| | |
|---|---|
| Title | 学校教育におけるプログラミング教育の指導者向け入門ガイドのデザイン |
| Sub Title | Design of beginner's guide for teachers to teach programming as a school education |
| Author | 成富, 紘(Naritomi, Ko)<br>石戸, 奈々子(Ishido, Nanako) |
| Publisher | 慶應義塾大学大学院メディアデザイン研究科 |
| Publication year | 2015 |
| Jtitle | |
| JaLC DOI | |
| Abstract | |
| Notes | 修士学位論文. 2015年度メディアデザイン学 第464号 |
| Genre | Thesis or Dissertation |
| URL | https://koara.lib.keio.ac.jp/xoonips/modules/xoonips/detail.php?koara_id=KO40001001-00002015-0464 |

修士論文 2015年度（平成27年度）

# 学校教育における
# プログラミング教育の指導者向け入門ガイド
# のデザイン

慶應義塾大学大学院
メディアデザイン研究科

成富　紘

本論文は慶應義塾大学大学院メディアデザイン研究科に
修士 (メディアデザイン学) 授与の要件として提出した修士論文である。

成富　紘

審査委員：

石戸　奈々子 准教授　（主査）
岸　博幸 教授　（副査）
加藤 朗 教授　（副査）

修士論文 2015年度（平成27年度）

# 学校教育における
# プログラミング教育の指導者向け入門ガイドのデザイン

カテゴリー：デザイン

## 論文要旨

近年、プログラミング教育の重要性が高まっており、欧米諸国を中心に義務教育化が進んでいる。一方で日本は「世界IT最先端IT国家創造宣言」を発表するなど方針としてはあるものの、検討段階にある。現在行われているプログラミングワークショップへの参加や、プログラミング教育実践者、また学校教員へのヒアリングを通して、現状の課題点を整理した。その結果、教師のプログラミングへの知識不足・カリキュラムが無い・多忙による準備の負荷の大きく３つに絞ることができた。そしてそれを解決する手段としてHour of Cordが適しているとの仮説を立て、ワークショップへの参加を通して検証した。その上で、Hour of Codeを用いた授業を行うための基礎知識を未経験者向けに解説した教員向け入門ガイドを作成した。これを小学校の教師や教育者、プログラミング教育の実践者などに見てもらい、その有効性をインタビューを通して検討した。その結果、有効性が見えた一方、新たな課題も発見し、そのまま導入することには課題は残るものの、今後のプログラミング教育普及へのプロトタイプとしての価値はあると考えられる。本論文は、プログラミングが学校教育として普及することを目的とする。

キーワード：

プログラミング, 教育 ,IT人材 , ワークショップ ,Hour of Code

慶應義塾大学大学院 メディアデザイン研究科

成富　紘

Abstract of Master's Thesis of Academic Year 2015

# Design of Beginner's Guide for Teachers to Teach Programming as a School Education

Category: Design

## Summary

In recent years, there is a growing importance of programming education. It has been changed to compulsory education centering on Europe. On the other hand, Japan is still considering plan. In this thesis, I first participated in existing programming workshop and got interview with experienced person and teacher. Then I made problems cleared. As a result, it can be focused on 3 problems, shortage of people, shortage of time and shortage of money. I made a hypothesis that Hour of Cord is the best way to solve them, and checked the effectiveness through observation in workshops. After that, I design beginner's guide for teachers to have a lecture using Hour of Code. As a result, I found a good point and new issue through interview to verify its effectiveness by teachers of elementary school and high school. This thesis aims at spreading programming as a school education.

Keywords:

Programming, Education, IT Human Resources, Workshop, Hour of Code

Graduate School of Media Design, Keio University

Ko Naritomi

# 目　　次

# 図　目　次

# 第1章
# 序　　論

## 1.1. 研究の社会的背景

近年、技術の発展やデジタルデバイスの普及に伴い、世界的にプログラミング教育に注目が集まっている。アメリカではオバマ大統領がプログラミングを学ぶよう呼びかけるビデオ[1]を発表して話題となり、ヨーロッパなどでは学校でプログラミングが必修化された [1]。日本においては、国家戦略という面からも IT 人材の育成は重要視されている。平成 25 年 6 月に成長戦略の柱として「世界最先端 IT 国家創造宣言」が閣議決定され、国家の成長エンジンとして情報通信技術 (IT) を位置付け、日本の「閉塞状況の打破」や「持続的な成長と発展」を目指すとされた。

## 1.2. 研究の個人的背景

この研究に取り組もうと考えた背景には、こうした社会的な期待の高まりだけではなく、私自身の体験も大きく含まれている。幼稚園のころから今に至るまでテレビゲームが大好きである私は中学生の頃にゲームを自身で作りたいと考え、プログラミングを勉強し始めた。しかし、その際に選んだのが C 言語というあまりゲーム開発には向かない言語であったことや、プログラミング入門として始めた、ゲーム開発とは程遠い計算プログラムを書き写し続ける学習方法にうんざりしてやめてしまった。そして本研究科に入学した後に、Scratch などの初心者向け

1　https://www.youtube.com/watch?v=6XvmhE1J9PY 最終閲覧：2016 年 1 月 8 日

プログラミング言語 (4 章で後述) に触れたときは衝撃だった。パズルを組むようにプログラミングができ、かつ、それですぐに遊ぶことができるという環境は、中学時代の挫折と情熱を思い出させた。こうした経験からこの研究には、子ども達の可能性を広げる手段として、プログラミングを身につけていってほしいという思いを込めている。

## 1.3. 研究の目的

本論文の目標は、プログラミングを学校教育として普及させることで、すべての子どもたちにプログラミング教育の機会を提供することである。そのために 2 章では現状分析を行い、小学校に対象を絞った。小学校への導入には教員の知識不足・準備の負荷・機材不足・授業時間不足など様々なハードルが考えられるため、本論文では、教員に関わる問題に焦点を当て、その解決を目指す。本研究の具体的な最終目標は、プログラミングについて事前知識の無かった教師が、子どもに教えられるだけの基礎知識を得たうえで、授業に臨めるようにすることである。

## 1.4. プログラミング教育の意義

本節では、プログラミング教育を行う意義について述べていく。構成としては、まず子どもがプログラミングを学習することで養われる能力について総務省による文献・ヒアリング調査の報告書をもとに述べる。次に、日本の国家戦略から見た ICT 人材育成の重要性を国の方針をもとに述べる。

### 1.4.1 プログラミング学習で向上する能力

前述の総務省の報告書 [2] によると、プログラミングを学ぶことで向上する能力は以下のようにまとめられる。表の左側は、学説や有識者によって示された能力について記述してあり、右側はプログラミング教育事業者へのヒアリング調査の結果を、左側の能力に対応させてまとめてある。

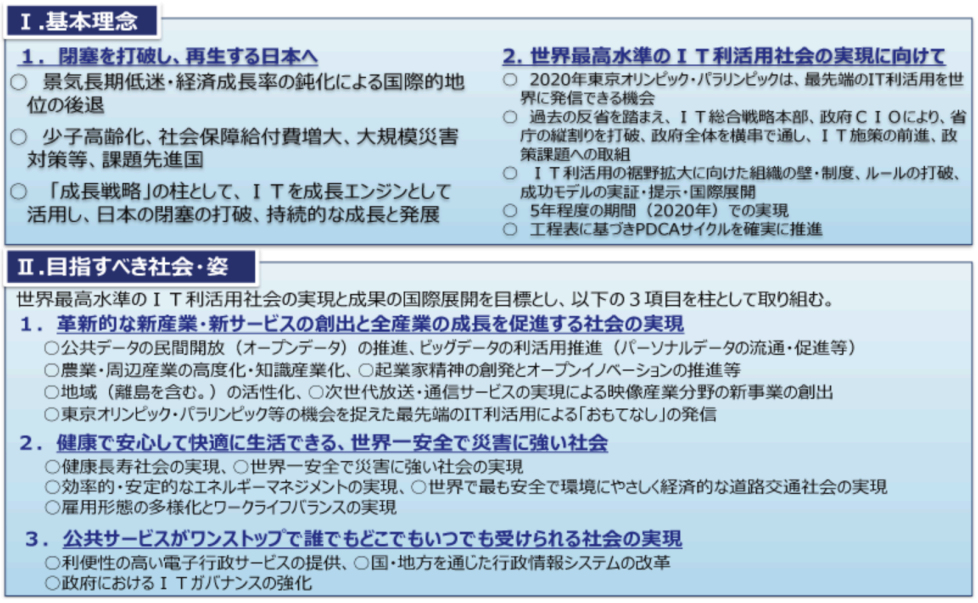
| 学説または有識者の意見 | 事業者の感じる効果 |
|---|---|
| 創造力[1]の向上 | ・プログラミングでは自由に作品を制作する課題を与えることが多いため、子供たちの作品制作を通じて、作りたいものを実現するという創造力が向上する。<br>・ものを作り出す面白さを実感させることで創造性が伸びる。 |
| 課題解決力[2]の向上 | ・プログラミングでは、結果がすぐにわかり、改善点が明確であることなどから、トライアンドエラーを繰り返しやすく、課題発見力、解決力が身につく。<br>・プログラミングを完成させるという目的達成のために前に進む主体的な行動力が身につき、完成させ、最後までやりきる力が身につく。 |
| 表現力[3]の向上 | ・プログラミングの過程で、プログラムの構想を書いたり、受講者同士で教えあい、学びあいをしたり、作品を発表したりすることによって、表現力が向上する。 |
| 合理性、論理的思考力[4]の向上 | ・フローチャートやプログラムの構想の作成とそれらに基づきプログラミングするという過程で、俯瞰的に考えたり、順序立てて考えたり、仕組みを考えるなどの合理的、論理的な思考が必要となるため、論理的な思考力が向上する。 |
| 意欲の向上（内発的な動機づけ効果）[5] | ・プログラミングという活動自体が子供たちにとって楽しいものであるため、子供は積極的に活動に取り組む。<br>・デバッグやトライアンドエラーを繰り返して作品を完成させるというプログラミングの作業特性から、子供たちがプログラミングの様々なタイミングで気づきを得るため、子供たちの忍耐力、集中力が持続し、学習意欲が維持、向上される。 |
| コーディング・プログラミングスキル[6]の向上 | ・テキスト型言語を用いて高度なプログラミングを行ったり、大人と同様の習熟を見せるなど、子供であってもプログラミングスキルが向上する。 |
| コンピュータの原理に関する理解[7] | ・プログラミングの過程で、不明点を調べたり、既存のライブラリを利用するなどの作業を行う際にインターネットによる情報検索を行うため、情報検索能力の向上など、情報利活用能力が向上する。<br>・プログラミングの活動を通じて、コンピュータの性質・原理に対する理解が深まる。 |

（プログラミング人材育成の在り方に関する調査研究報告書　p40より引用）

図 1.1: プログラミング学習で向上する能力

### 1.4.2 ICT 人材育成に対する社会的期待

日本における国家戦略という面からも、IT 人材の育成は重要視されている。平成 25 年 6 月に成長戦略の柱として「世界最先端 IT 国家創造宣言」が閣議決定され、国家の成長エンジンとして情報通信技術 (IT) を位置付け、図 1.2 の基本理念にも記載されているように、日本の「閉塞状況の打破」や「持続的な成長と発展」を目指すことになった。




平成26年度版情報通信白書[8]より引用

図 1.2: 世界最先端 IT 国家創造宣言

“ この方針においての「IT 人材」とは、IT 技術者のみならず、IT 社会を構成するすべての国民を対象して用いる ”とされているため、プログラミング学習者が増加することは、日本の経済成長にも寄与すると言える。

## 1.5. 論文の構成

本論文は以下の構成で成り立っている。まず本章にて、研究に至る背景や研究の目的、プログラミング教育の定義を述べた。２章では、プログラミングを広く

普及させるために働きかけるべき主体を分析した結果として小学校に絞り、課題点を整理したうえでデザインの方針を立てた。３章ではデザインを行ううえで関連している事例について述べた。４章ではデザインに適したツールを選ぶために行った予備調査の結果をまとめたうえで、デザインした入門ガイドについて述べた。５章ではその評価として、若い世代やPCに不慣れな人などに行ったユーザーテストと改善点をまとめた。最後に６章では評価のまとめと、今後の展望について述べる。

# 第2章
# 現状分析を通した方針設定

## 2.1. 研究対象の絞り込み

すべての子どもにプログラミング教育の機会を提供するうえで、研究対象を絞り込むためにまず国内におけるプログラミング教育主体の現状を整理した。

総務省の作成した「プログラミング人材育成のあり方に関する調査研究報告書」[2] によると、日本におけるプログラミング教育事業者の経営形態を割合で表すと図 2.1 のようになる。また大きく営利団体と非営利団体に分けて、その特性を図 2.2 のように整理している。これに加え、近年は学校でのプログラミング教育の実証実験も進められている。本節では、営利・非営利団体の活動内容について具体例をとって述べてから、学校における導入の現状について述べる。そして本論文で対象とする主体の絞り込みを行う。

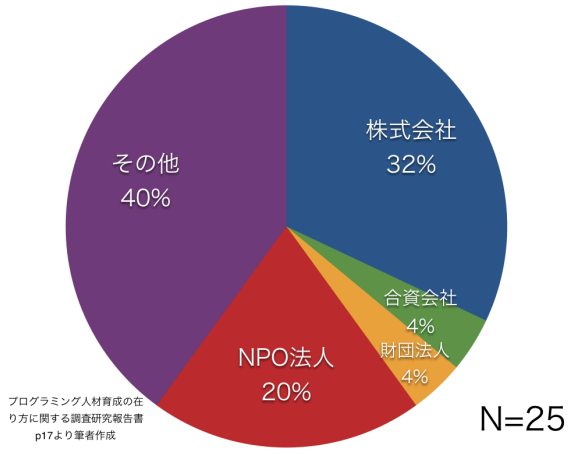



図 2.1: 教育関係団体の経営形態

| 教育関連団体の分類 | | | 主な教育関係団体 |
|---|---|---|---|
| **営利** | プログラミング教育専業事業者 | | CA Tech Kids、Life is Tech!、ロボット科学教育 Crefus、TENTO 等 |
| | ICT企業・教育ICT企業 | | IDEA Makers、NEL&M 等 |
| | 幼児教室・学習塾 | | Qremo、にいがたプログラミング塾、ロボ団、ラーニングセンター新浦安等 |
| | パソコンスクール | | 札幌プログラミングスクール、ThinkDo、パソコン教室 PLADY、科学技術館パソコン教室等 |
| **非営利** | NPO、法人化していない組織・個人 | プログラミング教育の提供を主目的とした NPO・組織 | CoderDojo、Ruby プログラミング少年団、Little Coder Mie、プログラミングクラブネットワーク(PCN)等 |
| | | こどもの教育、子育て を主目的としたNPO・組織 | CANVAS、子ども文化コミュニティ、新座子育てネットワーク、みどりっ子クラブ等 |
| | | モノづくりを主目的とした NPO・組織 | Maker Lab Nagoya、新発田科学技術教育ネットワーク等 |
| ※プログラミング人材育成の在り方に関する調査研究報告書 p17,18より筆者作成 | ICT企業 | CSR活動 | SCSK、日本 HP、NTT データ等 |
| | 大学・高専 | 公開講座等、地域・社会貢献の一環 | 東京大学、東京工科大学、埼玉大学、久留米工業大学等 |

図 2.2: プログラミングに関する教育を提供する教育関連団体の分類

### 2.1.1 営利団体

営利団体は、プログラミング教室やパソコン教室を経営する主体である。ここでは、小学生向けのプログラミングスクールと、中学生高校生向けのプログラミングスクールの２つについて、その教え方の特徴を記述する。

#### Life is Tech

株式会社ライフイズテックは「日本の IT 業界にイチロー並みの人材を送り出す!」を目標に、Life is Tech!というプログラミング教育事業を手がけている株式会社である。中学生・高校生を対象に、アプリやゲーム開発などが合宿を通して学べる ”キャンプ ”や通学型の ”スクール ”、ビデオ通話を使って遠隔地でも受講できる ”オンライン ”の 3 つのサービスを提供している。(この詳細については３章で触れる。) その一方で、自治体や企業、教育機関ともコラボレーションしながら、プログラミング関連のイベントや、新たなサービスを生みだしている。

#### CA Tech Kids

株式会社 CA Tech Kids は、株式会社サイバーエージェントの子会社であり、前述の株式会社ライフイズテックと運営協力をとり、小学生を対象としたプログラミング教室や短期集中型のキャンプ、教育機関へのプログラミング教育支援活動などを行っている [1]。教育の特徴としては、受講生一人一人に秘伝の書と呼ばれるテキストを配り、それを参考にしながら、わからないところをチューターに質問するという形式をとっていることである。

### 2.1.2 非営利団体

日本におけるプログラミング教育の実施主体として大きな割合を占めるのが非営利組織である [2]。ここでは、教え方としてワークショップを主にした団体と、学校外のプログラミング教室を主にした団体について述べる。

### NPO 法人 CANVAS

NPO 法人 CANVAS は、2002 年に石戸奈々子准教授によって設立された特定非営利活動法人であり、設立から現在にいたるまでに、30 万人以上の子どもに対して創造力を育むワークショップを開催してきた。プロジェクトの分野は多岐に渡っており、そのプロジェクトのひとつである PEG(ProgrammingEducationGathering) は、「すべての子どもたちに、豊かで継続的な」プログラミング学習の普及を目的としており、独自に子ども向けのプログラミングワークショップを開催するだけでなく、教育機関への研修の実施や、他団体と連携してワークショッププログラムや授業案の集約・公開も行っている。また、昨年には Google と連携し、小型コンピューター「ラズベリーパイ」5000 個を全国の教育機関に提供する事業が行われた

### Coder Dojo

Coder Dojo(コーダー道場) は、毎月の月謝を払って定期的に通うプログラミングスクールとは異なり、プログラミング教室をボランティアで運営し、子どもたちに無料で学べる場を提供する、プログラミング教育の支援活動である。この活動は 2011 年にアイルランドにおいて高校生のジェームズウェルトン氏が、地域の子ども達を相手にプログラミングを教える課外活動からスタートしたが、その後瞬く間に周りの大人や企業を巻き込み、ついにはアイルランド政府の支援をうけるまでに発展したムーブメントである。現在では、世界 57 カ国で 675 以上の認可団体 (2015 年 12 月現在) が存在している。日本においても Coder Dojo の理念に賛同し、東京・下北沢において Coder Dojo Tokyo がスタートした。その後 Coder Dojo Tokyo の活動が広がり、同じく賛同した人が Coder Dojo と名のつくプログラミング教室を全国各地でスタートさせ、現在では全国約 30 個所に広がっている。Coder Dojo Tokyo の場合、具体的な活動としては、毎週日曜日の午後に子どもが自分のコンピューターを持っていけば、現役のプログラマーやウェブデザイナー、エンジニアなど IT 関係の仕事に携わるプロの大人がプログラミングを教えてくれる。この活動を立ち上げた一人である河村奨氏は、”自分の作りたいものを

作りながら、カジュアルなコミュニケーションも楽しむ、まるで、プログラミングサークルのような雰囲気 ”と話している [1]。

### 2.1.3 学校

日本においてプログラミング教育が注目され始めた背景には 2014 年に安倍政権が発表した IT 戦略「世界最先端 IT 国家想像宣言」の改訂版の中で、初等・中等教育段階からのプログラミング教育や情報セキュリティなどの教育を推進すると言及したことが挙げられる。政府がこのようにプログラミングに関する具体的な発言をしたのには、日本も世界各国と同じように深刻な IT 人材不足の問題を抱えており、それを危惧した産業界から人材育成の要請が高まったためである。楽天の三木谷浩史氏は、2013 年 4 月に開かれた第 6 会産業競争力会議の席で、「エンジニアの質・量ともに大幅にレベルアップさせる必要が有る」と述べ、さらには、義務教育課程の IT 教育についても「例えば MIT が開発したスクラッチのような、楽しみながらプログラミングをマスターできるものがあり、こういったものはコストがかからないので、ぜひ導入してほしい」とかなり具体的に訴えかけた [1]。このような産業界からの働きかけによって、政府が IT 戦略の中にプログラミング教育を組み込んだことが、関心を高めるきっかけになったと言える。

#### 小学校

日本の公立小学校では、プログラミングの授業は必修化されていない。一方で、一部の学校では「総合的な学習の時間」という授業でプログラミング扱うことや、クラブ活動や放課後、土曜日の午後といった授業以外の時間でプログラミングが取り入れられるといった実践事例も出てきた。具体例としては以下の佐賀県武雄市では、東洋大学と DeNA が提携して、小学 1 年生を対象にプログラミングの実証研究をしたり、東京都多摩市立愛和小学校も、試験的な取り組みとして 2014 年秋に全学年を通してプログラミングの公開授業を開催した [1]。

