スモウルビー・プログラミング甲子園
公式説明書

～大会概要からAIプログラムの作り方までが１冊に！～

スモウルビー・プログラミング甲子園開催実行委員会

目次

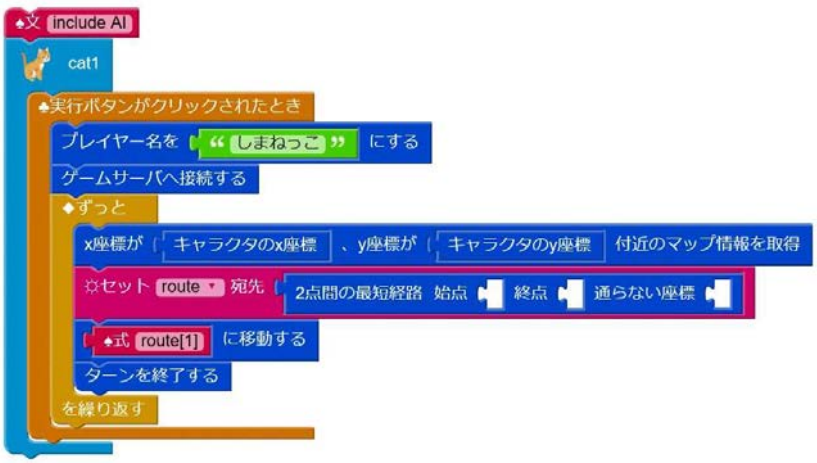
１．スモウルビー・プログラミング甲子園について

スモウルビー甲子園は、高校生以下のプログラミング競技会。
自作のＡＩプログラム同士がゲームで対戦し得点を競います。

第３回大会の作品募集中！
第２回大会は、「橋本兄弟」さんが優勝。

■募集期間　平成 29 年 7 月 3 日～平成 30 年 1 月 12 日
■募集作品　スモウルビーで作成した本大会ゲーム用のＡＩプログラム
■参加資格　高校生以下の個人又はグループ
（平成 11 年 4 月 2 日以降生まれの方。グループ参加の人数制限なし。）

ＡＩプログラムは、ビジュアル・プログラミングツール「スモウルビー」で作成します。このＡＩプログラムがスモウルビー甲子園ゲームのキャラクタを操作します。



予選上位１２組が決勝大会に出場可能。
決勝大会ではトーナメントで優勝を目指してもらいます！

■と　き　平成 30 年 3 月 24 日（土）
■ところ　くにびきメッセ（松江市）
■表　彰　優　勝：賞状、盾、15 万円相当の副賞（ＰＣ又は図書カード）
準優勝：賞状、盾、10 万円相当の副賞（ＰＣ又は図書カード）
第三位：賞状、盾、5 万円相当の副賞（タブレットＰＣ又は図書カード）
特別賞：賞状、盾、3 万円相当の副賞（タブレットＰＣ又は図書カード）

特別賞決定戦があるから、１回戦で負けても試合ができるよ！
また、タレントの池澤あやかさんも応援に来場予定。

２．ゲームの試し方

・デスクトップ上のスモウルビー甲子園 2017 アイコンをダブルクリック

・ゲームビューアが起動

・「START」ボタンをクリック

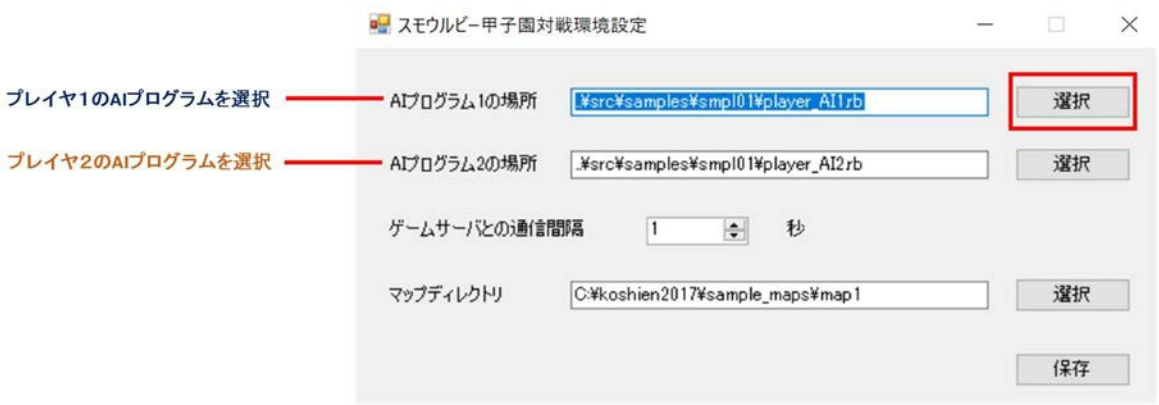
・ゲームは２ラウンドを行います。（２ラウンドの合計得点で勝敗を決定します。）
　ROUND1 が終了したら、もう一度「START」ボタンをクリックしてください。

・ゲーム終了後、再度ゲームを試したい場合は「NEW GAME」ボタンをクリック。

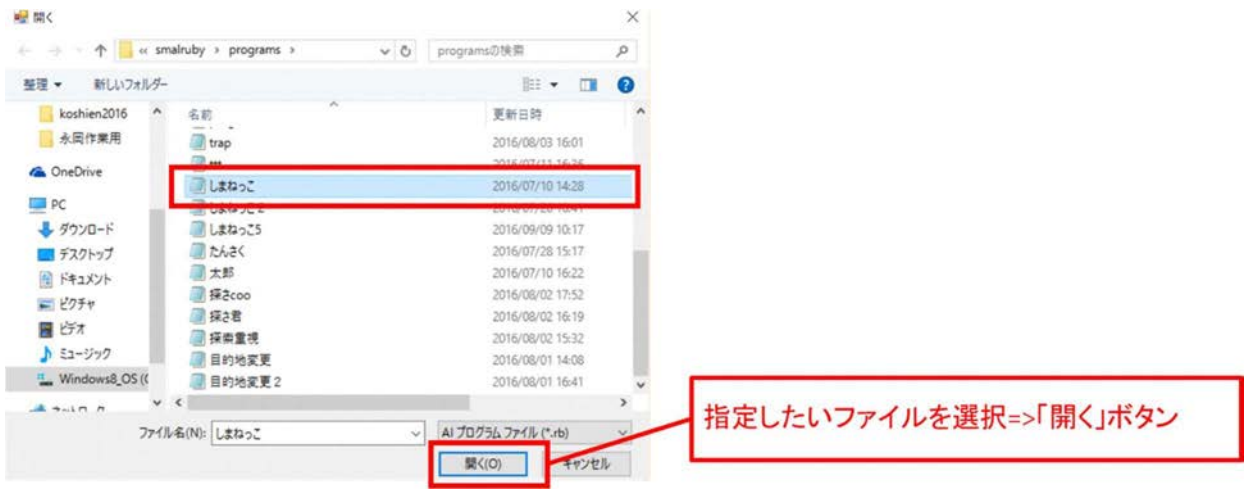
［使用する AI プログラムを変更する場合］

・「CONFIG」ボタンをクリック（使用する AI プログラムを変更する場合）

・変更したいプレイヤの「選択」ボタンをクリック



・ファイルを選択し、「開く」ボタンをクリック



［プログラムの保存場所］

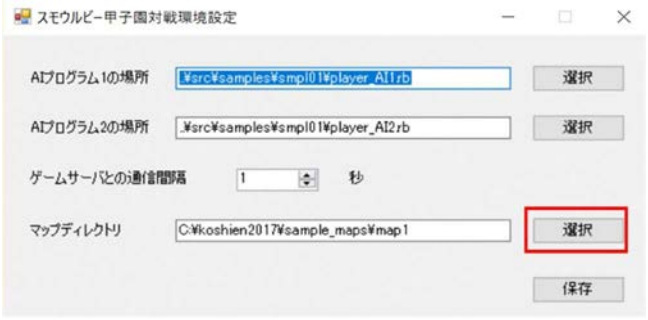
［スモウルビーで作ったプログラムの保存場所］
Cﾄﾞﾗｲﾌﾞ ＞ koshien2017 ＞ smalruby ＞ programs の中

［サンプルプログラムの格納場所］
Cﾄﾞﾗｲﾌﾞ ＞ koshien2017 ＞ src ＞ samples ＞ smpl01 / smpl02 / smpl03 の中
※順番に高度な内容になっています。=>

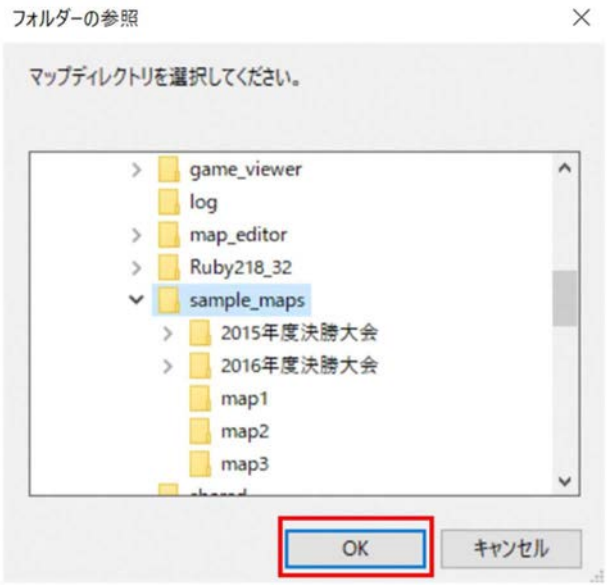
・「ゲームサーバとの通信間隔」を設定（秒数を少なくするとゲームの進行速度が速くなります。）

［対戦マップを変更する場合］

・マップディレクトリの「選択」ボタンをクリック



・フォルダを選択し、「ＯＫ」ボタンをクリック



［マップエディタで作ったプログラムの保存場所］
Ｃﾄﾞﾗｲﾌﾞ　＞　koshien2017　＞　map editor　＞　data　の中
※間違って編集しないよう保存したファイルはどこか別の移して置きましょう。
（koshien2016 ＞ map_editor ＞ data　に新しくフォルダを作って保存しておくと便利です。）

