イプシロン　2015.Vol.57, 95 − 102

Scratch プログラミングで「学びの楽しさをもう一度」
−内発的な学びを再体験する試み−

愛知教育大学　橋　本　行　洋

概要

「『自律的に学ぶ』という心の態度」—これは 2013 年度入学生から始まった初年次導入教育で学生に是非身につけて欲しいと掲げたテーマでした [2]. その際もっとも重視したのが学生から自然に湧き上がる「なぜ，どうして」でした．その後の様々な講義においてもこの自然に湧く自発的な疑問と興味の下，学生各自が試行錯誤し自分で答えを探す活動が維持されることが理想ではあるけれど，初年次教育ではわずか 5 回の講義のため，あっという間に受動的な学習態度に戻ってしまいます．そこで通常の座学中心の数学講義とは異質なものになり得る「プログラミング」の講義において，受動的な学習習慣に埋もれててしまった「内発的動機による学び」をもう一度掘り起こすことを最大の目標に定め，講義を設計しなおしました．このノートはその一つの実践記録を兼ねた試論です．

1　ソフトウェア世代の学び

1.1　子どもの頃は「なぜ」だらけ

振り返ってみると，子どもの頃の世界は「なぜ」や「不思議」で溢れていたと思う.「どうして太陽は雲に隠れるのに，雲は太陽に隠れないの？」とか，「湯が沸くとき出てくる泡はどこから生まれるの？」とか,「濡れた机の上に熱い汁の入ったお椀を置くと動き出すぞ！」とか．それに現代と違って身の回りの機械はみなアナログだった．つまりほとんどの仕組みがブラックボックスとなるデジタル化された現代とは異なり，子どもであっても中がどうなっているのか実際に目で見てその仕組みを体験できたものだった．例えば柱時計．五円玉をひもで釣り下げて振ってもやがて止まってしまうのに，一時間ごとに「ボーン」と鳴る柱時計の振り子はずっと止まらない．そして中からはなにやらカッ，カッ，と音がする．一体何が振り子を動かしてるのだろう，と振り子のために空いている穴を下からのぞき込むと，なにやらぐるぐる巻かれたものがちらりと見える．どうしてあんなぐるぐるの帯[1]が振り子を動かすのだろうと気になり始める．数年してその柱時計が故障したのでオモチャにできるようになった．そうなるとやることは一つ，ふたを開けてどんな仕組みか調べてみるのだ．ちょっとずつネジを外して動きを見る．ぐるぐるの帯は触ると強靭なバネのようで弾力があり，振り子を持って速く動かすと次第にそのぐるぐるが大きく広がっていく．そうか，バネがギュッと押し縮められているからなのか，と何となく分かってくる．振り子を動かす度に歯車たちが丹念に一つずつ回っていく．無理やり針を動かして 12 時になる直前まで持っていき振り子を動かして何が起こるか観察する．丁度 12 時になると，それまで気付かなかった部品たちが一斉に動き出す．計算されたこのメカニカルな動きの虜になる．そうやってあれこれいじっているうちにネジを無くし，針が取れ，どこに付いていたのか分からない部品が外れ，そしてついには全く元に戻せなくなる.「そういえば俺も子どもの頃，幾つも時計を壊して親に怒られたものだ.」と父がつぶやく．こうした「どうなってるんだろう？」と湧きあがる興味を抑えきれず，熱中して機械を分解しているうちに，子どもながらに仕組みを理解していったものだった．アナログ世代にとってこういった経験は比較的自然にできていたのだと思う．

1.2　デジタルな現代はブラックボックス

一方，そういった観点からするとデジタルな現代は子どもにとって内発的な学びを起こし辛い環境にあると言えそうだ．アナログな機械なら一つ一つの部品の役割は大抵一義的である．つまり，この部品は次のこの部分に力を伝えるためにある，といったように．それに対しデジタル機器では各部位が多義的，つまり一つの部位が何通りもの役割をする．もちろん集積回路の最小単位は on/off だけを表す単純なスイッチにすぎないが，そのスイッチ同士のつなぎ方が状況に応じて変化する，だから多義的になるのだ．あるいは仕組みのソフトウェア化という言い方もできる．ゲームができ，通信ができ，地図にもなり，カメラにもテレビにも音楽プレイヤーにもなり，ついでに会話もできるスマートフォンは，一台でいくつもの役割を果たす今や最も身近な万能

