igda international game developers association Japan chapter

ゲームAI連続セミナー

第3回事前資料Ⅰ

「迷路を解くAIを使ってゲームを作る」

企画　＆　プログラマー向け

三宅　陽一郎
y_miyake@fromsoftware.co.jp

2007.5.8

「迷路を解くAIを使ってゲームを作る」の主旨

(1) 前回のアンケートの感想から、事前資料を独立した資料として製作することにしました。講演資料だと中途のものしか準備できないからです。

(2) 事前に読まれることで、第3回のセミナーの内容をよりよく理解できるように製作しています。

(3) この資料は、「迷路を解くAI」を作ることを通して第1回から第3回のセミナーの内容を理解します。

(4) Q&A形式になっているので、考えながら気軽に読んでください。Q1～10まであります。巻末には付録としてパス検索アルゴリズムをまとめてあります。

「迷路を解くAIを使ってゲームを作る」の主旨

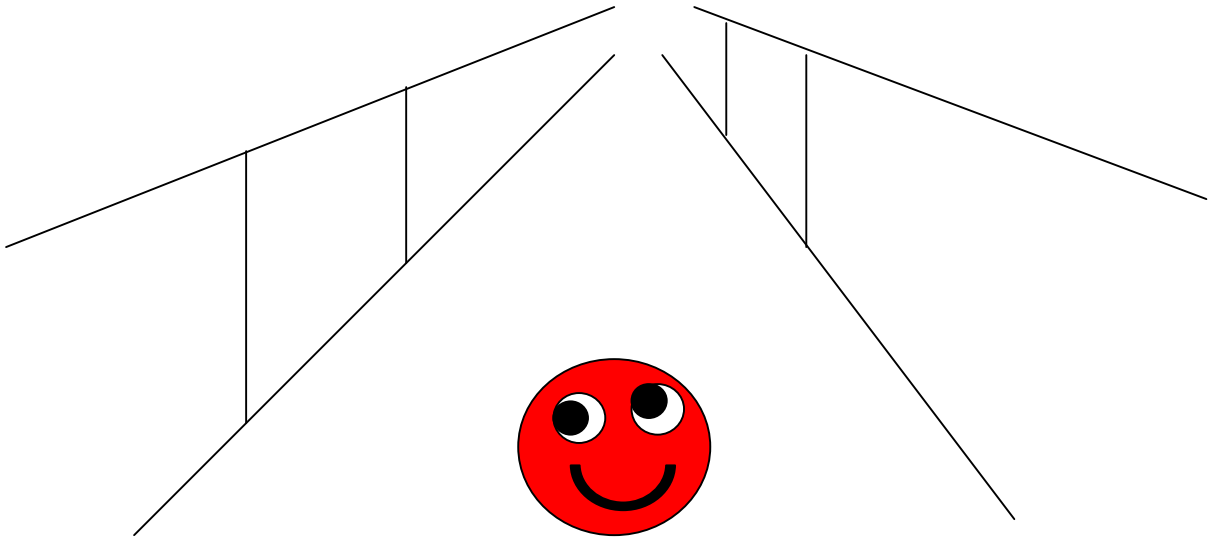
(5) この資料を作っている間に、IGDA日本の懇親会行くと
「研修のテキストみたいなものを作れないか？」という話題が
ありました。

そこで、各Q＆Aごとに「課題」を用意することにしました。
課題は、プログラマー向けと企画向けを用意してあります。
飛ばして読んでも問題ありません。

課題などという大層な名前がついていますが、
プログラマーと企画が和気あいあいと
「あーしようか、こうしようか、あーでもない、こーでもない」と、
話し合いながら、AIを使ったミニゲームを作って行く時間を提供
することを目的にしています。

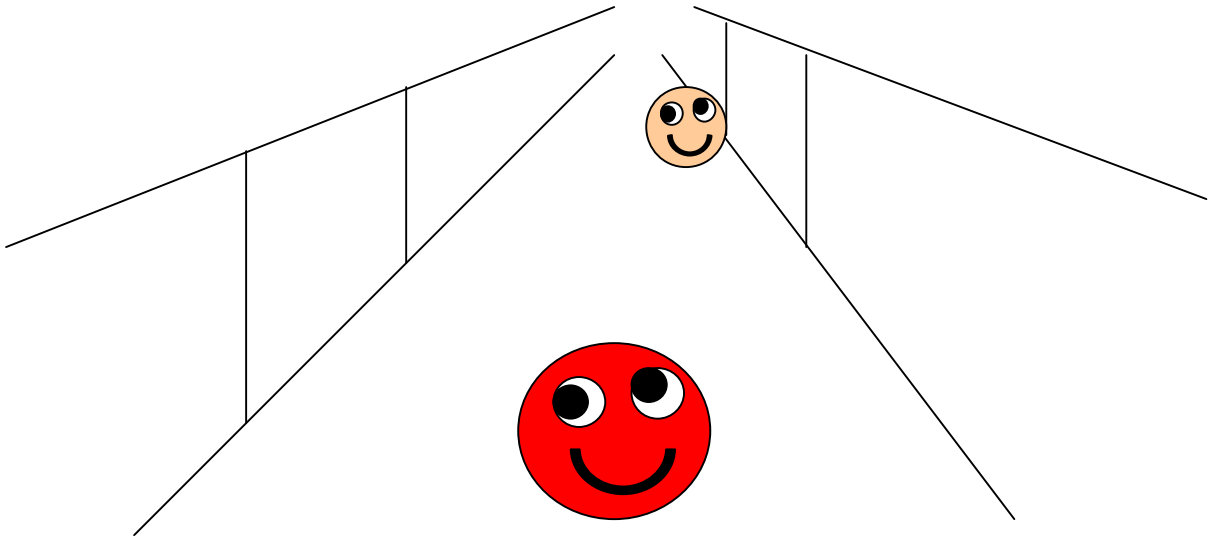
この資料の目標

迷路の中を自分で考え、
自由に移動できるAIたちを作って、
それを使ったゲームを考えよう！



ポイント

以下の解説は
「開発者が事前に行動を仕込むのではなく、
ゲーム内でAI自身に考えさせる」
という点に着目するとわかりやすいです！



作成工程

Step1 *NPCが迷路の中を自由に移動できるようにする。*

Step2 *NPCが自分で行動を考えられるようにする。*

Step3 *NPCたち互いにが協調できるようにする。*

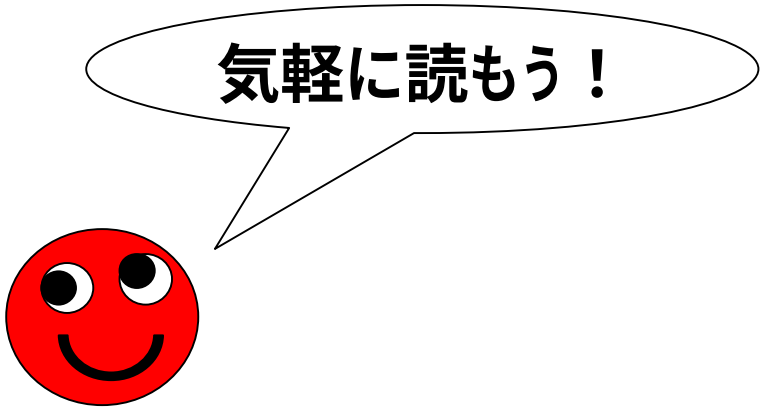
Step4 *NPCたちを使ってゲームを作ってみる。*

ステップを踏むごとにNPCは賢くなりますが、
それぞれのステップごとにどんなゲームデザインが
考えられるか考えてみよう！

設定は説明のため簡単な迷路とゲームルールにしますが、
任意に変えて考えてみてください

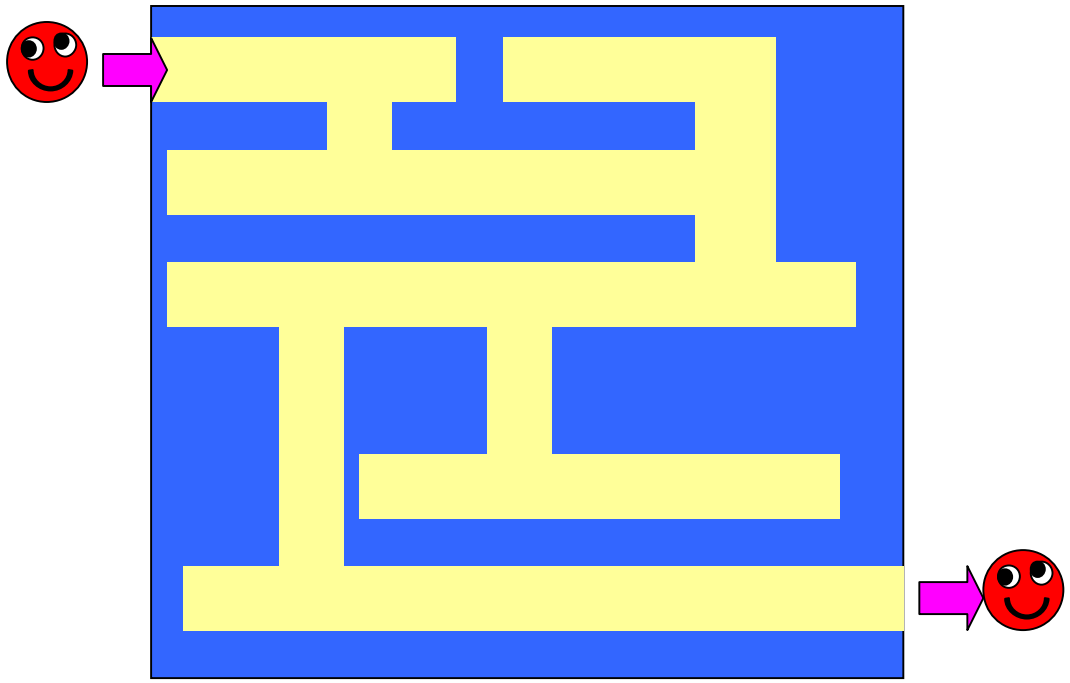
Step1

ＮＰＣが迷路の中を自由に移動できるようにする。



NPCの自由な移動　①

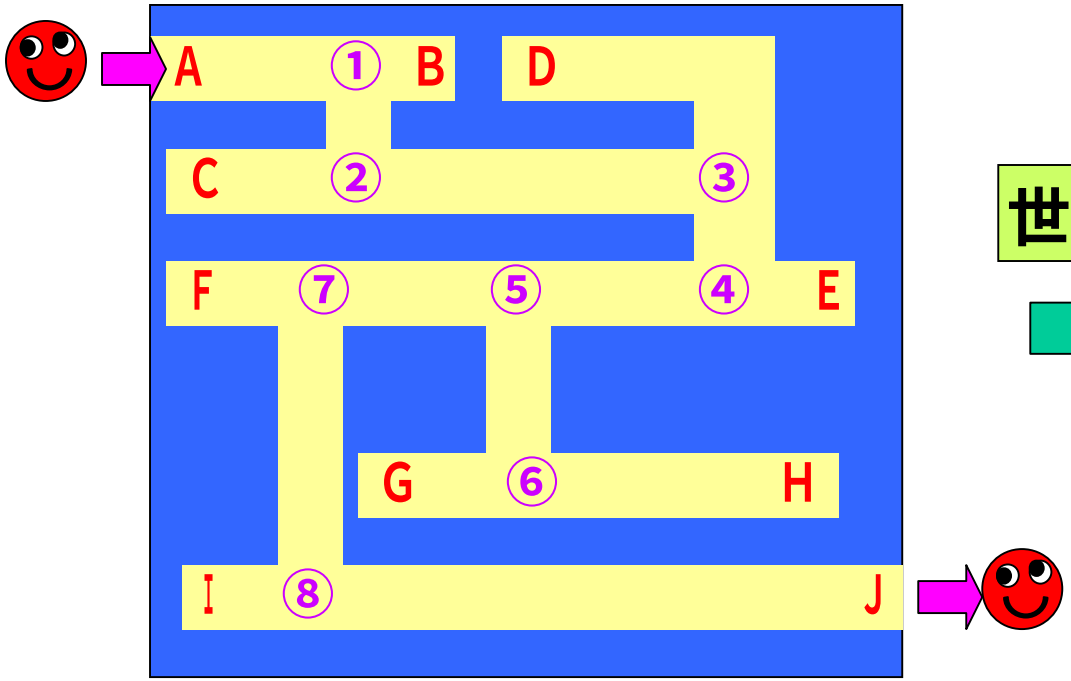
Q1：与えられた迷路を抜けることが出来るAIを考えてみよう！
どのように実装すればよいか？



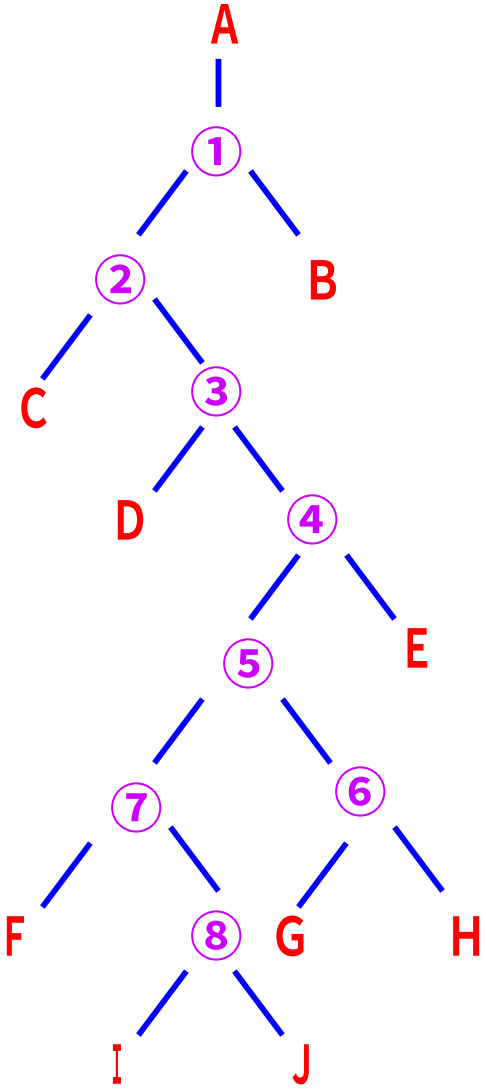
NPCの自由な移動 ①

A1：迷路をグラフとて表現する。

「迷路内の移動＝グラフ上の移動」と問題が簡単になる。
グラフ検索のアルゴリズムでAIを動かすことが可能になる。



世界表現

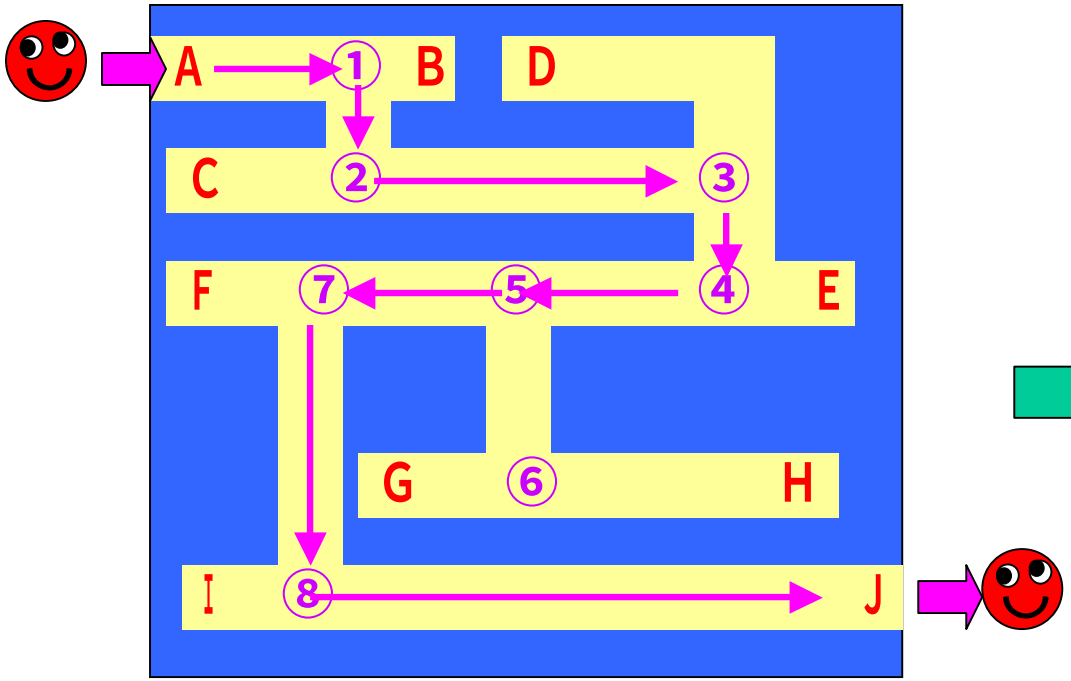


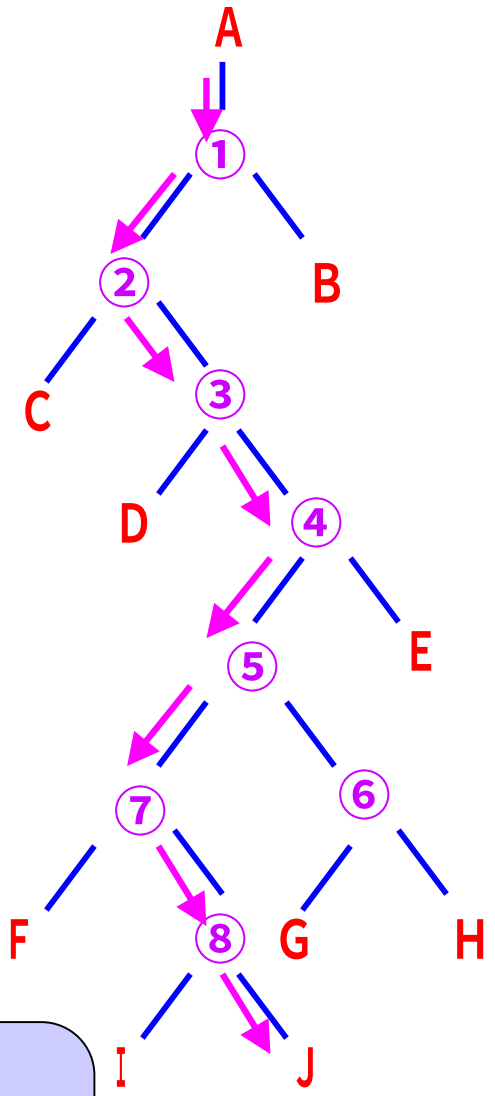
最大3分岐の合流しない迷路は
2分木(binary tree)と等価である

このように、ゲーム世界をAIの制御のために表現したものを世界表現という。

NPCの自由な移動　①

動作例





工程１：企画が迷路を製作する
工程２：プログラマーが世界表現（ここでは２分木）を実装する。
工程３：　グラフ検索アルゴリズムを実装する。
ゲーム中：スタート地点からゴール地点までパス検索を実行し、
ＮＰＣにパスにパスをたどらせる。

