知能プログラミング演習II 課題5

グループ8
29114142 湯浅範子

2019年12月3日

提出物　rep5(2914142.pdf),group08.zip

グループ　グループ8

メンバー

学生番号	氏名	貢献度比率
29114003	青山周平	
29114060	後藤拓也	
29114116	増田大輝	
29114142	湯浅範子	
29119016	小中祐希	

1　課題の説明

必須課題 5-1 目標集合を変えてみたときに，動作が正しくない場合があったかどうか，実行例を示して考察せよ．また，もしあったならその箇所を修正し，どのように修正したか記せ．

必須課題 5-2 教科書のプログラムでは，オペレータ間の競合解消戦略としてランダムなオペレータ選択を採用している．これを，効果的な競合解消戦略に改良すべく考察し，実装せよ．改良の結果，性能がどの程度向上したかを定量的に（つまり数字で）示すこと．

必須課題 5-3 上記のプランニングのプログラムでは，ブロックの属性（たとえば色や形など）を考えていないので，色や形などの属性を扱えるようにせよ．ルールとして表現すること．例えば色と形の両方を

扱えるようにする場合，A が青い三角形，B が黄色の四角形，C が緑の台形であったとする．その時，色と形を使ってもゴールを指定できるようにする（”green on blue” や”blue on box”のように）

必須課題 5-4 上記 5-2, 5-3 で改良したプランニングシステムの GUI を実装せよ．ブロック操作の過程をグラフィカルに可視化し，初期状態や目標状態を GUI 上で変更できることが望ましい．

発展課題 5-5 ブロックワールド内における物理的制約条件をルールとして表現せよ．例えば，三角錐（pyramid）の上には他のブロックを乗せられない等，その世界における物理的な制約を実現せよ．

発展課題 5-6 ユーザが自然言語（日本語や英語など）の命令文によってブロックを操作したり，初期状態／目標状態を変更したりできるようにせよ．なお，命令文の動詞や語尾を 1 つの表現に決め打ちするのではなく，多様な表現を許容できることが望ましい．

発展課題 5-7 3 次元空間 (実世界) の物理的な挙動を考慮したブロックワールドにおけるプランニングを実現せよ．なお，物理エンジン等を利用する場合，Java 以外の言語のフレームワークを使って実現しても構わない．

発展課題 5-8 教科書 3.3 節のプランニング手法を応用できそうなブロック操作以外のタスクをグループで話し合い，新たなプランニング課題を自由に設定せよ．さらに，もし可能であれば，その自己設定課題を解くプランニングシステムを実装せよ．

私は必須課題 5-4 の GUI 実装を行ったため，それについて記述する．

2 必須課題 5-4

> 上記 5-2, 5-3 で改良したプランニングシステムの GUI を実装せよ．ブロック操作の過程をグラフィカルに可視化し，初期状態や目標状態を GUI 上で変更できることが望ましい．

私の担当箇所は，得られた結果を基にした GUI 本体の実装である．

2.1 手法

GUI で求められる出力が行えるように，以下のような機能を加えた．

- ブロック操作の過程を示す
- ブロック操作の過程をグラフィカルに可視化する
- 初期状態と目標状態を GUI 上から自然言語で変更できる
- 属性の決定を GUI 上から行える
- 操作に必要なオペレータと属性のデータを表示する

これらのプログラムを実装するためのデータ受け渡し部分は青山君が作成してくれるため，ここでは受け取ったデータを基にしてどのような方法で表示を行うかを考えながら実装を行った．

また今回は初めての GUI 作成であったため，データの受け渡し部分の作成を待つ間に，データを既に得ていると仮定した仮データを基にして GUI の検討を行った．

2.2 実装

今週は仮のデータを基にどのようにして GUI を作成するかを考えたため，それに関する実装を以下に示す．

まず，ブロック操作の過程をグラフィカルに可視化するために，CardLayout を用いてブロック状態の変化の実装を行うこととした．これに関するプログラムをソースコード 1 に示す．