[1] もしかするともう今の 20 代は知らないかもしれない，ゼンマイのことだ．

機械である．しかしこういったデジタル機器のふたを開けても柱時計のように仕組みが分かるわけではない．どのパーツがどんな役割をしているかを試行錯誤から推測していくという作業などとてもできない．だから生まれたときからそのような道具たちに囲まれてきた今日の学生にとって，世の中のほとんどのことがブラックボックスのままななんじゃないかと想像する．たとえ教えられてもその仕組みをどこか遠くの世界のこととして仮想的にしか知りえず，だから体験に裏打ちされた根っこの生えた理解は結局されないまま過ごしていくのではなかろうか．実際この数年，具体的に自分で手を動かし試行錯誤して体験的に物事を理解していくという，我々からすると誰からも教わることなくごく自然にやってきた学び方がさっぱりできない学生が，どんどん増えてきていると強く感じる．世間的には「アクティブラーニングこそがこれからの時代に必要だ」などと叫んでいながら，将来教員になるはずの肝心の学生たち自身，そういった学び方ができないでいる．もちろん現行の大学カリキュラムの構造的な理由に依るものであったり，あるいは「教育の客体としての学生」から「学びの主体としての学生」への意識転換が大学教師陣側で進んでいないことも大きな原因であろうとは感じる．特に自分の所属する数学内容学ではどうしても座学中心となりやすく，学生に試行錯誤を経験させる機会が生まれにくい．

1.3 学生よ，試行錯誤しよう！

一つ一つ順に動かしてみて試行錯誤して仕組みを理解していく学び方，ああいった経験をスマートフォン世代の学生らにどうしたらさせられるのか？そういう問題意識をずっと抱いていたこともあって，私が長年担当している「プログラミング」の講義では，決して単なる座学とならぬよう仕組み，仕掛けをしてきたつもりである．少なくとも学生の目から見て他の座学の講義からは異質なもの，と受け止められることを目指してきた．講義では長年，白石和夫先生が windows 用に開発された（仮称）十進 BASIC[7] を教材に使ってきた．C や Java などと異なり変数宣言が簡素なうえ，インタプリタ言語のためコンパイルしてから実行という手間も無く，初心者に使いやすいという理由から選んできたのではあるが，よく観察してみると，どうやらプログラムが「英単語」で書かれていることが当数学教育講座の学生にとっては結構なネックだったらしく，結局使いこなせるようになった学生はいつも僅かだった．またプログラミングの初歩から話を展開しながら更に絵が動くといったビジュアル的なことまでを追究するには，どうしても講義回数が足りなくなり，学生から見て本当に面白くなるところまで行きつけないことも不満だった．

そこで 2013 年度から当大学で始まった初年次教育における私の講義で「主体的な学び」を意識させた学生たちが，丁度このプログラミング講義を受ける機会となるのに合わせ，使用言語を変更することにした．それが「子供用のプログラミング言語」Scratch[5] である．開発者は学習科学の研究者，MIT メディアラボ在籍のミッチェル・レズニック氏である．そのコンセプトは子どもが LEGO ブロックで遊びながら様々なものを作っていくような感覚で，ごく自然にプログラムが組めるようにしようというものである．誰からも教わることなく見よう見真似で小学生の頃からプログラミングしてきた私からすると，この考えには強く共感するものである．実際子どもたちが Scratch でごく自然にプログラムを学んでいく様子がレズニックの TED におけるトークで紹介されているので是非御覧頂きたい[6]．面白いと思いさえすれば子どもはどこまででも学んでいくのである．（因みに，この「子どもは面白いと感じられれば勝手にいくらでも学んでいく」という現象を積極的に教育に持ち込み，世界中の教育困難地域で web を利用した自己学習環境—いわゆる教師を必要としない自発的学習の場—を置くことで子どもたちが学ぶ場を提供する試みを教育科学者スガタ・ミトラが行っている．教育を根底から変える可能性のあるこの試みは，彼自身の TED におけるトークとして公開されているので注目されたい[3].）そしてまさにこれと同じ現象を学生に—彼らが子どもの頃にもきっとあったであろう，純粋に何かを面白いと思い夢中になって追究した記憶を呼び覚まして—再体験してもらうことが当講義の大きな目的なのである．

