて各プレイヤーが交互に手を決める．また，複数のプレイヤーが同じ盤面を共有していて，盤面が同じ状態を繰り返す可能性がある．

Problem

各プレイヤーが同じような手を繰り返すことで，ゲームの盤面が同じ状態を繰り返して，試合の進行が止まり，観客が退屈に感じる．

Forces

- ターン制を採用したい
- 盤面が同じ状態を繰り返して試合の進行が止まることを防ぎたい

Solution

盤面が同じ状態を繰り返した際に，強制的に盤面の状態を変化させたり，同じ手を打てなくさせたりして，試合を仕切り直しさせる．

Consequences

試合を仕切り直すことで，盤面が同じ状態になるのを防ぎ，観客が退屈しない試合にする．

Our Uses

なし．我々が開催した 3 つの AI コンテストでは何度か仕切り直しが必要な試合があったため，本パターンの適用が有効である．

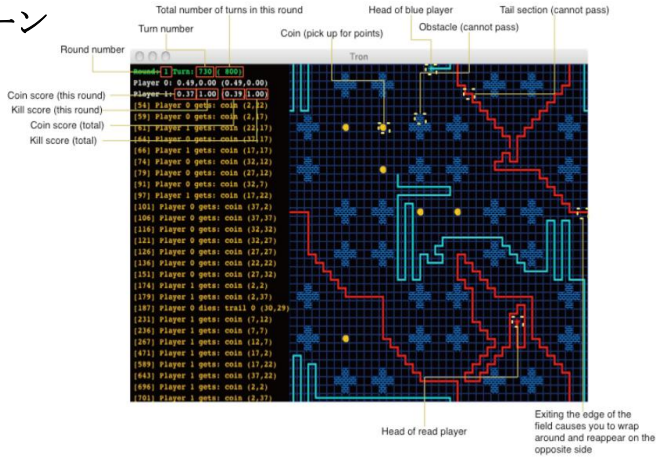
Known Uses

将棋における千日手や，チェスにおけるスリーフォールド・レピティション．

Related Patterns

- クライマックスパターン：観客を退屈させないという目的が同じである点に関連がある．
- 擬似リアルタイムパターン：ターン制の採用を前提としている点に関連がある．
- 戦略と戦術スパターン：観客を退屈させないという目的が同じである点に関連がある．

3.4. 縮退盤面パターン



Context

クライマックスパターンと仕切り直しパターンの両方の Context を持つ.

Problem

クライマックスパターンと仕切り直しパターンの両方の Problem を抱えるが，別々にパターンを適用すると手間がかかる.

Forces

- クライマックスパターンと同じ問題を抱える
- 仕切り直しパターンと同じ問題を抱える
- パターンを適用する手間を減らしたい

Solution

盤面を徐々に縮退させ，プレイヤーの行動範囲を制限する.

Consequences

盤面が徐々に縮退することで，同じ盤面の状況が繰り返されなくなる．そのため，仕切り直しパターンが抱える問題を解決する．また，プレイヤーの行動範囲を制限することで，試合に緊迫感を与えて，観客を退屈させない.

Our Uses

なし．我々は運営側ではなく一般参加者として参加した JavaChallenge2010 アジア地区予選 [12] では，**二次元上のマップをキャラクタが移動するゲームを題材とした．キャラクタの軌跡は記憶され，軌跡に触れたキャラクタは敗北する．そのため，フィールドは不可逆に変化して仕切り直しパターンの適用が不要となる．その上，移動可能な範囲が減っていくため，ゲームが進行するにつれ試合の緊迫感が増して，クライマックスパターンの適用が不要となる.**

Known Uses

ハドソン社のボンバーマン [13] では，タイムアップが近づくと盤面が徐々に縮退する.

Related Patterns

- クライマックスパターン：同じ Context と Problem を持つ点に関連がある.
- 仕切り直しパターン：同じ Context と Problem を持つ点に関連がある.

