サムライジョッキーゲームルール

情報処理学会プログラミングコンテスト委員会

2017/12/26

概 要

SamurAI Coding 2017–18 に用いるゲームのルールを述べる．

1　ゲームの概要

AI の制御するふたりのプレイヤがスタート位置から開始し，障害物が設置されたレースコース上でステップごとに位置を変えながら，ゴールに早く到達することを競うゲームである．

1 ゲームは，同じレースコースを用い両プレイヤのスタート位置を交換した 2 レースからなる． 両レースのゴールタイムの合計が短い方がゲームの勝者である．合計タイムが同一である場合は，そのゲームは引き分けである．

2　レースコース

レースコースは 2 次元格子状であり，ゲーム毎に大きさや障害物の位置が異なる．以下ではレースコースの幅を w，コース長を l とし，各格子点の位置の座標を (x, y) $(0 \leq x < w, 0 \leq y)$ とする．

コース上の格子点の一部は**障害点**である． 障害点あるいは縦横斜めに隣接するふたつの障害点を結んだ線分を**障害物**と呼ぶ． 障害物は固定されており，レース中に動くことはない．障害物を飛び越すことはできないので，障害物に囲まれた領域に入ることはできない．

プレイヤを制御する AI プログラム (以下単に **AI** と呼ぶ) に与えられる障害物の情報は，プレイヤの視界の範囲内のもの，すなわちプレイヤの位置の前後一定範囲の y 座標にあるものについてのみである． 加減速は限定された範囲でしかできないので，速度を上げ過ぎると障害物を避けられなくなる可能性がある．

図 1 にレースコースの例を示す． 茶色の小円は障害点，線分や塗りつぶした領域は障害物およびそれらに囲まれた領域である．

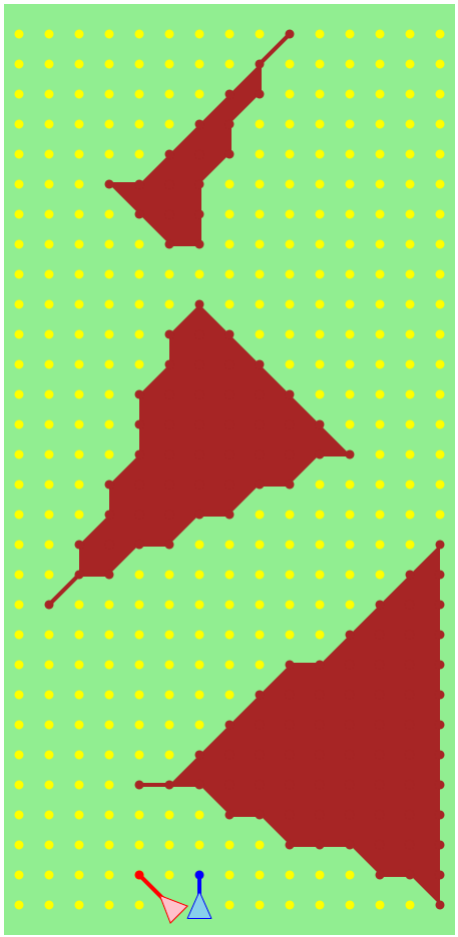

図 1: レースコースの例

レース中，両プレイヤはいずれかの相異なる格子点にある．レース開始時の両プレイヤの位置の x 座標は異なり，y 座標は 0 である． 両プレイヤはステップごとに位置を変えて行き，コース長 l 以上の y 座標の地点に達したらゴールしたことになる．

コースには行き止まりがないものとする． レース開始時の両プレイヤの位置から到達可能なコース内のどの格子点からも，十分小さい速度で移動すれば，一度も y 座標を減らすことなく障害物やコース端にぶつからずに y 座標がより大きい点に移動できる．

3　レースの進行

各レースでは，ステップごとに AI にレースの状況についての情報を与え，それに対する AI からの応答である加減速指示に従ってレースの状況を更新することを繰り返す． レース中に両プレイヤが同時に同じ位置を占めることはない．

ステップは 0 から 1 ずつ進み，両プレイヤ共にゴールまたは失格（3.6 節）すればレースは終了である． ただし，制限ステップ数に達してもゴールしないプレイヤがある場合，その時点でレースは終了し，ゴールに達していないプレイヤは失格となる．