［サンプルマップ及び過去の決勝大会のマップ格納場所］
Ｃﾄﾞﾗｲﾌﾞ　＞　koshien2017　＞　samples　の中

※以上で設定は完了です。

３．スモウルビー甲子園のルール

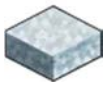
（１）基本ルール

・１５×１５マスのマップ上でゴールを目指すゲーム

・最大５０ターンのターン制で進行

・プレイヤーは１ターンに１マスだけ移動可能

・マスには、移動可能な「空間」マス、移動できない「壁」マスなど複数種類あり

種類	値	内　　容
	0	空間（自由に移動可能なマス）
	1	壁（移動できないマス）
	2	蔵（移動できないマス）　※外壁
	3	ゴール（到着するとゲーム終了し、通過はできないマス）
	4	水たまり（次回の移動命令が１回実行されないマス）

・早いターン数でゴールすれば高い得点を獲得（ゴールボーナス）

ﾀｰﾝ数	1-10	11-20	21-30	31-40	41-50	未ｺﾞｰﾙ
得　点	100 点	90 点	80 点	70 点	60 点	0 点

・マップ上には加点・減点アイテムあり

［加点アイテム］

種類	お茶 	和菓子 	丁銀 	ｼﾛｲﾙｶ 
値	a	b	c	d
得点	10 点	20 点	30 点	40 点

［減点アイテム］

種類	毒ｷﾉｺ 	蛇 	ﾄﾗﾊﾞｻﾐ 	爆弾 
値	A	B	C	D
得点	▲10 点	▲20 点	▲30 点	▲40 点

・１回の接触につき１０点減点（何度でも接触します。）

・１～４０ターンはランダムで移動しますが、プレイヤキャラクタが３マス以内に入ると、プレイヤキャラクタへ向かって移動します。

・４１～５０ターンは最寄りのプレイヤキャラクタに向かって移動します。

・プレイヤー１とプレイヤー２が等距離にいる場合は、ラウンド１はプレイヤー１へ、ラウンド２はプレイヤー２へ向かって移動します。

 ⇒ 

（１～４０ターン）　　（４１～５０ターン）

・１０マス移動で１点を加点（歩行得点）

（２）マップ情報について

・ゲーム開始時点では、プレイヤーはマップ情報を持っていません。
（自分の位置とゴールの位置のみを把握）

=>マップ情報は「マップ情報の取得」を実行した範囲（５×５マス）のみ取得できる

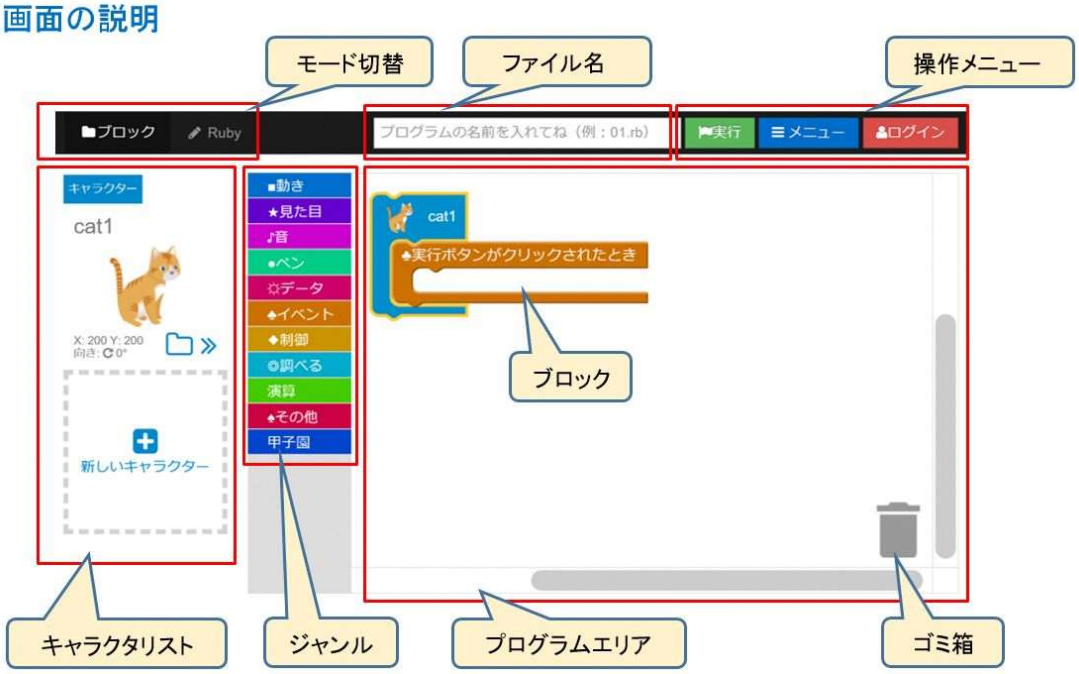
 ⇒ 

４．スモウルビーの使い方

（１）スモウルビー・エディタの起動

デスクトップ上にある「スモウルビーAI 作成（甲子園 2017)」のアイコンをダブルクリック
スモウルビー・エディタが起動します。

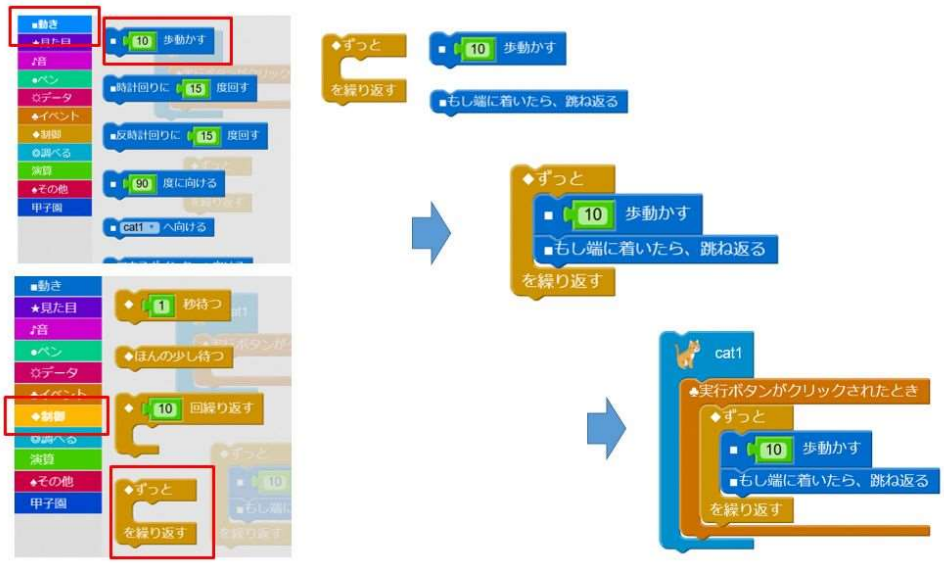
（２）スモウルビー・エディタの画面構成は次のとおりです。



プログラムエリア	スモウルビーのプログラムを組み立てるキャンバスとなる領域
ジャンル	プログラムエリアに配置できる各種命令ブロックをグルーピングした領域
ゴミ箱	プログラムエリアで不要になったブロックを廃棄するためのツール （廃棄したい命令ブロックをゴミ箱にドラッグ＆ドロップ）
キャラクタリスト	スモウルビーの実行画面に表示するキャラクタの画像を管理する領域
操作メニュー	作成したプログラムの実行や、保存・読み込みなどの各種操作を行う領域
ブロック	ジャンルより選択した命令ブロックがプログラムエリアに表示
モード切替	「スモウルビー」と「Ruby」モードの切り替え画面 （「Ruby」モードではRubyのコードに変換されて表示）