#### 中学校

中学校においては、技術・家庭の「プログラムによる計測・制御」という単元で扱われている。この単元は、従来は選択項目であったが、平成 24 年度に施行された新学習指導要領で必修項目となり、中学生全員がプログラミングを学ぶこととなった。しかし実際には、プログラミング以外にも学ぶ内容が多いため、プログラミングを取り入れた同単元に割かれる時間は、中学 3 年間を通して５～６時間にとどまっている [1]。

#### 高等学校

高等学校の指導要綱では情報の授業において、「社会と情報」と「情報の科学」の２つの情報科目が定められている。このうち学校側がひとつを選択するが、その際に情報の科学を選ぶことでプログラミングが必修項目となる。

## 2.2. 本論文の対象

本論文においては、学校教育を対象とする。理由としては、営利・非営利団体のワークショップや教室は数が限られていることや、そこに参加するのは教育意識の高い家庭や月謝を払える能力のある家庭に限られ、すべての子どもに学習機会を提供することは難しいためである。一方で学校教育では、より均等な機会を提供することが可能になると考えた。

また、その中でも小学校を対象として絞り込む。理由としては、プログラミング学習を早いうちから始めることで、より発展的な内容に移行する場合、そのタイミングを早めることができ、子どもの可能性を広げることになるためである。また、プログラミングに必要な論理的思考力が身につくタイミングとしても適しているためである。発達論の研究者 J. ピアジェは、11,12 歳以上から、抽象的一般的な形で論理形式的に考えることができるようになる形式的操作期 (operational

stage) になるとしている[1]。また総務省によるプログラミングに関わる教育実施者へのヒアリング調査[2] [2] においても、

- IF-THEN 型の論理構成は 9 歳 10 歳まで理解できない。また、論理的思考は、11 歳以降でなければ理解するのは難しい。
- 論理的な考え方、抽象的な能力が限られているので、それが備わってからやるべきである。Scratch の主なターゲットは 8 歳以上である。
- フローチャートの学習は小学校 5 年生以上になってからでないと難しい。よほど優秀な場合は小学校 4 年生でもできる場合もあるが、小学校 3 年生以下にはまず不可能である。

とされている。よって本論文では、小学校でも高学年を対象とした教育に焦点を当てる。なお、以降ではプログラミング教育の導入を目的とするため、対象は子どもではなく、実施主体である教師と学校とする。

## 2.3. 小学校における課題点の整理

小学校にプログラミング教育を導入するには、いくつかの課題点をクリアしなくてはならない。本節ではその課題点を整理し、本論文で解決を目指す課題点を絞り込んでいく。

前述の NPO 法人 CANVAS の PEG プロジェクトのメンバーの方や、小学校教師の方にヒアリングを行い、以下のように課題点を整理した。

教師側：

---

1 発達論の研究者　J. ピアジェ　(Jean　Piaget)(2016　年　2　月閲覧) http://hermes321.com/developmental-theory/reseacher/jean-piaget/

2 ”プログラミングに関する教育の現状・実態の調査で実施した教育関係団体 16 社に加え、有識者 3 名に、プログラミングに関する教育がもたらす効果や、青少年の発達段階に応じたプログラミングに関する教育にふさわしいスキルとそのレベルについて、ヒアリング調査を実施した。”,「プログラミング人材育成の在り方に関する調査研究」報告書,　p4

- 教師のプログラミングへの知識不足
- カリキュラムが無い
- 多忙のため準備の負荷が大きい

学校側：

- 機材不足
- 他の授業との時間的兼ね合い

これらの課題のなかで、“カリキュラムが無いこと” に着目した。ここで述べているカリキュラムとは図 2.3 のような構造となっている。

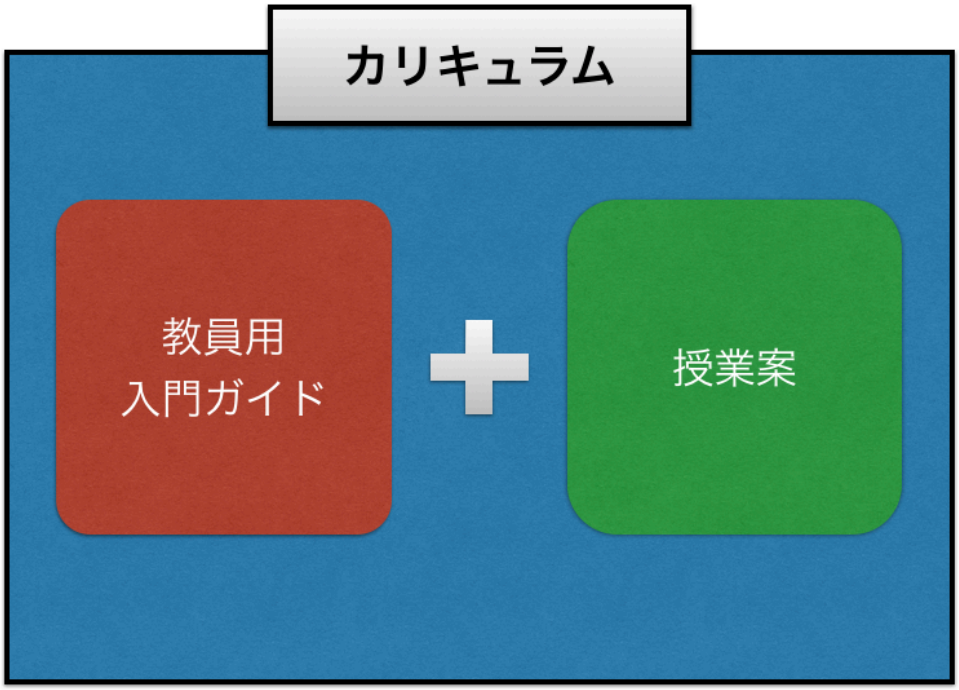



図 2.3: カリキュラムの概念図

授業案は、何をつかってどのように教えるかといった具体的な授業内容や進行方法について記載したものである。一方、教員用入門ガイド (以下、入門ガイド)

は、教師に対して、これから子どもに何を教えるのか、という授業を実施するにあたって身につけるべき事前知識について記載したものである。これらを踏まえ、カリキュラムが無いことに着目した理由は大きく２つある。まず１つに、プログラミングへの知識不足と準備の負荷は、カリキュラムが無いことによる影響が大きい。よって、これらの課題はカリキュラムを作成することで解決が可能であると考える。２つ目として、学校側の課題を考えるに、現状ではプログラミング教育に関するカリキュラムが整っておらず、学校側での機材不足や時間的兼ね合い等を検討する段階に至っていない。

本論文で具体的にデザインを行う対象として、入門ガイドに焦点を絞った。なぜなら、授業案は現場の状況や他の授業との時間調整で変わっていくと考えられるが、教師にどう教えるかという点は扱う内容 (授業案) が変わっても、構成や教え方は一定であるためである。よってこの部分をデザインして評価を受けることは、ノウハウの蓄積につながり、将来的にプログラミング教育を普及させてる上で重要となる。以上の考えから、本論文では特に入門ガイドをデザインすることとする。

# 第3章
# 先 行 研 究

前章では、デザインの方針として “教師が授業を行うために必要な知識を得られる入門ガイドの作成” を挙げた。本章ではそれを作成する上で参考となる “学校におけるプログラミング教育事例” と “基礎知識と習得する方法” を先行研究として整理した。

## 3.1. 国内外におけるプログラミング教育の実践例

### 3.1.1 イギリスの実践例

イギリスにおいては、初等教育段階（日本における小学校）におけるプログラミング教育が必修となっている。文部科学省が公開した「諸外国におけるプログラミング教育に関する調査研究 [3]」によると、以下の通りである。

イギリスにおいては2014年9月にナショナルカリキュラムが改定され、プログラミングが必修となった。具体的には従来の教科「ICT」に代わって「Computing」と呼ばれる科目がカリキュラムに新設され、アルゴリズム、プログラムのデバッグ、デジタルコンテンツの作成や編集、個人情報や安全な使い方を学ぶ情報モラルなど、コンピューターに関する幅広い内容を網羅し、その中の1つにプログラミングが位置付けられている。このカリキュラムは図3.4に示したように、義務教育課程の11年間をキーステージと呼ばれる4つのステージに分けて、それぞれのステージで学ぶ内容が決められており、プログラミングに関しては、14歳までに2つ以上のプログラミング言語を学ぶ。

| キーステージ | 学習内容 |
|---|---|
| 1(小学校1年〜2年) | . アルゴリズムの理解や実装の課程、プログラムが正確な命令を実行することを知る<br>. 簡単なプログラムの作成、およびデバッグ<br>. 簡単なプログラムの動きを論理的に推測する<br>. 情報技術を用いたデジタルコンテンツの作成、保存、整理<br>. 学校外での情報技術の用途を知る<br>. 個人情報を公開せず、オンラインのコンテンツは大人の保護のもと扱う |
| 2(小学校3年〜6年) | 具体的な目的を達成するプログラムの設計・実装・デバッグ<br>人事処理、条件分岐、繰り返しをプログラムで実現、変数やさまざまな入出力を扱う<br>簡単なアルゴリズムを理論的に説明し、実装したプログラム内のエラーを認識・解決できる<br>コンピューターネットワークの仕組みを理解し、実社会でどのように利用されているかを知る<br>検索技術や検索結果を学び、よりデジタルコンテンツの評価を適切にできるようになる<br>さまざまなソフトウェアを使って多様なデジタル課題を解決できるようになる<br>情報技術の安全な利用を理解する。不適切な行動を認識し、報告ができるようになる |
| 3(中学校1年〜3年) | 現実の問題や物理現象の動き<br>状態を抽象化したモデルの設計・利用・評価<br>並び替えや検索など主要なアルゴズムの理解。複数のアルゴリズムの比較<br>２つ以上のプログラミング言語の理解(最低１つはテキストベース)<br>簡単な論理演算とこれらがカイロやプログラムにどのように使用されているかの理解<br>コンピューターがハードウェアとソフトウェアから成り立っていることの理解<br>命令がどのようにコンピューター上に蓄えられ、実行されるかの理解<br>複数のアプリケーションを使って、課題解決を目標としたプロジェクトを実施する<br>信頼性や操作性など使う人のことを考えてデジタルコンテンツを作成・再利用・編集できる<br>個人情報やプライバシーの保護を意識し、情報技術を安全かつ責任を持って利用出来る |
| 4(高校1年〜2年) | すべての生徒が情報技術やコンピューターサイエンスを多様に学べる環境を与える<br>コンピューターサイエンス、デジタルメディア、情報技術に関する技能と創造性、知識の強化<br>分析能力や、問題解決能力、設計能力、論理思考を成長させ、問題解決に適用できる |

※[1]より筆者作成

図 3.1: イギリスにおける義務教育における教科「Computing」のカリキュラム

### 3.1.2 ハンガリーの実践例

2003 年にネット検索やペインティングなどの IT 利用授業を小学 1-4 年生において開始し、現在では「Informatika」を 1-12 年生 [1] で連続して教えている。
「Informatika」は①IT ツールの利用法、②アプリケーションの知識、③問題解決のツールとテクニックとしての IT、④21 世紀におけるインフォコミュニケーション、⑤情報社会、⑥図書館情報学、という 6 つの分野から構成され、それぞれの内容や授業数は学年により異なっている。①においてキーボード、モニターといった周辺機器及び、ファイル、フォルダをも含めたコンピュータに触れていく上で基礎となる概念を習う。②においては OS、ソフトウェア、アプリケーション等の知識とそれらを用いて各種ドキュメントを作成する。③においては②のもととなるアルゴリズム、モデル化、データの取り扱い、プログラミング等を学び、各種問題解決にあたる。④はインターネット及び各種サービス、⑤においてはセキュリティや著作権等の社会との接点、⑥で図書館関連の知識を学ぶ。
プログラミングに関しては特に③の「問題解決のツールとテクニックとしての IT」の授業において、論理的思考、アルゴリズム化、基本的な一連の手続き及び制御構造を学び、実際にコンピュータプログラムを作成しテストする。また、異なる分野の問題や現象をプログラムを用いて学びシミュレーションし、アルゴリズムを理解してプログラムを作成する、及びアプローチ手法の開発において、問題解決を学ぶ。学年別には 1-4 年生では簡単なアルゴリズムを習得し、5-6 年生では簡単なプログラムの実装、検証、7-8 年生ではステップバイステップの計画手順、9 年生以降では改良の原理まで学ぶ。

### 3.1.3 ロシアの実践例

ロシアでは、プログラミング関連の授業を 2-11 年生で一貫して教え、必修と定めている。初等教育においては、2009 年より (他の関連教科の一部としての)「インフォルマティカと ICT」の中でアルゴリズム教育を実施している。算術演算を

1　ハンガリーでは 1-4 年生が初等教育、5-8 年生が前期中等教育、9-12 年生が後期中等教育にあたる

数字と数式にて表現し、問題を解決する能力や簡単なアルゴリズムを構築する力を付けさせている。また、幾何学的な図形を特定し描き、表、チャート、グラフや図表、順序集合を用いたり、データを集計、分析し解釈する力を付けさせ、プログラミング教育とつなげている。
中等教育におけるプログラミング教育は 2010 年より導入されており、独立教科「インフォルマティカと ICT」の中で指導されている。現代社会のプロフェッショナルな仕事のために必要なアルゴリズム的思考の開発、アートやデザインのためのアルゴリズムを作成するための技能を身につけさせるとしている。また、アルゴリズムの値、論理的操作に関する知識、主要なプログラミング言語やアルゴリズムの構造の一つ (順次、条件分岐、繰り返し) に精通させるとされている。なお教科書に関しては、関連する DVD 等をも含め、出版社が発行するものを国が認定し、その中から地域・学校ごとに選択する仕組みとなっている。2014 年 12 月には、教育科学省及び情報技術・通信省と連携し、「Hour of Code」が実施されている。

### 3.1.4 日本の実践例

日本においてもいくつかの学校においてプログラミング教育の実践が行われている。文部科学省が作成した「プログラミング教育実践ガイド」[3] では、以下の小学校における授業が紹介されている。

- １年生からのプログラミング体験（１年生：生活科、特別活動）
- めざせ！行列のできるおすし屋さん！（４年生：図画工作科）
- プログラミングロボット学習（４・５・６年生：総合的な学習の時間）
- 調べた人物をプログラムで表現してみよう（６年生：総合的な学習の時間）

この中から「プログラミングロボット学習」と「調べた人物をプログラムで表現してみよう」の内容を、「プログラミング教育実践ガイド」から抜粋する (図 3.2, 3.3)。この２つを選んだ理由は、生徒の年齢層が論文の対象である高学年であるためである。


図 3.2: 大阪市立苗代小学校の実践例

## 調べた人物をプログラムで表現してみよう

宮城県仙台市立大野田小学校，授業者：栄利 滋人

校種：小学校，　実施教科：総合的な学習の時間
実施単元：人物ヒストリー，　実施学年：6 年生

### 使用したプログラミング言語や実行環境

- Scratch（Ver2.0）。
- Scratch はブラウザ（http://scratch.mit.edu/）上で稼働。Scratch の ID を児童 1 人 1 人に割り当て，作品を保存・共有できるようにしている。パスワードは教員が管理。

### 教室の設備・環境

- **実施場所：**コンピュータ室と一般教室
  - コンピュータ室には，Windows7 マシンが 40 台あり，各児童が 1 人 1 台ずつ使用。インターネットは仙台市教育委員会がフィルタリングソフトで有害なサイトをブロックする環境で使用。
- **実施前の準備**
  - 児童用の Scratch ID を全員分登録し，ログインや保存が可能かどうかを確認。
  - 授業の前に児童のログイン画面でパスワードを入力し，1 人 1 人が Scratch を使えるように立ち上げておく。

### 授業の様子

導入ではボストンの小学生の作品を見せ，ブロックの命令で動く仕組みを知らせる。


Scratch の操作ガイドを参考にいろいろなブロック操作を試していく。青のブロックで歩かせ，紫のブロックでセリフを言わせるなど，色を指示しながら色分けされたブロックを操作させていく。

基本操作を学んだあと，自分でいろいろなブロックを操作して Scratch を試させる。

児童が図工で描いた作品の画像を中京大学に送る。絵を Scratch で動かすプロジェクトに参加する。

中京大学の学生が作成した Scratch で絵が動く様子を見せる。どのようなブロックの組み合わせで動いているかを理解させる。


学生のプログラムを参考にして，児童は調べた人物を Scratch で表現する。

図 3.3: 宮城県仙台市立大野田小学校の実践例

## 3.2. 基礎知識の習得方法

本節では、未経験の教師がプログラミングの授業をするための基礎知識を習得する方法の参考とするために、イギリスにおける教師向けサポートの取り組み・指導者向け講習会・パソコン初心者向け書籍・プログラミング学習サービスの４つについて調査を行った。

### 3.2.1 イギリスにおける教師へのサポート

イギリスにおいての教師へのサポートとしては、「QuickStart Computing ; A CPD toolkit for primary teachers」というナショナルカリキュラムを補足するガイドが提供されている。これは 2015 年に CAS[2] が発行した DVD 付きの CPD[3] 教材で、教科「Computing」の授業プランの立て方や教え方、プログラミング言語の解説、リソース等が示されている。オンラインでの提供の他、ハードコピーが 40,000 部、無償で配布されている。具体的な中身としては、図 3.4 のように、教師の疑問に答えるという形式で、図やスクリーンショットが多く用いられている。

2　もともとは Microsoft Research Cambridge の中に設立された有志のワーキンググループであったが、現在は学校における教科「Computing」の実施を推進する任意団体である。 教科「Computing」の実施にあたり、学校や教員をサポートすることを目的としている。CAS は、様々な研修プログラムを実施しており、このための資金として、教育省から 2 年間で約 300 万ポンドが提供された。

3　Continuing Professional Development の略であり、技術者の継続教育を意味する

# Programming

Introduction to programming

## What is programming?

Programming is the process of designing and writing a set of instructions (a program) for a computer in a language it can understand.

This can be really simple, such as the program to make a robot toy trace out a square; or it can be incredibly sophisticated, such as the software used to forecast the weather or to generate a set of ranked search results.

Programming is a two-step process.

- First, you need to analyse the problem or system and design a solution. This process will use logical reasoning, decomposition, abstraction and **generalisation** (see pages 6–17) to design algorithms to solve the problem or model the system.
- Secondly, you need to express these ideas in a particular programming language on a computer. This is called coding, and we can refer to the set of instructions that make up the program as 'code'.

Programming provides the motivation for learning computer science – there's a great sense of achievement when a computer does just what you ask it, because you've written the precise set of instructions necessary to make something happen. Programming also provides the opportunity to test out ideas and get immediate feedback on whether something works or not.

## *What should programming be like in schools?*

It's possible to teach **computational thinking** without coding and vice versa, but the two seem to work best hand-in-hand.

Teaching computational thinking without giving pupils the opportunity to try out their ideas as code on a computer is like teaching science without doing any experiments. Similarly, teaching coding without helping pupils to understand the underlying processes of computational thinking is like doing experiments in science without any attempt to teach pupils the principles which underpin them.

This is reflected in the new computing curriculum, which states that pupils should not only know the principles of information and computation, but should also be able to put this knowledge to use through programming. One of the aims of the national curriculum for computing is that pupils can analyse problems in computational terms, and have repeated practical experience of writing computer programs in order to solve problems.

In key stage 1, pupils should be taught *how* simple **algorithms** are implemented as programs on digital devices. The phrase 'digital devices' encompasses tablets, laptop computers, **programmable toys**, and perhaps also distant **web servers**. It can be useful for pupils to be able to see their algorithms, in whatever way they've recorded these, and their code side by side.