NPCの自由な移動　課題　①

プログラマー

実際に、与えられた迷路に対して二分岐ツリーを実装し、
パス検索によってＮＰＣをスタート地点から出口まで導いてみよう。

全幅検索（ＢＦＳ）、深さ優先検索（ＤＦＳ）など検索アルゴリズムを
試してみよう。

迷路を自動生成するプログラムを作ってみよう。

企画

迷路をAIの為のデータとして表現し、グラフ検索のプログラムにより、
与えられた迷路を最短で抜けるAIを実現できる。

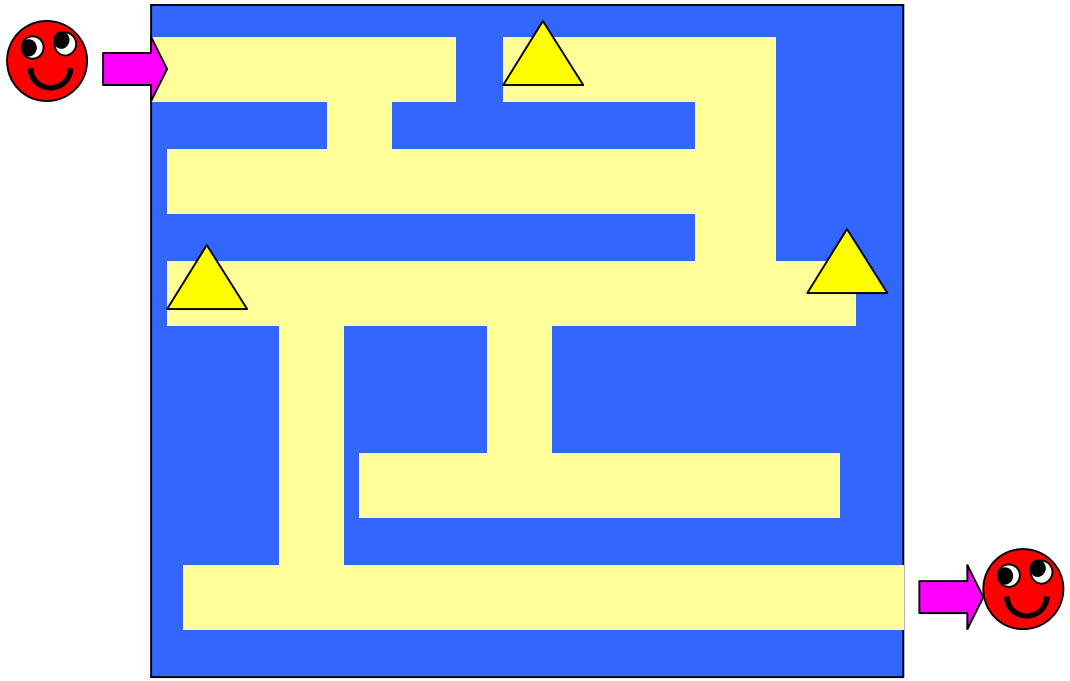
これだけの能力から、どんなゲームが設計できるだろうか？

また、迷路を自動生成できるとすれば、どんなゲームデザインが
考えられるだろうか？

NPCの自由な移動　②

Q2：金塊を集めて迷路を脱出するゲームを考えてみよう！

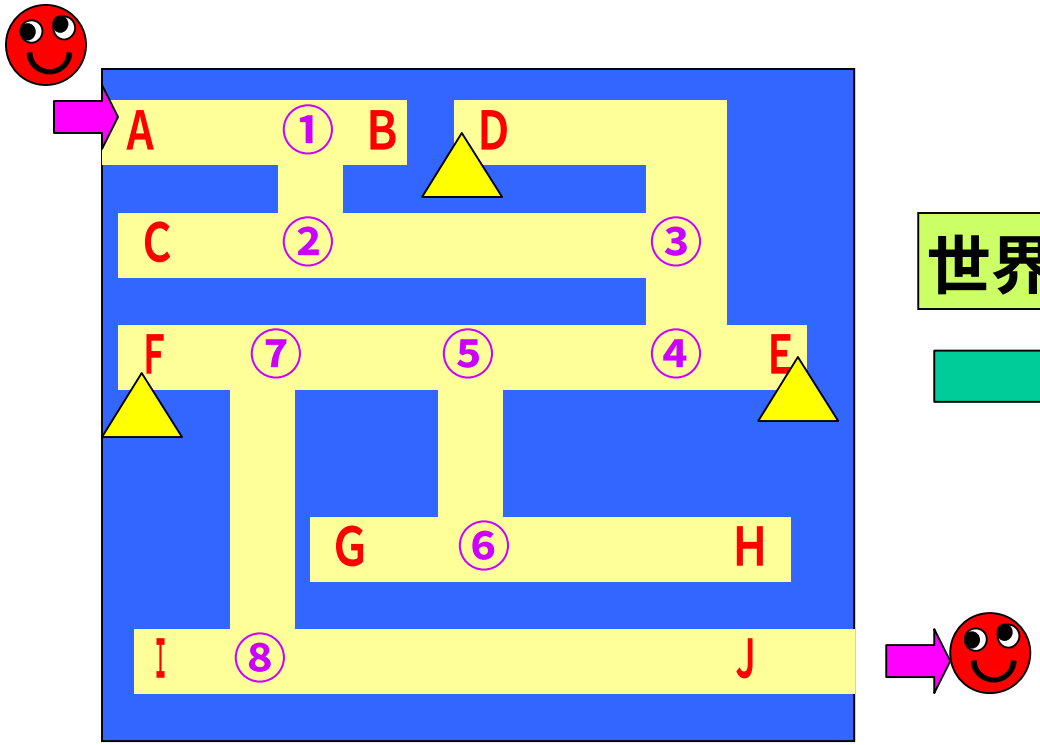
設定：　3つの金塊を集めて迷路を脱出する。



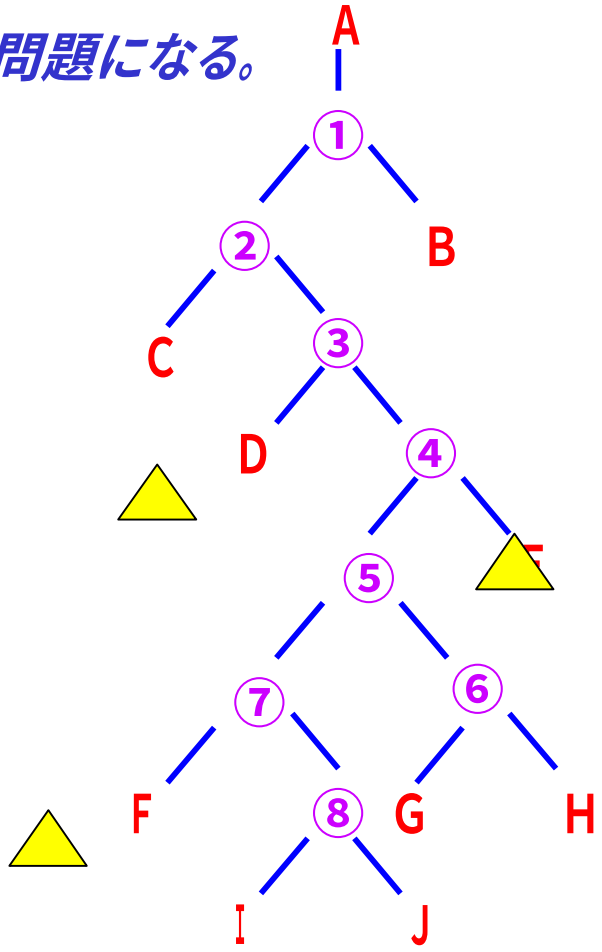
NPCの自由な移動 ②

A2：世界表現の上に情報を埋め込む

ノードに情報を埋め込むことで、条件付きパス検索の問題になる。



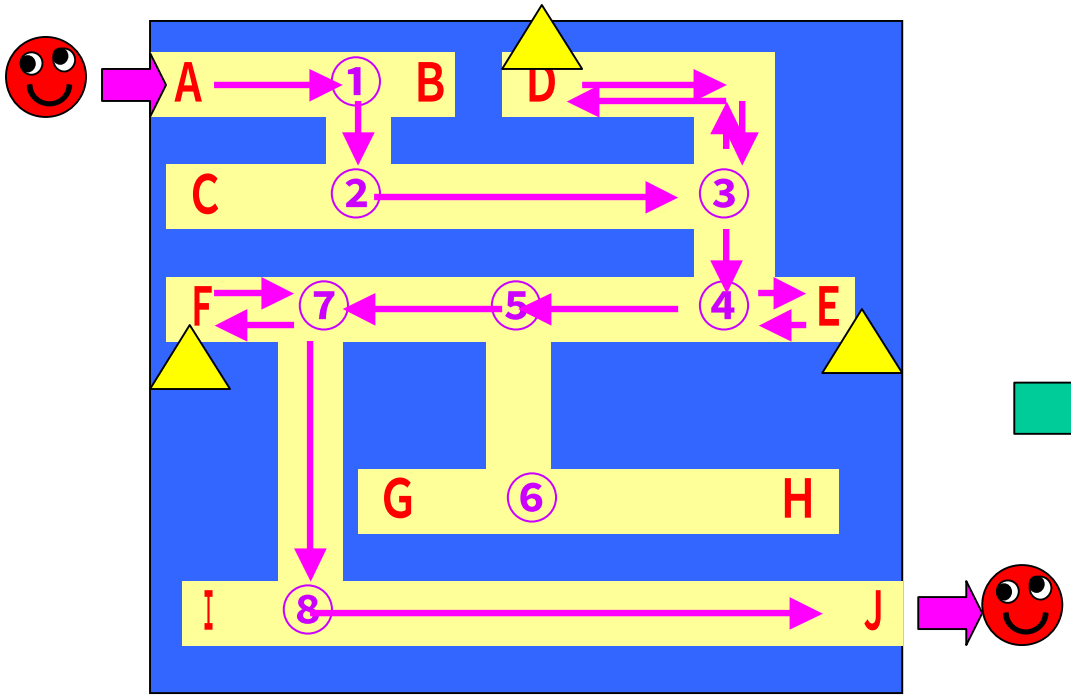
世界表現

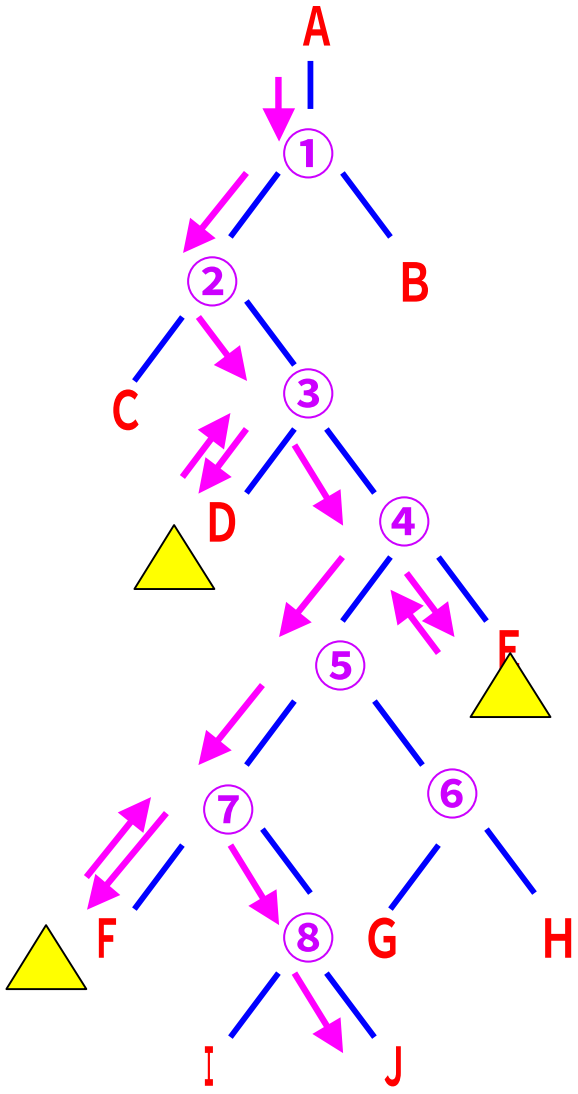


作成した世界表現に情報を埋め込んで行くことで、
ＮＰＣにより賢明な能力を付与することが出来る。

NPCの自由な移動　②

動作例





NPCの自由な移動　課題　②

プログラマー

各ノードに情報を埋め込めるようにデータ表現を設計してみよう。

3つの金塊の情報を、ランダムに末端のノードに配置できるようにしよう。

3つの金塊を取って迷路を脱出する最短経路を導く
アルゴリズムを実装しよう。

企画

世界表現に情報を埋め込むことで、例えば、ゲーム起動ごとに
ランダムに金塊の位置を変えても、最短経路でそれらを集めて
迷路を抜けるＮＰＣを実現できる。

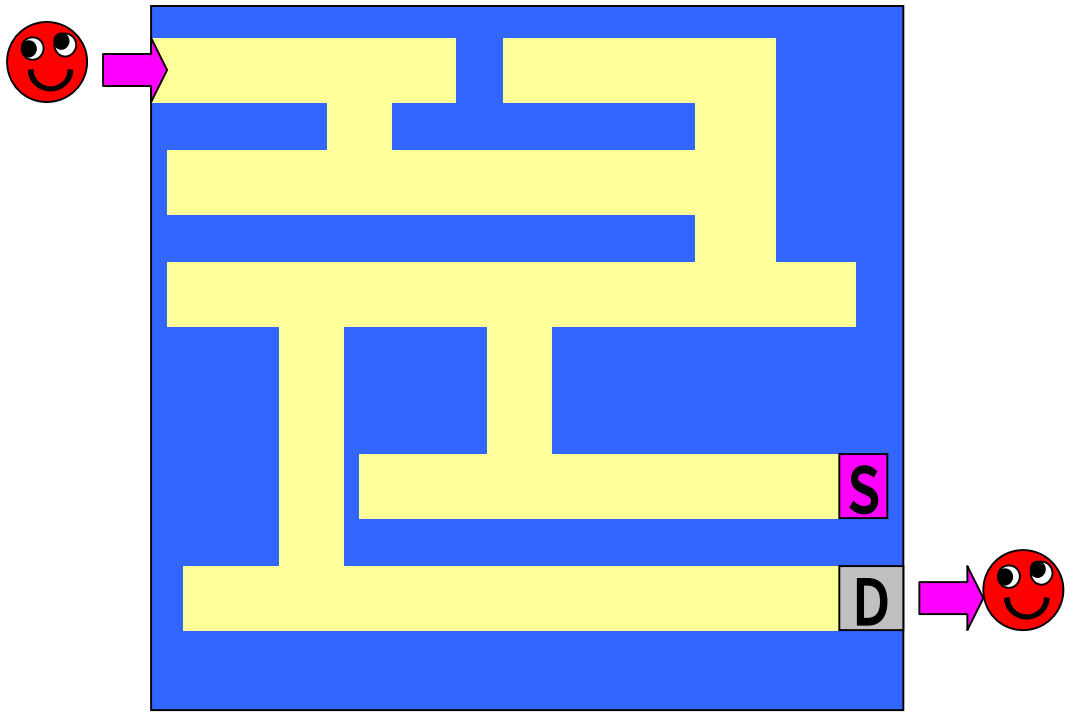
このＮＰＣを使って、どんなゲームが考えられるだろうか？

また、金塊以外にもいろいろな設定を加えて、
ＮＰＣにさせることの幅を広げてみよう。

NPCの自由な移動　③

Q３：AIに基本的な情報を与えよう！

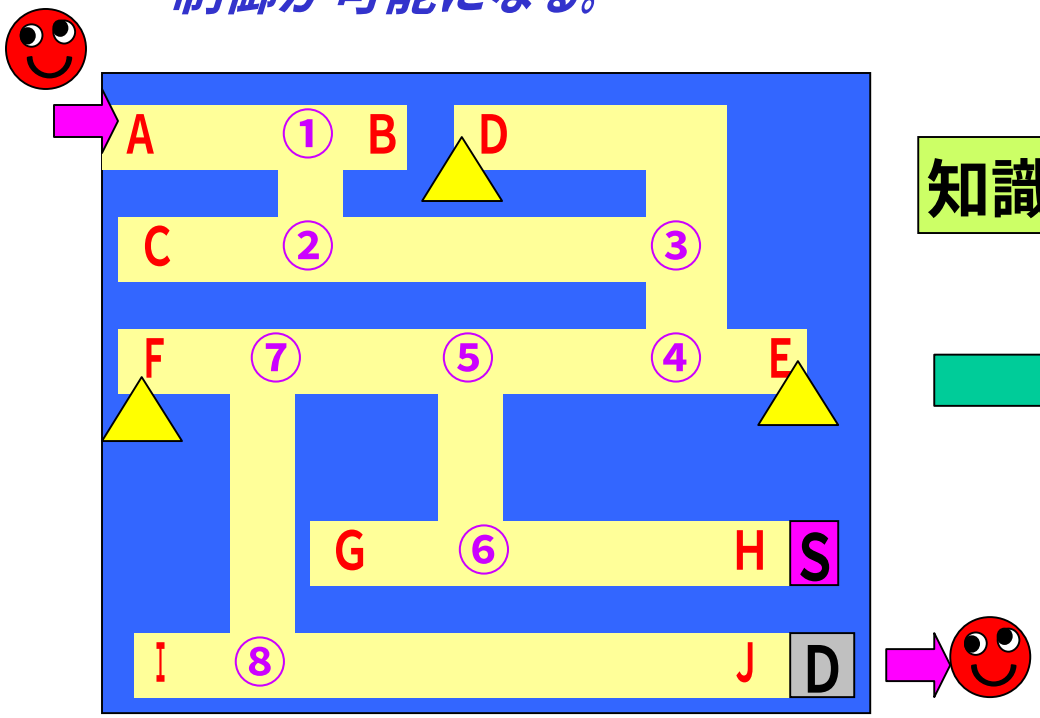
設定：　スイッチ（S）を押すと迷路を脱出する扉（D）が開く。



NPCの自由な移動　③

A３：「スイッチを押せばドアが開く」という知識を表現する

AIが使える形で知識を与えることで、AIは知識をベースとした制御が可能になる。



知識表現

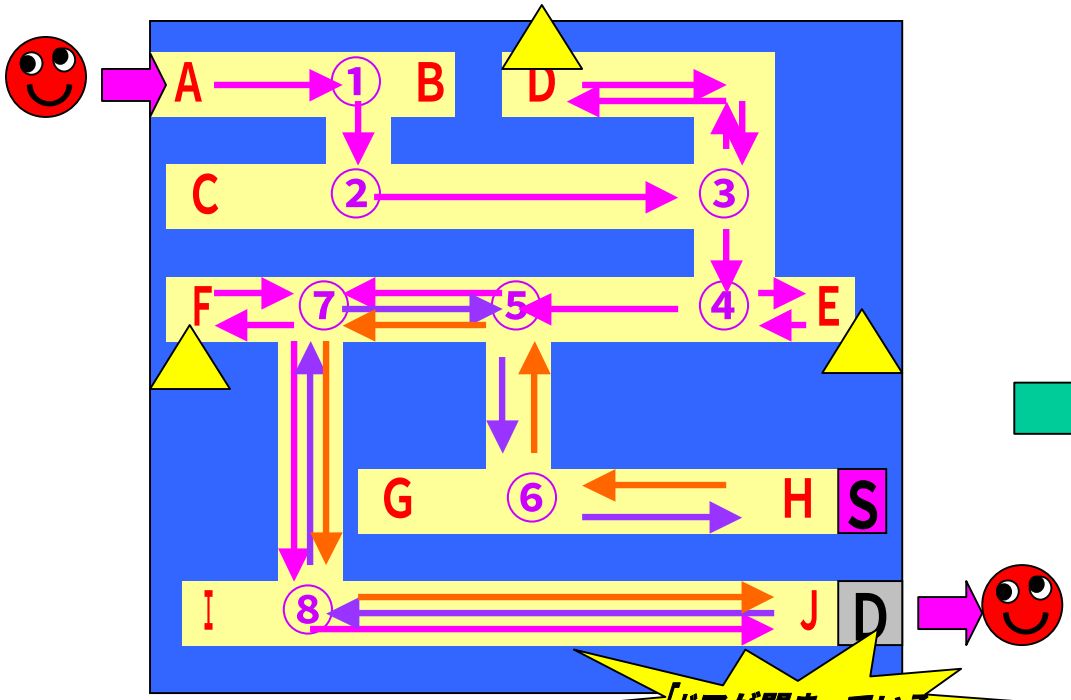
Dを「開いている」状態にするにはSを押せ。

if(D is closed) push(S)

AIが解釈するべき情報はAIが利用できる形で与えることで、初めてAIに知識に基づいた行動を取らせることが出来る。これを知識表現という。（人工知能の基本）

NPCの自由な移動　③

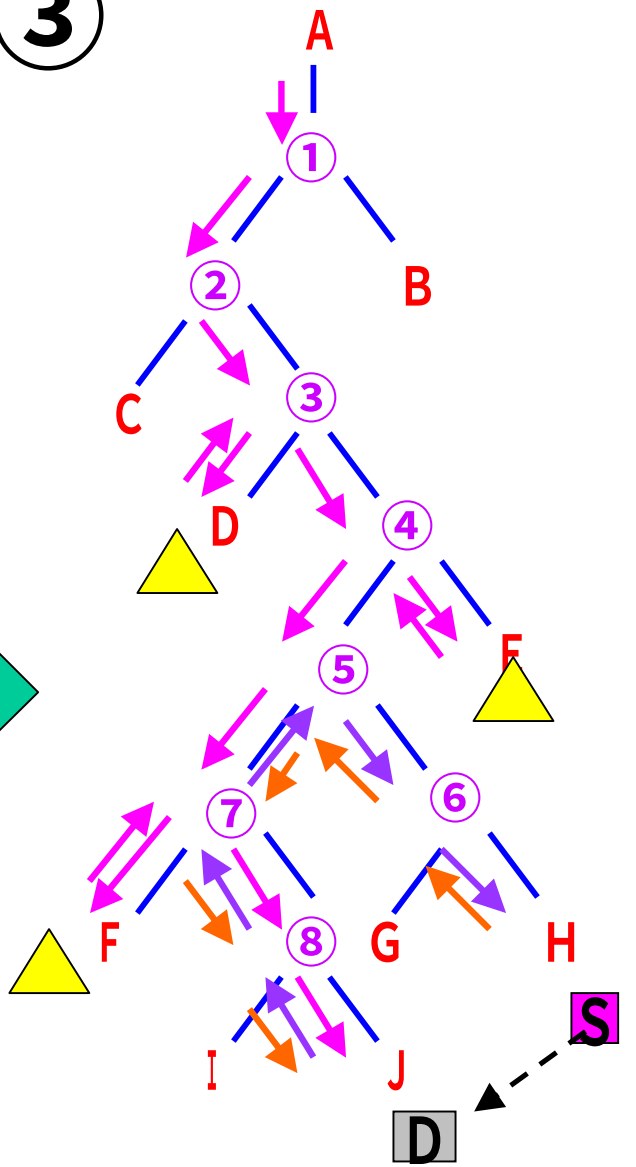
動作例



「ドアが閉まっている」ことを目撃する

if(D is closed) push(S)

