RITSUMEIKAN

いずみ研的 Geister C Library

Rgeisterlib

http://www.ritsumei.ac.jp/se/re/izumilab/dist/Rgeister/

立命館大学 理工学部 電子情報工学科 いずみ研

Geister はドイツ語でオバケを意味する。英語圏では Ghost や Phantom という名前になっているらしい。

これを利用される方へ

- 細かいことを気にせず公開しています。学術研究の良識に従ってご活用下さい。
- もしもお役に立ちましたら、以下を参考文献に挙げて頂けましたら幸いです。

Rgeister*

> “Rgeister Project”, http://www.ritsumei.ac.jp/se/re/izumilab/dist/Rgeister/

SonodaGameAI*

> 園田夕莉, 泉知論,「モンテカルロ法に基づく駒属性推定によるガイスターゲームAI」, 第4回 Game AI Tournament (GAT2019), ポスター発表 P-1 , Mar. 2019.

NakashimaGameAI*

> 中島拓弥, 泉知論,「ガイスターにおける不完全情報ゲーム先読み手法の検討」, 第5回 Game AI Tournament (GAT2020), ポスター発表 P-1 , Mar. 2020.

概要

C言語で Geister Player を開発するためのライブラリとサンプルコード

- 駒や盤に関する各種定義
- ゲーム進行を扱う各種関数
- 審判との通信を扱う各種関数
- 乱数によるプレイヤのサンプルコード
- main関数のサンプルコード
- Makefileのサンプルコード

必要な環境

- Windows
- cygwin のgcc環境
  - https://www.cygwin.com/
  - 要 gcc, make
- JAVA SE Development Kit 8 ※例えば jdk-8u251-windows-x64
  - https://www.oracle.com/technetwork/java/javase/downloads/
- わさらぼ三好さんによる geister 審判 server
  - https://github.com/miyo/geister_server.java/
- 松江高専Geister大会ルール
  - http://www2.matsue-ct.ac.jp/home/hashimoto/geister/GAT/

推奨フォルダ構成

- 技術系ツールはいわゆる半角英数以外の文字が混じっていたり、フォルダ階層が深かったりするとトラブルになることがあるので、避ける。
- C:¥home¥[RAINBOW ID]¥Geister¥ というフォルダを作成してその下にGeister関連のファイルやフォルダを作成することを推奨する。
- 例えば、RAINBOW IDが ri0123xy なら…
  - C:¥home¥ri0123xy¥Geister¥Doc¥　資料など
  - C:¥home¥ri0123xy¥Geister¥Host¥　審判
  - C:¥home¥ri0123xy¥Geister¥Player¥　自作プレイヤ

試用(1)～審判キット

三好さん審判キットの動作確認と修得

- インストールと実行方法
  - キットの資料参照 (README.md)
- １台のPCで実行
  - PC上で審判を実行、先攻後攻ともに乱数プレイヤを実行
- ２台のPCで実行
  - 審判用PCで審判および表示(viewer.html)を実行、乱数プレイヤも実行
  - プレイヤ用PCで乱数プレイヤや人間プレイヤを実行
  - 先攻・後攻を入れ替えて何度か試行すること

試用(2)～いずみ研的サンプル

- インストール
  - プレイヤ用フォルダを作成し、ファイルを展開する
  - cygwin terminal 上でそのフォルダに移動し make を実行
- 実行方法