図 1: Scratch のキャラクター．この様なキャラクターがあらかじめ多数用意されている．

```
INPUT PROMPT "数を入れてください":n
LET k=1
DO UNTIL k>n
  IF MOD(n,k)=0 THEN
    PRINT k;
  END IF
  LET k=k+1
LOOP
END
```

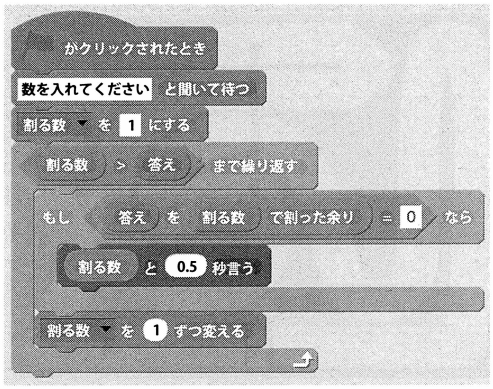


図 2: 入力した数の約数を表示する BASIC プログラム (左) と Scratch スクリプト (右)．ただし BASIC では Scratch の文法に合わせて DO～LOOP 文で記述した．

2　実践報告

何はともあれ，図 2 を見て頂きたい．入力された整数の約数を表示するプログラムを十進 BASIC と Scratch で書いたものである．十進 BASIC が英単語の羅列にも見えるのに対し，Scratch では日本語で書かれたいくつかのブロックの組合せでプログラムが出来ている．しかも繰り返しや場合分けがブロックの形で直感的に見えるようにできている．まさに LEGO ブロックを組み立てていくようにプログラムが組めるのである．それに加え，スプライトと呼ばれる行動主体（キャラクター）ごとにプログラムを組んで，とても容易にそれらをアニメーションとして動かせるよう作られているのだ．「日本語で (つまり母国語で) プログラムが書ける」「目の前のキャラクターがブロックの組合せで自由に動作させられる」この 2 点によってプログラミング初心者に対する垣根を取り払い，その結果，小学校低学年であってもプログラムを容易に組み立てられるようになる．

しかし，垣根がどんなに低くなろうと，そもそも学ぶ側にその気が無ければ何も起こらない．初年次教育においては数理手品を用い，目の前で起こる，あるいは自分の手の中で起こる一見不思議な現象によって内発的な「なぜ」を自然に引っぱり出すことを目論んだ．一方，プログラミングにおいては目の前でキャラクターたちが容易に動くため，最小限の道具の説明と大雑把な目標さえ用意しておけば，学生側からごく自然にキャラクターたちを思うように動かしたいという欲求が生まれる．それこそがこの講義における学びの火種であり，だから講義をする側はこれをうっかり消してしまわぬよう細心の注意を払いながら，その火種から学びの大きな炎へと導けるよう設計しておくことが必要だ．気付けばクラス全体が「知の Kindergarten」となっている，これが理想とするところである．以下にその試行錯誤の一端を報告する．

2.1　チートを尽くして迷路を駆け抜けろ

初回は Scratch そのものをインストールし，直ちにキャラクターを動かすスクリプト（Scratch ではキャラクターごとに付与できるプログラムをスクリプトと呼んでいる）を組んでもらう．しかしこれは丁度あらゆるプログラミング言語で最初に登場する「Hello world」と同程度の易しさで，すぐにあちこち動き回るキャラクターが生まれる．しばらく遊んでいると当然次はこのキャラクターたちをキー操作などで意のままに動かしたい，という衝動が学生らの間に生まれる．そこで間髪入れず「もし」ブロック (IF 文) と「○キーが押された」ブロックを紹介し，押されたキーで向きを変えるスクリプトを作る．さらに実行画面の背景に道を描いて道からはみ出したらアウトという条件のもとキャラクターを動かし続けるために，条件付き繰り返しである「○○まで繰り返す」ブロックを紹介する．さて，これだけでレーシングゲームが作れてしまう．それでは，ということでこちらで指定したコースをタイムアタックで走破するゲームを作り，実際にゲームを行ってクリアーした時間を報告する，という最初の課題を課

