How to create sliding puzzles on GeoGebra
– A tutorial of GeoGebra Script –

藤本 光史（福岡教育大学）*

概要

GeoGebra でアプリを作成しようとすると、通常のプログラミング言語との違いに戸惑うことが多々ある。これは GeoGebra スクリプトに様々な癖があるからである。本稿は GeoGebra 初心者がスクリプトを用いてアプリを作成する際の手助けになるように、スライドパズルの作成方法を通してその部分の解説を試みたものである。

1　はじめに

GeoGebra は動的幾何学ソフトウェアの中でも非常に使いやすいものの一つである。最初から組み込まれているツールは豊富であり、通常の利用であればこれだけで事足りるであろう。しかし、GeoGebra で少し凝った数学教材を作成しようとすると、新しい機能を割り当てたツールを作成したくなる。これは「ツール」メニューの「新規ツールの作成」から可能であり、作成したツールはデフォルトのツールバーの右端に出現するツールボックスに登録されていく。この方法だとすべての新規ツールが同じツールボックスに収まるため、多くの新規ツールを作成した場合はわかりにくくなってしまう。そこで、本稿では「新規ツールの作成」は行わず、描画エリア内にボタンを設置するスタイルを採用する。また、本稿で作成する教材にはタブレット端末での利用を考えて 15 パズルを選んだ。

2　15 パズル完成までのステップ

15 パズルを作成するまでのステップは以下の 10 ステップである。

1. 格子点の生成
2. ピースの生成
3. ピースへの数字の貼り付け
4. クリックによるピースの移動
5. 誤動作の防止

* fujimoto@fukuoka-edu.ac.jp

6. 初期化ボタンの作成
7. 互換ボタンの作成
8. シャッフル機能
9. 正解判定機能
10. 効果音の設定

以下で各ステップの作業内容を詳しく述べるが、その前の準備としてオブジェクトのラベルを非表示に設定しておこう。

「オプション」メニュー → 「ラベル付け」 → 「新規オブジェクトには付けない」

これは 15 パズルを作成するまでにたくさんのオブジェクトを生成しなくてはならず、全オブジェクトのラベルが表示されると非常に煩わしいためである。

2.1 【ステップ 1】格子点の生成

座標が (3,5) の点を作りたい時は、入力バーに単に `(3,5)` と入力するだけで（A などの）ラベル付きの点が (3,5) に作成される。複数の点を格子状に作りたい時は以下のように入力する。

```
Sequence(( Mod(i,4), Div(i,4) ), i, 0, 15)
```

Sequence コマンドによって、(Mod(i,4), Div(i,4)) を座標とする点を（変数 i を 0〜15 まで変化させながら）生成する。Mod(i,4) は i を 4 で割った余りを、Div(i,4) は i を 4 で割った商を求めるので、4 × 4 の格子点になる。しかし、この格子点にはラベルが付かない。

P0〜P15 というラベルを付けた 4 × 4 の格子点を生成するには、入力バーに以下のように入力する。枠外の数字は行番号[*1]であり、入力してはいけない。

```
1 Execute(Sequence("P" + (i) + "= (" + 2*Mod(i,4) + "," +
    (-2*Div(i,4)) + ")  ", i, 0, 15))
```

Sequence コマンドによって、文字列リスト

```
{"P0 = (0,0)", "P1 = (2,0)", ⋯, "P15 = (6,-6)"}
```

を生成し、それを Execute コマンドに渡すことでリストの各要素が実行され、ラベル付きの格子点が作られる。このように Execute と Sequence を組み合わせることにより、反復処理が行える。

