2000年度卒業ゼミナール報告書

Logo言語の教育場面での応用

296108　服部真一
297027　上野明日香
297709　今井由果
297710　岩崎悦子
297726　小林佳子
297728　坂倉彰仁
297759　山田和代

三重大学教育学部

目 次

Chapter1

全般に関する報告

1.1　テキストとした文献と活動

この卒業研究は,

正田　良『Logoで知る認知科学』東京電機大学出版局, 1999

をテキストとし, 本学の 正田　良 助教授を指導教官として行った. 前期は, このテキストの内容の基本的な部分を学習した. インタネットのサイトである, http://www.logob.com/users/rio/ に収められている譜例をコピー・アンド・ペーストする形でパソコンで確認をとりながら読み進めた.

処理系としては, 当初フリーウエアである Logo 坊を用いたが, 動作速度などの関係で Logo Writer for Windows 及び, その無料体験版を用いて開発, 研究を進めるものもいた.

この講読を基礎に, 各自の興味を持つ領域を選択し, その範囲で夏休みにプログラムを作ることとなった.

1. プログラムに関する質問をする.

2. プログラムを持ち寄り相互批評する.

という目的で「登校日」を設けたが, 他は自宅でのプログラミング, 及び, 指導教官とのメールでの質問のやり取りでもっぱら研究を進めた. この報告書の第2章以下は, この各自の分担による執筆によるものである.

後期では, この各自の分担部分の原案に対して相互批評を行ない, より高い完成度を求めることとした.

1.2 逐次的活動報告

1.2.1 前期

主な前期の活動内容はテキストの講読であるので，その内容の詳細はテキストに譲ることとし，繰り返さないが，対応する章の名称などを記す．

4月14日 Turtle Graphics

4月21日 海岸へのロケーションによるHP作成への準備．if，プログラムの呼び出し，print．

4月28日 インタネット実習．データ構造．再帰的定義．make，word，list，outputなど．

5月12日 新規のパソコンや機器室の整備．

5月19日 第3章 自由落下．Quick Basic英語版のサンプルプログラムゴリラ．

5月26日 第3章 Dyna Turtle．処理系の違い．

6月 2日 第7章 会話をするコンピュータ．

6月 9日 課題例の提示．

6月16・23日 自主的研究（教育実習期）．

6月30日 自主的研究の報告．TEX実習（名刺作り）．

7月 7日 プログラムの概要．演習問題[1]の解答(1)．

7月14日 演習問題の解答(2)．

当初の選択した領域から研究が進むにつれて，やや異なった研究に変わって行った分担があるが，概ね全体としてのバランスは保たれたと思える．また，その変更は研究の深まりとして妥当なものであった．提示された課題例と当初の選択者とを表の形で記す．

[1] テキストのp.21

1	手作りの数式処理	
2	巨大数	
3	数列と順列	今井
4	行列	坂倉
5	系統的試行錯誤	
6	ダイナタートル	上野
7	いろいろな曲線	小林
8	アニマル	岩崎
9	おおかみと羊	
10	2体問題	
11	ティーチングマシン	服部・山田
12	ファイルの処理	

1.2.2 8月と9月

大学での普通の講義が行われないこの時期は各自の研究を主にしたが, 以下のことを取り決めた.

1. 週当たり実効的な研究時間が平均8時間以上はあるようにする.
2. 8月10日までに指導教官へ近況を報告する.
3. 8月30日を登校日として, 自作プログラムのできについて相互に披露する.
4. 9月14日を登校日として, 後期の初めに提出する各自の分担部分原案の提出に向けて質問などを行なえるようにする.

残念ながら, 第1項の実行が必ずしも全員が取り決めを守ったとは言えず,「登校日」の意味が後日にずれる傾向が少なからずみられた. また, 10月1日に

`http://math1.edu.mie-u.ac.jp/~rio/semi2000/oct/`

を作ったが, これは各自の自己評価にも役立った.

1.2.3 後期

次のスケジュールによった.

10月13日 分担部分原案提出. 画面取り込みの復習. 相互批評の順番の決定.

10月20日 相互批評(1)[2]

10月27日 相互批評(2).

[2] 1回に2名を標準とし, 再度の批評もありうる.

11月10日 相互批評(3).

11月17日 相互批評(4).

11月24日 相互批評(5). 原稿の提出.

12月 1日 総合的な調整.

12月 8日 TEX実技.

12月15日 発表の企画, 技法の工夫.

1月12日 最終調整. 製本.

1.3 分担の大略

既に述べたように, 第2章以下は各自分担した研究によるものである. 執筆者（学生番号順）の対応する部分を記す.

296108 服部真一 4章
297027 上野明日香 3章
297709 今井由果 6章
297710 岩崎悦子 2章
297726 小林佳子 5章
297728 坂倉彰仁 7章
297759 山田和代 8章

1.4 活動の評価

指導教官でおられる正田 良 助教授の言葉を借りてここでは叙する.

> 本来なら, 半期科目程度でLogoの基本的な文法事項を学んでから着手するのが卒業研究としては妥当な方法であろう. しかし, 私が着任後間もないことや, 科目担当上の理由もあって, この研究では, まったくLogoを知らない状態から始めた学生がほとんどであった. そのため, 前期には, その補足的な授業を行わざるを得なかったことは, この期の卒業研究生にとってのハンディとなったことを, 彼らの名誉の為に記さねばならないだろう.
>
> しかしながら, そのハンディは, せいぜい6月中に克服されるべきことで, それ以降についての遅滞を全部これに帰することは不適切である. 特に, 夏休み中の要求に対して誠実であったならば, ほとんどの部分は, もっと完成度の高いものとなったことは, 指導教官である私, 正田の名誉のために是非記すべきことである.

よって，当論文の各自の分担部分は，やはりそれがそれとして完成度を評価されるべきものであろう．

また，卒業研究生の感想として次のことを付け加えたい．

- 今までのことを振り返ってみると，最初は，初めて学ぶことは Logo 言語だけだと思ったのですが，卒論は TEX を使ったので，いろんな事をしたな〜と思いました．

  卒論に関して，プログラムは何とか後期に入るまで終えて，卒論というように取りかかったのですが，その卒論が意外と曲者でかなり悪戦苦闘しました．しかし，今までの悩んだことや考えたことで得たものは，今の私の知識となったに違いありません．

- Logo 言語でのプログラムは小学生にもできて簡単だと聞いていましたが，実際プログラムをしてみると意外と難しかったです．そして，Logo より難しかったのが TEX です．慣れた頃にゼミが終わってしまうのは残念です．

- 1 年間，Logo というプログラミング言語を勉強してきて，最初のうちはいざプログラムを組もうとしても何から手をつけていいのかわからず，そしてプログラムの理論を頭で考えてても，いざプログラムを作ろうとしてもどのようにプログラムを組んでいいのかわからず悪戦苦闘の連続であったがなんとかここまでたどり着きました．

- このゼミを通して，授業の中でパソコンを生かすことを考えることで，パソコンを使うことで授業の面白さがより引き出せることを知りました．そして，これからも授業の組み立てを考えるときにパソコンを使っていきたいと思いました．

- 研究材料ははじめ軽い気持ちで選んだのですが，研究していくうちにどんどん面白くなり，この研究を選んで良かったです．自分の一生の中で最後の勉強かもしれないので，いい思い出になったと思います．

- 最初 Logo をやると聞いたときには，一体どんなものができるか想像がつきませんでした．実際に出来上がったものや，他の学生の卒論などを見て「Logo ってこんなに色んなことが出来るんだ．」とさまざまな可能性を見せてもらった気がします．

  Logo も難しかったのですが，TEX も融通がきかなかったりして苦労しました．

- 大学の最後の有終の美を飾るために，このゼミでお世話になりました．数学教育を考えるのは予想外に難しく，さらに今まで扱かったことのないコンピューターでの研究であったため，最初は混乱ばかりでしたが，なんとか，形にすることができました．まだまだ至らない論文ではありますが，将来，教師になる身として，これから数学教育を考えていく基礎はで

きたかなと思います．一緒にゼミで勉強した学生のみなさんに，心から感謝したいとおもいます．どうもお世話になりました．

なお，この卒業研究に当たっては，各自のパソコン操作，及び，相互批評などのプレゼンテーションが必要不可欠なものであった．しかし，年度当初は必ずしもそのための環境が充分とは言えなかった．このような環境整備に対して，数学教室が非常に迅速に環境整備をされたことに対して深く感謝の意を表したい．（卒業研究生一同）

Chapter2

アニマルとつきあってみて

私がLogoを学習した中で，一番興味を持ったのがアニマルである．私にとってアニマルは最初お遊びのようなプログラムに思え，私の苦手な勉強とはかけ離れた一つの遊び道具のように感じられた．そんな感覚でこのアニマルを研究しようと考えていたので，はじめはプログラム自体の改良ばかりを考えていたが，実際にアニマルを動かして，データを読み込ませて試行錯誤，試行的動作を繰り返していると，面白い発見が多く，ただの遊び道具という感覚がなくなった．ここではそのアニマルを研究して気づいた点を取り上げていく．

2.1　アニマルのプログラム

以下は私が研究したアニマルのプログラムである．テキスト『Logoで知る認知科学』に載っているプログラムを私なりに使い易く少し改造したものである．（右側の(1)～(3)の番号は私が改造した部分を示す．）

```
to Animal
 make "World ［ひとみちゃん］
 Stop_or_Continue
end
to Stop_or_Continue
 insert "パパがだれのことかんがえているかあててあげようか？
 if YesOrNo ［ Guess :World stop ］
 insert "じゃあ，またあとであそうぼうね．これおぼえとく？
 if YesOrNo ［ savename "World ( word AskFile ". nam ) ］
end
to AskFile
 print "どのファイルへセーブしますか？
 output first readlist
end
to Guess :Scope
 local ［ RightAns Ques ］
 if ( count :Scope ) = 1
```

```
  [ insert ( sentence :Scope "でしょ？ )
    if YesOrNo
    [ print "ひとみちゃんおりこうでしょ．じゃあいまね,
      Stop_Or_Continue
    ]
    print "まちがえちゃった．じゃあだあれ？
    make "RightAns readlist
    show :Scope                                          }
    show :RightAns                                      }
    if :RightAns = :Scope                                } -------------(1)
    [ print "あれ？あってるじゃん．じゃあいまね,       }
      Stop_Or_Continue                                    }
    ]                                                     }
    print ( sentence :RightAns "がｙで,
    :Scope "がｎになる質問なあに？ )
    make "Ques readlist
    GetSmarter :Scope :Ques :RightAns
    [ print "じゃあいまね,
      Stop_Or_Continue ]
  ]

 insert ( sentence ( first butfirst :Scope ) "? )
 if YesOrNo  [ Guess last :Scope ]
 Guess first :Scope
end
to YesOrNo
 local "C
 make "C readchar
 print ( word "| | :C )
 if :C = "y  [ output 1 = 1 ]
 if :C = "ｙ [ output 1 = 1 ]  ---------------------------(2)
 if :C = "n  [ output 0 = 1 ]
 if :C = "ｎ [ output 0 = 1 ]  ---------------------------(3)
 print  [ y か n じゃないとやだ！ ]
 output YesOrNo
end
to GetSmarter :WrongAns :Ques :RightAns
 make "World Swap ( list :WrongAns :Ques :RightAns ) :WrongAns :World
 print ( list "$$$__ :World )
end
to Swap :New :Old :In
 local "Genkou
 print ( list "###_Doing__Swap :New :Old :In )
 if empty? :In  [ output P  [] ]
 if word? first :In
 [ if :In = :Old
   [ output P :New ]
   output P :In
 ]
 make "Genkou fput ( Swap :New :Old ( first :In ) )
 ( Swap :New :Old ( butfirst :In ) )
 output P :Genkou
end
to P :L
 print ( list "###_output_is :L )
```

```
 output :L
end
```

2.2　プログラムについて

アニマルのプログラムについての概要はテキスト通りなので，ここでは私が改造した部分の説明のみ取り上げる．以下にこのプログラムによる会話の状況を簡単に示す．

A『パパが誰のこと考えてるかあててあげようか？』－－－y

B『ひとみちゃんでしょ？』－－－－－－－－－－－－－－－n

C『まちがえちゃった．じゃあだあれ？』－－－－－－－ママ

D『ママが y で，ひとみちゃんが n になる質問なあに？』

E －－－－－－－－－－－－－－－－－－－－－－－－－おとな

F『じゃあいまね，パパが誰のこと考えてるかあててあげようか？』－－－n

G『じゃあ，あとでまた遊ぼうね．これ覚えとく？』－－y

（終了）

2.2.1　(1) について

アニマルは利用者とコンピュータがほとんどYesを表すｙとNoを表すｎで答えていく対話のプログラムである．きちんとした言語を答えるのは一連の会話の中で２回だけである．Cの質問の答えである項目（ここでは［ママ］）とDの質問の答えである基準（ここでは［おとな］）がそれに当たる．しかし，アニマルはこの２回の質問で提示された答えから知識構造を造り上げて行くので，これはとても重要な部分になる．従来のプログラムではBでコンピュータが回答した項目と，Cで利用者が提供した項目が同じだったとしてもそれに気づかずにプログラムは進んでいたので，(1)の部分を付け加えてBの項目ととCの提供された項目とを識別できるようにした．例をあげて説明すると，上の会話のBの質問からCの質問を答える過程において，

B^{0}『ひとみちゃんでしょ？』－－－－－－－－－－－－－－－n

C^{0}『まちがえちゃった．じゃあだあれ？』－－－－－－－ひとみちゃん

のような答え方をしたとする．従来のプログラムでは次のコンピュータからの質問が，

D^{0}『ひとみちゃんが y で，ひとみちゃんが n になる質問なあに？』

となってしまい，どう答えれば良いのか訳の分からない質問になる．そこで，プログラム内に(1)を加えることにより，D^{0}の部分が

『あれ？あってるじゃん．じゃあいまね，パパが誰のこと考えているかあててあげようか？』

となり，B^{0}でｙと答えたときと同じ動作を行うようにしたので矛盾がなくなり，会話も奇麗に繋がるようになった．この(1)を加えたことにより，画面上での矛盾のある質問を無くしただけでなく，アニマルが作り上げていく知識構造での矛盾も防ぐことができるのである．

2.2.2　(2), (3) について

従来のプログラムでは利用者がyやnを答えるとき，英字入力モードの場合のyとn，すなわち，半角のyとnしか使えなかった．そこで，日本語入力モードのｙとｎ，つまり，全角のｙとｎでも同じように使えるように(2)と(3)を付け加えた．アニマルは元々日本人が造り上げたプログラムではないので，日本語入力モードは必要なかったと思うが，私たちがアニマルを扱っていく上で，yやnで答える時は半角入力モードで，日本語を答えるときは全角入力モードとその都度入力モードを変えるのでは手間もかかり，プログラムの質も落ちてしまう．アニマルを使いやすくするために付け加えたものである．

2.3　ゲームの概略

まずはじめに、次のことが言える．

命題１

利用者が正直に答えなければゲームは成り立たない．

アニマルは，テキスト書いてあるようにコンピュータが利用者との会話によって，知識を増やして行くゲームである．その得た知識から２分木と同型の知識構造を形成していくが，コンピュータは，利用者の答えたことをそのまま取り入れてしまうので，でたらめに答えるとそのまま間違った知識がコンピュータに蓄積される．そうなると事実と違った知識構造が形成され，アニマルのゲームは成り立たないし，ゲームの面白さも無くなってしまう．
コンピュータははじめ［ひとみちゃん］しか知らない知識構造，つまり枝の無い［ひとみちゃん］のみの２分木から始まる．そして利用者との会話から新しい項目（例えば［まま］や［じじ］）を知識構造に加えて行き，２分木を形成して行く．その時，ただ項目を教えてもらうだけでは２分木のどこの枝に繋げて行けばいいのか分からないので，整頓された２分木を造るため，その項目に対する基準（既に存在するある項目との違い）も教えてもらう．そしてその基準に対してYesであるかNoであるかで項目を整理し，２分木を形成して行く．また，その２分木を使って，次に利用者が頭に思い描いている項目を当てようとする．コンピュータの知識構造にすでに含まれている項目を利用者が思い描いていれば当てることができるが，まだコンピュータの知らない項目を利用者が答えようとしているならば，当てることはできず，また新しい項目を増やして２分木を広げる．すなわち，

命題２

知識構造の変化はコンピュータが項目を当て間違えたときにのみに起こる．

と言える．逆に言えば，利用者が同じ項目ばかり思い描いていては，知識構造は広がらないと言うことである．第２節のプログラムからわかるように，知識構造を変化させるGetSmarterが起動

されるのは「あれ，まちがえちゃった」という時のみである．
まとめると，利用者はコンピュータの中に既に形成されている知識構造にはない項目を頭の中に思い描いて一連の質問に答え，最後にコンピュータから回答されるある項目との差異を弁別する基準を答えることによって新しい知識を提示することができると言えよう．

2.4　知識構造のタイプ

ここではコンピュータが利用者との会話から形成していく知識構造に注目しよう．以下にアニマルによるコンピュータと利用者との会話の例を２つ上げる．それぞれをパターン A とパターン B とする．

会話例：パターン A

『パパが誰のこと考えてるかあててあげようか？』－－－－－－－－－－－－－y
『ひとみちゃんでしょ？』－－－－－－－－n
『まちがえちゃった．じゃあだあれ？』－－－－－－－－－－－－－－－－－－－らいおん
『らいおんが y で，ひとみちゃんが n になる質問なあに？』－－－－－－－－－鬣がある

『じゃあいまね，　パパが誰のこと考えてるかあててあげようか？』－－－y
『鬣がある？』－－－－－－－－－－－－－－－n
『ひとみちゃんでしょ？』－－－－－－－－n
『まちがえちゃった．じゃあだあれ？』－－－－－－－－－－－－－－－－－－－あすか
『あすかが y で，らいおんが n になる質問なあに？』－－－－－－－－－－－－隣りの家に住む

『じゃあいまね，　パパが誰のこと考えてるかあててあげようか？』－－－y
『鬣がある？』－－－－－－－－－－－－－－－n
『隣りの家に住む？』－－－－－－－－－－－y
『あすかでしょ？』－－－－－－－－－－－n
『まちがえちゃった．じゃあだあれ？』－－－－－－－－－－－－－－－－－－－しんいち
『しんいちが y で，あすかが n になる質問なあに？』－－－－－－－－－－－男

『じゃあいまね，　パパが誰のこと考えてるかあててあげようか？』－－－y
『鬣がある？』－－－－－－－－－－－－－－－n
『隣りの家に住む？』－－－－－－－－－－－n
『ひとみちゃんでしょ？』－－－－－－－－n
『まちがえちゃった．じゃあだあれ？』－－－－－－－－－－－－－－－－－－－いるか
『いるかが y で，ひとみちゃんが n になる質問なあに？』－－－－－－－－－足がない

『じゃあいまね，　パパが誰のこと考えてるかあててあげようか？』－－－y
『鬣がある？』－－－－－－－－－－－－－－－n
『隣りの家に住む？』－－－－－－－－－－－n
『足がない？』－－－－－－－－－－－－－－－n
『ひとみちゃんでしょ？』－－－－－－－－n
『まちがえちゃった．じゃあだあれ？』－－－－－－－－－－－－－－－－－－－クワガタ
『クワガタが y で，ひとみちゃんが n になる質問なあに？』－－－－－－－－鉄を持っている

『じゃあいまね，　パパが誰のこと考えてるかあててあげようか？』－－－y
『鬣がある？』－－－－－－－－－－－－－－－n
『隣りの家に住む？』－－－－－－－－－－－n

『足がない？』－－－－－－－－－－－－－－ｎ
『鋏を持っている？』－－－－－－－－－－－ｎ
『ひとみちゃんでしょ？』－－－－－－－－ｎ
『まちがえちゃった．じゃあだあれ？』－－－－－－－－－－－－－－－－－－－－まーくん
『まーくんが ｙ で，ひとみちゃんが ｎ になる質問なあに？』－－－－－－－－弟

『じゃあいまね，　パパが誰のこと考えてるかあててあげようか？』－－－ｙ
『鬣がある？』－－－－－－－－－－－－－－ｎ
『隣りの家に住む？』－－－－－－－－－－－ｎ
『足がない？』－－－－－－－－－－－－－－ｎ
『鋏を持っている？』－－－－－－－－－－－ｎ
『弟？』－－－－－－－－－－－－－－－－－ｎ
『ひとみちゃんでしょ？』－－－－－－－－ｎ
『まちがえちゃった．じゃあだあれ？』－－－－－－－－－－－－－－－－－－－－じじ
『じじが ｙ で，ひとみちゃんが ｎ になる質問なあに？』－－－－－－－－－－髭がある

『じゃあいまね，　パパが誰のこと考えてるかあててあげようか？』－－－ｙ
『鬣がある？』－－－－－－－－－－－－－－ｎ
『隣りの家に住む？』－－－－－－－－－－－ｎ
『足がない？』－－－－－－－－－－－－－－ｎ
『鋏を持っている？』－－－－－－－－－－－ｎ
『弟？』－－－－－－－－－－－－－－－－－ｎ
『髭がある？』－－－－－－－－－－－－－－ｎ
『ひとみちゃんでしょ？』－－－－－－－－ｎ
『まちがえちゃった．じゃあだあれ？』－－－－－－－－－－－－－－－－－－－－ま
『ままが ｙ で，ひとみちゃんが ｎ になる質問なあに？』－－－－－－－－－－化粧をする

『じゃあいまね，　パパが誰のこと考えてるかあててあげようか？』－－－ｙ
『鬣がある？』－－－－－－－－－－－－－－ｎ
『隣りの家に住む？』－－－－－－－－－－－ｎ
『足がない？』－－－－－－－－－－－－－－ｙ
『いるかでしょ？』－－－－－－－－－－－ｎ
『まちがえちゃった．じゃあだあれ？』－－－－－－－－－－－－－－－－－－－－さんま
『さんまが ｙ で，いるかが ｎ になる質問なあに？』－－－－－－－－－－－魚類である

『じゃあいまね，　パパが誰のこと考えてるかあててあげようか？』－－－ｙ
『鬣がある？』－－－－－－－－－－－－－－ｎ
『隣りの家に住む？』－－－－－－－－－－－ｙ
『男？』－－－－－－－－－－－－－－－－－ｙ
『しんいちでしょ？』－－－－－－－－－－ｎ
『まちがえちゃった．じゃあだあれ？』－－－－－－－－－－－－－－－－－－－－おじさん
『おじさんが ｙ で，しんいちが ｎ になる質問なあに？』－－－－－－－－－－大人

『じゃあいまね，　パパが誰のこと考えてるかあててあげようか？』－－－ｙ
『鬣がある？』－－－－－－－－－－－－－－ｎ
『隣りの家に住む？』－－－－－－－－－－－ｙ
『男？』－－－－－－－－－－－－－－－－－ｎ
『あすかでしょ？』－－－－－－－－－－ｎ
『まちがえちゃった．じゃあだあれ？』－－－－－－－－－－－－－－－－－－－－おばさん
『おばさんが ｙ で，あすかが ｎ になる質問なあに？』－－－－－－－－－－大人

『じゃあいまね，　パパが誰のこと考えてるかあててあげようか？』－－－ｙ
『鬣がある？』－－－－－－－－－－－－－－ｎ
『隣りの家に住む？』－－－－－－－－－－－ｎ
『足がない？』－－－－－－－－－－－－－－ｎ

『鋏を持っている？』－－－－－－－－－－ｎ
『弟？』－－－－－－－－－－－－－－－－ｎ
『髭がある？』－－－－－－－－－－－－－ｎ
『化粧をする？』－－－－－－－－－－－－ｎ
『ひとみちゃんでしょ？』－－－－－－－ｎ
『まちがえちゃった．じゃあだあれ？』－－－－－－－－－－－－－－－－－きりん
『きりんが ｙ で，ひとみちゃんが ｎ になる質問なあに？』－－－－－－－－人間ではない

『じゃあいまね，　パパが誰のこと考えてるかあててあげようか？』－－－ｙ
『鬣がある？』－－－－－－－－－－－－ｎ
『隣りの家に住む？』－－－－－－－－－－ｎ
『足がない？』－－－－－－－－－－－－－ｎ
『鋏を持っている？』－－－－－－－－－－ｎ
『弟？』－－－－－－－－－－－－－－－－ｎ
『髭がある？』－－－－－－－－－－－－－ｎ
『化粧をする？』－－－－－－－－－－－－ｎ
『人間ではない？』－－－－－－－－－－－ｙ
『きりんでしょ？』－－－－－－－－－－ｎ
『まちがえちゃった．じゃあだあれ？』－－－－－－－－－－－－－－－－－かめ
『かめが ｙ で，きりんが ｎ になる質問なあに？』－－－－－－－－－－－－－水の中に住む

『じゃあいまね，　パパが誰のこと考えてるかあててあげようか？』－－－ｙ
『鬣がある？』－－－－－－－－－－－－－ｎ
『隣りの家に住む？』－－－－－－－－－－ｎ
『足がない？』－－－－－－－－－－－－－ｙ
『魚類である？』－－－－－－－－－－－－ｙ
『さんまでしょ？』－－－－－－－－－－ｎ
『まちがえちゃった．じゃあだあれ？』－－－－－－－－－－－－－－－－－あゆ
『あゆが ｙ で，さんまが ｎ になる質問なあに？』－－－－－－－－－－－－川に住む

『じゃあいまね，　パパが誰のこと考えてるかあててあげようか？』－－－ｙ
『鬣がある？』－－－－－－－－－－－－－ｎ
『隣りの家に住む？』－－－－－－－－－－ｎ
『足がない？』－－－－－－－－－－－－－ｎ
『鋏を持っている？』－－－－－－－－－－ｎ
『弟？』－－－－－－－－－－－－－－－－ｎ
『髭がある？』－－－－－－－－－－－－－ｎ
『化粧をする？』－－－－－－－－－－－－ｎ
『人間ではない？』－－－－－－－－－－－ｙ
『水の中に住む？』－－－－－－－－－－－ｎ
『きりんでしょ？』－－－－－－－－－－ｎ
『まちがえちゃった．じゃあだあれ？』－－－－－－－－－－－－－－－－－とんぼ
『とんぼが ｙ で，きりんが ｎ になる質問なあに？』－－－－－－－－－－－－虫

『じゃあいまね，　パパが誰のこと考えてるかあててあげようか？』－－－ｎ
『じゃあ，あとでまた遊ぼうね．これ覚えとく？』－－ｙ
（終了）

会話例：パターンB

『パパが誰のこと考えてるかあててあげようか？』－－－－－－－－－－－－y
『ひとみちゃんでしょ？』－－－－－－－－n
『まちがえちゃった．じゃあだあれ？』－－－－－－－－－－－－－－－－－－－らいおん
『らいおんが y で，ひとみちゃんが n になる質問なあに？』－－－－－－－－人間ではない

『じゃあいまね，　パパが誰のこと考えてるかあててあげようか？』－－－y
『人間ではない？』－－－－－－－－－－－－n
『ひとみちゃんでしょ？』－－－－－－－－n
『まちがえちゃった．じゃあだあれ？』－－－－－－－－－－－－－－－－－－－あすか
『あすかが y で，らいおんが n になる質問なあに？』－－－－－－－－－－－家族でない

『じゃあいまね，　パパが誰のこと考えてるかあててあげようか？』－－－y
『人間ではない？』－－－－－－－－－－－－n
『家族ではない？』－－－－－－－－－－－－y
『あすかでしょ？』－－－－－－－－－－－－n
『まちがえちゃった．じゃあだあれ？』－－－－－－－－－－－－－－－－－－－しんいち
『しんいちが y で，あすかが n になる質問なあに？』－－－－－－－－－－－男

『じゃあいまね，　パパが誰のこと考えてるかあててあげようか？』－－－y
『人間ではない？』－－－－－－－－－－－－y
『らいおんでしょ？』－－－－－－－－－－－n
『まちがえちゃった．じゃあだあれ？』－－－－－－－－－－－－－－－－－－－いるか
『いるかが y で，らいおんが n になる質問なあに？』－－－－－－－－－－－水の中に住む

『じゃあいまね，　パパが誰のこと考えてるかあててあげようか？』－－－y
『人間ではない？』－－－－－－－－－－－－y
『水の中に住む？』－－－－－－－－－－－－n
『らいおんでしょ？』－－－－－－－－－－－n
『まちがえちゃった．じゃあだあれ？』－－－－－－－－－－－－－－－－－－－クワガタ
『クワガタが y で，らいおんが n になる質問なあに？』－－－－－－－－－－虫

『じゃあいまね，　パパが誰のこと考えてるかあててあげようか？』－－－y
『人間ではない？』－－－－－－－－－－－－n
『家族ではない？』－－－－－－－－－－－－n
『ひとみちゃんでしょ？』－－－－－－－－n
『まちがえちゃった．じゃあだあれ？』－－－－－－－－－－－－－－－－－－－まーくん
『まーくんが y で，ひとみちゃんが n になる質問なあに？』－－－－－－－－男

『じゃあいまね，　パパが誰のこと考えてるかあててあげようか？』－－－y
『人間ではない？』－－－－－－－－－－－－n
『家族ではない？』－－－－－－－－－－－－n
『男？』－－－－－－－－－－－－－－－－－－y
『まーくんでしょ？』－－－－－－－－n
『まちがえちゃった．じゃあだあれ？』－－－－－－－－－－－－－－－－－－－じじ
『じじが y で，まーくんが n になる質問なあに？』－－－－－－－－－－－－大人

『じゃあいまね，　パパが誰のこと考えてるかあててあげようか？』－－－y
『人間ではない？』－－－－－－－－－－－－n
『家族ではない？』－－－－－－－－－－－－n
『男？』－－－－－－－－－－－－－－－－－－n
『ひとみちゃんでしょ？』－－－－－－－－n
『まちがえちゃった．じゃあだあれ？』－－－－－－－－－－－－－－－－－－－まま

『ままが y で，ひとみちゃんが n になる質問なあに？』－－－－－－－－－－大人

『じゃあいまね，　パパが誰のこと考えてるかあててあげようか？』－－－y
『人間ではない？』－－－－－－－－－－－－y
『水の中に住む？』－－－－－－－－－－－－y
『いるかでしょ？』－－－－－－－－－－n
『まちがえちゃった．じゃあだあれ？』－－－－－－－－－－－－－－－－－－さんま
『さんまが y で，いるかが n になる質問なあに？』－－－－－－－－－－－－魚類である