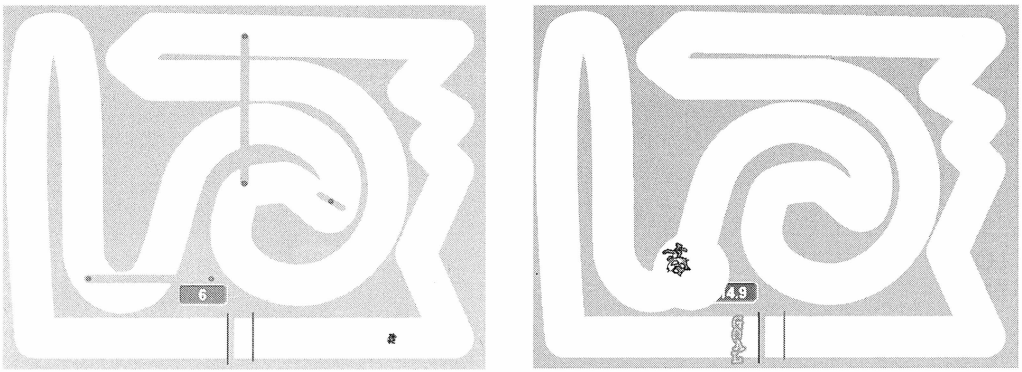


図 3: 最初の課題，チートを尽くしてコースを走破せよ．左は課したオリジナルのコースで障害となる棒が動き回るように作ってある．それに対し，例えば学生からは右のような作品が作られた．これはレーサーであるネコの周りに白いシールドを作ることでコースから外れたと判断させず，しかしこのままではゴールラインにも触れられないので [GOAL] という文字列に触れたらそのシールドが消えるように作られていた (2014 年度 京屋 朝香さんの作品)．

す（図 3 左）．ただ，これではキー操作の上手い下手の話になってしまうので，「ズルをして良い」と付け加えた．つまり，できるだけ短時間でゴールしてしまえるように思いつくあらゆるチートを考え，そのアイディアをスクリプトに組み込んで良い，としたわけである．そして「しなさい」ではなく「しても良い」としたところに意味があると思われるのだ．しかもそれは「ズルをして良い」である．少なくとも「課題をせねばならない」意識から多少とも気を逸らし，どんなズルをしようかなどと考えるうちに子どものように問題に向き合うことになるだろうという企てである．そして案の定，様々なチートが学生から生み出された (図 3 右)．

2.2 シューティングゲームを作れ

Scratch の制作環境に慣れてきたところで，次に目指したのはシューティングゲームの制作である．どんな学生でもしたことがあるゲームジャンルであり，かつ習得目的を明確に持ちやすい，つまり習得した事柄をそのまま自分の作品に反映させたいと思うため目的を持って学びやすく，何より容易に試行錯誤へと学生たちを誘い込めるだろうという意図から企画した．レーシングゲームと違いキー操作できる自機以外の敵キャラクターたちを自動的に行動させ，かつ自機と敵，あるいは弾との当り判定が必要となってくる．それでも目指すべき最終形は各学生に明確に見えているため，実際に講義を行ってみると学生内で試行錯誤が自ずと起こり始めた．そうしてある程度作り方が分かってくると，次々とアイディアが湧きそのアイディアを実現すべく更なる試行錯誤の海へと多くの学生が埋没していった．中にはまだ紹介していない技術や命令ブロックを使った作品も現れた（図 4）．学生に内発的な学びの炎が上がった結果だと言えよう．

2.3 算数・数学の教材を作れ

シューティングゲーム作りでは何より「思ったように動作させたい」という内的な欲求を呼び覚ますことが目的であった．その際，知的興奮という点きかけた炎を消してしまわぬよう，敢えて細かい注意は避け必要最低限と思われる事柄のみを講義にて教えた．したがって「良くは分からないけどこうするとなぜか上手く動く」程度の浅い理解に留まっている学生が多い状態である．そこでここからはいま一度丁寧にスクリプトを組む作業を行うこととした．まずは既に図 2 で見た自然数 N の約数を求めるスクリプトである．もっとも素朴な約数の求め方は小学校で習うように「1 から順に N を割って割り切った数を並べていく」であり，すべき手続きがはっきりしている．だからこの手続きをスクリプトとして実現できるか，例を見せる前に時間をかけて試行錯誤させた方が，その後の理解は深まったかも