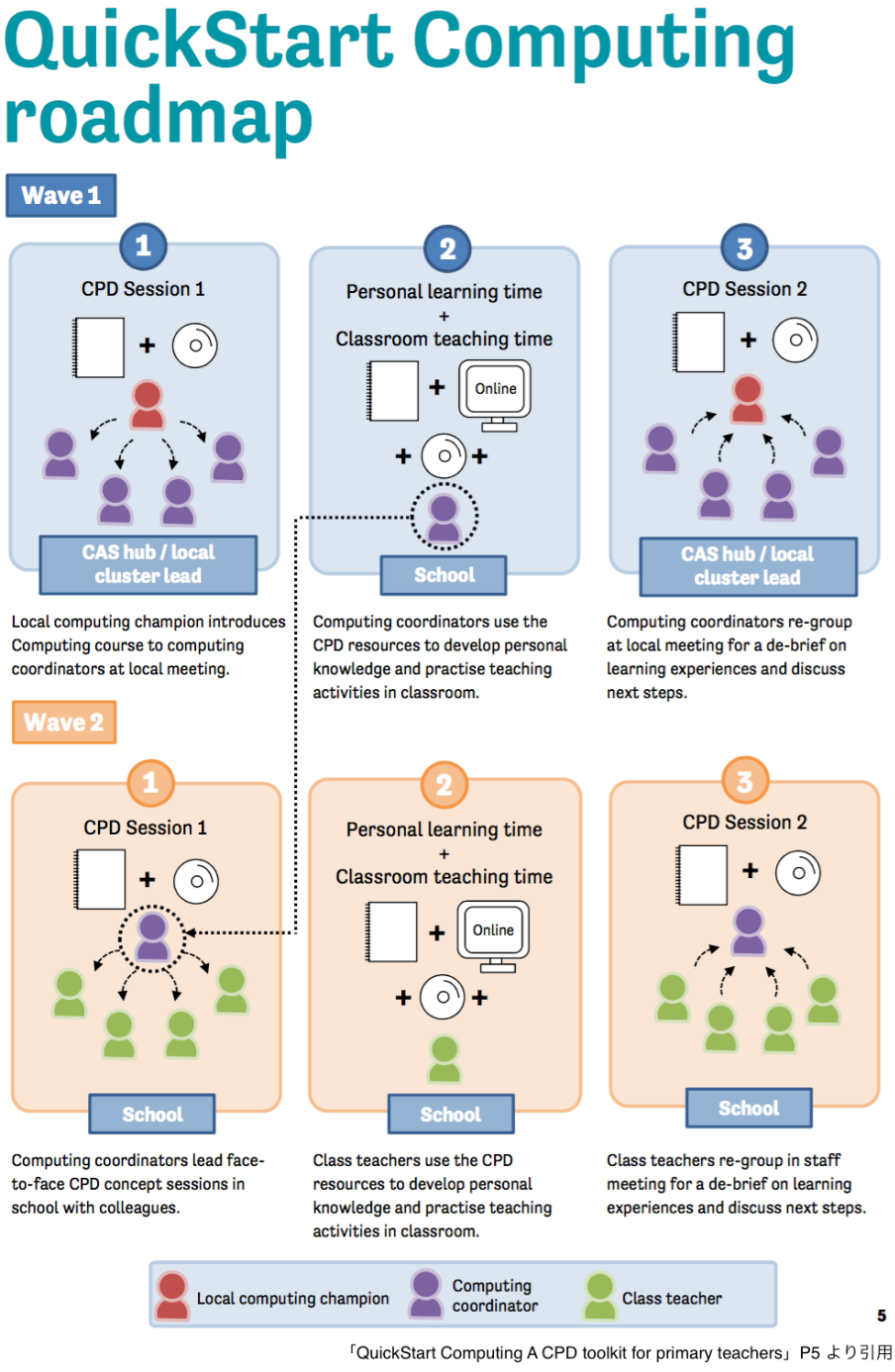
Children can use simple arrow cards to record algorithms for programmable toys.

18

「Quick Start Computing A CPD toolkit for primary techers」P18より引用

図 3.4: イギリスにおけるナショナルカリキュラム補足用のガイド

また、教師向けの研修も存在する。その取り組みの一つとして前述の「QuickStart Computing」を用いたCPD研修がある。内容としては、まず各地域で「Computing」教育を先導する教員である “Local computing champion” が各学校の「Computing」教員の代表である “Computing coordinator” に『QuickStart Computing;A CPD toolkit』の使い方を指導する。その後、“Computing coordinator” がツール・キットやオンラインリソースを活用して独自に学習した内容を、“Local computing champion” にフィードバックし、次のステップについて議論する。同様のことが、各学校において “Computing coordinator”-“Class Teacher” 間でも行われる。すなわち、“Computing coordinator” が学校内の “Class Teacher” に直接『QuickStart Computing;A CPD toolkit』について指導する。“Class Teacher” は独自に学習し、“Computing coordinator” にフィードバックし、次のステップについて議論する。この仕組みを図にしたものが以下の図3.5である。




「QuickStart Computing A CPD toolkit for primary teachers」P5 より引用

図 3.5: イギリスにおける指導者向け研修の仕組み

### 3.2.2 指導者向け講習会

NPO 法人 CANVAS の PEG チームでは、これまでにプログラミングの指導者向け研修会を多く行ってきた。先行研究の一環として、そのうちの一つである関東学院大学の教員向けの入門講座にスタッフとして参加した。この講座の内容は “教員が子どもにプログラミングワークショップを行うための準備として、CANVAS のスタッフが講師となり、ワークショップ内容の提案と、プログラミングツールの操作方法や指導方法の共有を行う” というものであった。ここで用いられたツールは Raspberry Pi という小型コンピューターと Scratch であり、これらに加えモニターとマウスがあれば学習を開始できる。そのため機材の準備が楽という特徴があった。具体的には以下のような内容で行われた。

＜開催概要＞

日時:2015 年 10 月 28 日 13:00-16:00(各 3 時間)
会場:関東学院大学金沢八景キャンパス 6 号館 504 号室
スタッフ:CANVAS の講師 2 名
参加人数:12 人
パソコン:１人につき１台
使用ツール：Raspberry Pi, Scratch
目的：体験を通したワークショップ内容の選択と指導方法の学習

＜流れ＞

会場にパソコンモニターや Raspberry Pi などの機材を会場に搬入し、事前に接続など起動準備を行う。また、子ども達に行うワークショップの流れを書いたマニュアル (図 3.6) を渡しておく。

①講師がワークショップ案の概要（狙いや難易度）を説明を行う
②参加者は自身の Raspberry Pi 上で、マニュアルに沿って組み立てながら、プログラミングの概念について解説を受ける。不明点は随時スタッフに質問する
③自由制作にも少し挑戦を行う

＜様子＞


図 3.6: 指導者向け研修会で用いたマニュアル

マニュアルに沿う部分については質問は出てこなかった。一方で、自由制作の段階では繰り返し処理の理解などにおいていくつか質問が見られた。こうしてプログラミングへの理解が深まるだけでなく、複雑なプログラムにすると Raspberry Pi の動作が遅くなることや、ファイルの保存場所など、細かな気づきを得ることができていた。

### 3.2.3 パソコン初心者向け書籍

パソコンの操作に慣れていない人を対象とした教え方を調べるために複数のパソコン初心者向け書籍 [4] [5] [6] を参照し、共通する特徴を整理した。参照した書籍は以下の 3 冊である。

- 「できるゼロからはじめるパソコン超入門　ウィンドウズ 10 対応」[4]
  著者：法林 岳之　ページ数：257 ページ
  パソコン初心者に向けて、基本的な操作方法から文書作成や画像の取り込み

などの発展的内容までを説明している。各項目は”文字を削除しよう”や”直前に表示していたWebページに移動しよう”といった具体的な目的ごとに記述されている。

- 「超入門　まったく分からない人のパソコン入門 Windows7」[6]
  著者：若松 和紀　ページ数：223ページ
  上述の書籍と教える内容はほぼ同じ (操作方法から文書作成など) だが、パソコンやメールの仕組みや何ができるのかという説明が多く書かれている。また、各項目は”入力や変換を間違ってしまったら”や”インターネットエクスプローラーの便利な機能”などの大きな括りとなっており、具体的なものはそのなかに列挙するという形式で記述されている。

- 「よく解る！世界一やさしい　超パソコン入門用語」[5]
  著者：パソコン用語研究会　ページ数：314ページ
  パソコンに関連する用語をアルファベット・５０音順に並べ、気になった言葉があった際に調べられる辞書のようなスタイルをとっている。各用語は５行程度にまとめられており、参考画像が添えられているものもある。用語の説明は全くの初心者には難しいものも多く、用語説明内にでてきた用語をさらに調べる必要がある場合も有る。

#### 教え方の構成

書籍の構成としては、まず初めに “パソコンとはなにか？” と “パソコンではなにができるのか？” といったパソコンの操作を学ぶ上での全体像を読者に示し、各操作説明に進んでいる。また、各項目 (例：アプリを起動しよう) においても、この単元で行うこと (例：ここではニュースのアプリを開いてみましょう) を述べたのち、各ステップ (例：スタートメニューのアイコンをクリック) を詳細に述べている。

#### 教え方の特徴

参照した書籍全てにおいて、操作を説明する際には文章のみではなく、パソコン画面のスクリーンショットを掲載し、そこに操作説明を書き込むという方法をとっている。また、操作説明も可能な限り細かくステップに分け、ひとつひとつの動作を説明するよう配慮が行われている。加えて、“ドラッグ＆ドロップ” を “手で持って動かすイメージ” など、直感的にわかりやすい言葉に言い換えた表現が多用されている。

### 3.2.4 プログラミング学習サービス

#### Codecademy

CodecademyとはHTMLやCSSなどマークアップ言語及びPython、PHP、Javascript、Rubyなどのプログラミング言語関連のコーディング講座を無料で提供しているオンラインプラットフォームである。2012年6月時点で5万人以上の利用者が1億件以上の演習を終えており、[4] ニューヨークタイムズ[5] やTechCrunch[6] など多くのブログやウェブサイトから高評価を得ている。 利用者の参加を促すため、課題完了時のバッジ付与や合計点数を公開する機能があるほか、利用者が新しい課目を作成し公開することができる。また、その進め方も特徴であり、カリキュラムは小さな課題を連続して解いていくことで進められ、図3.7のように画面左側

4 Indvik, Lauren. “ Codeacademy Releases Free Ruby Development Courses ”(URL：http://mashable.com/2012/10/04/codecademy-ruby-courses/).(2016年2月閲覧) Mashable. Mashable.

5 Wortham, Jenna. “ Codecademy Offers Free Coding Classes for Aspiring Entrepreneurs ”(URL：http://bits.blogs.nytimes.com/2011/09/14/codecademy-offers-free-coding-classes-for-aspiring-entrepreneurs/?_r=0).(2016年2月閲覧) The New York Times.

6 Cincaid, Jason. “ Codecademy Surges To 200,000 Users, 2.1 Million Lessons Completed In 72 Hours ”(URL：http://techcrunch.com/2011/08/22/codecademy-surges-to-200000-users-2-1-million-lessons-completed-in-72-hours/).(2016年2月閲覧) TechCrunch

に問題、中央側にコード記入画面、右側に結果画面が配置されており、自身で学習環境を構築する必要もなく、手軽に進めていくことができる。

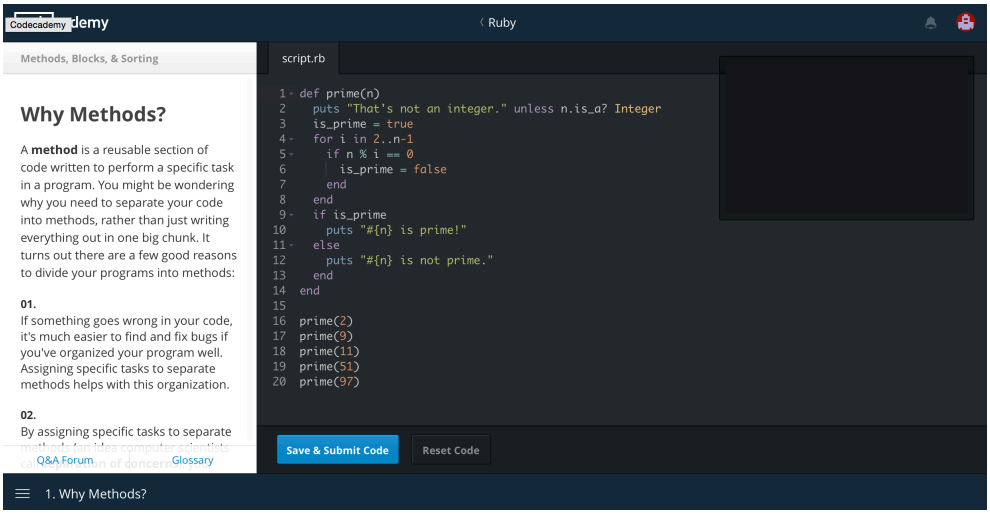



図 3.7: Codecademy のコース一例

### Life is Tech!

株式会社ライフイズテックは「日本の IT 業界にイチロー並みの人材を送り出す！」を目標に、Life is Tech!という国内最大規模でプログラミング教育事業を手がけている株式会社である。中学生・高校生を対象に、アプリやゲーム開発などが合宿を通して学べる “キャンプ” や通学型の “スクール”、ビデオ通話を使って遠隔地でも受講できる “オンライン” の３つのサービスを提供している。学習コースは多岐にわたっており、iPhone/Android アプリ開発・Web デザイン・Web プログラミング・映像編集などがある。

“キャンプ” では、図 3.8 [7] のように、５日間合宿のなかで、午前に３時間、午後に５時間を使って、プログラミングの学習と開発を行う。その他のアクティビティや受講生同士での交流も特徴のひとつである。

7　Life is Tech！ホームページから（URL：`https://life-is-tech.com/xmas2015/`）(2016 年 2 月閲覧)

<table>
<tr><th></th><th>23日（水・祝）</th><th>24日（木）</th><th>25日（金）</th><th>26日（土）</th><th>27日（日）</th></tr>
<tr><td>Morning<br>↓</td><td rowspan="3">集合＆移動＆昼食(バス内)<br>※昼食はこちらでご用意致します。<br>※現地集合の方は、昼食を済ませてからお越しください。</td><td>モーニング<br>ビュッフェ</td><td>モーニング<br>ビュッフェ</td><td>モーニング<br>ビュッフェ</td><td>モーニング<br>ビュッフェ</td></tr>
<tr><td>AM<br>9:00〜<br>12:00<br>↓</td><td>開発</td><td>開発</td><td>開発</td><td>開発</td></tr>
<tr><td>Lunch<br>↓</td><td>レストラン</td><td>レストラン</td><td>レストラン</td><td>レストラン</td></tr>
<tr><td rowspan="3">PM<br>13:00〜<br>18:00<br>↓</td><td rowspan="3">13:00〜14:00<br>オープニング<br>14:00〜18:00<br>開発</td><td rowspan="3">開発</td><td rowspan="3">開発</td><td rowspan="3">開発</td><td>発表会＆認定式</td></tr>
<tr><td>15:30　終了予定</td></tr>
<tr><td>移動&解散</td></tr>
<tr><td>Dinner<br>↓</td><td>ディナービュッフェ</td><td>ディナービュッフェ</td><td>クリスマスディナー</td><td>ディナービュッフェ</td><td></td></tr>
<tr><td>Night<br>Event</td><td>秘密の企画 1</td><td>秘密の企画 2</td><td>スペシャルクリスマス<br>パーティー第1部</td><td>スペシャルクリスマス<br>パーティー第2部</td><td></td></tr>
</table>

図 3.8: Life is Tech！キャンプスケジュール

“スクール” では、Life is Tech!の所有する教室へ定期的に通学することで、iPhone や Android アプリ開発・Web デザイン・ゲームプログラミング・Web サーバープログラミングのなかからひとつを選び、少人数指導を受けることができる。タイムスケジュールは以下の図 3.9)[8] の通りである。

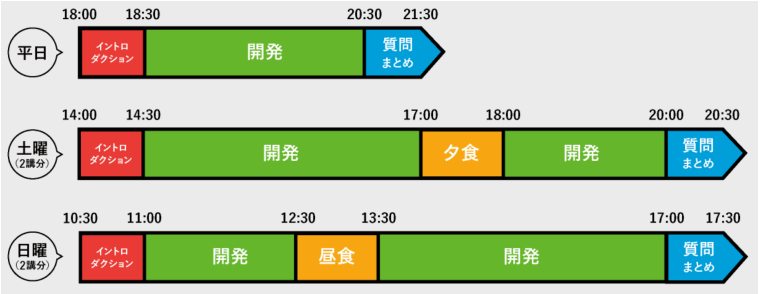



図 3.9: Life is Tech！スクールスケジュール

8　Life　is　Tech！ホームページから（URL：https://life-is-tech.com/school/class-schedule/）(2016 年 2 月閲覧)

“オンライン” では、生徒が自宅でプログラミング授業を受けられるようにインターネット電話サービスを用いている。具体的な流れとしては、事前に動画教材と教科書を見ながらアプリを開発し、週１回インターネット電話で、3〜5 人のクラスメイトと一緒に１時間、ポイントの復習、質問、応用講義などを行うというものである。この流れは反転学習と呼ばれるものであり、「授業で講義、家で復習」という通常のスタイルではなく、「家で事前講義、授業で復習まとめ」という反転した授業スタイルをとっている。

# 第4章
# 入門ガイドのデザイン

本章では前章までの内容を踏まえ、入門ガイドの具体的な内容の検討と実装を行う。デザインプロセスとしては、まずプログラミング教育で何を教えるか (学習目標) を定義したうえで、どのツールを用いるべきかをファシリテーターとして参加したワークショップへの考察を通して判断する。そしてそのツールを使った授業案を準備し、その授業案の事前知識となる入門ガイドを作成するという順序をとる。

## 4.1. 学習目標の定義

プログラミング教育を通じて子どもに学んでほしいこととして、“高度なプログラミングスキルを習得すること” ではなく、まず “プログラミングの概念を理解すること” がある。これは、今後プログラミング学習を進める場合、どんな言語であっても使える土台となる知識となる。また、プログラミング学習を進めない場合であっても、ロジカルに物事を考える経験を生かすことで、自分で考えることのできる人間になることができる。プログラミング教育の専門家である青山大学・津田塾大学 非常勤講師の阿部和広先生は、以下のように述べている [1]。

”(中略) プログラミングをなんのために学ぶのかについては、もう１つあります。それは、今のデジタルネイティブと呼ばれる子どもたちのスマートフォンやタブレット、コンピューターの使い方です。彼らは生まれながらにして、デジタル機器に囲まれ器用に使いこなしますが、実際何をしているのかを観察すると、単にゲームや動画コンテンツを消費しているだけだったりします。このような使い方

は、子どもたちの楽しみでもあるのですべてを否定はしませんが、そればかりではデジタル機器の使い方として受け身すぎるでしょう。つまり言い換えれば、このような状況は、子どもたちはデジタル機器の”読み”はできるけれども”書き”はできないということです。学校においても、ワードやエクセルの使い方は教えますが、それがどうやってできているのかは教えません。コンプガチャについても問題視はするけれども、それがどういう仕組みでできているかを教える機会はないし、ましてや、その仕組みを作るような体験はさせません。このようなアンバランスな状況を埋めるのが、プログラミング学習の役目だと考えています、コンピューターを操る経験は、物事の考え方や捉え方に変化を与えます。コンピューターを操った経験がなければ、周りのモノにたいして「便利だけど、なんだかよく分からないもの」で終わってしまいますが、プログラミングの経験があれば、どんなモノにも仕組みがあることや、それを変えることも可能だと知ることができます。そして「自分が理解できるもの」としてとらえることができ、物事を違う視点で考えたり、一段うえから見たりできるようになります。私は、プログラミング教育でデジタルネイティブをそのような意識に変えていくことができると思っています。”

また、イギリスで実施されている初等教育のカリキュラムでは、CS(Computer Science) を学ぶために小学校で、以下図 4.1 の学習内容を設定している [1]。

1 「National Curriculum in England: computing programmes of study」より筆者作成。アドレス：https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study (2016 年 2 月閲覧)

| キーステージ | 学習内容 |
| --- | --- |
| 1(小学校1年〜2年) | . アルゴリズムの理解や実装の課程、プログラムが正確な命令を実行することを知る<br>. 簡単なプログラムの作成、およびデバッグ<br>. 簡単なプログラムの動きを論理的に推測する<br>. 情報技術を用いたデジタルコンテンツの作成、保存、整理<br>. 学校外での情報技術の用途を知る<br>. 個人情報を公開せず、オンラインのコンテンツは大人の保護のもと扱う |
| 2(小学校3年〜6年) | 具体的な目的を達成するプログラムの設計・実装・デバッグ<br>人事処理、条件分岐、繰り返しをプログラムで実現、変数やさまざまな入出力を扱う<br>簡単なアルゴリズムを理論的に説明し、実装したプログラム内のエラーを認識・解決できる<br>コンピューターネットワークの仕組みを理解し、実社会でどのように利用されているかを知る<br>検索技術や検索結果を学び、よりデジタルコンテンツの評価を適切にできるようになる<br>さまざまなソフトウェアを使って多様なデジタル課題を解決できるようになる<br>情報技術の安全な利用を理解する。不適切な行動を認識し、報告ができるようになる |