『じゃあいまね，　パパが誰のこと考えてるかあててあげようか？』－－－y
『人間ではない？』－－－－－－－－－－－－n
『家族ではない？』－－－－－－－－－－－－y
『男？』－－－－－－－－－－－－－－－－－－y
『しんいちでしょ？』－－－－－－－－－－n
『まちがえちゃった．じゃあだあれ？』－－－－－－－－－－－－－－－－－－おじさん
『おじさんが y で，しんいちが n になる質問なあに？』－－－－－－－－－－大人

『じゃあいまね，　パパが誰のこと考えてるかあててあげようか？』－－－y
『人間ではない？』－－－－－－－－－－－－n
『家族ではない？』－－－－－－－－－－－－y
『男？』－－－－－－－－－－－－－－－－－－n
『あすかでしょ？』－－－－－－－－－－n
『まちがえちゃった．じゃあだあれ？』－－－－－－－－－－－－－－－－－－おばさん
『おばさんが y で，あすかが n になる質問なあに？』－－－－－－－－－－－大人

『じゃあいまね，　パパが誰のこと考えてるかあててあげようか？』－－－y
『人間ではない？』－－－－－－－－－－－－y
『水の中に住む？』－－－－－－－－－－－－n
『虫？』－－－－－－－－－－－－－－－－－－n
『らいおんでしょ？』－－－－－－－－n
『まちがえちゃった．じゃあだあれ？』－－－－－－－－－－－－－－－－－－きりん
『きりんが y で，らいおんが n になる質問なあに？』－－－－－－－－－－－草食である

『じゃあいまね，　パパが誰のこと考えてるかあててあげようか？』－－－y
『人間ではない？』－－－－－－－－－－－－y
『水の中に住む？』－－－－－－－－－－－－y
『魚類である？』－－－－－－－－－－－－－n
『いるかでしょ？』－－－－－－－－－－－n
『まちがえちゃった．じゃあだあれ？』－－－－－－－－－－－－－－－－－－かめ
『かめが y で，いるかが n になる質問なあに？』－－－－－－－－－－－－－歩く

『じゃあいまね，　パパが誰のこと考えてるかあててあげようか？』－－－y
『人間ではない？』－－－－－－－－－－－－y
『水の中に住む？』－－－－－－－－－－－－y
『魚類である？』－－－－－－－－－－－－－y
『さんまでしょ？』－－－－－－－－－－－n
『まちがえちゃった．じゃあだあれ？』－－－－－－－－－－－－－－－－－－あゆ
『あゆが y で，さんまが n になる質問なあに？』－－－－－－－－－－－－－川に住む

『じゃあいまね，　パパが誰のこと考えてるかあててあげようか？』－－－y
『人間ではない？』－－－－－－－－－－－－y
『水の中に住む？』－－－－－－－－－－－－n
『虫？』－－－－－－－－－－－－－－－－－－y
『クワガタでしょ？』－－－－－－－－－－n
『まちがえちゃった．じゃあだあれ？』－－－－－－－－－－－－－－－－－－とんぼ

『とんぼが y で，クワガタが n になる質問なあに？』－－－－－－－－－－羽がある

『じゃあいまね，　パパが誰のこと考えてるかあててあげようか？』－－－n
『じゃあ，あとでまた遊ぼうね．これ覚えとく？』－－y
（終了）

2.4.1　２つの会話の相違点

この２つの会話例を比べてみる．相互の共通点は，

利用者が提示する項目が等しく，提示された項目の順番は

＊{　［らいおん］［あすか (友達)］［しんいち (友達の兄)］［いるか］［クワガタ］［まーくん (弟)］［じじ］［まま］［さんま］［おじさん］［おばさん］［きりん］［かめ］［あゆ］［とんぼ］　}

である．しかし，形成される知識構造は同じにはならない．会話全体を比べても全く同じではない．相互の会話の違いを明らかにするため，それぞれの会話で形成された知識構造を２分木で表す．２分木は基準を節とし，節に対して項目がNoであれば左の枝，Yesであれば右の枝へと繋がってできている．つまり，１つの節に対して左部分木がNoのもの，右部分木がYesのものに当たる．会話例：パターンAの知識構造を図A，会話例：パターンBの知識構造を図Bとする．２つの２分木の葉の部分（項目）は同じであるが，枝の広がり方が異なっている．図Aは左部分木を中心に枝を広げているタイプで，図Bは左右にバランス良く枝を伸ばしているタイプになっている．ここで，次のことが言える．

命題３

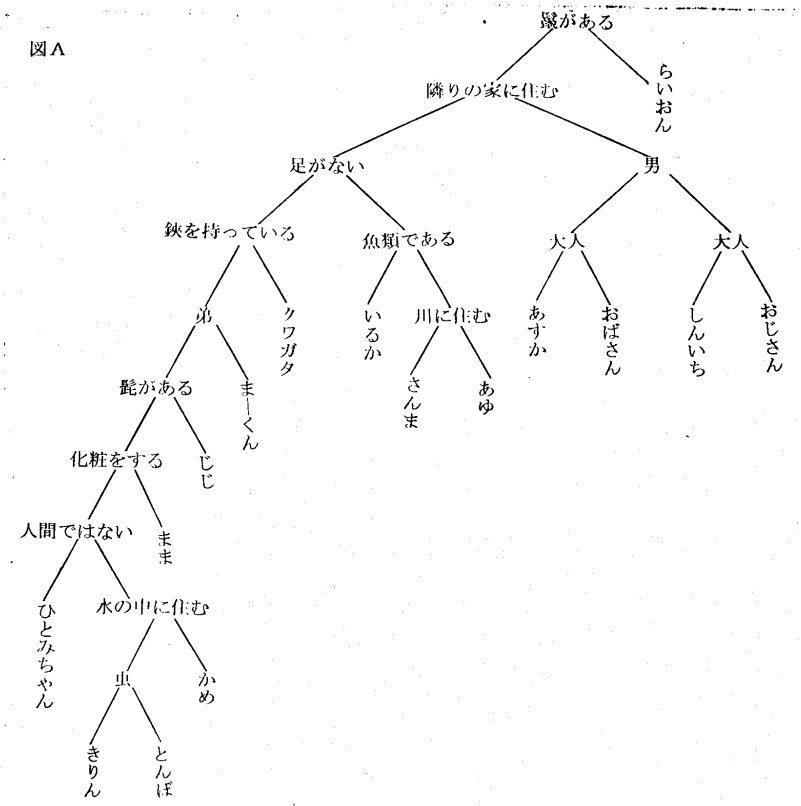
利用者が提示する項目は，形成される知識構造のタイプに影響しない．

なぜ，項目は全く同じなのに，知識構造はここまで違うのだろうか．すなわち，何が知識構造のタイプに影響するのか．利用者がコンピュータに対して制御可能な要素は提示する項目とそれに伴う基準だけである．命題３のように項目は影響しないとすれば，基準が知識構造のタイプの鍵を握っていると思える．そこで，基準に注目してみよう．２つの会話の提示する基準を比べてみると，例えば，パターンAでは

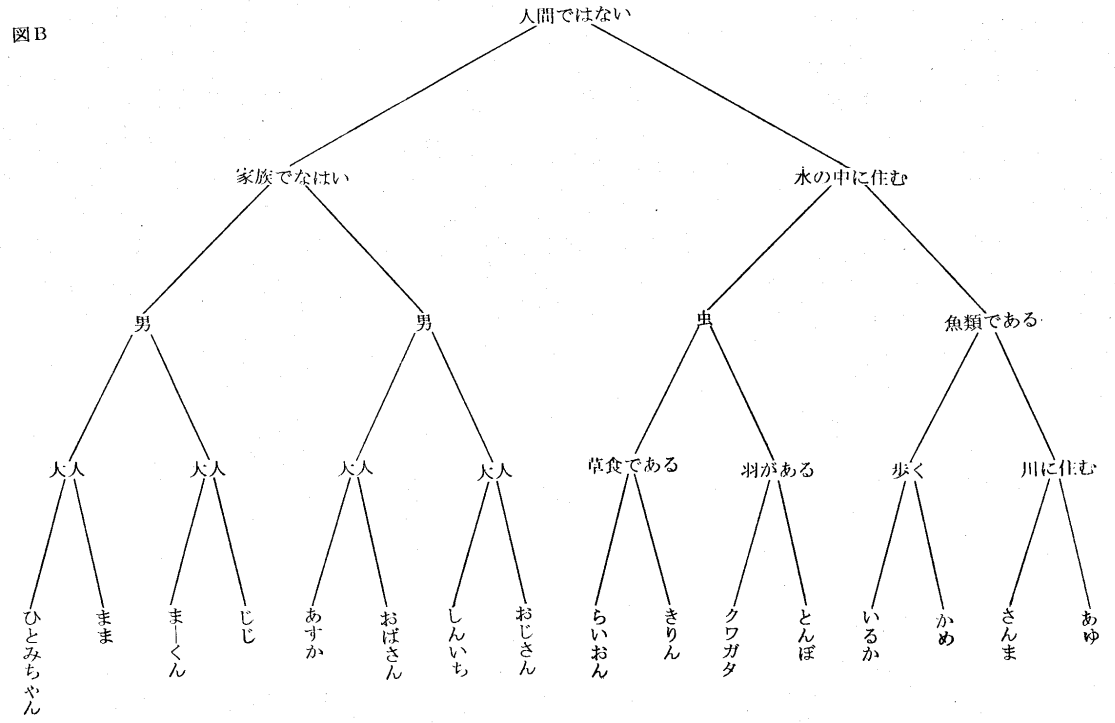
『らいおんがyでひとみちゃんがnになる質問なあに？』＿＿＿＿鬣があると，項目［らいおん］に対する基準を［鬣がある］と答えているのに対し，パターンBでは

『らいおんがyでひとみちゃんがnになる質問なあに？』＿＿＿＿人間であると，項目［らいおん］に対する基準を［人間である］と答えている．これ以外の質問でも，２つの会話は項目は同じものを提示しているが，それに対する基準が異なっている．よって，次のことが言える．

図A


鼠がある
隣りの家に住む
らいおん
足がない
男
鋏を持っている
魚類である
大人
大人
弟
クワガタ
いるか
川に住む
あずか
おばさん
しんいち
おじさん
髭がある
ま―くん
さんま
あゆ
化粧をする
じじ
人間ではない
まま
ひとみちゃん
水の中に住む
虫
かめ
きりん
とんぼ

図B


人間ではない
家族でなはい
水の中に住む
男
男
虫
魚類である
大人
大人
大人
大人
草食である
羽がある
歩く
川に住む
ひとみちゃん
まま
ま―くん
じじ
あすか
おばさん
しんいち
おじさん
らいおん
きりん
クワガタ
とんぼ
いるか
かめ
さんま
あゆ

命題４

一連の会話の中で提示された項目が全く同じでも，基準が異なれば，知識構造は異なる．

また，パターン**A**で項目［おじさん］に対する基準と［おばさん］に対する基準がどちらも［大人］を提示している．パターン**B**でも基準を［大人］とする項目が４つ（［じじ］［まま］［おじさん］［おばさん］）もある．しかし，会話にも知識構造にも問題はない．よって，

命題５

基準の内容が重複しても，知識構造に影響しない．

と言える．第２節で述べたプログラム改良の考察中，はじめ基準の重複は知識構造で矛盾が生じてしまうと思っていたので，重複した場合，『一度使った基準はだめだよ．』などと出力させようと考えていたが必要ないことがわかった．

2.4.2　会話から分かる２分木の偏り

会話例の項目［いるか］に対するコンピュータからの基準の問い方に注目する．

Aでは『いるかが **y** で，ひとみちゃんが **n** になる質問なあに？』－－－－－－－－－足がない

Bでは『いるかが **y** で，らいおんが **n** になる質問なあに？』－－－－－－－－－－－水の中に住む

となっていて，確かに利用者が提示する基準も異なっているが，コンピュータからの問い方も異なっている．パターン**A**が［いるか］と［ひとみちゃん］との基準を聞いているのに対し，パターン**B**では［いるか］と［らいおん］の基準を聞いている．ここで，会話全体の様子を比べてみても，確かに提示する基準は違っているが，同時にその基準に対する質問の仕方も異なっている．さらに詳しく見てみると，パターン**A**はパターン**B**に比べて，［ひとみちゃん］との基準を問う場合が多い．つまり，

（＊１）『［項目**1**］が **y** で，［項目２］が **n** になる質問なあに？』

の質問の中で，［項目**2**］の位置が［ひとみちゃん］である場合がパターン**B**では４回しかないのに対し，パターン**A**では８回もある．ここで２つの**2**分木を比べてみると，図**A**の２分木は，図**B**の２分木に比べてバランスが悪く，左に偏っている．それに比べて図**B**の２分木は左右対称でバランスが良い．この２つの２分木は１つの基準に対して**Yes**の項目は右側に，**No**のものは左側にくる．（＊１）の質問文でいう［項目１］は右側に，［項目**2**］は左側の枝に繋がることになる．［項目１］は利用者が提示した項目で［ひとみちゃん］が［項目１］の部分にくることはないので，

命題６

［ひとみちゃん］は常に左部分木に位置し，全体の中でも，一番左に位置する．

と言える．つまり，会話例パターン**A**で，提示した項目と［ひとみちゃん］との基準を聞く場合

が多いのは，同時に，知識構造の中で左の方ばかりに枝が広がっているということである．実際，パターン A の２分木は左の方に偏っている．従って，

命題７

会話の中で［提示した項目］と［ある項目］の基準を聞かれる場合に，その［ bf ある項目］が何度も同じ項目で重なる時，知識構造は偏る．

と言える．では，なぜパターン A は（＊１）の［項目2］の位置が［ひとみちゃん］ばかりなのだろうか．［項目2］の位置が［ひとみちゃん］であるなら，同時にその前のコンピュータの回答は

『ひとみちゃんでしょ？』

となっている．この『ひとみちゃんでしょ？』と回答する場合の例を上げると，パターン A の会話の途中に，次に利用者が項目［じじ］を思い浮かべている場面を上げる．会話例を見るとそれを当てる為にコンピュータが質問した基準は

『鬣がある？』－－－－－－－－－－－－－－－n
『隣りの家に住む？』－－－－－－－－－－－－n
『足がない？』－－－－－－－－－－－－－－－n
『鋏を持っている？』－－－－－－－－－－－－n
『弟？』－－－－－－－－－－－－－－－－－－n

と，［じじ］に対して全て No になるものばかりである．コンピュータが基準に対して全て No となる項目を２分木で辿っていくと，［ひとみちゃん］に辿り着くので，当然コンピュータは

『ひとみちゃんでしょ？』

と回答するのである．他の項目の場合でも全て No で答えた場合は最後に『ひとみちゃんでしょ？』と回答している．つまり，コンピュータが最後に『ひとみちゃんでしょ？』と聞くのは，それまでの基準に対する質問に利用者が全て No と答えたからである．

会話全体の様子を見ても，パターン A は基準に Yes，No で答える質問に，No と答える場合が全部で 51 回，Yes と答える場合が 9 回と Yes に比べて No は遥かに多い．その分知識構造が No にあたる左の方に偏って広がっているということになる．では，Yes を多く答えればいいのかというと，そうではない．No を多く答えた場合と同様で，今度は知識構造が Yes にあたる右に偏るだけである．

命題８

会話の中の基準に対する質問で，Yes ばかりで答えれば右に，No ばかりで答えれば左に知識構造は偏る．

Yes ばかりで答えた場合と No ばかりで答えた場合の違う点は命題７が成り立たないことである．利用者が新たに項目を提示した時，その項目は節の右側の枝につく．利用者が基準に対して Yes

ばかりで答えて行くと，２分木は節の右側へ右側へと進んで行くので，最後に辿り着く項目は常に前回提示した項目になる．よって，命題７は成り立たないのである．

Yes，Noをどちらかに偏って答えているからは２分木は偏るので，Yes，Noを均等に答えれば２分木は偏らないのか考えてみる．知識構造が偏っていないパターンBの会話に注目する．会話全体の中で，基準に対するYes，Noの応答がどのような割合で行われているか明らかにするため，例のように必要なｙとｎだけ取り出してみる．

(例)

『じゃあいまね，パパが誰のこと考えているかあててあげようか？』

『人間ではない？』－－－－－－－－－－－ｎ

『家族ではない？』－－－－－－－－－－－ｙ

『あすかでしょ？』

『まちがえちゃった．じゃあだあれ？』

の場合，（ｎｙ）とするように，会話の区切りごとに取り出す．

したがって，

（ｎ）（ｎｙ）（ｙ）（ｙｎ）（ｎｎ）（ｎｎｙ）（ｎｎｎ）（ｎｙｙ）（ｎｙｎ）（ｙｙ）

（ｙｎｎ）（ｙｙｎ）（ｙｙｙ）（ｙｎｙ）

となる．それぞれの数で並べ替えると，

（ｙ）（ｎ）

（ｙｙ）（ｙｎ）（ｎｙ）（ｎｎ）

（ｙｙｙ）（ｙｙｎ）（ｙｎｙ）（ｎｙｙ）（ｙｎｎ）（ｎｙｎ）（ｎｎｙ）（ｎｎｎ）

となる．結果，基準に対する質問が34回ある中で，ｙとｎを答えている回数はそれぞれ17回ずつで同じで，奇麗に並んでいる．さらに，パターンBの会話全体を見てみると，例えば『人間ではない？』の質問は14回出てくる．それに対するｙとｎの応答はそれぞれ７回ずつである．他の例で『家族ではない？』の質問は全部で６回出てくるが，ｙとｎはそれぞれ３回ずつ応答している．その他の質問に対してもパターンBの会話例を見てわかるように一つの質問に対してのｙとｎの応答の割合は同じになっている．すなわち，

命題９

2分木を左右バランス良く形成させるには，会話中にある基準に対する質問がk回あったとすれば，利用者のｙとｎの応答をk/2で応答する．

と言える．ここで，上と同様にパターンAの基準に対するｙ，ｎの応答も取り出すと，

（n）（ny）（nn）（nnnn）（nnnnn）（nnnnnn）（nny）（nyn）
（nyn）（nnnnnnn）（nnnnnnny）（nnyy）（nnnnnnnyn）

となる．上のパターンBとくらべてnが多いのは明らかであるが，もう一つ大きな違いは，（）の中のy，nの数である．上のパターンBの（）の中のy，nは一番多くて3つであるのに対し，パターンAでは最高9つもある．y，nの数が多いのは会話中のコンピュータからの質問の数が多いということになる．

例えば，項目［きりん］を利用者が思い浮かべている場合を上げる．パターンAでは7回基準を問い掛けてから『＊＊＊でしょ？』と回答しているのに対し，パターンBでは3回基準を問うだけで，『＊＊＊でしょ？』と回答している．別の例の項目［とんぼ］を思い浮かべている場合でも，パターンAで基準が9回質問されているのに対し，パターンBでは4回しか質問していない．コンピュータは形成された2分木を辿りながら出会った基準を質問して行き，最後に葉に辿り着いた時，『＊＊＊でしょ？』と項目を回答する．パターンAの2分木は一方に偏っているため，一つの項目に到達するまでが長くなるので質問の回数が多いのである．知識構造のタイプと質問回数の関係を明らかにするため，図Aよりも極端に偏った知識構造の例を上げて以下に証明する．

証明　8つの項目$i_1, i_2, \cdots, i_8$　を含む2つの知識構造SとTがあるとする．Sは2分木で表すと図Bのように左右対称でバランスが良い知識構造で，Tは左部分木に一直線に延びた2分木で，各節の右側の枝の葉が全て項目になっていて，図Aよりもさらに左に偏っている知識構造である．qを基準としてそれぞれリストで表すと，

$$S = [[[\ i_1\ \ q_{000}\ \ i_2\ ]\,q_{00}\ \ [\ i_3\ \ q_{001}\ \ i_4\ ]]\,q_0$$
$$[[\ i_5\ \ q_{010}\ \ i_6\ ]\ \ q_{01}\ \ [\ i_7\ \ q_{011}\ \ i_8\ ]]]$$
$$T = [[[[[[\ i_1 q_1 i_2\ ]\,q_2\ \ i_3\ ]\,q_3\ \ i_4\ ]\,q_4\ \ i_5\ ]\,q_5\ \ i_6\ ]\,q_6\ \ i_7\ ]\,q_7\ \ i_8\ ]$$

となる．この2つの知識構造SとTにおいて，8つのうち1つの項目i_jを正解として，それに辿り着くまでに要する質問の回数kの期待値Eを考える．

Sの場合:

正解がi_jとして，そこに辿り着くのに要する質問の回数がk_jとすると，jが1，2，3，…，8のどの値であっても，

$$k_j = 3$$

i_jが正解である確率P_jは，jが1，2，3，…，8どの値であっても，

$$P_j = \frac{1}{8}$$

従って，求める質問回数の期待値E_Sは

$$E_S = (\frac{1}{8} \times 3) \times 8$$

$$= 3$$

よって，どの項目であっても３回で正答に到達する．T の場合:
正解i_jまでに要する質問の回数k_jは

$$k_j = j (j = 1, 2, 3, \cdots 7)$$

$$k_8 = 7$$

i_jが正解である確率P_jはSの場合と同様に

$$P_j = \frac{1}{8}$$

従って，求める質問回数の期待値E_tは

$$E_t = (\frac{1}{8} \times 1) + (\frac{1}{8} \times 2) + (\frac{1}{8} \times 3) + (\frac{1}{8} \times 4) + (\frac{1}{8} \times 5) + (\frac{1}{8} \times 6) + (\frac{1}{8} \times 7) + (\frac{1}{8} \times 7)$$

$$= \frac{35}{8}$$

$$= 4.375$$

一般に，2^n個の項目で深さがnになる2分木の知識構造を作ったとき，質問回数の期待値は

$$E_S = n$$

よって，m個の項目でSの知識構造を形成した場合の質問回数の期待値は

$$E_S = \log_2 m$$

である．各基準に一つの項目が対応するような直線状の知識構造を作った場合は質問回数の期待値は,

$$E_t = \frac{2n + 1}{2} \div \frac{1}{2^n}$$

よって，m個の項目でTの知識構造を形成した場合の質問回数の期待値は

$$E_t = \frac{m + 1}{2} \div \frac{1}{m}$$

である．mが十分大きいとき，$\log_2 m < \frac{m + 1}{2} \div \frac{1}{m}$ となるので，知識構造Sの方が質問回数が少なくなると証明された.

従って,

命題１０

知識構造に含まれている項目を利用者が互いに同様な確率で頭に思い描くとき，正答に至るまでの質問回数の期待値は，図Ｂの方が図Ａよりも少なく，２分木が深くなるほど言える．

つまり，パターンＢは，パターンＡに比べて，少ない質問回数でコンピュータが項目を回答することができ，これは項目の数が増えるほど言えるのである．

アニマルと会話をする時，どちらの会話例の方が良いか悪いかは決め難いが，質問回数のみに注目すれば，パターンＢの方が経済的であることが分かった．パターンＢのようなバランスの取れた２分木を形成するには命題９で書いたように，基準に対する質問にｙとｎを均等に答えて行けば良いのだが，そのコンピュータが問いかける基準はもともとは利用者が提示した基準である．即ち，ｙやｎを均等に答えると言うよりも，その基準を提示する時点で，その基準に対して，Yesの項目とNoの項目が半数ずつになる基準を提示することがバランスの取れた２分木を形成する上で重要と言える．実際に会話例を見ても，パターンＡで最初に提示した基準の『鬣がある』に対してYesとなる項目は［らいおん］のみで，後は全てNoとなるのに対し，パターンＢの最初の基準『人間ではない』に対してYesとなる項目は［らいおん］［きりん］［クワガタ］［とんぼ］［いるか］［かめ］［さんま］［あゆ］の８つで残りの８つの項目はNoとなっている．しかし，アニマルはその瞬時に思いついた項目を答えていくゲームなので，どのような基準を答えればｙとｎが半数になるかは分からない．この１６つの項目では『人間ではない』に対してｙとｎが半数ずつになったが他の１６つの項目で半数ずつになるとは限らないのである．しかし，利用者が,『パパが誰のこと考えてるかあててあげようか？』と聞かれた時，思い浮かべる項目は利用者が既に知っているもののどれかである．今，自分が知っているもの中で,『人間ではない』に対してYesのものも，Noのものも数多く知っているが,『鬣がある』に対してNoのものは数多く知っているがYesのものはらいおん以外で，馬…? ぐらいしか思いつかない．つまり，奇麗に半数ずつに分かれる基準ではなく，ｙのものも，ｎのものも数多くある基準を提示することが重要であると言える．また,『人間ではない』という基準はパターンＡの会話でも提示しているが，会話の後の方で提示したため，２分木のタイプは変わらない．よって，次のことが言える．

命題１１

偏らない知識構造を形成するには，基準を提示する時，その基準に対してYesのものも，Noのものも数多くある基準を提示する．

もし，図Ｂのように左右対称な２分木を形成させたいならば，命題９のようにｙやｎを均等に答えて行き，それに合った項目を後から考えて提示するのも面白いと思う．

2.5　まとめ

会話の中で利用者がコンピュータの質問に答えて行くことは，同時に２分木内でコンピュータを誘導していることであると分かった．コンピュータははじめ，根にある基準を問いてｙかｎで，右か左の枝を進み，行った先の節に当たる基準を問い，また，ｙかｎで次の行き先を決め，最後に辿り着いた葉の項目を回答する．その項目が当っていれば根に戻るが，間違っていれば新しい項目と基準を提示してもらい，新たに枝を形成する．アニマルのゲームは根をStar ｔ地点とし，辿り着きたい項目をGool とする迷路とも考えられる．また，ｙやｎを調節して答えて，バランスの取れた２分木に限らず，わざと偏った２分木や，右部分木と左部分木が一本ずつしかない２分木等，形成したい知識構造のタイプを予め予想してからゲームを始めるのも楽しいと思う．ただ，質問に答えていくだけでなく，形成される知識構造も考えながら会話していくのがアニマルを楽しむ上で重要である．

Chapter3

「三角形と四角形」の授業展開

3.1　授業プランの作製

近年，教育現場では様々な変化が求められている．勿論，その中には授業の指導も含まれているわけであるが実際に変化を成し遂げている授業というのは少ないように思える．そこで，以下では教育実習を通して，実際の子どもの疑問から，私が問題として思ったことを考えていく．

3.1.1　教育実習を通して

大学３年生の４週間実習で小学校２年生を担当した．このことは，実際の教育現場の経験のない私にとってとても貴重な経験となった．そのときに，算数の「三角形と四角形」という単元を学習した．その単元の中で正方形を学習するところがあるのだが，その正方形の応用として図１のようないろいろ描かれている図形の中から正方形を選ぶ，という課題を与えた．

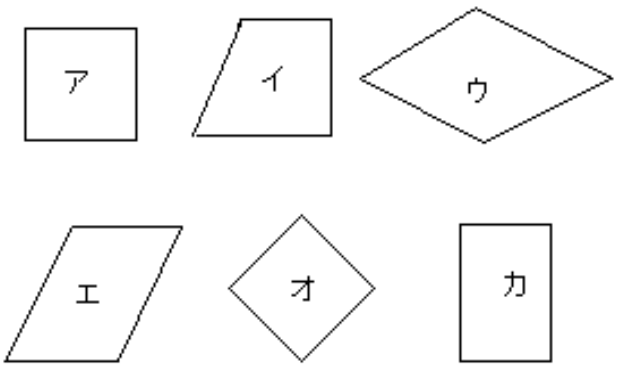


図１： どれが正方形かな？

この課題を与えたときに子ども達が考えた答えとしては大きく２つに分かれた．

1. （ア）だけ,
2. （ア）,（オ）

というものだった．この子ども達の答えをみる限り，（ア）が答えであることはすぐに見分けがつくが，（オ）に関しては何人かの子ども達が正方形であることに疑問を抱いた．この課題の答えというのは（ア），（オ）であるのだが，子どもにとってはそれを自分で導くのが難しい．

ここで（ア），（オ）の図形の違いについてふれておこう．これらの図形は２つとも全く同じ正方形であるのだが，子どもの目には違う図形のように映ってしまうようだ．

まず，この学習内容で扱うことになっている正方形という図形は「４辺の長さが同じで，４つの角が直角である図形」というように定義されている．このことから辺の長さを調べること，また角が直角であることを調べることによって，正方形であるかどうか分類することは容易である．

そこで，そのことを定規を使ったりして調べることにした．しかしそのことを実際にやってみた後でもまだ（オ）の図形が正方形ではない，という子どもがいた．

そこで次に，同じ大きさの図形（この図形は子どもの目に正方形だと映る．）を持ってきて（オ）のように傾いているのではなくてそれを図２のように回転させて　見せることにした．

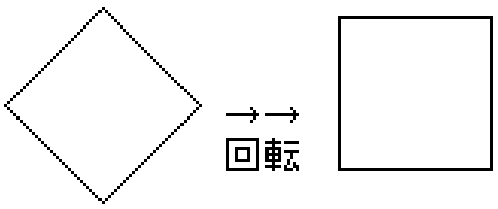


図2： 正方形を回転させる

すると初めて（オ）の図形が正方形ではないと思っていた子どもも（オ）の図形が正方形であることを納得した．

では，なぜそのように同じ図形であるのに少し傾いたりすると，子どもの目には違う図形のように映ってしまうのだろうか．それは図形そのものを１つの文字と同じ感覚でとらえているからではないかと考えられる．ここでの文字とは，例えばエレベーターのボタンで上に行くなら「△」というボタンを押し，下に行くなら「▽」というボタンを押すようにその２つが意味の違う別の記号として捉えられていることを言う．この記号の考え方が影響して，同じ三角形でも子どもにとっては「△」とその逆向きである「▽」が違うものであるように見えているのだと思われる．図形において，この２つを較べると三角形の定義が「３つの直線で囲まれている図形」とされているために「△」も「▽」も三角形であることに違いはなく同じ三角形として分類していくのであるが，それが子どもの目には違う形として映ってしまうのだ．そのため，子どもは記号である三角と図形である三角の区別がつかない．

まずその原因としてこれらの図形の導入の仕方に問題があるのだと思われる．なぜなら小学校で形を導入する際，三角形ならば「△」という図形をまず見せる．あまり最初から「△」でなく，