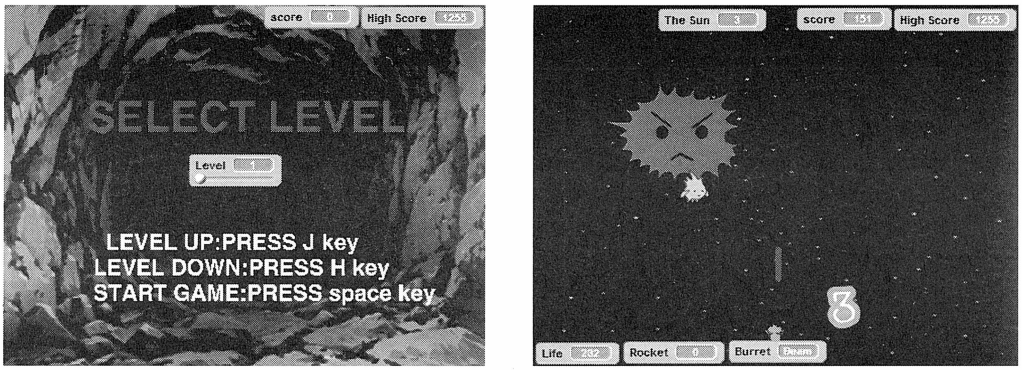


図 4: 課題：シューティングゲームを作れ．学生それぞれが思い思いのゲームを作った．非常に凝った作品も作られ，まだ紹介していない技術や命令ブロックを使った作品も現れた．上記の作品ではゲームの難易度が設定でき（左），この後多彩なステージ，ボスキャラが現れる（右）．(2014 年度 山本 遼さんの作品)．

しれない．あらためて振り返るとそう思うこと頻りなのではあるが，実際の講義ではさらりと進めてしまった．「約数を求める」スクリプトではもう一つ，素朴だが重要な考え方である「条件に合うものの個数を数える」方法，いまの場合は約数の個数を求める方法について紹介できる．カウント用の変数を一つ作っておき，条件に合ったときだけ変数の中身を一つ増やすという方法だが，これは太古の時代から人々が行ってきた，一対一対応原理によるカウント法である．個数が分かるなら調べている数が素数か否かも判定できる．つまり約数が丁度 2 個ならそれは素数である．ただ，巨大な数 N の素数判定となると 1 から N まで割っていると時間がかかり過ぎるので，一つの工夫として数学的事実

$$N = ab \text{ ならば } a \leq \sqrt{N} \text{ または } b \leq \sqrt{N}$$

を紹介し，2 から $\sqrt{N}$ までで割り切れなければ素数と判定するよう作り変えてもらう[2]．講義の進め方としては，まず「1 から N まで全て割る」方法でかかる時間を体験してもらう．特に $N = 11111111111$(1 が 11 個) は素数か？と問えばとても時間がかかることになる[3]．そこで「2 から $\sqrt{N}$ まで割る」方法に変更すれば劇的に時間短縮されるのが実感できる．数学的な工夫の有り難味がよく分かる瞬間である．またプログラムの工夫ひとつで効率が格段に変わることがあるという好例でもある．一方これを「約数だったときその約数を新たに加える」に変えれば約数の和も求められるので，完全数探しといった話題にも展開できる．さらに続く講義では高校数学にもはっきりと登場することとなった，そして計算アルゴリズムの典型ともいえる Euclid の互除法を Scratch で再現した．このときは複数の変数を扱い，順番を間違えず手順通りに変数の中身を移し替えねば正しい答えが得られない．この機会にブロックを並べる順に気をつけねばならない場面があることを経験してもらった．