[*1] これ以降、紙面の関係で行の折り返しが必要な時は行番号を付けることにする。行番号が付いている行は、改行しないで入力することに注意されたい。

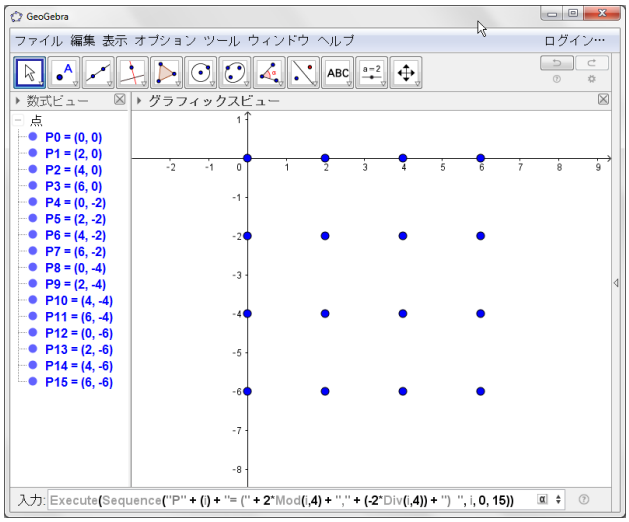

図 1　ラベル付き格子点の生成

2.2　【ステップ 2】ピースの生成

15 パズルには 15 個の正方形ピースが必要である。正方形を作成する方法は複数あるが、ここでは RigidPolygon コマンドを用いる。RigidPolygon コマンドによって作られた多角形は始点の位置を変更することでオブジェクトの移動が可能であるため、15 パズルのようにオブジェクトの移動が必要な際には便利である。ステップ 1 で作成した点 P0〜P15 を左上に持つ幅 2 の正方形 q0〜q15 を生成するために、入力バーに以下のように入力する。

```
1 Execute(Sequence("q"+(i)+"=RigidPolygon(P"+(i)+",P"+(i
    )+"+(0,-2),P"+(i)+"+(2,-2),P"+(i)+"+(2,0))",i,0,15))
```

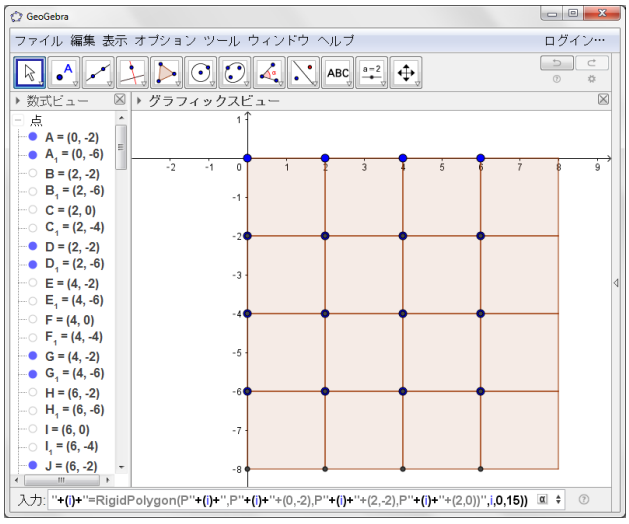

図 2　ピースの生成

2.3 【ステップ 3】ピースへの数字の貼り付け

ステップ 2 で作成したピースに 1〜15 の数字を「見出し (Caption)」として貼り付けたい。これを行うには、入力バーに以下のように入力する。

```
1 Execute(Sequence("SetCaption(q"+(k)+",Text("+(k+1)+"))",k
  ,0,15))
```

