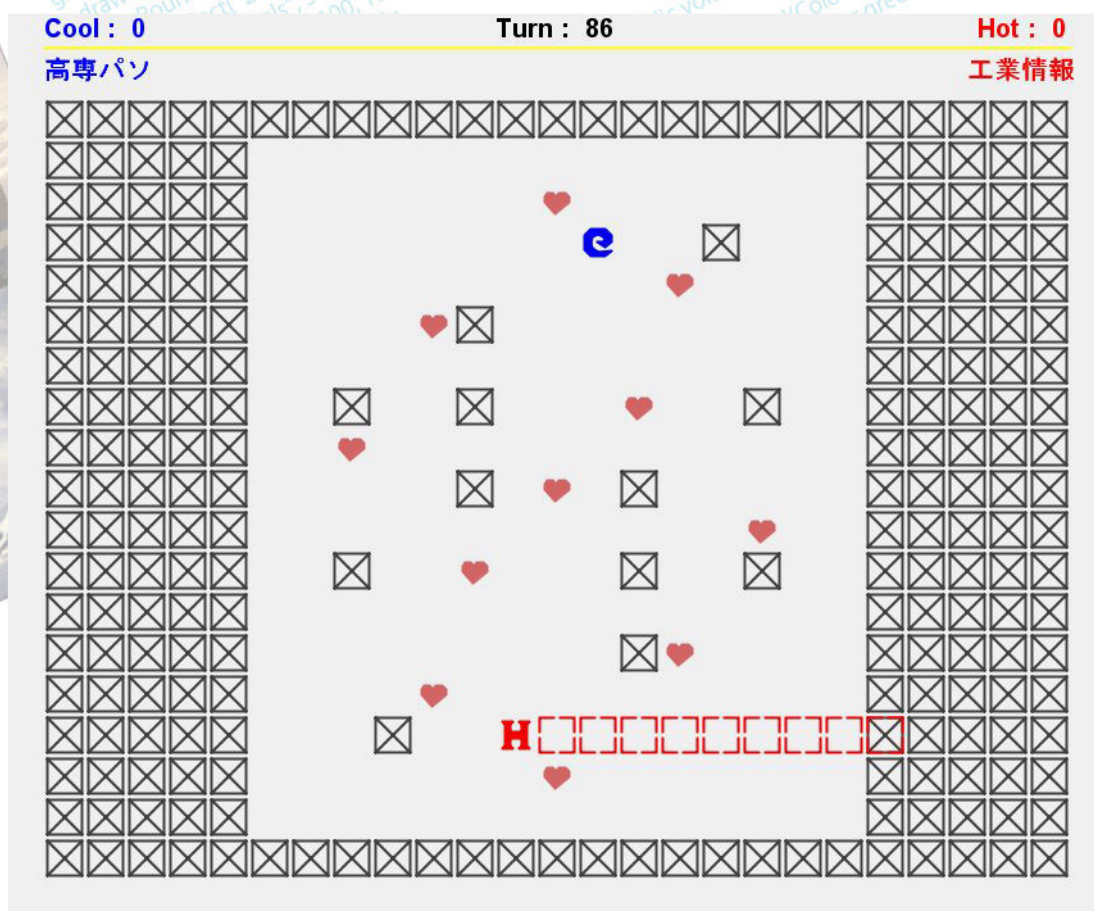


第1回 U-16 旭川プログラミングコンテスト

思考と達成が未来へ!!