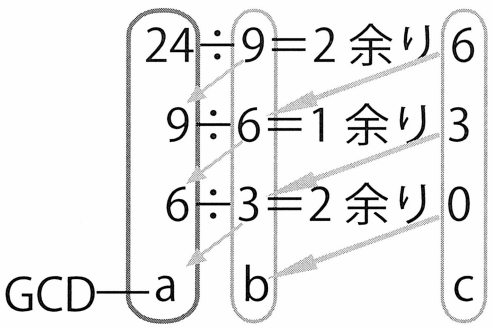
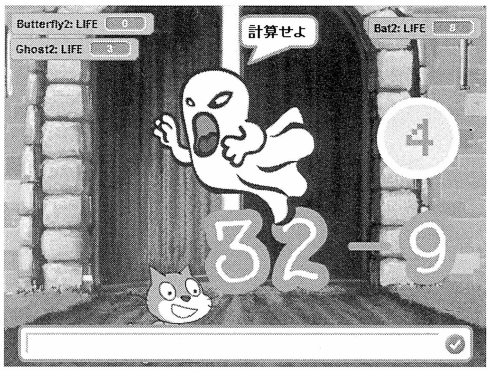


図 5: Euclid の互除法．変数の代入順は $b \to a$ のち $c \to b$ なのか，$c \to b$ のち $b \to a$ なのかが問われる．

最大公約数を求められれば最小公倍数，分数の約分などが Scratch の上で実現できるようになるので，

[2]ただしこの方法では，$N = 1$ の場合だけ「素数でない」と表示するよう別扱いで処理せねばならない．

[3]$11111111111 = 21649 \times 513239$. 実行してみると十万まで割るのにおよそ 0.5 秒かかったので，全て割るまでに $0.5 \times 10^5 = 50000sec \fallingdotseq 14$ 時間かかることになる．



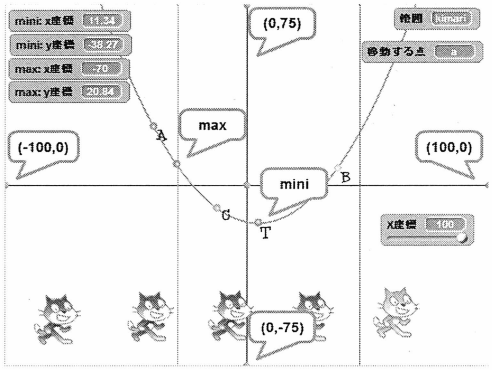


図 6: 課題：算数・数学の教材コンテンツを作れ．教育学部の学生らしい作品が多数作られた．左は出題される足し算引き算に答えるとステージが進むゲーム仕立ての作品（2014 年度 坂井 由季さんの作品）．右は 3 頂点を決めるとそこを通る二次関数が描かれ，指定した範囲内の最大値最小値を表示するなど，動的に二次関数の性質を調べられる自己学習用コンテンツ（2014 年度 森崎 智香さんの作品）．

算数・数学の教材のアイディアの幅が広がる．そこでこのシリーズの前半では「整数を素材にした教材やゲームを作れ」という課題を設定した．あるいは数学的な理屈をプログラムの上で実験して確認する，というコンピュータの使い方も体験してもらう目的で，プロジェクト・オイラー [4] に上げられている問題[4]を Scratch で解く，でも良いとした．

一方，Scratch には貧弱ではあるが描画機能がある．「ペン」ブロックたちである．これを利用して数式のグラフ描画を行ったり，あるいは図形の頂点を動かすなど動的に幾何学的性質を観察できる簡易図形教材を作る試みも行い，課題を「図形を素材にした教材やゲームを作れ」とした．その結果いずれの課題でも教育学部らしい教材が多数提出された（図 6）．このシリーズでの体験が将来教育現場でデジタルコンテンツを扱ったり，あるいは制作サイドに回った際に役立てば幸いである[5]．

2.4 クローンと配列と最終課題

講義の最後のシリーズでは，どうしても紹介しておきたかった「クローン」と「配列（Scratch ではリストと呼ばれる）」を利用した．クローンとは同等な動作をするキャラクターたちを自動で複製し制御する方法であり，これを知っていると無数の敵が現れるシューティングゲームなどは格段にプログラムがシンプルになる．もちろん，課題として提出されたシューティングゲームの中には，既にこのクローンを使ったものがいくつか現れていた．これも学生が内発的に反応した結果と考えられよう．一方，配列とは番号づけられた変数の束であり，あるいはそれがサイズ n の配列なら n 次元ベクトルだともいえる．多数のデータを保存し，利用や加工をしたいときに用いる仕組みだ．講義では 2 オクターブ分の音階に合わせたボタンを配置し，どのボタンが押されたか順次記録し，またそれを再生する「簡易録音再生装置」の作り方を紹介した（図 7）．