  ./Rgeister.exe〔ホストのIPアドレス〕〔ポート番号〕〔1ターン毎の停止時間〕

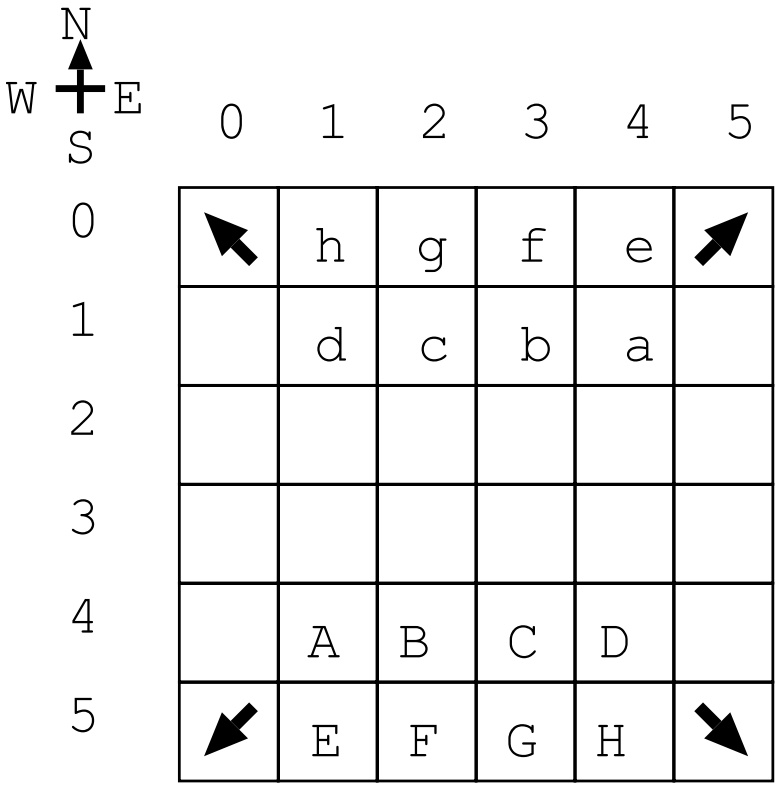
  例 `./Rgeister.exe localhost 10000 0`

  例 `./Rgeister.exe localhost 192.168.102.99 10001 1`
- ２台のPCで実行
  - 審判用PCで審判および表示 (viewer.html) を実行、乱数プレイヤも実行
  - プレイヤ用PCでいずみ研的乱数プレイヤサンプルを実行
  - 先攻・後攻を入れ替えて何度か試行すること

Rgeisterlib ファイル構成

- geister.c, geister.h
  - main関数を含むガイスタープログラムの本体
  - これを改造する
- geisterlib.c, geisterlib.h
  - 駒や盤面の定義、ゲーム進行用の基本的な関数群
  - 原則、これは改造しない
- geisterclient.c, geisterclient.h
  - 審判との通信用の関数群
  - 原則、これは改造しない
- Makefile
  - プログラムをコンパイル（ビルド）するための設定ファイル

盤面と駒の表現



- 座標系は x=0…5, y=0…5
- 方向は (0,0) を北西角とする
- 自駒は南側に配置し北側の角を脱出口とする
- 敵駒は北側に配置し南側の角を脱出口とする
- 自駒の0～7番を文字A～Hで表す
- 敵駒の0～7番を文字a～hで表す
- 初期配置は図の通り、敵に向かって前衛左から右に順に並べる

geister.h

geister.c の定義ファイル

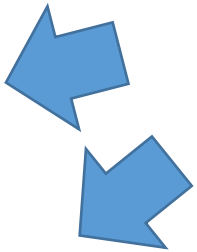
- 関数プロトタイプ宣言

geister.c

RITSUMEIKAN

main関数とプレイヤ本体

- GstSetGhost(ゲーム状態 s) : 成否
  - sの盤上に駒の初期配置をする
- GstPlay(ゲーム状態 s, 指し手a) : 成否
  - ゲーム状態sに対する指し手aを返す
- main()
  - 起動時引数の処理
  - ゲームの初期化、進行、終了
- GstDie(トラブル報告メッセージ msg)
  - トラブル時に安全にプログラムを終了する
  - メッセージmsgを標準エラー出力に表示する



主に
このへんを
改造する

geisterlib.h (1)