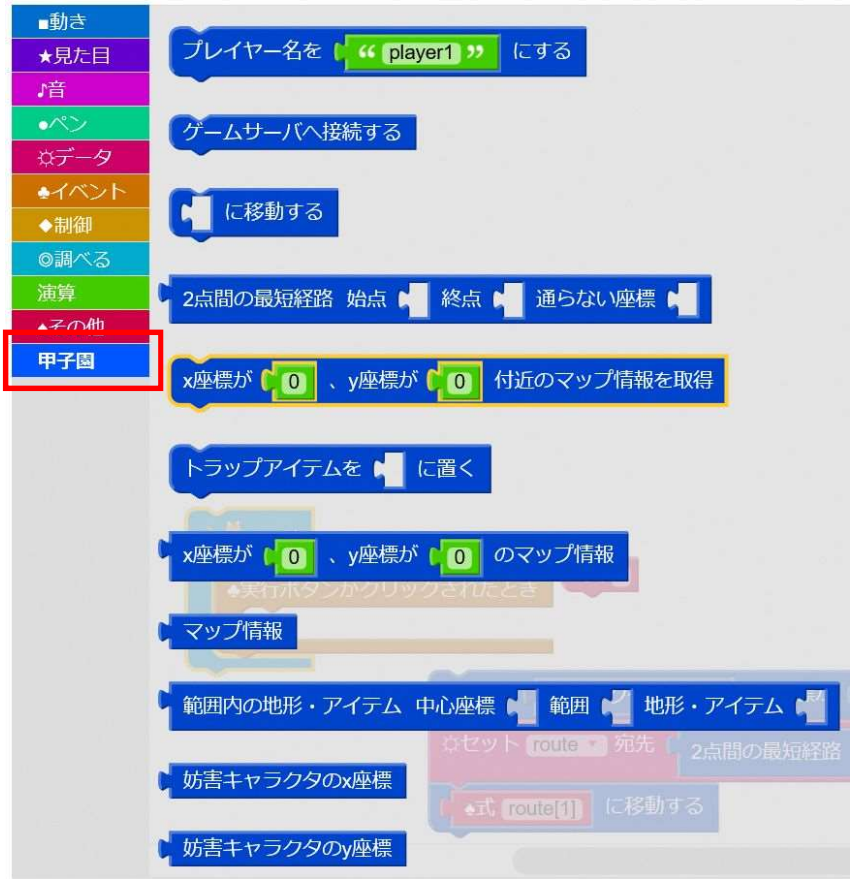
（３）ブロックの種類と使い方

・ブロックの種類と格納場所
ブロックはジャンルの中に格納されています。
このブロックを組み合わせることにより、プログラムを作ります。



・AI プログラム作成では次のジャンルのブロックをよく使います。

［甲子園］
▸スモウルビー甲子園専用に用意したブロックを格納



［データ］

▸変数を設定する場合に使用するブロックを格納

［制御］

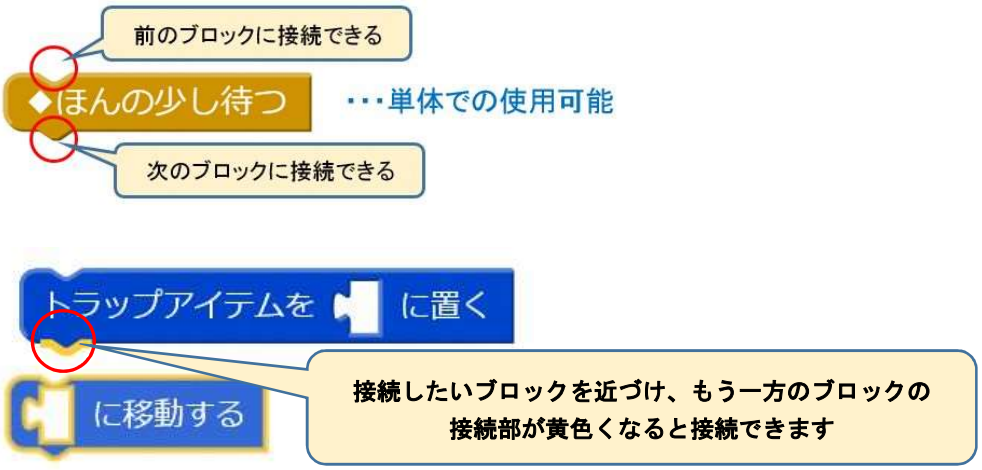
▸条件分岐で使用するブロックを格納

［その他］

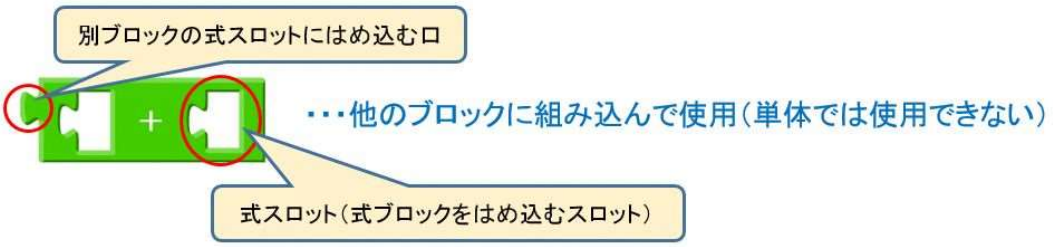
▸コード入力に使用できるブロックを格納

・ブロックの組み合わせ
ブロックには、「文ブロック」と「式ブロック」の２種類があります。

［文ブロック］
▸プログラムの処理を構成する主要なブロック
▸プログラムは、結合された文ブロックを上から下へ処理



［式ブロック］
▸式ブロックは、戻り値を伴う Ruby の式を実行するためのブロック
▸他の文ブロックや式ブロックの中にある「スロット」に組み込んで使用



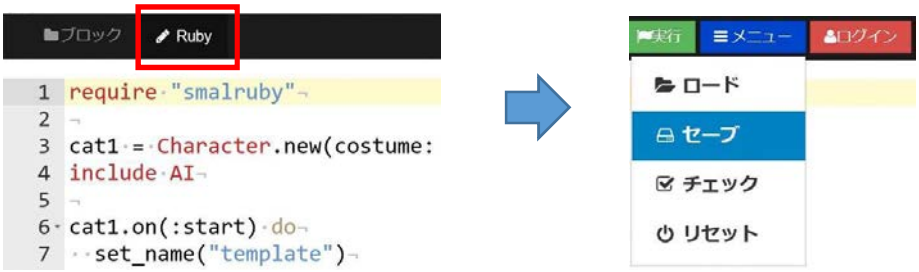
［参考］
▸文ブロック３つの組み合わせ　=>この内容が上から下に実行される
▸式ブロックは文ブロックに組み込んで使用（緑囲い部分）

（４）プログラムの保存

・作成したプログラムには名前を付けてから保存します。
※ファイル名の最後に.rb を必ず付けてください。

・ファイルは以下のディレクトリに保存されます。
C ドライブ ＞ koshien2017 ＞ smalruby ＞ programs

・「Ruby」モードに切り替え、同じようにセーブを行います。



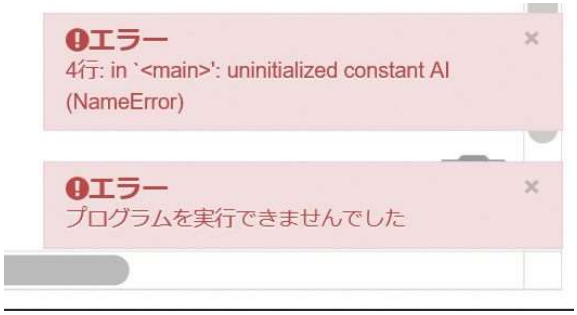
※必ず「ブロック」と「Ruby」の２つのモードでセーブを行ってください。
=>ゲームは Ruby ファイル（.rb）を使用しますが、「Ruby」モードでセーブを行わないと Ruby ファイルが更新されないためです。

（５）プログラムの実行

・作成したプログラムは、「実行」ボタンを押すことにより、実際に動かすことができます。
※AI プログラムは「実行」ボタンで動かすことはできません。

※プログラム実行画面は「ESC」キーは「×」ボタンで終了できます。

・AI プログラムを動かすことはできませんが、デバックに利用することができます。
=>AI プログラムを実行した場合、必ず２つのエラーが表示されます。
=>これ以外のエラーが表示された場合はプログラムに問題があります。



５．AI プログラムの作り方

（１）最初に覚えておくこと

・マップにはｘ軸とｙ軸があり、０から１４までの番号が付いています。

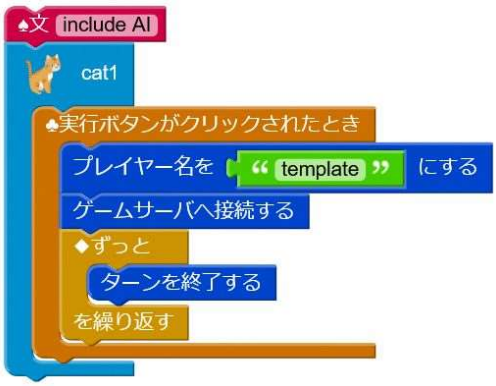
・このｘ、ｙ軸と番号を使って、特定のマスの位置を表すことができます。
マスの位置を座標と言い、[ｘ軸の番号, ｙ軸の番号]で表します。
マーカー部の座標を表すと下のようになります。

y軸＼x軸	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0															
1															
2			[2,2]					[7,2]					[12,2]		
3															
4															
5															
6															
7			[2,7]					[7,7]					[12,7]		
8															
9															
10															
11															
12			[2,12]					[7,12]					[12,12]		
13															
14															

・複数の座標を扱う場合は、次のように記載します。
[[7, 7], [7, 8], [7, 9]]

（２）テンプレートの使い方

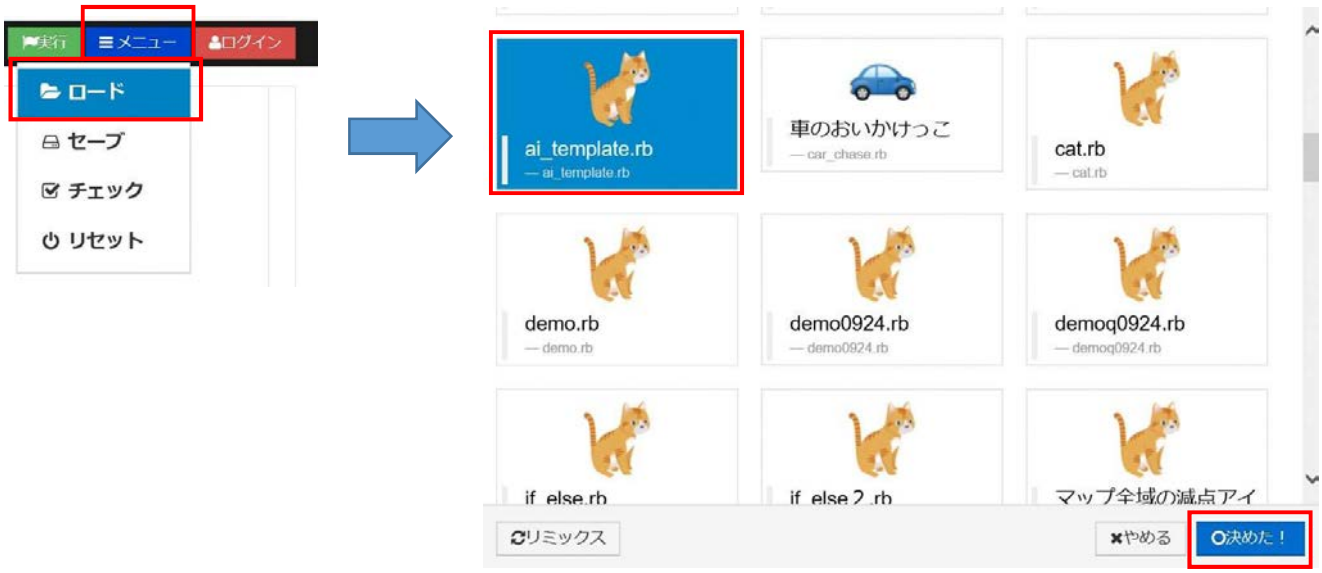
・AI プログラムは次のテンプレートを使って作成します。



・テンプレートの読み込み

スモウルビー・エディタを立ち上げ、「メニュー」―「ロード」

「ai_template.rb」を選択し「決めた」をクリック



・テンプレートの構成

=> “” の中にプレイヤ名を入力

=>「◆ずっと　　を繰り返す」の中のプログラムが１ターンの行動
　この行動がゲームの間、繰り返し実行される
=>行動の最後に必ず「ターンを終了する」を組み込むこと

（３）スモウルビー甲子園メソッドの種類と使い方

① １マス移動する

ブロック	 1回のみ ①③④のいずれか２回まで
コード	move_to（引数）
実行内容	・指定した座標にプレイヤが１マス移動します。 ・指定できるのは現在地から東西南北の１マスです。（斜めには移動できません。）
引　　数	移動先の座標を配列または変数で指定します。
解　　説	・移動できるのは空間と水たまりだけで、壁には移動できません。 ・移動できない座標を指定した場合、使用回数はカウントされますが、実行は無視されます。 ・水たまりに移動した場合は、次回の移動命令が１回無視されます。（使用回数はカウントされます。）