スイッチSを押しに行こう！



知識表現

NPCの自由な移動　課題　③

プログラマー

「スイッチを押せばドアが開く」という知識をＮＰＣが利用できる形で実装してみよう。

実装した情報を適切な場所、タイミングで発動させるように、実装してみよう。

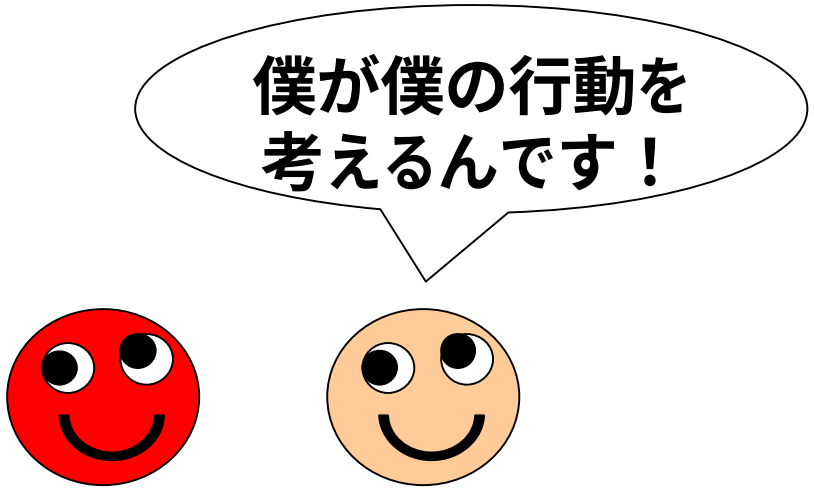
企画

知識表現を用意することによって、ＮＰＣはゲーム内の様々なオブジェクトを適切に使うことが出来るようになる。

この例で、いくつかのギミックを加えることを考えてみよう。

Step2

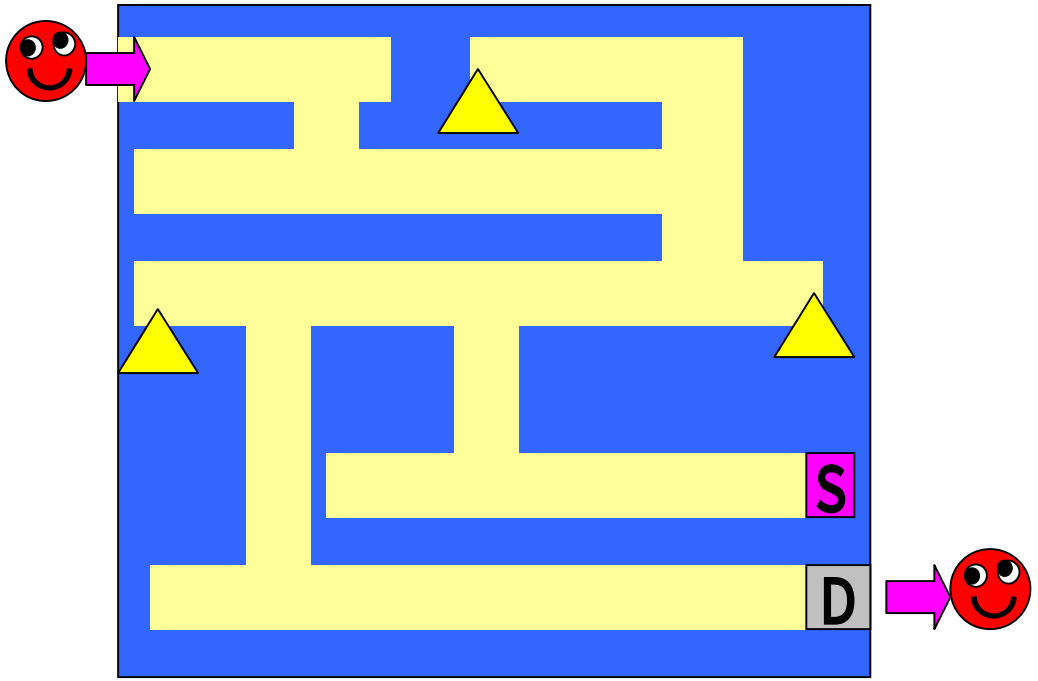
ＮＰＣが自分で行動を考えられるようにする。



自分で行動計画を立てるNPC　①

Q４：「金塊を取ってスイッチを押して迷路を抜ける」ように
ＮＰＣに計画を立てさせるにはどうすればいいだろうか？

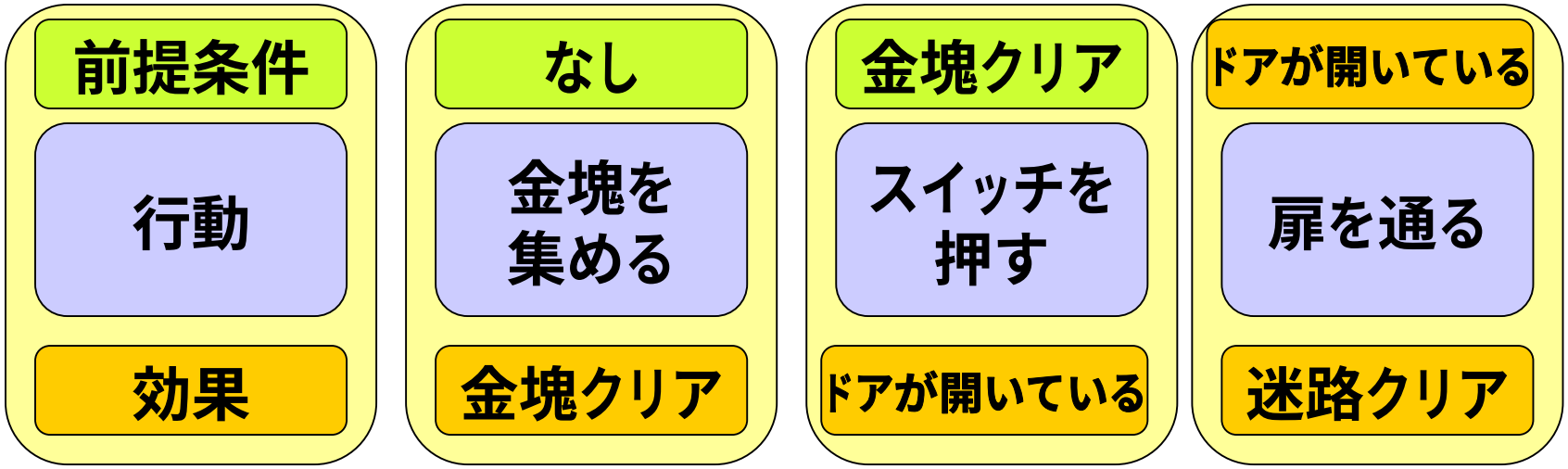
設定：　３つの金塊を集めてから、スイッチ（S）を押すと迷路を脱出する扉（D）が開く。



行動計画を立てるNPC ①

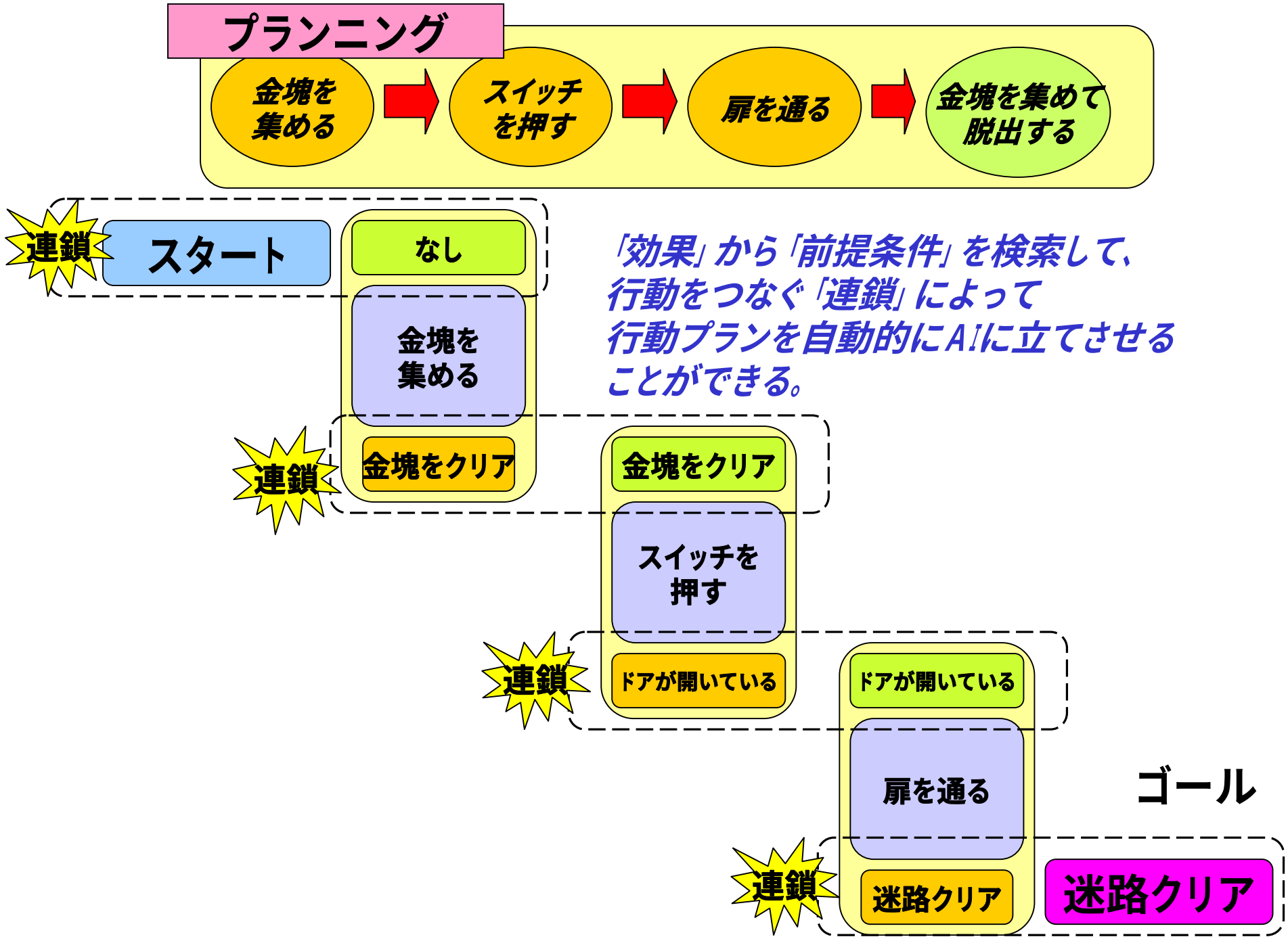
A4：「プランニング」を用いよ

「プランニング」とは、AIが未来の行動のプランを自ら作成する能力のことである。詳しくは第2回テキストを見て頂くとして、そのための下準備が必要である。ここでは、連鎖によるプランニングの準備を解説しよう。



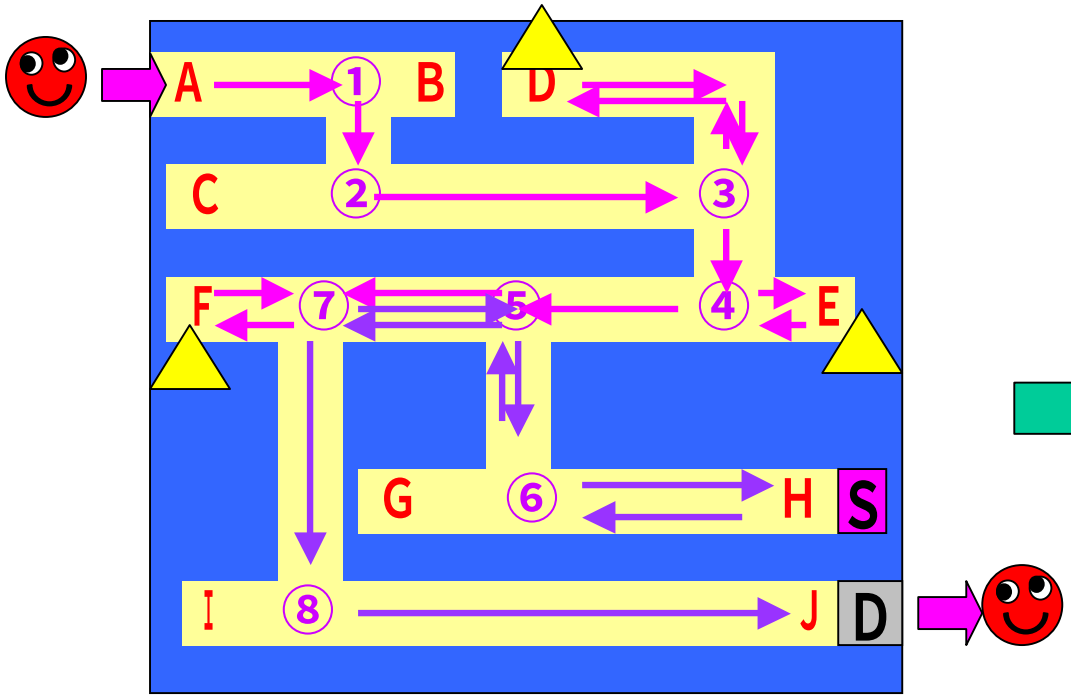
前提条件 … 行動をするために必要な条件
行動 … 実際のアクション
効果 … 行動による結果

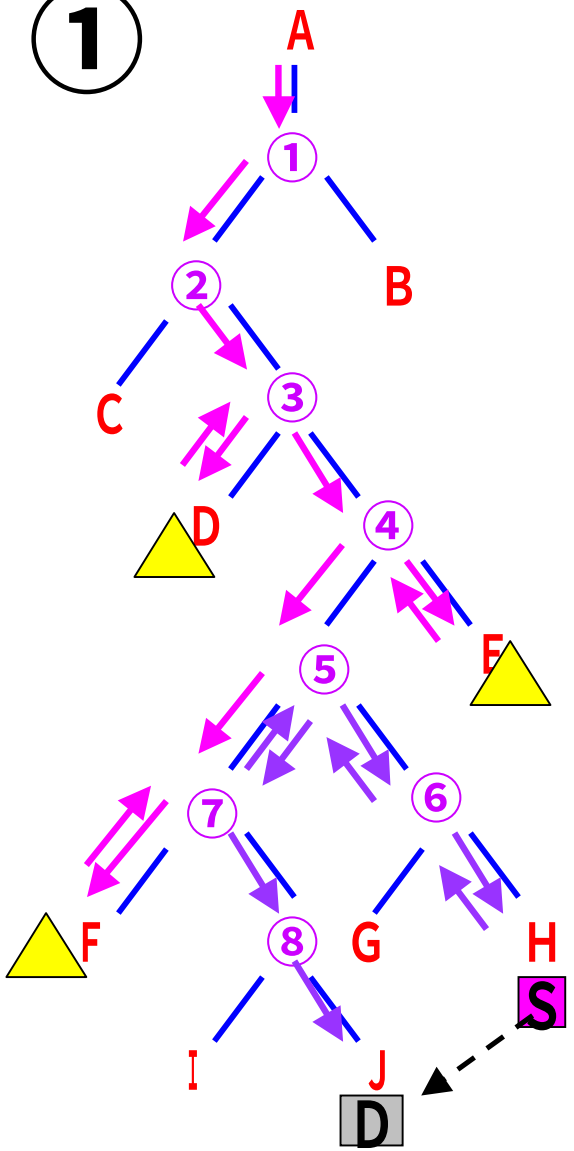
このように、「行動のデータセット」を作っておくと…


プランニング
金塊を集める
スイッチを押す
扉を通る
金塊を集めて脱出する
連鎖
スタート
なし
金塊を集める
金塊をクリア
連鎖
金塊をクリア
スイッチを押す
ドアが開いている
連鎖
ドアが開いている
扉を通る
迷路クリア
連鎖
ゴール
迷路クリア
「効果」から「前提条件」を検索して、
行動をつなぐ「連鎖」によって
行動プランを自動的にAIに立てさせる
ことができる。