「▽」の逆三角形を見せることはしない．それによって，△を「三角」という言語として認識するが，回転した▽になりうる図形としては認識しない，すなわち未分化なままなのである．それは四角形など他の図形も共通に言えることであってそれは図形のいろいろな定義を知った後も影響していくと考えられる．定義を知った後でも定義に戻ることよりも自分の中で作られた図形に戻って考える傾向が出てきてしまうのだ．

3.1.2　授業の導入について

記号である形と図形である形を未分化にさせない例として，DIME Mathematicsの「FEELY BOX」[1]を使った活動が思いつく．これは，箱の中に入っている物体が何であるのか，を当てるゲームである．その物体は見ることが出来ないため，箱の中に手だけを入れて，物体の特徴をつかんで，当てていく．物体の特徴，すなわち角や辺をとらえるため，記号として形を捉えることがない．しかしながら，これは立体図形だからできるのであって，平面図形に生かそうとしても手で触って特徴をとらえるという長所は生かしきれないと思われる．

そこで，パソコンを使うことでそのハンディをなくそうと試みた．箱の中の見えない形がパソコンでは画面上に出てこない図形であって，自分の手がパソコンでは亀となって，やはり図形の特徴をとらえていくことで，隠れた図形が何であるのかを当てるというものである．

以上のことを踏まえて，パソコンによって三角形や四角形の特徴をつかんでいく，またそれぞれの図形の共通点を見つけていくような教材を作ることにした．

3.1.3　三角形・四角形のゲームの見解

このゲームは，最初にも述べたように，導入部分において役立てようと考えたものである．そのため，以下のようなことを念頭に置きながら，プログラムを考えた．

まず１つ目に，このゲームは授業の導入に使う，ということである．そのために，三角形でも「△」の形だけを持ってくるのではなく,「▽」という形も取り入れて，今までの導入時における先入観のようなものをなくそうと試みている．

そして２つ目に，遊び感覚を生かす，ということである．これは，子どもの興味を引くためにはもっとも大切なものであり，この先の子どもの授業の取り組みにも関わってくるものである．この「遊び」ということに関しては，ディーンズ[2]が「遊びがより高いレベルでの認知活動と密接に結びついているということが，もし事実であるとすれば，遊びを多様なものにすることにより，この事実を学習の促進のために役立てることができます.」と述べている．このゲームにおいての「認知活動」というのは，子ども達が三角形や四角形の定義となる事柄を見つけることであって，それは実際にゲームをやるという「遊び」の中から見つけられるのである．

---

[1] イギリスのTARQUIN PUBLICATIONS扱い(1989)

[2] Ｚ・Ｐ・ディーンズ　『ディーンズ選集５　算数・数学学習の実験的研究』（新数社）p.31

またゲームをするのだから，当然，それをする人，すなわち子どもが動き出さなければ，何も始まらない．このことは，言い換えると，子ども達を中心とする授業の第一歩であるということもできるのではないだろうか．

さらに３つ目として，このゲームの対象は，小学校２年生である，ということである．そのため，隠れた図形を見るとわかると思うが，比較的簡単で，また大きい図形を用いている．そして，教育実習での経験を生かして，asuka 1 と asuka 2では，同じ正方形を用いて，それらの向きが異なるように隠してある．

ここでは，小学校２年生を対象としているということで図形は簡単になっているが，図形を変えたり，複雑にすることで他学年での教材として用いることも可能になるであろう．

3.1.4 授業案の作製

実際の教育現場でどのようにして授業にパソコンを取り入れていくのか，を考えよう．

そのために授業ではプログラムで作った図形の中で１つを取り上げて，授業案を組み立てた．実際にこの図形でないものを取り上げるときには若干の違いが出るであろうがながれとしてはこのように進むのではないかと思われる．またこの授業案の中ではこのゲームを使う場面が２回出てくるのであるがその時に，１回目に三角形を使ったなら，２回目は三角形でない図形を取り上げた方が応用としても使えるように思える．

また授業の流れとしては１時間（45 分）を目安とするのでなくて2時間を基準にして組み立てることにした．そのようにすることでこのパソコンをつかうことが突発的に終わるのではなくて子ども達の中に残るだろうし，また本来の形式張った授業からの脱出として柔軟性をはかることも期待したい．さらにパソコンを使うことと本来教えなければならないことが別々になることなく子ども達の中でも同じ授業の中での流れとして残るであろうと思う．

授業案中に出てくる図１

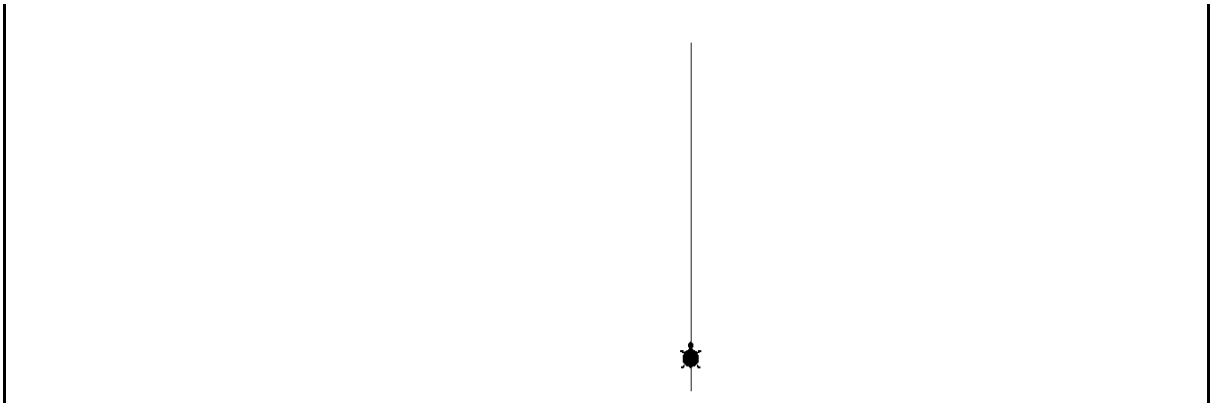


図１: ここから先に進まないパターン

Ｉ題材 三角形と四角形

Ｖ．本時の学習

１、目標

ゲームすることを通していままで「三角」と言ったものが三角形であることを知り、三角形が「３本の直線で囲まれている図形」であること、今まで「四角」と言ったものが四角形であることを知り、四角形が「４本の直線で囲まれている図形」であることを知る。

三角形や四角形の角が「頂点」と言うこと、直線が「辺」ということを知る。

２、課題

「三角形ですか、四角形ですか。」

隠れた図形が三角形であるのか、四角形であるのかを当てる。

３，準備物

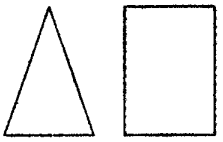
（教師）三角形、四角形の図形

画用紙

パソコン

プリント

４、本時の流れ

本時の学習	指導内容	指導上の留意点
1「三角」や「四角」を思い出す。	1年生で習った三角や四角を思い出すこと。 	・まず、三角形と四角形の図形を見せて、「1年生でこの形を習ったよね。」という。 ・子ども達は口々に「習った。」というであろう。 ・「この形は何と言いましたか。」と1つずつ尋ねる。 ・子ども達はまず三角形を見て、「三角」「三角形」と答えるだろう。 ・ここではまず「三角でしたね。」と言って三角形の図形の下に三角と書く。 ・同様にして四角形を見せて、子どもが「四角形。」や「四角。」と答えるであろうからその声を聞いて、四角形の図形の下に四角と書く。
2 ゲームの説明	・ここで扱うゲー	・「では、今日はこの三角や四角を使って、ゲ

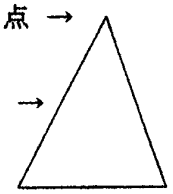
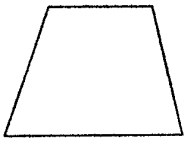
をする。	ムはasuka3とする。 キーを動かして亀が動くことを確認させる。 1→左を向く。 2→前に進む。 3→右に向く。	ームをしましょう。」と言う。 ・まずパソコンの画面上に亀を写し出す。 ・子どもは「亀だ。」「かわいい。」などと反応するであろうからその声を待って、「今日はこの亀を使って三角、四角のゲームをします。」と言う。 ・「この中には一つの形が隠れています。今日はみんなで亀さんにその形が何か教えてあげましょう。」と言う。 ・「この亀さんはキーボードを押すと動きます。誰か押してくれますか。」と言って、誰か1人を当てる。 ・「まず1を押すと左に亀が向きます。W君、押して下さい。」と言って、W君に押させる。 ・同様にして「2は前に進みます。」「3は右に向きます。」と言って、押させる。これらの説明の後でその事を書いた紙を黒板に貼る。 ・「亀は形にぶつかると動かないよ。ではみんなでW君にどのキーを押すか言いながら形を当てていこうね。形は三角か四角かどちらでもない形のどれかだよ。では、始めましょう。」と言って、始めさせる。 ・子ども達は始め一斉に「1を押して。」「2を押して。」「3を押して。」と言うであろう。そこで、混乱させないために一人づつに指示を出させる。 ・ここで注意したいのは子どもが好きなように指示を出していくことを中心にするが図1*のようになった場合は、図形が見えてこないことに気づかせる。子ども達の間で気づかない場合は教師が指示を出す。(*但し,図1は授業案の前の部分にあるものとする。) ・図形とぶつかったらそこから離れて考えるのでなく、なるべく近いところで線を動かしながらどんな図形かを考えさせる。 ・しばらくその作業を続けて図2のようになったら、子どもから声があがることを待つ。出ないときは「まだどんな形かわからないかな。」と言って子どもに疑問を持たせる。

3、隠れた図形がどんな図形か考える。		・疑問を持たせることで子どもの意見は次の4通りに分かれると考えられる。 （１）三角である。 （２）四角である。 （３）どちらかまだわからない。 （４）どちらの形でもない。
	・隠れた図形について考えること。	（１）の理由としては「なんとなく。」「角が2つあるから。」という意見が出るだろう。 （２）の理由としては「なんとなく。」「三角じゃないから。」「角がもっとありそうだから。」という意見が出るだろう。 （３）の理由としては「まだ線を引かないとわからないから。」「角が2つ見えただけで後何個あるかわからないから。」という意見が出るだろう。 （４）の理由としては「三角でも四角でもないから。」「なんとなく。」と言った意見がでるだろう。 ・一通り意見が出たらもう一度、「隠れた形はどんな形ですか。」と尋ねて、4通りに手を挙げさせる。そしてみんなが（３）であることを納得してまた線を引く。納得しないときは同じようにして意見を言わせて、討論させる。
4、隠れた図形がわかるように線を引く。		・次に「どこに線を引いたら隠れた形がわかりますか。」と言って、線を引く場所を考えさせる。子どもに前に出てきてどこに線を引いたらいいかを指示させる。 ・指示をさせながら線を引く。これを繰り返す。 ・図3のようになったら子どもから「この形がわかった。」という声が出るのを待つ。出ないときは教師が「この形がどんな形かわかりましたか。」と尋ねる。
	9のキーを押して隠れた図形を見せる。	・子どもは「三角です。」と答えるだろう。その答えを待って、「では、どんな形かだしてみようね。」と言ってキーを押す。答えは図4のような直角三角形である。
5、まとめ		・子どもにプリントを配って、授業の感想を

		書かせる。 ・次の時間にそのプリントを使うことを伝える

＜2 時間目＞

本時の学習	指導内容	指導上の留意点
1、前時の内容を思い出す。	・前時で行ったゲームについて思い出すこと。	・子ども達に前の時間に書いたプリントを用意させる。子ども達が机の上にそれを出したら、「前の時間にやったゲームの感想を言ってもらいましょう。」と言って、何人かを当てる。 ・子ども達は「面白かった。」「自分でキーを押してみたかった。」「楽しかった。」「亀が動いたのがビックリした。」といった意見を出すだろう。
2、どのようにして三角と四角を区別したかを考える。	・隠れた図形が三角か四角かをどのようにして区別したか、を考えること。	・いくつか出たところで次にどのようにして三角と四角を区別したかを思い出させる。 ・パソコンの画面に前の時間の隠れた図形を出して、「この形は三角だったけど、みんなはどうして三角ということがわかりましたか。」と尋ねる。 ・子ども達は「角が3つあったから。」「角が4つなかったから。」「角が3つしかなかったから。」「線を引いてわかったから。」「亀がぶつかったのを見てわかった。」といった意見を出すだろう。その出た意見全てを板書していく。
3、三角形は「3本の直線で囲まれた図形」であることを知る。	・三角形は「3本の直線で囲まれた図形」であることを知ること。	・「亀がぶつかったのを見てわかったから。」という意見を指して、「亀はどこにぶつかったのですか。」と尋ねて、子どもを前に出てこさせてぶつかったところを指させる。 ・「三角はいくつの線に囲まれていますか。」と尋ねて、「3本です。」という答えを待つ。その答えが出たら「今まで、三角と言っていたのを三角形と言いましょう。この三角形は3本の直線で囲まれています。」と言ってその事を板書する。

4、今まで角と言っていたところが「頂点」ということを知り直線のことを「辺」ということを知る。	・角と言っていたところが「頂点」であることを知ること。 頂点 → 辺 → 	・「角が3つあったから。」という意見を指して、「角と言っているところをこれからは頂点と言います。」と言って、黒板に三角形の図形を貼って、角を指して「ちょうてん」と書く。 ・「もう一つ新しい言葉を習います。」と言って「三角形の直線のことをこれからは辺と言います。」と言って、黒板の三角形の図形の隣に「へん」と書く。
5、四角形が「4本の直線で囲まれた図形」であることを知る。	・四角形が「4本の直線で囲まれている図形」であることを知ること。 	・次に四角形の図形を貼って「この四角の辺はいくつありますか。」と尋ねる。そして「4本です。」という答えを待つ。 ・「4本です。」という答えが出たら「そうだね。四角は4つあるよね。こらからはこの四角のことを四角形と言いましょう。そして四角形は4本の直線で囲まれています。」といって、その事を板書する。 ・「では、四角形には頂点はいくつありますか。」と言って、「4つあります。」という答えを待つ。
6、亀のゲームをもう一度してまとめをする。	・まとめをすること。 ・ここで扱うゲームはasuka2とする。	・「それでは、もう一度ゲームをしましょう。」と言って、パソコンの画面に亀を出す。 ・「亀の動かし方は前の時間と同じです。」と言って、キーの動かし方の書いてある表を黒板に貼る。 ・「では、また誰かキーを押して亀を動かして下さい。」と言って、一人を指名する。 ・前の時間と同様にして一人ひとりにどこへ進かを言わせて図形とぶつからせる。 ・途中で図5のようになったら子どもから「もうどんな図形かわかるよ。」と言ってくるだろうから、理由を付けてどんな図形かを言わせる。子どもの理由としては「亀がぶつかったから。」「頂点が3つ以上あるから。」「線を引いていてわかったから。」という意見も出るだろうが、ここでは「頂点が4つあるから。」という意見がでるまで待つ。 ・この時に気をつけたいことは頂点のことを角

	・９のキーを押して隠れた図形を出す。	という子どもがいたら、「今日は角のことを新しく何て習いましたか。」と言って、「頂点」という言葉を出させるようにする。 ・上記の場合はどの答えも「四角形」を前提にしている。この時に「三角形です。」と答える子どもがいたら「角はどこにあると思いますか。」と言って、パソコンの画面を指させて、四角形であることに気づかせる。 ・「頂点が4つあるから、四角形です。」という意見が出たら、キーを押して隠れた図形を見せる。

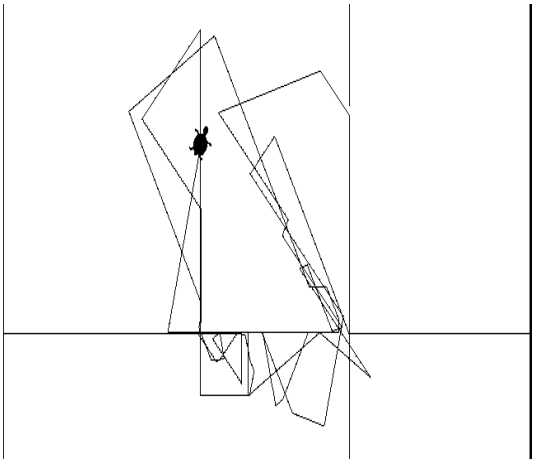


図2: 立ち止まって考えるようなパターン

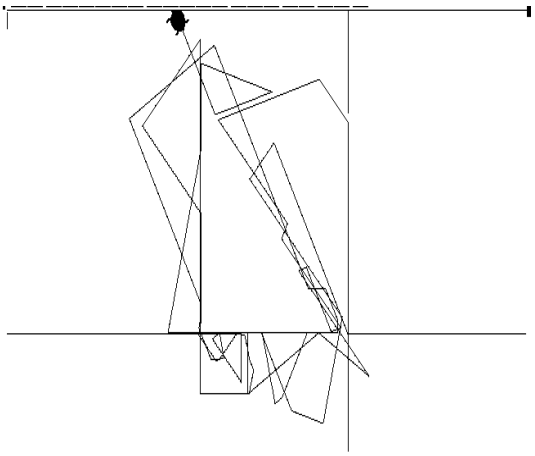


図3:正解が見つかるパターン

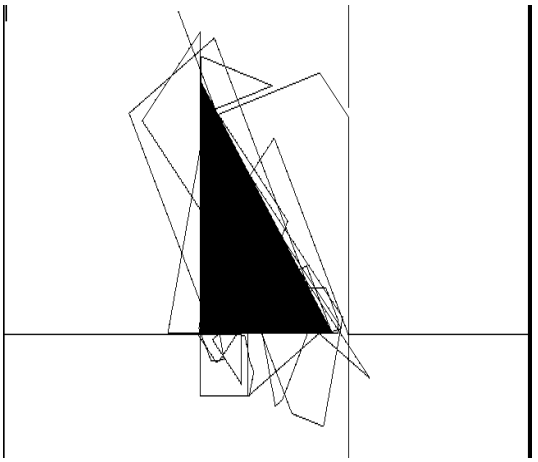


図 4:正解パターン

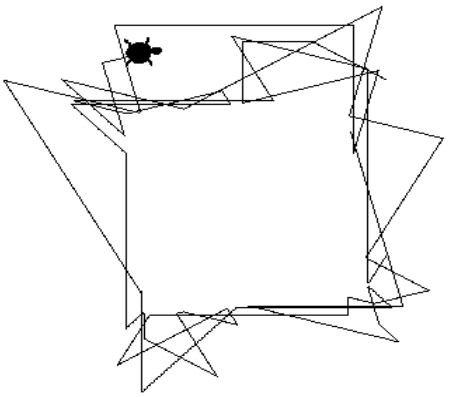


図 5:正解が見つかるパターン

3.2 算譜の作製

3.2.1 算譜の全容

```
to Asuka :n
   cg
   penup
   setpos [ 100 100 ]
   setcolor 1
   pendown
   c [/* n は図形を表す引き数 */]
   if :n = 1 [make "Vex [ pu [80 80][-80 80][-80 -80][80 -80]] ]
   if :n = 2 [make "Vex [ pu [72 88][-88 72][-72 -88][88 -72]] ]
   if :n = 3 [make "Vex [ pu [80 -80][-80 120][-80 -80]] ]
   if :n = 4 [make "Vex [ pu [80 0][0 160][-80 0][0 -160]] ]
   if :n = 5 [make "Vex [ pu [80 80][-80 80][0 -80]] ]
   if :n = 6 [make "Vex [ pu [80 0][0 80][-80 0][-60 -20][20 -40]] ]
   if :n = 7 [make "Vex [ pu [80 80][-80 40][-100 -60][100 -80]] ]
   WaitKey
   setcolor 2
```

```
   DrawFig
   setcolor 1
end
to C :Dummy
end
to Naibu?
   make "x xcor
   make "y ycor
   c [/* n = 1 まともにおいた正方形 */]
   if :n = 1 [
     output and ( ( abs :x ) < 80 ) ( ( abs :y ) < 80 )
   ]
   c [/* n = 2 ちょっと傾いた正方形 */]
   if :n = 2 [
     output and (( abs :x + .1 * :y ) < 80.8 )(( abs -.1 * :x + :y ) < 80.8 )
   ]
   c[/* n = 3 直角三角形 */]
   if :n = 3 [
     output ( and ( :x  > -80 )(  :y  > -80 )(  1.25 * :x + :y  < 20 ) )
   ]
   c[/* n = 4 菱形 */]
   if :n = 4 [
     output and ( ( abs 2 * :x + :y ) < 160 )( ( abs 2 * :x - :y ) < 160 )
   ]
   c[/* n = 5 二等辺三角形 */]
   if :n = 5 [
     output ( and ( :y  < 80 )( -2 * :x + :y > -80 )( 2 * :x + :y > -80 ) )
   ]
   c[/* n = 6 五角形 */]
   if :n = 6 [
     output ( and ( :x - :y > -80 )( :x + :y < 80 )( :x - :y < 80 )
           ( :x + :y > -80 )( .25 * :x + :y > -35 ) )
   ]
   c[/* n = 7 台形 */]
   if :n = 7 [
     output ( and ( -5 * :x + :y < 440 )( -.25 * :x + :y < 60 )
           ( .1 * :x + :y > -70 )( 8 * :x + :y < 720 ) )
   ]
end
to WaitKey
   make "Key readchar
   if :Key = 1 [ left 15 ]
   if :Key = 2 [ Walk10  ]
   if :Key = 3 [ right 15 ]
   if :Key = 9 [ stop ]
   WaitKey
end
to Walk10
   repeat 10 [
     forward 1
     if Naibu? [
        penerase
        back 1
        pendown
     ]
```

```
   ]
end
to DrawFig
   if empty? :Vex [
      penup
      setpos [0 0]
      pendown
      fill
      stop
   ]
   if ( first :Vex ) = "pu [
      penup
      make "Vex butfirst :Vex
      make "Vex lput first :Vex  :Vex
      seth towards first :Vex
      setpos       first :Vex
      pendown
      DrawFig
   ]
   make "NextPoint first :Vex
   seth towards :NextPoint
   setpos :NextPoint
   make "Vex butfirst :Vex
   DrawFig
end
```

3.2.2 算譜の解説

Asuka まず最初に亀がどこから出発するかを決める．その時，画面を座標軸のように定めて，必ず（100,100）から亀が出発するようにした．そこから亀が動いて，線を描いていく．ここでは，その時の線の色は　setcolor 1　すなわち「青」と定めている．

また隠れた図形を描くのに，図形の頂点を座標と見て，頂点がどのような座標になっているかによって図形を描くことができるようになっている．この時の図形は頂点を結ぶことによって決まる．また表す色は　setcolor 2 すなわち「赤」と定めている．

Naibu? 隠れた図形がどのような図形を描くかを示している．ここでも座標軸を基準にして，どのような領域を表しているのかを考えてそこから図形を描いている．どの図形も高校数学で学ぶ領域の考えを使って図形を考えていく．

また，この中に出てくる　abs　とは絶対値のことを表している．

WaitKey ここでは，実際にゲームを行う際の被験者の使うパソコンのキーについて取り扱っている．この時にどのキーを押せば，どのようにして進むかは容易に決めることができる．

Walk10 前の **WaitKey**　で示したことの実行をこのプログラムでは行っている．まずは，１０歩前に進むことを繰り返すことを覚えさせている．

その後の **if Naibu?** においては，隠れた図形の中に亀が入り込まないように，もし亀が図形にぶつかったら，線を描くことはできず，また **back 1** において１歩後退することを定めている．

DrawFig ここでは隠れた図形を描くことについてさらに詳しく定めている．

3.3 被験者を使ってのゲーム実践

ここでは，三角形・四角形のゲームの実践をして，被験者がどのような反応をするか，またはどのようにして亀を動かしていくか，さらにどのようにして三角形や四角形であることを見分けるかを記録した．

ゲームを実践するのは小学生と大学生であり，大学生においては小学生とは多少異なる反応を示す場合もあるだろうが，動かし方や反応については，誰がやってもかなり同じような反応をするであろうと思う．また，大学生については，今後の課題としての参考にもなると思われる．

記録については，［時：分：秒］で表し，会話でない反応は＜＞の中に記すことにした．また，インタビュアーは以下，Ｉで示し，小学生と大学生の被験者については以下，Ｓで示すことにした．

3.3.1 大学生の実践 (1)

＜画面上には**asuka 7**が設定してある．＞

Ｉ：［**20:58:17**］「１」のキーを押して．

Ｓ：［**20:58:42**］はい．

Ｓ：［**20:59:07**］何もならんよ．＜１回押して**15**度しか動かないためわからなかったようだ．＞

Ｉ：［**20:59:14**］もうちょい押して．

Ｓ：［**20:59:30**］ああ．

Ｉ：［**20:59:48**］そしたら次に「３」のキーを押して．

Ｓ：［**20:59:55**］わかった．

Ｉ：［**21:00:01**］「２」のキーを押して．

Ｓ：［**21:00:13**］おっ，進みましたね．

Ｉ：[21:00:23] じゃ，自由に押して，線を作っていって下さい．
＜Ｓは自由にキーを押して，線を作り始める．＞

Ｓ：[21:01:11] うーん ….

Ｓ：[21:01:47] 三角形でも四角形でもないでしょ！

Ｉ：[21:01:50] そんなに早くわかる !?

Ｓ：[21:01:55] うーん ….
＜Ｓは，また亀を動かし始める．＞

Ｓ：[21:04:30] 確実にならんなぁ．

Ｉ：[21:05:17] ぶつかるのはわかる？

Ｓ：[21:05:21] それは，大丈夫！

Ｓ：[21:05:29] あの，ぶつかってきた方にしか戻れませんか？

Ｉ：[21:05:37] んっ !? 先に「１」か「３」を押したら，違う方に動かせるやん．

Ｓ：[21:05:54] いや，それはわかるんですけど ….

Ｉ：[21:06:16] 領域に入らない限りは進むよ！
＜Ｓは頷きながらも懸命に亀を動かそうとしている．＞

Ｓ：[21:06:30] うーん，わからん．

Ｓ：[21:07:35] 領域に沿って，進めないですか？

Ｉ：[21:07:42] うまいこといったら，進むよ．

Ｓ：[21:08:53] ぶつかってから直角に進めませんか？

Ｉ：[21:09:00] いやいや．

Ｓ：[21:09:40] ここ，90度で進めないんすよ．

Ｉ：[21:09:53] あぁ，そういうとこもあるかも．

Ｓ：[21:10:53] 三角だったら斜めに線を書くこと出来ますよね !?
＜ＩはＳの言ったことに思わず感動してしまった．＞

Ｓ：[21:14:15] どんなんでも四角ですか？台形でも？

Ｉ：[21:14:23］台形も四角の仲間やで．どんなんでも四角！

Ｓ：[21:17:01］四角ですね!?

Ｉ：[21:17:10］何か理由でもある？あったら言って！

Ｓ：[21:17:40］…(線の描かれていない）領域を見て！
＜「９」のキーを押し，隠れた図形を出すと，Ｓが答えたとおり，四角形であった．＞

この記録から,

1. Ｓの[21:07:35] 領域に沿って、進めないですか？
図形全体を大きく見るのでなくて，領域の境の「辺」という１つの要素に注目している.

2. Ｓの[21:10:53] 三角だったら斜めに線を書くこと出来ますよね!?
被験者は，隠れた図形が三角形か，頂点が４つ以上ある図形か，を考えている．そのため，まだ辺があるとはわからない２頂点間に線が引けるかどうかを試みようとしている．もし引けるなら，そこは辺で[3]，引けないなら図形の内部なので，三角形ではない.

3. Ｓの [21:14:15］どんなんでも四角ですか？台形でも？
四角形の定義である「４本の直線で囲まれた図形」に戻って考えると，勿論，台形も四角形の一種である．しかし，日常言語においてはその数学的理論があやふやになっている.

ということができる.

3.3.2　大学生の実践 (2)

次にまた違った被験者の実践を記す.　＜画面上にはasuka 7が設定してある．＞

Ｉ：[20:49:16］まず「１」のキーを押して下さい.

Ｓ：[20:49:35］左に動いた.

Ｉ：[20:49:42］じゃぁ,「３」のキーを押して.

Ｓ：[20:49:49］あっ，右ね.

Ｉ：[20:49:53］最後に「２」のキーを押して下さい.

Ｓ：[20:49:59］前に進んだ！

Ｉ：[20:50:12］前に進んで，線を引いてくのわかった？
＜Ｓは頷く．＞

[3] ここでは，凹多角形でない，凸多角形のみを候補として考えている.

Ｉ：[20:50:16］そしたら，自由に勝手に動かして．
＜Ｓは自由に亀を動かしていく．＞

Ｓ：[20:50:31］すげぇなぁ！＜亀が動いたりすることに感動したようだ．＞

Ｓ：[20:50:57］なぁ，動かへんで．
＜Ｓは少し考えるために手を休めた．が，すぐにまた亀を動かし始めた．＞

Ｓ：[20:51:04］＜また亀が止まったため＞何かがあんのや！その形を当てるんやろ？

Ｉ：[20:51:13］そうそう！

Ｓ：[20:51:18］三角？

Ｉ：[20:51:21］えらい早いなぁ！三角なんや !?
＜Ｓはまた考え始めて，亀を動かし始めた．＞

Ｓ：[20:51:44］四角でもないんちゃうの？

Ｉ：[20:51:47］えー，それはなぁ …

Ｓ：[20:52:03］＜隠れた図形のことを＞真ん中にあんの？

Ｉ：[20:52:05］うん，真ん中らへんにあるよ．

Ｓ：[20:53:01］三角じゃないのは，横と … 三角じゃない！

Ｓ：[20:53:23］縦の線があるやん．で，次に横の線が引けたんな．そのことを考えてくと直角三角形しかありえへんのやけど，反対に回っていたら三角形が出来へんのさ．三角じゃないなぁ … あっ，でも斜めには動けるかぁ．

Ｉ：[20:54:00］動けるよ．

Ｓ：[20:54:10］複雑な形？

Ｉ：[20:54:23］うーん，そんなことないよ．

Ｓ：[20:54:47］三角でも四角でもない！

Ｉ：[20:54:50］何か理由はありますか？あればでいいけど …

Ｓ：[20:54:55］角が４つ以上ありそう …

Ｉ：[20:55:06］ありそう？
＜Ｓはまた考えるためか，亀を動かし始めた．＞

S：[20:54:56] 斜めになっとる？

I：[20:54:58] なっとるかもしれないよ.