競技部門概要 Chaser2011



<競技内容>

競技サーバに接続し、2次元のタテ・ヨコに広がるフィールドで4種類のメソッド（命令）を駆使しながら、相手プログラムと対戦する競技です。

勝利条件





- ・アイテム(♥)を対戦相手より多く集める。
- ・ブロック☒を対戦相手に乗せる。
- ・対戦相手の上下左右をブロックで囲んで動けなくする。





<事前プログラミング講習会>

プログラム未経験者でも講習会に参加すれば誰でも参加できます。





【 競技概要 もくじ 】

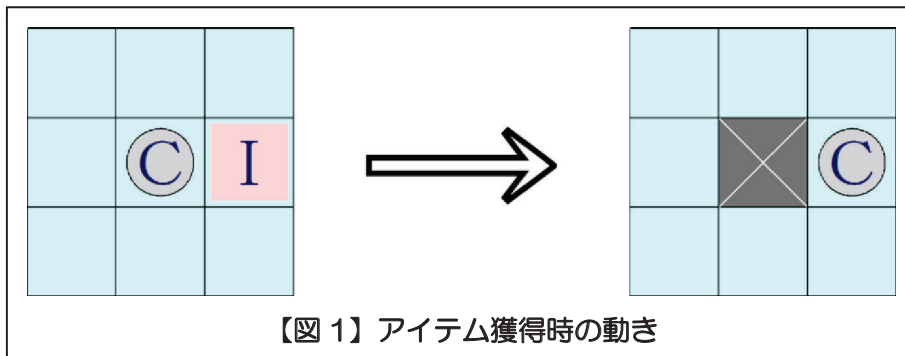
P.1	0.もくじ
P.2	1.基本ルール
P.3	2.メソッド
P.4	3.動作
P.5	4.マップ
	5.フィールド
	6.ターン
	7.ラウンド
	8.その他
P.6	9.周囲情報について
P.8	10.通信の仕様
P.9	通信の流れ図

このテキストで用いる図中の  はクールターゲット、 はホットターゲット、 はアイテム、 はブロックの場所を表します。

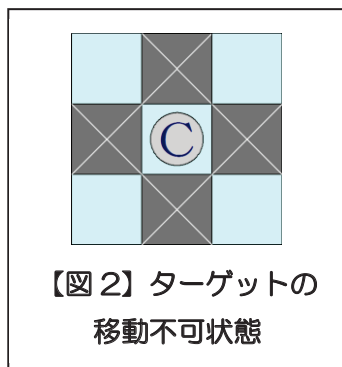
実際の競技では、クールターゲットは  、ホットターゲットは  、アイテムは  、ブロックは  となります。

1. 基本ルール

- ◆ 競技は、競技サーバに接続した 2 つのターゲット(クライアント)が 1 対 1 で対戦する形式で行います。
- ◆ 先にサーバに接続したターゲットを” COOL” () と呼び、先攻となります。
- ◆ 後からサーバに接続したターゲットを” HOT” () と呼び、後攻となります。
- ◆ 両プレイヤーは、フィールド上に配置されたアイテム() を集めながら、敵を追いかけます。
- ◆ 相手の上にブロック を配置すると、勝ちとなります。
- ◆ また、すでに設置されているブロックに自分のターゲットが乗ると、負けとなります。
- ◆ あらかじめ決められた制限ターン以内に勝敗が決まらない場合は、獲得したアイテムの数の多いほうを勝ちとします。
- ◆ アイテムを獲得すると、直前までターゲットのいた位置にはブロックが配置されます。(【図 1 参照】)



- ◆ ターゲットの周りが全てブロックで埋められ、身動きができない状態となると負けとなります。(【図 2】 参照)



どこか 1 か所でも、脱出が可能ならば負けにはなりません。

2. メソッド

- ◆ ターゲットが各ターンに行う動作のことを”メソッド”と呼びます。
- ◆ メソッドは、大きく[walk 系][look 系][search 系][put 系]の 4 つの系統に分けることができます。
- ◆ また、各系統それぞれに[Up][Down][Right][Left]の 4 方向のメソッドが用意されます。(合計では 16 個のメソッドが用意されています)
- ◆ プログラム中では、” [系統][方向]() ” と組み合わせて使います。
例) walkRight() 、 putDown() など

walk 系 < 例 : walkRight() >	
	<ol style="list-style-type: none"> 1. 指定した方向に 1 マス進む 2. 移動先の周囲 9 マス分の情報を取得
look 系 < 例 : lookRight() >	
	<p>実行した場所から、指定した方向の 9 マスの情報を取得します。 情報を取得するマスの範囲は図を参照してください。</p>

search 系 < 例 : searchRight() >	
	<p>実行した場所から、指定した方向に真っ直ぐ 9 マス分の情報を取得します。</p>
put 系 < 例 : putRight() >	
	<ol style="list-style-type: none"> 1. 指定した方向のマスにブロックを置きます。 2. 自分の周囲 9 マスの情報を取得します。