ソースコード 1: CardLayout 関連プログラム

```
1 // 切り替え用ページの設定
2 ArrayList<JPanel> card = new ArrayList<>();
3 // ページの追加 (必要ページ数分だけ繰り返す)
4 card.add(new JPanel());
5 card.get(0).add(page1);
6 // パネルの設定
7 cardPanel = new JPanel();
```

```
8  layout = new CardLayout();
9  cardPanel.setLayout(layout);
10
11 cardPanel.add(card.get(0), "start");
12 cardPanel.add(card.get(1), "label1");
13 ...
14 cardPanel.add(card.get(card.size()-1), "finish");
15
16 // カード移動用ボタン
17 JButton firstButton = new JButton("First");
18 firstButton.addActionListener(this);
19 firstButton.setActionCommand("First");
20 JButton prevButton = new JButton("Prev");
21 prevButton.addActionListener(this);
22 prevButton.setActionCommand("Prev");
23 JButton nextButton = new JButton("Next");
24 nextButton.addActionListener(this);
25 nextButton.setActionCommand("Next");
26 JButton lastButton = new JButton("Last");
27 lastButton.addActionListener(this);
28 lastButton.setActionCommand("Last");
29
30 // カード遷移設定ボタンを一列の画面に
31 JPanel btnPanel = new JPanel();
32 btnPanel.add(firstButton);
33 btnPanel.add(prevButton);
34 btnPanel.add(nextButton);
35 btnPanel.add(lastButton);
36
37 // カード移動用ボタンの動作設定
38 public void actionPerformed(ActionEvent e){
39         // 画面遷移用
40         String cmd = e.getActionCommand();
41         if (cmd.equals("First")){
42                 layout.first(cardPanel);
43         }else if (cmd.equals("Last")){
44                 layout.last(cardPanel);
45         }else if (cmd.equals("Next")){
46                 layout.next(cardPanel);
47         }else if (cmd.equals("Prev")){
48                 layout.previous(cardPanel);
49         }
```

```
50 }
```

状態遷移をグラフィカルに可視化するため，CardLayout を用いた．これにより任意の数だけ状態遷移画面を作成でき，それらが全てボタンによって移動できるようになっている．さらにカードの一番初めと一番最後に移動するためのボタンを作成したことにより，いつでも初期画面に戻ることが出来る．

次に，実際にブロック操作の状態を表すためのプログラムをソースコード 2 に示す．

ソースコード 2: ブロックの状態を表すプログラム

```
1  // 結果表示用ページで共通の変数
2  row = 4;
3  col = 3;
4  ImageIcon arm = new ImageIcon("arm't.png");
5  ImageIcon icon1 = new ImageIcon("sanRt.png");
6  ImageIcon icon2 = new ImageIcon("enGt.png");
7  ImageIcon icon3 = new ImageIcon("squBt.png");
8  String armname = " ";
9  String icon1name = "A";
10 String icon2name = "B";
11 String icon3name = "C";
12 String no = "";
13 // 新規ページの作成
14 card.add(new JPanel());
15 JPanel page2 = new JPanel();
16 JLabel[][] p2Label = new JLabel[row][col];
17 page2.setPreferredSize(new Dimension(row*80, col*80));
18 page2.setBackground(Color.WHITE);
19 GridLayout page2layout = new GridLayout();
20 page2layout.setRows(row); // 行数
21 page2layout.setColumns(col); // 列数
22 page2.setLayout(page2layout);
23 // 中身の設定 (2ページ)
24 p2Label[0][0] = new JLabel(no);
25 p2Label[0][1] = new JLabel(arm);
26 p2Label[0][1].setText(armname);
27 p2Label[0][2] = new JLabel(no);
28 p2Label[1][0] = new JLabel(no);
29 p2Label[1][1] = new JLabel(no);
30 p2Label[1][2] = new JLabel(no);
```

```
31 p2Label[2][0] = new JLabel(no);
32 p2Label[2][1] = new JLabel(no);
33 p2Label[2][2] = new JLabel(no);
34 p2Label[3][0] = new JLabel(icon1);
35 p2Label[3][0].setText(icon1name);
36 p2Label[3][1] = new JLabel(icon2);
37 p2Label[3][1].setText(icon2name);
38 p2Label[3][2] = new JLabel(icon3);
39 p2Label[3][2].setText(icon3name);
40 for (int i = 0; i < row; i++) {
41         for (int j = 0; j < col; j++) {
42                 page2.add(p2Label[i][j]);
43         }
44 }
45 // 出力結果(各ステップ)の表示
46 JPanel list = new JPanel();
47 list.setLayout(new BoxLayout(list, BoxLayout.PAGE_AXIS));
48 for (int i = 0; i < results.size(); i++) {
49         JLabel now = new JLabel(results.get(i));
50         if (i == 0) {
51                 now.setBackground(Color.ORANGE);
52                 now.setOpaque(true);
53         }
54         list.add(now);
55         if (i == results.size()-1) {
56                 list.add(new JLabel(" "));
57                 list.add(new JLabel(" "));
58         }
59 }
60 card.get(1).add(list);
61 card.get(1).add(page2);
```