S：[20:56:05] あっ，なっとるわ.

S：[20:56:58] 四角形 …? うーん，四角形もなぁ …

S：[20:57:11] 意外と難しいんやなぁ.

S：[20:57:30] なんちゅう形しとんのか，わからんくなってきた.

S：[20:58:19] 難しいんちゃう !?

I：[20:58:21] うーん，そうかなぁ …

S：[20:58:53] おかしな形やで.

I：[20:59:02] そんなに複雑な形じゃないよ.

S：[21:00:07] 台形でもない．… 四角 ?… じゃない.

S：[21:00:38] 四角じゃないっぽい.

S：[21:00:50] 角がいっぱいある気がする．どっちでもない図形？

S：[21:01:58] どっちでもない！

＜「9」のキーを押して，画面上に隠れた図形を示した．Sの予想に反し，四角形が浮かび上がった．＞

S：[21:02:11] やっぱ四角形やったんかぁ．難しいなぁ.

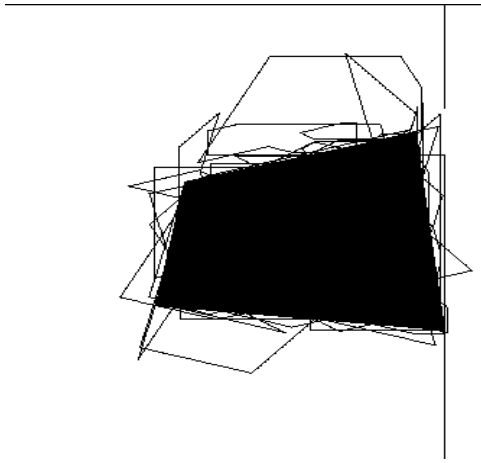


図6：Sの動かした軌跡より

＜ちょっとした雑談の後，２回目を行った．画面上にはasuka 4が設定してある．＞

Ｉ：[21:03:57］はい，始めていいよ．

Ｓ：[21:04:28］上まで行くよ．＜亀が画面の上に行ってしまった．＞

Ｓ：[21:04:37］あぁ，下から出てくるんか．

Ｓ：[21:05:31］真ん中にしかない？

Ｉ：[21:05:37］中心で考えてあるよ．

Ｓ：[21:06:54］斜めっぽい線が引けるな．

Ｓ：[21:07:05］三角じゃない？…． うーん．わからん．

Ｓ：[21:07:46］四角 …？菱形のような気がする．

Ｉ：[21:07:54］理由とかはある？

Ｓ：[21:07:58］特にないなぁ …．

Ｉ：[21:08:03］そっか．じゃぁ，正解出してみて．

Ｓ：[21:08:08］うーん，探してみるわ．

Ｓ：[21:08:30］三角形じゃないし，菱形や．あのな，平行線が引けたんさ．だからや．
＜「９」のキーを押し，画面上に正解を出すとＳの言うとおり，四角形が浮かび上がった．＞

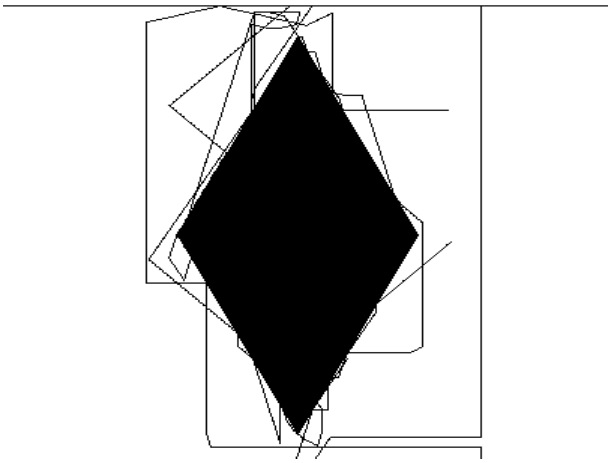


図７：Ｓの動かした軌跡より

この記録からＳの[21:08:30] 三角形じゃないし，菱形や．あのな，平行線が引けたんさ．だか

らや．という発言に注目する．この被験者は，初めは大きく線を引いて，図形の存在範囲をしぼりこんだ．そこで恐らく，四角形であると仮説を立てて，その次に四角形の種類をあてようとしている．その基準として平行な辺の組の有無に注目しているが，２本の平行線を引けたことを利用して，少なくとも平行四辺形であることを断定している．更に，線が入らない領域の対称性から，菱形と推論したのだろう．

四角形の性質による分類は，小学校２年生には，とても無理な話ではある．しかし，四角形の特徴に注目することにこのゲームが有効に作用しうることを示していると言えよう．

3.3.3　小学生の実践 (1)

＜画面上にはasuka 1が設定してある．＞

Ｉ：[18:38:55］まず，１のキー押して．
＜Ｓが押すのを見て＞もうちょい！

Ｉ：[18:39:03］亀が動くのわかる？

Ｓ：[18:39:07］うん．

Ｉ：[18:39:12］３のキー押してみて．

Ｓ：[18:39:15］うん．

Ｉ：[18:39:22］２のキー押してみて．亀が進むのわかる？

Ｓ：[18:39:29］わかる．

Ｉ：[18:39:35］じゃぁ，好きな線，書いてって．＜Ｓは無造作に亀を動かし始める．＞

Ｉ：[18:41:02］＜亀が＞止まるの，わかる？

Ｓ：[18:41:06］うん．

Ｓ：[18:41:12］そこに何かが隠れとんのかなぁ…?!

Ｉ：[18:41:20］そうそう．
＜Ｓはぶつかったところを確実にあらわそうとしていく．＞＜どんどん図形が明らかになる．＞
＜亀の向きだけでをくるくると回すことが多い．＞

Ｉ：[18:44:52］あんさ，三角形か，四角形か，そうじゃない形…で，わかったら言ってな．
＜Ｓはしばらく亀を動かしながら形を見つけようとしていたが＞

S：[18:45:03] 四角形？

I：[18:45:08] そう思ったら，９のキーを押して．

＜Sが９のキーを押すと，画面上には四角形が浮かび上がった．＞

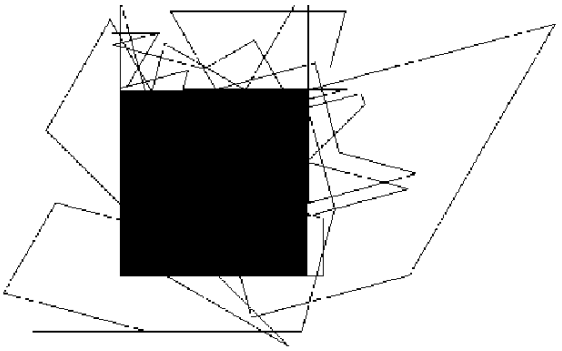


図４：Sの動かした軌跡より

＜ちょっとした雑談の後，画面上にはasuka 5を設定した．＞

I：[18:48:33] どんな感じかわかった？

S：[18:48:37] うん．

I：[18:48:43] 同じようにして，好きな線を引いていって．

S：[18:48:47] １から３，使って？

I：[18:48:51] うん．＜Sは，亀を動かし始めた．＞

S：[18:50:00] どんなんでもいいんでしょ？

I：[18:50:02] うん．

I：[18:50:10] 隠れた形がわかったら，いいよ．

＜Sは前とは違って線を何本も大きく引き始めていた．＞

S：[18:51:59] これ，またどんな形か答えるの？

I：[18:52:00] うん．

S：[18:52:53] 三角形？

I：[18:52:58] そう思う？

S：[18:53:00] うん．

Ｉ：[18:53:02] また９のキー押して．どんな形でた？

Ｓ：[18:53:10] 三角形！

＜Ｓが言った通り，画面上には三角形が浮かび上がった．＞

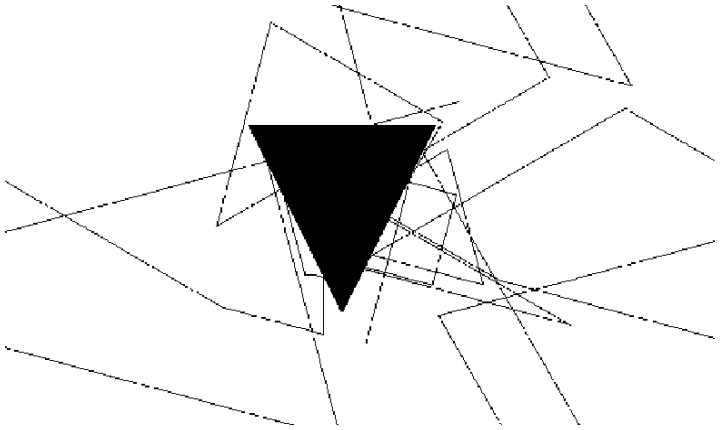


図５：Ｓの動かした軌跡より

この記録から，

1. 図形を見つけること．
   この被験者は，小学校５年生と言うこともあって，ぶつかったところにこちらが注意をひかせるとぶつかった所をなぞって隠れた図形を見つけるということを思いついている．そのことは三角形や四角形の辺の数につながることであるが，小学校２年生には少し気づきにくいことかもしれない．実際，後の小学校３年生の被験者を見ても，あまりそのことを使って形を当てようとはしていない．

2. [18:50:10] 線を大きく何本も引き始めたこと．
   １回目の実践でなぞるということに気づきながらも，「好きに引いてもいい．」という指示で自分の思うように線を引き，その一時期はちょっと隠れた図形を見つけるということを置き忘れてしまっているかのようであった．図４と図５で見比べると，図４はなぞるような収束的軌道が見られるのに対して，図５は発散的である，この違いはなぜだろうか．そのことを後日，被験者に聞いたところ，図形を見つけることを忘れていたのでなく，そのことを頭に置きながら，それよりも亀を使って絵を書こうとしたそうだ．

ということができる．

3.3.4　小学生の実践 (2)

＜画面上にはasuka 1が設定してある．＞

S：[18:20:41] まず，亀が見えるのわかる?

I：[18:20:47] うん.

S：[18:20:52] じゃぁ，１のキー，動かして.

I：[18:20:56] うん.

I：[18:21:04] もうちょっと，動かして．亀が動くのわかる?

S：[18:21:13] もう１回，押していい?

I：[18:21:17] いいよ.

S：[18:21:21] 動くのわかる！

I：[18:21:28] じゃぁ，３のキー動かして.

S：[18:21:29] うん.

I：[18:21:32] 動くのわかる？

S：[18:21:35] うん.

I：[18:21:42] 次に２のキー動かして.

S：[18:21:48] なんか動いた！

I：[18:21:55] 動くのわかる?

S：[18:21:59] うん.

I：[18:22:05] 線はわかる?

S：[18:22:09] わかるよ.

I：[18:22:11] 自分の好きなように亀を動かしてって.

S：[18:22:15] １から９を使っていいの？

I：[18:22:22] １から３のキーを使ってな.

＜Sは自由に亀を動かして，線を作り始める．＞

I：[18:22:45] どんな絵，書いてもいいよ.

S：[18:23:10]＜亀が上にはみ出した後，下から出てくるのを見て＞あれ !? あぁ！
＜亀が止まったが，気にせず続ける．＞

S：[18:25:47]＜疲れてきたらしく＞はぁ.

I：[18:26:02]亀が止まったりするのわかる？

S：[18:26:09]うん，わかるよ！

I：[18:26:22]止まるよなぁ，亀.

S：[18:26:28]そこに何かがあるの !?

I：[18:26:30]おっ，そんな感じ！止まらんように動かしてみ.

S：[18:26:47]＜Sは画面上の右上を指さして＞こっちに？

I：[18:26:50]どっちでもいいよ.

I：[18:27:02]亀がさぁ，止まるとこに何かあるの．何か当ててみて！

S：[18:27:15]ここだけ止まるんじゃないの？

I：[18:27:30]そこだけじゃないかも.
＜Sは亀が止まることを最初に較べたら気にしつつも，無造作に動かす．＞

S：[18:28:18]あれ，止まる．＜画面上の図が完成とは言えないが …．＞

S：[18:29:02]三角形？

I：[18:29:09]三角形？そう思う？

S：[18:29:15]ちょっと待って.

S：[18:30:09]三角形でも四角形でもどっちでもないわ.

I：[18:30:22]そう思う？

S：[18:30:35]うん.

I：[18:30:40]じゃぁ，９のキー押してみて．赤で隠れてたのが出るよ.

S：[18:30:45]うん.
＜画面上には四角形が浮かび上がった．＞あっ，四角形やったんや.
＜ちょっとした休憩の後＞

Ｉ：[18:31:30］やり方わかった？

Ｓ：[18:31:36］隠れた形，見つけるんやろ?! わかった.

＜画面上にasuka 2を設定して＞

Ｉ：[18:32:22］そしたら，次やっていいよ.

＜Ｓは考えながら画面上の亀を動かし始める．＞

Ｓ：[18:33:07］＜手で動かした線をなぞってみる．＞

Ｓ：[18:35:58］四角形？

Ｉ：[18:35:59］そう思ったら，９のキー押して.

＜Ｓが９のキーを押すと，画面上には四角形が浮かび上がった．＞

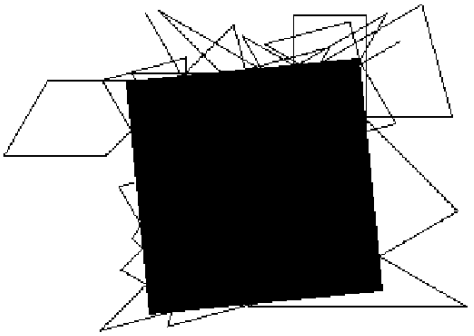


図３：Ｓの動かした軌跡より

この被験者は小学校３年生ということもあって，小学校２年生に一番近い実践記録となったと思われる．また，この被験者は家にパソコンがあって，パソコンを触ることには慣れている．そのため，パソコンへの抵抗感はなく，むしろゲームを楽しみにしていた.

この記録から,

1. 亀を無造作に動かすこと.

   亀の動きを見て，亀が急に止まってしまうことを気づかせようとしたが,「好きなように亀を動かしてもいいよ.」と言ったのが逆効果で，亀が止まることにはあまり疑問を持たずに次へ行ってしまった．このことは被験者が絵を描くということに集中していたのだと思われる.

2. ディスプレイに対する疲労.

   この疲労に関しては，Ｓの［18:25:47］はぁ．という部分でよく表れている．この疲れが授業中では「飽き」につながっていく危険性がある．これは被験者の中で「絵を書く.」と

いうことが先行してしまって，本来の課題である「隠れた図形を見つけること.」があやふやになってしまたからだろう.

ということができる.

3.3.5　実践のまとめ

４つの実践の記録からわかることは,

1. ゲームを扱う上での言語に関して.
   小学生でも，大学生でも，使う言葉に違いはあっても，ゲームを扱う上では，たいした差はなく, 十分に楽しめるものだった.
   なお，大学生の実践(1)の中のSの [21:14:15］どんなんでも四角ですか？台形でも？という発言は，大学生でも日常言語では、台形が四角形の一種ではないとしていて，数学的理論に基づいた言い方をしていないといえる一例である.

2. ゲームからの数学化.
   ゲームから子どもは数学化していく．そのきっかけは，亀があるところで進めずに止まってしまうところにあった．そこから隠れた図形を見つけるという「課題」を見いだすことが出来る．このような流れから，このゲームのねらいである三角形や四角形の角や辺に目をつけることができ，それが数学的知識となるのである.
   このゲームには「コツ」がある．その「コツ」とは，図形の特徴をつかむことである.
   しかし,「遊び」に，あるいは「遊び」よりも図形を見つけるという「課題」に集中するなど人によって違うので,「コツ」を見つける時間や難易度に差が出てしまう.
   授業では，大学生の図形の見つけ方の「辺」を基準にするという「コツ」を子どもが気づかないなら教師側のヒントとして子どもたちに提示しても良いだろう.

3. 子どもとゲームに関する環境について.
   授業では、パソコンを一人に1台ではなく，教室に1台にして，子どもどうしのやりとりをさせ，考える時間や議論する時間を増やすことができる．そうすることで，他の人の意見を聞いて，考える幅に広がりを持たすことができるだろう.
   また，適宜ディスプレイから目を離させる機会を作る等してディスプレイに対する疲労を軽減させる必要がある.

3.4　今後の課題

このプログラムを作っていくときに，私は教育実習のことを生かしてあくまでも小学校２年生を対象にして，隠れた図形を考えた．しかしながら，このプログラムというのは，隠れた図形一

つでかなり応用のきくものであり，三角形や四角形の図形を増やすことで小学校３年生での具体的な三角形を習う時期にもつながるだろうし，多角形をたくさん入れて，頂点や辺を考えさせることで５年生で習う多角形にも発展するだろう.

また，パソコンを使って新しい授業を組むということで，変化のある授業の一つになったと思うし，これからもこのようにパソコンを使って授業することを考え，取り組んでいきたいと思う.

（上野明日香）

3.5　今後の課題

このプログラムを作っていくときに、私は教育実習のことを生かしてあくまでも小学校２年生を対象にして、隠れた図形を考えていった。しかしながら、このプログラムというのは、隠れた図形一つでかなり応用のきくものであり、そのことは、大学生を使った被験者の実践でも明らかになったと思う。例えば、三角形や四角形の図形を増やすことで小学校３年生での具体的な三角形を習う時期にもつながるだろうし、多角形をたくさん入れて、頂点や辺を考えさせることで５年生で習う多角形にも発展するだろう。

また見方を変えると平行線を考えることになったり、直角を考えることにもつながっていくと思う。

このようにして、ひとつのものから、限りない可能性があるのであって、その可能性は授業で実際に使ってみて、次につなげるための材料になると思う。そのためにも、教材というのは「ここでしか使えない。」と断言するのでなくて、いろいろなところに使える可能性を見いだしたり、また使えるように工夫をしていくことではじめて生きた教材になると思う。そのため、今回は小学校２年生の授業の中に取り入れることを考えたけれど、今後は他の授業の中にも取り入れることを考えていきたいと思う。

（上野明日香）

Chapter4

「21ゲーム」を取り入れた授業

4.1　ロゴによる「21ゲーム」の授業

4.1.1　はじめに

コンピューターを用いた授業で，コンピューターを道具として使うために「21ゲーム」を算数の教材に思いついた．それはコンピューターを先生とするCAI的な利用ではなく，子どもたちがゲームの必勝法を考えていく中で，様々にサーポートできる道具になるのではないかと考えたからである．さらに子どもたちがこのゲームで必勝法を考えていく過程は，いろいろな数学的な考え方に触れることができる．よってこの教材を授業で扱うことにより，その数学的な考え方の習得と，ゲームの攻略に算数が使われていることの発見に興味や関心を湧かせ，いっそうの算数に対する意欲の向上をねらった．ここでは以下のことを十分に考えながら研究を進めていきたいと思う．

è このコンピューターを用いた授業は子どもに対して何を教育しようとするものか

è コンピュータのプログラム設計を通じて，子どもの情報に対する作業の教育的意義．

「21ゲーム」のルールは次の通りである．

21ゲーム

今，21本のマッチ棒があります．そのマッチ棒を２人で取り合いをします．しかし一回に取れるマッチの数は３本までです．（１本でも２本でも３本でもかまいませんが，必ず１本は取ってください．）交代で取っていき，最後のマッチ棒を取れた人を勝ちとします．

ここでいう「21ゲーム」とは，ニム・ゲーム[1]としても知られている．

> ニム・ゲームとは，各プレイヤーがいくつかの制限を守りながら，銅貨（この場合は，マッチ棒である）を取り去っていき銅貨を取る最後の人になる，もしくは最後

[1] W.A. ウィケルグレン，矢野健太郎訳 『問題をどう解くか』秀潤社，1974/1980

の人にならない，ということを試みるゲームである．ニム・ゲームは，最適の戦略を決定するのに逆向きに考えていくことの有用性の素晴らしい例を与える．

そしてさらに，Logo の関連のホームページで実際にこの「21ゲーム」（ニム・ゲーム）をプログラミングし実践されている秋田先生にであった[2]（秋田先生の実践については後で考えていきたい.）．秋田先生はニム・ゲームを21本のマッチ棒でゲーム化し，小学校高学年対象の教材として扱われていた．私はこの先生の実践を参考にし，授業の計画を立てた．

4.1.2 「21ゲーム」のプログラム

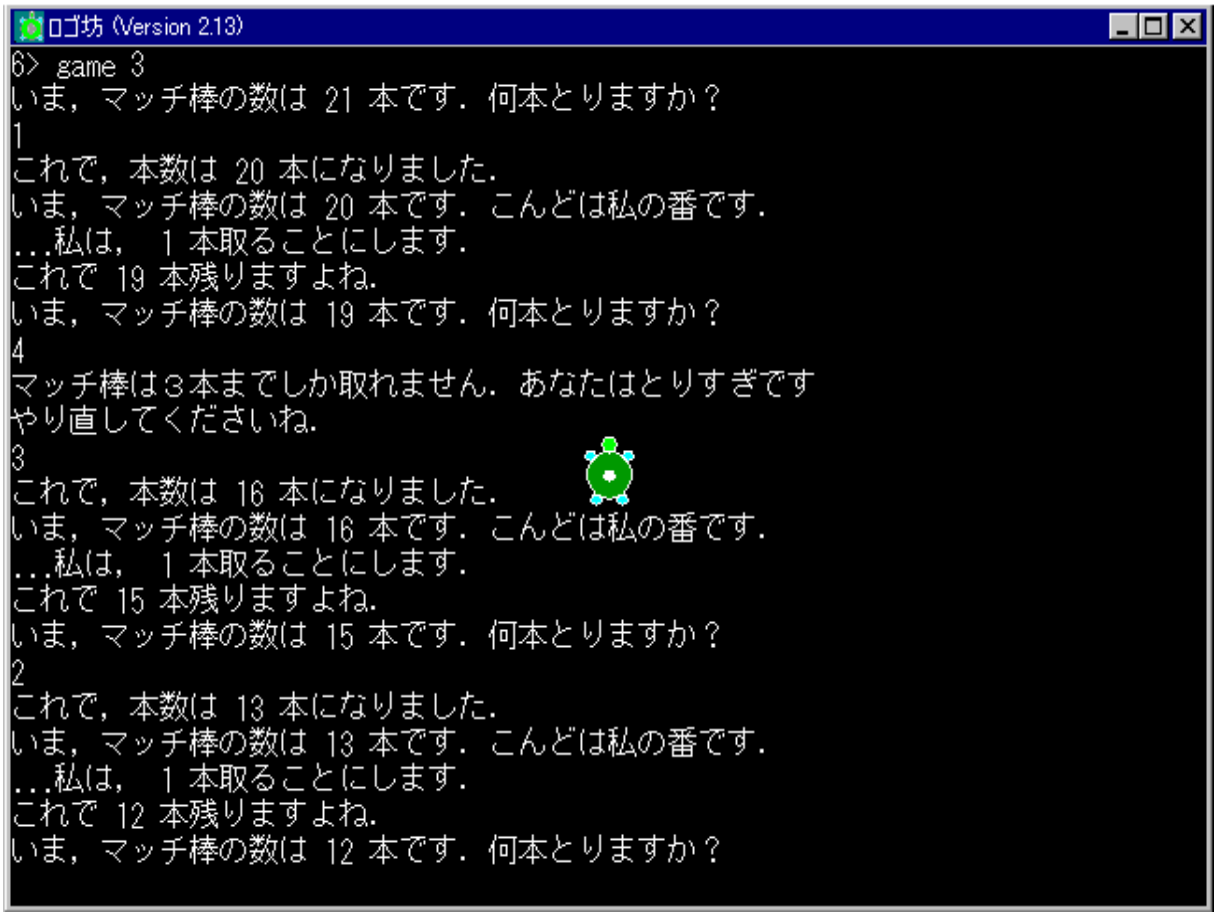


図１： 実際の画像例

この授業を考えていくにあたって，Logo 坊で試作を作ってみた．この21ゲームを Logo 言語で記述すると次のようになる．なお，後攻の競技者としてパソコンを設定し，プログラムを起動す

[2] http://plaza20.mbn.or.jp/～akitalogo/

るときに記すパラメータによって，このゲームの必勝法の発見に結びつくようにしてある．すなわち，次のような特色を持つプログラムをまず作った．

è コンピュータのゲームに関する強さ設定[3] ができること．

Game 1 † : コンピュータがあたかも気まぐれのように乱数を用いて選ぶ

Game 2 † : 初めのうちは気まぐれに（乱数），あとから必勝法を意識して選ぶ

Game 3 † : 必勝法を意識して選ぶ[4]

è コンピューターとの対戦をするゲームであること．

```
to game :x
   subgame 21
end

to subgame :n
   make "honsuu :n
   print ( list "いま，マッチ棒の数は :Honsuu "本です．何本とりますか？ )
   make "Aite readnum
   make "Honsuu :n - :Aite
   print ( list "これで，本数は :Honsuu "本になりました．)
   if :Honsuu < 1 [
       print "そうです．これであなたの勝ちです．おそれいいりました．
       stop
   ]
   print ( list "いま，マッチ棒の数は :Honsuu "本です．こんどは私の番で
.  )
   if :honsuu < 4

       print "じゃあ，全部私がとれますね．これで私の勝ちです．
       stop
   ]
   make "J jibun :x
   repeat 3 [ wait 1 type ". ]
   print ( list "私は， :J "本取ることにします．)
   make "Honsuu :Honsuu - :J
   print ( list "これで :Honsuu "本残りますよね．)
   subgame :Honsuu
end

to Jibun :x
   if :x = 1 [ output 1 + random 3 ]
   if :x = 2

     if :Honsuu < 10 [ output Jibun 3 ]
     output Jibun 1
```

[3] なおコンピュータの強さは数の大きくなるにつれて，強くなっていくように作ってある．

[4] 必勝法についての説明は後に記す．

```
   ]
   if :x = 3

     if 0 = remainder :Honsuu 4 [ output Jibun 1 ]
     output Majime :Honsuu
   ]
end

to Majime :Honsuu
   make "Mokuhyou    4 * (int  (:Honsuu - 1)  / 4 )  + 1
   output :Honsuu - :Mokuhyou + 1
end

to readnum
   make "rn  first readlist
   if :rn > :Honsuu [
      print (list "残っているのが :Honsuu "なのに :rn "本取れるわけないじゃ
ないですか)
      print "やり直して下さいね.
      output readnum
    ]
   if :rn > 3

      print "マッチ棒は３本までしか取れません．あなたはとりすぎです
      print "やり直してくださいね.
      output readnum
   ]
   if :rn < 0 [
      print "まじめにやってね．元に戻したらいかんよ.
      print "やり直してくださいね.
      output readnum
   ]
   if :rn = 0 [
      print "かならず１本はとってね．やりなおしてね.
      output readnum
   ]
   output :rn
end
```

4.2　知識・理解・技能・応用などの評価の側面

4.2.1　数学的考え方について

小学校指導要録には，高学年のところに次のように記されている[5]．

> 知識と技能の修得や活用を通して数学的な考え方を身につけ，理論的に考えたり，発展的，統合的に考えたりする．

[5] 文部省『新しい学力観に立つ算数科の学習指導の創造』大日本図書，1993

この数学的な考え方を子どもたちに身につけさせるために，21 ゲームの授業が果たす役割を考えていきたい．

4.2.2　この授業で扱う数学的考え方とは

片桐氏によると[6]，数学の方法に関係した数学的な考え方を 10 に分類されている．その中で，帰納的な考え方，抽象化の考え方，単純化の考え方，一般化の考え方，記号化の考え方，以上の考え方がこの授業で子どもたちに期待できる数学的な考え方であろう．

1. 帰納的な考え方

   片桐氏は[7]帰納的な考え方を次のように述べている．

   - いくつかのデータを集める．
   - それらのデータの間にみられるルールや性質を見出す．
   - そのルールや性質が，そのデータを含む集合（変数の変域全体）で成り立つであろうと推測する．
   - この推測した一般性が真であることを確かにするために，新しいデータで確かめてみる．

   この定義に沿ってかんがえると，この「21ゲーム」で，何回もゲームをして，相手が勝ったパターン，自分が勝ったパターンをのデータを集めて，分析し，ある発見した必勝法の法則を確認する，といった思考過程を踏まえると，帰納的な考え方をしたといってよいであろう．

2. 抽象化の考え方

   抽象化の考え方について片桐氏は[8]，1 つまたはいくつかの性質を引き出そうとする，抽象しようとする考え方と，それと表裏の関係にある捨象しようとする考え方を 1 つにあげている．

   この定義に沿って考えると，どんな色でも，形でも，マッチ棒はマッチ棒で，本数で数えるときは同じであると単に棒などで表すといった，考え方を進めた子どもは，抽象化の考え方をしたと，考えていいであろう．

3. 演繹的な考え方

   片桐氏は[9]，演繹的な考え方の中に解析的な考え方を紹介している．これは，逆向きに考え

[6] 片桐重男『数学的な考え方の具体化』明治図書，1988
[7] 片桐重男『数学的な考え方の具体化』明治図書，1988，p.130
[8] 片桐重男『数学的な考え方の具体化』明治図書，1988，p.171
[9] 片桐重男『数学的な考え方の具体化』明治図書，1988，p.128

ることを定義したものである．このことについては，後で詳しく考えていく．

4. 一般化の考え方

片桐氏は[10]問題を解決するために，そこにみられる一般性を見出したり，問題の解決をもとにして，この問題を含む集合全体で成り立つ一般性を求めていくという一般化の考え方を定義している．

この「21ゲーム」における一般化の考え方については，秋田先生の実践のねらいにも書かれている．

秋田先生の実践のねらいは次のようである．

> 子どもたちには，まず以下のことを見つけてほしい．唱える数が４つおきになっていること，最後の１は，21を４で割った余りであることなどを見つけていけると考える，すると，次に,「21ゲーム」ではなく「31ゲーム」の必勝法,「41ゲーム」の必勝法も，計算で出すことができる．つまり，一般化できるのである．「21を言った方が負け」というゲームもできるようになっている．最初のゲームで必勝法を理解した子は，今度は，試行錯誤ではなく，論理的に必勝法を見つけることがでるはずである．[11]