3.5. 戦略と戦術パターン

Context

参加者は AI プログラムを作成する際に様々な戦略を立て，観客も観戦する際にその戦略を理解して試合を楽しむ．また，戦略の種類には限りがあり，同じ戦略を取るプレイヤー同士で戦う場合がある．観客は同じ戦略を取るプレイヤーであっても強弱が存在して，どちらかが勝利することを楽しむ．なお，戦略とは大局的な戦い方，戦術とは局所的な戦い方を示す．

Problem

観客にプレイヤーが選択した戦略の差異が分からないと観客にとって面白味に欠ける．また，同じ戦略を取るプレイヤー同士でも強弱の差異がつかないと試合の内容が短調になり，参加者にとって面白味に欠ける．

Forces

- 観客が観戦する際にプレイヤーが取る戦略の差異を理解できるようにしたい
- 同じ戦略を取るプレイヤー同士に強弱が存在してほしい

Solution

戦略と戦術の両方で幅を持たせる．各戦略はそれぞれ大きく差があり，また，同じ戦略であってもどのようにその戦略を実現するかという点で戦術に幅がある．戦術の差が観客に分からなくとも，その強弱が試合結果に反映されるようにする．

Consequences

戦略に幅を持たせることで，戦略の差異を際立たせて，観客が理解できるようになる．また，戦術に幅を持たせることで，戦術に強弱を与えて，同じ戦略であっても強弱が付くようにする．これにより，ゲームに深みを与える．一方で，あまりに戦略と戦術の幅を持たせるとゲームの複雑度が増し，AI プログラムの開発コストが増大して，参加者の敷居が高くなってしまう．

Our Uses

早稲田楽天プログラミングコンテスト 2010 では，プレイヤーの戦略として大きく守りと攻めの二種類の戦略が存在した．また，一言で守りといっても，様々な守り方が存在して，戦術の幅から，同じ戦略を取るプレイヤーにも強弱が存在した．

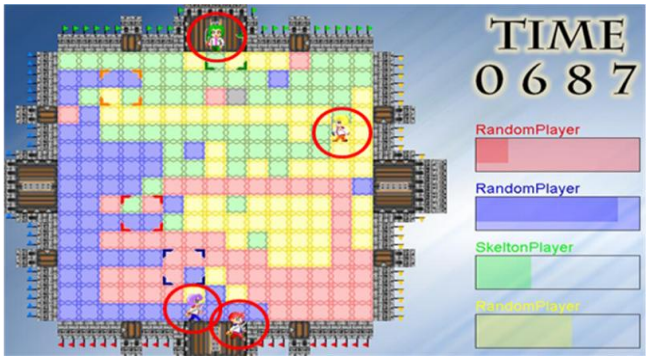
Known Uses

サッカーでは攻めと守りの戦略が存在するが，戦術においてパス回しで陽動して攻める戦術，ドリブルの上手い選手にボールを集めて攻める戦術が存在する．

Related Patterns

- 仕切り直しパターン：観客を退屈させないという目的が同じである点に関連がある．
- クライマックスパターン：観客を退屈させないという目的が同じである点に関連がある．
- 3 プレイヤー以上対戦パターン：戦略に幅を持たせるという同じ目的がある点に関連がある．
- 三すくみパターン：戦略に幅を持たせるという同じ目的がある点に関連がある．

3.6. 3 プレイヤー以上対戦パターン



Context

参加者と観客が楽しめるような戦略に幅のあるゲームを開発する．また，2 プレイヤー対戦のゲームのアイデアが出ているが，戦略に幅のあるゲームのアイデアが思い付いていない．

Problem

戦略に幅のあるゲームは参加者と観客に好まれるが，そのようなゲームの考案は難しい．

Forces

- 戦略に幅を持たせたい
- ゲームの考案に時間をかけられない
- 2 プレイヤー対戦のゲームのアイデアが出ている

Solution

3 プレイヤー以上が対戦するゲームを考案する．2 プレイヤー対戦のゲームをそのまま 3 プレイヤー以上に拡張できないか考える．

Consequences

3 プレイヤー以上が対戦するゲームでは，各プレイヤーが 2 プレイヤー以上の手を想定しながら戦うため，試合の大局を見据えた戦略が自然に生まれる．一方で，取りうる手と局面が複雑になるため，全ての局面を探索して手を考えるようなアルゴリズムを AI プログラムで採用することが難しくなる．