そして、数式ビューから 4 角形を右クリックし、「ラベルの表示」を有効化する。

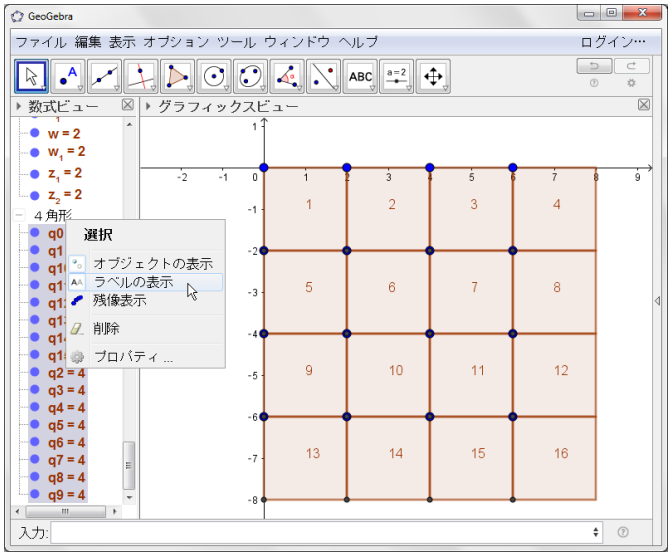

図 3　数字の貼り付け

ただし、この状態では空白ピースにも 16 という数字が割り当てられているので、空白ピースのラベルは非表示にする。

2.4 【ステップ 4】クリックによるピースの移動

ピースをクリックすることで移動できるようにしよう。まず、入力バーから変数 flag を作成し、初期値を 0 にする。(他のオブジェクトからも参照される変数は、このように入力バーから生成しておく。)

```
flag=0
```

ピースが移動可能なのは空白がピースの上下左右にある時だけである。ピースをクリックした際、空白が右にある場合は変数 flag の値を 1 に、上にある場合は 2 に、左にある場合は 3 に、下にある場合は 4 に設定する。この変数 flag の値を元に、移動するピースの（左上の点の）座標と空白ピースの（左上の点の）座標を入れ替える。これを行う GeoGebra スクリプトが以下である。

```
1 Px=x(P14)
2 Py=y(P14)
3 If((Px,Py)==(x(P15)-2,y(P15)),SetValue(flag,1),(Px,Py)==(x(
    P15),y(P15)-2),SetValue(flag,2),(Px,Py)==(x(P15)+2,y(P15
    )),SetValue(flag,3),(Px,Py)==(x(P15),y(P15)+2),SetValue(
    flag,4),SetValue(flag,0))
4 SetValue(P14,(Px+if(flag==1,2,flag==3,-2,0),Py+if(flag
    ==2,2,flag==4,-2,0)))
5 SetValue(P15,(x(P15)+if(flag==1,-2,flag==3,2,0),y(P15)+if(
    flag==2,-2,flag==4,2,0)))
```

これは、数字 15 が書かれたピース（左上の点は P14）のためのスクリプトであり、そのピースを右クリックして「プロパティ」を選択後、「スクリプティング」タブの「On Click」欄に入力する。大変面倒であるが、（空白ピースを除く）全ピースの「On Click」欄にこのスクリプトをコピーし、「P14」の部分（３箇所ある）を各ピースの左上の点の名前に変更[*2]する。通常のオブジェクト指向プログラミングであれば、自身のオブジェクトを指す予約語として `this` を使用できるが、GeoGebra スクリプトではサポートされていない。よって、このような書き換えが必要となる。

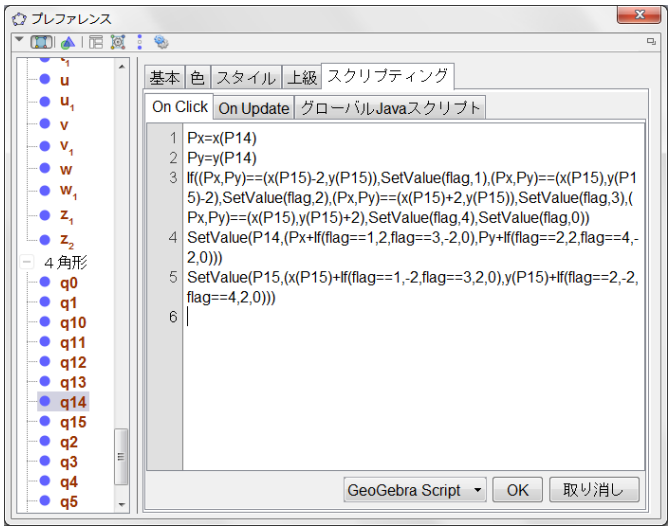

図 4　スクリプトを入力

2.5　【ステップ 5】誤動作の防止

ステップ 4 でクリックによりピースが移動できるようになったが、ピースをドラッグすることでも動いてしまう。このようなプレイ中の誤動作を防ぐためにピースが動かないように設定しよ