3. 動作

- ◆ ターゲットは、自分のターンが来ると、競技サーバに “getReady()” を送信し、自分の周囲 9 マスの情報を戻り値として取得します。
- ◆ getReady() の情報を得た後、先に挙げたメソッドのうち 1 つを実行し、そのメソッドに応じた 9 マスの周囲情報を取得します。
- ◆ getReady() の効果の範囲については、【図 3】を参照してください。



※getReady() では、5 番目の周囲情報には自分が当てはまりません。
その場合はサーバから” 0” が返されます。

4. マップ

- ◆ マップファイルには、[マップ名][制限ターン数][アイテムの位置][ブロックの位置][ターゲットの出現位置]が記述されています。
- ◆ 競技サーバでは、起動時にマップファイルの読み込み及び初期配置を行います。
- ◆ マップファイルは、各ユーザで自作することも可能です。

5. フィールド

- ◆ 競技に使用するフィールドの大きさは 15×17 マスです。
- ◆ フィールドの外側はブロックで囲まれていて、フィールド外にターゲットが出ることはできません。
- ◆ アイテム・ブロックは、中心を基準とする点対称の配置となっています。
- ◆ アイテムの数は、ラウンドごとに変動します。その際、アイテムの数は奇数になるか偶数になるかはわかりません。

6. ターン

- ◆ 2つのターゲットが交互に動作を行います。各ターゲットの1回の動作を”ターン”と呼びます。
- ◆ 制限ターン数はラウンドごとに、100～150 ターンの間で変わります。

7. ラウンド

- ◆ 2つのターゲットが競技サーバに接続し、競技が開始してから終了するまでをラウンドといいます。
- ◆ 勝利条件に早く到達したほうが、そのラウンドの勝者となります。
- ◆ アイテム獲得数が同数だった場合は、引き分け再試合となります。

8. その他

- ◆ 競技サーバとの通信の際、COOL は 2009 番ポート、HOT は 2010 番ポートを使用します。
- ◆ ブロックの上にブロックを上書き (put) した場合、ブロックで上書きされます。
- ◆ アイテムの上にブロックを上書きした場合、アイテムはブロックで上書きされます。その際、アイテムは獲得したことにはなりません。

9. 周囲情報について

- ◆ `getReady()` など取得する周囲情報は、配列 `value` を用いてクライアントに渡されます。
- ◆ 配列 `value` は、1 個の制御情報と 9 個の周囲情報の計 10 個の領域を持ちます。それぞれの値の持つ意味は【表 1】【表 2】を参照してください。

制御情報 value[0]	
0	通信終了
1	通信続行

周囲情報 value[1] ~ value[9]	
0	なし (床)
1	相手ターゲット
2	ブロック
3	アイテム

- ◆ 制御情報が 0 の場合は通信終了 (試合終了) となり、`exit()` メソッドを実行し、競技サーバから切断、プログラムを終了します。
- ◆ 周囲情報は、自分の周囲 9 マスの情報を、左上→右下の順で配列 `value` に格納します。(【図 4】参照)