Our Uses

早稲田楽天プログラミングコンテスト 2010 では，4 人対戦のゼロサムゲームであった．そのため，ある時点の 1 位のプレイヤーは他の 3 プレイヤーから狙われることになり，敢えて試合の途中で極力 1 位にならないような戦略や他のプレイヤーに狙われているプレイヤーを狙うといった戦略が自然と生まれた．

Known Uses

トランプゲームのハーツ，麻雀は 4 人で対戦するゲームである．

Related Patterns

- 戦略と戦術パターン：戦略に幅を持たせるという同じ目的がある点に関連がある．
- 三すくみパターン：戦略に幅を持たせるという同じ目的がある点に関連がある．

3.7. 三すくみパターン

Context

参加者と観客が楽しめるような戦略に幅のあるゲームを開発する．また，3 プレイヤー以上対戦ゲームの良いアイデアがない．

Problem

戦略に幅のあるゲームは参加者と観客に好まれるが，そのようなゲームの考案は難しい．

Forces

- 戦略に幅を持たせたい
- ゲームの考案に時間をかけられない
- 3 プレイヤー以上対戦の良いゲームアイデアがない

Solution

三すくみのように有利不利がある戦略を取れるゲームを考案する．なお，すくみの数はいくつでも良い．

Consequences

相手の戦略を予想して自分の戦略を立てるといったことが可能になり，ゲームに深みが増す．一方で，際立って強い戦略が存在すると，戦略を選択する面白みがなくなる．そのため，各戦略に同じように有利不利が成り立つルールを考案する難しさがある．

Our Uses

JavaChallenge2010 国内予選では，大きな得点を狙う，小さな得点を狙う，相手を妨害するという 3 つの戦略で三すくみを採用して，ゲームに面白さを与えた．しかし，ゲームのルール調整に失敗して，実際は小さな得点を狙う戦略が他の戦略よりも強いという結果になってしまった．

Known Uses

じゃんけんでは，それぞれ勝ち負け引き分けの決まった 3 つの手を出し合って対戦する．

Related Patterns

- 戦略と戦術パターン：戦略に幅を持たせるという同じ目的がある点に関連がある．
- 3 プレイヤー以上対戦パターン：戦略に幅を持たせるという同じ目的がある点に関連がある．

4. 運営パターン

4.1. 予選と本戦パターン

Context

運営者はできる限り多くの参加者を望む．会場で試合を放映した方が，会場に集まった人同士で一体感が生まれ，試合が盛り上がる．しかし，会場の選択に幅がなく収容人数や試合の放映時間に制約がある．また，事前に AI コンテストの参加者数を予想できない．

Problem

参加者数を想定できない上に，会場の収容人数や試合の放映時間に制約があり，参加者が少なすぎたり多すぎたりした場合に対処できない．

Forces

- 会場で試合を放映したい
- 参加者数を予想できない
- 会場の収容人数に制約がある
- 試合の放映時間に制約がある

Solution

予選と本戦を設けて，収容人数の制約を満たすように本戦に参加できる人数を固定する．また，予選によって得られた情報を用いて，本戦の会場設営の参考にする．

Consequences

予選を開催することで参加者数を確定して，そこで得た経験を元に人数を固定した本戦を開催できる．予選終了後から本戦開催までの期間を利用して会場設営が可能である．

Our Uses

早稲田楽天プログラミングコンテスト 2010 では，インターネット上で予選を行った上で，楽天テクノロジーカンファレンス 2010 にて本戦を行った．本戦参加者数を固定することができ，予選で得た情報と経験を利用した迅速な会場設営を実現した．また，会場で放映したため，参加者と観客の両方を盛り上げることに成功した．