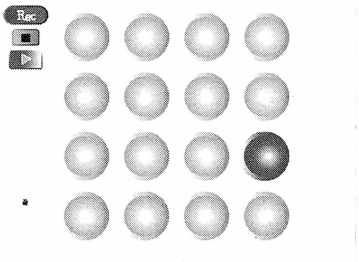


図 7: 配列の使い方の例としての録音再生装置．クリックした順番が記録され，後で再生できる．

[4]Project Euler はプログラムを組んで数学の問題を解くという世界中の愛好家たちが参加しているプロジェクト．2015 年 11 月現在，530 問以上が掲載され参加者は解いた問題数に応じてレベルが付与される．レベルは 1 から 17 まであり，レベル 17 は全世界で 65 名，そのうち日本人は 6 名とのこと．

[5]実際，この講義後に教育実習に行った学生の中には Scratch で教材を作り研究授業で利用した，という強者も現れた．予想外であった．

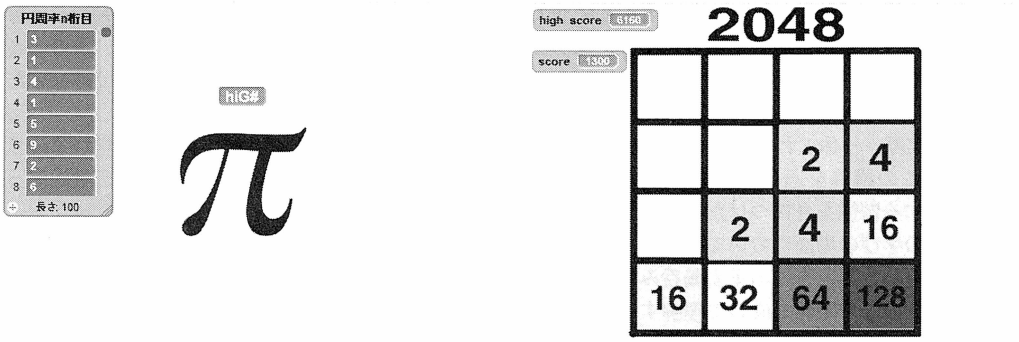


図 8: 最終課題：クローンまたは配列を使って自由に作れ．それぞれがそれぞれの力量の範囲で様々な作品を作った．左は円周率の数字を元にして自動的に奏でられる「円周率の音楽」．ルールにしたがってコード（和音）も一緒に鳴る仕組みになっていた（2014 年度 安達 駿弥さんの作品）．右は有名な数理ゲーム 2048．完全オリジナルゲームというわけではないが，膨大なプログラムを解析し遂に自力で作り上げた（2014 年度 橋本 一慶さんの作品）．

そしていよいよ最後の課題である．「配列かクローンを使っていること」を条件として自由課題とした．学生それぞれがその力量の範囲で様々な作品を作ってきた．図 8 はそのほんの一例に過ぎない．背景も一緒に流れる横スクロールゲームをまったくオリジナルな方法で作ったものや，Scratch のアニメーション効果をフルに活用し，季節の移り変わりや夜明けを表現した物語性のある作品，確率現象をシミュレーションによって確かめられるようにした作品，さらには非常に精巧なプログラミングによってトランプゲーム「大貧民」を作った者もいた[6]．こうして講義の最後まで，学びの炎を燃やし続けてくれた学生が多く残ってくれたことに感謝する次第である．

3 Learning over Education

前著 [2] でも引き合いに出したのであるが，MIT メディアラボ所長 伊藤穰一氏の唱える教育論，いや学習論を表すこの端的な言葉を最後の節のタイトルとした．新聞紙面では「教育より学びを」とあたかも教育と学びが対峙するような言い回しで紹介されるが，むしろ教育システムを超えて学ぼうという呼びかけだと解釈している．教育者の客体としての存在から学びの主体になろう，ということだ．むしろ「教育」とは「学び」という広範な知的活動の一部分であるという認識こそが，学ぶ側，教える側双方にとって今後ますます必要になってくるだろう．学びへと広がっていかない，謂わば閉ざされた教育は両者にとって悲劇以外の何物でもない．なぜならデシとフラストの研究 [1] にも示されているように，20 世紀半ば頃から自己原因性―学び手当人が学びたいから学んでいるのだという感覚―を伴わない教育の不毛さを科学的根拠と共に世界は十分に認識し始めているのだから．