3.1　考慮時間

AI が 1 レースで使える**考慮時間**の合計には，制限を設ける． 考慮時間はゲーム管理プログラムがプレイヤに情報を送信し終わってから，プレイヤからの応答を受信し終わるまでの実時間である． 考慮時間の合計が制限値を超えたプレイヤは失格となる．

3.2　プレイヤの状態と加減速指示

プレイヤは各ステップの開始時において**位置** (x, y) と**速度** $(v_\mathrm{x}, v_\mathrm{y})$ を持つ．

AI はステップごとに速度を変更する**加速度** $(a_\mathrm{x}, a_\mathrm{y})$ を指定する． $a_\mathrm{x}, a_\mathrm{y}$ は各々 $-1, 0, 1$ のいずれかである．

後述するコースアウトや衝突によって動けない場合を除き，次ステップのプレイヤの位置の座標は $(x + v_\mathrm{x} + a_\mathrm{x}, y + v_\mathrm{y} + a_\mathrm{y})$ となる． この位置を以下では次ステップにおける**予定位置**と呼ぶ． また，次ステップにおけるプレイヤの速度はコースアウトや衝突の有無に関わらず $(v_\mathrm{x} + a_\mathrm{x}, v_\mathrm{y} + a_\mathrm{y})$ となる．

3.3　動線とコースアウト

プレイヤの位置と次ステップにおける予定位置を両端とする線分を，このプレイヤの当該ステップでの**動線**と呼ぶ．

以下の場合，当該プレイヤは**コースアウト**を生じているという．

- 予定位置がコース外に出る，すなわち予定位置の座標 (x, y) が $0 \leq x < w$ かつ $0 \leq y$ を満たさない場合
- 動線が障害物と交差あるいは接触する場合

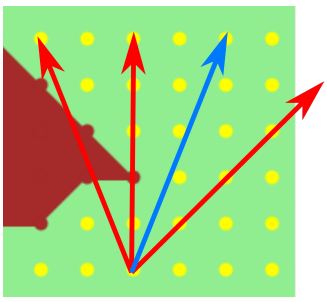

図 2: コースアウト
赤い動線はコースアウト

コースアウトしたプレイヤの次ステップでの位置は元のままになる． 速度についてはコースアウトしなかった場合と同じ加速度が適用される．

3.4　衝突と優先権

コースアウトがない場合に，両プレイヤの動線が交差あるいは接触する場合，**衝突**が生じたという．

衝突が生じた場合は，**優先権**を持つプレイヤの次ステップの位置は予定位置に，優先権のないプレイヤの位置は元のままになる． いずれのプレイヤについても，次ステップの速度は衝突しなかった場合と同じ $(v_\mathrm{x} + a_\mathrm{x}, v_\mathrm{y} + a_\mathrm{y})$ となる．

優先権を持つのは，位置の y 座標がより小さいプレイヤである． y 座標が同一である場合，位置の x 座標がより小さいプレイヤが優先権を持つ． ただし，動線が相手プレイヤの動作前の位置に達するか通過する場合は，優先権が相手プレイヤに移る．

両プレイヤとも互いに相手の位置に達するか通過するような動線を持つ場合は，両者とも優先権を失い，次のステップでの位置は元のままとなる．

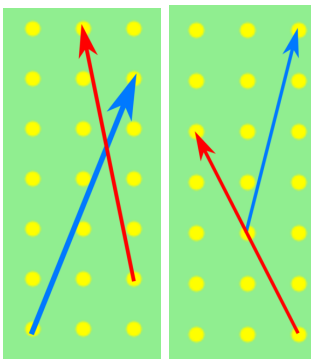

図 3: 衝突
いずれも青い動線のプレイヤが優先権を持つ

3.5 ゴールタイム

プレイヤがステップ s で座標 (x, y) の位置にあり，次ステップで位置 (x', y') に達し，$y' \geq l$ (l はコース長) であるとき，そのプレイヤはゴールしたものとされる． このときゴールタイムは $s+(l-y)/(y'-y)$ で与えられる．