図 4.1: イギリスのカリキュラムの一部

こうした事例からも、コンピューターの概念を理解する必要性は高いと社会的に認識されていることがわかる。

## 4.2. ツール候補の絞り込み

### 4.2.1 プログラミング言語の選択

本節では、前節で設定した学習目標を達成するために適したツールの選択を、観察を通して行う。この場合のツールとは、プログラミング言語とその開発環境を合わせた “教材” としての意味で用いる。

プログラミングツールの絞り込みを行う上で、まずプログラミング言語という観点から、図 4.2 のように “ビジュアルプログラミング言語” と”テキストプログラミング言語” に大別した。

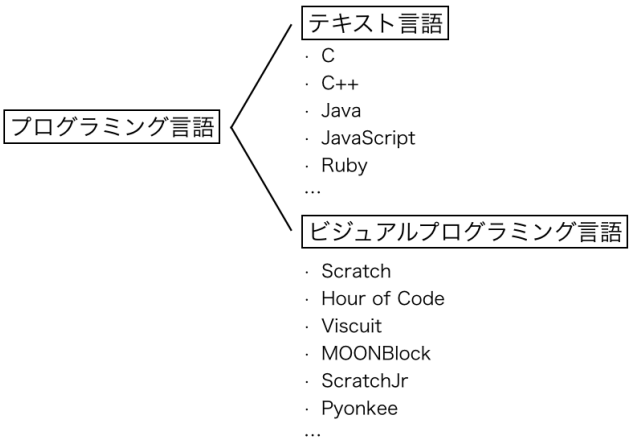



図 4.2: プログラミング言語の種類分け

テキスト言語とは、図 4.3 のように、プログラムをテキストで記述する言語である。一方でビジュアルプログラミング言語とは、視覚的なオブジェクトでプログラミングするプログラミング言語である。グラフィカルプログラミング言語とも言う。図 4.4 のように視覚表現でプログラミングが可能で、空間上でテキストやグラフィックシンボルを配置することでプログラムが形成される。

```
def write_info
student ={}
puts"生徒の名前を入力してください:"
  student[:name] = gets.chomp
puts"得点を入力してください:"
  student[:score] = gets.to_i

return student

end

def check_info(a_students)
  if a_students.length == 0
    puts "エラー：データがありません！"
  end

  sum = 0
  line = "----------"
  puts line
  a_students = a_students.sort{|a, b| b[:score] <=> a[:score]}
  a_students.each do |student|
    puts "#{student[:name]}:#{student[:score]}点"
  sum += student[:score]
  end
  puts "平均点：#{sum / a_students.length}"
  puts line
end
```

図 4.3: テキスト言語のイメージ (Ruby の場合)


図 4.4: ビジュアルプログラミング言語のイメージ (Scratch の場合)

プログラミングの概念を理解することが目的である場合、細かな文法を変えるという手間を省いた、グラフィックシンボルを移動させるだけのような直感的に学習を進められることが重要だと考えた。また、諸外国のプログラミング教育でも多く用いられていることから [7]、本論文では使用するプログラミング言語としては、ビジュアルプログラミング言語を選択する。

### 4.2.2 ツール選択における仮説設定

次にビジュアルプログラミング言語の中から適したツールを選択するために仮説を立て、その検証のための予備調査を行う。ツールを選択する基準は、前述した課題である “教師への負荷” が低いこととした。教師への負荷を高める要因のひとつとしては、ツールの自由度が考えられる。ここでの自由度とは、“ツールを扱う際に生徒の取れる選択肢の多さ” と定義する。そこで、以下の仮説と検証するポイントを立てた。

仮説：自由度の低いツールが入門ガイドには適切である。
検証するポイント：ツールの自由度の高さと教師への負荷の大きさは比例するか

この検証点の理由は以下である。自由度が高いツールであるほど子どもが何でも制作できるというメリットがある一方、教師にとっては教える手間が増え、負荷が高まると考えられる。逆に自由度が低くなると、子どもが制作できるものが限られる一方、教師にとっての負荷は低くなると考えられるためである。

以降において、自由度の高いツールと低いツールを実際に使った様子を比較し、この仮説を検証することで、デザインで用いるツールを決定することとする。具体的なツールとして、自由度の高いツールとして Scratch を選び、自由度の低いツールとして Hour of Code を選んだ。それぞれのツール詳細を以下に記述する。

### 4.2.3 Scratch とは

Scratch(スクラッチ) は MIT メディアラボが開発した子ども向けの教育用プログラミング言語 (環境) である。Scratch の特徴としては次のようなことが挙げられる。図 4.4 のように Scratch はスプライトと呼ばれるオブジェクトにブロックと呼ばれる命令を組み合わせることでプログラミングが可能である。操作はドラッグ＆ドロップのみで組み込むことができ、パソコンに不慣れで、プログラミング知識を有していない児童、生徒および学生であっても、ブロックを組み合わせる感覚でプログラミングができる。スプライトは PC 画面上に目に見える形で表示され、自身で作成したプログラムが自ら意図したように作動するかどうかを視覚的に判断することができる。Scratch はフリーソフトウェアであり、Windows、Mac、UNIX などの主要なプラットフォームで動作可能である。ある程度情報環境の整った教育現場ならば比較的容易に導入可能なソフトウェアであるといえる。また、海外の教育現場での導入事例がある [?] ほか、日本での実証実験でも事例の多い [1] [3] ツールである。

自由度という点では、ブロックの種類が豊富であり、複雑なゲームやアニメーションなども時間をかければ作成することが可能であるなど、とても高い。一方で、あくまでプログラムの開発環境であるため、それを用いた学習内容は教師が作成する必要がある。

### 4.2.4 Hour of Code とは

Hour of Code とは、アメリカの非営利団体 Code.org が 2013 年に開発した、教育用プログラミング言語 (環境) である。ブロック状の命令を組み合わせてプログラミングを行なう点や命令の種類によってブロックの色を区別するなど、インターフェイスは図 4.5 のように Scratch と近しい。

大きな違いとしては、Scratch が何を作るかは完全に自由であるのに対し、カリキュラムに沿って設定してある課題を順番に解いていくことで、プログラミング学習を進める点である。また、課題内容も独自のストーリーが設定されており (みつばちが花を回ってみつを集める、など)、子どもが取り組みやすい工夫がされている。加えて、教師が事前に生徒のアカウントを作成しておくことで、各生徒の課題の進捗状況を確認・管理することも可能である。アメリカのカリフォルニア州では高校におけるコンピューターサイエンスの授業単位として Hour of Code が認められた [2] ほか、授業として 400 万人の子どもが受講した [3] など、実績として広がりつつあるツールと言える。

また、Hour of Code には、1 時間で完結する講習用 Hour of Code と、20 時間かけて行う授業用 Hour of Code が存在し、前者は主に単発的なワークショップなどで用いられ、後者は継続的な教育の場で用いられる。

## 4.3. ツール選択のための予備調査

この Scratch と Hour of Code のどちらを用いるかを決めるために、NPO 法人 CANVAS の協力のもと、それぞれを使ったワークショップに参加することを通して、自由度と負荷の大きさについて観察と考察を行った。本節では、各ワークショップの概要と結果を述べていき、どちらのツールを使うかを考察として述べる。

2　Code.org ホームページより　アドレス：https://code.org/promote(2016 年 2 月閲覧)

3　Code.org ホームページより　アドレス：https://code.org/about/2014(2016 年 2 月アクセス)


図 4.5: Hour of Code 使用画面のイメージ

### 4.3.1　Scratch 参加ワークショップの概要

#### アクセンチュア主催「課題解決型」ワークショップ

ワークショップ名:「課題解決型ロボットプログラミング」[4]
ワークショップ概要：イノベーション創出型 STEM 人材の育成を目的として、株式会社アクセンチュアの社会貢献活動の一環として数回行われている活動である。このうち、調査のために 8 月と 11 月の 2 回に参加した。内容としては、まず講師が子ども達がロボットへのプログラミングを学習して、ロボットの仕組み (センサーやモーターなど) を理解する。そして、社会課題を解決するというお題に沿ったロボットを、ファシリテーターのサポートを受けながら発想し、制作する。講師と子どものサポートは基本的にアクセンチュアの社員が担当し、CANVAS スタッフは会場全体の管理を中心にサポートを行った。８月は２日間に分けて実施し、１１月はその内容を１日に短縮したものを実施した。

８月開催時

4　イベントページ：http://canvas.ws/project/robot_acn(2016 年 2 月閲覧)

＜開催概要＞
日時:2015 年 8 月 6 日、7 日 10:30〜16:00(各 5 時間半)
会場:日本科学未来館
スタッフ:講師を含むファシリテータ 8 名、CANVAS スタッフ４名
参加人数:22 人
基本グループ構成:４人１グループ
ファシリテーター:１グループにつき１人
パソコン:１グループにつき１台
参加者年代:小学校高学年
使用ツール：Scratch, Studuino(ロボット), 小型ドローン

ワークショップ内容：大きな流れとしては、初日は Scratch を含む使用ツールの練習、２日目は自由制作と発表というものであった。初日においては、まず講師がプログラミングの概念やロボットの仕組みを説明し、次に子ども達がグループごとに配られたマニュアルに沿ってプログラミングの学習を行った。マニュアルには、“手を上下に動かすロボットを作ろう” といった目標と、そこに至るまでの手順が記載されていた。これを実際にやってみることで、プログラミング方法を学ばせることが狙いであった。

結果：マニュアルを用いたことで、墨田区のような “指示に追いつけない子ども” はいなかった。しかし一方で、マニュアルで指定された数値 (角度や秒数など) を大きな数字 (999999 のような) に変えてみたい子がおり、マニュアルに沿って進もうとする子と衝突し、喧嘩になる場面があった。

11 月開催時

＜開催概要＞
日時:2015 年 11 月 8 日 10:30〜16:00(5 時間半)
会場:IID 世田谷ものづくり学校
スタッフ：講師を含むファシリテータ 7 名・CANVAS スタッフ 2 名

参加人数:14 人
基本グループ構成: 3 人 1 グループ
ファシリテーター: 1 グループにつき 1 人
パソコン: 1 グループにつき 1 台
参加者年代：小学校高学年
使用ツール：Scratch, Studuino, Romo

ワークショップ内容：８月開催時の流れを１日でできる短縮版として午前と午後で行った。ワークショップ時に講師が問いかけたところ、半数以上が Scratch 経験者だということがわかった。また、進行中も講師の説明を先取りする子どももいるなど、全体的に事前知識がついている様子が窺えた。Scratch の練習時間は、８月と同様のマニュアルを用いた。

結果： 今回はマニュアルの指示を手早く終わらせて、作品の話し合いを始めるパターンが目立った。プログラミングについては特にサポートの必要はなく、自由制作のアイディア出しやプレゼンの準備のサポートが中心となった。

### サマーキャンプ

ワークショップ名:「プログラミングでオリジナル楽器を作ってみよう!」[5]
ワークショップ概要：CANVAS 主催の、Makeymakey というツールと段ボールなど身近な材料を組み合わせて、楽器を作ろうというワークショップ。楽器の形を材料と Makey makey で作り、音や効果などを Scratch でプログラミングした。

＜開催概要＞
日時：2015 年 8 月 4 日、5 日①午前の部 9：30〜12：30
②午後の部 14：00〜17：00(各 2 時間)
会場:代官山 SodaCCo
スタッフ：CANVAS のファシリテーター 4 名

5 イベントページ：http://www.canvas.ws/kenkyujo/sc2015/programming/index.html(2016 年 2 月閲覧)

参加人数：①6 名　②4 名
基本グループ構成:グループは作らず、基本的に個人作業
ファシリテーター:子ども 1 人につき 1 人
パソコン:子ども 1 人につき 1 台
参加者年代：小学生 3〜6 年生
使用ツール：Scratch, makey makey, 紙コップや段ボールなどの材料

ワークショップ内容：大きな流れとしては、初日に Scratch と makey makey の使い方を学び、2 日目で自由制作と発表を行うというものであった。このワークショップの主目的は makey makey を扱って自由制作を行うことであったことから Scratch の練習時間は短めに設定してあった。

結果：子どもが Scratch で何かを表現したい場合、ファシリテーターが子どものやりたいことを聞き出し、ヒントと指示を多く出すことが多かった。

### 墨田区 Scratch ワークショップ

ワークショップ名：すみだプログラミングワークショップ[6]
ワークショップ概要：墨田区役所との共同で行われている「やさまち宣言」の普及プロジェクトの一環として行われたワークショップ。やさまち宣言とは、墨田区におけるまちづくりの基本方針をまとめたものであり、“あいさつをする”・“高齢者を大事にする” などの目標が含まれている。ワークショップではプログラミングを用いてそれらを表現することで「やさまち宣言」の普及と子どもたちへのプログラミング学習機会の提供を行った。

＜開催概要＞
日時：2015 年 11 月 3 日①10：00〜12：00
②13：30〜15：30(各 2 時間)
会場:墨田区役所
スタッフ：講師を含むファシリテーター 8 名

6　イベントページ：http://canvas.ws/workshop/4587(2016 年 2 月閲覧

参加人数：①20 名　②13 名
基本グループ構成: 4 人ずつ集まって座るが、基本的に個人作業
ファシリテーター:特に配置は固定せず随時子どものサポートに回る
パソコン: 1 人につき 1 台
参加者年代：①小学生　②小学生と中学生
使用ツール：Scratch

ワークショップ内容：大きな流れとしては、前半で Scratch の扱い方を学び、後半で自由制作を行うというものであった。前半においては、講師が Scratch の画面をプロジェクターで投影し、お手本を見せ、子どもたちにそれをやらせるということを繰り返すことで扱い方の学習を進めた。例えば、“命令をずっと繰り返させる方法” を前でブロックを一つづつ組み合わせながら実演し、同じ事を子どもたちにやらせるというものである。

結果：数人の子どもが講師の話を聞かずに目の前の Scratch を触り続ける様子が確認された。ファシリテーターが注意をし、マウスから手を離しても画面が気になるなど、講師からの指示を聞く時間と作業の時間を切り離せない様子であった。この事によって、講師の指示を理解しないまま学習が進み、何をしていいかわからない状態に陥っていた。そうした子どもにはファシリテーターが個別について指示をすることでキャッチアップが図られていた。そうした子どもが Scratch を触っている時には、数値を自由に変更することや、違うブロックを置いてみるなど、興味を持った様々な動作を試す様子が確認された。

### 4.3.2 Hour of Code 参加ワークショップの概要

ワークショップ概要：NPO 法人 CANVAS・株式会社 Salesforce・隅田小学校の三者が共同で行っているワークショップ。子どもたちにプログラミング教育の機会を与えることを目的とし、講習用 Hour of Code を 2 人 1 組で進めさせ、わからないことがあれば周りのスタッフに質問をするという形式をとった。

＜開催概要＞
日時：2015 年 10 月 26 日・11 月 9 日・12 月 14 日 14:30〜15:20(各 50 分)

会場:墨田区立隅田小学校
スタッフ：小学校教員 1 名・Salesforce 社員 9 名・CANVAS スタッフ 4 名
参加人数：20 人
基本グループ構成: 2 人で 1 グループ
ファシリテーター:特に配置は固定せず随時子どものサポートに回る
パソコン:子ども 2 人につき 1 台
参加者年代：小学生 3〜6 年生
使用ツール：講習用 Hour of Code

ワークショップ内容：基本的には、Hour of Code を自由に進めさせ、わからない点については適宜教員またはファシリテーターに質問するという進め方をとった。
結果：驚くほどファシリテーターの出番が少なかった。3 回目の授業を例にとると、生徒からファシリテーターに質問が来ることは一度もなく、手持ち無沙汰のファシリテーターが生徒に何かわからないところはないかと聞くほどであった。
考察：この要因としては、各ステップをグループごとのペースで進められることで講師の指示についていけない子どもが発生しないことや、ステップに関係のないブロックは表示しないようになっているため、脱線しづらい環境になっていることが挙げられる。全てのグループが Hour of Code を最初から最後のステップまでをやり通すことができた点からも、自学自習を主とした学習ツールであると言える。

### 4.3.3 考察

#### ファシリテーターに対する負荷の比較

ファシリテーターがどのタイミングでどれだけ忙しくなるかを、2 つのワークショップを通して比較した。具体的には上述した、“墨田区 Scratch ワークショップ” と “隅田小学校における Hour of Code ワークショップ” において、“ワークショップのどのタイミングで何人のファシリテーターが子どものサポートを行っていた

か” をそれぞれグラフ (下図 4.6, 図 4.7) にまとめ、比較した。なお、このグラフはワークショップ後に自身の記憶を元に作成し、一緒に参加した CANVAS のメンバーの方々にグラフが正しいかをチェックをいただき信頼性を向上させたものである。また、この 2 つのワークショップを選んだ理由は、ファシリテーターの配置方法がどちらも同じ (グループごとではなく、全体に対して配置された) であったためである。

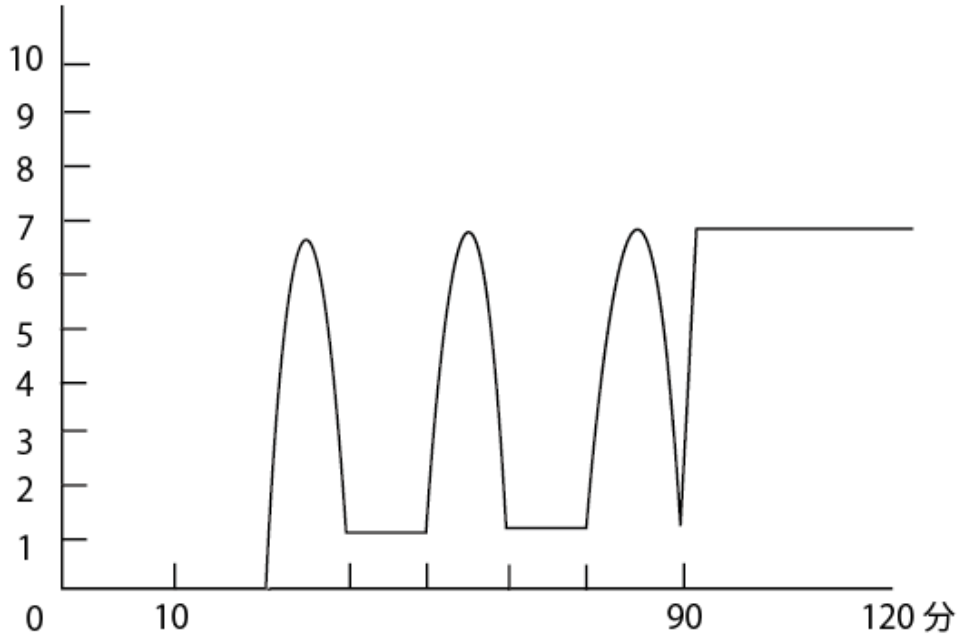



図 4.6: 墨田区 Scratch ワークショップにおけるファシリテーターの忙しさ

＊縦軸が子どものサポートに回っている人数、横軸がワークショップの経過時間を指す

墨田区 Scratch ワークショップでは、開始から２０分は講師の挨拶や最初の説明があり、子どもたち全員が説明を聞いており、ファシリテーターは特にサポートはしていなかった。そして講師からの指示を受け、子どもがパソコンを触り始めると、ほぼ全員が子どものサポートについた。そして手を止めて再び講師の指示を聞く際には、数名の子どもが指示を聞かずパソコンを触り続けてしまったため、数名が子どもへのフォローに回った。これを 90 分まで繰り返し、残りの自由制作時間は常に子どもの対応に追われていた。