Known Uses

ICPC では，国内予選がインターネット上で開催され，アジア地区予選が会場で開催された．

Related Patterns

- クライマックスパターン：本戦を会場で開催することによって，観客が自分のペースで試合を観戦できなくなる点に関連がある．
- フィードバック期間パターン：予選の開催を前提条件としている点に関連がある．

4.2. フィードバック期間パターン

Context

予選が開催された後に本戦が開催される．参加者と観客が予選と本戦の両方を観戦することを想定する．参加者と観客は，安易に推測できない試合に参加もしくは観戦することを楽しむ．

Problem

予選結果から本戦の結果が想定できると，参加者にとっても観客にとっても本戦の魅力が失われる．

Forces

- 予選開催後に本戦まである程度の期間を取れる
- 予選から本戦の結果が想定できず予選と本戦の両方を楽しめる
- 参加者は予選開催後も継続して AI プログラムを開発可能である

Solution

予選開催後から本戦まで参加者に AI プログラムを改良する期間を与える．

Consequences

参加者は予選結果を踏まえた改良を楽しむことができる．また，対戦する AI プログラムが異なるため，観客は予選結果から本戦を想定せずに楽しめる．

Our Uses

なし．早稲田楽天プログラミングコンテスト 2010 では，予選開催後に改良期間を設けなかったため，多くの参加者から改良期間をアンケートにて求められた．

Known Uses

組込みシステム技術協会が主催する ET ロボコン [14] では，予選通過後に本戦まで改良期間が与えられる．

Related Patterns

- 予選と本戦パターン：予選の開催を前提条件としている点に関連がある．

4.3. API 固定パターン

Context

参加者が AI プログラムを作成する期間が比較的長く与えられており，開発者がゲームとその API を改良する余裕がある．

Problem

開発者は参加者の AI プログラム作成期間中に API を改良する余裕があるが，API を変更することで参加者の AI プログラムが正常に動作しなくなったり，余計な API の学習コストを要したりする可能性がある．

Forces

- 参加者の AI プログラムの開発期間が比較的長い
- 開発者は API を改良したい

Solution

API は致命的なバグ対処以外では決して変更しない．できる限りドキュメント修正や告知等で対処する．

Consequences

API の変更をしなかったため，参加者に付加的な負担を与えない．一方で，開発者は API を改良したいという欲求を抑える必要がある．

Our Uses

JavaChallenge2009，2010 では，API を固定したため，参加者に付加的な負担を与えなかった．一方，早稲田楽天プログラミングコンテストでは，API の追加や修正を何度か行ったために，参加者を混乱させる結果となった．

Known Uses

情報処理学会が主催した Cell スピードチャレンジ 2007 [15] では，API が固定された状態でコンテストが開催された．

Related Patterns

- フィードバック期間パターン：フィードバック期間の直前であれば API を修正した際のユーザへの負担が軽減されると思われる点に関連がある．

5. まとめと今後の展望

我々は，3 回の AI コンテスト開催経験と 3 種類のゲームの開発経験から，コンピュータプレイヤー同士の対戦を通したプログラミングコンテストのパターンランゲージを提案した．また，題材となるゲームのゲームデザインパターンと AI コンテストの運営パターンの 2 種類のパターンへ整理した．

今後も AI コンテストの開催を通してこれらのパターンランゲージの有用性を評価して，さらに，より多くのパターンを追加して，全ての運営者と開発者に役立つパターンランゲージへと進化させたい．ゲームを開発するに当たって，アーキテクチャパターン，デザインパターン，イディオム，開発プセスのパターンも存在しており，今後それらのパターンも取り上げられることが期待される．