行動計画を立てるNPC ①

動作例





行動計画を立てるNPC　①

プログラマー

「連鎖によるプランニング」についての技術を調査して、
実際に実装してみよう。

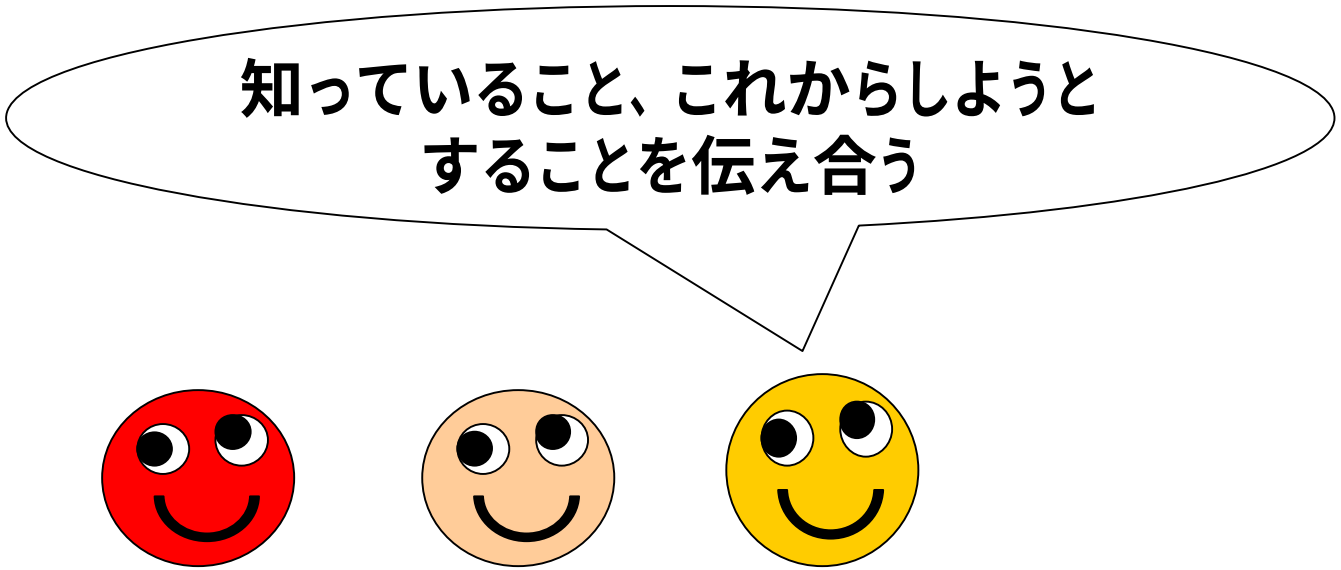
企画

「連鎖によるプランニング」は、行動のセットを増やすと、
多彩な行動プランのバリエーションが自動的に形成される。

行動のセットを追加して、より多彩な行動が可能なようにしてみよう。

Step3

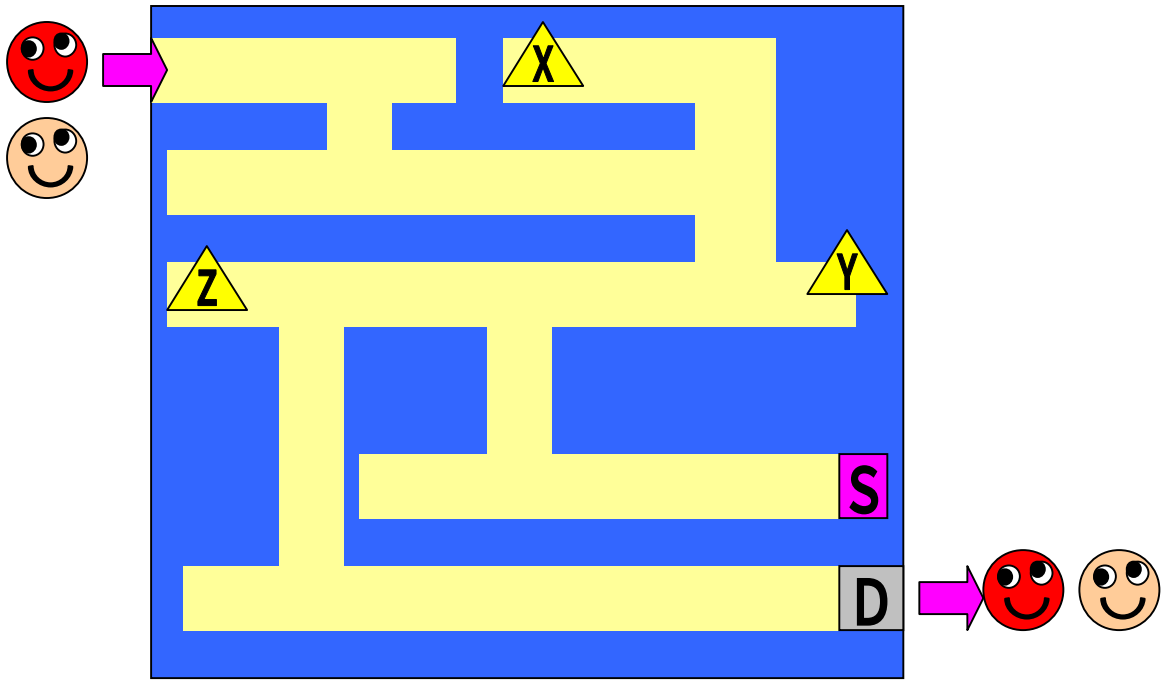
ＮＰＣたち互いにが協調できるようにする。



協調するAIを作る　①

Q5：2体のNPCが「金塊を取ってスイッチを押して迷路を抜ける」ようにチームとして行動させるにはどうすればいいだろうか？

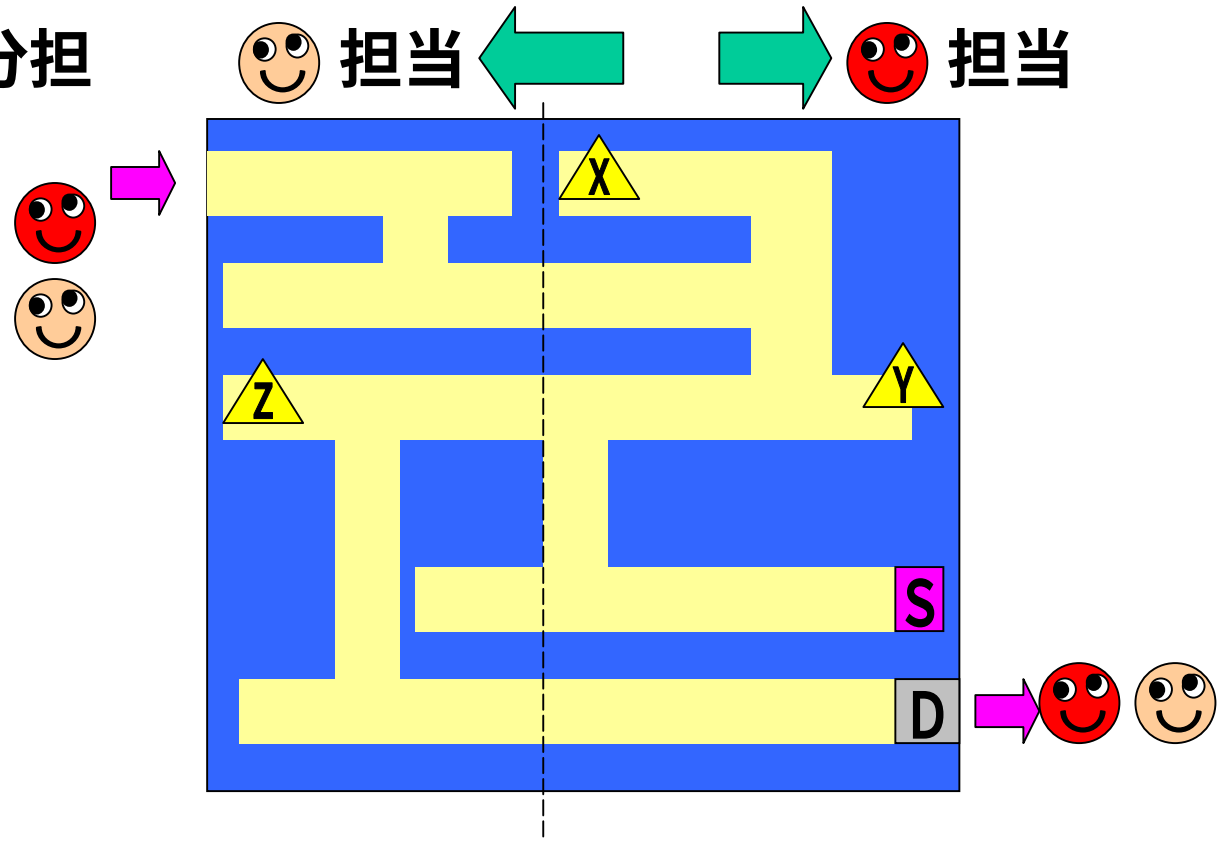
設定：　3つの金塊を集めてから、スイッチ（S）を押すと迷路を脱出する扉（D）が開く。