[*2] 数字 1 が書かれたピースでは「P14」の部分を「P0」に、数字 2 が書かれたピースでは「P1」に、という書き換え作業を行う。

う。このためには各ピースのプロパティの「オブジェクトの固定」を有効化すればよい。以下のように、入力バーからすべてのピースのプロパティを一気に変更できる。

```
Execute(Sequence("SetFixed(q"+(k)+",true)",k,0,15))
```

各ピースの左上の点も表示されたままだとドラッグで動いてしまう危険があるので、数式ビューから点を右クリックし、「オブジェクトの表示」を無効に変更しておこう。この他に、マウスポインタがオブジェクト上にあるときツールチップが表示されるが、これもプレイ中は目障りなのでオフにしよう。数式ビューから 4 角形と線分を右クリックして「プロパティ」を選択後、「上級」タブの「ツールチップ」を off に変更する。

2.6 【ステップ 6】初期化ボタンの作成

移動したピースの位置を初期状態に戻す初期化ボタンを設置しよう。ステップ 9 の正解判定機能に必要となるため、初期化ボタンを押したかどうかをブーリアン変数 rflag に記憶させる。そのために、入力バーから変数 rflag を作成し、初期値を false にする。

```
rflag=false
```

次に、Reset というラベルの付いたボタンを作成する。そのために、入力バーに以下のように入力する。

```
reset=Button("Reset")
```

そして、描画エリアの左上に出現した Reset ボタンをマウスで適切な位置に移動し、右クリックから「プロパティ」を選択後、「テキスト」タブで大きさを「大」に変更する。さらに、「スクリプティング」タブの「On Click」欄に以下の GeoGebra スクリプトを入力する。

```
1 Execute(Sequence("SetValue(P" + (i) + ", (" + 2*Mod(i,4) +
    "," + (-2*Div(i,4)) + "))  ", i, 0, 15))
2 SetValue(rflag,true)
```

2.7 【ステップ 7】互換ボタンの作成

ステップ 8 のシャッフル機能で必要となる（ピースの交換を行う）互換ボタンを作成しよう。そのために、入力バーに以下のように入力する。

```
transpose=Button("Transpose")
```

そして、描画エリアの左上に出現した Transpose ボタンを右クリックして「プロパティ」を選択後、「スクリプティング」タブの「On Click」欄に以下の GeoGebra スクリプトを入力する。

```
Idx=Sequence(i,i,0,14)
Ia=RandomElement(Idx)
Ib=RandomElement(Remove(Idx,{Ia}))
Pz=(0,0)
Execute({"SetValue(Pz,P"+(Ia)+")","SetValue(P"+(Ia)+",P"+(
  Ib)+")","SetValue(P"+(Ib)+",Pz)"})
Delete(Pz)
```

ここでは、リスト Idx=$\{0,1,2,\ldots,14\}$ から RandomElement コマンドにより 2 個の数 a,b をランダムに選び、点 Pa と点 Pb の座標を交換することで、ピースの入れ替えを行っている。

このボタンはプレイ中は使用しないので、正常に動作することが確認できたら、右クリックから「オブジェクトの表示」を無効に変更しておく。

2.8 【ステップ 8】シャッフル機能

15 パズルの数字の配置には元に戻せる「解ける配置」と元に戻せない「解けない配置」がある。数字の並びが偶置換であることが「解ける配置」であるための必要十分条件になっている。ここでは、ステップ 7 で作成した Transpose ボタンを偶数回[*3]実行するシャッフルボタンを作成しよう。そのために、入力バーに以下のように入力する。

```
shuffle=Button("Shuffle")
```