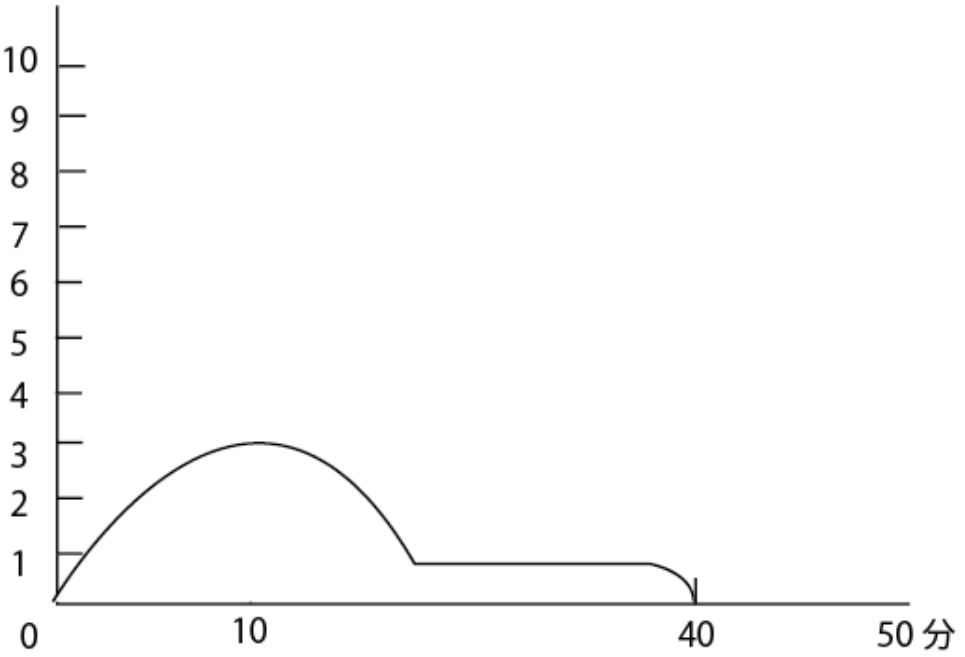



図 4.7: 隅田小学校 Hour of Code ワークショップにおけるファシリテーターの忙しさ

隅田小学校の Hour of Code ワークショップでは、開始から１０分は機材トラブルや Hour of Code へのアクセス方法などで数名のファシリテーターがサポートに回った。準備が一通り済んでからは、Hour of Code を解き進めていったが、ほとんど質問も出ず、ファシリテーターの大半は手持ち無沙汰という様子であった。最後の１０分は先生が教壇に立って今日学んだことの振り返りなどを行い、その間もファシリテーターは子どもに付く必要はなかった。

### Scratch の考察

Scratch は自由度の高さから、生徒の好奇心を刺激しやすく、また作れるものも多様なことから、創造性を生かして”作りたいものを作りながら、失敗から学ぶ”スタイルの学習方法に適していると言える。しかし、小学校への導入を考えた場合、そのスタイルは教員のサポートが必要となるため、負荷が大きいと考えられる。

**Hour of Code の考察**

Hour of Code はステップに関係のないブロックは表示されない。このように自由度を低くしたことが、生徒の脱線的な行動を抑えることになったと考えられ、教員にとっての負荷が小さいと言える。また、全てのグループが Hour of Code を最初から最後のステップまでをやり通すことができた点からも、自学自習を主とした学習ツールであると言える。しかし、ゲーム的な要素が強く、プログラミングについての理解に結びつくかどうかは不明であるなど、生徒の創造や自主的な試行錯誤の幅が狭まると考えられる。

以上のことから本デザインでは Hour of Code をツールとして用いることとする。

## 4.4. 授業案の準備

### 4.4.1 Hour of Code ワークショップで気づいた課題の一例

ワークショップを通じて、Hour of Code にも留意すべき課題点があることがわかった。具体的には、Hour of Code は自由度を少なくした代わりに “ゲーム的な要素が強く、クリアしても必ずしも理解に結びついていない” という課題である。以下にその根拠となる観察事例を示す。

男子二人の様子：アナと雪の女王ステージ１８ (図 4.8 の問題) にて

この問題の正解は、上記コマンドから「前に動く」ブロックを除いたものであったが、大人が ”これはいらないんじゃない ”と声をかけるまで気付くことができなかった。“雪の結晶の枝を描く” は、キャラクターの前に雪の結晶を歩きながら描くという命令であったが、その前に「前に動く」ブロックを置いていたことから、ブロックの意味を理解していなかったことが推察される。

また、ワークショップ全体を通して詰まった際にはお互いに意見を出し合いながら何度もプログラムの書き換えと実行を行う様子が見られた。しかし、その試行錯誤が、ループや角度の概念を理解したうえでではなく、正解してもコマンドを直感的に当てはめているうちに偶然うまくいくケースが多いようにみえた。こ


図 4.8: Hour of Code の stage18

れは、”さっきのコマンドってなんでうまくいったか教えてくれない？”と問いかけた際に、”うーん… ”と言ったまま答えられなかったことにも表れていた。

このことから次節からデザインする授業案ではクリアするだけでなく、前もって概念を理解できるように教える段階を加えていく。

### 4.4.2 概要

上述したように、ワークショップで用いられていた講習用 Hour of Code では単発的でありプログラミングの概念の学習に最適とは言い難い。そこで今回は、授業への導入のために、開発元の団体である Code.org が作成した２０時間タイプの授業案をベースとすることとした。

この授業案は Code.org のホームページ上で非営利目的の使用に限り、著作権フリーで公開されており[7]、誰でも手にいれることができる。アメリカにおいて 9 万校以上で実施されている実績[8]や、外部団体からの評価も高い[9]点から、独力で

7　https://code.org/curriculum/docs/k-5/complete.pdf

8　https://code.org/about/2014

9　シカゴ大学の評価レポート：*http : //outlier.uchicago.edu/evaluation*$_c$*odeorg/*

１から作るよりも、より完成度の高い授業案になると考え、採用する。しかし、この授業案ならびに資料は、英語版が公開されているものの日本語版は存在していない。そこで、まず 20 時間分の授業案すべてをわかりやすく日本語に説明し直した。

### 4.4.3 構成

その結果、授業案は全 76 ページのテキストになった。構成としては、アルゴリズム・ループ・デバッグ・条件分岐・イベント・ネットリテラシーという学習目標について、パソコンを使わない (“アンプラグドな” と呼ばれる) アクティビティを１コマ行い、その内容を復習かつ実践するために Hour of Code を用いた授業を数コマ用いた。以下の図 4.9 は授業案の概要であり、ここでのパソコンを使わないアクティビティは、１・２・５・９・１２・１４・１５・１８である。

| # | Lesson Name | Description |
|---|---|---|
| 1 | マス目プログラミング | 生徒たちはあらかじめ決められた命令を使って、クラスメイトが同じ絵を再現できるようにアルゴリズム(命令の集まり)を書きます。 |
| 2 | 身の回りのアルゴリズム | この授業では私たちがアルゴリズムをいかに毎日つかっているかを意識させます。この授業はコンピューターサイエンスの概観とそのなかでいかにアルゴリズムが大事か、という点に焦点をあてています。 |
| 3 | 迷路：シーケンス | キャラクターを迷路から脱出させるために、プログラム(コンピューターに対するアルゴリズム)を書きます。これを通してプログラムのなかでシークエンスがいかに大事かを学びます。 |
| 4 | アーティスト：シーケンス | 異なる線や形を描くプログラムを書きます |
| 5 | 繰り返す感じ！ | ここではループ（繰り返しの命令）の考え方を、ダンスを通して紹介します。子ども達は簡単な振り付けを習って、それを繰り返します。 |
| 6 | 迷路：ループ | ループを使って迷路から脱出するプログラムを書きます |
| 7 | アーティスト：ループ | コードのなかにあるパターンを見つけて、違う形を描くプログラムを書きます。<br>これを通して子ども達は自分のプログラムをもっと効率的にしてくれる"ループ"の概念を理解します。 |
| 8 | みつばち：ループ | みつばちの世界でループを使ったプログラムを書いてみましょう |
| 9 | プログラミング・リレー | 子ども達には"マス目プログラミング"(第一回授業)のアルゴリズムを校庭に描く、リレー競走をしてもらいます。<br>ここでは一度にひとつの命令しか書けず、もし間違えたらまたーからやりなおしです！ |
| 10 | みつばち：デバッグ | 途中まで完成したプログラムを見せて、そのデバッグと修復をさせます |
| 11 | アーティスト：デバッグ | 次はある絵と、それを描くための未完成なプログラムを渡します。<br>そして、そのプログラムのデバッグをしてもらいます。 |
| 12 | 条件分岐 | 条件分岐を理解するために、子ども達には自分でルールを作るカードゲームをやってもらいます。<br>例えば、"もし赤いカードをポイントをゲット！"とか、<br>もし "黒いカードを引いたら相手にポイントが入る"などです |
| 13 | みつばち：条件分岐 | みつばちの世界で、条件分岐を使ったプログラムを書きます |
| 14 | 二進数ブレスレット | ここでは２進法で表した自分の名前の１文字目(アルファベット)を描いた紙のブレスレットを 作ります。<br>これを通して同じ情報でも違う表しかたがあるということを学びます。 |
| 15 | ビッグ・イベント | プログラミングにおける"イベント"の概念を学んでもらいます。イベントというのはコンピューターが常にこないか確認してる行動のことです。<br>これを体感するためのワークでは、まず先生にはおもちゃのリモコンを持ってもらいます。<br>そして、いろいろなボタンを押すと、子ども達がボタンごとに違う言葉を話すということをします。 |
| 16 | フラッピー | イベントの考えを使って、オリジナルのゲームを作らせましょう。<br>例えば"マウスがクリックされたら、鳥が羽ばたく"そして"鳥が地面に当たったらゲームオーバー"みたいな |
| 17 | プレイラボ：お話作り | これまでに習ったすべての概念をつかって、オリジナルの、そしてインタラクティブなゲームを作ってみましょう |
| 18 | ネットリテラシー | 先生は生徒に、ネット上では足跡や痕跡が残る、<br>という考え方を生徒に紹介します. |
| 19 | アーティスト：入れ子のループ | "入れ子のループ"をつかって、面白くて綺麗な模様をプログラムで描いてみましょう |

図 4.9: 授業案の概要

各授業内容としては、以下の構成となった。
まず、パソコンを使わないアクティビティは、”導入・アクティビティ・まとめ・振り返りクイズ”という構成になっており、各内容は以下のようになっている。

- 導入：前回の振り返りや、新しく覚える言葉の説明など
- アクティビティ：体を使ったゲームや工作で新しい概念を学ぶ
- まとめ：授業で学んだ言葉などをもう一度振り返る
- 振り返りクイズ：個人作業として取り組む復習用教材を解く

## 4.5. 入門ガイドの概要

### 4.5.1 新規性と意義

#### 新規性

Hour of Code を用いた指導者向け入門ガイドを作成する新規性について述べる。まず Hour of Code のガイドブックとなる書籍について Web 通販サービス amazon で検索したところ、図 4.10 に示したように、Hour of Code について取り扱っている書籍は 0 件であった[10]。この理由としては、Hour of Code は 2 年前にアメリカで開発された新しいツールであるため、日本において十分認知されていないことが考えられる。

10 amazon.co.jp において、“Hour of Code” というキーワードで検索を行った (2016 年 1 月 25 日時点)

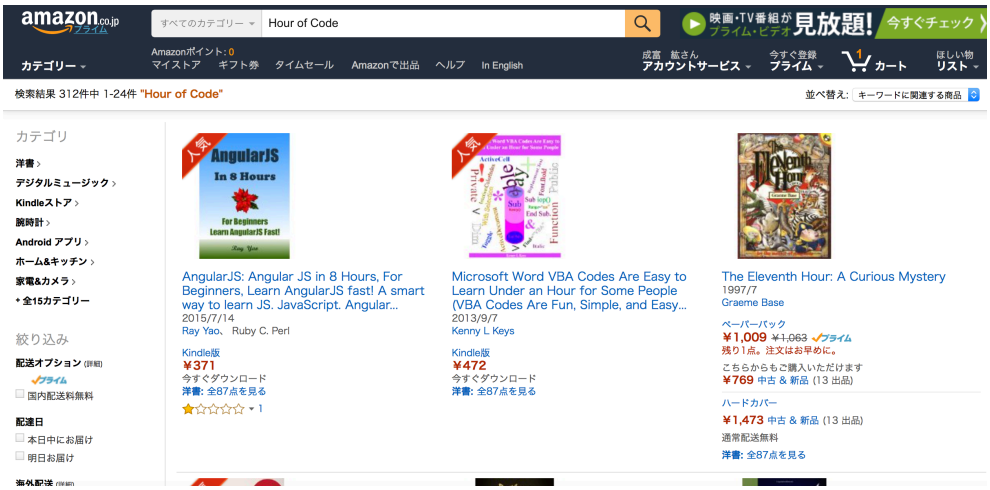



図 4.10: Hour of Code 関連書籍の検索結果

### 意義

まだ授業案を読んで、教師の予習としては子どもの行う内容を自身でもやってみることが考えられる。しかし、それだけでは子どもと同じレベルの理解に留まってしまい、知識の全体から見た位置づけなどの教える側としての理解には結び付きづらい。そこで、入門ガイドを作成することで、教員は短時間で効率的に授業実施に必要な知識が身につけることができ、Hour of Code を使った授業案をより活用できる。加えて、入門ガイドをデータで共有できる (直接教員に指導を行う必要が無い) 形にすることで、普及という点からも効果的と言える。

## 4.5.2 構成

入門ガイドの要件としては、授業案で生徒に教える用語の説明がある。しかし、授業案の解説 (用語説明) だけだと、そもそもなぜ教えるのかという意義やプログラミング教育全体から見た授業の位置づけまで理解することができない。また、Hour of Code の扱いや準備など、具体的な操作方法も未経験の教員にとっては必要である。以上のことから、入門ガイドは大きく導入編・事前知識編・準備編の

３つのセクションで構成した。導入編ではプログラミング教育の概要やその意義などの概念を解説し、事前知識編では授業で生徒に教える用語を解説する。また準備編では Hour of Code の具体的な操作方法やアカウント作成などのセッティング方法について解説する。

## 4.6. 入門ガイドの詳細

### 4.6.1 導入編

#### 概要

導入編では、コンピューターの概念や、なぜプログラミング教育を行うのかといった疑問点を解消するために、図 4.11 のように Q & A という形でそれらを説明していった。この疑問点は、プログラミング未経験の小学校教員の方へのヒアリングから作成した。具体的には以下の５つの疑問点について解説を行った。なお、カッコ内はそこで説明する内容を示している。

- Q1:プログラミングとは何？(プログラミングの定義)
- Q2:なぜ子どもにプログラミングを教えるの？(プログラミング教育で伸びる能力)
- Q3:教える内容は？どこまで教えるの？(プログラミング教育のステップ)
- Q4:どうやって教えるの？(ツールの簡単な解説)
- Q5:とは言ってもプログラミングについて何も知らないんだけど？(次章への導入)

**<導入編>**

**Q1：プログラミングとは何？**

"プログラミング(programming)"とは、"コンピューターへの命令を作ること"です。コンピューターへの命令をプログラム(program)と呼び、プログラム(program)すること(ing)がプログラミング(programming)です。

ではコンピューターとは何でしょう？

**コンピューターとは、"自動計算機"のことです。**言い換えると、"指示を与えれば自動で計算や処理をしてくれる機械"です。機械にコンピューターが組み込まれることにより、様々な機械が自動で動いてくれます。

ちなみに混同しがちな言葉としては、"パソコン"があります。これはパーソナルコンピューターの略であり、コンピューターのなかでも個人が使い易い価格や用途のものを指します。工場の組み立て機械や研究施設の大型コンピューターなどは、これには当てはまりません。概念図としては以下のようになります。(とても簡略化してありますので、参考程度にしてください)


"コンピューター"の概念図

こう考えると、身の回りはコンピューターで溢れていると言えます。自動ドア・信号機・携帯電話・自動車などなど…

こうした機械が"どんな動きをすればいいのか？""どんな計算をすれば良いのか？"という"指示"を作ることがプログラミングと言えます。

**Q2：なぜ子どもにプログラミングを教えるの？**

大きく以下の２つの理由があります。

・子どもの能力を伸ばす

図 4.11: 導入編のページ例

#### 工夫点

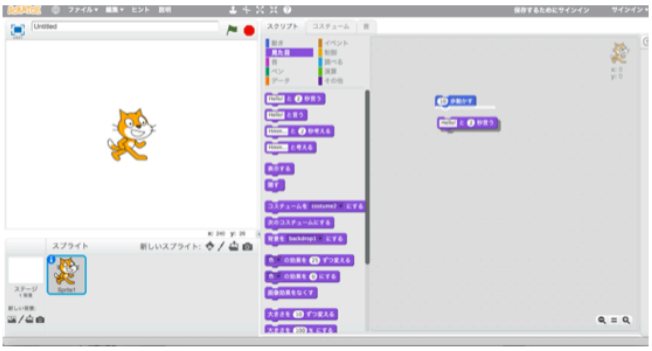
プログラミング未経験者が読むことを想定して、Q1 において、“そもそもコンピューターとは何か” という説明から行った。具体的にはコンピューターの定義を述べて、パソコンやスマートフォン以外にも産業用ロボットや研究用コンピューターなど、プログラミングが使われている例を挙げた。これにより、生活の中でプログラミングはどのように使われているのかを理解しやすくした。

### 4.6.2 事前知識編

#### 概要

事前知識編では、条件分岐や繰り返し処理といった、プログラミングの概念を理解してもらうために、図 4.12 のようにパソコンを使って手を動かしながら学ぶ手順を作成した。

**Hour of Codeで扱う用語の理解**
**アルゴリズムとは、**
**→命令の集まり**
例えば、"10歩動かす"の下に、今度は"見た目"パレットの中にある"Hello！と２秒言う"をつけてみましょう。



ついたら、どちらでも良いのでブロックをクリックしてみましょう。
すると、ネコが１０歩動いてからHello！と言いました。

Scratchに限らず、プログラミング言語は、基本的に上から順につながったものが実行されていきます。
この"つながった命令のかたまり"をアルゴリズムと言います。

**ループとは**
**→命令の一部または全体を繰り返し行い続けること**

先ほどの命令にループを組み込んでみましょう。
まず真ん中の"カテゴリー"から"制御"を選択してください。

"制御"のなかには、"ずっと"や"もし"など、"動きや見た目を、どういう順番で進めていくか"に関係する命令が入っています。

図 4.12: 事前知識編のページ例

### 工夫点

ここではあえて Scratch をツールとして選んだ。その理由としては、Hour of Code は課題を解き進めていくという積み重ねを前提としたツールであるため、短時間で用語の概念を理解することには向かないと考えたためである。具体例を挙げると、条件分岐という用語がある。この言葉の定義は “ある条件に当てはまったら、特的の動きをしなさいという命令” とした。これを Hour of Code で学習するためには 4.2.4 で述べたように、ストーリーに沿って課題を解いていく必要がある。そのためにはストーリー独自のコマンド (みつをとる、など) や課題の目的 (みつばちを動かし、花からみつを集める) を理解する必要があり、その練習に図 4.13、4.14 のような、数ステップを行う必要がある。また、キャラクターを何歩前に進めるか、いつ方向転換をさせるかなど、前回までの復習を兼ねた要素があり、図 4.15 のように条件分岐という概念に直接関係するものだけでなく、その他の要素についても考えなければならない。


図 4.13: 条件分岐のステップ 1


図 4.14: 条件分岐のステップ 2


図 4.15: 条件分岐のステップ 3

### 4.6.3 準備編

#### 概要

準備編では、教員が授業をする上で必要となるツールの扱い方を解説した。具体的には、図 4.16 にような Hour of Code の操作練習と、図 4.17 にような生徒の進捗管理画面への登録方法の 2 つである。なお、ここでの進捗管理とは、どの生徒がどの問題まで解き終わったかを教員が一括して確認できる Hour of Code の機能のひとつを指している。これらを学ぶことで、実際に授業を開始するまでの手順を学ぶことができる。

#### 工夫点

隅田小学校で行われた Hour of Code のワークショップにおいて、進捗管理の機能が用いられていなかった。その結果、早く解き終わった子どもは別のカリキュラムを始め、遅れている子どもは焦って途中でカリキュラムを変えるなど、どの子がどれだけ進んだかを把握できなかった。授業においても、その把握を行うことは授業の進行度合いや方針の把握にとって必要となる。これに加えて、Hour of Code について CANVAS の PEG チームの方々と話し合いをした際に、進捗管理機能があること自体を知っている方がいなかった。このことから、存在自体を伝えるという意味でも記載するべきだと判断した。

⑥以下のように、実際に問題を解いていく画面が開きました。


⑦各要素を解説すると以下の通りです。


図 4.16: 準備編の操作練習ページ例


図 4.17: 準備編の登録方法ページ例

# 第5章
# 評　　価

## 5.1. 評価方法の概要

デザインした入門ガイドについて、小学校教師3名高校教師1名に依頼をして評価をいただいた。また、事前知識について未経験の人がどれだけ理解しやすいかという点においては、評価者が小学校教師でなくとも参考になると考え、教員でない一般の方9名からも同様に評価してもらった。評価手段としては以下の3つを行った。

- 半構造化アンケート
- パソコン操作の有無確認
- 用語理解度の口頭テスト

## 5.2. アンケート

### 5.2.1 評価内容

アンケートでは主に入門ガイドを読み、指示に沿ってパソコンを操作してもらってから理解のしやすさなどを評価してもらった。具体的には以下の項目を質問した。

1. 指示の理解しやすさ (10段階評価)
2. 用語の理解しやすさ (10段階評価)
3. 所要時間

4. 良い点 (自由記述)

5. 悪い点 (自由記述)

6. その他の意見など (自由記述)