Geister 用基本ライブラリ等の定義ファイル

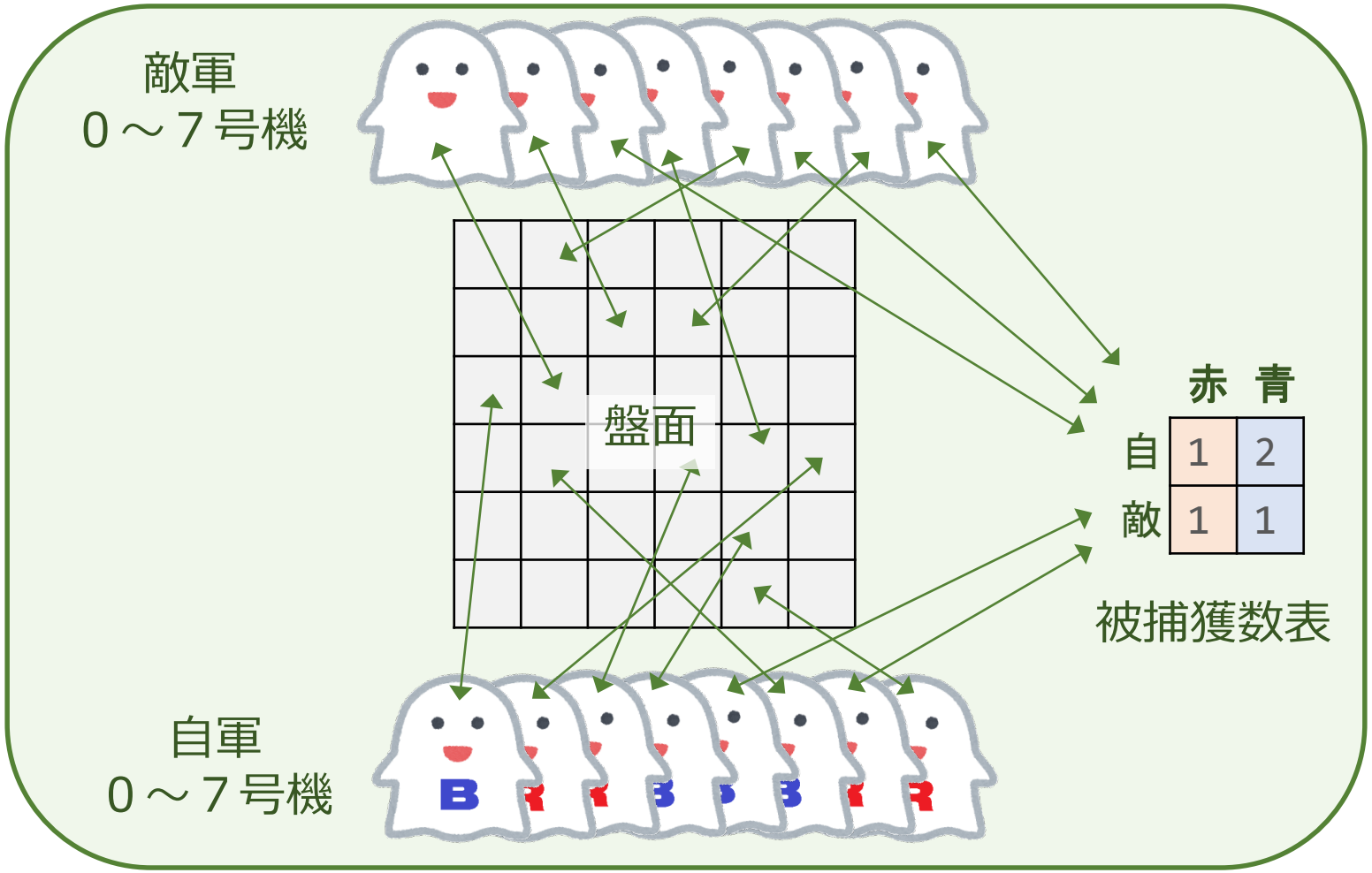
- 定数
  - プレイヤの種類 … 自身(Ply0)、敵(Ply1)、なし・不在など(None)、反対側
  - 色の種類 … 赤、青、不明
  - 駒の状態 … 盤上、捕獲済、脱出済
  - 盤の大きさ … 幅、高さ
  - 盤上の方向 … 北、東、南、西
  - ゲームの進行状態 … 勝ち、負け、続行
  - 真理値 … 真、偽

geisterlib.h (2)

- 構造体
  - 駒の構造体 … プレイヤ種類、駒番号、色、駒の状態、ボード上の座標
  - ゲーム状態の構造体 … 駒構造体の配列（２人×８個）、盤面の配列（各座標の駒の構造体へのポインタ、空ならNULL）、捕獲済駒数（２人×２色）、ターン
  - 指し手の構造体 … 駒の構造体へのポインタ、移動方向
  - プレイヤアルゴリズムで必要なら、改造も可
- 表（定数配列）
  - 方向（北東南西）から座標変位（Δx, Δy）への変換表
  - 方向（北東南西）から反対方向（南西北東）への変換表
- 変数
  - ログファイルのファイルポインタ (gstlog)
- 関数のプロトタイプ宣言

駒構造体

ゲーム状態構造体



geisterlib.c (1)　ログ系

Geister 用基本ライブラリ関数群

- GstLogOpen() : 成否
  - ログファイルを開く
  - 開いたあとプログラムの随所で fprintf(gstlog, “”,….) で情報をログファイルに記録できる
- GstLogClose() : 成否
  - ログファイルを閉じる

geisterlib.c (2)　ゲーム状態初期化・複製

- GstInitGame(ゲーム状態s) : 成否
  - ゲーム状態を初期化する
  - 盤面にコマを後述の通りに配置
  - 駒の色はすべて不明
- GstCopyGame(ゲーム状態 dst, src, プレイヤ反転フラグ r) : 成否
  - ゲーム状態 dst を src と同じ状態にする
  - 反転フラグが0ならそのまま、それ以外ならプレイヤを反転し、盤面を180°回転する