※１ターンの使用回数に制限のある命令があります。詳細は P20 をご覧ください。

② 最短経路を検索する

ブロック	
コード	calc_route（引数）
実行内容	・指定した２点間の最短ルートを検索します。 ・戻り値として、始点から終点までの座標配列が返されます。
引　　数	・引数としてハッシュを指定します。 ・指定できるハッシュは、「始点」「終点」「通らない座標」です。 ・それぞれのハッシュは省略も可能です。 ［ハッシュを省略した場合の設定］ ▸始点：現在地 ▸終点：ゴール ▸通らない座標：未設定
解　　説	・戻り値の座標配列は次の順番に並んでいます。 [[始点],[次の移動先],･･･経路順の座標･･･,[終点]] ・戻り値の中の各座標は並び順に 0,1,2,3･･･の番号で指定できます。 ・指定した条件での経路がない場合は始点の座標が１つだけ返されます。 ・マップ情報を取得していない範囲のマスは全て移動可能とみなして経路探索するので注意が必要です。 ・関数ブロックとセットで使用します。  ・コード入力する場合のハッシュ指定は次のとおりです。 =>src（始点）、dst（終点）、except_cells（通らない座標）

	=>入力例１　全てのハッシュを指定する場合 calc_route(src:[13,9],dst:[7,7],except_cells:[9,9]) 入力例２　通らない座標のみを指定する場合 calc_route(execpt_cells[9,9]) 入力例３　ハッシュを省略する場合 calc_route

③　５×５マスのマップ情報を取得

①③④のいずれか２回まで

ブロック	
コード	get_map_area（引数）
実行内容	・指定した範囲（５×５マス）のマップ情報を取得します。 ・戻り値として、指定した範囲のマップ情報が返されます。
引　　数	指定したい範囲の中心座標を配列または変数で指定します。
解　　説	・取得できるマップ情報は、指定した範囲の以下の情報です。 ▸マップ構成（空間・壁・水たまり・ゴール） ▸加点アイテム、減点アイテムがある場合は、その座標と種類 ▸対戦相手が指定範囲内にいる場合はその座標 ▸妨害キャラクタの現ターン開始時点の座標と前ターン開始時点の座標 （妨害キャラクタのみは指定範囲内にいなくても情報取得が可能） ・取得した情報は AI ライブラリに保存され、後から呼び出すことができます。

④　トラップアイテムを置く

①③④のいずれか２回まで

ブロック	
コード	set_trap_item（引数）
実行内容	・トラップアイテムとして爆弾（▲40 点）を現在地又は隣接する東西南北のマスに設置します。 ・トラップアイテムは１ラウンドに２回まで設置できます。 ・アイテムがあるマスには設置できません。
引　　数	・トラップアイテムを設置したい座標を配列または変数で指定します。 ・引数を省略した場合は現在地にトラップアイテムを設置します。
解　　説	・無効な座標を指定した場合、トラップアイテムは１つ消費されますが、配置はされません。（使用回数もカウントされます。） ・両プレイヤが同じ座標に同時にトラップアイテムを設置した場合、設置されるトラップアイテムは１だけになります。

⑤　指定した座標のマップ情報を呼び出す

ブロック	
コード	map（引数）
実行内容	・指定した座標のマップ情報を呼び出します。 ・指定した座標のマップ情報が返されます。
引　　数	呼び出したい座標を配列で指定します。
解　　説	・呼び出せるマップ情報は「③　マップ情報の取得」で取得した情報です。 ・マップ情報の取得を実行していない座標を指定した場合は、-1 が返されます。 ・マップエリア外を指定した場合は、nil が返されます。

⑥　マップ全体のマップ情報を呼び出す

ブロック	
コード	map_all
実行内容	・マップ全体のマップ情報を呼び出します。 ・マップ全体のマップ情報が返されます。
引　　数	―
解　　説	・呼び出せるマップ情報は「③　マップ情報の取得」で取得した情報です。 ・マップ情報の取得を実行していない座標は、-1 が返されます。

⑦　指定した範囲に、指定した要素が存在するかどうかを確認する

ブロック	
コード	locate_objects（引数）
実行内容	・指定した範囲に、指定した要素が存在するかどうかを確認します。 ・指定した要素がある座標が返されます。
引　　数	・引数としてハッシュを指定します。 ・指定できるハッシュは次のとおりです。 ▸中心座標：指定する範囲の中心座標（省略時は現在地） ▸範囲：指定する範囲の縦横のマスの長さ（省略時は５） ▸地形・アイテム：確認したい要素の値（省略時は減点アイテム全種類）
解　　説	・戻り値は座標配列で返されます。 ・配列の中の各座標は y 軸の番号が小さい順に並んでいます。（y 軸の番号が同じ場合は x 軸の小さいほうが先になります。） ・マップ情報を取得していない座標の情報は確認できません。

	・関数ブロックとセットで使用します。  ・マップ全体を範囲指定したい場合は次のようになります。 =>減点アイテム  =>加点アイテム  ・コード入力する場合のハッシュ指定は次のとおりです。 =>cent（中心座標）、sq_size（範囲）、objects（地形・アイテム） =>入力例１　マップ全体の加点アイテムの座標を確認する場合 locate_objects(cent:[7,7],sq_size:15,objects[“a”,”b”,”c”,”d”]) =>入力例２　マップ全体の水たまりの座標を確認する場合 locate_objects(cent:[7,7],sq_size:15,objects[4])

⑧　対戦キャラクタのｘ軸、ｙ軸の座標

ブロック	
コード	other_player_x　　other_player_y
実行内容	・対戦相手の座標を呼び出します。 ・最後にマップ情報の取得で把握した対戦相手の座標（ｘ、ｙ）を返します。
引　　数	―
解　　説	・得られる情報は、マップ情報の取得で把握した時点の情報です。 ・対戦相手の座標は、マップ情報の取得の範囲に対戦相手がいないと把握できません。 ・対戦相手の座標を把握していない場合はnilが返されます。 ・マップ情報の取得を繰り返し行っている場合、情報が上書きされていくため、一度把握した対戦相手の座標を見失う場合があります。

⑨　妨害キャラクタのｘ軸、ｙ軸の座標

ブロック	
コード	enemy_x　　enemy_y
実行内容	・妨害キャラクタの座標を呼び出します。 ・最後にマップ情報の取得を実行した時点での妨害キャラクタの座標（ｘ、ｙ）を返します。
引　　数	―
解　　説	・得られる情報は、最後にマップ情報の取得を実行した時点の情報です。

	・妨害キャラクタの座標は、マップ情報の取得の範囲に妨害キャラクタがいなくても把握できます。

⑩ ゴールのｘ軸、ｙ軸の座標

ブロック	
コード	goal_x　　goal_y
実行内容	・ゴールの座標を呼び出します。 ・ゴールの座標が返されます。
引　　数	―
解　　説	・ゴールの座標は、マップ情報の取得に関わらず呼び出し可能です。

⑪ キャラクタのｘ軸、ｙ軸の座標

ブロック	
コード	player_x　　player_y
実行内容	・キャラクタの座標を呼び出します。 ・キャラクタの座標が返されます。
引　　数	―
解　　説	・キャラクタの座標は、マップ情報の取得に関わらず呼び出し可能です。

⑫ プレイヤの名前を付ける

ブロック	
コード	set_name（プレイヤ名）
実行内容	・プレイヤ名で設定した名前がゲーム時にプレイヤ名として表示されます。
引　　数	・引数としてプレイヤ名を入力します。
解　　説	・文字列１４文字まで入力できます。

⑬ ターンを終了する

ブロック	
コード	turn_over

実行内容	・現在のターンを終了させ、次のターンを待ちます。
引　　数	ー
解　　説	・必ずターンの最後に１回実行する必要があります。 （ターン終了を実行しないとタイムアウトでゲームが終了します。）

［甲子園メソッドの使用制限］

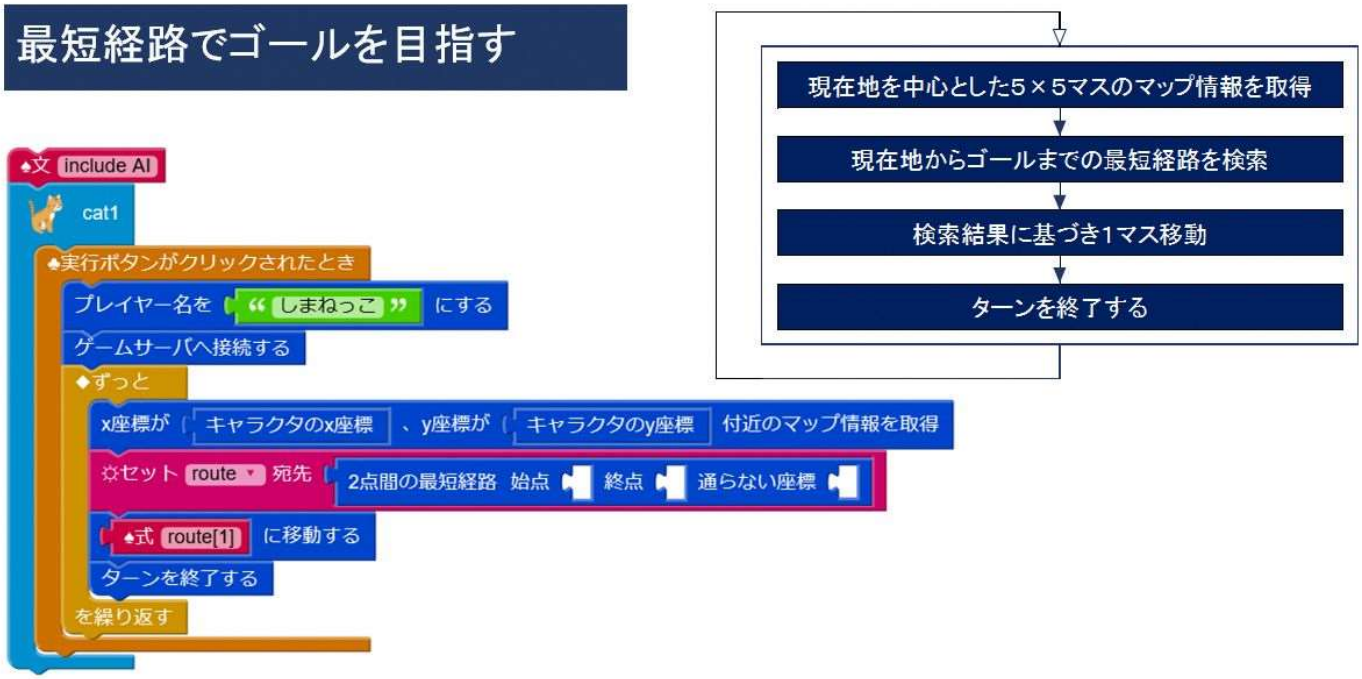
・次の命令には１ターンでの使用回数に制限があるので、注意してください。
（使用回数を超えた命令は無視されます。）