1,2 では内容の理解のしやすさについて 10 段階で評価してもらった。この段階には特に明確な基準は設けず、主に自由記述の前に一度振り返りを行うというステップとして項目に加えた。3 では、読み終わるまでの所要時間を聞いた。そして 4 5 で良い点と悪い点という 2 つの視点から自由記述を行ってもらった。

### 5.2.2 結果

#### 10 段階評価

10 段階評価の結果は以下の通りであった。

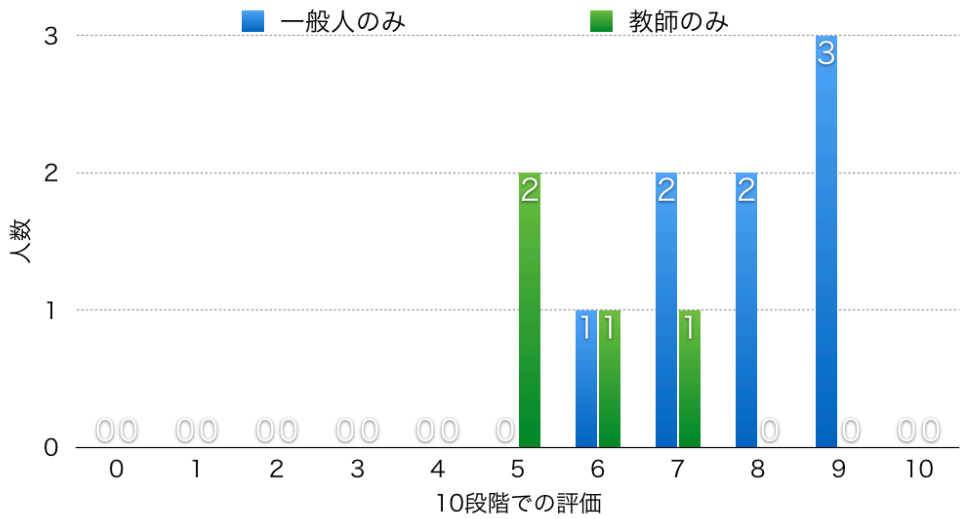



図 5.1: 指示の理解しやすさ (10 段階評価)

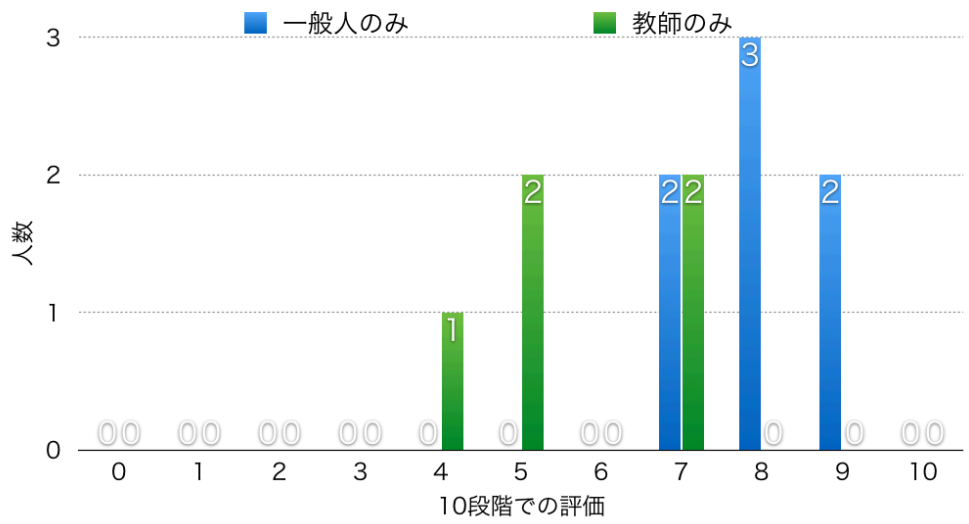



図 5.2: 用語の理解しやすさ (10 段階評価)

### 所要時間

読み始めてから終わるまで (パソコン操作も含む) の所要時間は以下の通りであった。なお、プログラミング経験者かどうか、とパソコンを操作したかという２軸で分類した。

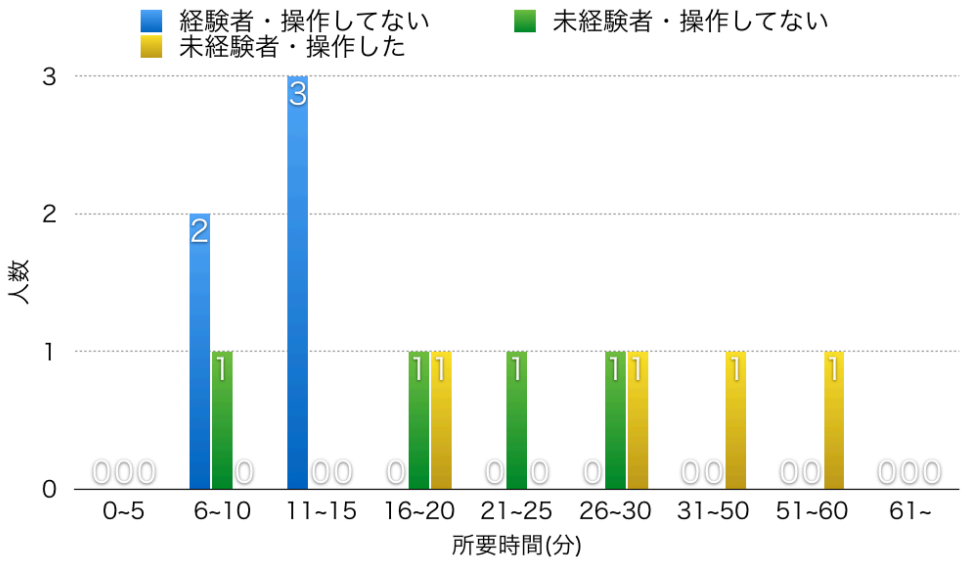



図 5.3: 所要時間

### 自由記述

自由記述の内容をチェックし、似た意見をまとめて整理したところ以下の通りとなった。

- 良い点 (自由記述)
  - 図示による手順説明のわかりやすい
    - 操作の手順ごとに例が示されるので、それを見ながら操作できるのはわかりやすい (40 代女性)
    - 図と合わせて提示されているので、操作がわかりやすい (20 代女性)
    - Scratch の説明は手順がすごく丁寧に書かれていてわかりやすかった！プログラミングできるかもって思えた (20 代女性)
    - 指示に図がついているのでわかりやすい (30 代男性)
  - 導入編の Q & A 形式がわかりやすい

    - Q & A 形式がわかりやすい。特にコンピューターの概念から説明してもらえたのは導入としてよかった (30 代男性)
    - Q & A 形式は良い (30 代男性)
  - その他の良い点
    - 用語が太字になっており、また簡潔にまとまっているため、理解を整理しやすい (30 代男性)
    - 教える目的や前提知識が書かれていて、理解しやすい (20 代男性)

- 悪い点 (自由記述)
  - 導入編が長い
    - 前置きが長い (はやく本題に入りたい)(30 代男性)
    - 自分はプログラミング経験がある程度があり、意義も理解しているので、序盤の解説は不必要だった。(30 代男性)
    - 前文が長いです。(20 代女性)
    - 目次がないから全体像がわからない (20 代男性)
  - プログラミングの意義が不明瞭である
    - 伸びる能力や意義が不明瞭に感じ、読む気力が下がった。立派な文言が並んでいるが、本当にそれが正しいのかは疑問を感じる教師は多いだろう (30 代男性)
    - プログラミングの良さが一方通行で、学校側の主張が組み込まれていないと思います。(20 代女性)
  - パソコンに慣れていない読者への説明が足りない
    - 一番手こずったのは、パソコン操作に慣れていないせいか、ダウンロードしてサイトを開くことだった (40 代女性)
    - 受け取った zip ファイルを閲覧するまでに苦労した (50 代男性)
  - その他の悪い点

* Q3 は、プログラミングで何ができるのかが分からないのにどこまで行くかを説明されてもピンとこない。「完全にオリジナルなアプリケーションを作成したり〜」のような一文があれば、初心者はそれで満足するかも。(20 代女性)
* 小見出しの出てくるタイミングが突然すぎる印象 (20 代男性)

### 5.2.3 考察

#### 10 段階評価

指示のわかりやすさ、用語のわかりやすさともに、ほぼ全て 5 以上であり、ある程度のわかりやすさがあると言える。また、一般の方に比べ、教師からの評価は全体的に低くなっている。これは自由記述からも見えるように、プログラミング教育を実施する立場から見るとより高いレベルを求めることや、プログラミング教育を行うこと自体への拒否感が影響していることが考えられる。

#### 所要時間

所要時間からは、全員が 60 分以内に読み終わったことがわかった。また、パソコンを操作せずに読んだ人は操作した人よりも所要時間が 5 10 分ほど短くなっている。なお、30 分以上かかったのは 2 人であり、その 2 人は普段パソコンを使わない年配の方であった。よって、普段からパソコンの操作になれている人は 30 分以内、そうでない場合 60 分以内に終わる内容であったと言える。日本標準教育研究所によると、小学校教師の授業準備時間は「31〜60 分」が 51.4 %と最も多く、次いで「30 分以下」22.5 %、「61〜120 分」16.6 %となっている [1] ことから、平均的または少し短い時間であると言える。

1 日本標準教育研究所「小学校教師の現状と課題　小学校教師の意識についてのアンケートー実施報告書ー」平成 25 年 8 月公表

#### 自由記述

自由記述の結果から得た、 教員用入門ガイドの優れた点と課題点を整理すると以下の通りとなる。

＜優れた点＞

- 図示による手順説明がわかりやすい
- 導入編の Q & A 形式がわかりやすい

＜課題点＞

- 導入編が長い
- プログラミングの意義が不明瞭である
- パソコンに慣れていない読者への説明が足りない

ここから、説明方法がわかりやすかったと言える一方で、３つの新たな課題が見つかった。これらを踏まえた改善案については次章にて記述する。

## 5.3. パソコン操作の有無確認および口頭テスト

### 5.3.1 評価内容

口頭テストでは、用語の理解度を測るために各用語 (全５つ) の概念について口頭で説明してもらい、何個正しく説明できたかをチェックした。また、口頭テストの結果を詳細に分類するためにプログラミング経験の有無とパソコンを操作したかどうかを事前に確認した。なお、依頼の際には “パソコンも操作して欲しい” とだけ伝え、強制はしなかった。これは、このガイドが実践的に用いられる際には、パソコンを操作せず読むだけで理解しようとする人が出ることが想定されるため、“どれだけの人がパソコンを操作せず読むだけにするか” と、“操作の有無が理解度に与える影響” を観察するためである。

### 5.3.2 結果

ガイドの指示に沿ってパソコンを操作した人は、１２人中４人であった。また、①“プログラミング未経験かつ操作を行わなかった人”、②“プログラミング未経験かつ操作を行った人”、③“プログラミング経験かつ行わなかった” の３タイプの人計５人に対して行った口頭テストの結果は、①4 点と 3 点、②5 点と 4 点、③2 点であった。これらの関係は下図 5.4 に整理した。

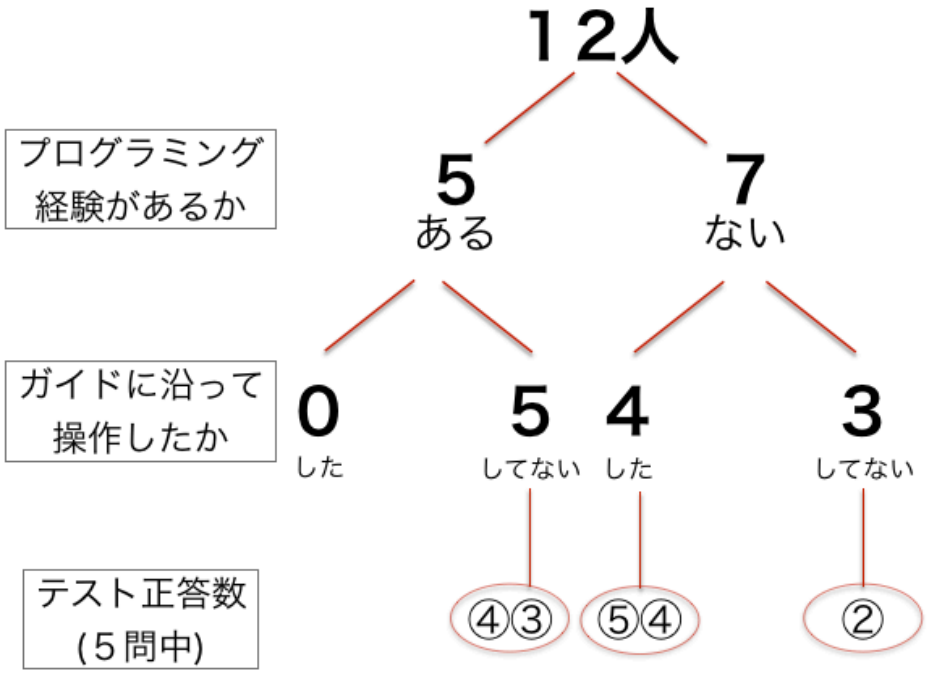



図 5.4: パソコン操作の有無と口頭テストの結果

### 5.3.3 考察

パソコン操作の有無確認から分かったこととして、まずプログラミング経験がある人はパソコンの操作は行わず、読むだけにする傾向があると言える。また、未経験者でも操作をせず読むだけにする人はある程度存在すると言える。この状況へのアプローチとしては “読むだけでもより理解しやすくすること” か、“パソコンを触るモチベーションを向上させる” かの 2 つがあるが、私は後者を重視すべきだと考える。理由としては、Hour of Code で用いるビジュアルプログラミング言語の操作練習になるほか、試行錯誤が必要になるプログラミングにおいては暗記しているよりも感覚的に理解できていることが大事だと考えたためである。

口頭テストからは、プログラミング未経験者であっても、入門ガイドを読んで操作もすることでプログラミング経験者と同じほどの理解を得ることができることが分かった。一方で、未経験者が読むだけに留めた場合は、なんとなく分かった気がする、といった理解度の低い状態にとどまることが示唆された。

## 5.4. 課題点と改善案

考察で述べた課題点を以下のように整理し、それぞれについて今後の改善案を述べる。

- パソコンを操作する人が少ない
  - “プログラミングの意義” の部分でプログラミングはパソコンに触れて試行錯誤することが重要であると強調する
  - 用語ごとに”◯◯するプログラムはどうすれば作れるでしょうか”といった練習問題を提示する
- 導入編が長い点
  - 概要の部分に入門ガイド全体の目次を記載することで、読者が頭から順番に読んでいく必要をなくし、自身に合った場所から読めるようにする

  - 入門ガイドを”〜編”ごとにファイルで分け、そもそも不要なファイルは開く必要すら無いようにする

- プログラミングの意義が不明瞭である点

  - 本ガイドの文章だけでプログラミング教育へのモチベーションが低い教師を高い教師に変えることは現状においては難しい。モチベーションの低い教師を変えるためには、学校でのワークショップなどの事例を増やし、教師達の意識を変えていく必要がある。そのため、教員用入門ガイドの修正としては事例の紹介を増やすようにするが、その一方で、モチベーションが既に高いがプログラミング教育への知識が低い教師に本ガイドを活用してもらうことが有効だと考える。

- パソコンに慣れていない読者への説明が足りない点

  - 現在の若い世代は幼いころからパソコンに触れる機会も多く、ファイルの保存やデータの開き方などを直感的に行うことができる。しかし、年配で普段からパソコンに触れない人の場合、ウィンドウを “消す” という表現がわからなかったり、flash のアップデート方法がわからず途方にくれたりと、基礎的な練習が必要になる。そこで、マニュアルにおいては、年配の人のために操作やトラブルシューティングを細かく載せたものを違うバージョンとして作成することが求められる。

# 第6章
# 結　　論

## 6.1. 全体の考察

本研究では、プログラミングを学校教育として普及させることを目的とし、現在行われているプログラミングワークショップへの参加や、プログラミング教育実践者、また学校教員へのヒアリングを通して、小学校にプログラミング教育を導入するうえでの課題点を整理した。その結果、教師のプログラミングへの知識不足・カリキュラムが無い・多忙による準備の負荷の大きく3つに絞ることができた。そしてそれを解決する手段としてHour of Cordが適しているとの仮説を立て、ワークショップへの参加を通して検証した。その上で、Hour of Codeを用いた授業を行うための基礎知識を未経験者向けに解説した教員向け入門ガイドを作成した。これを小学校の教師や教育者、プログラミング教育の実践者などに見てもらい、その有効性をインタビューを通して検討した。その結果、わかりやすさや、所要時間の短さなどの特徴が見えた一方、パソコンを操作せずに読むだけで済ませる人がいることや、より詳しく説明する必要のある箇所があることなど新たな課題も発見した。よってこのまま導入していくことに課題は残るものの、今後のプログラミング教育普及へのプロトタイプとしての価値はあると考えられる。

## 6.2. 今後の展望

今回の評価では隅田小学校の西中先生に、入門ガイドと授業案を見せた上で、今後の展望についても伺った。その回答は、”授業案をこのままの形で教育現場で行うことは現状難しいが、この一部を課外活動の一環として導入することや、現

在用いられている通常の授業に絡めることで、少しずつプログラミング教育の実施事例を積み上げていくことができる”というものであった。また、今回は入門ガイドのデザインに注力したため、授業案の内容までは変えなかった。しかし今後はこれまでの経験や授業の実施を踏まえて、授業案自体もブラッシュアップしていきたい。

# 謝　　辞

本研究は、多くの方々のご協力によって成立し、独力では決して完成しないものでした。

まず本研究の指導教員であり、研究の方向性や内容の修正について、指示ばかり求めたり諦めかけたりする自分に対し、的確な指導と暖かい励ましをしていただきました慶應義塾大学大学院メディアデザイン研究科の石戸奈々子准教授に心から感謝いたします。

また、的確な指摘をいただきました慶應義塾大学大学院メディアデザイン研究科の岸博幸教授に心から感謝いたします。研究室やイベントに参加させていただいたことでも視野が大きく広がったと感じています。

そして研究の方向性について様々な助言や指導をいただきました慶應義塾大学大学院 メディアデザイン研究科の加藤朗教授に心から感謝いたします。お時間の無い中、何度も丁寧にレビューをしていただき、論文の改善がぐんと進みました。

研究に大きくご協力いただいた NPO 法人 CANVAS の PEG チームの土橋様・熊井様・寺田様に深く御礼申し上げます。ワークショップへの参加という貴重な機会をいただいただけでなく、論文の相談にも乗っていただき、誠にありがとうございました。これからこのご恩を何かの形で返していければと思います。

急なお願いにもかかわらずアンケートにご協力いただいた、西中先生・春藤先生・岡田先生を始めとした回答者の方々にも心から感謝申し上げます。

執筆に際して、同研究室の仲間である梅ちなみさん、佐藤祥子さん、虫明奈緒さん、岸研究室の土居真也さんなど、様々な方に相談に乗っていただきました。特に土居さんには落ち込んでいる時に檄を飛ばしていただき、大きな支えとなっていました。重ねて御礼申し上げます。

最後に、挫けたときや弱気になっているときにいつも支えになってくれた福岡

にいる両親と姉に感謝いたします。

# 参 考 文 献


[1] 神谷加代. 子どもにプログラミングを学ばせるべき６つの理由　「２１世紀型スキル」で社会を生き抜く. 株式会社インプレス, 2015.

[2] 総務省. 「プログラミング人材育成の在り方に関する調査研究」報告書. 総務省, 2015.

[3] 一般社団法人ラーンフォージャパン. 「プログラミング教育実践ガイド」 (平成 26 年度文部科学省委託事業情報教育指導力向上支援事業). 文部科学省, 2014.

[4] 法林岳之. できるゼロからはじめるパソコン超入門　ウィンドウズ 10 対応. インプレス, 2015.

[5] パソコン用語研究会. よく解る！世界一やさしい　超パソコン入門用語. 株式会社レゴリスイノベーション, 2013.

[6] 若松和紀. 超入門　まったく分からない人のパソコン入門 Windows7. 西東社, 2010.

[7] 文部科学省. 「諸外国における プログラミング教育に関する調査研究」 (文部科学省平成 26 年度・情報教育指導力向上支援事業) 報告書. 文部科学省, 2015.

[8] 松井良平, 加藤優一, 伊藤稔. 協同学習を取り入れた出前授業の改善―レゴロボットによるプログラミング体験―. 2012.

[9] 田口浩, 糸賀裕弥, 毛利公一, 山本哲男, 島川博光. 個々の学習者の理解状況と学習意欲に合わせた プログラミング教育支援. 情報処理学会論文誌, Vol. 48, No. 2, 2007.