謝辞　本研究の一部は早稲田大学グローバル COE プログラムによった．

参考文献

[1] ACM: The ACM-ICPC International Collegiate Programming Contest Web Site sponsored by IBM, http://cm.baylor.edu/welcome.icpc, obtained on 6 Feb 2011.
[2] 情報処理学会: 情報処理学会-コンピュータ将棋プロジェクト, http://www.ipsj.or.jp/50anv/shogi/index2.html, obtained on 6 Feb 2011.
[3] IBM: Robocode Home, http://robocode.sourceforge.net/, obtained on 6 Feb 2011.
[4] IBM: alphaWorks : CodeRally : Overview, http://www.alphaworks.ibm.com/tech/coderally, obtained on 6 Feb 2011.
[5] ACM 日本支部: ACM/ICPC Asia Regional Contest 2009, Tokyo, Japan, http://www.waseda.jp/assoc-icpc2009/jp/, obtained on 6 Feb 2011.
[6] 坂本一憲, 大橋昭, 志水理哉, 高橋周平, 村上真一: Java チャレンジ 2010 Summer, http://www.washi.cs.waseda.ac.jp/javachallenge2010summer/, obtained on 6 Feb 2011.
[7] 楽天株式会社: 楽天テクノロジーカンファレンス 2010, http://tech.rakuten.co.jp/rtc2010/, obtained on 6 Feb 2011.
[8] 坂本一憲, 大橋昭, 志水理哉, 高橋周平, 村上真一: プログラミングコンテスト　～最強の AI を作ろう！～, http://www.washi.cs.waseda.ac.jp/rakuten-waseda-pc2010/, obtained on 6 Feb 2011.
[9] 久保淳人: ソフトウェアパターン・リポジトリの構築に向けたデータモデルの検討, ウィンターワークショップ 2010・イン・倉敷, Jan 2010.
[10] At 25, Tetris still eyeing growth, http://www.reuters.com/article/2009/06/02/us-videogames-tetris-idUSTRE5510V020090602, obtained on 6 Feb 2011.
[11] 株式会社サイバーフロント: シヴィライゼーション４【完全日本語版】, http://www.cyberfront.co.jp/title/civ4/, obtained on 6 Feb 2011.
[12] 国立情報学研究所: Java チャレンジ | ACM-ICPC: ACM International Collegiate Programming Contest Asia Regional Contest 2010 in Tokyo, view-source:http://icpc2010.honiden.nii.ac.jp/regional-contest/java-challenge, obtained on 6 Feb 2011.
[13] 株式会社ハドソン: ボンバーマン 公式サイト | Bomberman Official Site, http://bomberman.jp/, obtained on 6 Feb 2011.

[14] ET ロボコン実行委員会: ET ロボコン 2011 公式サイト□"組込みソフトウェア技術教育をテーマとした「ET ロボコン」開催のご案内, http://www.etrobo.jp/2011/, obtained on 6 Feb 2011.
[15] Cell スピードチャレンジ 2007 実行委員会: main, http://www.hpcc.jp/sacsis/2007/cell-challenge/, obtained on 6 Feb 2011.
[16] 坂本一憲, 中村悠人, 庄山昭彦, 内山諭, 城間祐輝, 野本悠太郎: javachallenge2009 - Project Hosting on Google Code, http://code.google.com/p/javachallenge2009/.
[17] 坂本一憲, 大橋昭, 志水理哉, 高橋周平, 村上真一: javachallenge2010 - Project Hosting on Google Code, http://code.google.com/p/javachallenge2010/.
[18] 坂本一憲, 大橋昭, 志水理哉, 高橋周平, 村上真一: quinte - Project Hosting on Google Code, http://code.google.com/p/quinte/.

付録
2009 年度 JavaChallenge アジア地区予選

2009 年度に開催された ICPC のアジア地区予選内の併設コンテストである JavaChallenge のゲームを開発して運用した．1 チーム 3 人の構成で 35 チームが参加した．1 チームに 1 台のコンピュータが与えられ，1 時間半の期間で Java 言語で AI プログラムを作成して，ゲーム上で対戦させて競い合う．書籍などの紙媒体のものは持ち込めるが，キーボードやディスプレイ等を含めた電子機器の持ち込みやインターネットの使用は許されない．開発環境は一般的なテキストエディタに加えて Eclipse のみ利用可能である．