協調するAIを作る　①

A5：NPCを協調させる

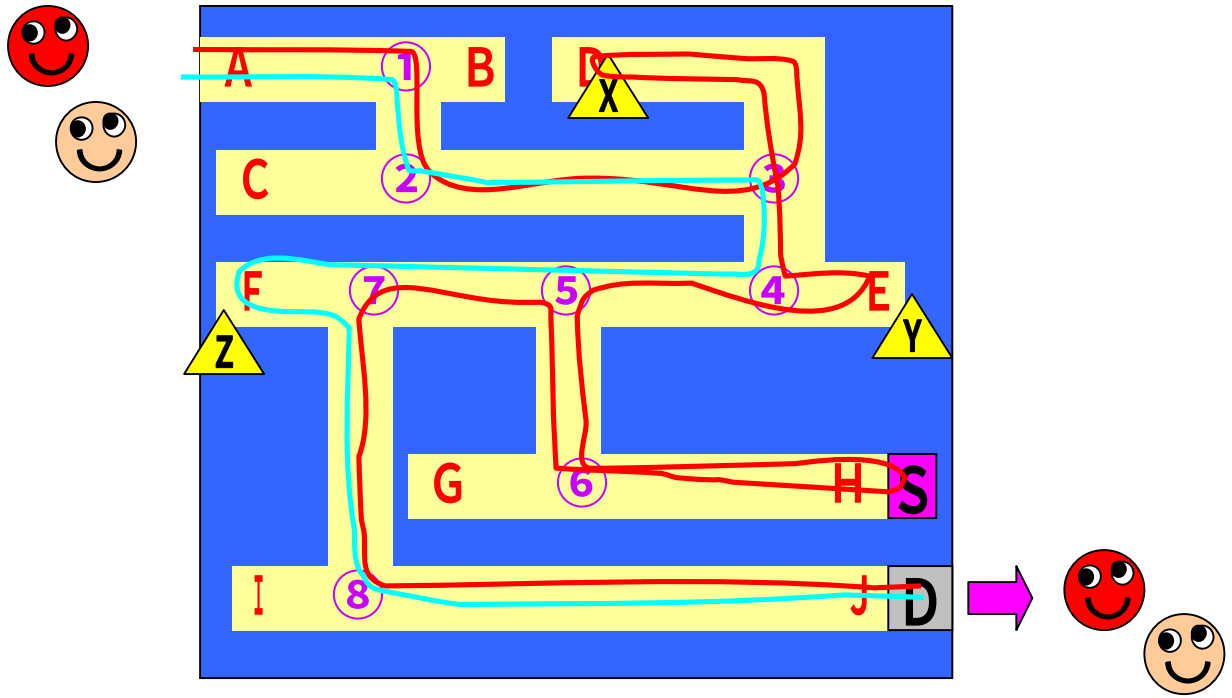
①領域による分担



2者の間であらかじめ領域を決めて実行する

行動計画を立てるNPC　①

動作例

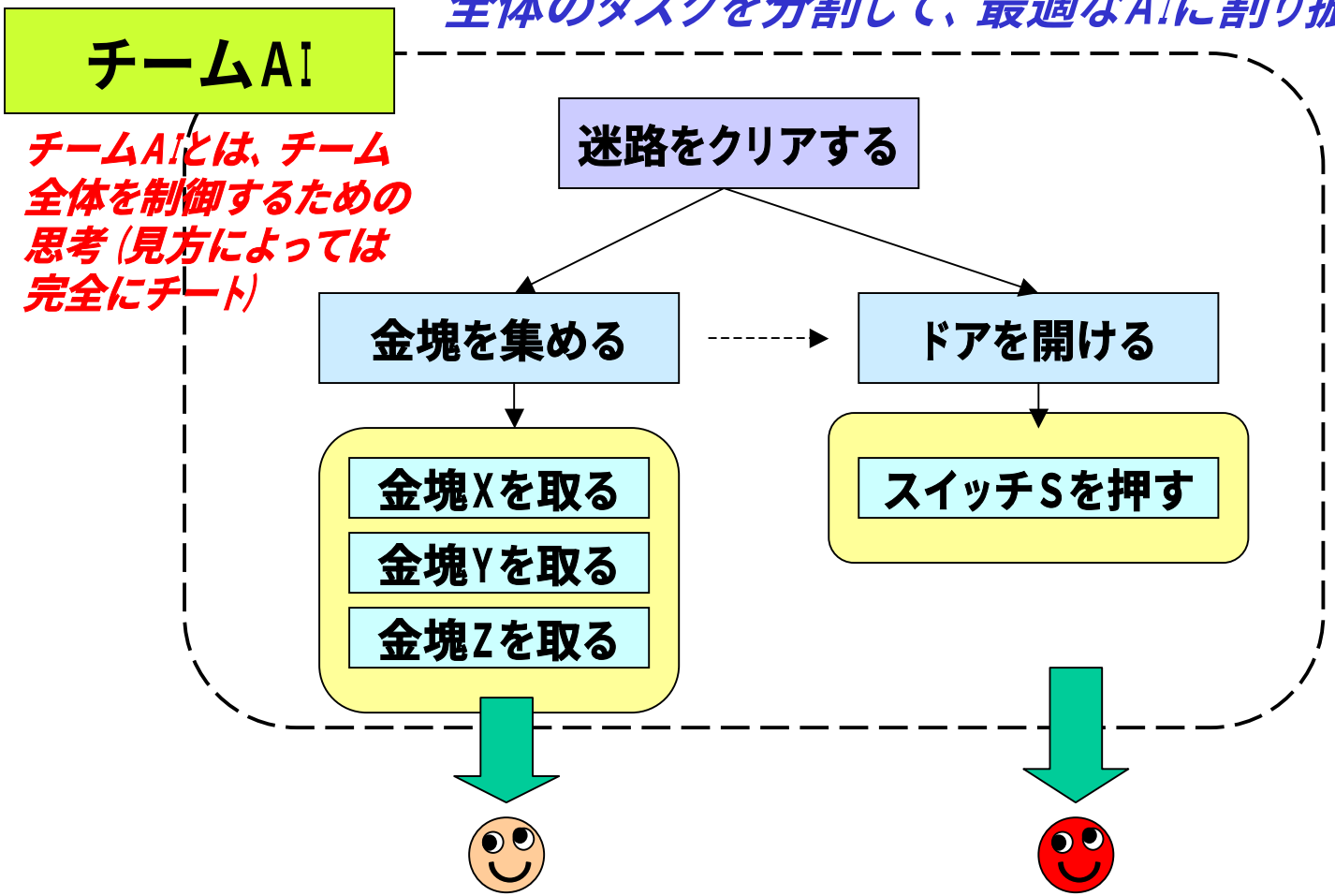


協調するAIを作る　①

A5：NPCを協調させる

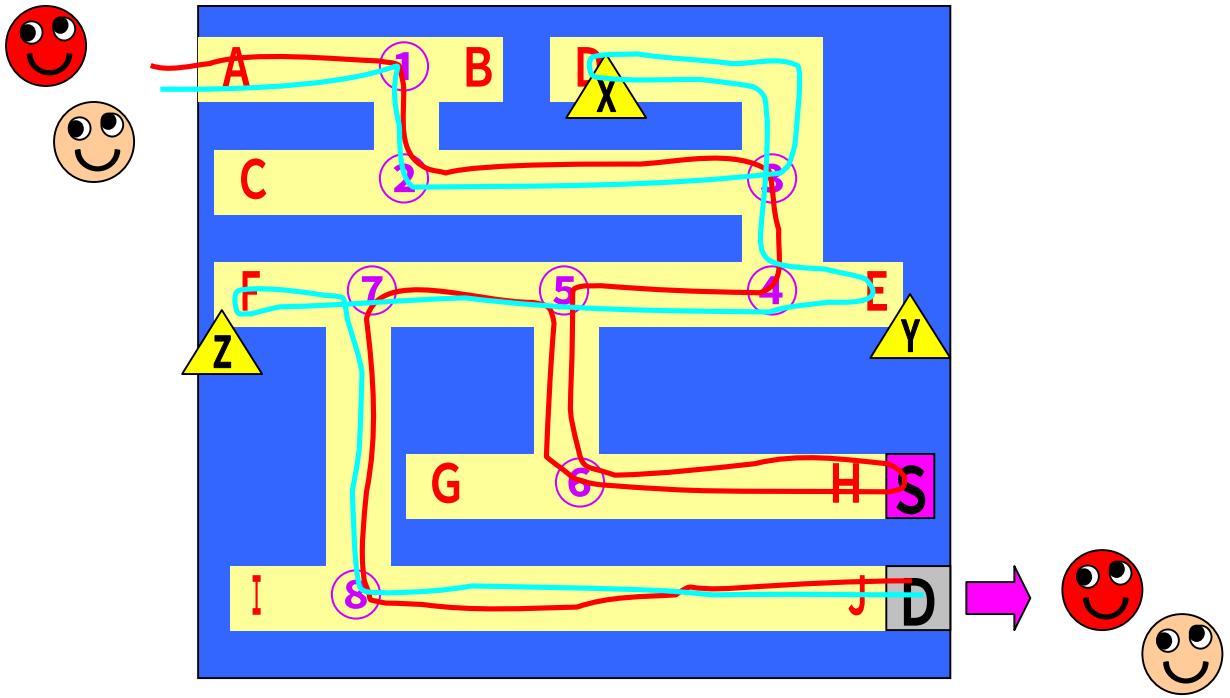
② チームAI：階層型タスクネットワーク（タスクによる分担）

全体のタスクを分割して、最適なAIに割り振る。



協調するAIを作る　①

動作例

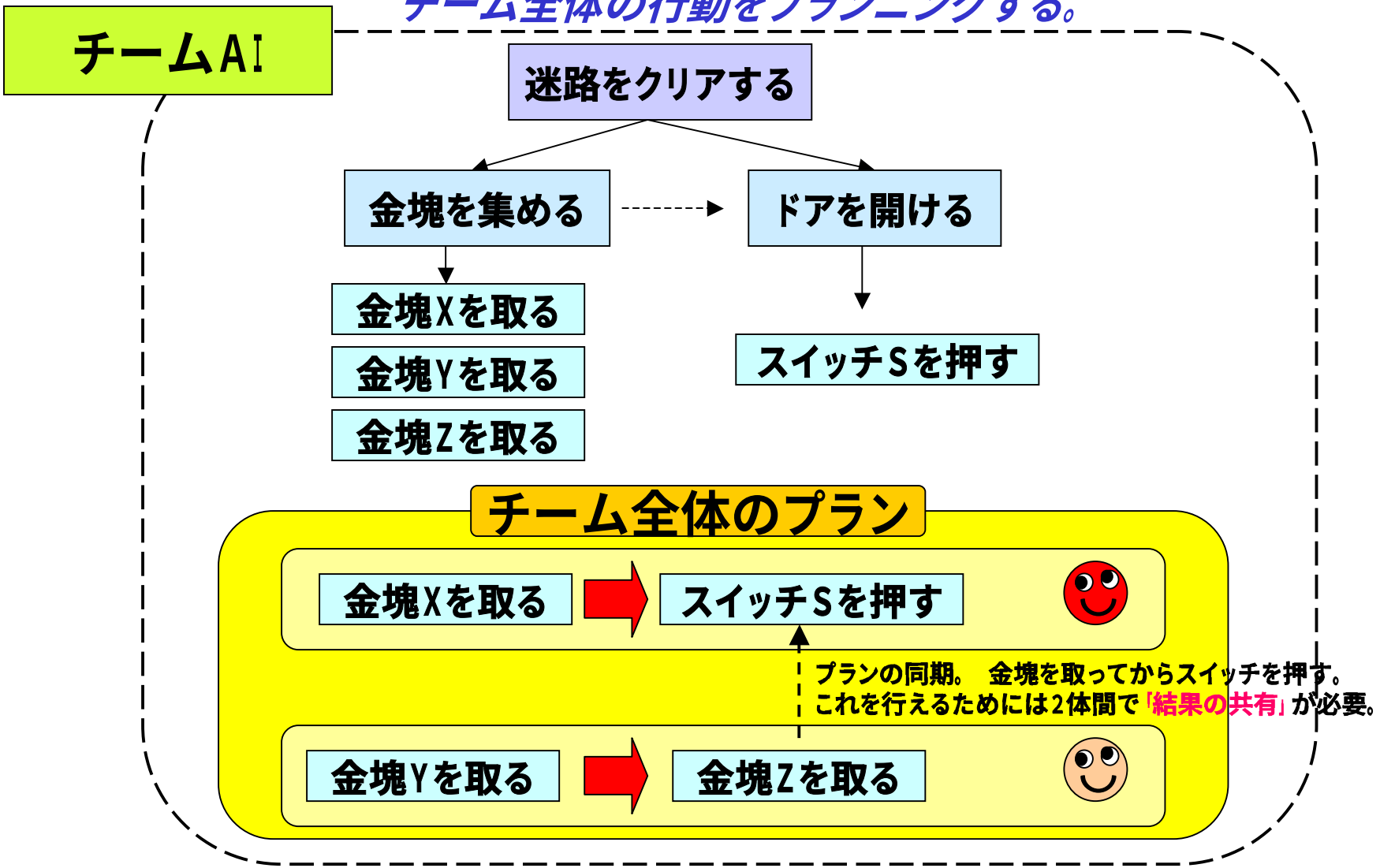


協調するAIを作る　①

A5：NPCを協調させる

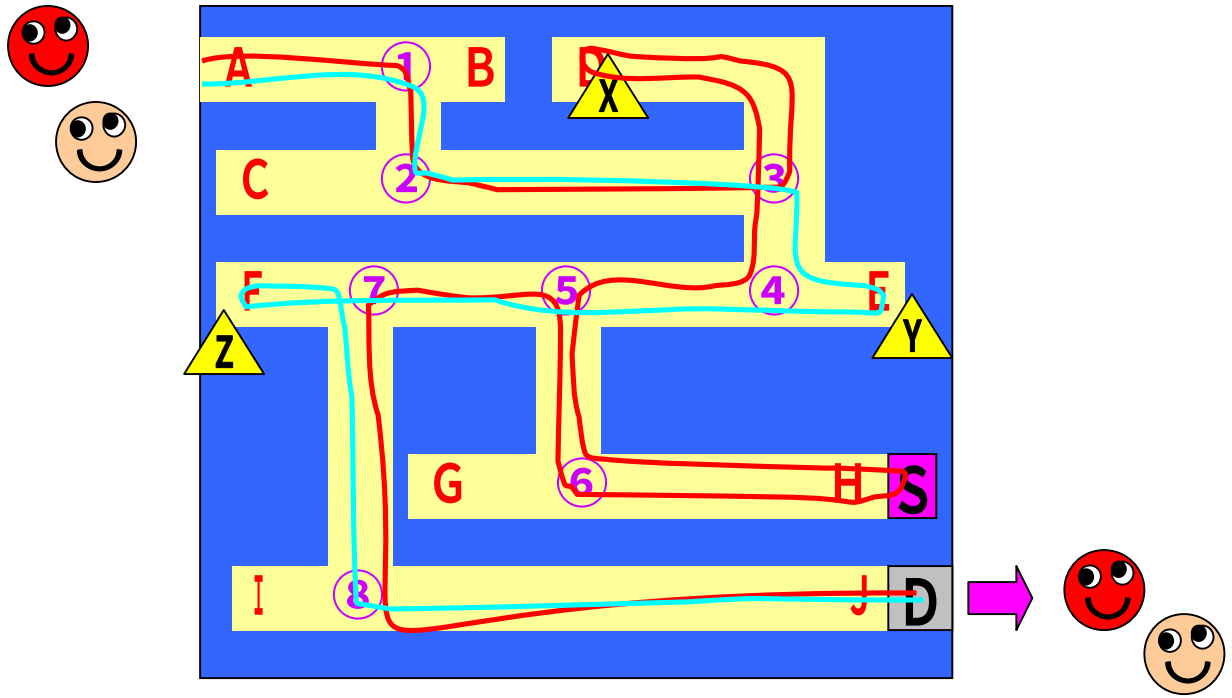
③　チームAI：マルチエージェント・プランニング

チーム全体の行動をプランニングする。



協調するAIを作る　①

動作例



協調するAIを作る　課題①

プログラマー

「階層型タスクネットワーク」(HTN)について調査して実装しよう。

「マルチエージェント・プランニング」について調査して実装しよう。

企画

「階層型タスクネットワーク」(HTN)では、分解されたタスクに
適したメンバーを選ぶ必要がある。

例えば、「金塊を集める」「スイッチを押す」二つのタスクに適した
AIをどちらか選ぶための条件とは何だろうか？

ＮＰＣを協調させることで、
チームとしての行動を実現することができる。

ＮＰＣの協調のポイント

情報の共有

結果の共有

これまでの行動の結果
を共有する。或いは、
得た情報を共有する。
（例）金塊Ａを取った、
金塊Ｃを取った
…

プランの共有

これからの行動のプラン
を共有する
（例）私ＮＰＣ１は金塊Ａを取りに行きます
では、私ＮＰＣ２は
スイッチＳを押しに行きます

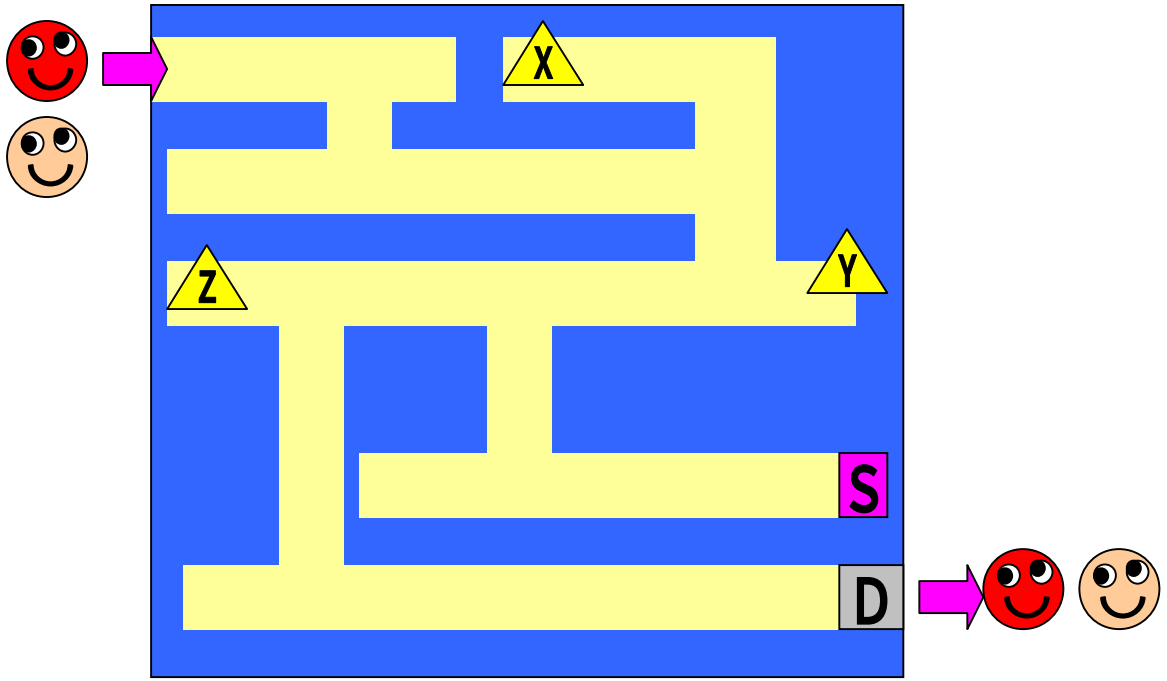
行動の同期

行動を同期させる
（例）金塊を全て取ってから
スイッチＳを押す

協調するAIを作る　②

**Q6：2体のNPCが情報を共有するためには
どうすればよいだろうか？**

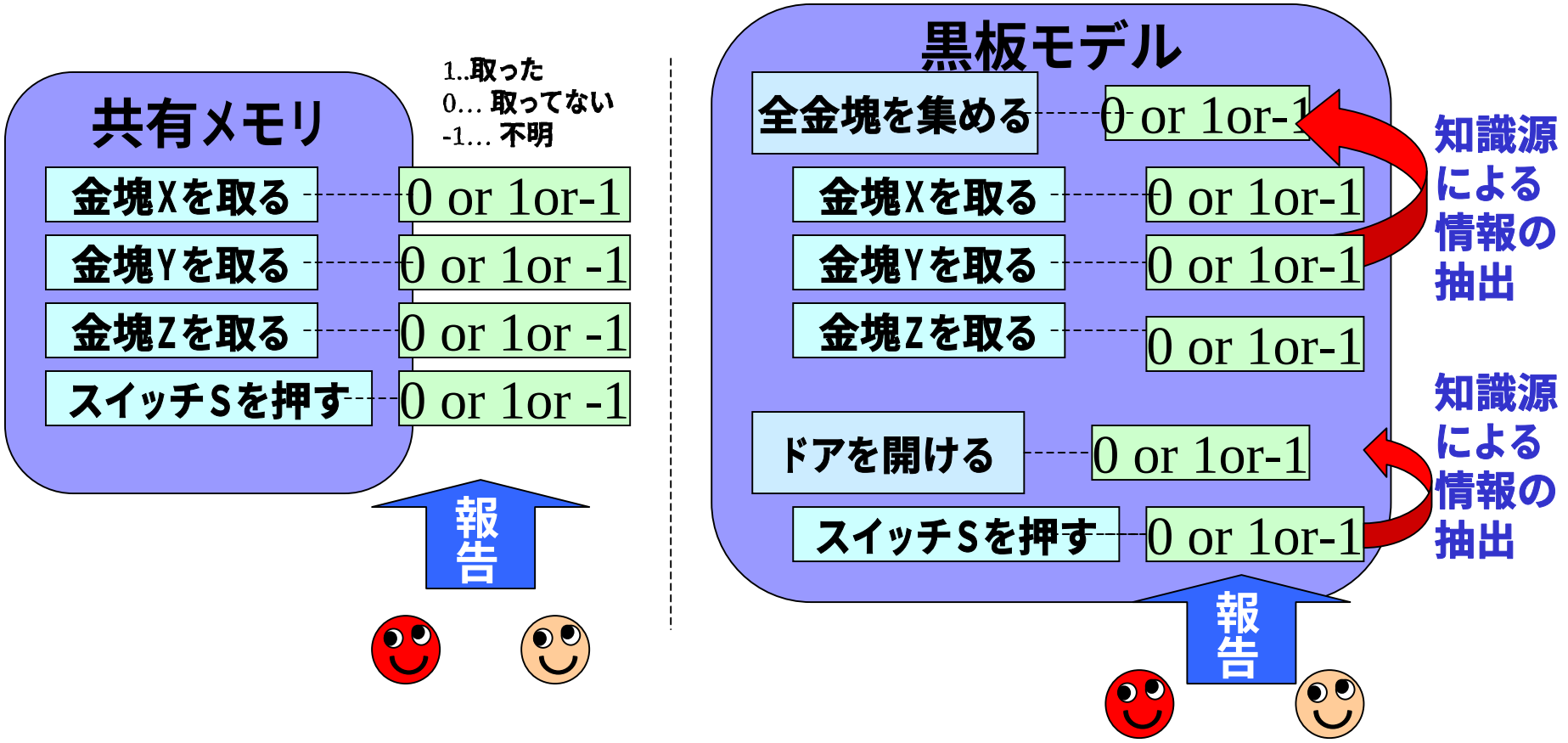
設定：　3つの金塊を集めてから、スイッチ（S）を押すと迷路を脱出する扉（D）が開く。



協調するAIを作る　②

A6-1：共有メモリ、或いは黒板に認識した結果を書き込む

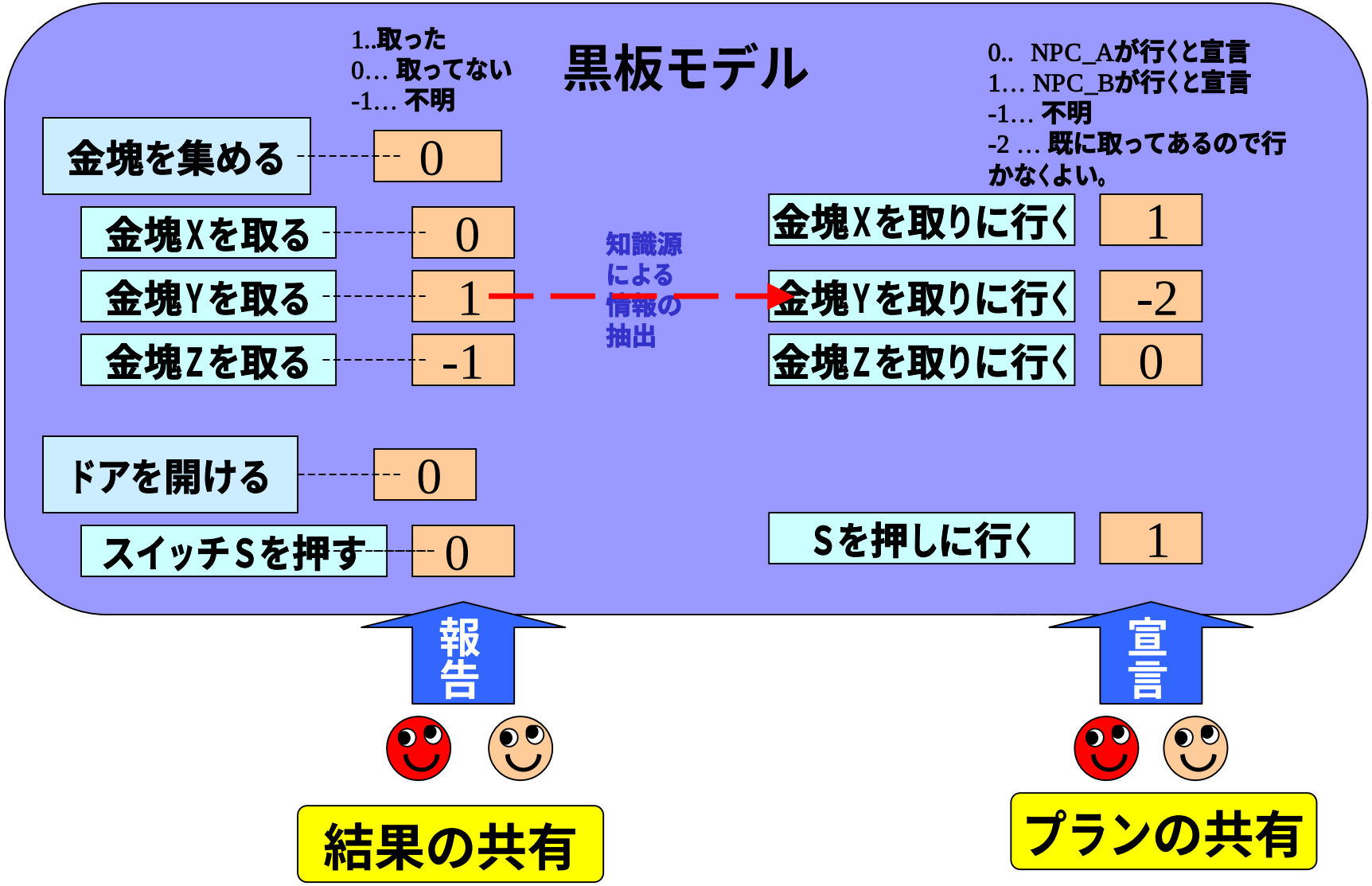
結果の共有



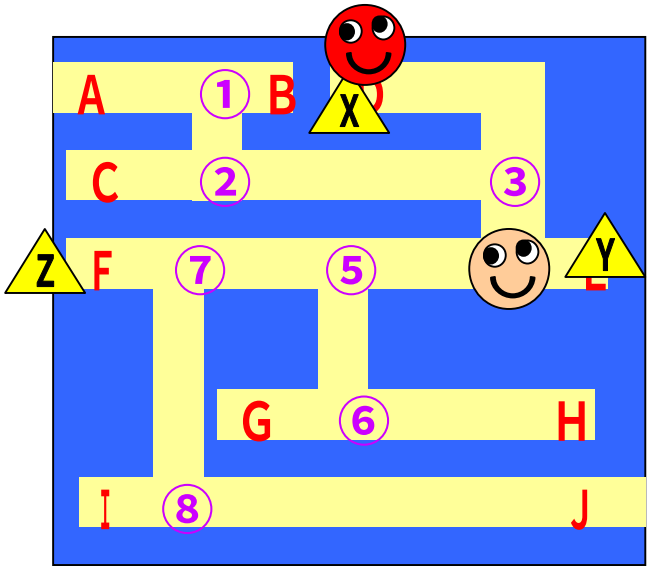
黒板モデルは「共有メモリ＋知識源」からなり、知識源は、
書き込まれた情報から2次、3次…情報を抽出する。
共有メモリに自動的な情報処理機能をつけた発展版と思ってください。

協調するAIを作る　②

A6-1：　(発展) 黒板に記憶と行動プランを書き込む



動作例



この時点で、NPC_A
が金塊Xを取った。
NPC_Bは金塊Yを取ろうとしている。

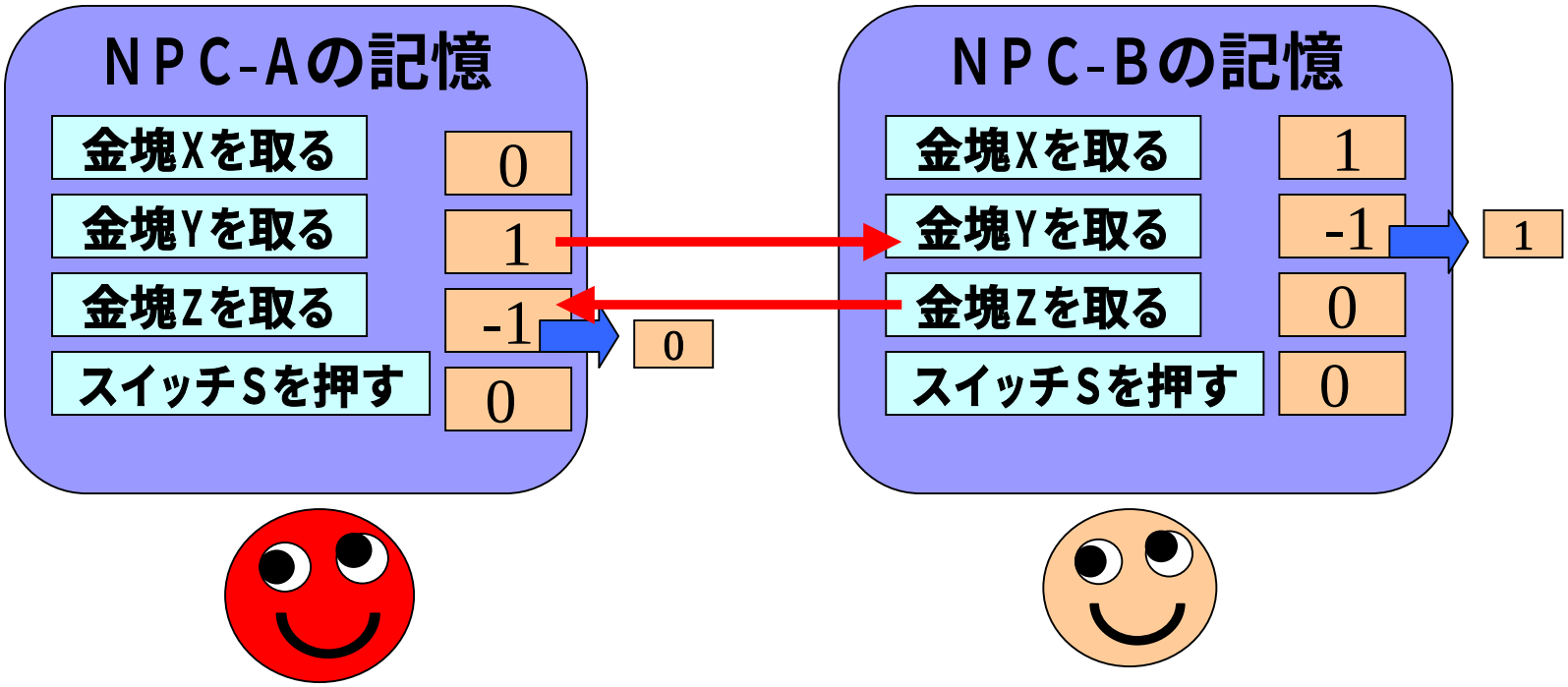
従ってAは、YはBに任せて
「次は金塊Zを取りに行く」
と考える。

黒板モデル

金塊を集める	0		
金塊Xを取る	1	金塊Xを取りに行く	1
金塊Yを取る	-1	金塊Yを取りに行く	0
金塊Zを取る	0	金塊Zを取りに行く	-1
ドアを開ける	0		
スイッチSを押す	0	Sを押しに行く	-1