そして、描画エリアの左上に出現した Shuffle ボタンをマウスで Reset ボタンの下に移動し、右クリックから「プロパティ」を選択後、「テキスト」タブで大きさを「大」に変更する。さらに、「スクリプティング」タブの「On Click」欄に以下の GeoGebra スクリプトを入力する。

```
RunClickScript(reset)
Repeat(30,RunClickScript(transpose))
```

GeoGebra スクリプトにはプログラミング用の「関数」機能がないので、別の仕事を行うオブジェクト（今の場合は Transpose ボタン）を作成し、それを RunClickScript コマンドで実行することで代用している。

2.9 【ステップ 9】正解判定機能

15 パズルが完成した際、**Congratulations!** と表示する正解判定機能を作成しよう。まず、入力バーから文字列変数 result を作成し、点 P0〜P15 の初期状態の座標をリスト OL に格納しておく。

[*3] ここでは実行回数を 30 に設定する。

```
result=""
OL={}
Execute(Sequence("SetValue(OL,Append(OL,("+2*Mod(i
  ,4)+","+(-2*Div(i,4))+")))", i, 0, 15))
```

そして、数式ビューから result を右クリックして「プロパティ」を選択後、テキストの大きさを「大」に、色を「青」に変更する。

ピースが移動する時は必ず空白のピースの位置が変更されるので、正解を判定する以下のスクリプトは空白ピースの「スクリプティング」タブの「On Update」欄に記述すればよい。

```
RL={}
Execute(Sequence("SetValue(RL,Append(RL,(x(P"+(i)+"),y(P"+(
  i)+"))))", i, 0, 15))
If(flag>0 && rflag==false && RL==OL, SetValue(result,"
  Congratulations!"))
SetValue(rflag, false)
```

ここでは、現在の点 P0〜P15 の座標をリスト RL に格納し、初期状態リスト OL と比較することでパズルが完成したかどうか判定している。

パズル完成後、Reset ボタンをクリックした際に **Congratulations!** の文字列が消えるように、Reset ボタンの「スクリプティング」タブの「On Click」欄の最後に以下を追加する。

```
SetValue(result,"")
```

2.10 【ステップ 10】効果音の設定

ピースが移動する時にクリック音を、パズルが完成した時には正解音を出すようにしよう。まず、正解音用の音声ファイルを準備する。GeoGebra は mp3 形式の音声ファイルを再生できるので、フリー素材サイト [3] などから正解音に相応しいものを入手する。ここでは、`crrect_answer3.mp3` というファイルを準備し、C ドライブの直下[*4]に保存した。そして、空白ピースの「スクリプティング」タブの「On Update」欄を以下のように修正する。

[*4] 相対パスによるファイル場所の指定方法が不明だったため、絶対パスで指定し易いように「C ドライブの直下」としたが、絶対パス名がわかる場所ならどこでもよい。

```
RL={}
Execute(Sequence("SetValue(RL,Append(RL,(x(P"+(i)+"),y(P"+(i)+"))))", i, 0, 15))
If(flag>0 && rflag==false && RL==OL,SetValue(result,"Congratulation!"))
#次の行を追加
If(flag>0 && rflag==false && RL==OL,PlaySound("C:/crrect_answer3.mp3"))
SetValue(rflag, false)
#次の行を追加
If(flag != 0, PlaySound(sin(440 2Pi x), 0, 0.1))
```

ここでは、クリック音として、GeoGebra に「ラ」の音（440 Hz）を直接出力させている。以上の全 10 ステップにより、GeoGebra で次のような 15 パズルを作成することができた。