[10] 西村晃一, 皆月昭則. プログラミング言語の学習家庭における継続性の差異に関する一考察. 情報処理学会第 73 回全国大会, 2010.

[11] 梶浦文夫. プログラミング教育におけるプレゼンテーションの効果. 2001.

[12] 大場みち子, 伊藤恵, 下郡啓夫. プログラミング力と論理的思考力との相関に関する分析. 情報処理学会研究報告, 2015.

[13] 荻嶺直孝, 森山潤. 中学校技術科「プログラムによる計測・制御」の学習指導に関する実践研究の展望と課題. 学校教育学研究, Vol. 26, pp. 83–94, 2014.

[14] 文部科学省. 学校における教育の情報化の実態等に関する調査結果. 文部科学省, 2015.

[15] 吉永卓矢, 脇田建. ビジュアルプログラミング言語とその実行・開発環境の提案. 情報処理学会第 73 回全国大会, 2011.

[16] 阿部和宏. 小学生からはじめるわくわくプログラミング. 日経 BP 社, 2013.

[17] キャロル・ヴォーダマンほか [著]・山崎正浩 [訳]. 10 才からはじめるプログラミング図鑑. 創元社, 2015.

[18] 総務省. 平成 26 年度版情報通信白書. 総務省, 2014.

[19] ICT ドリームスクール懇親会事務局. 教育・学習分野の情報化に係る 国内外の動向と先進事例. 三笠書房, 2014.

[20] 山内祐平, 森玲奈, 安斎勇樹. ワークショップデザイン論ー創ることで学ぶー. 慶應義塾大学出版会, 2013.

[21] 美馬のゆり, 山内祐平. 「未来の学び」をデザインする　空間・活動・共同体. 東京大学出版会, 2005.

[22] 横山監, 高籔学. Scratch プログラミングを用いた金融教育の試み. ＰＣカンファレンス論文集 2011, 2010.

# 付　　録

## A. アンケート

Hour of Code 入門ガイド
アンケート用紙

この度は調査にご協力いただき誠にありがとうございます。
下記の内容に沿って、アンケートにお答えいただけると幸いです。
ご氏名：
ご回答年月日：

**【調査の背景】**

小学校におけるプログラミング教育の導入を目的とした、授業案と先生向け事前ガイドのデザインを行っています。このアンケートはそのガイドと授業案を読んだ上で、その有効性(わかりやすさ、時間的負担など)を測るためのものです。

**【お願いしたいこと】**

送付してある”Hour of Code 入門教材”フォルダ（”指導ガイド”、”授業カリキュラム”、”授業カリキュラムワークシート”）につきまして、お時間の許す範囲で、以下のことを実施し、評価していただけると幸いです。

＜指導ガイドについて＞

“指導ガイド”をお読みいただき（可能であればパソコンを実際に動かしていただき)、以下の質問項目にご回答ください。

① 指示のわかりやすさを１０段階評価(1:何をすれば良いかわからなかった～10:大変よくわかる)で表すと何点でしょうか？：＿＿点
② 事前知識編の用語の理解度を１０段階評価(1:覚えていない～10:すべての用語を簡潔にまとめられる)で表すと何点でしょうか？：＿＿点
③ 全体を読み終わるのに何分ほどかかりましたか？：＿＿分
④ (教員の方のみお答えください)他の授業準備もあるなかで、このガイドを読むことの時間的負担感は、１０段階評価(1:全く負担にならない～10:負担が大きすぎる)で表すと何点でしょうか？：＿＿点
⑤ 良いと感じた点(自由記述)：
⑥ 悪いと感じた点(自由記述)：
⑦ その他気づいたことや感想(自由記述)：

＜授業カリキュラムについて＞

“授業カリキュラム”をお読みいただき、以下の質問項目にご回答ください。
＊量が多いため、すべてを丁寧に読まなくても結構です。まずは全体にざっと

Hour of Code 入門ガイド
アンケート用紙

目を通していただき、いくつか気になった授業について重点的に読んでいただけると幸いです。

⑧ 指示のわかりやすさを１０段階評価(1:何をすれば良いのかわからない〜10:大変よくわかる)で表すと何点でしょうか？：＿＿点

⑨ 日本語のわかりやすさを１０段階評価(1:不自然で理解できなかった〜10:自然で簡単に理解できた)で表すと何点でしょうか？：＿＿点

⑩ １つの授業案をしっかり読み終わるには何分ほどかかりましたか？：＿＿分

⑪ (教員の方のみお答えください)他の授業準備もあるなかで、このカリキュラム案を用いた際の準備にかかる負担感は、１０段階評価(1:全く負担にならない〜10:負担が大きすぎる)で表すと何点でしょうか？：＿＿点

⑫ 良いと感じた点(自由記述)：

⑬ 悪いと感じた点(自由記述)：

⑭ その他気づいたことや感想(自由記述)：

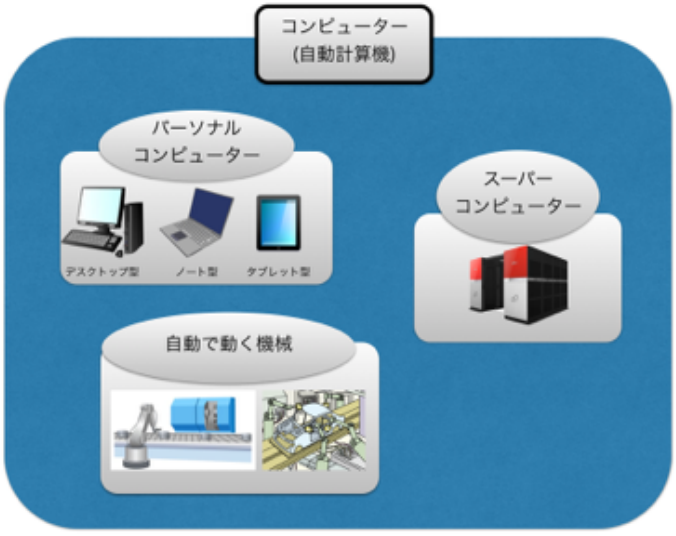
# B. 入門ガイド

# Hour of Code実施に向けた プログラミング教育入門ガイド

**概要：**
このマニュアルはHour of Codeの授業を行おうとしている教育事業者・教員を対象としています。既存のHour of Codeサイト上の授業計画は、子ども達にいかにわかりやすく伝えるかに留まり、教育者への説明が足りていませんでした。そこで今回、これからHour of Codeを教育に使おうとしている方の実践に役立つマニュアルを作成しました。
本ガイドは大きく３つ、導入編・事前知識編・準備編で構成されています。
導入編では、プログラミングとは何か？からプログラミングを学ぶ意義、そして本マニュアルの位置付けなどのプログラミングを教える前段階にあたる知識をまとめています。
事前知識編では、後の授業計画で教える内容について俯瞰します。子どもに何を教えているのかを前もって理解することを目的としています。
準備編では、Hour of Codeに教師として登録する方法など、授業実施前の準備について述べます。
すべて完了したら、別添ファイル「授業案日本語版」に移動して授業を始めましょう！

**<導入編>**
**Q1：プログラミングとは何？**
”プログラミング(programming)”とは、”コンピューターへの命令を作ること”です。コンピューターへの命令をプログラム(program)と呼び、プログラム(program)すること(ing)がプログラミング(programming)です。

ではコンピューターとは何でしょう？
**コンピューターとは、”自動計算機”のことです。**言い換えると、”指示を与えれば自動で計算や処理をしてくれる機械”です。機械にコンピューターが組み込まれることにより、様々な機械が自動で動いてくれます。
ちなみに混同しがちな言葉としては、”パソコン”があります。これはパーソナルコンピューターの略であり、コンピューターのなかでも個人が扱い易い価格や用途のものを指します。工場の組み立て機械や研究施設の大型コンピューターなどは、これには当てはまりません。概念図としては以下のようになります。(とても簡略化してありますので、参考程度にしてください)




“コンピューター”の概念図

こう考えると、身の回りはコンピューターで溢れていると言えます。自動ドア・信号機・携帯電話・自動車などなど…
こうした機械が”どんな動きをすればいいのか？””どんな計算をすれば良いのか？”という”指示”を作ることがプログラミングと言えます。

**Q2：なぜ子どもにプログラミングを教えるの？**
大きく以下の２つの理由があります。
・子どもの能力を伸ばす
　-論理的思考力　プログラミングでは、口頭で人に指示をする場合と違い、機械相手なので曖昧さが通用しません。そのために命令は必ず論理的である必要があります。
　-問題解決力　論理的な命令を作ったと思っても、思い通りに動かないことが頻繁に起こります。その際に、なぜダメだったかを考え、改善し、試すという、試行錯誤のサイクルを何度も繰り返すことで、問題に取り組み続け、解決する力が養われます。
　-楽しい！　プログラミングでは、自分の考えをその場で形として見ることができます。このゲームのような環境は子どものモチベーションを維持することに大きく役立っています。

・可能性を広げる
　今の世の中はIT化が進み、日々の暮らしのなかでコンピューターに触れないことはない状況です。これからの時代を生きて行く子ども達にとって、こうしたコンピューターの仕組みや扱い方を理解していることは、技術者にならないとしても、とれる選択肢を大きく広げると言えます。

※詳しくは以下の参考資料をご覧ください
- プログラミング人材育成の在り方に関する調査研究報告書（p32・プログラミングに関する教育がもたらす効果）：http://www.soumu.go.jp/menu_news/s-news/01ryutsu05_02000068.html
- 子どもにプログラミングを学ばせるべき6つの理由 「21世紀型スキル」で社会を生き抜く：http://www.amazon.co.jp/子どもにプログラミングを学ばせるべき6つの理由-「21世紀型スキル」で社会を生き抜く-できるビジネス-神谷-加代/dp/4844338285/ref=tmm_pap_title_0

**Q3:教える内容は？どこまで教えるの？**
日本ではまだカリキュラムが存在しないため、イギリスで2014年度から始まった義務教育課程における「コンピューティング」のカリキュラムを参考に解説します。(次ページ)
イギリスのカリキュラムでは、基礎的な考え方から始まり、コンピューターサイエンスの入門段階まで育成を行います。
このカリキュラムの発達段階に合わせて、１～４の分けられており、**本マニュアルの提案する内容は、キーステージ１と２を対象としています。**すなわち、基礎的な考え方を把握したレベルと言えます。そのため、このマニュアルの授業では、完全にオリジナルなアプリケーションを作成したり、好きなゲームを作るといったことまでは扱いません。

| キーステージ | 学習内容 |
| --- | --- |
| 1(小学校1年〜2年) | . アルゴリズムの理解や実装の課程、プログラムが正確な命令を実行することを知る<br>. 簡単なプログラムの作成、およびデバッグ<br>. 簡単なプログラムの動きを論理的に推測する<br>. 情報技術を用いたデジタルコンテンツの作成、保存、整理<br>. 学校外での情報技術の用途を知る<br>. 個人情報を公開せず、オンラインのコンテンツは大人の保護のもと扱う |
| 2(小学校3年〜6年) | 具体的な目的を達成するプログラムの設計・実装・デバッグ<br>人事処理、条件分岐、繰り返しをプログラムで実現、変数やさまざまな入出力を扱う<br>簡単なアルゴリズムを理論的に説明し、実装したプログラム内のエラーを認識・解決できる<br>コンピューターネットワークの仕組みを理解し、実社会でどのように利用されているかを知る<br>検索技術や検索結果を学び、よりデジタルコンテンツの評価を適切にできるようになる<br>さまざまなソフトウェアを使って多様なデジタル課題を解決できるようになる<br>情報技術の安全な利用を理解する。不適切な行動を認識し、報告ができるようになる |
| 3(中学校1年〜3年) | 現実の問題や物理現象の動き<br>状態を抽象化したモデルの設計・利用・評価<br>並び替えや検索など主要なアルゴズムの理解。複数のアルゴリズムの比較<br>２つ以上のプログラミング言語の理解(最低１つはテキストベース)<br>簡単な論理演算とこれらがカイロやプログラムにどのように使用されているかの理解<br>コンピューターがハードウェアとソフトウェアから成り立っていることの理解<br>命令がどのようにコンピューター上に蓄えられ、実行されるかの理解<br>複数のアプリケーションを使って、課題解決を目標としたプロジェクトを実施する<br>信頼性や操作性など使う人のことを考えてデジタルコンテンツを作成・再利用・編集できる<br>個人情報やプライバシーの保護を意識し、情報技術を安全かつ責任を持って利用出来る |
| 4(高校1年〜2年) | すべての生徒が情報技術やコンピューターサイエンスを多様に学べる環境を与える<br>コンピューターサイエンス、デジタルメディア、情報技術に関する技能と創造性、知識の強化<br>分析能力や、問題解決能力、設計能力、論理思考を成長させ、問題解決に適用できる |

**Q4:どうやって教えるの？**

Hour of Code(アワーオブコード)というプログラミング学習ツールを主に使います。Hour of Codeとは米国の非営利組織Code.orgが開発したプログラミング学習ツールであり、ネット環境があればどこでも受けられる点や、初学者にもわかりやすいようパズルのようなプログラミング方法が特長です。このツールは180カ国以上で使われ、オバマ大統領やfacebook創設者のマーク・ザッカーバーグ氏が支援ビデオに出演するなど、大きなムーブメントとなっています。

Code.orgはこの学習ツールを用いた全２０時間の授業案を作成しており、本マニュアルでは、その日本語訳と解説を追加しました。その授業の構成としては、”パソコンを使わないアクティビティ”→”パソコン(Hour of Code)を使った実習”の繰り返しで進んでいきます。これは、概念の理解→練習による定着を狙いとしています。授業案の１ページ目には授業全体の流れを掲載していますので、参考にしてください。

**Q5:とは言ってもプログラミングについて何も知らないんだけど？**

授業を実施するにあたり、次の＜事前知識編＞で、授業に登場する用語を説明しています。子ども達にこれから何を教えようとしているのかを把握してほしいという狙いで作成しました。あくまで基礎レベルですので、軽い気持ちで目を通してください。

**<事前知識編>**

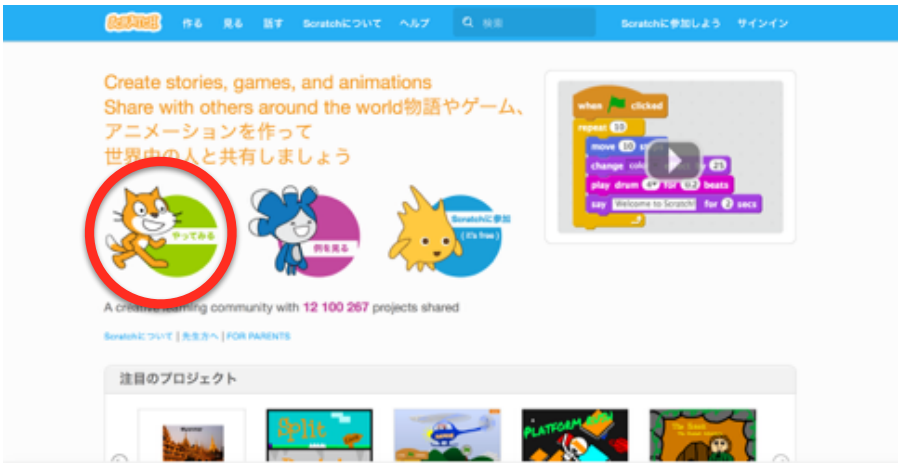
本章では、事前知識編として、授業で子ども達に教える内容を概観できるようにまとめました。なお、本章では言葉による解説だけでなく、実際にプログラミングをすることで用語の理解を深めるとともにHour of Codeのプログラミング方法についても勉強します。ここではScratchというプログラミングツールを使います。Scratchを使う理由は、Hour of Codeは教材として完成している一方で自由度が低いため、”概念を理解するためにちょっと触ってみる”ということがしづらいためです。また、操作画面もビジュアルプログラミング言語という、Hour of Codeとほぼ同じものを使っているため、子ども達と同じ操作感覚を身につけることができます。

**Scratchとは**

Scratchは、MITメディアラボのライフロングキンダーガーテンという研究チームが開発したビジュアルプログラミング言語です。世界中のプログラミング教育の現場で広く使われているツールで、登録ユーザー数が約720万人もいます。対象年齢は8歳からとなっていますが、マウスと簡単な入力操作ができれば小学1年生や幼児からでも楽しめます。操作も簡単で、「命令」「制御」「変数」などのカテゴリからブロックを選び、ジグソーパズルのように組み合わせることでプログラムができます。

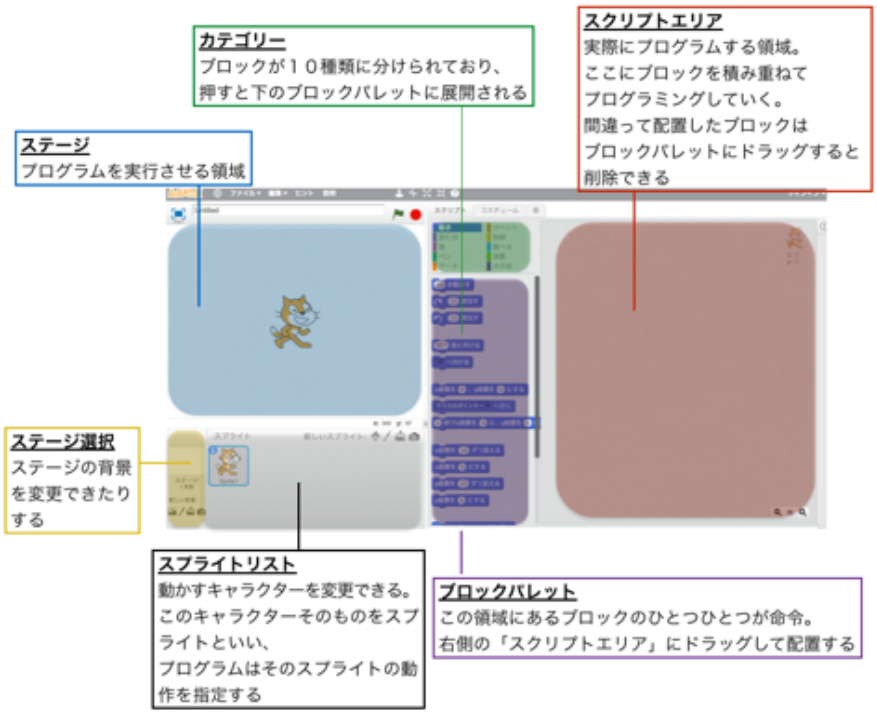
**Scratchを準備しましょう**

①ScratchのWebサイト(https://scratch.mit.edu)にアクセス
②左側の”やってみる"をクリック

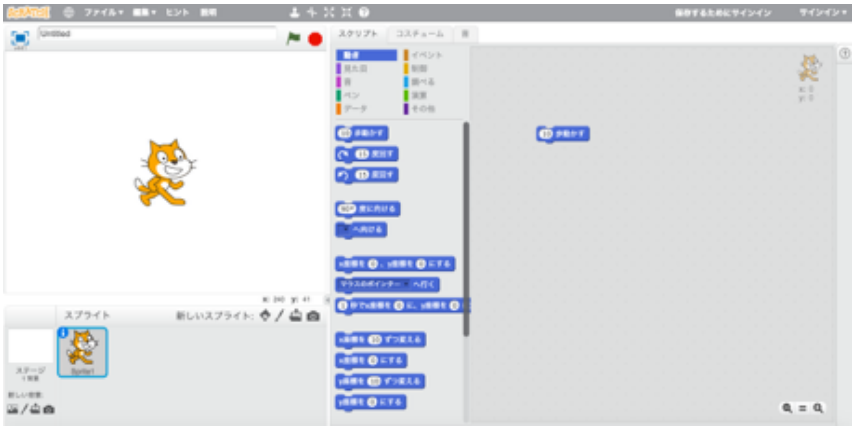



**Scratchの基本操作**

この画面の構成は以下のようになっています。



では実際に触ってみましょう！
真ん中のブロックパレットの中にあるブロックは、ひとつひとつがプログラム(命令)です。



ブロックの中から”10歩動かす”をドラッグして右側に乗せてみましょう
そして、その”１０歩動かす”をクリックしてください
そうすると、ネコが右に少し動いたのがわかりましたか？
わからなかったらもう何度かクリックしましょう。

これでいま、あなたはネコを10歩動かすプログラミングをしたことになります！

**操作方法の補足(進めながら参照してください)**

- ブロック同士をくっつけるには、まずつけたい元のブロックをドラッグして、つける先のブロックの下でドロップしてください。つく際には以下のような白い線が表示されますので、そのままドロップすればOKです。