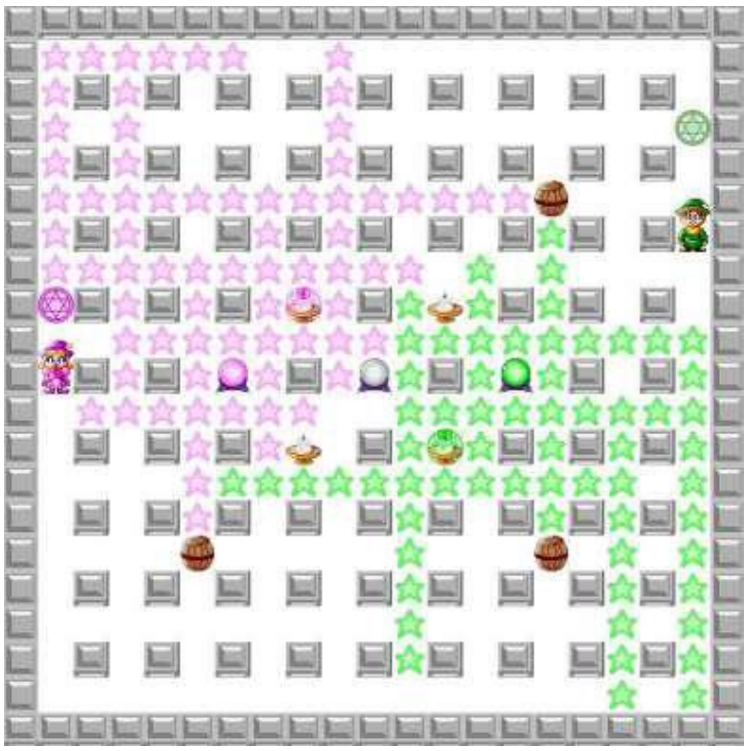


図 2. 2009 年度 JavaChallenge のゲーム画面

題材となったゲームは，2 プレイヤーで対戦する二次元座標上の陣取りゲームである．プレイヤーはキャラクタを移動させながら，一定時間が経過すると発動する魔法陣を設置させることで戦う．魔法陣が発動すると，一定範囲内のマスの所有権を得られて，図 2 のように色づけされる．また，範囲内にいるキャラクタにペナルティを与える．マスにはいくつかの種類があり，何もないマス以外にロウソクと水晶玉があり，これらは周囲 8 マスの所有権を得ることによって所有できる．何もないマスを所有することで 1 点が，ロウソクを所有することで 20 点が，水晶玉を所有することで魔法が発動した際の効果範囲が 2 マス広がる．なお，水晶玉は一度所有すると所有権は不変だが，それ以外は所有権が変わる．ゲーム時間が決まっており，終了時に最も得点の高いプレイヤーの勝利となる．ゲームのソースコードは Google Code にてオープンソースとして公開している [16].

2010 年度 JavaChallenge 国内予選

2010 年度に開催された ICPC の国内予選内の併設コンテストである JavaChallenge のゲームソフトウェアを開発して運用した．1 チーム 3 人の構成で 45 チームが参加した．1 チームに 1 台のコンピュータが与えられ，4 時間の期間で Java 言語で AI プログラムを作成して，ゲーム上で対戦させて競い合う．インターネットを利用した情報収集は可能だが，キーボードやディスプレイ等を含めた電子機器の持ち込みは許されない．開発環境の制限はない．