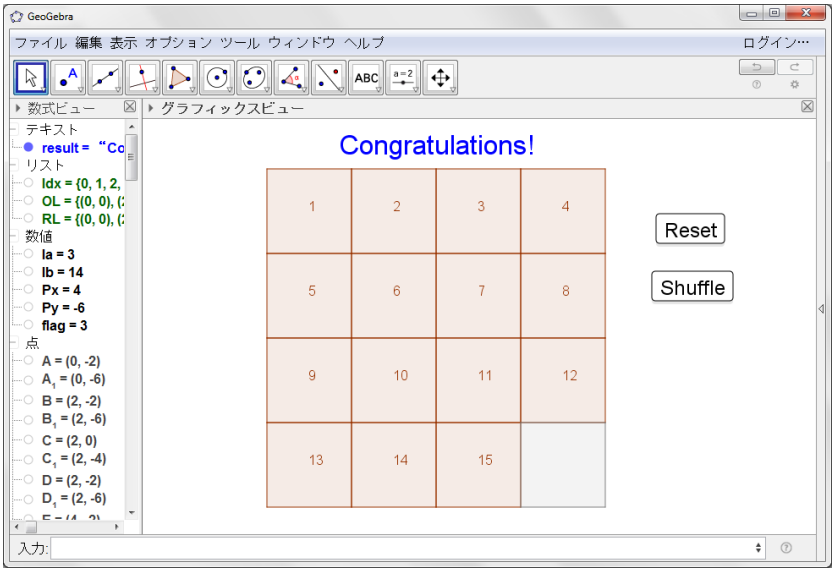

図 5　完成した 15 パズル

2.11　演習課題

ここまでの学習で GeoGebra スクリプトの基本が身に付いたことだろう。この後、次のような課題に取り組むことで更に理解が深まるはずである。

課題 1　カウンタを作成し、移動回数を表示できるようにしよう

課題 2　数字ラベルの代わりに画像を貼り付けよう

課題 3　解答ボタンを用意し、解法を表示できるようにしよう

3　おわりに

本稿では、GeoGebra スクリプトを用いて 15 パズルの作成を行い、GeoGebra で教材アプリを作成するための基本を解説した。GeoGebra の少し凝った教材は JavaScript を用いる場合が多いが、GeoGebra スクリプトだけでもこのような教材アプリが作成できることがわかっていただければ幸いである。なお、研究集会での講演スライドは [4] から入手可能である。また、本稿で解説したスクリプトをまとめたテキストファイルは [5] から入手できる。実際にスクリプトの入力を行う際は、このファイルからコピーすると便利である。

最後に、本チュートリアルを作成する過程で GeoGebra スクリプトに関して改善が必要と思われる事項がいくつかあったので、ここに挙げておく。

- 文の途中で改行できないため、長く読みづらくなってしまう。
- if コマンドは C 言語の switch 文のような多分岐処理が可能であるが、処理内容に複文（複数コマンドの実行）に対応していない。
- ループ文がないため、Sequence と Execute コマンドを併用する必要がある。
- 関数が作成できないので、関数に相等する別のオブジェクトをボタンなどで作成し、それを呼び出す必要がある。
- 自身のオブジェクトを指す予約語 `this` がサポートされていない。
- オブジェクトに割り当てるスクリプトを入力バーから入力できない。
- オブジェクトのラベル位置を固定する機能がない。
- 音声ファイルなどの外部ファイルの相対パスによる指定方法がわからない。

これらの GeoGebra スクリプトの欠点を補うために、GeoGebra では JavaScript も利用できるようになっていると思われる。しかし、小学校や中学校で GeoGebra を利用していく際、複数のスクリプト言語の使用は混乱を招くであろう。JavaScript を使用せず GeoGebra スクリプトだけで全てのことができるようになることが望まれる。

参考文献

[1] Scripting Commands, `https://wiki.geogebra.org/en/Scripting_Commands`

[2] Tutorial: Scripting: Functions and subroutines, `https://wiki.geogebra.org/en/Tutorial:Scripting:Functions_and_subroutines`

[3] 無料効果音素材, `http://taira-komori.jpn.org/freesound.html`

[4] 藤本光史, 講演スライド, `http://ww1.fukuoka-edu.ac.jp/~fujimoto/geogebra2017/fujimoto_slide.pdf`

[5] 藤本光史, GeoGebra で 15 パズル (スクリプトメモ), `http://ww1.fukuoka-edu.ac.jp/~fujimoto/geogebra2017/fujimoto_memo.txt`