すなわち,「21ゲーム」を含む集合を考えたとき,「31ゲーム」,「41ゲーム」,「Nゲーム」という集合内の要素全てについて，同じように必勝法が考えられることが片桐氏の一般化の考え方に当てはまるのである.「21ゲーム」の学習の発展として，ルールを変えおこなう授業をしたとき，子どもは一般化の考え方をしたといっていいであろう．

4.2.3　解析的な考え方

杉山氏[12]によると，証明ができないとか，証明が苦手な生徒が多いという．そこで杉山氏は原因を次のようにあげている．証明ができないのには，筋道を立てて正しく推論することができない，論証の糸口が発見できない，補助線がみつけられない，どう考えていってよいのか分からない，ということである．私も家庭教師や塾の講師の経験してきたことから思い当たることである．私が生徒の数学の勉強で図形の証明を教えていたときに，思い浮かんだのであるが，証明問題を考えていく中で，逆の思考が定着していないのではないかということだ．つまり，証明問題を進めるにあたって，Aの条件からDのことを証明する場合，必要十分条件が成り立つ範囲で

[10] 片桐重男『数学的な考え方の具体化』明治図書，1988，p.178
[11] http://plaza20.mbn.or.jp/～akitalogo/jugyou/21game.htm　引用
[12] 杉山吉茂『公理的方法に基づく算数・数学の学習指導』東洋館出版，1986，pp.106-107

言い換えていく．しかし注意しなくてはいけないことが，記述と思考が逆であることだ．この点も杉山氏は同書で述べられている．例えば，証明を進めるにあったって，頭のなかでは，Dを言いたいんだから，Cが言えれば良くて，Cを言うためには，Bが言えればいい．そのBはAの条件から言えるから，証明ができると考える．しかし，記述する段階においてはAだからB，BだからC，CだからDよって証明終わり，と書く．まさしく逆の思考で物事を考えていることになる．その発想に辿りつくためには，中学生にとって多くの時間と練習を必要とする．ウィケルグレイン氏は[13]，逆向きに考える有効さを，ニム・ゲームの例を用いて述べている．ニム・ゲームと，「21ゲーム」の共通点は第1節で述べた通りであるように，以上の事実から，逆の思考の有用性を定着させるうえで，21ゲームが足がかりになるのではないだろうかと考えられる．

4.3　21ゲームの批判的見解

ロゴ坊による「21ゲーム」のプログラムは試作であったため，授業をおこなうとなると，やや工夫が足らない．そこで，秋田先生のプログラムも参考にさせていただきながら，プログラムの改良を考えていきたい．

4.3.1　グラフィックを用いたプログラム

第２節で掲載してあるプログラムを実際に起動する，とまず一番最初に物足りなさを感じるのが，文章で表されたコンピューターのコメントである．（２節の図参照）この実践計画は小学校の高学年を対象としているが，この表示のしかたでは子どもたちはすぐに飽きてしまうだろう．秋田先生も[14]登場人物にドラえもんや，ピカチュウなど子どもたちが喜びそうなキャラクターをプログラム中に登場させている．まず最初に，子どもたちにとって関心や，やりたいと思う気持ちを持たせるために，グラッフィックを用いたプログラムに変更しようと思う．その手段として，ロゴライターの主役である亀をこの「21ゲーム」のなかでも主役として，21匹の亀を子どもたちに取らせ，いろいろな所に移動していくといった演出をしていきたい．

4.3.2　条件をつける

21ゲームの試作プログラムの場合，必勝法に関するヒントがかなり与えられている．例えば「今何本残っています」と表示されることで子どもたちに残りの本数に注目しなくてはいけないと暗にヒント与えている．しかし実際のマッチ棒を用いたゲームをする段階を考えると，山積みにされているマッチ棒を子どもたちは目の前に見る．そこから，必勝法をあみだそうとすれば，子どもたちは取った本数や，真中に残っている本数，順番そして色をつけたり，さらに大きさなどの違うマッチ棒を用いれば，それらに着目する子どもも出てくるであろう．もしくは，教師が予想

[13] W・A・ウィケルグレイン著，矢野健太郎訳『問題をどう解くか』秀潤社，1974/1980
[14] http://plaza20.mbn.or.jp/～akitalogo/zisaku.htmでダウンロード可能

もしなかった考え方が飛び出てくる可能性だってある．しかし，それらの考え方では，どれも必勝法にたどり着くことができない．どんな条件が与えられていても，マッチ棒はマッチ棒であり，問題を解くことに条件が左右されることはない．すなわち，子どもたちに抽象化，単純化，記号化という数学的な考え方を身に付けさせるために，本数という条件だけでなく，様々な条件を付け足したプログラムにしていきたい．それは，以下の通りである．

è 亀に色をつける．（３色用いる）

è 亀に出席番号をつける．

è 自分の取った亀と相手の取った亀の履歴を残す．

è 真中に，残っている亀を表示する．

4.3.3　21匹亀さんゲームのプログラム

以下は，21匹亀さんゲームを実現するための簡単な算譜である．Logoの処理系であるLogo Writer for Winに準拠して書いてある．Loadしてから，下記のようにすれば実行が開始される．

s　†でゲームがはじまる．なおこのゲームは子ども同士の対戦型プログラムである．

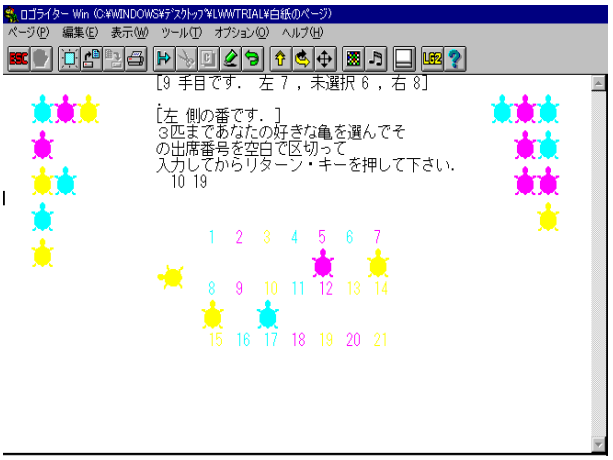


図２： 画像例

```
to InitialPos :Id
  local [ i ix iy ]
  c [/* :Id は亀の出席番号    1～21*/]
  make "i    :Id - 1
  make "ix remainder :i  7
  make "iy int ( :i / 7 )
  output ( list -90 + :ix * 30    50 - :iy * 50 )
end

to Iro  :Id
   local [ L ]
   make "L [ 5 3 6 5 3 5 3   5 3 6 5 3 6 6  6 5 5 3 6 3 6 ]
```

```
   output item :Id :L
end

to Initial_Stamp
  local [ i ]
  make "Non_Selected [ 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
21]
  ht
  seth 0
  make "i 0
  repeat 21 [
     make "i    :i + 1
     tell :i    st
     penup setpos InitialPos :i
     pendown setc Iro :i
     penup back 25 pendown label :i penup
         forward   25
  ]
end

to startup
   cc cg cleartext  show "s
   print_ "さあ！21匹亀さんゲームをしよう.
   print_ "対戦相手の友達を探してくれ.
   print_ "ゲームのルールは，交代で亀を取り合っていき,
   print_ "最後の亀を取った人が勝ちだ.
   print_ "しかし，一回に取れる亀は３匹まで,
   print_ "そして必ず一匹は取らないといけない.
   print_  [さあ，Let's Try！]
    Waitkey
    Initial_Stamp
    Game 1 0 21 0
end

to s
  startup
end

to Waitkey
    local [Dummy]
    print_ ".
    print_ [ スペースバーを押してください. ]
    make "Dummy readchar
    cleartext
end

to Game :k :NLeft :NMid :NRight
  local [L N]
  cleartext
  print_ (list :k "手目です.  "左 :NLeft ", "未選択 :NMid ", "右 :NRight )
  print_  ".
  if :NMid < 4 [
      print_ ( list Left_or_Right "側の勝ちです. )
      stop
  ]
```

```
  print_ ( list Left_or_Right "側の番です. )
  print_ "３匹まであなたの好きな亀を選んでそ
  print_ "の出席番号を空白で区切って
  print_ "入力してからリターン・キーを押して下さい.
  tell 25   ht
    penup  setpos [ -100 -150 ] pendown st
    label  "|                                                          |
    penup  setpos [ -100 -150 ] pendown st

    ht
  make "L Read_1to3
  make "N count :L
   T_Move :L :k 1
  if ( remainder  :k 2  ) = 1 [ Game :k + 1 :NLeft + :N  :NMid - :N
:NRight ]
  if ( remainder :k  2 ) = 0  [ Game :k + 1 :NLeft         :NMid - :N
:Nright + :N ]
end

to Left_or_Right
  output item ( 1 + remainder :k 2 ) [ 右 左 ]
end

to T_Move :L  :k   :h
   local [ i Moku]
   if empty? :L [stop]
   make "i first :L
   if not member? :i :Non_Selected [ print "その亀さんはもういません. もう一回！
   WaitKey
   cleartext
   cg
   s
   ]
   make "Non_Selected But :i :Non_selected
   tell :i  slowturtle
   seth 0    st
   make  "Moku  Mokuhyou ( -1 ) ^ :k   ( int  ( :k - 1 ) / 2 )  :h
   c [ /** show (list "## :i who :k :h :Moku ) **/ ]
   seth towards  :Moku
   forward  distance :Moku
   seth 0
   pendown stamp penup
   fastturtle
   T_Move butfirst :L                :k          :h + 1
   c [/*  選ばれて動いてない  何手目    その手のなかで何匹目*/]
end
to But :x :L
   c[/* :L のうちで:x以外*/]
   if empty? :L [output [ ] ]
   if :x = first :L [ output butfirst :L ]
   output fput first :L But :x butfirst :L
end

to Mokuhyou :s :n :h
      c [/* 先攻なら :s=-1, 後攻なら：s=+1 */]
```

```
   output ( list  :s * ( 300 - 25 * :h )      150 - :n * 35 ）
end

to print_ :x
   print ( list "|                  | :x    )
end

to Read_1to3
   local [L K]
   make "L readlist
   make "K count :L
   if :K > 3 [
   print_ "３匹までですよ.
   WaitKey
   cleartext
   output Read_1to3
   ]
   if :K < 1 [
   print_ "どれかは選んでよ
   WaitKey
   output Read_1to3
   ]
   output :L
end

to c :dummy
end
```

4.4　授業実践への利用

授業の構想は最初に改良型プログラムで対戦させ，後では，試作型プログラムでのコンピューター対戦という流れでおっていきたい.

改良型プログラムを用いて，友達と対戦し，ゲーム勝つためのデータをとる. :::::::::: １時間

試作プログラムを用いて，必勝法を考える.　:::::::::::::::::::::::::::::: １時間（本時）

4.4.1　この授業のねらい

1. 数学的考え方における，抽象化の考え方を身に付けさせたい.　::::::::::::::::::第１時

2. 逆向きの考え方の有効性をかんがえさせたい.　::::::::::::::::::::::::::::::第２時

4.4.2　「21ゲーム」学習指導計画

本時の学習

1. 目標 逆向きに辿っていく方法を用いて，必勝法を考える.

2. 課題 21本のマッチ棒があります．最後に，マッチ棒を全部取った人が勝ちとします．一回交代でゲームを進めていきます．ただし一人の人が取ることのできるマッチ棒の数は3本までです．さて，このゲームで必ず勝つ方法はあるでしょうか？

3. 準備物 コンピューター（児童の人数分），ロゴ坊による自作ソフト「21ゲーム」．

4. 過程　（45分）

4.4.3　授業の流れ

授業の流れを把握するために，実際に授業をしていると想定して，架空の授業記録をとる．[15]

（前回までに，亀の残りの数に注目すれば，必勝法にたどりつきそうなことは確認してある．）

（T）：前の授業で，21匹亀さんのゲームをしましたね．このゲームで必勝法を考えるためには，亀の残りの数に注意すれば発見できそうでしたね．それでは今日は，前回取ったデータや，新しいゲームを用いて，必勝法を考えていきましょう．

（T）：今日使う新しいゲームのルールを説明します．

（T）：21本のマッチ棒があります．それをコンピューターを対戦相手として，戦います．ルールは21匹亀さんとほとんど同じで，21本のマッチ棒を交互にとっていき最後に全部取り尽くせた人が勝ちです．一回につき３本までしか取れなかったのは，同じです．さあ，コンピューターにかてるかな？やってみましょう．

（C）：（子どもたちは各自コンピューターに向かって，対戦ゲームをおこなっている．）

（C）：（勝てた，負けた，このコンピューターは強い，こいつは弱い，など口々に発している．）

（T）：（なお，各自に当てたコンピューターの21ゲームのプログラムの設定はgame1，game2，game3，をロードしてある．すなわち，コンピューターによって，あたかも強さが違うように感じるであろう．）

（C）：先生，このコンピューターは弱いのと，強いのがあるんですか？

（T）：どうして，そう思うの？

（C）：だって，A君のコンピューターは誰がやってもほとんど勝てるけど，僕のコンピューターは絶対勝てないよ．僕のコンピューターで友達にしてもらっても，勝てないよ．

[15] ( )で書いてあるのは発言以外のことを表す

（T）：そうか，コンピューターにも強さがあるんだ．でも本当にその強いコンピュータに勝てないのかな？先生がやってみようか？

（T）：（もちろん必勝法を知っているため，勝つ.）

（C）：すごい，勝てたね．なんでなん？

（T）：なんで勝てるのかをみんなで考えてみましょう．どうすれば勝てるだろうね？前の授業のときに取ったデータが役に立つんじゃない？あと，コンピューターに聞いてみたら？

（C）：そんなー，コンピューターは教えてくれやんよ.

（C）：（コンピューターと対戦したり，前回のデーターを調べたり，新しいデータを取ったりしている.）

（T）：何か気付いた人はいますか？

（C）：全然わからへん.

（C）：でも，残りの本数を考えたらよかったんと違う？

（C）：先生，強いコンピューターを調べとったら，取る本数は関係ないんやけど，残りの数を絶対に20，16，12，8，4，ってなるように取っています．それで，私も同じように残りの本数をあわせてみたら，勝てました．絶対に勝てるかどうかは自信ないです.

（T）：じゃあ，ほかの人にもやってもらって，本当にその方法で勝てるかどうかやってもらいましょう.

（C）：（コンピューターと対戦中）

（C）：先生，その方法で勝てたよ.

（C）：僕も勝てた.（ほとんどの子どもたちが確認できたようである.）

（T）：でも，どうしてそのように４の倍数ずつ残すように取れば勝てるのかな？

（C）：そんなん，わからんよ．でも，こうしたら勝てるんだからいいやんか.

（T）：でもせっかくだから，考えてみよう.

（C）：（コンピューターと対戦したり，友達と話し合ったり，絵を書いてみたり，思考錯誤しながら考えている.）

（C）：先生，やっぱり分からんよー．なんでなん？ヒントちょうだいよ.

（T）：じゃあマッチ棒の数を減らしてみたらどうかな？別に21本じゃなくても，もっと少なかったら考えやすいかな？考えやすいだけ，少なくしてみたらどう？

（C）：じゃあ，俺は3本にしてみよ．あ，いかん．4本やったらええか．

（C）：（子どもたちは，ああでもない，こうでもない，と言いながら，考えている．）

（C）：先生分かったよ．あのさ，こう考えたらいいんじゃないかな？

（C）：例えば，4本ゲームやとすると，最初に取る人は，絶対に，1本か2本か3本か取らないといけないから，最初の人が何本取っても，最初に取る人は絶対に勝てないの．だから，勝ちたい人は，4本残しておけば，絶対に勝てるというわけなんやろ．

（T）：そう，いいとこにきがついたね．最後に4本にしておかなければならないのは，さっき説明してもらった通りです．それでは，どうして，21ゲームの場合20，16，12，8，4，というように残していったら，必ず勝てるのか説明できますか？

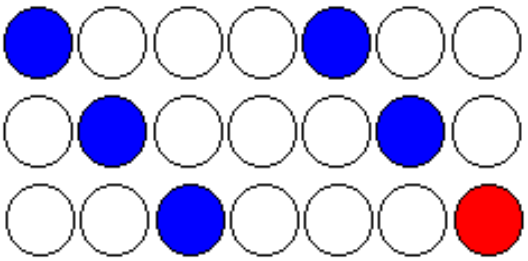
（C）：分かりました．8本残したとすれば，相手は，絶対に3本までしか取れないんだから，相手が，1本取ったとしても，残りは７本で，自分の番のときに，３本取れば，残り４本にできるし，相手が2本取ったとしても，残り6本で，自分の番のとき，２本取れば，また4本残せるし，もし，3本取ったとしても，残り5本で，自分の番のとき1本しか取らなかったら，残り4本にできるからです．あと同じように考えて，12本，16本，20本，と残していったらいいんです．

（C）：えーようわからん．

（T）：分からないといっている人がいますが，誰か，分かりやすく説明してくれる人はいませんか？

（C）：あっ，分かったよ．僕が，絵で説明してみるよ．

（C）：今マッチ棒を円で表すよ．赤色の最後の丸をとるためには，自分の番で青色の丸までとればいいんだよ．自分の取る本数は自由なんだから，最後から考えて，3つおきに青丸を書いて，さっきの説明のように取っていけばいいよ．



（C）：そうか，分かった．そうや，逆から考えていったら簡単やった．

（T）：どうですか？みなさん分かりましたか？

（T）：そうですね．逆から考えていくとわかりやすかったですね．

4.4.4　感想

今回この卒業研究を通して，コンピューターのプログラミングというものを初めて体験し，その難しさをつくづく実感した．それと同時に，教育ソフトを用いて，プログラミングし授業実践をおこなうことは膨大な時間と知識が必要であることも感じた．この卒業研究にあたっておよそ一年間の時間があったが，今現段階で，自分の力でプログラミングをする自信はないと思う．しかし，これからの教育現場で，様々なところでコンピューターが進出してくるであろうが，少なくとも，コンピューターに対する抵抗感はなくなった．そして，自分がコンピューターを取り入れた授業の研究者に加われたことは，自分の教師になってからもコンピューターを考え続けたいという意欲を持たせてくれた．この卒業研究はこれで終わりではなく，この研究を進めていく中で湧いてきた様々な疑問，推測を，これから現場に出る教師として持ちつづけ，研究していきたいと思う．この卒業研究は将来教師になりたい私にとっての足がかりとなることができたように思う．（服部　真一）

Chapter5

Logoを使って面積を求めよう

5.1　はじめに

　このプログラムを作るにあたって，まず，パソコンを用いることにより，数学の授業がよりわかりやすくすることを考えました．黒板だけでは，動画的なことは難しいので，パソコンを用いることにより，動きを取り入れ生徒たちにイメージをわかせたいと思いました．この教材を選んだ理由は，私が高校生のとき，区分求積法の授業では黒板にグラフを描かれていて，細かいところまで描くと黒板が汚くなるし見にくくなるので，省略されていました．そこで，区分する数を増やす過程が動画的で見られるプログラムがあったら面白く，よりわかりやすい授業ができると思ったからです．

　また，このプログラムは区分を増やし面積を求めていく区分求積法の過程だけでなく，マス目を数える方法やモンテカルロ法も作り生徒の考え方にも対処しました．このプログラムによっての改善点は，

1. 区分を増やすことによって本当に求める面積になりそうだと思えば，計算をして確かめようと意欲を持たせられることができること
2. 上積分，下積分などの面積を求める方法を多様に取り入れ，生徒の様々な意見に対応したこと

パソコンを使うが，すべてをさせるわけではなく，パソコンはあくまで教具として考えました．だから，生徒が自分の手を動かして計算することなどを残しました．後に書かれた授業案を見ていただいたらわかるように，考え，話し合う授業にしました．文章問題で具体的なものを表わすのに紙で作って黒板に貼っていたように，身近にパソコンが使えるようになればいいと思います．

5.2　プログラム

```
to menseki :Mode
   cg  cleartext
```

```
   showturtle
   penup setc 0
   setpos ( list GetX_Cor 0 GetY_Cor 0 )
   pendown forward 260
   Graph [  0 ] 0
   Graph [ :x ^ 2 ]  1
   hideturtle
   if :Mode = 1 [ Masu stop ]
   if and ( :Mode >= 2 ) ( :Mode <= 5 ) [ Sikaku stop ]
   if :Mode = 6 [ MonteCarlo_Method stop ]
   print [7つ以上のレパートリーはありません]
end

to Graph  :siki :Color
   local [ x y Y_cor ]
   penup  setc :Color
   Draw [ 0 0 ]
   seth 0
   repeat 320
     [ make "x GetX ( xcor + 1 )
       make "y run :siki
       make "Y_cor GetY_Cor :y
       if (abs :Y_cor) > 140
          [ penup ]
       if (abs :Y_cor) <= 140
          [ setpos
            ( list ( xcor + 1)  :Y_cor  )
          pendown
          ]
     ]
end

to Draw :p
   local [ x y pos ]
   make "x first :p
   make "y last  :p
   make "pos ( list GetX_Cor :x  GetY_Cor :y )
   seth    towards  :pos
   forward distance :pos
end

to GetX_Cor :x
   local [ x1 x2 z ]
   make "x1  -1
   make "z ( :x - :x1 )
   make "x2  2
   make "z :z / 3
   output   640 * :z - 320
end

to GetY_Cor :y
   local [ y1 y2 z ]
   make "y1  -1
   make "z ( :y - :y1 )
   make "y2  2
```

```
   make "z :z / 3
   output   640 * :z - 340
end

to GetX :x_cor
   local [  z ]
   make "z  3 / 640
   make "z  :z * ( :x_cor + 320 )
   output -1 + :z
end

to Masu
   tell [24 12 6 ]
   each
   [ showturtle
     setc who - 4
     penup
     setpos ( list GetX_Cor 0 GetY_Cor 0 )
     pendown
     JuuouMujin
     repeat 5 [
        wait 30
        insert "#
        ]
   ]
end

  to JuuouMujin
     c [/* マス目 */]
     each
     [ repeat 216 / who
        [ penup
          setpos ( list ( xcor + who ) GetY_Cor 0 )
          pendown
          forward 216
         ]
      ]
      each
      [ penup
        setpos ( list GetX_Cor 0 GetY_Cor 0  )
        pendown
        right 90
        repeat 216 / who
          [ penup
            setpos ( list GetX_Cor 0 ( ycor + who ) )
            pendown
            forward 216
           ]
       ]
      hideturtle
   end

c [---------------------------------------]

to Sikaku
```

```
   local [ Delta ]
   tell [ 24 12 6 1 ]
   each
   [ showturtle
     setc who * 3 - 2
     make "Delta who
     penup
     Draw [ 0 0 ]
     pendown
     Tanzaku :Mode :Delta
     WaitKey
    ]
end

to WaitKey
   print [Press any Key]
   make "Dummy readchar
   cleartext
end

to Tanzaku  :Mode :Delta
   local [ x1 x2 ]
   showturtle
   make "x1 GetX  xcor
   make "x2 GetX  xcor + :Delta
   Draw ( list  :x1 Y_Value 1 :Mode )
   Draw ( list  :x2 Y_Value 2 :Mode )
   Draw ( list  :x2  0 )
   hideturtle
   if 1 <= GetX xcor [ stop ]
   Tanzaku :Mode :Delta
end

to Y_Value :J :Mode
  if :Mode = 2 [ output :x1 ^ 2 ]
     c [/* 下積分 */]
  if :Mode = 3 [ output :x2 ^ 2 ]
     c [/* 上積分 */]
  if :Mode = 4 [ output Y_Value :J :J + 1 ]
     c [/* 台形則 */]
  if :Mode = 5 [ output ( ( :x1 + :x2 ) ^ 2 ) / 4 ]
     c [/* 区間の中間での値 */]
end

to c :dummy
end

c [---------------------------------------]

to MonteCarlo_Method
   c [/* モンテカルロ法 */]
   local [ r ]
   make "r 0
   repeat 100  [ Random_sampling ]
      print ( list "100回のうち，面積内は，  :r "回です. )
```

```
   repeat 900  [ Random_sampling ]
      print ( list "1000回のうち，面積内は，  :r "回です. )
end

to Random_sampling
   local [ m n ]
   make "m random 216
   make "n random 216
   setc 3
   if  :m < :n ^ 2  / 216
      [setc 2
       make "r :r + 1
       ]
   penup
   setpos ( list GetX_Cor  0 GetY_cor  0 )
   setpos ( list ( xcor + :n ) ( ycor + :m ) )
   pendown
   seth 0  fd 1
end
```

5.3　プログラムについて

「216 / who」とは，0≦x≦1 の間に216点あるので，216点を亀の番号で割った数が区分の数になります．そして，線の色は,「who Ä 4」や,「who * 3 Ä 2」で表しています．亀の番号で表わした理由は，区分した長さどうやって表わそうかと考えていたとき，はじめは亀の番号を使わずにプログラムを作っていたら，すごく長いプログラムになり，区分の長さをかえて同じことを繰り返せば，よりわかりやすいプログラムになると思ったからこのように表わしました．そして，whoを使えば亀の番号として何度も表わせるから使いました．プログラムだけを見ると，なぜ，亀の番号が分母に来ているのかと思われるかと思いますがこういう理由です．また，はじめの区分の数を10等分でプログラムを組む予定でしたが，216は10では割れないのでできませんでした.

このプログラムは「ロゴライター　for Win」を用いたプログラムです．算譜の解説として，次の通りです.

menseki :Mode 全体のプログラムのもとになるプログラムです.「Mode」は，1 から 6 まであり，その説明は，後で述べてあります.

Graph :siki :Color 「menseki」の中にある「Graph [0] 0」や「Graph [: x^2] 1」を実行させるためのプログラムです.「Graph [0] 0」と「Graph [: x^2] 1」の説明として，前者が y = 0 のグラフを黒で，後者が $y = x^2$ のグラフを青で描くことを表しています.

Draw :p p には座標が入ります．p の位置からグラフを描き始めます.

GetX Cor :x, GetY Cor :y それぞれ x，y の範囲を表しています.

このプログラムを実行するためには,「menseki」と打ち，1マス空白をあけ,「:Mode」の番号を打ち込みます．例えば,「menseki 1」のように打ちます．すると,「:Mode」の内容が実行されます.「:Mode」の内容は次のようになっています.「:Mode」の内容を選んで実行してください.

5.3.1　マスを数えて面積を出す方法（:Mode = 1）

x 軸と y 軸に平行な線を同じ区分で引きマスを作ります．そのマスを数えて面積を出す方法です．つまり，基準の正方形の面積が求める範囲にいくつ入るかで面積を求めます.

まず，図1のように，9等分します．それから，18等分して，27等分します．この変化より，区分を細かくすると面積が近似できます．以下，上積分や下積分などの区分の数もそろえた方がいいと思い，9等分，18等分，27 等分にしました.

プログラムでは,「Masu」をもとにしてあります．(0,0) に持っていき,「Juuou Mujin」で,亀にマスを描かせています．2.5秒待って，次の区分へ行くようにしました.