=>１ターン１回

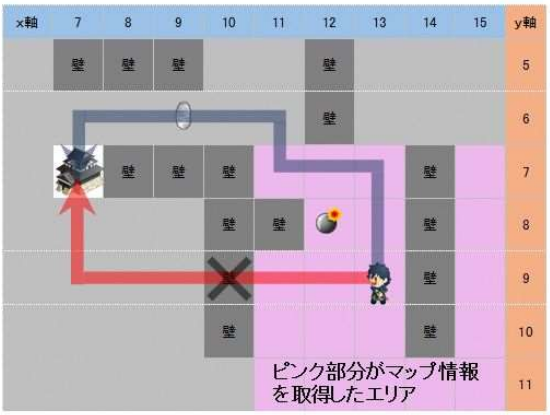
=>次の３つの使用回数はいずれか２回
（移動する以外は同じ命令を２回使用することも可能です。）

（４）簡単な AI プログラムを作ってみる

・次のプログラムを作ってみましょう



- 戻り値は、経路順の座標が返される（0からの番号で指定可能）
- 引数を省略した場合は次の設定で経路を検索
  - ▶始点：プレイヤキャラクタの現在地
  - ▶終点：ゴール
  - ▶通らない座標：設定なし
- 未探索エリアは通行可能として経路検索（通れない経路を返す場合がある）
- 変数をつけることで後から経路検索の結果を呼び出し可能



・先ほど作ったプログラムを改良します。

減点アイテムを避けてゴールを目指す

- 指定したマップ要素やアイテムの座標を確認するためのメソッド
- 取得したマップ情報から情報を呼び出すため、未探索エリアの情報は取得できません。
- 引数として以下を指定
  - ▶中心座標：呼び出す範囲の中心座標を指定。マップ全体の場合は[7,7]
  - ▶範囲：呼び出す範囲のマップサイズを指定。マップ全体の場合は15
  - ▶地形・アイテム：呼び出す地形・アイテムの値を指定
    - =>省略した場合は減点アイテムを呼び出す
    - =>値については別途説明します。
- 変数をつけることで後からこの情報を呼び出すことが可能

- 減点アイテムを避けてゴールする経路がない場合もあり。
- このため、経路がなかった場合の行動もプログラムする必要あり。
  =>ある場合とない場合の2種類の行動が必要
- こうした場合に条件分岐を使用
- 条件に合致した場合（真）のみ、ブロックの中のプログラムを実行し、合致しない場合は、このブロックの処理は実行されない。

- 上のブロックでは変数「route」が2つあり、それぞれ中身が違う
- プログラムは上から順番に処理
  =>最初にAの内容が変数「route」となる
  =>減点アイテムを避ける経路がない場合は、Bが実行され、Bの内容が変数「route」になる
  =>減点アイテムを避ける経路がある場合は、Bの内容は実行されないため変数「route」はAのまま

６．便利なメソッドの紹介

［紹介する内容］

▸30 点と 40 点の加点アイテムの場所を確認する
▸配列に含まれる座標の数を確認する
（例：経路検索した経路があるかを確認する）
▸ゴールの１マス手前にトラップアイテムを置く
▸一番近い加点アイテムを目指して移動する
▸ターン数を管理する

［参考］

・スモウルビー甲子園で用意した命令ブロックは、ジャンル「甲子園」の中に入っています。
・また、ジャンルとブロックの色は一緒になっていますので、色を参考にブロックを探すこともできます。
・よく使う関数ブロックや文・式ブロックは次のジャンルに入っています。

（１） 30 点と 40 点の加点アイテムの場所を確認する

・下のブロックを使うと、マップ探索で取得したマップ情報から指定したマップ要素やアイテムの座標を呼び出すことができます。

=>マップ全体の情報を呼び出す場合は、中心座標[7,7]、範囲 15 を式ブロックで指定します。

・30 点と 40 点の加点アイテムの座標を確認したい場合は、地形・アイテムは次のように指定します。
（加点アイテムの値　丁銀 30 点：c、シロイルカ 40 点：d）

["c","d"]

[使用例]

▸水たまりの場合 

▸加点アイテム全部の場合 

▸シロイルカ（加点アイテム 40 点）の場合 

▸減点アイテム全部の場合 

（引数未指定）

▸爆弾（減点アイテム 40 点）の場合 

=>引数は配列で指定するため、必ず[ ]の中に指定したい値を入力してください。
　また、文字の場合は”” を必ず付けてください。
=>指定できるマップ要素、アイテムの種類と値とは次のとおり
=>引数を省略した場合は減点アイテム 4 種類の場所が返されます。

[アイテム等の種類と値]

加点アイテム	値	得点
お茶	a	10
和菓子	b	20
丁銀	c	30
シロイルカ	d	40

減点アイテム	値	得点
毒キノコ	A	△10
蛇	B	△20
トラバサミ	C	△30
爆弾	D	△40

マスの種類	値
空間	0
壁	1
蔵(外壁)	2
ゴール	3
水たまり	4

・変数ブロックに、要素確認ブロックを組み込み、名前「item_over30」を付けます。
（名前は自由に付けられます。）
これで後からこの情報を使用することができます。

［注意］
・マップ探索を実施したエリアの情報のみが確認できます。
（マップ探索をしていないエリアの情報は確認できません。）
・また、確認できる情報はマップ探索を行った時点での情報です。
アイテムを対戦相手が取得してなくなっているなど、情報が古くなっている場合がありますので、注意してください。

（２）配列に含まれる座標の数を確認する
（例：経路検索した経路があるかを確認する）

・Ruby の length メソッドを使用することにより、配列に含まれる座標の数を確認することができます。（length メソッドでは、指定した配列に含まれる要素の数を返します。）
・例えば、経路検索では、指定した条件の経路がない場合は、始点の座標を１つだけ返します。このことから、length メソッドを使うことにより、経路のありなしを確認できます。（戻り値が１の場合は経路がないことが分かります。）
また、length メソッドを使って、加点アイテムの有無なども確認できます。

［使用例］
“減点アイテムを避けてゴールする経路を検索し、
もし、この経路がない場合は、ゴールまでの最短経路を検索する”

［ブロックの処理内容］
・減点アイテムの座標を確認し、その結果に「traps」と名前を付ける
・始点：現在地、終点：ゴール、通らない座標：減点アイテムで、経路を検索し、その結果に「route」と名前を付ける
・もし、「route」に含まれる座標の数が１であれば（経路がない場合は）、始点：現在地、終点：ゴール、通らない座標：未設定で経路検索する

（３）ゴールの１マス手前にトラップアイテムを置く

・length メソッドを使って、ゴールの１マス手前かどうかを判定することができます。
経路検索の結果は、現在地からゴールまでの経路順の座標が返されるので、ゴールの１マス手前では、現在地とゴールの２つの座標が返されます。よって、length=2 の場合が１マス手前になります。

・このブロックを条件分岐ブロックに組み込みます。

・続いて条件分岐ブロックの中に、条件が真の場合の処理を組み込みます。
ここでは「トラップアイテムを置く」と「移動」ブロックを組み込みます。
これで完成です。

[ブロックの処理内容]
・現在地からゴールまでの最短経路を検索し、その結果に「route」という名前を付ける。
・「route」に含まれる座標の数が 2 かどうかを判定する。
・真（座標数が 2）の場合は、トラップアイテムを現在地に置き、移動（ゴール）する。
・偽（座標数が 2 以外）の場合は、次の処理に進む。

（４）一番近い加点アイテムを目指して移動する

・マップ上の指定した要素（アイテムマップ要素）の座標を確認する方法は(3)でやったとおりですが、今度はマップ上の加点アイテムの座標を確認し、一番近い加点アイテムを終点として移動するブロックを作ります。

・最初に加点アイテムの座標を確認し、この結果に「items」という名前を付けます。

・次に「items」に含まれる座標を現在地から近い順に並べ替えます。
（「items」に含まれる座標は、y 軸の座標数が小さい順に並んでいます。）
この処理を「sort_by!」「calc_route」「size」メソッドを使って行います。

sort_by!	ブロックを使って配列に含まれる要素をソートし、配列自身を変更するメソッドです。ブロック引数に配列に含まれる各要素を入れながらブロックを繰り返し実行し、ブロックの戻り値を集めます。集めた戻り値を<=>演算子で比較して、小さい順に要素を並べます。 配列.sort_by!{|変数| ブロック（ブロック引数）} 例） animals = [“mouse”, “cat”, “elephant”, “lion”] animals.sort_by! {|animal| animal.size } ※ .size メソッドは文字数を返します。 =>配列の中身が次のように小さい順に並べ替えられます。 animals = [“cat”, “lion”, “mouse”, “elephant”]
calc_route	スモウルビー甲子園で用意した２点間の最短経路を検索するメソッドです。（下のブロックと同じ内容です。）  引数として始点（src)、終点（dst)、通らない座標（except_cells）を指定できます。 なお、引数を省略した場合は、始点：現在地、終点：ゴール、通らない座標：未設定で経路検索します。 calc_route(src:配列,dst:配列,except_cells:配列)

	例） calc_route(src:[11,8],dst:[7,7],except_cells:[9,9]) =>次のような経路順の座標配列が返されます。 [[11,8],[11,7],[10,7],[9,7],[8,7],[7,7]]
size	配列の要素の数を返します。 配列.size 例） numbers = [1,2,3,4,5] numbers.size =>4