協調するAIを作る　②

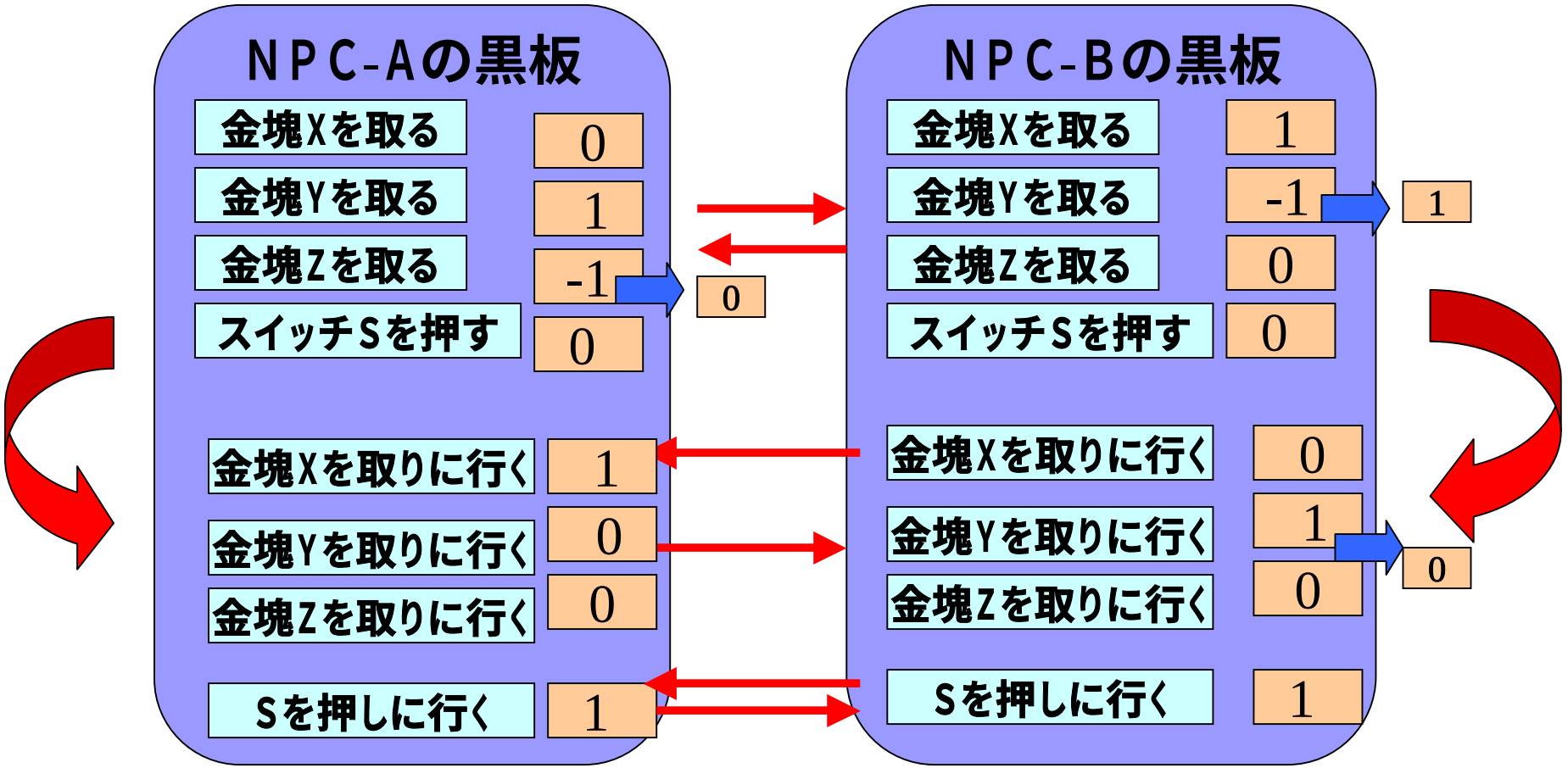
A6-2：情報を交換する



知識交換(Knowledge exchange)モデルは、お互いのエージェントが
決められた形式を持っていることを前提とする（同じ構造でなくてもよい）。

協調するAIを作る　②

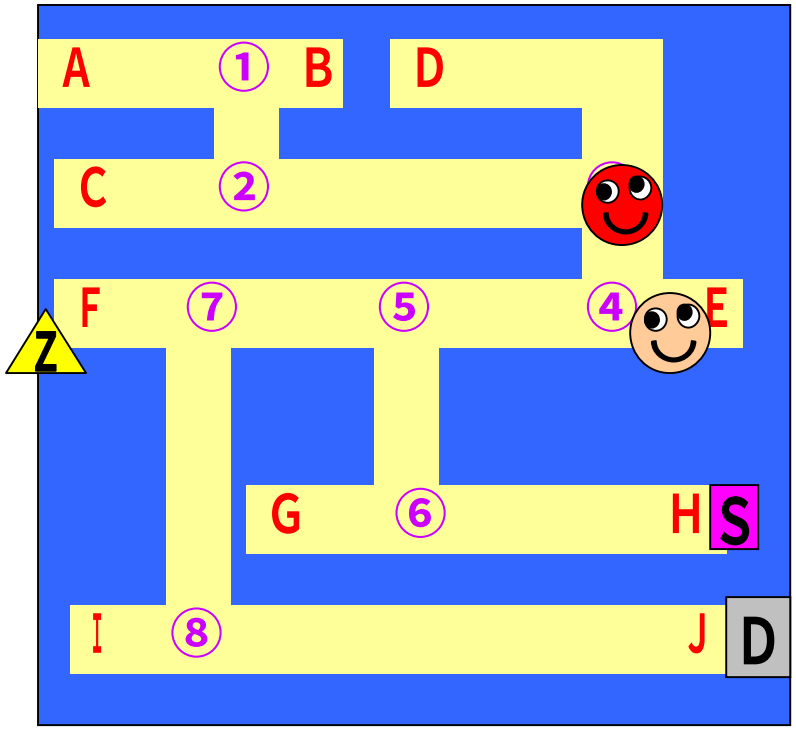
A6-2：情報を交換する



知識交換と行動の計画の調整

協調するAIを作る　②

動作例



この時点で、NPC_A
が金塊Xを取っており、
NPC_Bは金塊Yを取っている。

NPC_A，Bは互いに情報を交換しあい、
金塊X，Yを獲得したことを知る。

NPC_Aは、自分がスイッチSを押しに
行くことを宣言し、その宣言を受けて、
NPC_Bは、金塊Zを取りに行く。

協調するAIを作る　課題②

プログラマー

「黒板モデル」(Blackboard architecture) を実装して、協調動作を実現しよう。

また、知識交換モデルを実装して、企画と相談して交換のタイミングを決め、協調動作を実現しよう。情報の交換のキャッチボールをし続けないように、交換のタイミングとトリガーを工夫しよう。

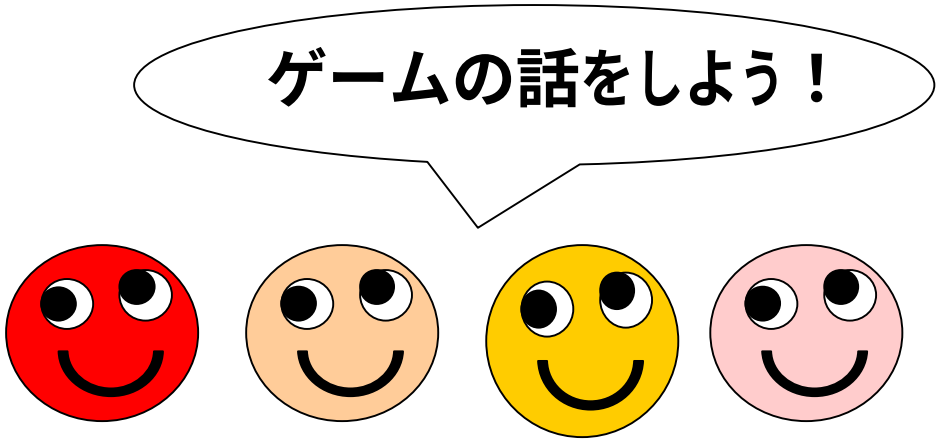
企画

黒板モデルでは、ＮＰＣ同士の位置に関係なく、情報が共有されてしまう。プレイヤーから見た場合、完全にチートとなる。チートは、プレイヤーを興ざめさせてしまう恐れがあるが、逆に、チートしないことにこだわっても、ＮＰＣの動きを鈍くしてしまう。

また、知識交換モデルでは、ＮＰＣ同士にいつ交換を可能にするべきかを設定する必要がある。この例で、そのタイミングを考えよう。

Step4

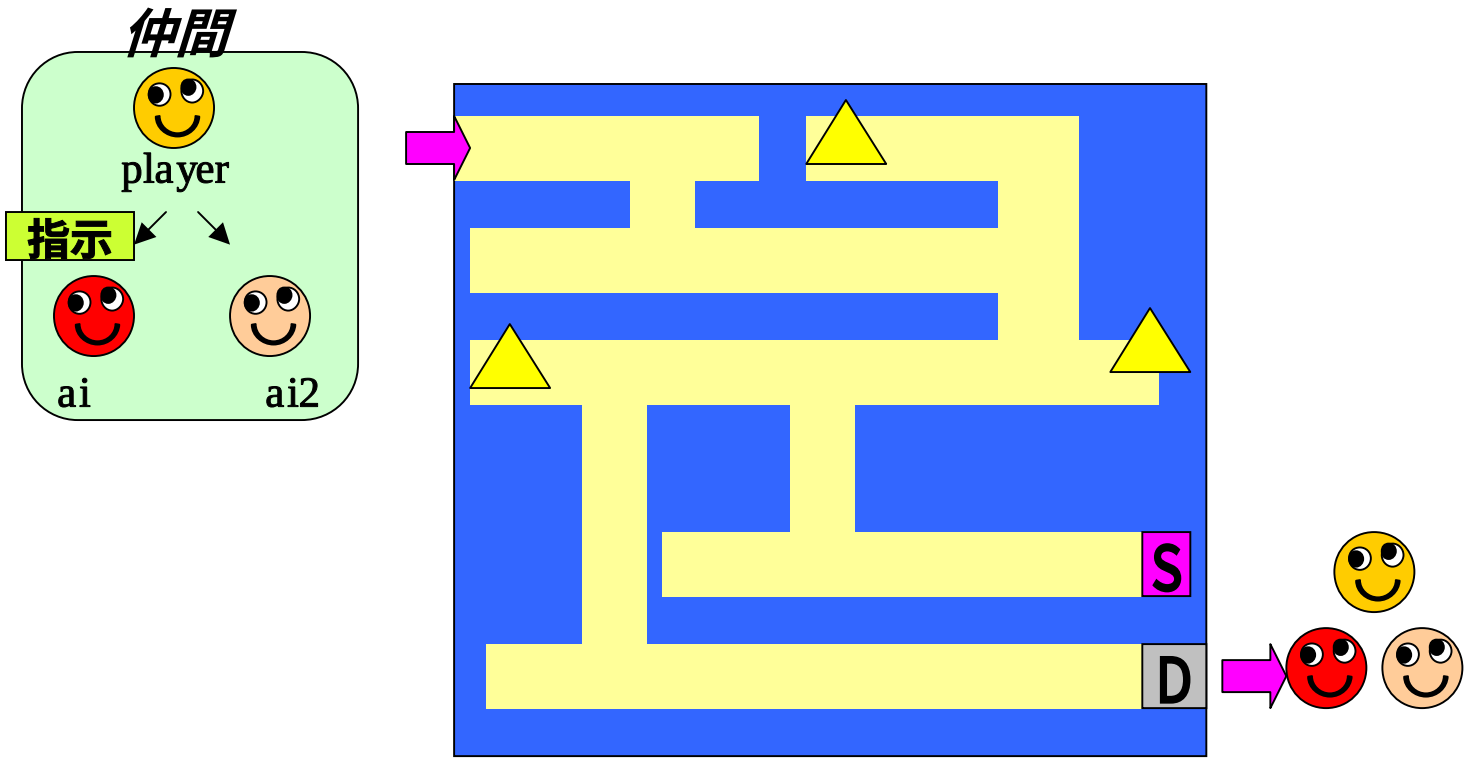
このNPCたちを作ってゲームを作ってみる



ゲームを作る①

**Q7：プレイヤーを加えて、
知性を持ったＮＰＣを仲間として動かすには？**

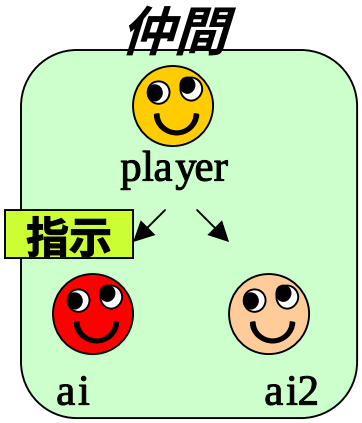
設定：　3つの金塊を集めてから、スイッチ（S）を押すと迷路を脱出する扉（D）が開く。

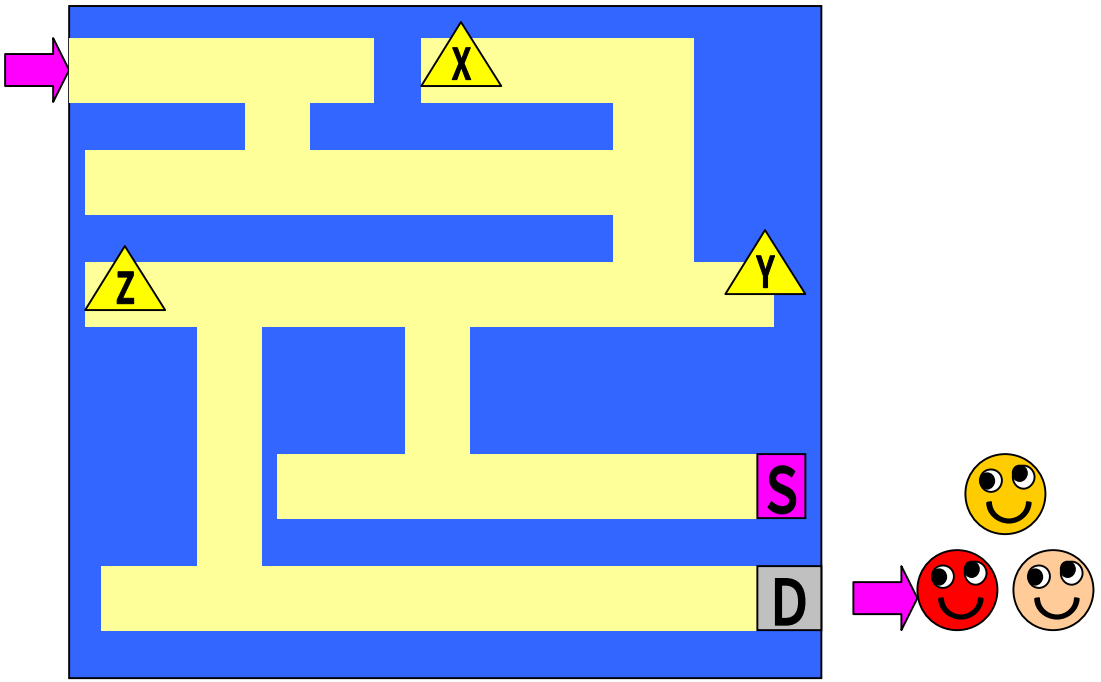


ゲームを作る①

A7：インターフェースを作ってAIに指示が出せるようにする

例えば、「金塊Xを取れ」という命令を出すと、
NPCに、何処からでもパス検索をして金塊を取りに行かせることが出来る。
つまり、チームAIの部分を、プレイヤーが果たす。
命令を受けたAIは、自分でパスを検索して命令を実行する。





ゲームを作る①

動作例

ai

ai2

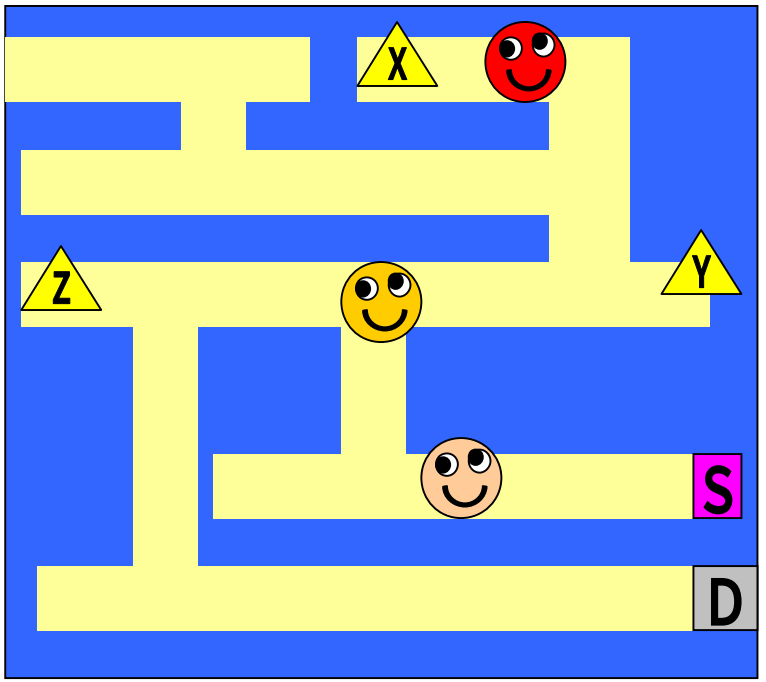
金塊Xを取れ

金塊Yを取れ

金塊Zを取れ

スイッチSを押せ

キャンセル



プレイヤーは抽象的な命令を出すだけで、命令を実行するための細かいプランは、NPC自身が考える。

ゲームを作る　課題①

プログラマー

インターフェースを実装して、プレイヤーからＮＰＣに
指示が出せるようにしよう。

企画

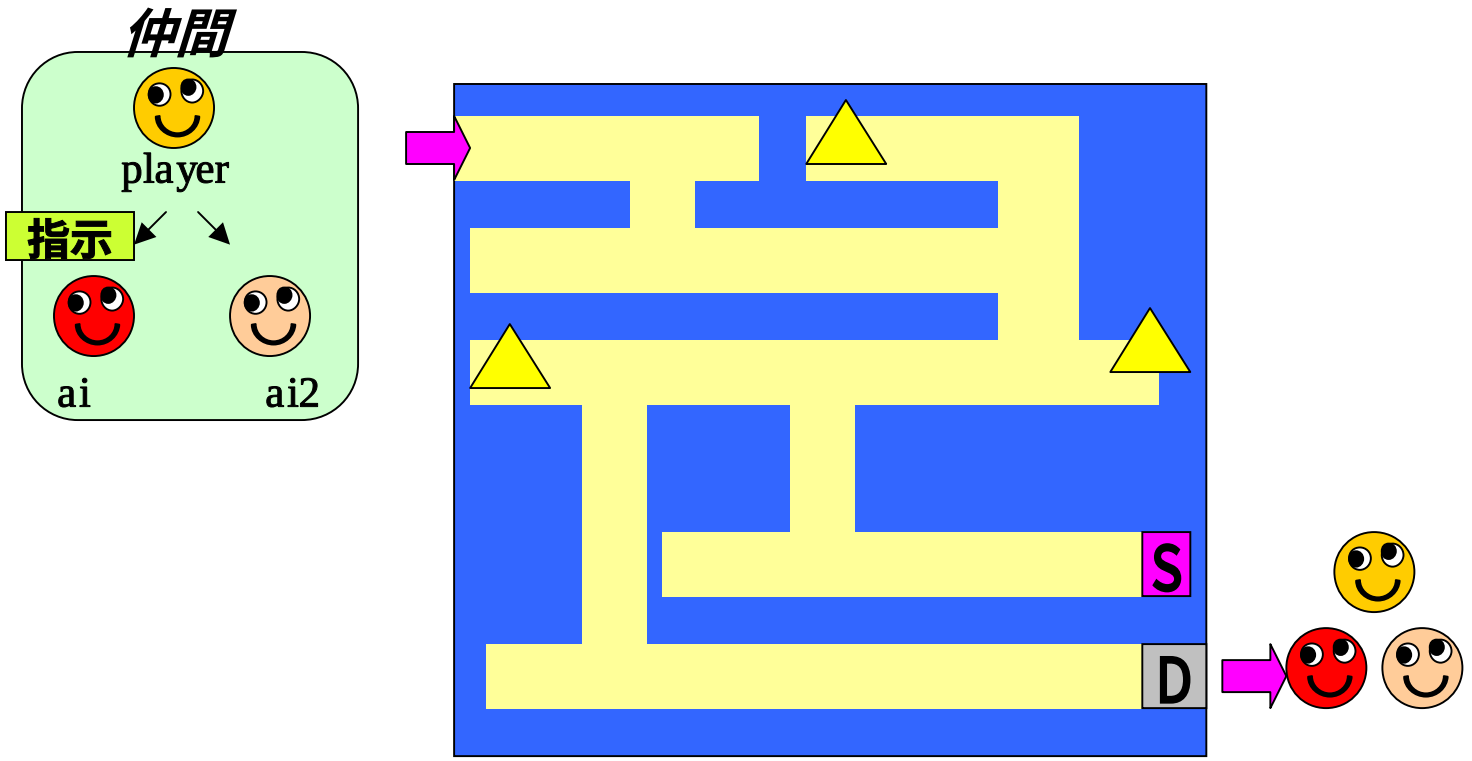
インターフェースのデザインとパッドのコマンド入力に
仕方を考えよう。

どうすれば、プレイヤーがストレスなくAIに指示を出せるだろうか？

ゲームを作る②

**Q8：プレイヤーを加えて、
知性を持ったAIを仲間のチームとして動かすには？**

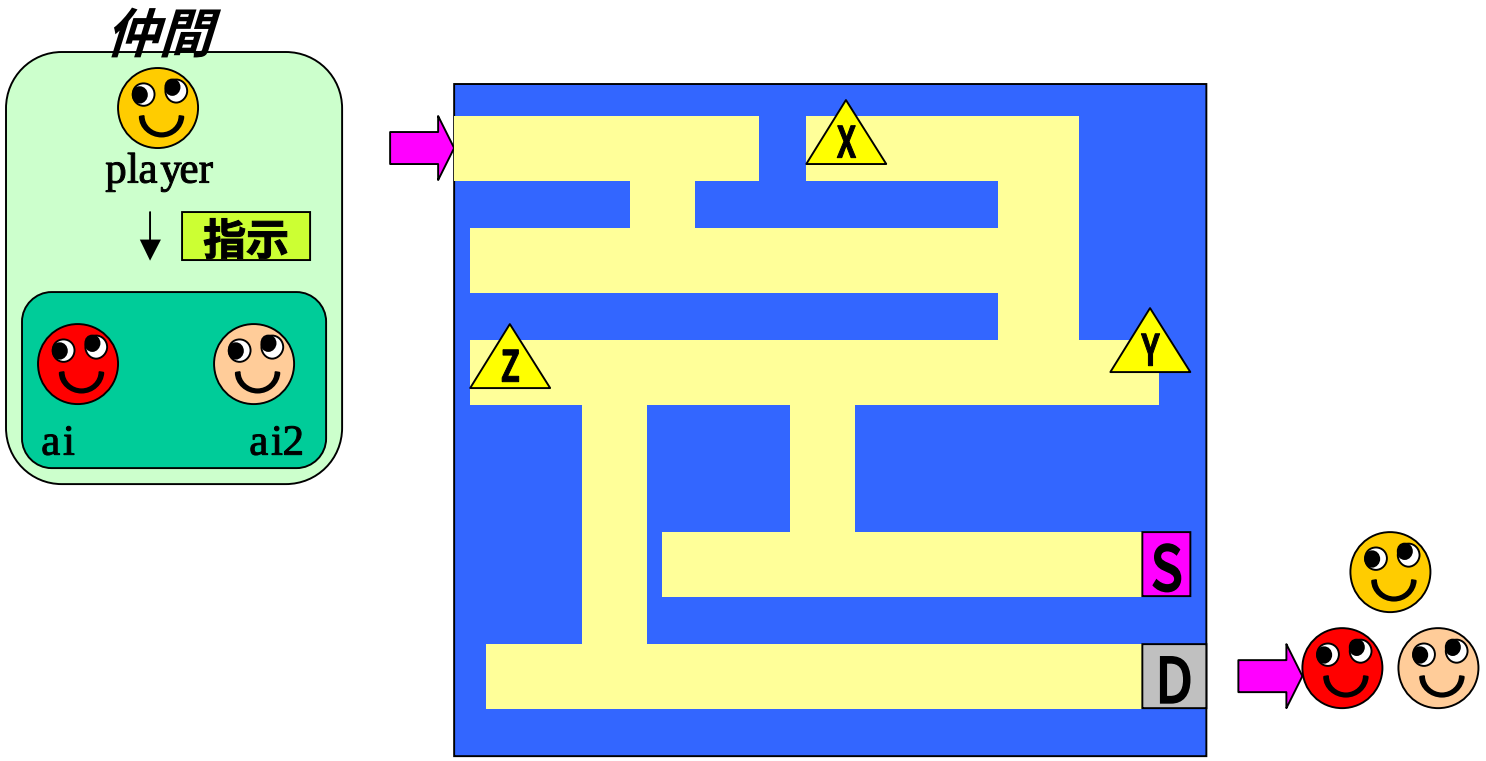
設定：　3つの金塊を集めてから、スイッチ（S）を押すと迷路を脱出する扉（D）が開く。