ブロックの状態を表すため，BoxLayout を用いて実装を行った．表示する図形については，新たに作成した png ファイルにあらかじめ図形を描き，これをアイコンとして挿入することで GUI 上での描画を行った．さらに，現在行われている描画が操作手順のどこに当たるのかを，全体の出力結果を用いて表した．

次に，初期状態と目標状態，属性の決定を行うためのプログラムをソースコード 3 に示す．

ソースコード 3: 初期状態と目標状態，属性の決定

```
1  // 初期状態選択画面
2  card.add(new JPanel()); // ページの追加
3  // 選択用ラジオボタンパネル
4  JPanel allRadio = new JPanel();
5  allRadio.setLayout(new BoxLayout(allRadio, BoxLayout.
       PAGE_AXIS));
6  // 色選択
7  JPanel p2 = new JPanel();
8  p2.setLayout(new BoxLayout(p2, BoxLayout.PAGE_AXIS));
9  radio = new JRadioButton[4];
10 radio[0] = new JRadioButton("Blue");
11 radio[1] = new JRadioButton("Green");
12 radio[2] = new JRadioButton("Red");
13 radio[3] = new JRadioButton("Yellow");
14 // ボタンのグループ化
15 ButtonGroup group = new ButtonGroup();
16 group.add(radio[0]);
17 group.add(radio[1]);
18 group.add(radio[2]);
19 group.add(radio[3]);
20 p2.add(new JLabel("Select Color"));
21 p2.add(radio[0]);
22 p2.add(radio[1]);
23 p2.add(radio[2]);
24 p2.add(radio[3]);
25 // 形状選択 (色選択と同様の操作を行う)
26 ...
27 // 属性決定ボタンの作成
28 allRadio.add(p2);
29 allRadio.add(Box.createRigidArea(new Dimension(10,10)));
30 allRadio.add(p3);
31 JButton set = new JButton("Set");
32 allRadio.add(Box.createRigidArea(new Dimension(10,10)));
33 allRadio.add(set);
34 // 属性入力用パネルの作成
35 JPanel attribution = new JPanel();
36 attribution.setLayout(new BoxLayout(attribution, BoxLayout
       .LINE_AXIS));
37 JPanel p4 = new JPanel();
38 p4.setLayout(new BoxLayout(p4, BoxLayout.PAGE_AXIS));
39 p4.add(new JLabel("Determine Attribution"));
40 JPanel attribute = new JPanel();
```

```
41 attribute.setPreferredSize(new Dimension(100, 350));
42 attribute.setBackground(Color.WHITE);
43 p4.add(attribute);
44 attribution.add(allRadio);
45 attribution.add(p4);
46 // 入力確認用ボタン
47 JPanel buttonpanel = new JPanel();
48 JButton button = new JButton("Search");
49 button.addActionListener(this);
50 buttonpanel.add(button);
51 // 手動入力用パネル
52 JPanel natural = new JPanel();
53 natural.setLayout(new BoxLayout(natural, BoxLayout.
      PAGE_AXIS));
54 // 入力 (setInitialState)
55 JPanel sI = new JPanel();
56 sI.setLayout(new BoxLayout(sI, BoxLayout.PAGE_AXIS));
57 JLabel setInitial = new JLabel("SetInitial");
58 JTextArea iArea = new JTextArea(9, 20);
59 JScrollPane iScroll = new JScrollPane(iArea);
60 String ii = "";
61 for(String i : initialState) {
62         ii += i + "\n";
63 }
64 iArea.setText(ii);
65 sI.add(setInitial);
66 sI.add(iScroll);
67 // 入力 (setGoal)(setInitialState と同様の操作を行う)
68 ...
69 natural.add(sI);
70 natural.add(Box.createRigidArea(new Dimension(10,5)));
71 natural.add(sG);
72 JPanel page1 = new JPanel();
73 page1.setPreferredSize(new Dimension(500, 350));
74 page1.setLayout(new BorderLayout(20, 5));
75 page1.add(attribution, BorderLayout.LINE_START);
76 page1.add(natural, BorderLayout.CENTER);
77 page1.add(buttonpanel, BorderLayout.PAGE_END);
78 card.get(0).add(page1);
```