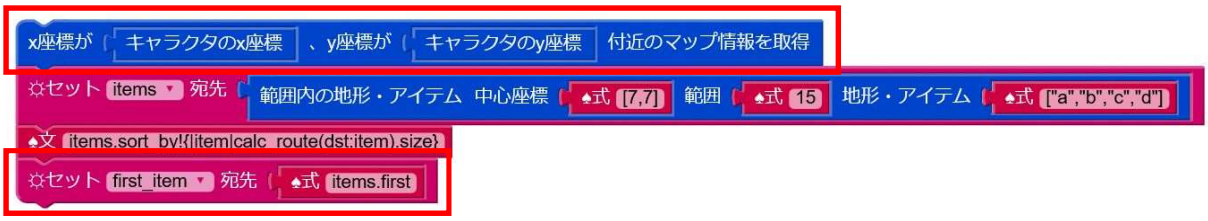
・上で説明した「sort_by!」「calc_route」「size」メソッドを次のように組み合わせます。

[処理内容]
・「items」に含まれる加点アイテムの各座標に「item」という変数を設定
・calc_route を繰り返し実行（「item」（加点アイテム）が 5 つであれば 5 回繰り返す）
　始点：現在地
　終点：item（「items」に含まれる各加点アイテムの座標）
　通らない座標：未設定
・「item」の各戻り値の座標数（要素数）を求める
・この座標数が少ない順に「items」の中の座標を並べ替える
　※calc_route は経路順の座標を返すので、座標数が少ない＝現在地から近いになる

・次に一番近い加点アイテムを終点として設定できるよう「items」に含まれる先頭の座標に「first_item」という名前を付けます。

first メソッドは、配列の最初の要素を返します。
items.first では、「items」の先頭の座標を返します。

・一番近い加点アイテムを取得すると、次に近い加点アイテムを目指して移動する必要があります。
この処理を正しく行うためには、マップ情報を新しくし、「items」の情報を更新していく必要があります。
このため、この一連のブロックの先頭にマップ情報の取得ブロックを組み込んでおきます。
=>これにより、ターンのたびに、まずは現在地周辺のマップ情報を取得し、「items」の情報が更新されます。
=>ただし、情報が更新されるのは現在地周辺（5×5 マス）だけのため、この範囲外で対戦相手が加点アイテムを取得した情報は取得できません。



・最後に「first_item」を終点に設定して移動するブロックを組み込んで完成です。

（５）ターン数を管理する

・１ターンの行動として、AI プログラムの「◆ずっと　　を繰り返す」の中のプログラムが１回実行されます。よって、「◆ずっと　　を繰り返す」の実行回数がターン数と同じ数になります。
このことから、「◆ずっと　　を繰り返す」が実行された回数を変数設定することによりターン数を管理することができます。

・最初に「◆ずっと　　を繰り返す」より前に「turn = 1」を設定します。
（これで最初のプログラム実行は「turn = 1」としてスタートします。）

・次に「ターンを終了する」の前に「turn = turn + 1」を組み込みます。
（ターン終了時に「turn」に 1 を加えることにより、ターン数が加算されていきます。）

・ターン数を管理することにより、次のようにターン数毎の行動をプログラムすることができます。

７．作戦の考え方

（１）やり方１

▸やりたいことを書き出す
▸実装するためのプログラムを考える（まずはできることから）
▸ゲームで試す
▸改良する

例）
・やりたいこと
「加点アイテムをとってからゴールに向かう」
・実装するためのプログラムを考える
「最初に全マップの情報を取得する」
「加点アイテムの座標を確認する」
「加点アイテムの座標を近い順に並べ替える」
「一番近い加点アイテムに向かう」
「加点アイテムがなくなったらゴールに向かう」
・ゲームで試す
「加点アイテムを全部とっているとゴールできないなあ
（５０ターン経過してしまう)」
・改良する
「３０点と４０点の加点アイテムを優先しよう！」
「３０ターン経過したらゴールに向かおう！」　など

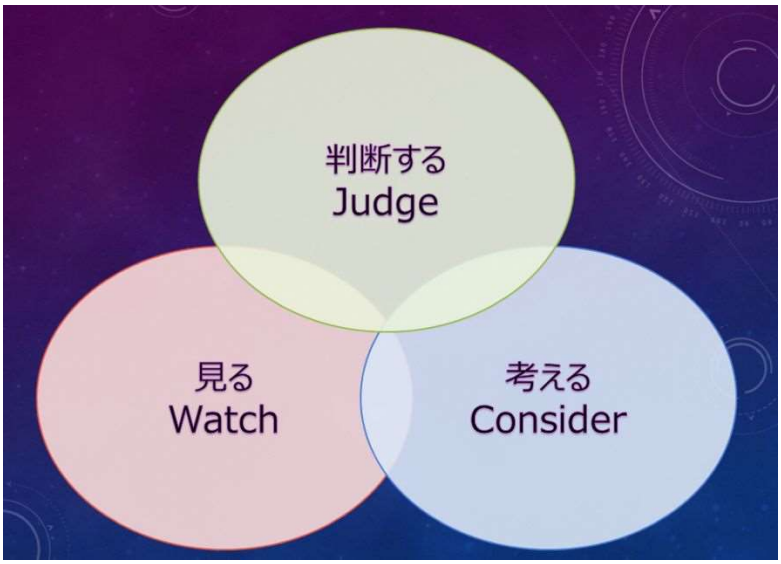
（２）やり方２

▸サンプルプログラムを流用する
▸サンプルプログラムでゲームを試す
▸改良したい点を考える
▸実装方法を考える
▸ゲームで試す

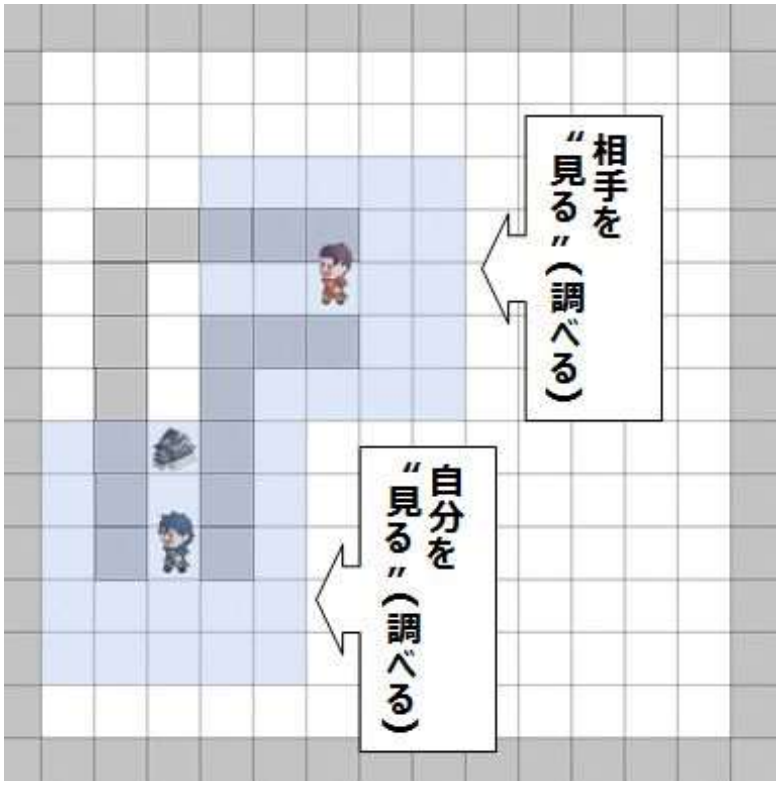
※サンプルプログラム３は高度だけど流用するためにはプログラム内容を理解することが重要