ゲームを作る②

A8：インターフェースを作ってグループに指示が出せるようにする

例えば、「金塊を取れ」という命令を出すと、
AIたちは、自分たちで協調して金塊を取る。

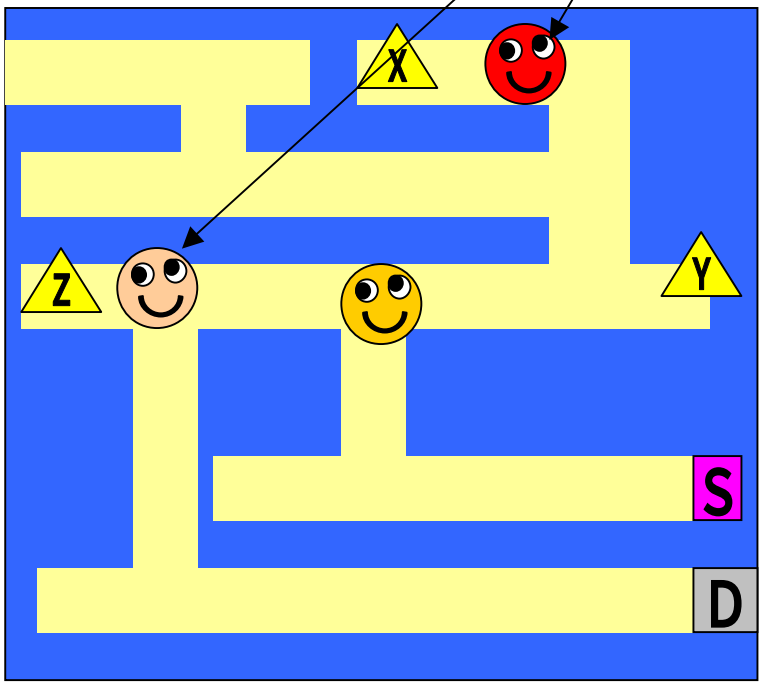


ゲームを作る②

金塊を揃える
ために協力して行動

動作例

- 金塊を取れ
- スイッチSを押せ
- キャンセル



プレイヤーはチームに抽象的な命令を下すだけで、
チーム内のプランはAIが自ら考えて実行する。

ゲームを作る　課題②

プログラマー

ＮＰＣチームに命令を指定することができるようにし、
ＮＰＣチームは受けた命令をチーム内で協調して行動できるように
実装しよう。

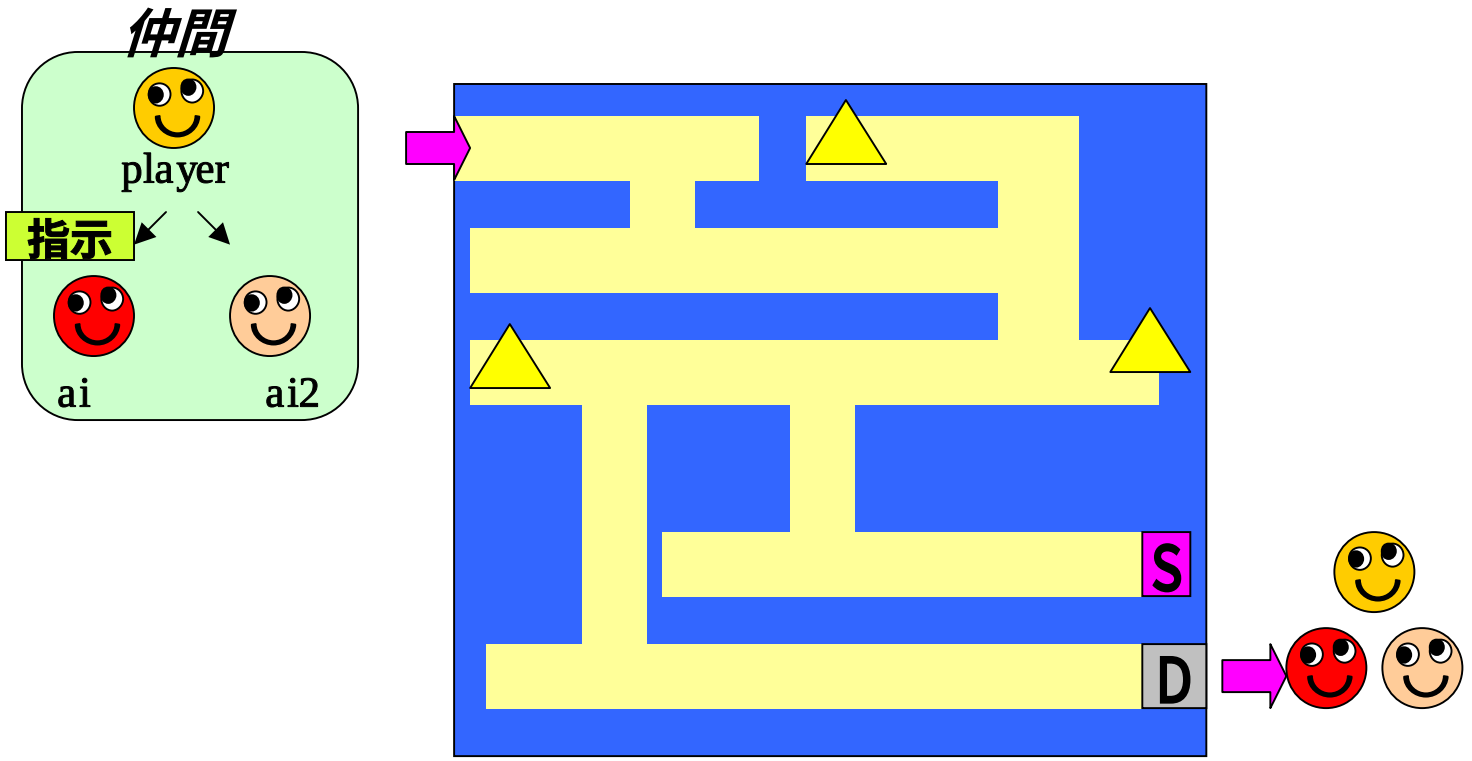
企画

ＮＰＣチームに渡す命令は、どのようなものが考えられるだろうか？

ゲームを作る③

Q9:プレイヤーをＮＰＣと戦わせるには？

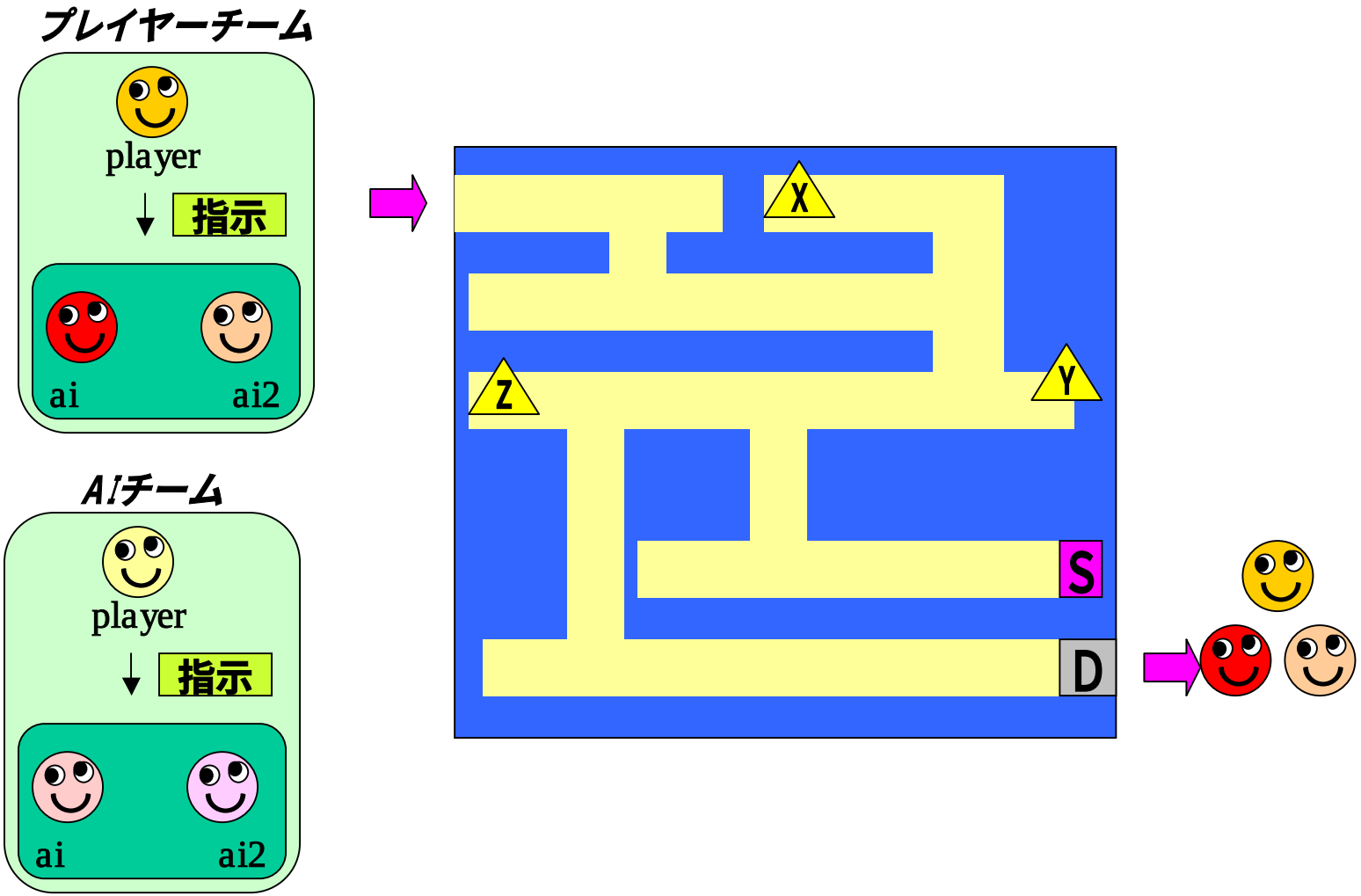
設定：　3つの金塊を集めてから、スイッチ（S）を押すと迷路を脱出する扉（D）が開く。



ゲームを作る③

A9：AIチームを作って同じ条件で対戦させる

全てのメンバーが先に迷路から出れば勝利



ゲームを作る　課題③

プログラマー

ＮＰＣチームとプレイヤーチームが対戦できるようにする。

企画

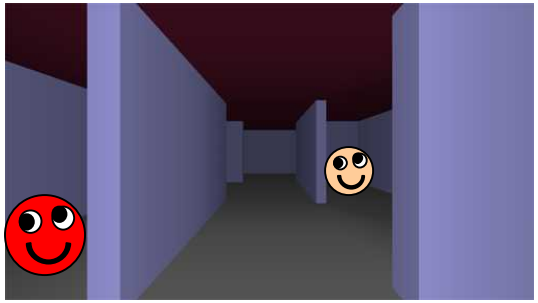
このゲームはこのままで面白いだろうか？

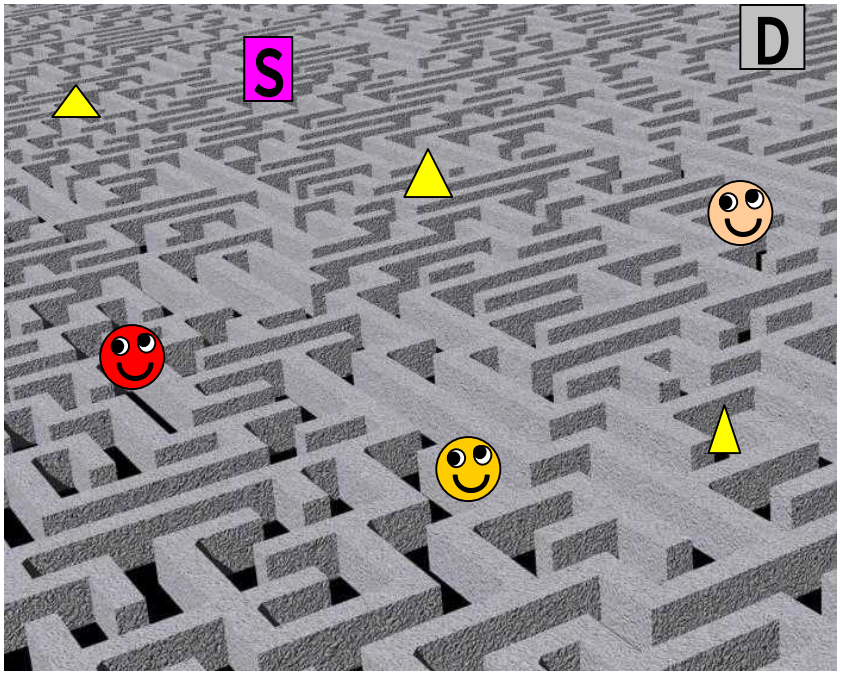
ゲームを作る④

Q10：Q1~9で作ったゲームはまだ全然面白くない。
自分のアイデアや設定を変えて、
AIの技術を使った面白いゲームを作ろう！

ゲームを作る④

A１０（例）　（1）　ゲーム設定を変える。
（2）　迷路を広くする。
（3）　３Ｄにする。
（4）　…



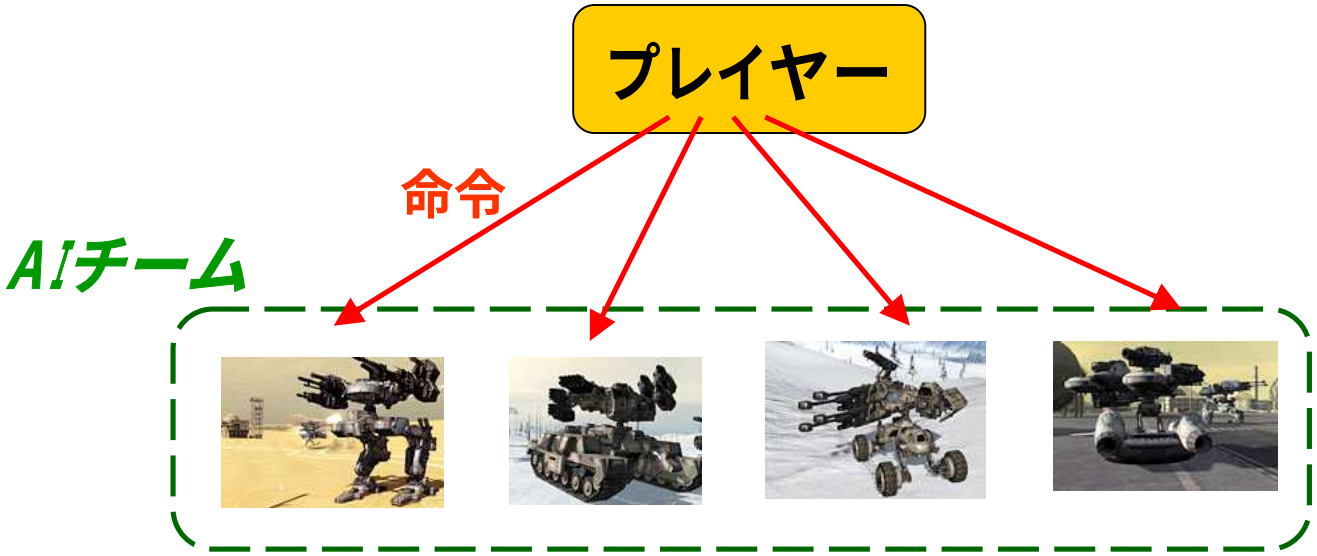


企画とプログラマーで
自由にAIを使ったミニゲームを考えて作ってみよう！

(応用例)　クロムハウンズ

オフライン　コマンダーミッション

「オンラインにおいて人間のチームメンバーに指示を出してチームを連携させる」練習として、オフラインのストーリーモードにおいては、フィールドの情報をアンテナで収集しながら、数対のＮＰＣに命令を出してミッションをクリアするステージが用意されている。



ＮＰＣの処置としては、「命令」を「ゴール」として扱い、与えらたゴールを遂行するためにプランニングを行って実行する。

(注) ただしミッションの都合上、AIの自律的能力はかなり抑えてある。

インターフェース

十字キーで、コマンドを
形成して行く。見えているのは、
コマンダー自機の背中

それぞれの段階で、
様々な指令や対象が
用意されている。

プレイヤーは命令を出した後は、自分も移動しながら戦局を判断し、新しい命令を出すことでAIチームを協調させながら、戦局をコントロールする。

AIは「何をするべきか？」というゴールを人間から与えられるが、「如何にするべきか？」はプランニングによって自分で考える点が新しい点である。

お疲れさまでした。

(1) 迷路というゲーム世界に限定して、
第1回から第3回の内容を解説しました。

(2) 特に迷路の設定が本質ではないので、一般にゲーム
全般についてAIを使ったアイデアを考えてみてください。

(3) ここで説明した技術の Killzone, Halo2, F.E.A.R, Chrome Hounds における応用は、第1，2回テキスト、
第3回の参考資料をご覧ください。

[参考文献] 第1、2回講演資料と第3回参考資料 (第3回申し込みページよりリンク)
http://www.igda.jp/modules/eguide/event.php?eid=41