start メソッドでは，ファイルの読み込みによるルールの構築のみを行う．また fetchRules では現在所持しているルールを RuleBase クラスか

らgetRulesメソッドで取得し返却を行うことで一覧表示が行えるようにした.

最後に，プログラムで使用するオペレータを表示するプログラムのソースコード4のように実装した.

ソースコード 4: オペレータの表示

```
1  // 最終ページ
2  card.add(new JPanel());
3  // オペレータの取得
4  ArrayList<Operator> operators = presenter.getOperatorList
       ();
5  JPanel tostring = new JPanel();
6  tostring.setLayout(new BoxLayout(tostring, BoxLayout.
       PAGE_AXIS));
7  tostring.setBackground(Color.WHITE);
8  int i = 1;
9  for (Operator operator : operators) {
10         tostring.add(new JLabel("●Operator" + i));
11         tostring.add(new JLabel("NAME: " + operator.
               getName()));
12         tostring.add(new JLabel("ADD: " + operator.getName
               ()));
13         tostring.add(new JLabel("DELETE: " + operator.
               getName()));
14         if (i < operators.size()) {
15                 tostring.add(new JLabel(" "));
16         }
17         i++;
18 }
19 JScrollPane scrollpane = new JScrollPane(tostring);
20 scrollpane.setPreferredSize(new Dimension(250, 310));
21 BevelBorder border = new BevelBorder(BevelBorder.LOWERED);
22 scrollpane.setBorder(border);
23 // 編集用ボタン
24 JPanel edit = new JPanel();
25 edit.setLayout(new BoxLayout(edit, BoxLayout.PAGE_AXIS));
26 edit.add(new JButton("追加"));
27 edit.add(Box.createRigidArea(new Dimension(10,5)));
28 edit.add(new JButton("編集"));
29 edit.add(Box.createRigidArea(new Dimension(10,5)));
30 edit.add(new JButton("削除"));
```

```
31 JPanel tostring2 = new JPanel();
32 tostring2.setPreferredSize(new Dimension(200, 300));
33 tostring2.setBackground(Color.WHITE);
34 // ページへの追加
35 card.get(card.size()-1).add(scrollpane);
36 card.get(card.size()-1).add(edit);
37 card.get(card.size()-1).add(tostring2);
```

ここではプログラムで使用したオペレータの表示を行う．オペレータの取得は，班の人が作成してくれたオペレータ情報を取得する getOperatorList を利用した．さらに，オペレータの編集用にボタンを作成したが，話し合いの結果プログラム中でオペレータに対しての編集は行わないこととしたため，このボタンは使用しないことになった．