> 人には，自分の自律性あるいは自己決定の感覚―ド・シャームが自己原因性と呼んだ感覚―を経験したいという生得的な内発的欲求があると思われる．このことを言い換えるなら，人は自らの行動を外的な要因によって強制されるのではなく自分自身で選んだと感じる必要があるし，行動を始める原因が外部にあるのではなく内部にあると思う必要がある，ということである．
> （デシ&フラスト「人を伸ばす力―内発と自律のすすめ」第 3 章「自律を求めて」p40）

本実践は Scratch プログラミングという形を通して，長年の受動的教育によって学生の内に眠ってし

[6] これら最終課題の作品のいくつかは，拙論ブログ「遊び tokidoki 仕事」の 2015 年 3 月 14 日の記事に載せてある．http://tokidoki.hatenablog.jp/entry/2015/03/24/203000

まった内発的な「学ぶ心」を再発掘するという大きな目標を掲げて試みたものである．傍目には子ども騙しの飛び道具で学生の興味を引くことに成功した，ただそれだけではないのか，という見方もあるだろう．実際，一体どれぐらいの学生が半年間にも満たないこの講義を通して内発的な動機の下で試行錯誤をする習慣を身につけたのだろうか．そしてそれが他の学びの場面においても発揮できているのか，教えられたことをそのまま鵜呑みにせず，必ず一度は自分の頭で咀嚼し直して理解するよう努めているのかは分からない．つまり受動的な学習態度にどれほどの変化を与えられたのか検証できない．けれども方向性は間違っていないと思われる．少なくとも「試行錯誤によって学ぶという方法がこうして確かにあるのだ」という経験を得る機会を提供できたらしいことが，各学生から提出された作品のそれぞれの進化と深化から読み取れるからである．学び方の幅が広がったと学生らに感じてもらえたなら幸いである．更には，単位取得のためと外発的に動機づけられた作品作りが，(たとえ短期間の疑似的なものであったとしても）知らぬ間に内発的な知的欲求を満たすために作る，と動機が内在化していったのなら願ってもないことである．

教育の本分は，それを受けた者が自分自身の力でこの世界を生き抜き，できるだけ人間的に豊かな人生を創造するための智慧と力を育てることにある．そしてこの大学はその教育を将来担う学生を育てる場所だ．その彼らが教壇に立ったとき，自発的に学ぶ子供たちを育てられるか否か，それは他ならぬ学生自身が内発的動機の下で自律的に学ぶ経験をどれだけ積んできたか，にかかっている．身体化した経験でなければ伝えられるはずもない．だから何より学生自身が生き抜くための智慧を身につけねばならない．そして教員養成大学の教師陣には単位取得や高得点を最大目的とした外発的で虚しい「お勉強」からいち早く学生を卒業させ，内発的動機に基いた自発的・自律的な探求へと誘うという大切な仕事があるように思う．それは教壇に立つ度，次のことをちょっと自問してみるだけで変わるのではないだろうか．

「我々のしていることが，学習者の学びを侵していないか？彼らの学びにおける自己原因性を奪ってはいないだろうか？」

参考文献

[1] E.L. デシ, R. フラスト，人を伸ばす力—内発と自律のすすめ, 新曜社, 1999.

[2] 橋本 行洋, 自律的な学びへの誘い—初年次導入教育でのある試み—, イプシロン **55**, pp.53-65, 2013.

[3] S. Mitra, 自己学習にまつわる新しい試み, TED Global, https://www.ted.com/talks/sugata_mitra_the_child_driven_education?language=ja, 2010.

[4] Project Euler, https://projecteuler.net/, あるいはその日本語解説ページ http://odz.sakura.ne.jp/projecteuler/

[5] M. Resnick, 子供用プログラミング言語 Scratch, https://scratch.mit.edu/

[6] M. Resnick, 子供達にプログラミングを教えよう,TED at BeaconStreet, https://www.ted.com/talks/mitch_resnick_lets_teach_kids_to_code?language=ja, 2012.

[7] 白石 和夫, (仮称）十進 BASIC, http://hp.vector.co.jp/authors/VA008683/