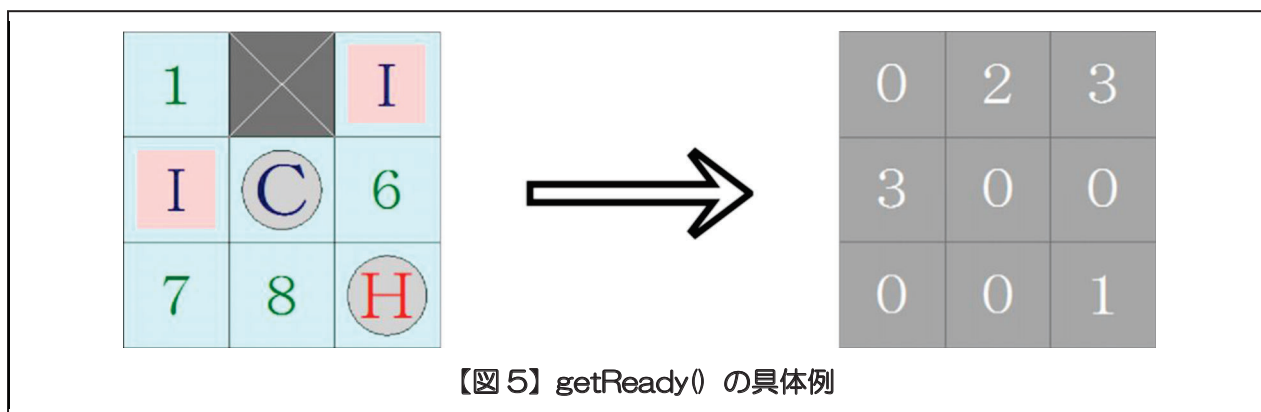
1	2	3
4	5	6
7	8	9

【図 4】 周囲情報の格納順序

※メソッドによっては、5 番目のマスには自分のターゲットがいます。
その場合は、何も無いことを示す” 0 ” が返されます。

この範囲の中に相手ターゲット・ブロック・アイテムのいずれかがある場合、それに対応する戻り値が返されます。

- ◆ 【図 5】【表 3】に、具体例を挙げますので、参考にしてください。



- ◆ 【図 5】の場合、[2]の位置にブロックが、[3][4]の位置にアイテムが、[9]の位置に相手ターゲットがいますので、返される値は右側の灰色の表のようになります。また、その時の配列 value の値は【表 3】のようになります。

【表 3】 配列 value の値

value[0]	制御情報	1
value[1]	周囲情報 1	0
value[2]	周囲情報 2	2
value[3]	周囲情報 3	3
value[4]	周囲情報 4	3
value[5]	周囲情報 5	0
value[6]	周囲情報 6	0
value[7]	周囲情報 7	0
value[8]	周囲情報 8	0
value[9]	周囲情報 9	1

- ◆ その他のメソッドでの戻り値の格納順序は、[2.メソッド]の項の各図を参照してください。図中の番号が【表 3】での周囲情報の番号に対応します。

10. 通信の仕様（上級者向け）

- ◆ Java 以外のプログラミング言語や、edu.procon.Connect2010 クラスを使わずにプログラムを組む場合は、次の手順で通信します。
- ◆ サーバのソケット番号は、クールが” 2009”、ホットが” 2010”です。