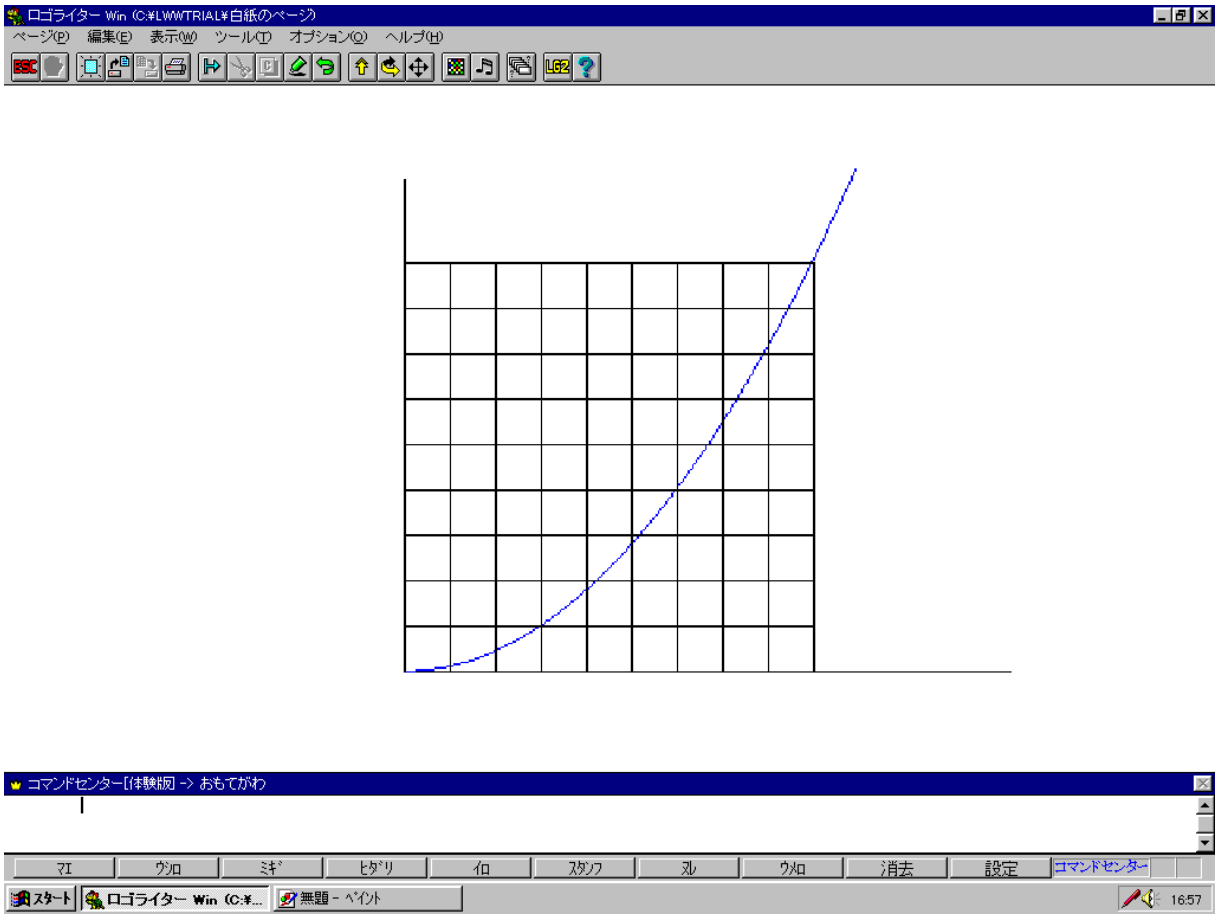


図1：マスを数える

5.3.2　下積分（:Mode = 2）

下積分で面積を求める方法です．今度は，$y = x^2$ のグラフにおいて，$0 \leqq x \leqq 1$の範囲で x 座標を等分して，等分された (x; 0) から y 軸に平行に線を引きグラフとぶつかったところでグラフの下から長方形を取り，長方形の面積を求め全てを合わせれば，面積を求めることができます．求める面積の式は，区分を n 等分すると，k 番目の長方形の面積が，

$$\left(\frac{k-1}{n}\right)^2 \frac{1}{n} = \frac{(k-1)^2}{n^3}$$

これを 1 から n まで足し合わせればいいので，

$$\sum_{k=1}^{n} \frac{(k-1)^2}{n^3}$$

プログラムでは,「Sikaku」をもとにしてあります．(0,0) に持っていき,「Tanzaku」で，x_1，x_2 を比較させることにより，亀に長方形を描かせています.「Mode」は，2 なので，y の値（長方形の高さ）は，x_1^2 で表すようにしています．この方法だと「:Mode 1」のように数えずに長方形の面積を足し合わせることにより面積が求められるので効率がいいです．

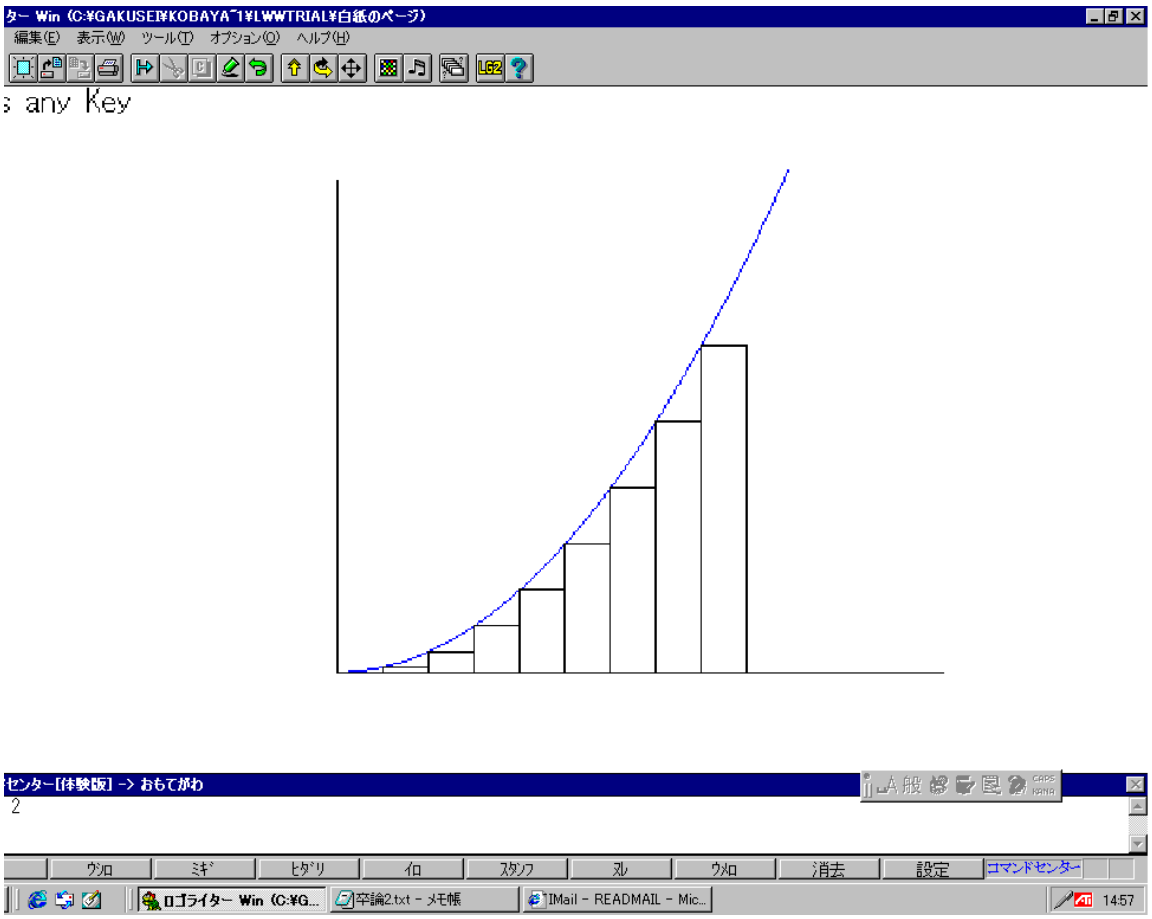


図 2：下積分

5.3.3　上積分（:Mode = 3）

上積分で面積を求める方法です．$y = x^2$ のグラフにおいて，$0 \leqq x \leqq 1$ の範囲で x 座標を等分して，等分された (x;0) から y 軸に平行に線を引きグラフとぶつかったところでグラフの上から長方形を取り，長方形の面積を求め全てを合わせれば，面積を求めることができます．求める面積の式は，区分を n 等分すると，k 番目の長方形の面積が，

$$\left(\frac{k}{n}\right)^2 \frac{1}{n} = \frac{k^2}{n^3}$$

これを 1 から n まで足し合わせればいいので

$$\sum_{k=1}^{n} \frac{k^2}{n^3}$$

このプログラムも,「Sikaku 」をもとにしてあります．下積分のところで書かれている内容とほぼ同じです.「Mode 」は，3 なので，y の値（長方形の高さ）は，x_2^2 で表すようにしています．図 3 を見れば分かるように，上積分はグラフの上の部分から長方形の面積を近似するので実際の面積より大きいところから始まります．

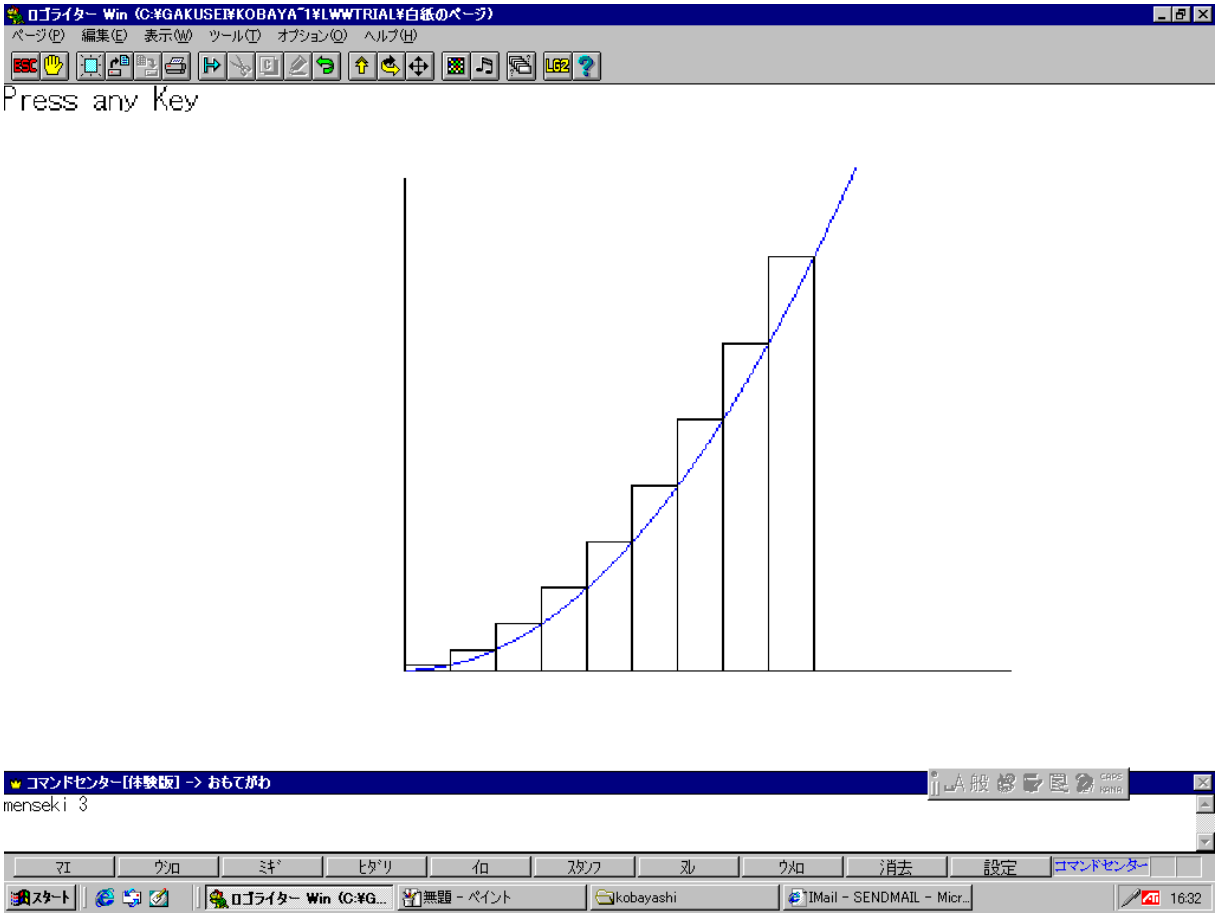


図 3：上積分

5.3.4　台形則（:Mode = 4）

今までは，長方形の面積で考えていたところを，台形の面積で考える方法です．つまり， x の範囲を n 等分したとき， (n Ä 1) 番目の y 値を上底， n 番目の y の値を下底，1割る区分の値を高さとした台形の面積を n 個足し合わせて面積を近似しています．式で表すと，

$$\left(\frac{k-1}{n}\right)^2 + \left(\frac{k}{n}\right)^2 \cdot \frac{1}{n} \cdot \frac{1}{2} = \frac{2k^2-2k+1}{2n^3}$$

なので求める面積は，

$$\sum_{k=1}^{n} \frac{2k^2-2k+1}{2n^3}$$

このプログラムも,「 Sikaku 」をもとにしてあります.「 Mode 」は， 4 なので，y の値は， x_1 では x_1^2 ， x_2 では x_2^2 で表すようにしています．曲線の面積を求めるから，長方形より台形の方が近似率がいいことを計算することによって確かめられます．実際に授業のときに生徒に計算させて近似がいいことを確かめたいです．

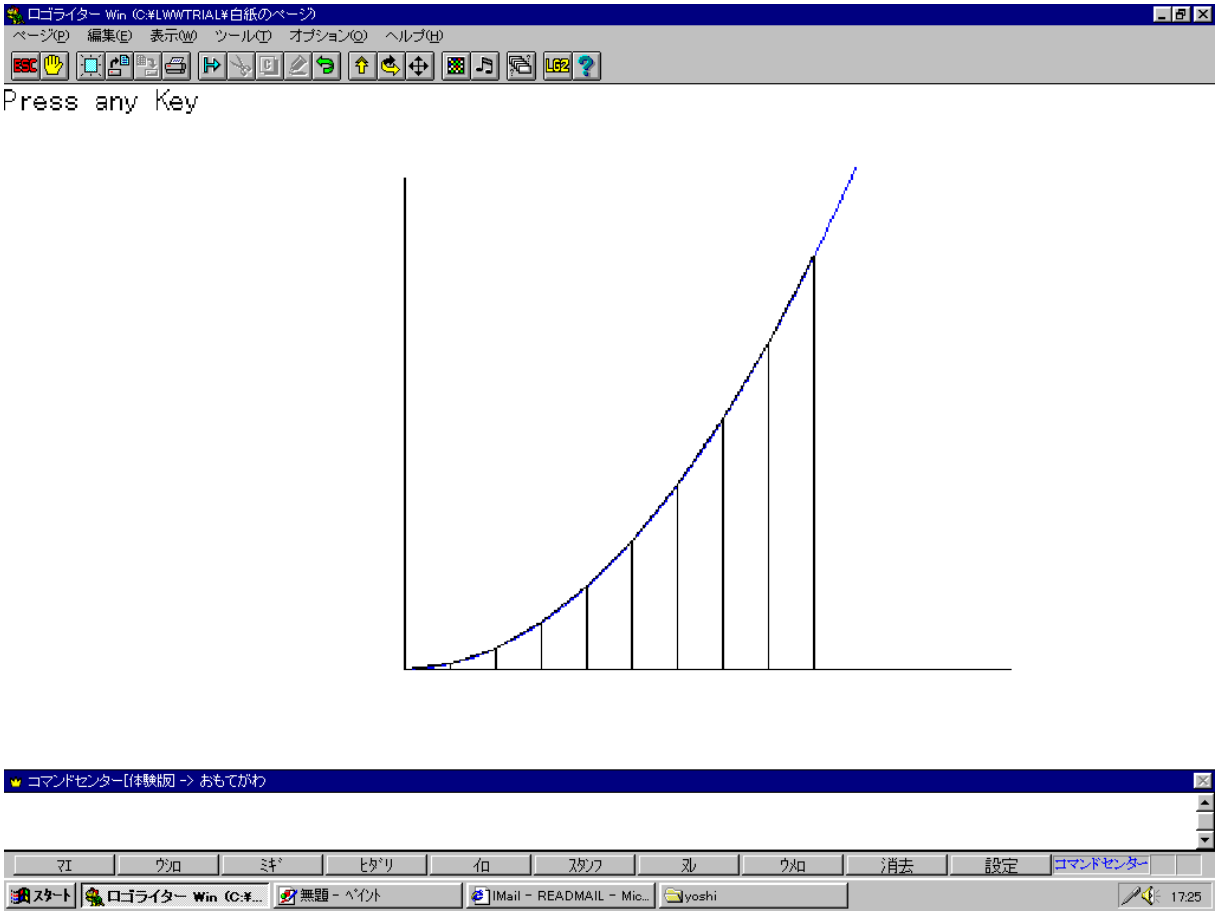


図 4：台形公式

5.3.5　区間の中間での値（:Mode = 5）

今度は下積分や上積分に似ているんですけど，区間の中間での値で面積を考える方法です．つまり，(区分の数分の１）を底辺，(区間の中間での値）を高さとして長方形の面積を，足し合わせて面積を近似しています．一般的に求める面積の式は，区分を n 等分すると，k 番目の長方形の面積が，

$$\left(\frac{k-1+k}{2n}\right)^2 \frac{1}{n} = \frac{(2k-1)^2}{4n^3}$$

これを 1 から n まで足し合わせればいいので

$$\sum_{k=1} \frac{(2k-1)^2}{4n^3}$$

このプログラムも，「Sikaku」をもとにしてあります．「Mode」は，5 なので，y の値（長方形の高さ）は，$\frac{(x_1+x_2)^2}{4}$ で表すようにしています．上積分や下積分ができるのなら区間の中間でもできると生徒が考えると思って作りました．

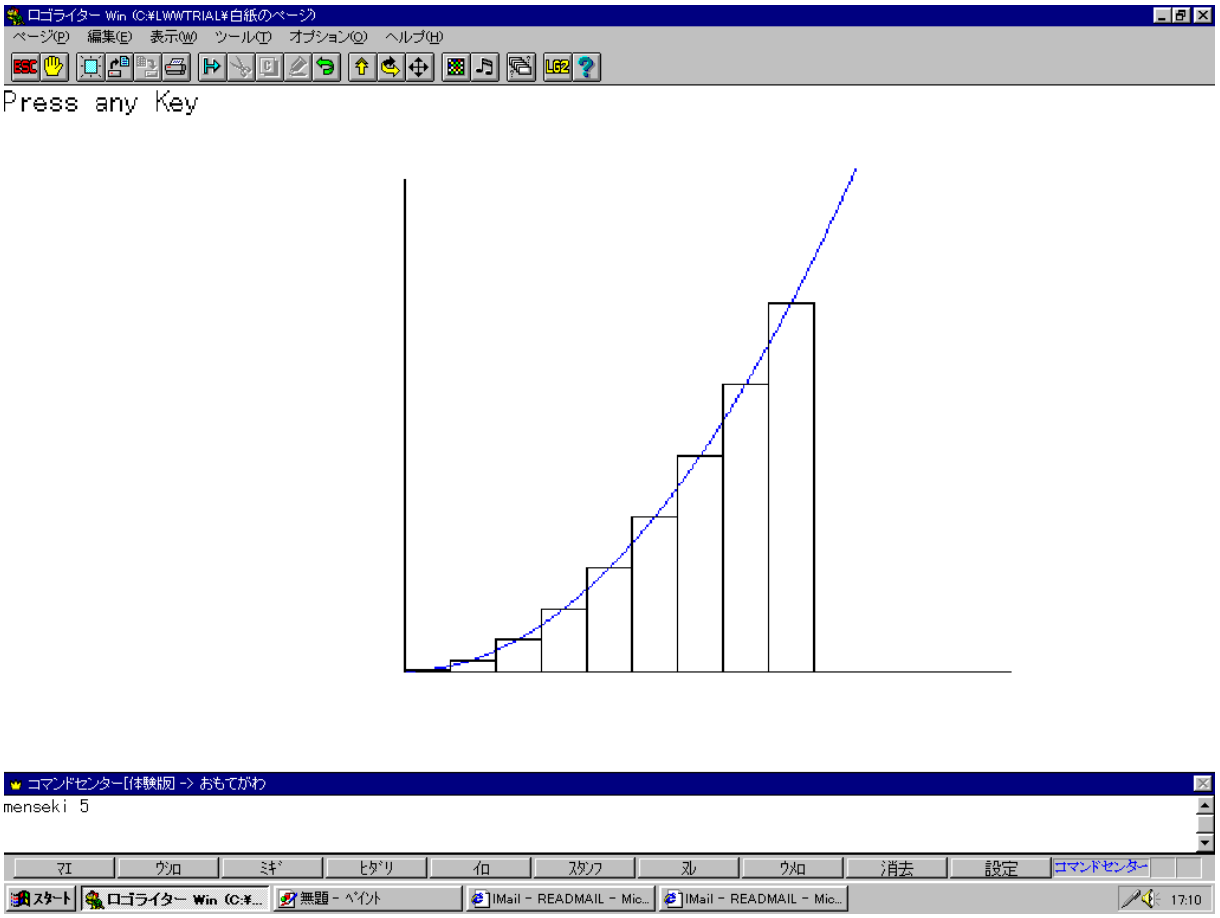


図５：長方形の底辺の中点での関数の値による方法

5.3.6　モンテカルロ法（:Mode = 6）

モンテカルロ法で求めます．つまり，0≦x≦1，0≦y≦1 の範囲から任意抽出した点が求める面積内かどうかを調べる方法です．つまり，全体の面積が 1 だから，求める面積はその確率になるはずです．

これは，試行が多ければ多いほど正確な面積に近づきます．この方法は生徒からは発表されないことを前提にしていますので，分かりやすいように，範囲でランダムにとってきた座標が100回中と1000回中で面積の部分に何回があるかを表わすことしました．

プログラムでは,「MonteCarlo Method」をもとにしてあります．ランダムに点を抽出するために,「Random sampling」を用いています．この点を (m; n) で表して，求める面積の内部なら赤で，面積の外部ならピンクで点を打つようにしました．そして，100 回中と 1000 回中の数を数えるために :r を用いました．ランダム抽出された点が面積の内部なら :r + 1 して数えています．

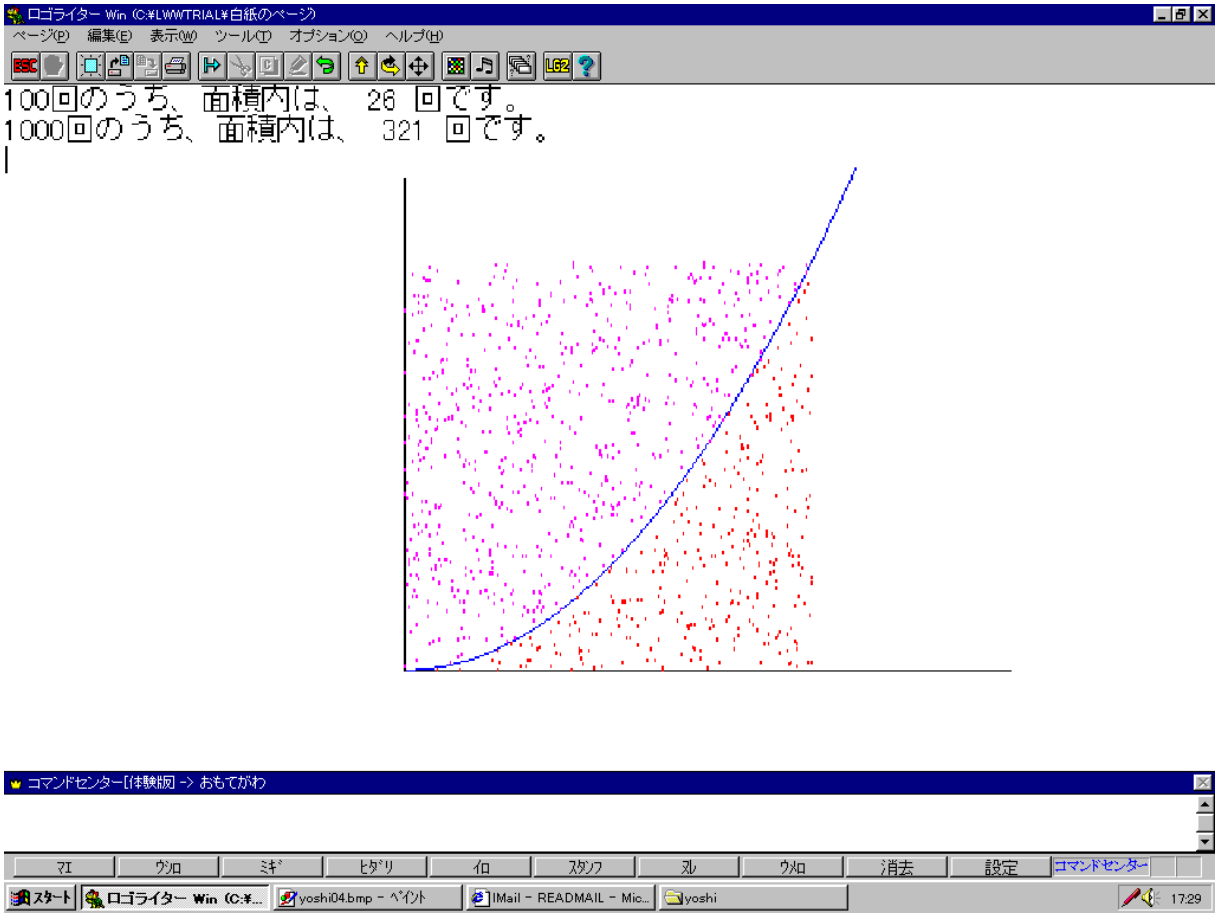


図6：モンテカルロ法

5.4　学習指導案

I　題材　面積を求める.

II　目標

1. **面積を区分することにより曲線で囲まれた面積を求めることができることを気づかせる.**
2. **上積分・下積分・区間の中間での値・台形則の中で効率のいい区分求積法を見つける.**

III　学習計画　(2時間)

1. **面積の求め方を考える.　　(1時間)**
2. **区分求積法で面積を実際に計算する.　　(1時間)**

IV　本時の学習

1. **指導過程(学習活動と指導上の留意点)(50分)**

初めに,「x 軸と x = 1 と $y = x^2$ で囲まれた面積について考えてみよう.」という問題を生徒に考えさせます. 生徒が考えやすいように x 軸と y 軸に目盛りを入れた座標に $y = x^2$ を描いたプリントを配ります. そして, 班にわけ, 線を加えて, 面積を求めるよう指示します. その間, 机間巡視をして解がでないようなら, x 軸と y 軸を平行に引かせ, マスにして考えさせたりします. 次に, 班ごとに黒板を使い発表させます. 生徒が考えると予想される解答としては,

1. **マスを数えて面積を出す.**
2. **上積分で面積を求める.**
3. **下積分で面積を求める.**
4. **区間の中間での値で面積を求める.**
5. **台形の面積を合わせて面積を求める.(台形則)**

1つ1つ全てのプログラムを見せると生徒は飽きてしまうので, 1番多かった解のプログラムを実際に見せて, 解に近づいていくことを確かめます. また, 発表されなかった解があれば, 別の方法としてそれを紹介します.

最後に生徒に気づいたところをまとめさせます. まとめのプリントを配り, 自分がどうやって考えたか, 別の生徒の考えについてまとめさせます. これは, 生徒自信のこれから微分積分を学ぶにあたっての変化を認識してもらいたいからです.

まとめプリントの内容

1. どういう考え方で考えましたか？
2. 他の考え方についてどう思いましたか？
3. 気づいたこと疑問に思ったことについてまとめてみましょう.
4. この面積は何になると思いますか？

2. 指導過程 (学習活動と指導上の留意点) (50 分)

前回の区分求積法で面積を実際に計算させるため４班に分けます.「x 軸と $x = 1$ と $y = x^2$ で囲まれた面積を前回の方法で計算してみよう.」と問題を与え，次の４つの方法について班ごとに振り分けて計算させます.

1. 上積分で面積を求める.

2. 下積分で面積を求める.

3. 区間の中間での値で面積を求める.

4. 台形の面積を合わせて面積を求める.（台形則）

そして，最終的に何に近づいていくのか予測を立てさせます．また，気づいたことを班ごとに話し合わせます．そして，計算の結果を元にして，班で話し合ったことを前で発表させます．その時，計算結果を表にしたり，班で立てた予想を発表させたりします．また，それが終わったら全体で４つの特徴や気づいたところを話し合わせます．１通り意見が出たら，別の方法としてモンテカルロ法をパソコンで見せます．そして，生徒は別の方法があることを知ります．確率の復習としても役立ったらいいと思います.

最後に，前回同様に気づいたことをまとめさせます．まとめのプリントを配り，話し合ったことや，前回と比較してわかったことをまとめます.

まとめプリントの内容

1. 計算結果を書きましょう.
   (a) 9等分のとき
   (b) 18等分のとき
   (c) 27等分のとき
2. あなたはこの計算結果を元に何に近づくと思いましたか？
3. 他の考えや前回思ったことと比較して気づいたことなどまとめてみましょう.

5.5 まとめ

コンピュータは，人間よりも早く計算を行い，プログラムを組めば正確に図形を描くこともできます．しかし，授業でコンピュータを使うとき，生徒が行う作業を取ってはいけません．つまり，計算問題を練習する授業で計算機を使っては練習にならないのと同じです．コンピュータがすべてをしてしまい生徒がただ見ているだけでは，生徒の頭に残りにくいと思い，生徒に考えさせたり，話し合わせたり，計算をさせたりすることを中心にしました．生徒が考えそうなマスを数える方法，上積分，下積分，区間の中間での値，台形則は図を描くだけにしました．そして，モンテカルロ法については，生徒から発表されるとは考えにくいことと，この時間は積分の授業ということもあって，$0 \leqq x \leqq 1$ ，$0 \leqq y \leqq 1$ の範囲でランダムにとってきた座標が100回中と1000回中で面積の部分に何回があるかを表わすことしました．またこれを用いて確率の復習もできればいいと思います.

面積を考えるにあたって，生徒がイメージしやすいようにこのプログラムを作りました．これを活用させることによって，生徒に次のようなことを授業で発見させたいです.

1. 区分求積法は区分を細かくすれば面積を求めることができること
2. 確率を用いても面積を求めることができること

そして，発見することで数学の面白さや感動を与えたいです．生徒にこのような感動を与えることが授業の本質だと思います．そして，みんなで話し合ったり，考えたりすることが大切だと思います.

（小林　佳子）

Chapter6

階差数列

高等学校の数学の授業に数列がありますが，そのなかで特に階差数列に関する計算をパソコンにさせるためのプログラムを作製しました．すなわち，生徒が計算という負担に気を取られて考える力を損なわないような，……“計算機”を実現しようと試みました．

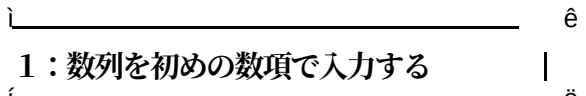
6.1　授業案

これをどのように授業の中で使えるかを，１つの例として以下に書きます．

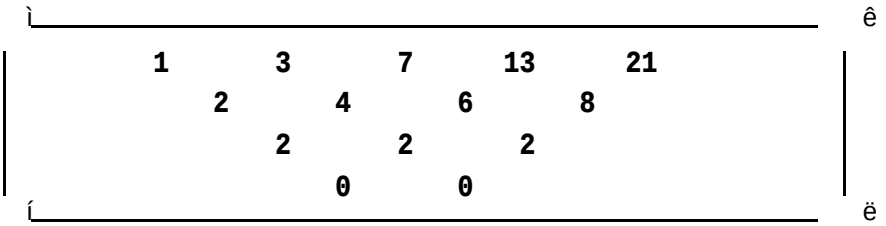
数列を習うときによく始めの方に出てくる問題で，例えば

1; 3; 7; 13; 21; ::: (1)

の一般項を求めよ．というような問題があります．これを生徒たちがどのように解くか，数列の規則性を見つけるのに隣り合った項の差を作ってみるとよい場合があります．そこで，このプログラムの



を使ってみると，



というように階差数列が出てきます．(1) の数列の階差数列

2; 4; 6; 8; ::: (2)

は初項 2 ，公差 2 の等差数列 2n だと，その次の階差数列

2, 2, 2, ……

から分かってきます．(1) と (2) の数列の関係をみると，(1) の数列は初項 1 と (2) の数列の和から求めることが出来ます．そこで，それを使って一般項 $\{a_n\}$ の式を書いてみると，

$$\begin{aligned} a_n &= 1 + \sum_{k=1}^{n-1} 2k \\ &= 1 + 2 \times \frac{(n-1)\{(n-1)+1\}}{2} \\ &= 1 + n(n-1) \\ &= n^2 - n + 1 \end{aligned}$$

となります．そして，その一般項が正しいかどうかをこのプログラムの

```
┌──────────────────┐
│ 2：一般項で入力する │
└──────────────────┘
```

を使ってみて，結果表示が同じならその一般項は正しいと判断できます．

6.2 プログラム

このプログラムは Logo 言語仕様です．

6.2.1 Logo の関数・変数一覧

関数名，変数名をそれぞれの意味を以下に書きます．

＜関数名 … 意味＞

((プログラム本体))
このプログラムの本体となる部分で，利用者に使いやすいよう３択にしています．

S … プログラムの本体（ Start ）に行くためのもの．
コマンドセンターで１文字で実行するため．
Start … プログラムの本体．
数列で計算されるか一般項で計算させるか終了するか選択するため．
Else … ２種類ある数列の計算方法を決めるもの．

((数列で計算))
利用者に数列を入力させます．

No1 … 数列で計算させるもの．

((一般項で計算))
利用者に一般項と項数を入力させます.