2.3 実行例

現時点で作成したプログラムの実行結果を順に示す．

実行すると初めに初期状態・目標状態・属性の決定画面になる (図 1)．

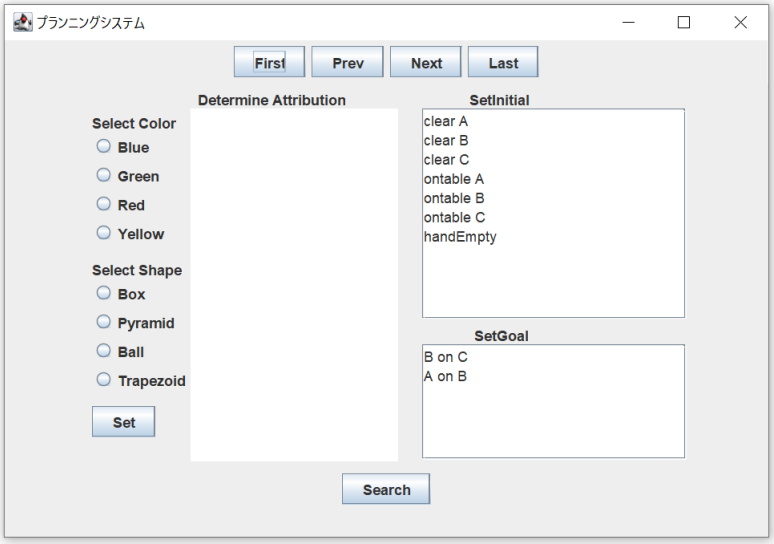

図 1: 初期状態と目標状態，属性の決定画面

この状態で Next ボタンを押すと次の画面に遷移し，ブロック操作の過程の描画画面となる．この状態で Next ボタンを押すと次の状態へ，Prev ボタンを押すと一つ前の状態へ遷移する様子が確認できる．(図 2，次ページ図 3，次ページ図 4)．