geisterlib.c (3)　表示系

- GstPrintGame(出力先 fp, ゲーム状態 s) : 成否
  - ゲーム状態sを表示する
  - 出力先が stdout なら実行画面に表示する
- GstPrintBoard(出力先 fp, ゲーム状態 s) : 成否
  - ゲーム状態sの盤面のみを表示する
- GstPrintGhostChar(出力先 fp, 駒 g) : 成否
  - 状態表示用に駒 g を表示する

geisterlib.c (4)　ゲーム進行系

- GstApplyAction(ゲーム状態 s, 指し手 a, 実行フラグ exec) : 成否
  - ゲーム状態 s に指し手 a を適用する
  - exec=0 のときには、可否確認のみで適用はしない
- GstCheckAction(ゲーム状態 s, 指し手 a) : 可否
  - 指し手の可否を確認する
  - GstApplyAction(s,a,0) のマクロ
- GstExecAction(ゲーム状態 s, 指し手 a) : 成否
  - GstApplyAction(s,a,1) のマクロ
- GstJudgeGame(ゲーム状態 s) : 勝敗／継続
  - ゲーム状態 s の勝敗または継続を判定する

geisterlib.c (5) その他

- GstCheckGame(ゲーム状態 s) : 合否
  - ゲーム状態 s のデータの整合性を確認する
  - デバグ用、データの不具合・矛盾のチェック
- WeightedRandom(生成率重み列 w[], 個数 n) : 乱数
  - 重み付き乱数生成
  - 0〜n-1 の整数を返す
  - 乱数値 x の生成確率は w[x] / Σ w[i]

geisterclient.h

- 関数のプロトタイプ宣言

geisterclient.c (1)　開始･終了系

三好さん審判サーバとの通信ライブラリ関数群

- GstConnectHost(IPアドレス a, ポート番号 p) : 成否
  - 審判サーバに接続する
  - 接続したソケット番号は static 変数として保持する
- GstCloseHost() : 成否
  - 審判サーバとの接続を終了する

geisterclient.c (2)　通信系

※通信内容はログファイル socklog*.txt に記録される

- GstSendGhostInfo(ゲーム状態 s) : 成否
  - 審判サーバに初期状態 s（赤駒のID）を送る
- GstRecvBoardState(ゲーム状態 s) : 成否
  - 審判サーバからゲーム状態 s を受け取る
- GstSendAction(指し手 a) : 成否
  - 審判サーバに指し手 a を送る

geisterclient.c (3)　内部で使用する関数

- GstSockLogOpen() : 成否
  - 通信ログファイルを開く
  - ファイルポインタはstatic変数として保持する
- GstSockLogClose() : 成否
  - 通信ログファイルを閉じる
- GstRecvLine(受信バッファ buf) : 成否
  - 審判サーバから1行メッセージを受信する
- GstSendLine(送信バッファ buf) : 成否
  - 審判サーバに1行メッセージを送信する

Rgeisterlib によるゲーム進行概要

geister.c: main() を参照のこと

GstConnectHost(addr,port);	審判と接続
GstInitGame(s);	ゲーム状態の初期化
GstSetGhost(s);	初期配置
GstSendGhostInfo(s);	初期配置を審判に送信
while(1){	ゲームのメインループ
r=GstRecvBoardState(s);	盤面状態を審判から受信
if (r!=GstContinue) break;	勝敗確認
GstPlay(s,a);	指し手の計算
GstSendAction(a);	指し手を審判に送信
GstExecAction(s,a);	指し手をゲーム状態に反映
}	
GstCloseHost();	審判との接続を切断

課題 A

- 貪欲捕獲戦略のプレイヤを実装する。
- 捕れる敵駒があれば捕る。
- なければ敵駒に近づく。
- 候補が複数ある場合は乱数で決定する。

課題 B

- 貪欲脱出戦略のプレイヤを実装する。
- 青駒が脱出マスにいれば脱出する。
- いなければ青駒を脱出マスに近づける。
- 候補が複数ある場合は乱数で決定する。
- 候補がない場合は乱数で決定する。

課題 C

- 既存プレイヤに追加できる必勝パタンチェッカを実装する。

C level 1

- 青が脱出マスに居るとき、脱出する。

C level 2

- 青駒と脱出マスと敵駒の位置関係（距離）を考慮した必勝パタンを考察し、実装する。

C level 3

- 青駒脱出の脅威となる敵駒の捕獲を支援してくれる自駒を考慮した必勝パタンを考察し、実装する。

課題 D

- 園田ガイスターに勝て
  - ソースコードは SonodaGameAI20190310b.zip
  - コンパイル方法、実行方法は Rgeister のサンプルと同じ

課題 E

- 中島ガイスターに勝て
  - ソースコードは NakashimaGameAI20200315b.zip
  - コンパイル方法、実行方法は Rgeister のサンプルと同じ

素材集