（３）第１回大会優勝者「高田.exe」さんからのアドバイス

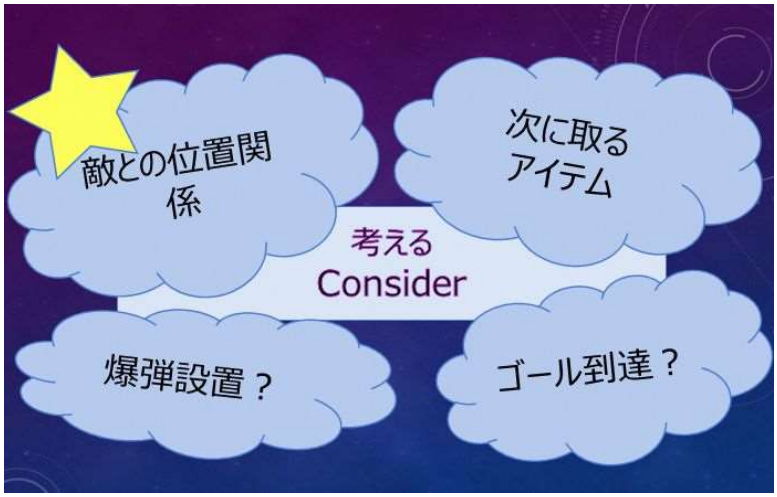
・「見る」「考える」「判断する」が重要

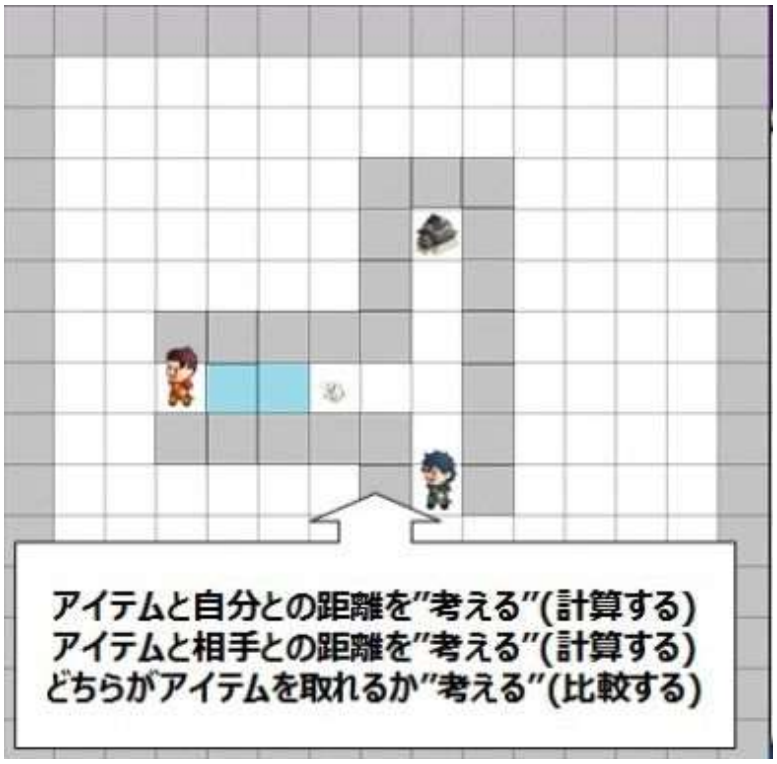


・「見る」＝「調べる」、自分だけでなく相手も見る



・「考える」＝「計算する・比較する」





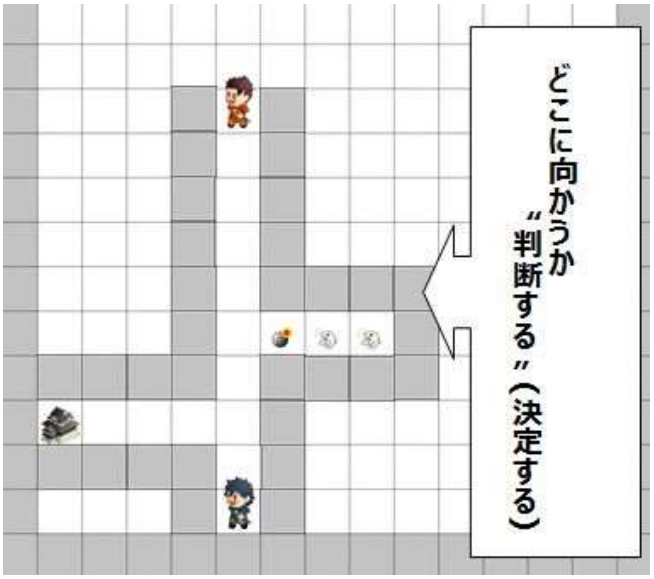
「自分と加点アイテムとの距離」
「相手と加点アイテムとの距離」
=>相手の方が近い場合は、その加点アイテムは相手がとるだろう！
=>だったらゴールに向かった方が有利？
=>ついでにゴール前にトラップアイテムを置いて相手の減点を狙おう！

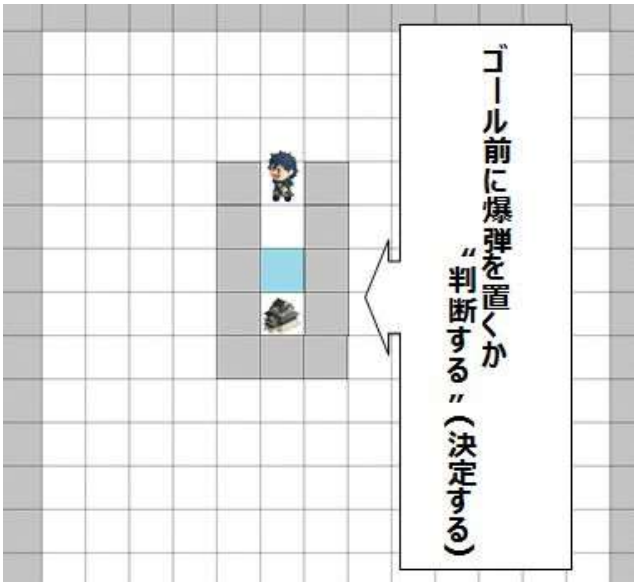
・「判断する」＝「見る」「考える」の結果に応じて

=〉 アイテムを妥協

=〉 目的地を判断

=〉 爆弾をどこに置くかを判断



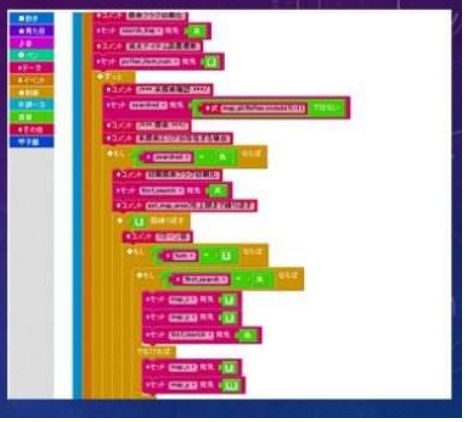


ここからはじめよう

テキストエディタ(メモ帳等)に
自分のやりたいことを書き出してみる

ここからはじめよう

スモウルビーに慣れる



ここからはじめよう

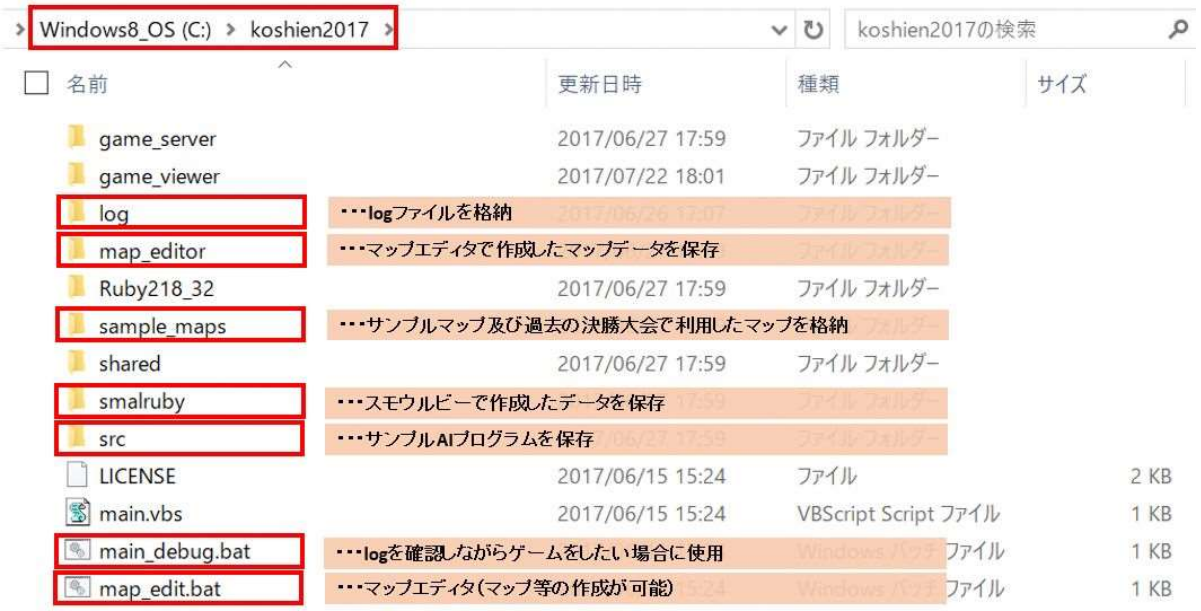
できればRubyにチャレンジ

あなただけのAIプログラムを

８．その他の説明

（１）スモウルビー甲子園の構成

・スモウルビー甲子園は、Ｃドライブ上の「koshien2017」フォルダに保存されています。

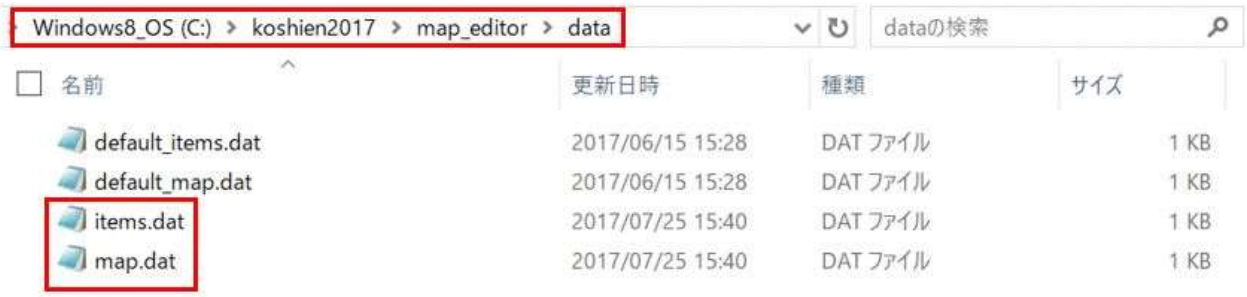


（２）マップエディタの使い方

・マップエディタを使って、オリジナルのマップを作ることができます。

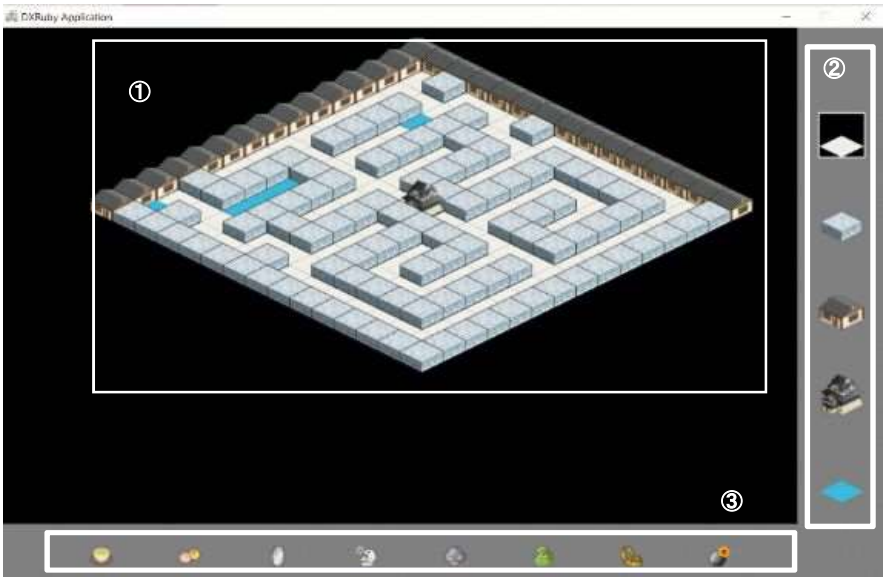
・マップエディタで使用するファイルは以下に格納されています。
このファイルを game_server 上のファイルと置き換えることで作成したマップでゲームを試せます。

[map_editor]