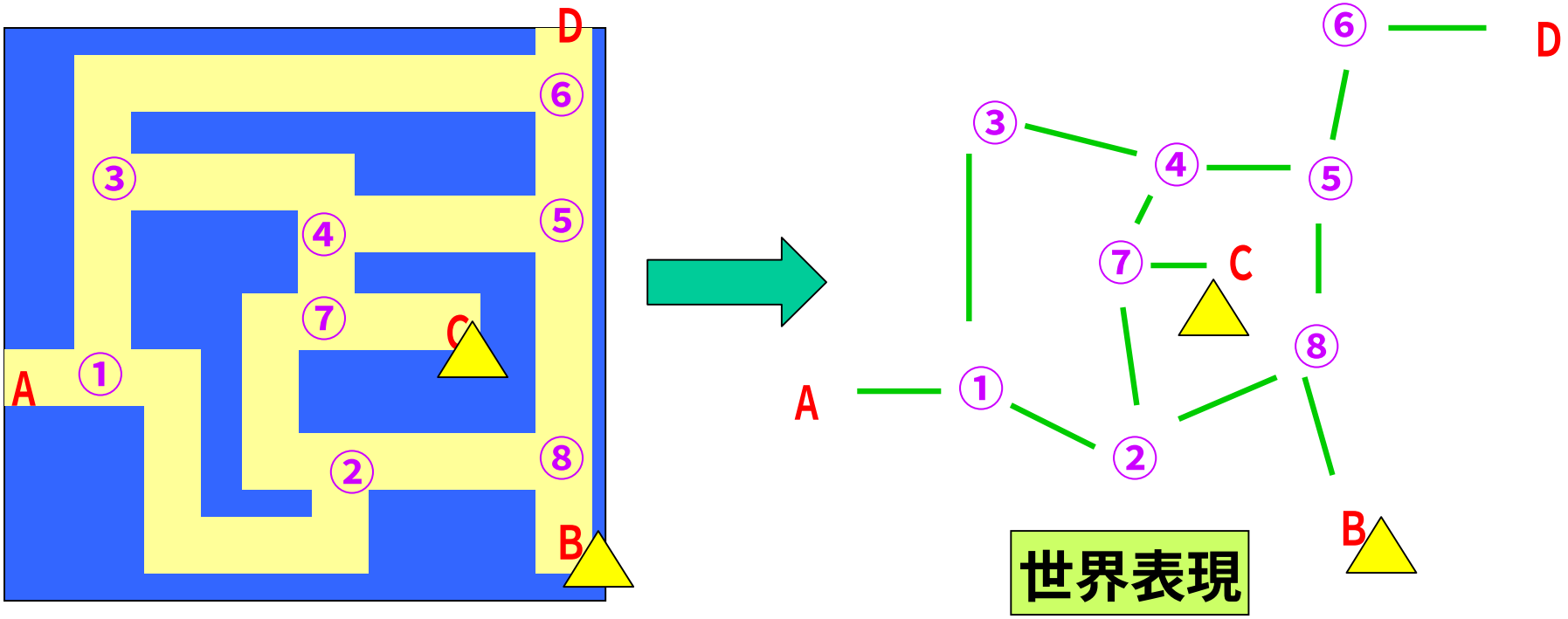
付録１：技術的発展

Ｑ.Ｅ１　一般の迷路を扱えるようにする。

Ｑ.Ｅ２　上記の例では、あらかじめ迷路の
全体像が与えられていたが、動的にAIが認識した
部分だけ迷路の構造が明らかになるようにする。

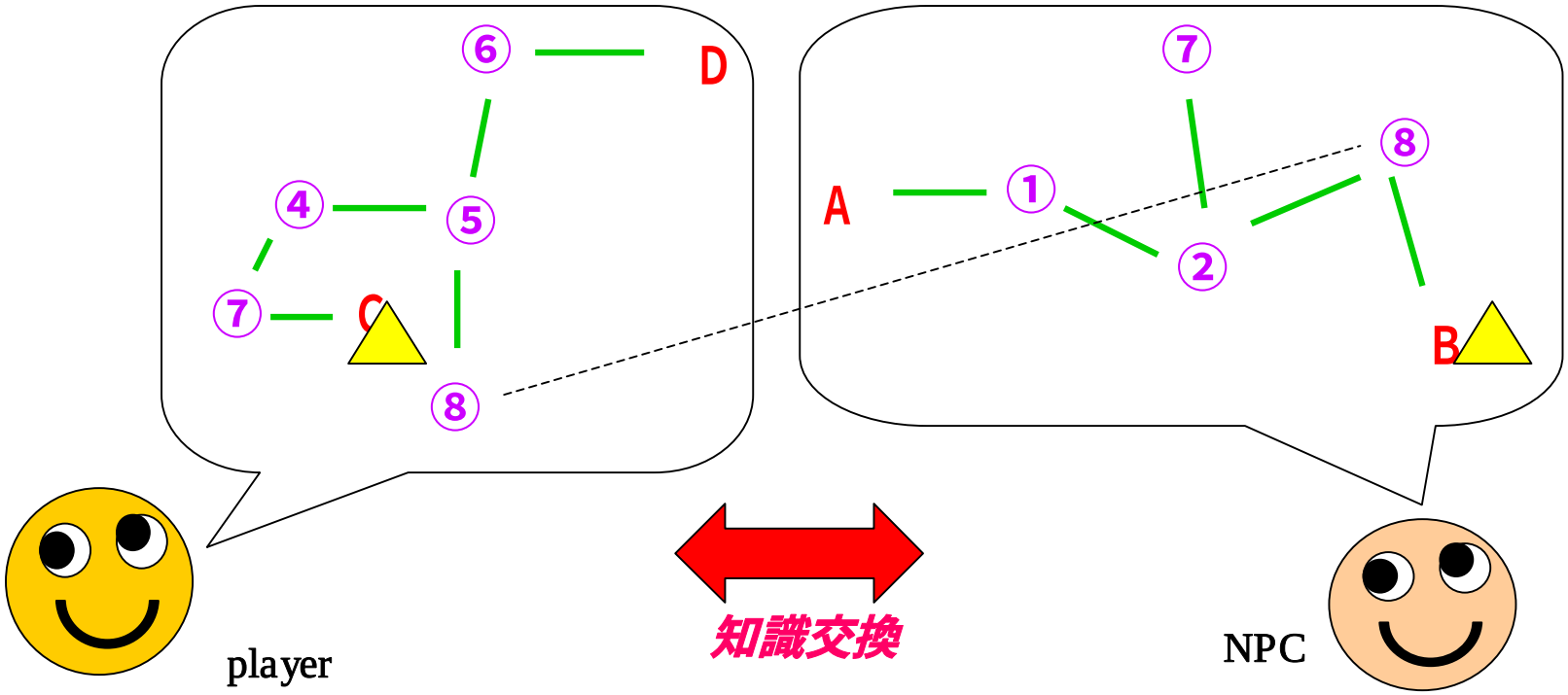
Ｑ.Ｅ３　3体以上のAIについて協調させる。

A.E1　一般の迷路はネットワークグラフになり、
パス検索のアルゴリズムは、ダイクストラの
アルゴリズムかA＊アルゴリズムを用いることができる



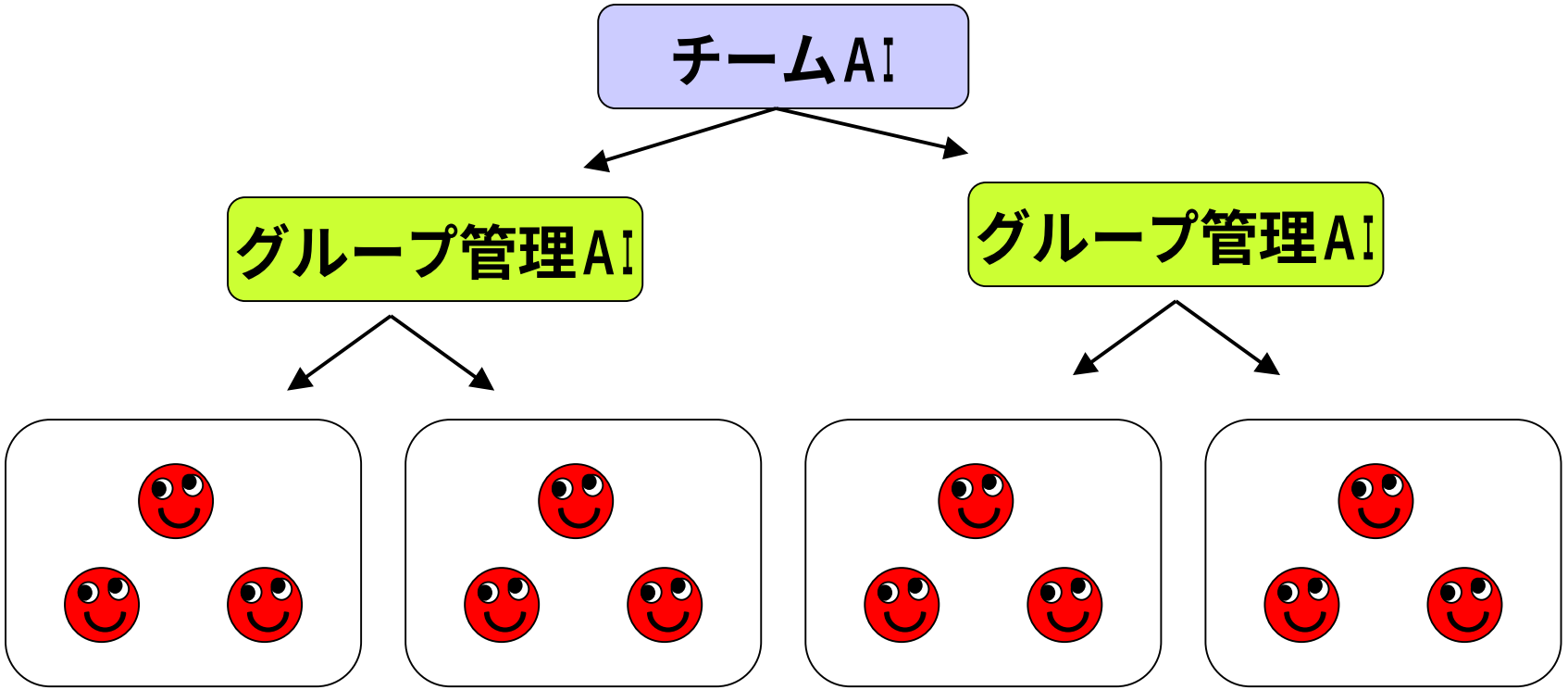
A.E２　チートをしないためには、認識した領域だけ使うようにする。
また、AIが未知の洞窟を調査する様子は、プレイヤーから見ても、
リアリティーのあるAIとなる。動的にグラフ構造を発展させる方法
あらかじめマスクしたグラフ領域のマスクを解除して行く方法、
を取ることが出来る。

自分の行かない方角にNPCを派遣して情報を持って来させて、
自分の情報と会わせてダンジョンの全体像を知っていくゲームは面白いと思う。



A.E3

相互作用モデル（多数のAIがそれぞれ相互作用をするシステム）は頻繁にコミュニケーションする場合は調整がしずらくなる。そういった場合は、階層構造を導入すると制御がしやすくなる（集中管理型）。



ご感想や質問があれば、メイルかセミナーでお伝えください。

y_miyake@fromsoftware.co.jp

(IGDA Japan登録アドレス　yoichi-m@pk9.so-net.ne.jp　)

ＷＥＢ上の意見交換にはＩＧＤＡ Japanのサイトをご利用ください

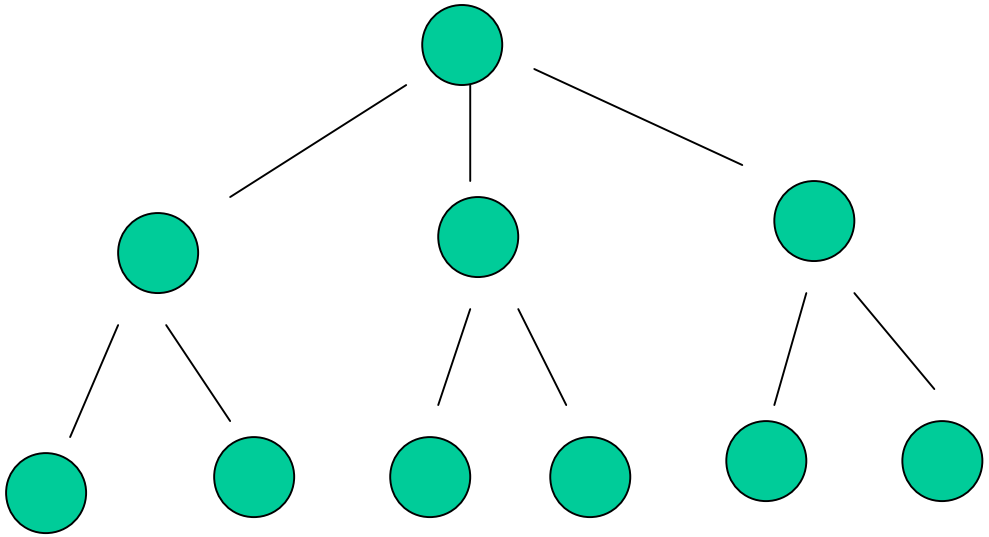
http://www.igda.jp

付録2:パス検索法　解説

DFS, BFS, Dijkstra, A*

グラフ検索アルゴリズム

パス検索はデータ構造が出来れば、
グラフ検索の問題である



主な検索アルゴリズム

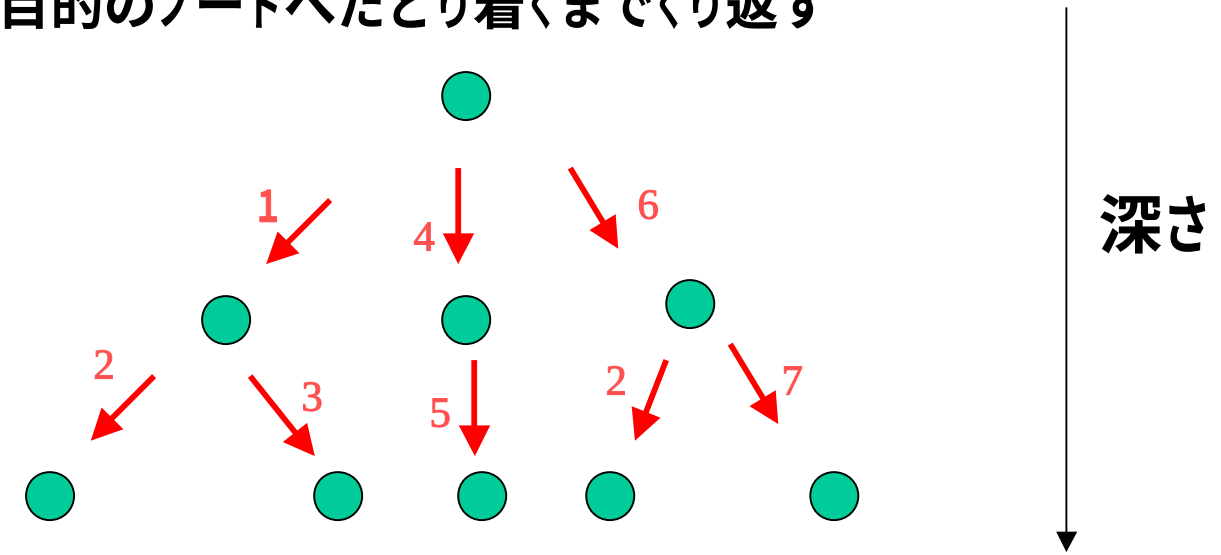
DFS	Depth First Search	深さ優先検索
BFS	Breadth First Search	幅優先検索
Dijkstra	Dijkstra’s algorithm	ダイクストラ
A*	A* algorithm	エースター

DFS Depth First Search

深さ優先検索

深いノードを優先して検索して行く

1. 左優先として、ノードの終点の深さまで行く
2. まだ行っていない子ノードがあるノードまで戻る。
3. ノードを終点の深さまで行く
4. 目的のノードへたどり着くまでくり返す

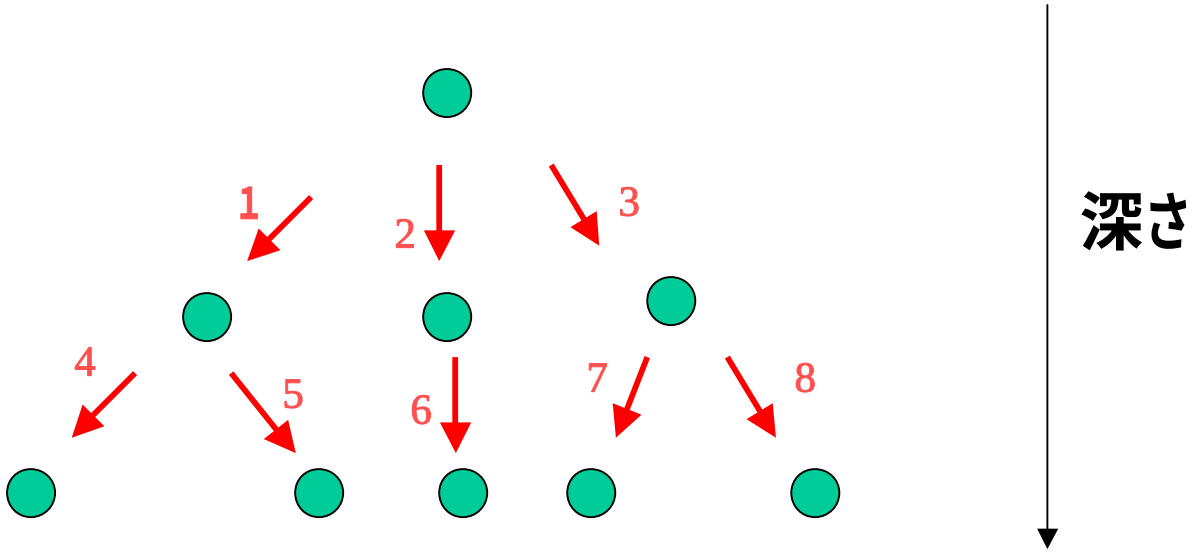


BFS Breadth First Search

幅優先検索

浅いノードから同じ深さのノードを優先して検索して行く

1. 左優先として、ノードの終点の深さまで行く
2. まだ行っていない子ノードがあるノードまで戻る。
3. ノードを終点の深さまで行く
4. 目的のノードへたどり着くまでくり返す



Dijkstra Dijkstra's Algorithm

ネットワークの中で最短のパスを探す

[前提] ノード間に距離があり、あるノードからあるノードまでコストが定義される（コスト＝2点間の距離）

(1) 開始ノードにつながれているノードたちへのコストをチェックする
(2) チェックしたリストの中から最も小さな値のノードの先につながれているノードへのコストを計算し、リストに加える
(3) （2）へ戻り、くり返し、目的のノードへのパスを見つける
(4) それがリストの中で最小の値なら終了。そうでなければ（2） へ

AからFへの最短経路を求めよ

B 23
C 34
D 15

→ Dが最も小さい

B 23
C 34
D-G 35

→ Bが最も小さい

B-E 53
B-C 33
C 34
D-G 35

→ ACより ABCが小さい

B-E 53
B-C 33
D-G 35

→ B-Cが最も小さい

B-E 53
B-C-E 41
B-C-F 42
B-C-G 42
D-G 35

→ D-Gが最も小さい

B-C-E 41
B-C-F 42
B-C-G 42
D-G -C 44
D-G-F 40

→ B-C-Eが B-Eより小さい

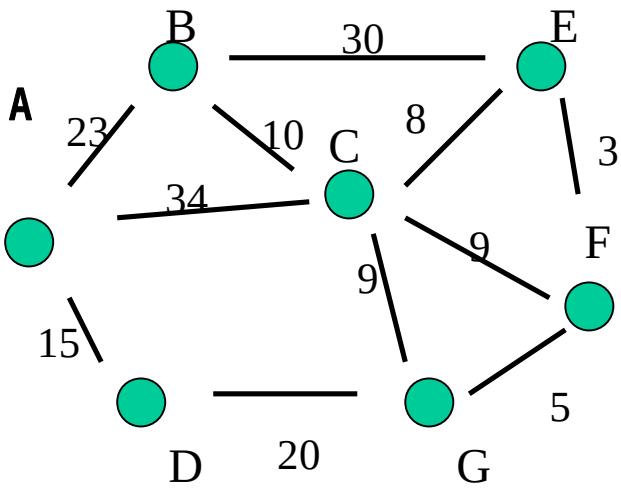
B-C-E 41
B-C-G 42
D-G -C 44
D-G-F 40

→ B-C-Fより D-G-Fが小さい

A-D-G-F 40

最短経路

← D-G-Fがリストの中で最小



A*

A* Algorithm

2点間に複雑に絡まったパスがあり、最短のパスを探す

[前提]ノード間に距離があり、あるノードからあるノードまでコストが定義される
Dijkstra Algorithm のコストを見積もりコストに変更したもの。そして、目的地へのパスが一つ見つかれば終了 (コスト＝2点間の距離＋ゴールまでの見積もりコスト)
見積もりコストは、実際の経路の値より小さい値でなければならない。
(普通はユークリッド距離。以下では説明のため、適当に見積もりコストをつけています)

AからFへの最短経路を求めよ

B 23+29
C 34+8
D 15+23

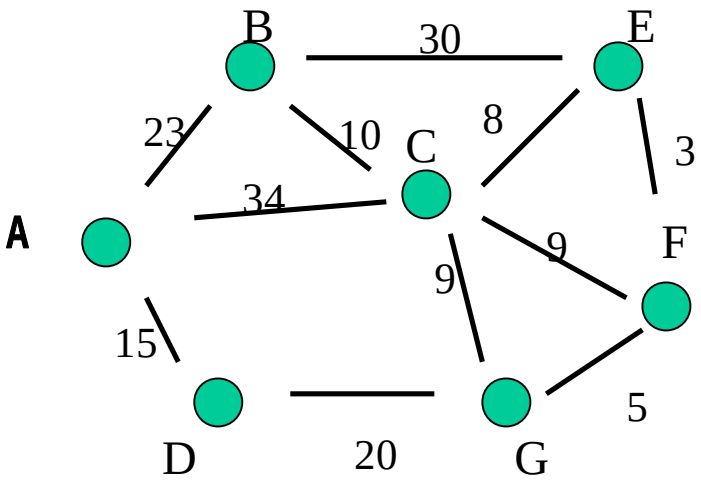
Dが最も小さい →

B 23+29
C 34+8
D-G 35+3

DGが最も小さい →

B 23+29
C 34+8
D-G-F 40
D-G-C 44+6

→ A-D-G-F 40 最短経路



このように発見的にパスを検索する方法を、Heuristic なパス検索という