コースアウトしたプレイヤ，あるいは衝突を起こし優先権を持たないプレイヤは，たとえ予定位置の y 座標がコース長以上であっても，次ステップの位置は元の位置のままとなるので，そのステップではゴールしない．[1]

ゴール時，相手プレイヤがゴールも失格もしていなければ，ゴールしたプレイヤをコースから取り除いた状態でレースを継続する．

3.6 失格

プレイヤは以下のいずれかを行った場合，失格となる．

- 制限ステップ数に達してもゴールできなかった（3 節冒頭）．
- 考慮時間の合計が制限値を超えた（3.1 節）．
- 4.2 節や 4.4 節の規定に反する出力を行った．

失格時，相手プレイヤがゴールも失格もしていなければ，失格したプレイヤをコースから取り除いた状態でレースを継続する．失格したプレイヤのゴールタイムは制限ステップ数の 2 倍とする．

なお，失格の範囲は当該レース内に限られ，次以降のレースには通常通り参加できる．

4 AI の動作

AI は初期化時にレース全体に関する情報を入力し，初期化終了を示す応答を出力する． ステップごとにはレース状況の情報を入力し，応答として加速度指示を出力する．

入力はただひとつの空白あるいは改行で区切った 10 進整数の並びである．負の数にはマイナス符号を前置する．初期化時の入力およびステップごとの入力の最後は必ず改行を置く．

AI の応答出力も空白あるいは改行で区切った 10 進整数の並びで，負の数にはマイナス符号を前置する． **出力の最後には必ず改行を置く．**

[1] ゴール上やゴールより先に障害物が存在することもあるので注意されたい．

4.1 初期化時の入力

初期化時の AI の入力は以下の項目がこの順に並ぶものである．各項目の直後には改行が置かれ，1 つの項目（1 行）に含まれる複数の整数は**ひとつの空白文字**で区切られる．

考慮時間 t_{given} このレースで使える考慮時間をマイクロ秒単位の整数値として．

制限ステップ数 $s_{\max}$ ゴールまでのステップ数の上限を整数値として．

コースサイズ コースの幅 w および長さ l を，ふたつの整数値として．

視界 d 視界に入る y 座標の範囲を整数値 d として．位置 (x, y) にあるプレイヤを制御する AI をに与えられる相手プレイヤや障害点の情報は，y 座標が $y - d$ から $y + d$ の範囲 (両端を含む) にあるもののみが与えられる．

初期化時の入力の形式を以下に示す．

t_{given}↵
$s_{\max}$↵
w l↵
d↵

4.2 初期化時の出力

AI は初期化が終了したことを整数 0 ひとつ及び改行で伝える．

0↵

4.3 ステップごとの入力情報

ステップごとの AI の入力は以下の項目がこの順に並ぶものである．各項目の直後には改行が置かれ，1 つの項目に含まれる複数の整数は**ひとつの空白文字**で区切られる．

ステップ s 当該のステップの値を整数ひとつで与える．

残り考慮時間 t_{left} このレースで使える考慮時間の残りをマイクロ秒単位の整数値として．

自プレイヤの状態 当該の AI が制御するプレイヤの位置の x 座標と y 座標，速度の x 成分 v_{x} と y 成分 v_{y} の 4 整数．

相手プレイヤの状態 相手プレイヤの位置の x 座標 x_{e} と y 座標 y_{e}，速度の x 成分 $v_{\mathrm{x_e}}$ と y 成分 $v_{\mathrm{y_e}}$ の 4 整数．ただし，相手プレイヤの位置が視界の外であれば，y 座標は -1，他は 0 になる．

障害点 視界内の各格子点が障害点であるか否かの情報で，$(2 \times d + 1) \times w$ 個の整数からなる． 視界内の $y - d$ から $y + d$ までの各 y 座標に対して w 個の整数 $o_{0,y}, o_{1,y}, \ldots, o_{w-1,y}$ が，y 座標の順に与えられる． $o_{x,y}$ が 1 ならば格子点 (x, y) は障害点であり，0 ならばそうではない． ただし，$y < 0$ であるコース外の格子点については 1 が与えられる．なお障害点にのみ，$o_{w-1,y}$ と $o_{0,y+1}$ の間には空白文字でなく改行が区切りとして用いられる．すなわち $2 \times d + 1$ 行 w 列が入力として与えられる．