※編集したいマップファイルをここに貼り付けます。

・マップエディタの使用方法は次のとおりです。



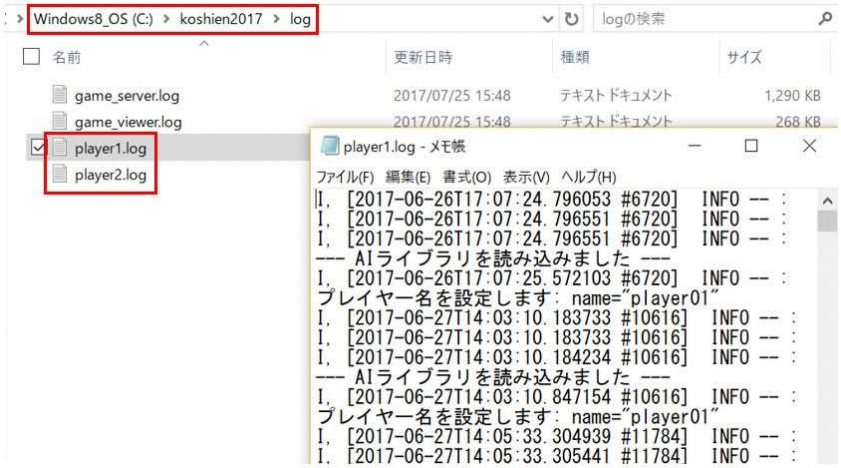
① マップ描画エリア
② マップチップ選択エリア
③ アイテム選択エリア

[操作方法]
・マップチップ選択エリアから配置したいマップチップを選択（右クリック）し、マップ描画上の配置したい場所に右クリックすると、新しいマップチップが配置されます。
・同様にアイテム選択エリアから配置したいアイテムを選択（右クリック）し、マップ描画上の配置したい場所に右クリックすると、新しいアイテムが配置されます。

[編集したマップの保存方法等]
・Esc キーを押すことにより、データが保存され、マップエディタが終了します。
・ファイルは　koshien2017 > map_editor > data　に保存されています。
・間違って編集しないよう保存したファイルはどこか別の移して置きましょう。
（koshien2017 > map_editor > data　に新しくフォルダを作って保存しておくと便利です。）

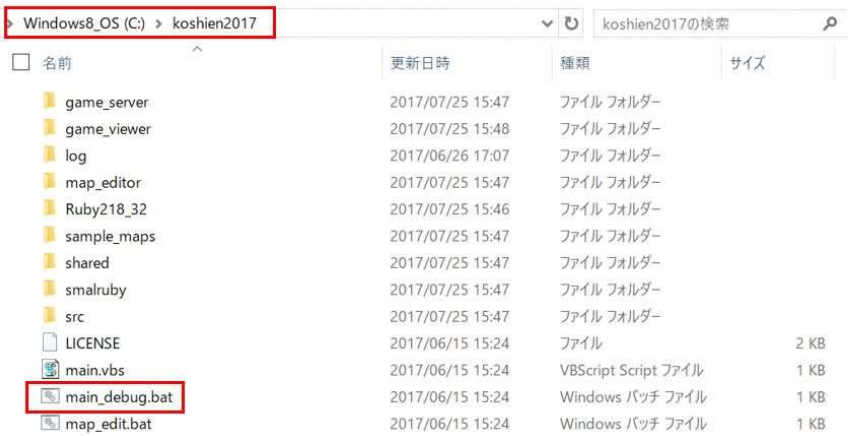
（３）ログの確認方法

・ｌｏｇファイルを確認することができます。



・また、ｐメソッドを組み込んでおくことにより、変数の中身を確認することもできます。
（ｌｏｇファイルに変数の中身も記録されます。）

・main_debug からゲームを起動することで、コマンドプロンプトを見ながらゲームを試すこともできます。



９．参考

（１）例題

（例題１）

１．最初にマップ全体の「マップ情報の取得」を実施

２．マップ探索が終わったら、全ての加点アイテムの座標を確認

３．一番近い加点アイテムを終点に移動（アイテムを取ったら、次の加点アイテムに向かう）

４．３０ターンになったらゴールに向かう

※注意１：加点アイテムを取った後に移動先を見失わないように！

（例題２）

１．最初にマップ全体の「マップ情報の取得」を実施

２．マップ探索が終わったら、
・３０点の加点アイテムの座標を確認
・４０点の加点アイテムの座標を確認
・▲３０点と▲４０点の減点アイテムの座標を確認

３．次の条件で移動
・終点：４０点の加点アイテム
・通らない座標：▲３０点、▲４０点の減点アイテム
※条件を満たすルートがない場合は、通らない座標は未設定で移動（終点は同じ）

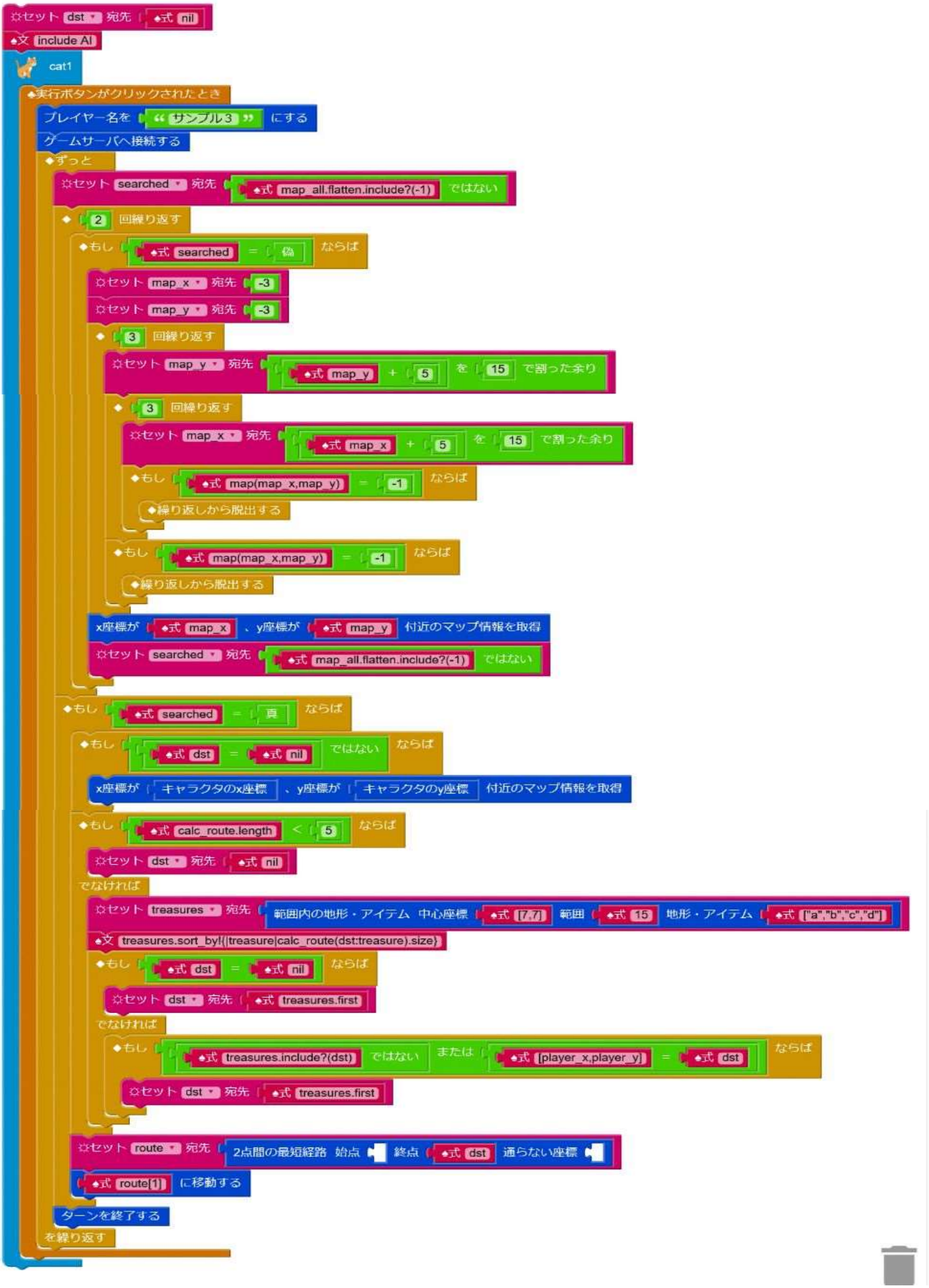
４．４０点の加点アイテムがなくなったら、次の条件で移動
・終点：３０点の加点アイテム
・通らない座標：▲３０点、▲４０点の減点アイテム
※条件を満たすルートがない場合は、通らない座標は未設定で移動（終点は同じ）

５．３０点の加点アイテムもなくなったら、ゴールに向かう

６．ゴールの１マス手前でトラップアイテムを置く

※注意１：加点アイテムを取った後に移動先を見失わないように！
注意２：自分で設置したトラップアイテムに引っかからないように！

（２）サンプル３


☆セット dst 宛先 ◆式 nil
◆文 include AI
cat1
◆実行ボタンがクリックされたとき
プレイヤー名を “ サンプル３ ” にする
ゲームサーバへ接続する
◆ずっと
☆セット searched 宛先 ◆式 map_all.flatten.include?(-1) ではない
◆ 2 回繰り返す
◆もし ◆式 searched = 偽 ならば
☆セット map_x 宛先 -3
☆セット map_y 宛先 -3
◆ 3 回繰り返す
☆セット map_y 宛先 ◆式 map_y + 5 を 15 で割った余り
◆ 3 回繰り返す
☆セット map_x 宛先 ◆式 map_x + 5 を 15 で割った余り
◆もし ◆式 map(map_x,map_y) = -1 ならば
◆繰り返しから脱出する
◆もし ◆式 map(map_x,map_y) = -1 ならば
◆繰り返しから脱出する
x座標が ◆式 map_x 、y座標が ◆式 map_y 付近のマップ情報を取得
☆セット searched 宛先 ◆式 map_all.flatten.include?(-1) ではない
◆もし ◆式 searched = 真 ならば
◆もし ◆式 dst = ◆式 nil ではない ならば
x座標が キャラクタのx座標 、y座標が キャラクタのy座標 付近のマップ情報を取得
◆もし ◆式 calc_route.length < 5 ならば
☆セット dst 宛先 ◆式 nil
でなければ
☆セット treasures 宛先 範囲内の地形・アイテム 中心座標 ◆式 [7,7] 範囲 ◆式 15 地形・アイテム ◆式 ["a","b","c","d"]
◆文 treasures.sort_by!{|treasure|calc_route(dst:treasure).size}
◆もし ◆式 dst = ◆式 nil ならば
☆セット dst 宛先 ◆式 treasures.first
でなければ
◆もし ◆式 treasures.include?(dst) ではない または ◆式 [player_x,player_y] = ◆式 dst ならば
☆セット dst 宛先 ◆式 treasures.first
☆セット route 宛先 2点間の最短経路 始点 終点 ◆式 dst 通らない座標
◆式 route[1] に移動する
ターンを終了する
を繰り返す