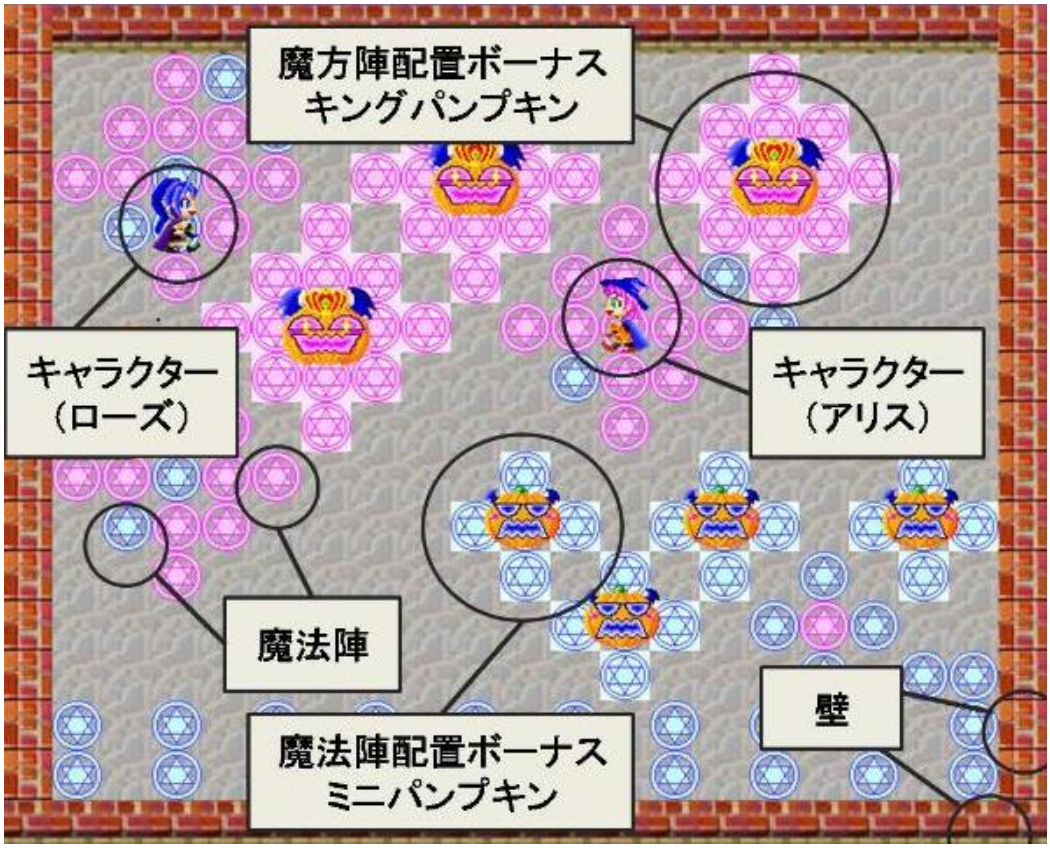


図 3. 2010 年度 JavaChallenge のゲーム画面

題材となったゲームは，2 プレイヤーで対戦する大小二種類のひし形の陣地を形成することに重きを置いた二次元座標上の陣取りゲームである．プレイヤーはキャラクタを移動させながら，キャラクタが位置するマスに魔方陣を配置させることで所有権を得る．自分が所有するマスは図 3 のように色づけされる．所有するマスにより大小二種類のひし形を形成することで，図 3 のように所有するマスのグラフィックが変化する．これを魔方陣配置ボーナスと呼び，小さいひし形 1 つにつき 20 点，大きいひし形 1 つにつき 50 点，それ以外の所有するマス 1 つにつき 1 点を得ることができる．また，必要なターン数が多くなるものの，相手の所有するマスを奪うこともできる．特に，相手の魔方陣配置ボーナスの図形を壊すたびに 2 点を得られる．ゲーム時間が決まっており，終了時に最も得点の高いプレイヤーの勝利となる．ゲームのソースコードは GoogleCode にてオープンソースとして公開している [17].