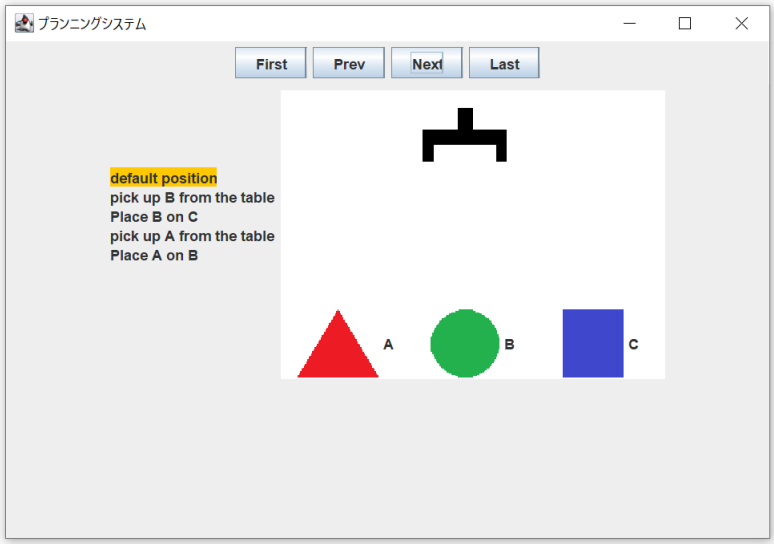

図 2: ブロック操作の過程 1

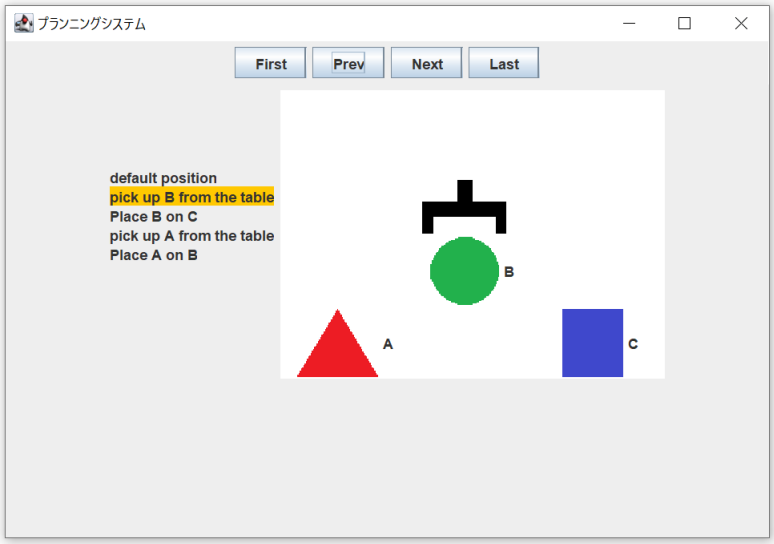

図 3: ブロック操作の過程 2

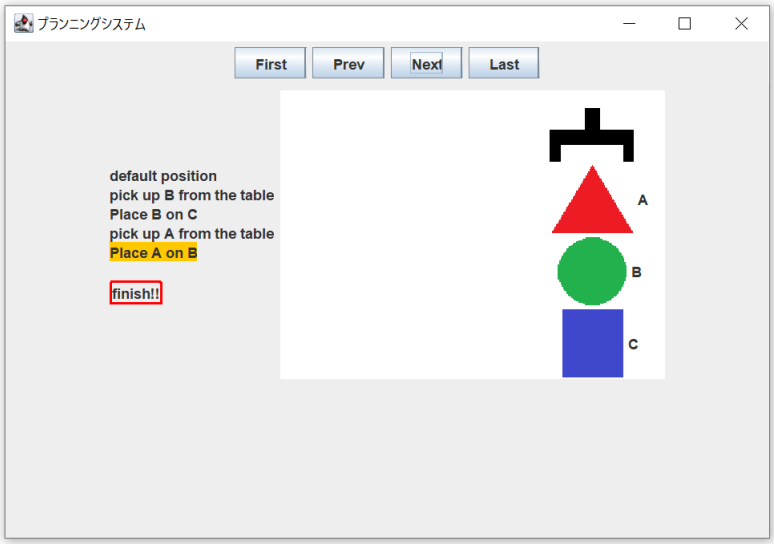

図 4: ブロック操作の過程 3

さらに Last ボタンを押すと，一番最後のページに移動することが出来，ここではオペレータの情報が示されている (図 5).

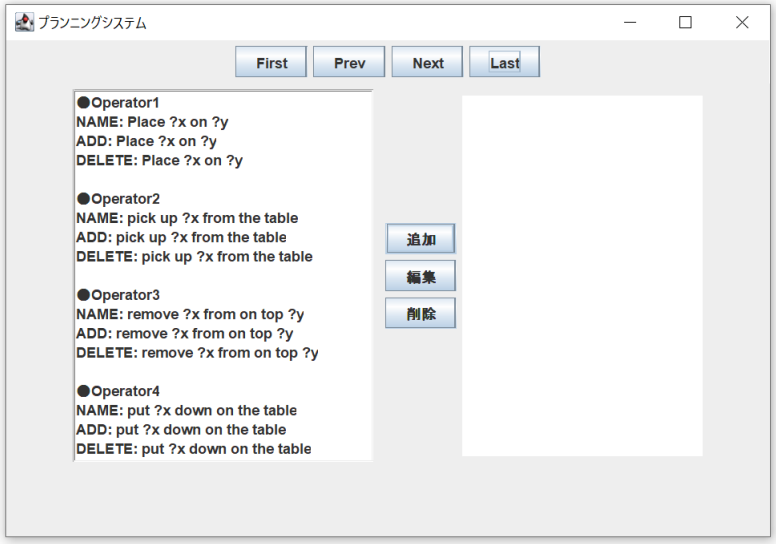

図 5: オペレータの情報

2.4 考察

状態遷移をグラフィカルに可視化するため，ページを移動するたびに一つずつ動作が進んでいく状態が理想であると考えた.

また，状態空間を描画するために，BoxLayout で空間を座標のようにとらえることを考えた．初めはプログラム自身で図の描画を行う予定であったが，三角形などの描画は座標を指定してプログラムをしなければならず，その場合はレイアウトマネージャーを利用できなくなってしまう．これでは GUI の実装が困難であるため，今回はプログラムでの図形の描画は行わないこととした．そのため，新しく自分で png 画像を作成し，これを取り込んでアイコンとして利用することで，レイアウトマネージャーを利用して図形を描画することが可能となった.

しかし，この方法を用いると，取り込んだ図に対して操作を行うことは出来ないことが分かった．そのため，画像の上に重なるように図形の名称 (A,B,C 等) を入れることや，取り込んだ画像の色を上書きすること

が出来ないため，今回は図形の横に名称を記載し，画像は色違いのものをはじめから用意しておくことでこの問題を解決した．

3　感想

GUI は初めて作成したため慣れない部分も多く大変だったが，少し理解を深められた．

まだ完成には程遠いので，今週も頑張りたい．

参考文献

[1] 新谷虎松『Java による知能プログラミング入門』コロナ社，2002 年．

[2] Let's プログラミング　Swing を使ってみよう，`https://www.javadrive.jp/tutorial/`（2019 年 12 月 3 日アクセス）．