ステップごとの AI の入力の形式を以下に示す．

s↵
t_{left}↵
x ␣ y ␣ v_{x} ␣ v_{y}↵
x_{e} ␣ y_{e} ␣ $v_{\mathrm{x_e}}$ ␣ v_{ye}↵
$o_{0,y-d}$ ␣ $o_{1,y-d}$ ␣ $\cdots$ ␣ $o_{w-1,y-d}$↵
$o_{0,y-d+1}$ ␣ $o_{1,y-d+1}$ ␣ $\cdots$ ␣ $o_{w-1,y-d+1}$↵
$\cdots$
$o_{0,y+d}$ ␣ $o_{1,y+d}$ ␣ $\cdots$ ␣ $o_{w-1,y+d}$↵

4.4 ステップごとの出力

AI は当該ステップの加速度 $(a_{\mathrm{x}}, a_{\mathrm{y}})$ をふたつの整数で指定する．各整数はひとつ以上の空白で区切り，最後に改行を置く．以下にステップごとの出力の形式を示す．ただし空白は 2 つ以上あってもよい．

a_{x} ␣ a_{y}↵

4.5 入出力例

右に実際の AI への入力例（黒字）および AI からの出力例（赤字）を示す．これは，初期化から最初の 2 ステップまでの入出力である．

最初の 4 行は初期化時の入力であり，考慮時間，制限ステップ数，コースの幅，長さおよび視界がそれぞれ $20,000\,\mu\mathrm{s}$，100，15，100，8 であることを示している．

この入力を受け取ったら AI は 0 と改行をを出力しなければならない．

以降はステップごとの入出力である．まず最初のステップについて，現在のステップ数と残り考慮時間に対応する入力値がそれぞれ 0，19981 となっていることから，この AI は初期化に $19\,\mu\mathrm{s}$ かかり，$19,981\,\mu\mathrm{s}$ の考慮時間が残されていることがわかる．

次の 2 行には自プレイヤと相手プレイヤの状態が示されている．この例では，自プレイヤの現在の位置 (x, y) が $(5, 0)$ で速度 $(v_{\mathrm{x}}, v_{\mathrm{y}})$ は $(0, 0)$ であること，および相手プレイヤの現在位置 $(x_{\mathrm{e}}, y_{\mathrm{e}})$ が $(9, 0)$ で速度 $(v_{\mathrm{x_e}}, v_{\mathrm{ye}})$ は $(0, 0)$ であることが確認できる．

次の 17 行（視界が 8 なので $(8 \times 2 + 1)$ 行）は視界内の各格子点が障害点か否かを示している．この例では $y < 0$ におけるすべての格子点が障害点であること，および前方正面に大きな障害物があることがわかる．

AI はこの入力を受け取ったら，加速度 $(a_{\mathrm{x}}, a_{\mathrm{y}})$ を出力する必要がある．この例では $(-1, 1)$ を出力している．すると，また次のステップの入力が第 1 ステップと同じ形式で与えられるので，AI はまた加速度を出力する．これがレース終了まで繰り返される．

```
20000
100
15 100
8
0
0
19981
5 0 0 0
9 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 1 0 0 0 0 0
0 0 0 0 0 0 1 1 1 1 0 0 0 0 0
0 0 0 0 0 1 1 1 1 1 1 0 0 0 0
-1 1
1
19955
4 1 -1 1
8 1 -1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 1 0 0 0 0 0
0 0 0 0 0 0 1 1 1 1 0 0 0 0 0
0 0 0 0 0 1 1 1 1 1 1 0 0 0 0
0 1
```

4.6　情報の記憶

ゲーム管理システムは，各ステップについて出力を終了した AI の動作を一時停止し，次ステップの情報を与える際に動作を再開させる． このため，ステップの間に計算を進めることはできないが，変数値などの実行コンテキストは 1 レースの間保持し続けることができる．

AI はファイル出力やネットワークアクセスができない環境で動作し，ゲーム管理システムはレース毎に AI を初期状態から立ち上げ直す． このため，AI は複数のレース間で情報を受け渡すことができない．

以上