2010 年度 早稲田楽天プログラミングコンテスト

2010 年度に開催された楽天テクノロジーカンファレンスの 1 セッションである早稲田楽天プログラミングコンテストのゲームを開発して運用した．1 チーム 3 人の構成で 19 チームが参加した．参加者は延長も含めた 17 日間の期間で Java 言語，Ruby 言語，Python 言語，JavaScript 言語，Scala 言語のいずれかで AI プログラムを作成する．コンテストは，AI プログラム同士で対戦させることで，ユーザー同士が競い合う．なお，リフレクションやリバースエンジニアリング行為は禁止されるものの，それ以外の制約は一切ない．

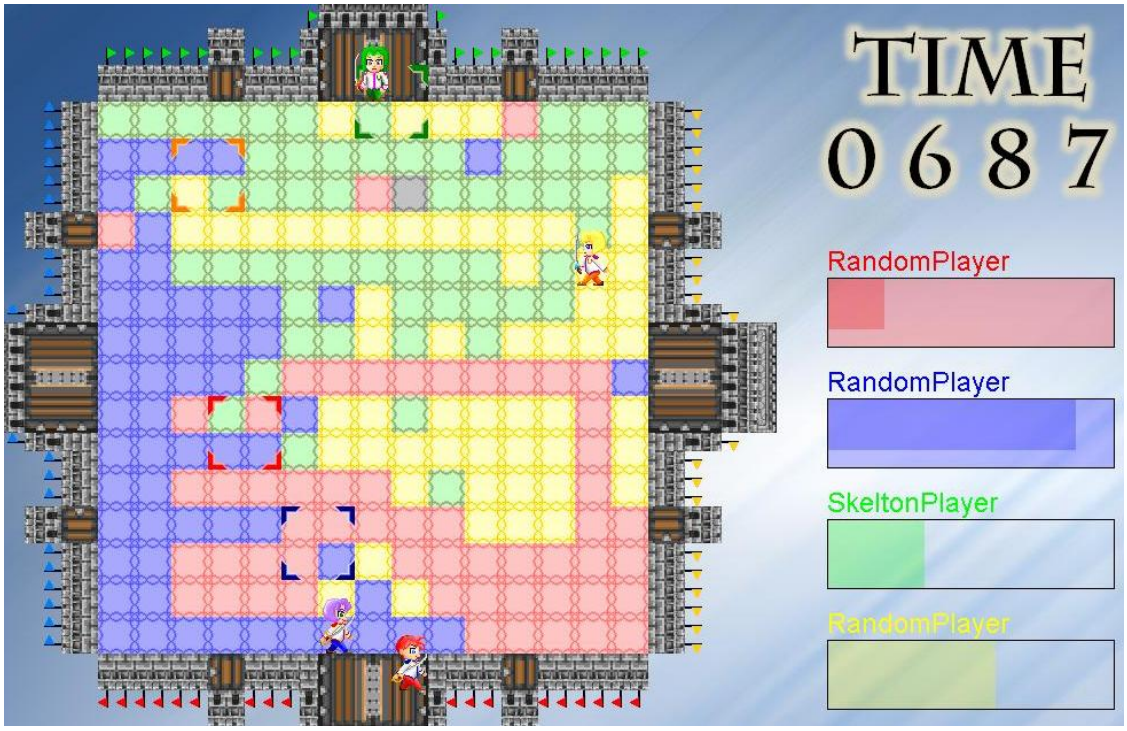


図 4. 早稲田楽天プログラミングコンテストのゲーム画面

題材となったゲームは，図 4 のように 4 プレイヤーで対戦するカーソルと兵士の 2 つのキャラクタを動かす二次元座標上のパズル感覚のゲームである．カーソルは任意の場所に移動させることができ，2 × 2 マス内の色を時計回りもしくは反時計回りに回転できる．一方，兵士は割り当てられた自分の色と同じ色のマスに 1 歩ずつ移動できる．兵士を自分以外のプレイヤーの城の門に移動させることで，該当プレイヤーの点数を奪うことができる．つまり，カーソルによってマスの色を変化させて，兵士が移動できる道を作成して，兵士を相手プレイヤーの門まで移動させて戦うゲームである．大きな門に移動させた場合は 20%，小さな門の場合は 10% 奪うことができる．ゲーム時間が決まっており，終了時に最も得点の高いプレイヤーの勝利となる．なお，ゲーム終了間際になると兵士のグラフィックが変化して，奪う得点の割合が 2 倍になる．ゲームのソースコードは Google Code にてオープンソースとして公開している [18].