No2 … 一般項で計算させるもの.
N … 関数を変数にするためのもの.
一般項で計算させる所（ No2 ）で :n と入力しなければいけないところを n にするため.
Ippankou … 一般項の引数 Siki で初項から k 項までの数列を出すもの.

((階差の計算))
階差数列を出します.

Kaisa … 階差数列を作るもの.
Sa … 数列の前後の差を出すもの.

((空白を作る))
結果表示がまとまって見やすくするために，空白を作ります.

Keisan … Keisan_Sub に行くためのもの.
引数 i に最初だけ 0 を代入するため.
Keisan_Sub … 階差数列をおもてのページに表示させるもの.
Kuuhaku … 空白を作るもの.
Keta … １つ１つの項を８桁にそろえるもの.

＜変数名 … 意味＞

((プログラム本体))

a … 選択肢の中から自分が選んだデータを入力するもの.
clear … プログラムを一時停止させるためのもの.

((階差数列の計算))

L … 数列を入力するもの.
Siki … 一般項を入力するもの.
k … 一般項で項数を入力するもの.
n … 一般項での n と同じ意味を表すもの.
TL … 最終的にまとめた数列を入出力するもの.
i … 空白を何個分入れるかの上限を表すもの.
x … 空白を何個分入れるかの下限を表すもの.

6.2.2 Logo 言語のプログラム

以下に書いてあるのは，Logo 言語のプログラムです.

```
c [/* ----------------------プログラム本体---------------------- */]

to S
   Start
end

to Start
   hideturtle
   cleartext
   print "〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜
   print "＊＊＊こんにちは，ゆかニャンわーるどへようこそ！＊＊＊
   print "・あなたに代わって数列の階差を計算します.
   print "・それでは，下の数字から選んで下さい.
   print "＊＊＊入力に関しては半角英数字を入力して下さいね. ＊＊＊
   print "〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜
   print "
   print "１：数列を初めの数項で入力する
   print "２：一般項で入力する
   print "０：終了する
   print "
   make "a readlist
   ifelse "0 = first :a [
      print "
      print "ご利用ありがとうございました.
      print "また使って下さいね.
      stop
   ]
   [
      Else
   ]
   print "
   print "消しても良かったら，何かキーを押して下さい.
   make "clear readchar
   Start
end

to Else
   ifelse ( or "1 = first :a "2 = first :a ) [
      if "1 = first :a [
         No1
      ]
      if "2 = first :a [
         No2
      ]
   ]
   [
      print "
      print "ダメですよ.
      print "１か２か０にして下さいね.
   ]
end
```

```
c [/* ----------------------数列で計算---------------------- */]

to No1
   print "
   print "それでは，項と項の間を空白で区切って数列を入力して下さい．
   print [（例えば　2 9 28 65 126　）]
   make "L readlist
   print "
   Keisan :L
end


c [/* ---------------------一般項で計算--------------------- */]

to No2
   print "
   print "それでは，一般項を入力して下さい．
   print [（例えば　n ^ 3 + 1　）]
   make "Siki readlist
   print "何項目までに関して計算しますか？
   print [（例えば　5　）]
   make "k first readlist
   print "
   Ippankou :k :Siki
end

to N
   output :n
end

to Ippankou :k :Siki
   local [ L n ]
   make "L [ ]
   make "n  0
   repeat :k [
      make "n  :n + 1
      make "L  lastput run :Siki :L
   ]
   Keisan :L
end


c [/* ----------------------階差の計算---------------------- */]

to Kaisa :L
   if ( count :L ) = 2 [
      output firstput Sa :L [ ]
   ]
   output firstput Sa :L Kaisa butfirst :L
end

to Sa :L
   output ( first butfirst :L ) - first :L
end
```

```
c [/* ----------------------空白を作る---------------------- */]

to Keisan :L
   Keisan_Sub 0 :L
end

to Keisan_Sub :i :L
   local [ TL ]
   if ( count :L ) = 1 [
      stop
   ]
   make "TL Kuuhaku 4 * :i + 1 "| |
   Keta :L
   print :TL
   Keisan_Sub ( :i + 1 ) Kaisa :L
end

to Kuuhaku :i :x
   if ( count :x ) > :i [
      output "####
   ]
   if ( count :x ) < :i [
      output word "| |  Kuuhaku :i - 1 :x
   ]
   output :x
end

to Keta :L
   if empty? :L [
      stop
   ]
   make "TL ( word :TL Kuuhaku 8 first :L )
   Keta butfirst :L
end
```

6.3　取扱説明書

前もって言っておきます．このプログラムはロゴライター **Win Version 1.013** しか動作確認していません．このプログラムの動かし方を，簡単に手順を追って説明します．

手順１：上のプログラムの部分（ **to S** から **to Ippankou :k :Siki** の **end** まで ）を
　　マウスなどで選択して，マウスの右クリックでコピーして下さい．
　　（編集**(E)**でコピー**(C)**でもかまいません．）

手順２：ロゴライター**Win Version 1.013** を起動して下さい．（図１：はじめの画面）
　　うらがわのページをマウスでクリックして，貼り付けをして下さい．
　　（編集**(E)**で貼り付け**(P)** をするか，貼り付けボタンで）

手順３：おもてのページ（亀がいるページ）をクリックしたら，
　　コマンドセンターのページをクリックし，
　　半角で s と入力してリターンキーを押して下さい.（ｓ　†　）
手順４：プログラムが作動しているので，おもてのページの指示に従って下さい.
　　（図２：プログラム開始画面）

図１：　はじめの画面

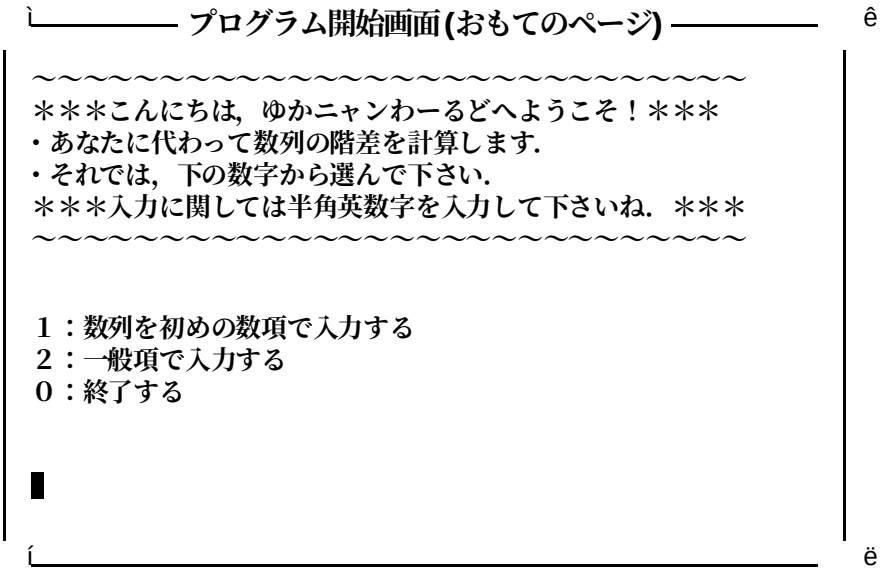


図２：　プログラム開始画面

6.3.1　プログラム開始からの説明

「１：数列を初めの数項で入力する」のと「２：一般項で入力する」動かし方をそれぞれ詳しく以下に説明します[1]．まず，上の手順１～４をしておきます．この後はキーボードのみ使います．（プログラム稼働中は全ておもてのページで処理します.）

＜１：数列を初めの数項で入力する＞
数字の１を入力します.
それから，自分が計算したいと思っている数列を入力します.

例）2 †9 †28 †65 †126　†　（以下，†は空白を意味します.）

そうすると，計算結果が出てきます.（図３：結果表示１）

[1]「１：数列を初めの数項で入力する」方は一応全角でも動きますが,「２：一般項で入力する」方は全角の n だとエラーが起こるので，必ず最初から半角になっていることをお勧めします.

ì———— 結果表示１ (おもてのページ) ———— ê

```
〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜
＊＊＊こんにちは，ゆかニャンわーるどへようこそ！＊＊＊
・あなたに代わって数列の階差を計算します．
・それでは，下の数字から選んで下さい．
＊＊＊入力に関しては半角英数字を入力して下さいね．＊＊＊
〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜


１：数列を初めの数項で入力する
２：一般項で入力する
０：終了する


1


それでは，項と項の間を空白で区切って数列を入力して下さい．
（例えば 2 9 28 65 126 ）
2 9 28 65 126


          2         9        28        65       126
               7        19        37        61
                   12        18        24
                         6         6


消しても良かったら，何かキーを押して下さい．
```

í———————————————— ë

図３：　結果表示１

＜２：一般項で入力する＞

数字の２を入力します．

それから，自分か計算したいと思っている一般項を入力します[2]．

例）nþ^þ3þ+þ1　þ

そして，項数を入力します．

例）5　þ

[2] マイナス（ Ä ）が１番最初に付く場合，そのまま入力するとエラーになるので，0 Ä n Ä 1 というように最初に 0 を入れて下さい．

そうすると，計算結果が出てきます.（図４：結果表示２）

結果表示２（おもてのページ）

```
〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜
＊＊＊こんにちは，ゆかニャンわーるどへようこそ！＊＊＊
・あなたに代わって数列の階差を計算します.
・それでは，下の数字から選んで下さい.
＊＊＊入力に関しては半角英数字を入力して下さいね．＊＊＊
〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜


１：数列を初めの数項で入力する
２：一般項で入力する
０：終了する


2


それでは，一般項を入力して下さい.
（例えば n ^ 3 + 1 ）
n ^ 3 + 1
何項目までに関して計算しますか？
（例えば 5 ）
5


          2         9        28        65       126
               7        19        37        61
                   12        18        24
                          6         6


消しても良かったら，何かキーを押して下さい.
```

図４： 結果表示２

6.3.2 プログラム終了の説明

計算結果が出てきて「消しても良かったら，何かキーを押して下さい.」と出てきたら何でも良いのでキーを押すと，計算結果は消えて，プログラム開始画面に戻ります.
（図２：プログラム開始画面）
そして，この画面の時に０を入力すると，このプログラムは終了します.

6.4　評価的まとめ

この計算機を使ってみると，最初に述べた“手間のかかる計算”をすばやく結果を出し，そして見やすくなっています．そして，この計算機によって，幾分か数列に対して興味をもたれるだろうと思います．しかし，やはりそこは電卓と同じで，階差を教えるのに初めからこの計算機を使ってしまうと，何をやっているのか，どれが階差なのかが分からなくなる危険性もあります．そうならないためにも，一通り基礎を知ってから，応用としてこの計算機を使ったほうが効率がよいかと思います．
階差数列は数列の一般項，次の項を求める１つの手段として用いられています．たとえ答えを求められなくても，その数列を考えていく上で，面白い規則性，例えば，

è 等比数列 2^n は階差をとっても等比数列．

è n 次式の階差数列は n Ä 1 次式．

è n 次式を n 回階差数列は定値数列になる．

等を発見したりします．
以下に，面白い規則性の１つとして一般項 $n^3 + 2^{n+1}$ を使ってみて述べたいと思います．
まず，この一般項の数列は，

5, 16, 43, 96, 189, 344, 599, 1024, 1753, ……

となります．これだけを見てもどのような数列か分からないと思います．そこで，この計算機を使って階差をどんどんと取っていくと…，

```
 5     16     43     96    189    344    599   1024   1753
    11     27     53     93    155    255    425    729
       16     26     40     62    100    170    304
          10     14     22     38     70    134
              4      8     16     32     64
                 4      8     16     32
                    4      8     16
                       4      8
```

となります．これを見ると，第４階差数列以降が，

4, 8, 16, 32, 64, ……

となり，これは初項が４，公比２の等比数列になることが分かります．これは，n^3 の部分が階差をとることによって次数が落ちて行き，４回階差を取った時点でとうとう０になってしまったことによるものです．だから，2^{n+1} の部分だけが残ったのです．

このように，第 k 回の階差数列が等比数列となる場合は，kÄ1 次式と階差数列の和の形であることがわかるのです.

最後に，このように階差数列１つ取ってみても色々と面白いことが分かってくると思います．前々から，数学離れする人が増えてきていると思います．その１つの理由として,「つまらない」というのがあると思いますが，この計算機が数学に興味を持つための一助となれば良いかと思います.

（今井　由果）

Chapter7

行列の計算

行列に関するプログラムを作りたいと思いました．行列のプログラムを作る上でのコンセプトは行列のプログラムを使って，生徒の考える力を養うのを目的とし，そこを目標にプログラムを考えていこうと思います．

まず,「行列」について簡単にまとめると,

> 一般に，数を長方形の形に並べたものを行列といい，その並べた数のことを行列の成分，または要素という．そして行列の横の並びを行，縦の並びを列といい，行列はその行の数と列の数によって，m行n列の行列，mÇn型の行列，又はmÇn行列という．また，行の数と列の数が等しい行列を正方行列といい，その行（列）の数を，正方行列の次数という．

ということになります．
つぎにこのプログラムについて述べていこうと思います．

7.1　プログラミングの経緯

まず行列をどのように表示するのか？について考えてみましょう．ただデータを表示すればいいというわけではなく行列の形にして表示する必要があります．それには見やすさやわかりやすさをふまえた上で行列を1行ずつ表示していく方法をとります．さらに表示できる要素の桁を4桁に揃えることにします．この要素の桁を揃えるのにはどういったことがいえるのかというと，要素の桁を揃えることにより行列の値を見やすくわかりやすいものにするものになるのです．

```
to PrintMat :Matrix
   if empty? :Matrix [ stop ]
   PrintFirst :Matrix
   PrintMat butfirst :Matrix
end
   to PrintFirst :M
```

```
      print   Line first :M
   end
   to Line  :I
      if empty? :I [ output "|   )|  ]
      output ( word  String$  4 first :I char 32 Line butfirst :I )
   end
   to String$ :k :x
      if ( count :x ) > :k [ output SharpRep$ :k ]
      if ( count :x ) < :k [ output word char 32 String$ :k - 1 :x ]
      output :x
   end
   to SharpRep$ :k
      if :k = 1 [ output "# ]
      output ( word "# SharpRep$ :k - 1 )
   end
```

7.1.1 行列の加減法

行列の基本的な計算について考えていきましょう．まず最初に行列の加法について考えます．行列の1行を1つのベクトルと考え，2つのベクトルの和を計算する．このことを使って行列の加法を説明すると2つの行列M; N のそれぞれの第1行を抜き出しその和を計算します．つぎに第2行，つぎに第3行と計算していくわけです．減法は加法と同様のことをするわけですから省略します．

```
to V_Sum :m :n
   if ( count :n ) = 0 [ output :m ]
   output fput  ( first :m ) + ( first :n ) V_Sum butfirst :m butfirst :n
end
to plusmat :m :n
   if ( count :n ) = 0  [ output [ ] ]
    output fput  (V_Sum first :m first :n )   plusmat butfirst :m butfirst :n
end
```

7.1.2 転置行列

乗法について考えていくわけですが，ひとまず転置行列について考えていくことにします．ここでなぜ，転置行列について考えていくかは7.1.3で述べます．では，転置行列について申し上げるとそもそも，転置行列とは行列の行と列を入れ替えたもので，列ベクトルの転置行列は行ベクトルになり，行ベクトルの転置行列は列ベクトルとなります．このことからプログラムでは行列を第1列と第2列以降とにわけて，その第1列を第1行とおき，同様に第2列，第3列を第2行，第3行とおいていくわけです．

```
to T_Mat :Matrix
    if empty? first :Matrix [ output [] ]
    output fput FirstCol :Matrix  T_Mat ButfirstCol :Matrix
end
to FirstCol :Matrix
   if empty? :Matrix [ output [] ]
   output fput ( first first :Matrix )  ( FirstCol butfirst :Matrix )
end
```

```
to ButFirstCol :Matrix
   if empty? :Matrix [ output [] ]
   output fput butfirst first :Matrix ButFirstCol butfirst :Matrix
end
```

7.1.3 行列の乗法

2つの行列M, Nの積をLとするとLの(i, j)成分はMの第i行ベクトルとNの第j列ベクトルとの内積になります．ここで前節の転置行列を用いるとMの第i行ベクトルとNの転置行列の第j行との内積といいかえることができます．よって内積を求める関数Inner_Prodを作れば行列の積を求める関数を作ることができます．

```
to Inner_Prod :m :n
   if ( count :n ) = 0 [ output 0 ]
   output ( first :m ) * ( first :n ) + Inner_Prod butfirst :m
butfirst :n
end
to Times_Mat :m :n
   output TM_Sub :m T_Mat :n
end
  to TM_Sub :m :n
     if empty? :n [ output [ ] ]
     if empty? :m [ output [ ] ]
     if Vector?  :m  [ output fput Inner_Prod :m first :n
                                      TM_Sub :m butfirst :n ]
     output fput TM_Sub first :m :n
                 TM_Sub butfirst :m :n
  end
  to Vector? :x
     output word? first :x
  end
```

7.1.4 プログラムの開発価値

さて，ここまでで，だいたいの行列の基本的な計算はできるようになりました．この時点でLogo言語をつかった教育現場における応用としてでいうと，このプログラムは行列を短い時間で計算できる「行列計算機」の形で利用できます．

7.2 行列プログラム

以上の考えによって，できたプログラム全体を記します．

```
to start
   cleartext
   print "-------------------------------------------------------
   print "||
   print "|                        行列計算機|
   print "||
   print "-------------------------------------------------------
   print "||
```

```
   print "|   1.）M＋N（加法）|
   print "|   2.）M－N（減法）|
   print "|   3.）M＊N（乗法）|
   print "|   4.）M＾N|
   print "|   0.）終了|
   print "||
   print "|     （注意）行列を入力する際|
   print "|      （例）( 2 1 ) は  [2 1][3 4]と入力してください. |
   print "|           ( 3 4 )  |
   output Nxt
end

to Nxt
   print "||
   print "選択肢の番号を選んでください.
   output readnum
end

to readnum
   make "rn first readlist
   if :rn = 0 [stop]
   if :rn = 1 [
       print "Mの行列を入力してください.
       make "m readlist
       print "Nの行列を入力してください.
       make "n readlist
       print "||
       print "答えは
       printmat plusmat :m :n
       output waitkey
              ]
   if :rn = 2 [
       print "Mの行列を入力してください.
       make "m readlist
       print "Nの行列を入力してください.
       make "n readlist
       print "||
       print "答えは
       printmat maimat :m :n
       output waitkey
              ]
   if :rn = 3 [
       print "Mの行列を入力してください.
       make "m readlist
       print "Nの行列を入力してください.
       make "n readlist
       print "||
       print "答えは
       printmat times_mat :m :n
       output waitkey
              ]
   if :rn = 4 [
       print "Mの行列を入力してください.
       make "m readlist
       print "何乗するのか入力してください.
```

```
        make "n first readlist
        print "||
        print "答えは
        printmat power_mat :m :n
        output waitkey
               ]
   if ( or ( :rn < 0 ) ( :rn > 4 ) ) [
   print ".
   print "あれれ？
   print "１．２．３．４．０のうちのいずれかで答えてくださいね.
   output readnum ]
end

to waitkey
   local "dummy
   print "なにかキーを押してください.
   make "dummy readchar
   output start
end

to c :Dummy
end

to PrintMat :Matrix
   if empty? :Matrix [ stop ]
   PrintFirst :Matrix
   PrintMat butfirst :Matrix
end
   to PrintFirst :M
      print   Line first :M
   end
   to Line  :I
      if empty? :I [ output "|   )|  ]
      output ( word  String$  4 first :I char 32 Line butfirst :I )
   end
   to String$ :k :x
      if ( count :x ) > :k [ output SharpRep$ :k ]
      if ( count :x ) < :k [ output word char 32 String$ :k - 1 :x ]
      output :x
   end
   to SharpRep$ :k
      if :k = 1 [ output "# ]
      output ( word "# SharpRep$ :k - 1 )
   end

to V_Sum :m :n
   if ( count :n ) = 0 [ output :m ]
   output fput  ( first :m ) + ( first :n ) V_Sum butfirst :m butfirst :n
end

to V_mai :m :n
   if ( count :n ) = 0 [ output :m ]
   output fput  ( first :m ) - ( first :n ) V_mai butfirst :m butfirst :n
end
```

```
to Inner_Prod :m :n
   if ( count :n ) = 0 [ output 0 ]
   output ( first :m ) * ( first :n ) + Inner_Prod butfirst :m
butfirst :n
end

to FirstCol :Matrix
   if empty? :Matrix [ output [] ]
   output fput ( first first :Matrix )  ( FirstCol butfirst :Matrix )
end

to ButFirstCol :Matrix
   if empty? :Matrix [ output [] ]
   output fput butfirst first :Matrix ButFirstCol butfirst :Matrix
end

to plusmat :m :n
   if ( count :n ) = 0  [ output [ ] ]
    output fput  (V_Sum first :m first :n )   plusmat butfirst :m butfirst :n
end

to maimat :m :n
   if ( count :n ) = 0  [ output [ ] ]
    output fput  (V_mai first :m first :n )   maimat butfirst :m butfirst :n
end

to T_Mat :Matrix
    if empty? first :Matrix [ output [] ]
    output fput FirstCol :Matrix  T_Mat ButfirstCol :Matrix
end

to Times_Mat :m :n
   output TM_Sub :m T_Mat :n
end
  to TM_Sub :m :n
     if empty? :n [ output [ ] ]
     if empty? :m [ output [ ] ]
     if Vector?  :m  [ output fput Inner_Prod :m first :n
                                      TM_Sub :m butfirst :n ]
     output fput TM_Sub first :m :n
                 TM_Sub butfirst :m :n
  end
  to Vector? :x
     output word? first :x
  end
to T0
   cleartext cc
   make "a [ [ 2 1 ]
             [ 1 2 ] ]
   make "b [ [ 2 0 ]
             [ 0 2 ] ]
   PrintMat Times_Mat :a :b
end
to Power_Mat :M :k
   if :k = 1 [ output :M ]
```

```
   output Times_Mat :M Power_Mat :M :k - 1
end
to T2
   cleartext cc
   make "a [[ 2 1  ][ 1 2 ]]
   make "k 0
   repeat 7 [
     make "k :k + 1
     PrintMat Power_Mat :a :k
     print ".. 
   ]
   print [** That's all! ** ]
end

to T3
   cleartext cc
   make "a [[ 2 1 1 ][ 1 2 1 ][ 1 1 2]]
   make "k 0
   repeat 10 [
     make "k :k + 1
     PrintMat Power_Mat :a :k
     print ".. 
   ]
   print [** That's all! ** ]
end

to T4
   cleartext cc
   make "a [[ 2 1 1 1 ][ 1 2 1 1 ][ 1 1 2 1][ 1 1 1 2]]
   make "k 0
   repeat 7 [
     make "k :k + 1
     PrintMat Power_Mat :a :k
     print ".. 
   ]
   print [** That's all! ** ]
end

to T1
   cleartext cc
   make "a [[1 1 1 1 ][ 1 1 1 1 ][ 1 1 1 1][ 1 1 1 1]]
   make "k 0
   repeat 7 [
     make "k :k + 1
     PrintMat Power_Mat :a :k
     print ".. 
   ]
   print [** That's all! ** ]
end
```

7.2.1　プログラムの関数の説明

PrintMat :Matrix	行列 :Matrix をページに記す
V _ Sum :m :n	2つのリスト : m;: n の和を出力する
V _ mai :m :n	2つのリスト : m;: n の差を出力する
Inner _ Prod :m :n	2つのリスト : m;: n の内積を出力する
FirstCol :Matrix	リストの最初の数を出力する
ButFirstCol :Matrix	リストの2番目以降の数を出力する
plusmat :m :n	行列 : m;: n の和を出力する
maimat :m :n	行列 : m;: n の差を出力する
T _ Mat :Matrix	行列 :Matrix の転置した行列を出力する
Times _ Mat :m :n	行列 : m;: n の積を出力する
Power _ Mat :M :k	行列 :Mの : k 乗を出力する
T1	行列 $\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$ の1乗から7乗までを表示するプログラムで, このプログラムを用いることにより行列の要素を入力しないですみます. さらに作りつけの行列での結果を表示することができます.
T2	行列 $\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$ の1乗から7乗までの実行結果を表示する
T3	行列 $\begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix}$ の1乗から10乗までの実行結果を表示する
T4	行列 $\begin{pmatrix} 2 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 \\ 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 2 \end{pmatrix}$ の1乗から7乗までの実行結果を表示する

7.2.2　プログラム実行方法

　このプログラムはロゴライターfor Winに準拠した算譜です.
まず，ロゴライターfor Winを起動します．つぎに，うらがわのページにプログラムを貼り付けます．そして，コマンドセンターのページでstartと入力すると次のような画面が出ます.

```
----------------------------------------------------

                    行列計算機

----------------------------------------------------

                1.）M＋N（加法）

                2.）M－N（減法）

                3.）M＊N（乗法）

                4.）M＾N

                0.）終了


（注意）行列を入力する際
（例）( 2 1 ) は  [2 1][3 4] と入力してください.
      ( 3 4 )

選択肢の番号を選んでください.
```

次に，いずれかの番号を選びます.
例として１を選んだとして，次にデータを入力します.

Mの行列を入力してください．と表示されるので
例）[1 2][3 4] †
Nの行列を入力してください.
例）[5 6][7 8] †
このようにデータを入力すると
答えが
　　６　　８　　）
　１０　１２　　）
と出力されます.
そして
なにかキーを押してください.

と出力されなにかキーを押すとスタート画面に戻ります.

つぎに考察する行列の実行方法の説明をします. まず, プログラムは T1〜T4 まであり T2 を実行例として, つぎのように入力します. コマンドセンターのページに半角で T2 と入力するとつぎのようなものが表示されます.

(例)：T2 の出力

```
   2    1    )
   1    2    )
..
   5    4    )
   4    5    )
..
  14   13    )
  13   14    )
..
  41   40    )
  40   41    )
..
 122  121    )
 121  122    )
..
 365  364    )
 364  365    )
..
1094 1093    )
1093 1094    )
..
** That's all! **
```

というのが出力されます

7.3 プログラムを使った授業例

橋本重治 [1] は,

知識 意味が分かっている上にそれが記憶されており, 再生・再認されることのできる観念内容.

理解 事態の内部関係の把握が成立している状態.

思考 新しい問題場面において, 事態を分析したり, 問題の核心をつかみ, 既有の知識・原理を関連づけてこれを解決したり, 解釈したりする.

創造 いわゆる拡散的思考を働かせ, 既存の要素や部分を総合して新しいものを創り出す. 思考と共通する面も多いが, それよりも広い概念で, 技能も態度も含む.

技能 一定の目的を遂行するのに効果的に適合するように分化し統合された精神的, 身体的な行動の様式や方法.

態度 見方・考え方・行動の傾向性であって, 人の行動に対して指示力をもつ一種の精神的身体的準備の状態.

[1] 橋本　重治　『新・教育評価法総説（上）』金子書房, 1976

を含む11種類に学力を分類しています[2]．ここで作ったプログラムは技能ではなく創造にあたります.

つぎに，T1～T4を使った授業として，プログラムの実行結果を表示し，その計算結果から創造を働かせ，生徒にどういったことが言えるかについて，考えさせます．そして，その気づいたことについて意見を発表してもらいます．このときT2を例すれば，意見は次のようなものが予想されます.

1. 斜め同士の要素が同じである
2. 上下の要素の差が1である
3. 行，または列を足したものが3の累乗になっている
4. $M_{1,1}$に着目した時，その要素の階差が3; 9; 27となり$3^1; 3^2; 3^3$の形で増加している
5. 行列Mの要素$M_{11} + M_{12}$が$2 Ç 2 + 1 = 5, 5 Ç 2 + 4 = 14$というように$M_{11} + M_{12}$が$n + 1$乗したもののM$_{11}$に等しい

そして，意見を発表し合うことよって他の人の意見からさらにまた違った観点から見た法則性を気づかせます.

7.4　まとめと今後の課題

この行列プログラムは画面への表示から始まり，基本的な行列の計算を経て，そして，その基本的な行列の計算をさらに応用して授業で教材として用いることのできるものになりました.
今後の課題として

- è $M^2 + 3M Ç N + N^2$ のように式を入力すれば答えが出力されるプログラムを作ること.
- è 現在は小数の形にしないかぎりエラーになるのでこれを有理数を分数の形のまま計算，出力をできるようにすること.