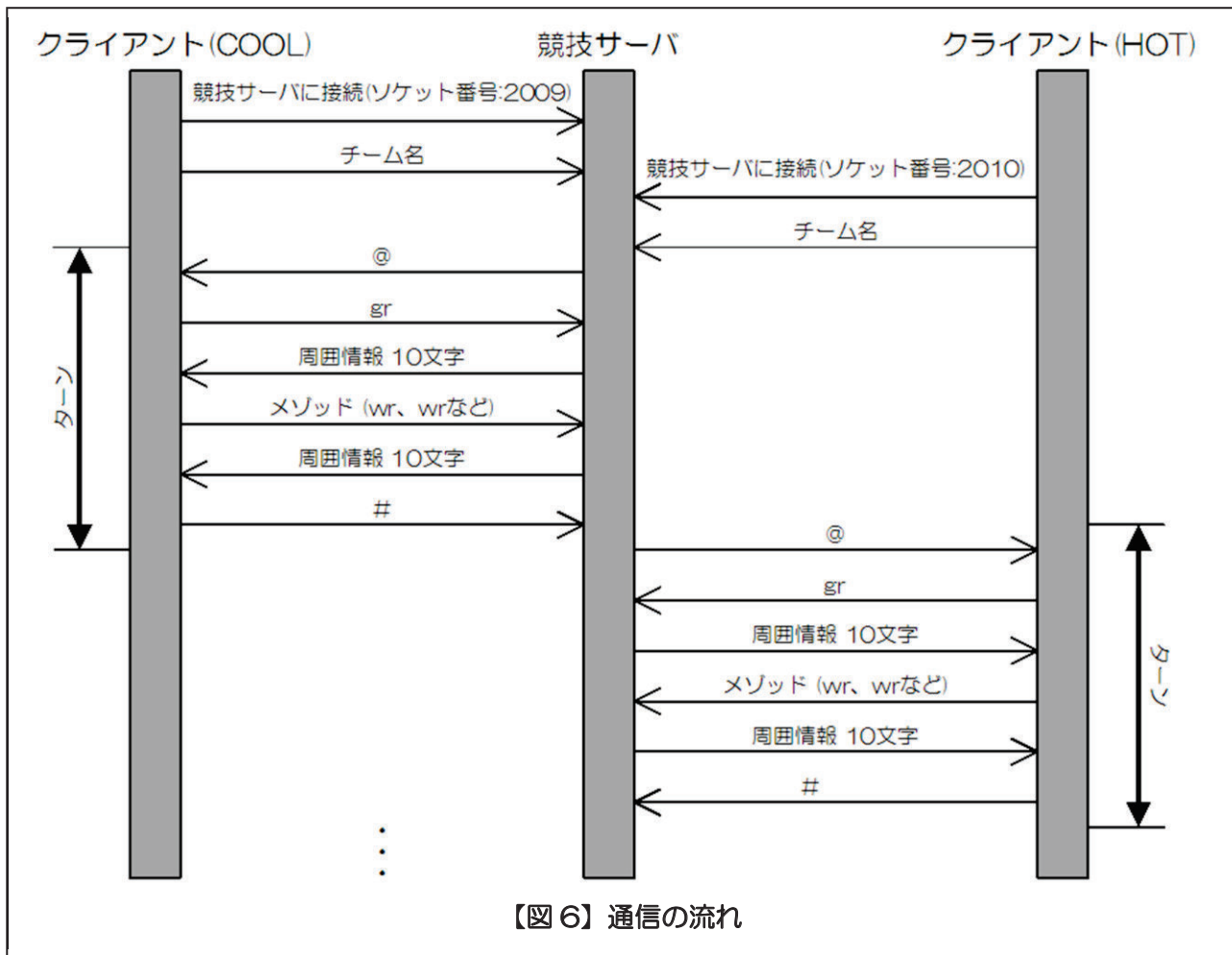
<手順>

- ① サーバと接続後、サーバへチーム名(文字列)を送信します。
チーム名の文字列は先頭 4 文字が有効となります。5 文字以降は無視されます。
また、””を送信した場合、自動的に、先攻は” COOL”、後攻は” HOT”に決定されます。
- ② サーバが” @”を送信しますので、これを受信します。
- ③ サーバに文字列” gr”を送信します。(getReady() に相当します)
文字列の末尾は” ¥r¥n”です。
- ④ サーバが送信する制御情報と周辺情報 10 文字分を受信します。
文字列は制御情報+周辺情報 9 文字で、” 1000000000”のようになっています。
(配列 value に相当します)
- ⑤ サーバにメソッドを示す文字列 2 文字を送信します。
メソッド文字列は、【表 4】を参照してください。
- ⑥ 再度、④と同様に返される文字列を受信してください。
- ⑦ サーバに文字列” #”を送信します。
- ⑧ 以降、制御情報が” 0”になるまで、②～⑦の処理を繰り返します。

【表 4】文字列によるメソッド判別

	Right	Left	Up	Down
walk	wr	wl	wu	wd
look	lr	ll	lu	ld
search	sr	sl	su	sd
put	pr	pl	pu	pd

- ◆ 【図 6】に通信の順序を図に表したものを載せておきますので、参考にしてください。



【図 6】 通信の流れ

- ◆ 全国情報技術教育研究会の競技プログラムを利用しています。
- ◆ 本ドキュメントは、旭川工業高等専門学校パソコン部によって作成されました。