というようなことが今後の課題として残されます.　　（坂倉　彰仁）

[2] 清水　利信　『学力構造の心理学』金子書房　1976，pp．7-8

Chapter8

ポケットコウノトリの試作

8.1　なぜ，遺伝の授業にパソコンが必要になるか

遺伝という物は一見難しそうで，とっつきにきそうである．例えばエンドウ豆の実験で，丸い種子同士を交配したのに出来上がった種子には丸い種子としわのある種子が混じっているという場合がある．最初はなぜ親のもつ形質とは違う形質を持つ子供が産まれるのか，不思議に思うかも知れない．しかし生物には優性遺伝子と劣性遺伝子という物があり，見た目には表れていないけれど，実は隠し持っている形質と言うのがあり得る．そこで，産まれた子供がどのような形質を表しているかによって，それを調べようと言うのである．

そこでなぜパソコンを使うと良いかという問題が出てくる．まず，実際に生物なり植物なりを使って実験しようとするとどうしても時間がかかる．（半年なり１年なり．）そして手間もかかるし，自分が作りたいだけ子供を作れるというわけではない．１対のつがいから100匹，1000匹の子供を作るなどというのは，ほ乳類ではまず無理な話である．

しかしパソコンならそんな問題も万事解決である．自分が好きな組み合わせで，好きなだけ短い時間で（これ重要）子供を表示させることができる．これがあれば授業中の間に遺伝の仮想実験なるものをすることが出来るのである．

このプログラムを使うことによって，遺伝の仕組みについて考えるきっかけとなれば良いと思う．

8.2　このプログラムの取り扱いについて

これは Logo Writer for Win に準拠した算譜である．以下に，取り扱い説明書をかく．

Logoライターを起動する．PKTと言う名前のファイルを開く．コマンドセンターにPKT 0 1 †[1] と打ち込むとプログラムが作動する．後は画面に説明が出てくるのでそれにしたがって動作する．

[1] この引き数0,1は「左側の系図のパターン，右側の系図のパターン」である．バリエーションは，Pat　で定義されている 0,1 の２種しかない．

8.3　動作例

＊＊プログラムの流れ

起動すると左右に亀の系図が表示される．両親と，子供が４匹で順に１～６の番号付けがされている．左右の系図の間には家マークが表示される（図１）．

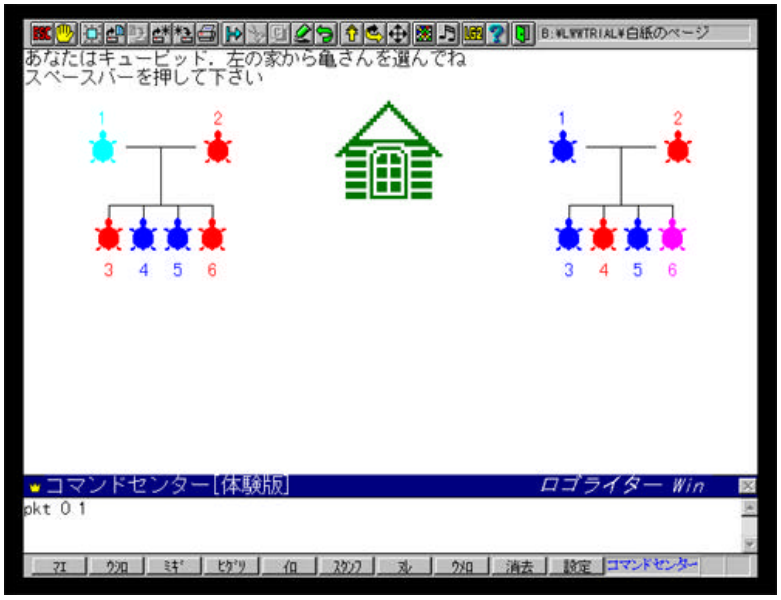


図１：系図の表示

左の系図のうちどれか一つの番号を選ぶと，選んだ亀が真ん中の家マークへゆっくりと移動する（図２）

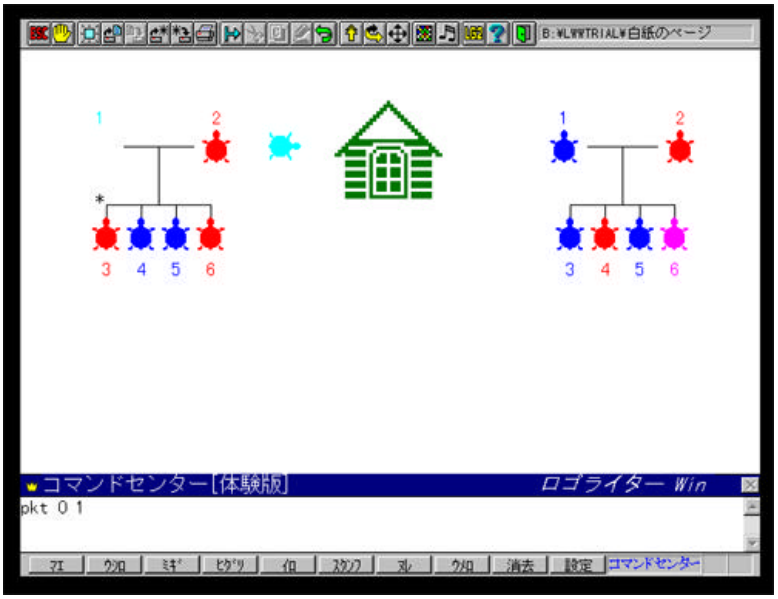


図２：左の亀の移動

右の系図のうちどれか一つの番号を選ぶと，選んだ亀が真ん中の家マークへゆっくりと移動する．（図３）

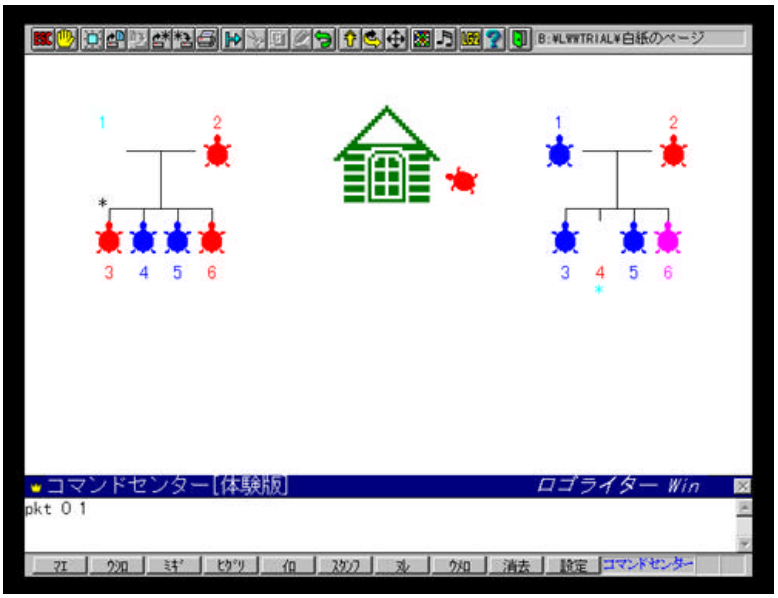


図３：右の亀の移動

自分が作りたいだけの子供の数を入力すると，真ん中の家マークから子供が誕生する．子供の数が４匹までならゆっくり，それ以上の場合は早く結果が表示されるようになっている．産まれた子供は左上にカウントされる仕組みになっている．(図４)

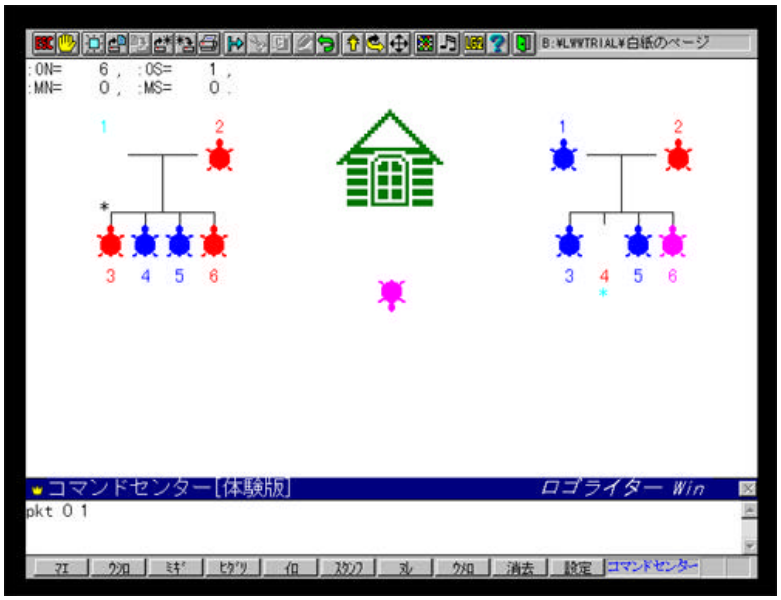


図４：結果の表示

左上に表示されているON、OS、MN、MSというのはONがオスの優性の形質が表れている亀（オス並），OSがオスの劣性の形質が表れている亀（オス薄），MNがメスの優性の形質が表れている亀（メス並），MSがメスの劣性の形質が表れている亀（メス薄）と言うことである．（　）の中の呼び名は，便宜的に付けた名前である．

8.4　実際のプログラム

```
to c :pkt
end
to Comment
   c [/*                          LogoWriter for Windows 準拠版    */]
end
to PKT :na :nb
   c [/* Pocket Kouno-Tori */ ]
   c [/* 高校の生物　遺伝の授業のために  */ ]
   c [/* ]
   cg
   cleartext fastturtle
   print [ある所にオスは青、メスは赤という分かりやすい色をした亀がいた。]
   WaitKey
   print [オスの亀です！よろしく。]
   setc 1 penup setpos  [ -100 0 ]
   slowturtle right 90
   penup forward 100 right 270
   st
   WaitKey fastturtle
   print [メスの亀です！よろしくね。]
   setc 2 penup setpos  [ 100 0 ]
   slowturtle left 90
   penup forward 100 left 270
   st
   WaitKey
   print [この亀同士の子どもになぜか薄い色の亀が生まれることがあったそうな。]
   WaitKey  fastturtle
   print [薄い色のオス亀です！よろしく。]
   setc 5 penup setpos  [ -100 0 ]
   slowturtle right 90
   penup forward 100 right 270
   st
   WaitKey fastturtle
   print [薄い色のメス亀です！よろしくね。]
   setc 3 penup setpos  [ 100 0 ]
   slowturtle left 90
   penup forward 100 left 270
   st
   WaitKey
   print [この事を念頭に置いておいてプログラムを使ってね！]
   WaitKey
   cg
   cleartext fastturtle
   print [遺伝って知ってる？子どもは親からの情報を受け継ぐんだね]
   WaitKey
   make "a Pat :na
   make "b Pat :nb
   penup
   forward 100
   setsh   9    setc 12 pendown
   bigstamp [3 3]      penup
   setsh   0
```

```
   left 90 forward 200 right 90
   DrawKeizu :a
   setpos [ -200 100 ]
   right 90 forward 400 left 90
   DrawKeizu :b
   c [/*左から選ぶ*/]
   print [あなたはキューピッド．左の家から亀さんを選んでね]
   WaitKey
   print [１から６の番号で選んで下さい]
   make "CodeA first readlist
   setpos DeCode 1 :CodeA
   st
   back 45 label "* forward 45
   setc  7
   pendown stamp penup
   WaitKey
   seth towards [ 0 100 ]
   slowturtle
   setc   TColor item :CodeA :a
   setpos [ 0 100 ]
   fastturtle
   WaitKey
   ht   seth 0
   c [/*右から選ぶ*/]
   cleartext
   print [こんどは，右の家から亀さんを選んでね]
   WaitKey
   print [１から６の番号で選んで下さい]
   make "CodeB first readlist
   setpos DeCode 2 :CodeB
   st
   back 45 label "* forward 45
   setc  7
   pendown stamp penup
   WaitKey
   seth towards [ 0 100 ]
   slowturtle
   setc   TColor item :CodeB :b
   setpos [ 0 100 ]
   fastturtle
   WaitKey
   ht
   c [/*子作り*/]
   print [子どもを産ませる人数を指定して，リターンキーを押して下さい ]
   make "n first readlist
   Kodukuri :n 0 0 0 0
end

to WaitKey
   print [スペースバーを押して下さい]
   make "Dummy readchar
   cleartext
end

to Kodukuri :n :ON :OS :MN :MS
```

```
  c [/* ON:オス並 OS:オス薄 MN:メス並... の孫の数 */ ]
  local [wa wb w nn Non Nos Nmn Nms ]
  make "wa item :CodeA :a
  make "wb item :CodeB :b
  make "Non :ON
  make "Nos :OS
  make "Nmn :MN
  make "Nms :MS
  c [/* これは伴性遺伝ではない優性・劣性遺伝の算譜です*/]
  if GenderCheck? [ print "これってレズホモカップル？ stop ]
  repeat :n [
    make "w  ( word "q Gensuu :wa  Gensuu :wb Gender )
     penup setpos [ 0 100 ] seth 180
     setc TColor :w           st
      if color =  1 [ make "Non :Non + 1  ]
       c [/* 1 の代わりにオス並の色番号を入れて下さい*/]
      if color = 5 [ make "Nos :Nos + 1  ]
      c [/*5 の代わりにオス薄の色番号を入れて下さい*/]
      if color =  2 [ make "Nmn :Nmn + 1  ]
       c [/* 2 の代わりにメス並の色番号を入れて下さい*/]
      if color =  3 [ make "Nms :Nms + 1  ]
       c [/* 3 の代わりにメス薄の色番号を入れて下さい*/]
      if not ( or ( color = 1 ) ( color = 2 ) ( color = 3 ) ( color = 5 ))
         [  print
         [ ねえ，ちょっと．ここに来るはずないない．色番号間違えてない？]
         ]
     fastturtle
     if :n < 5 [ slowturtle ]
     forward 150     ht   cleartext
     print ( list ":ON= String$ 4 :Non ", ":OS= String$ 4 :Nos ", )
     print ( list ":MN= String$ 4 :Nmn ", ":MS= String$ 4 :Nms ". )
  ]
  repeat 15 [ print ". ]
  print [また子どもを産ませる人数を指定して，リターンキーを押して下さい]
  print [終わりの時は 0 を入力しましょう．]
  make "nn first readlist
  if :nn = 0 [ cg st cleartext stop ]
  Kodukuri :nn :Non :Nos :Nmn :Nms
end

to DrawKeizu :x
   st
   setc 0
   left 90 pendown forward 30 penup forward 20
   right 90 setc TColor   item 1 :x pendown stamp
   penup forward 25 pendown label 1  penup back 25
   setc 0
   right 90 forward 20 pendown forward 60
   penup forward 20
   left 90  setc TColor item 2 :x pendown stamp
   penup forward 25 pendown label 2 penup back 25
   setc 0
   left 90  forward 20 pendown forward 30
   left 90  forward 50 right 90 forward 45
   right 90
```

```
   make "nLabel 3
   repeat 3 [
     back 10 penup back 15
     setc TColor item :nLabel :x pendown stamp
     penup   back 30 pendown label :nLabel penup
     setc 0
     forward 55 pendown
     right 90 forward 30 left 90
     make "nLabel :nLabel + 1
   ]
     back 10 penup back 15
     setc TColor item :nLabel :x pendown stamp
     penup   back 30 pendown label :nLabel penup
     setc 0
     forward 55 pendown
     penup
     ht
end

to TColor "w
  c [/* 0:優性な遺伝子，1:劣性な遺伝子，2:性別遺伝子 */]
  if :w = "q00 [ output 2 ]
  if :w = "q01 [ output 2 ]
  if :w = "q10 [ output 2 ]
  if :w = "q11  [ output  3 ]
  if :w = "q002 [ output  1 ]
  if :w = "q012 [ output  1 ]
  if :w = "q102 [ output  1 ]
  if :w = "q112 [ output 5 ]
  output 1
end

to Pat :n
  if :n = 1 [ output [ q012 q01 q002 q01 q102 q11] ]
  output [ q112 q00 q10 q102 q102 q10 ]
end

to DeCode :m :k
  local "p
  make "p ( list :m :k )
  if :p = [ 1 1 ] [ output [-250 100] ]
  if :p = [ 1 2 ] [ output [-150 100] ]
  if :p = [ 1 3 ] [ output [-245  25] ]
  if :p = [ 1 4 ] [ output [-215  25] ]
  if :p = [ 1 5 ] [ output [-185  25] ]
  if :p = [ 1 6 ] [ output [-155  25] ]
  if :p = [ 2 1 ] [ output [ 150 100] ]
  if :p = [ 2 2 ] [ output [ 250 100] ]
  if :p = [ 2 3 ] [ output [ 155  25] ]
  if :p = [ 2 4 ] [ output [ 185  25] ]
  if :p = [ 2 5 ] [ output [ 215  25] ]
  if :p = [ 2 6 ] [ output [ 245  25] ]
  output [ 0 0 ]
end
```

```
to String$ :k :x
   if ( count :x ) = :k [ output :x ]
   if ( count :x ) > :k [ output String$ :k "## ]
   output ( word char 32 String$ :k - 1 :x  )
end

to Gensuu :w
   local "r
   make "r random 2
   c [/* show ( list :r :w ) */ ]
   output item ( 2 + :r ) :w
end

to Gender
   if ( random 2 ) = 1 [ output "  ]
   output   2
end

to GenderCheck?
   print [ /* この GenderCheck? は工事中です. */ ]
   c [/* メスメス  カップルとか, オスオス  カップルとかチェックします*/]
   output 1 = 0
end

to DrawKeizu :x
   st
   setc 0
   left 90 pendown forward 30 penup forward 20
   right 90 setc TColor   item 1 :x pendown stamp
   penup forward 25 pendown label 1  penup back 25
   setc 0
   right 90 forward 20 pendown forward 60
   penup forward 20
   left 90  setc TColor item 2 :x pendown stamp
   penup forward 25 pendown label 2 penup back 25
   setc 0
   left 90  forward 20 pendown forward 30
   left 90  forward 50 right 90 forward 45
   right 90
   make "nLabel 3
   repeat 3 [
     back 10 penup back 15
     setc TColor item :nLabel :x pendown stamp
     penup   back 30 pendown label :nLabel penup
     setc 0
     forward 55 pendown
     right 90 forward 30 left 90
     make "nLabel :nLabel + 1
   ]
     back 10 penup back 15
     setc TColor item :nLabel :x pendown stamp
     penup   back 30 pendown label :nLabel penup
     setc 0
     forward 55 pendown
     penup
```

```
    ht
end

to TColor "w
  c [/* 0:優性な遺伝子，1:劣性な遺伝子，2:性別遺伝子 */]
  if :w = "q00 [ output 2 ]
  if :w = "q01 [ output 2 ]
  if :w = "q10 [ output 2 ]
  if :w = "q11  [ output  3 ]
  if :w = "q002 [ output  1 ]
  if :w = "q012 [ output  1 ]
  if :w = "q102 [ output  1 ]
  if :w = "q112 [ output 5 ]
  output 1
end

to Pat :n
  if :n = 1 [ output [ q012 q01 q002 q01 q102 q11] ]
  output [ q112 q00 q10 q102 q102 q10 ]
end

to DeCode :m :k
  local "p
  make "p ( list :m :k )
  if :p = [ 1 1 ] [ output [-250 100] ]
  if :p = [ 1 2 ] [ output [-150 100] ]
  if :p = [ 1 3 ] [ output [-245  25] ]
  if :p = [ 1 4 ] [ output [-215  25] ]
  if :p = [ 1 5 ] [ output [-185  25] ]
  if :p = [ 1 6 ] [ output [-155  25] ]
  if :p = [ 2 1 ] [ output [ 150 100] ]
  if :p = [ 2 2 ] [ output [ 250 100] ]
  if :p = [ 2 3 ] [ output [ 155  25] ]
  if :p = [ 2 4 ] [ output [ 185  25] ]
  if :p = [ 2 5 ] [ output [ 215  25] ]
  if :p = [ 2 6 ] [ output [ 245  25] ]
  output [ 0 0 ]
end

to String$ :k :x
   if ( count :x ) = :k [ output :x ]
   if ( count :x ) > :k [ output String$ :k "## ]
   output ( word char 32 String$ :k - 1 :x  )
end

to Gensuu :w
   local "r
   make "r random 2
   c [/* show ( list :r :w ) */ ]
   output item ( 2 + :r ) :w
end

to Gender
   if ( random 2 ) = 1 [ output "  ]
   output   2
```

```
end

to GenderCheck?
   print [ /* この GenderCheck? は工事中です. */ ]
   c [/* メスメス　カップルとか, オスオス　カップルとかチェックします */]
   output 1 = 0
end
```

8.5　プログラムについての実験結果

	ON	OS	M N	M S
1 A 1	1(25) < 25:7 >	1(25) < 25:5 >	1(25) < 24:1 >	1(25) < 24:7 >
1 Ä 2	1(25) < 23:9 >	1(25) < 25:3 >	1(25) < 26:6 >	1(25) < 24:2 >
1 Ä 3	1(50) < 54 >	0(0) < 0 >	1(50) < 46 >	0(0) < 0 >
1 Ä 4	1(25) < 23:8 >	1(25) < 25:3 >	1(25) < 25:1 >	1(25) < 25:8 >
1 Ä 5	1(25) < 26 >	1(25) < 24:5 >	1(25) < 24:5 >	1(25) < 25 >
1 Ä 6	0(0) < 0 >	1(25) < 49:2 >	0(0) < 0 >	1(25) < 50:8 >
	ON	OS	M N	M S
2 A 1	1(50) < 48:3 >	0(0) < 0 >	1(50) < 51:7 >	0(0) < 0 >
2 Ä 2	1(50) < 50:7 >	0(0) < 0 >	1(50) < 49:3 >	0(0) < 0 >
2 Ä 3	1(50) < 55:3 >	0(0) < 0 >	1(50) < 44:7 >	0(0) < 0 >
2 Ä 4	1(50) < 55:3 >	0(0) < 0 >	1(50) < 44:7 >	0(0) < 0 >
2 Ä 5	1(50) < 52:3 >	0(0) < 0 >	1(50) < 47:7 >	0(0) < 0 >
2 Ä 6	1(50) < 51:7 >	0(0) < 0 >	1(50) < 48:3 >	0(0) < 0 >
	ON	OS	M N	M S
3 A 1	3(37:5) < 37 >	1(12:5) < 13:3 >	3(37:5) < 39:3 >	1(12:5) < 10:3 >
3 Ä 2	3(37:5) < 39:3 >	1(12:5) < 11 >	3(37:5) < 37:3 >	1(12:5) < 12 >
3 Ä 3	1(50) < 47:3 >	0(0) < 0 >	1(50) < 52:7 >	0(0) < 0 >
3 Ä 4	3(37:5) < 37 >	1(12:5) < 11:3 >	3(37:5) < 39:7 >	1(12:5) < 13 >
3 Ä 5	3(37:5) < 36 >	1(12:5) < 12:3 >	3(37:5) < 40 >	1(12:5) < 11:7 >
3 Ä 6	1(25) < 25:7 >	1(25) < 23:3 >	1(25) < 30 >	1(25) < 21 >
	ON	OS	M N	M S
4 A 1	3(37:5) < 38:7 >	1(12:5) < 13:3 >	3(37:5) < 33:7 >	1(12:5) < 14:3 >
4 Ä 2	3(37:5) < 33:3 >	1(12:5) < 13 >	3(37:5) < 40:7 >	1(12:5) < 13:3 >
4 Ä 3	1(50) < 49:3 >	0(0) < 0 >	1(50) < 50:7 >	0(0) < 0 >
4 Ä 4	3(37:5) < 35:7 >	1(12:5) < 14:7 >	3(37:5) < 40 >	1(12:5) < 9:7 >
4 Ä 5	3(37:5) < 38:7 >	1(12:5) < 7:6 >	3(37:5) < 41 >	1(12:5) < 12:7 >
4 Ä 6	1(25) < 27 >	1(25) < 25:7 >	1(25) < 22:3 >	1(25) < 21:7 >
	ON	OS	M N	M S
5 A 1	3(37:5) < 34 >	1(12:5) < 16 >	3(37:5) < 35:7 >	1(12:5) < 14:3 >
5 Ä 2	3(37:5) < 38 >	1(12:5) < 9:7 >	3(37:5) < 37:7 >	1(12:5) < 14:7 >
5 Ä 3	1(50) < 54 >	0(0) < 0 >	1(50) < 46 >	0(0) < 0 >
5 Ä 4	3(37:5) < 38:3 >	1(12:5) < 10:7 >	3(37:5) < 42 >	1(12:5) < 9 >
5 Ä 5	3(37:5) < 39 >	1(12:5) < 12:3 >	3(37:5) < 37:3 >	1(12:5) < 11:3 >
5 Ä 6	1(25) < 24:7 >	1(25) < 26:3 >	1(25) < 23 >	1(25) < 26 >
	ON	OS	M N	M S
6 A 1	3(37:5) < 35 >	1(12:5) < 12:3 >	3(37:5) < 42:7 >	1(12:5) < 10 >
6 Ä 2	3(37:5) < 41:3 >	1(12:5) < 10:7 >	3(37:5) < 33:7 >	1(12:5) < 14:3 >
6 Ä 3	1(50) < 53:7 >	0(0) < 0 >	1(50) < 46:3 >	0(0) < 0 >
6 Ä 4	3(37:5) < 41 >	1(12:5) < 9:3 >	3(37:5) < 36:7 >	1(12:5) < 13 >
6 Ä 5	3(37:5) < 36 >	1(12:5) < 11 >	3(37:5) < 38:7 >	1(12:5) < 14:3 >
6 Ä 6	1(25) < 26:3 >	1(25) < 24:7 >	1(25) < 24:7 >	1(25) < 24:3 >

プログラムに関して，まずは遺伝子から考えられる組み合わせの結果と，実際にプログラムを使ってみたときに産まれてくる子供の結果のくい違いがあるかどうかを調べて表にまとめてみた.（オス同士，メス同士でも子供が出来ると考えたとして.）

8.5.1　実験結果

上の表は予想される結果の対比で，( ) の中が100匹子供を産んだ時に予想される子供の数，< > の中が実際の実験結果の子供の数である．ちなみにこの実験は、コマンドセンターにPKT 0 1と打ち込んだ場合の系図のものである.（左の系図が０番，右の系図が１番の系図となっている.）全ての考えられる組み合わせの亀に子供を100匹産ませ、それを３回繰り返した平均を取っている.（小数点第１位以下四捨五入.）

8.5.2　この実験結果で分かったこと

このプログラム上で，子供の実験結果に関してのばらつきは多少はあるものの，予想される結果とかけ離れて違うわけでもない，言うことが分かった.

現実はこのように綺麗な結果が出ないこともある．例えば４人子供がいたとしても全員女だったり，全員男だったりする場合も実際にありえることである．確率はあくまででランダムなので，続けて男ばかりが生まれる場合もあるし女ばかりが生まれる場合もある.

しかし突き詰めて考えていけば，上の実験結果のように遺伝には規則性があることが分かると思う．このことは遺伝の研究で有名なメンデルも結論づけていることである.

メンデルが行なった実験[2] を参考に取りあげてみる.

以下はメンデルが行なった実験の結果の比である.

	優性	劣勢
種子の形態	2:96	1
アルブミンの色	3:01	1
種皮の色	3:15	1
莢の形態（成熟）	2:95	1
莢の色（未成熟）	2:82	1
花の序列	3:14	1
主軸の高さ	2:84	1

メンデルの実験結果もほぼ３対１の比に近いということで，優性の形質と劣勢の形質がほぼ３対１の比で表れているということを結論付けている．しかしこれは何個もの個体からできた種子の数をたして平均化したものなので，一つ一つの個体を見た場合はもちろんバラつきが出てくる。

以下は種子の形態とアルブミンの色に対する実験結果のうち，10個体を参考にしたものである.

[2] メンデル他/山下孝介　訳「植物の雑種に関する実験」,『現代の科学1』世界の名著39：中央公論社，1866/1979

植物番号		1	2	3	4	5	6	7	8	9	10
種子の形態	丸	45	27	24	19	32	26	88	22	28	25
	しわ	12	8	7	10	11	6	24	10	6	7
アルブミンの色	黄	25	32	14	70	24	20	32	44	50	44
	緑	11	7	5	27	13	6	13	9	14	18

この表を見れば，形質の分離が個々の植物によって違うことが明らかであると思う．しかし１つ１つを見ると規則性が無さそうに見えるが，同じ植物を使って実験を繰り返し平均を取った結果，ある規則性にたどりつくことをメンデルは発見したのである．

8.6　感想とまとめ

このプログラムの目的としては遺伝の確率を求めるというよりも，両親とは違った色の子供が産まれてくることがあるという驚きや，なぜそうなるのか考えるための教材なのである．

例えば一見同じ色どうしの組み合わせでも，生まれてくる子供の比率が違うということがあり得る．

・例えば

左の系図で　１（メス並），右の系図で３（オス並）と
左の系図で　１（メス並），右の系図で５（オス並）

を選んだ場合など．

左１，右３で選んだ場合オス並，メス並の亀しか生まれないが左１，右５で選んだ場合はオス並，オス薄，メス並，メス薄の亀が生まれてくる．

このように一見同じような組み合わせでも、産まれてくる子供が違うことで「どうしてなのかな？」とか「こういう理由でじゃないかな？」と考えてもらいたい.「どうしてなのかな？」と考えたとき，優性遺伝子と劣性遺伝子が‥という考えにたどり着く人はあまりいないと思われるが，「一見同じように見えても何か隠された秘密があるのでは？」という考えにたどり着くことは出来るのではないかと思う．その隠された秘密が一体どういうものなのかを解説する．

外に現れた形質が同じでも，遺伝子情報が違うという場合がありえる．例を挙げて説明すると，見た目は優勢の形質が現れているが，遺伝子情報では優性と劣性どちらも持っている植物同士を掛け合わせたとする．その結果，優性の形質が現れている子供と，劣性の形質が現れている子供が３：１の割合で出現する．この実験結果のもっと詳しい遺伝子情報を考えてみると，優性だけの遺伝子を持った子供：優性と劣性の遺伝子両方を持った子供：劣性だけの遺伝子を持った子供が１：２：１の割合で現れているのである．優性と劣性の遺伝子両方を持った子供の場合，劣性の形質は優性の形質に抑えられて，見た目上は優性だけの遺伝子を持った子供と同じ形質を示す．だから優性の形質が現れている子供と，劣性の形質が現れている子供が３：１の割合で現れてい

るように見えるのである．隠された秘密と言うのは，見た目の形質だけでは一概に決めつけることが出来ない遺伝子情報のことなのである．

　このような外見の形質だけからはわからない遺伝情報のことを遺伝子という．形質に関する遺伝情報は２つあって，それを片方ずつ両親から引き継ぐ．そして，同じ形質の遺伝子が２つ揃った場合のみ出現する方の形質を劣性，反対に１つでもあれば出現する形質を優性という．だから劣性遺伝子を持って居る人が劣っているってわけではなく，出現の仕方が優性よりもまれになりやすいということなのである．

　反省点として，このプログラムにはオス同士，メス同士でも子供が出来るという欠点がある．そこらへんは利用者の方にあえてオスメスの組み合わせで使ってもらうことをお願いしたい．

(山田　和代）

執筆者一覧

296108　服部真一
297027　上野明日香
297709　今井由果
297710　岩崎悦子
297726　小林佳子
297728　坂倉彰仁
297759　山田和代
以上7名.

2000年度卒業ゼミナール報告書

Logo言語の教育場面での応用

2001（平成13）年1月12日印刷
2001（平成13）年1月17日提出予定

2001年 1月 19日TEX出力

三重大学教育学部
卒業研究（正田教官担当）生
別記7名