- また、ブロックを切り離したい時には、取りたいブロックをドラッグします。なお、ドラッグしているブロックの下に別のブロックが付いている場合、それも付いてくるという点に注意してください。

- いらないブロックを削除したい場合は、いらないブロックをドラッグして、真ん中のブロックパレットにドロップして削除します。

- 白で囲まれた数字や文字は書き換えることができます。例：「10」歩動く→「50」歩動く

- 動きを止めたい場合は、真ん中上にある赤い丸印を押します。

**Hour of Codeで扱う用語の理解**
**アルゴリズムとは、**
**→命令の集まり**

例えば、”10歩動かす”の下に、今度は”見た目”パレットの中にある”Hello！と２秒言う”をつけてみましょう。

ついたら、どちらでも良いのでブロックをクリックしてみましょう。
すると、ネコが１０歩動いてからHello！と言いました。

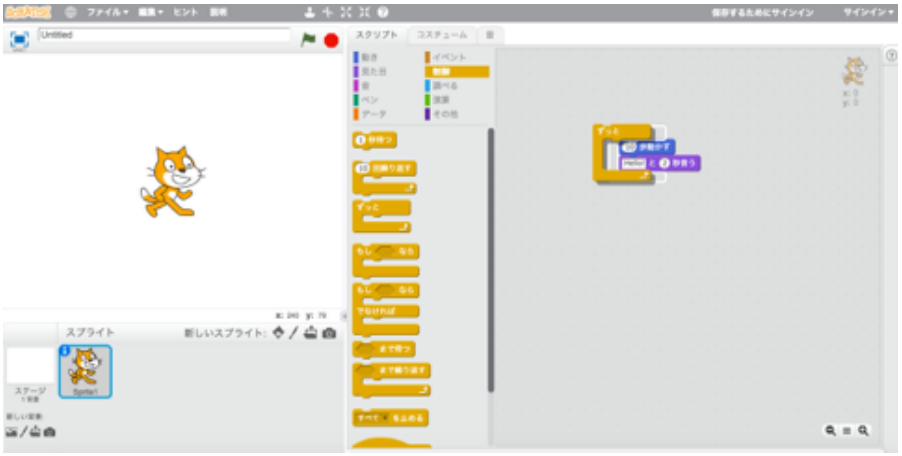
Scratchに限らず、プログラミング言語は、基本的に上から順につながったものが実行されていきます。
この”つながった命令のかたまり”をアルゴリズムと言います。

**ループとは**
**→命令の一部または全体を繰り返し行い続けること**

先ほどの命令にループを組み込んでみましょう。
まず真ん中の”カテゴリー”から”制御”を選択してください。

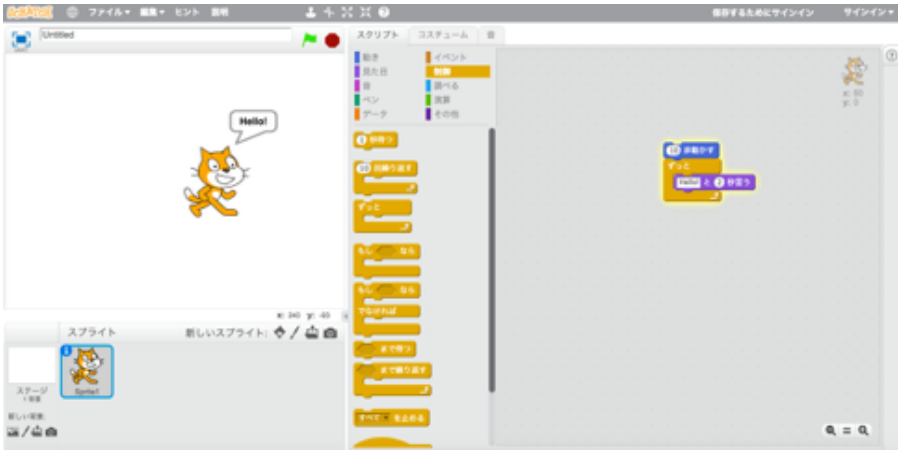
“制御”のなかには、”ずっと”や”もし”など、”動きや見た目を、どういう順番で進めていくか”に関係する命令が入っています。

では、”ずっと”を選んでみましょう。逆コの字型になっていると思いますが、これは”繰り返す範



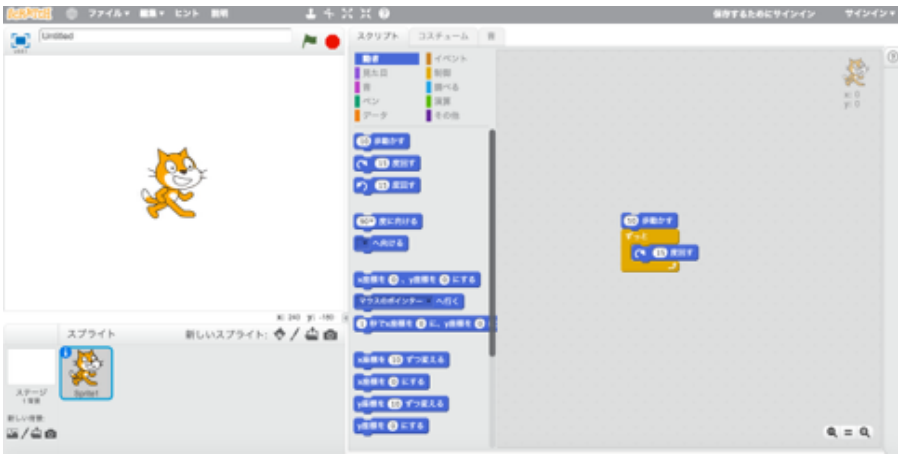
囲を挟んで指定する”ためにこの形になっています。クリックしたら、押したまま２つの命令の横までドラッグしてください。（上画像を参考に！）

そして以下のように命令全体をはさむこと(ループさせること)ができれば成功です。クリックすると、”１０歩動く→Hello!と２秒言う”を繰り返し行っています。
もしHello!と2秒言うだけを”ずっと”ではさむとどうなるでしょうか。１０歩動いたあと、”ずっと”Helloと２秒言い続けていると思います。




**デバッグとは**
**→思った通りに動かなかった際に、間違いを見つける作業のこと**

簡単に例を示すと以下のようになります。
やりたいこと：「ネコが大きな円を描いてぐるぐる回り続けるようにしたい」
間違ったプログラミング：以下の通り



実行結果：「その場でぐるぐると回り続けた」
何が間違いでしょうか？ぜひ自分で組み立ててみてください。
＊正解は本章の最後に記載しています。

**条件分岐とは**
**→ある条件に当てはまったら、特定の動きをする”制御”のひとつ**
※Hour of Codeでは条件が簡略化されているので、ここでは感覚的に理解できればOKです。

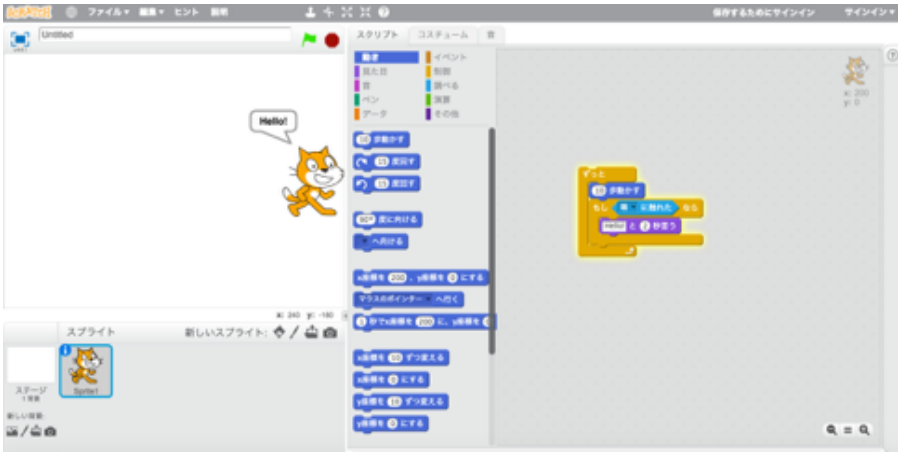
先ほどの「１０歩動かす→hello!と2秒言う」を改造して、
「歩き続けて、もし画面端についたら、hello!と2秒言う」にしてみましょう。

このためには”もし〜なら”ブロックを使います。
この”〜”の部分はひし形みたいになっていますね。ここに条件を当てはめます。

ここでは”調べる”カテゴリーの「〜に触れたら」を選びます。このとき”〜”部分をクリックすると選択肢が出ますで、ここでは”端”を選びましょう。

クリックして以下のように、

端についたときに、Hello!と言えば成功です。



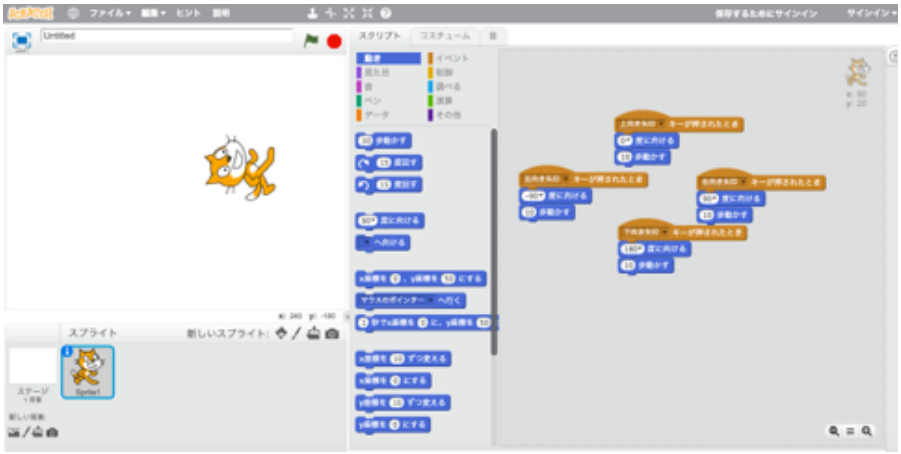

※この条件分岐が使えると、できることがぐっと広がります。Hour of Codeは基礎だけなので、他の書籍などで詳しく学ぶと(そんなに難しくありません！)、自由制作をする際などは大きく役に立ちます。

**イベントとは**
**→プログラムを始める”きっかけ”となる条件**

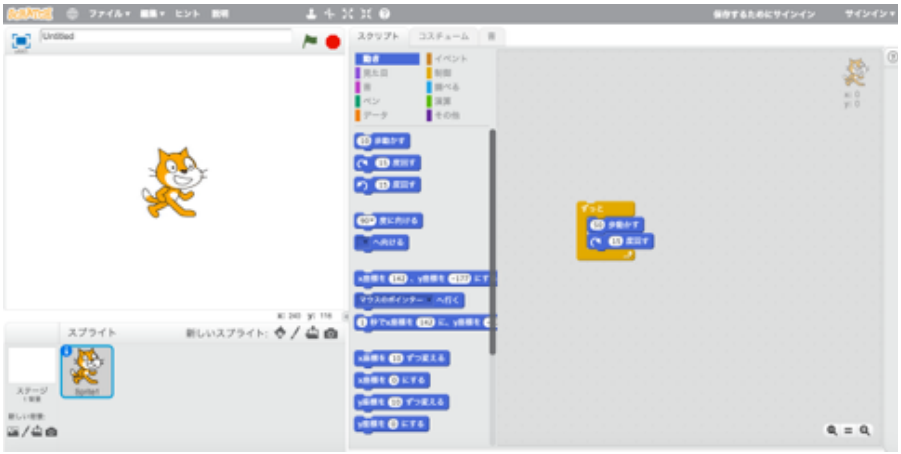
これまではクリックしてプログラムを動かしていました。これを、他のきっかけで動かすことができます。このきっかけをイベントと呼びます。

これを使うと、例えばネコをキーボードの上下左右キーで動かすことができます。
“イベント”カテゴリーにはいくつものきっかけとなるイベントが入っていますが、ここでは"〜キーが押されたときを選びます。そして上下左右のキーについてそれぞれ以下のように配置してみましょう。成功すれば、上下左右キーが押されたとき、そのアルゴリズムが動くようになります。なお、

わかりやすいようにこのような並べ方にしましたが、どのように配置しても動作は変わりません。



＊デバッグの答え
繰り返したい命令は”ずっと”の中に入っている必要があります。



**<準備編>**

授業を始める前に、Hour of Codeの操作に慣れておきましょう。このパートでは、以下の３点を解説します。

・ Hour of Codeの操作練習
・ 授業前のセッティング方法
・ 授業の始めにやること

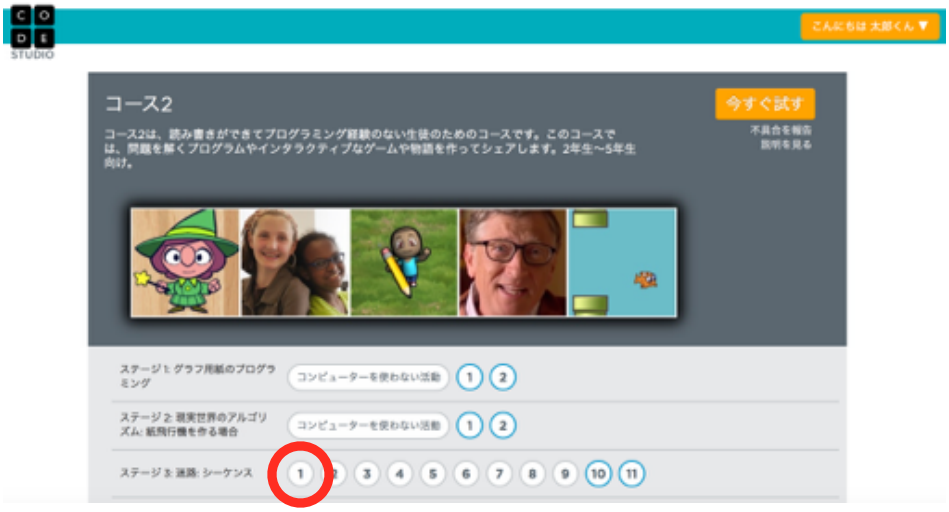
【Hour of Codeの操作練習】

事前準備編の経験をもとに、生徒が実際に扱うHour of Codeに慣れておきましょう。

①まず、Code.orgにアクセスし、”生徒"を選択します

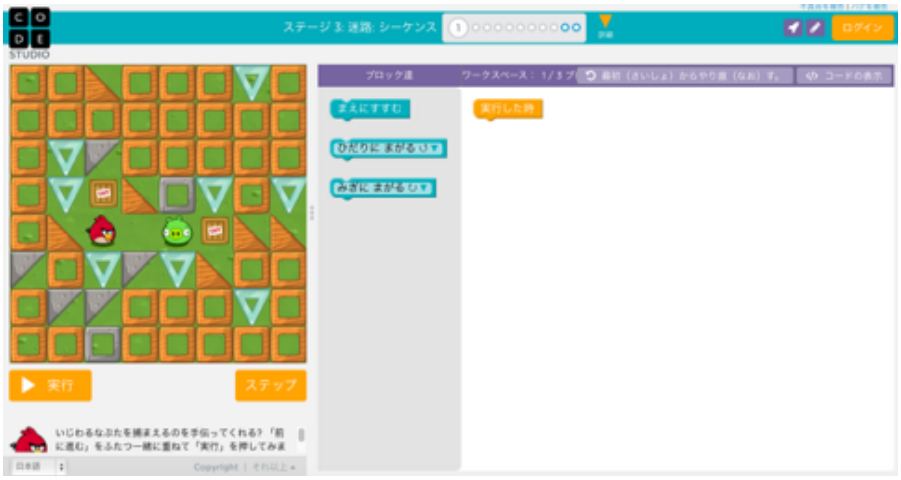

②そして、”コース２”を選択します。

③以下のように、授業に沿った項目が出てきます。今回はステージ３の１を選択しましょう。



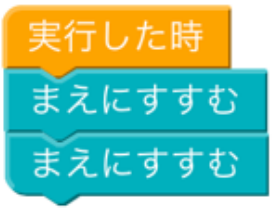

④選択すると、自動的に概要を説明するビデオが流れますが、英語なので閉じてしまってかまいません
⑤最初に問題が提示されます。内容を確認したらOKを押しましょう。

⑥以下のように、実際に問題を解いていく画面が開きました。




⑦各要素を解説すると以下の通りです。

⑧今回は以下のようにプログラムを組んで、実行ボタンを押すと正解となり、次の問題に進みます。
やってみましょう！




⑨これを繰り返して、学習を進めていきます。

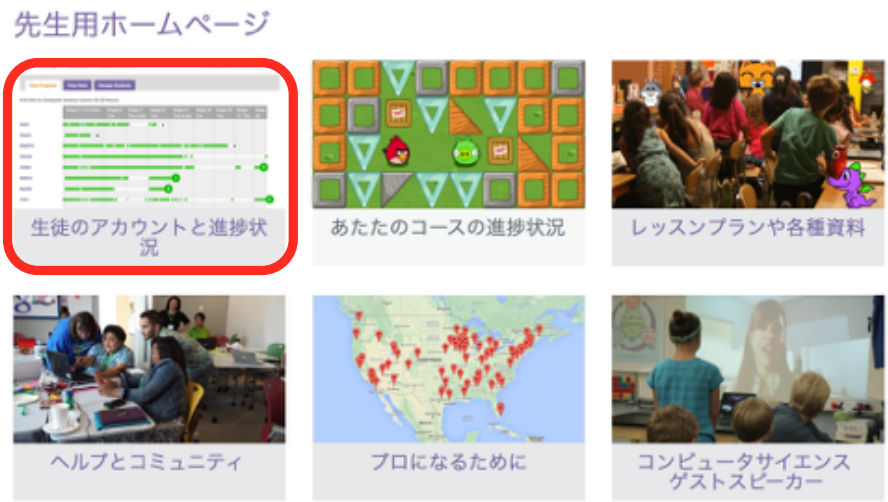
【授業前のセッティング方法】
授業を開始する前に、まず先生がHour of Codeに登録して、グループを準備をする必要があります。その後、生徒たちがクラスのグループに登録することで、先生が生徒の進捗を管理できるようになります。
以下に詳細な手順を説明します。

＜先生の操作＞
① Code.org（https://code.org）にアクセスする
　＊もし日本語になっていなければ画面一番右下から言語を選択することができます

② 画面下の教育者向けページにアクセスする

③　必要事項を記入して、先生としてCode.orgに登録する


④　「生徒のアカウントと進捗状況」で、各課題について各生徒がどこまで進んでいるか、などを管理することができます。



⑤　新しくグループ(セクションと呼ばれています)をつくるには、左上の「新しいセクション」を選択します。そうすると、記入画面がでるので、以下のように入力して、保存を押します。（授業名は自由に書きましょう）。※学年は、年齢と読み替えてかまいません


⑥　グループが完成しました！


⑦　ここに参加する生徒を登録していきます。「生徒の管理」を選択します

⑧以下のような画面になると思います。「生徒を追加」を押します

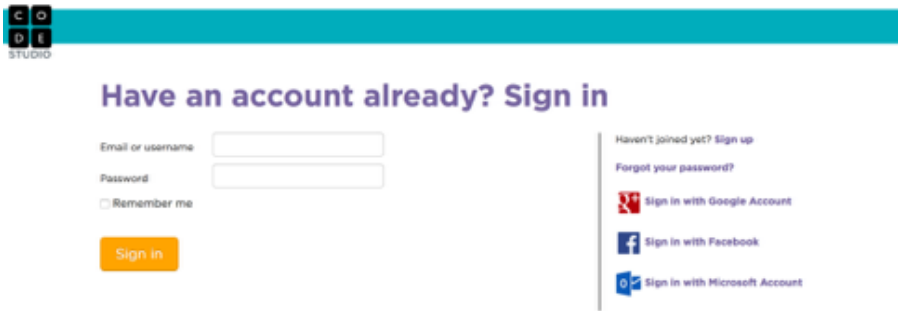

⑨　以下を参考に、生徒の情報を入力して、「保存」を押します。

10.　「秘密の表示」を押して秘密の言葉を表示させます。これが子どものログイン用パスワードになります。この「秘密の言葉」と「右下のログイン用アドレス」を子どもに伝えましょう。


＜子どもの操作＞
①　先生から共有されたリンクにアクセスする
②　自分の名前を選択して、先生から共有された「秘密の言葉」を入力してログインします。


③以下の画面になればログイン成功です！

【授業における使い方】
①教師としてHour of Code(https://studio.code.org/users/sign_in)にログインする



上の欄には、登録したメールアドレス、
下の欄には、パスワードを入力します

②生徒に前述の方法でグループにログインさせる
③授業を行うステージの１問目(①)を選択させて、解かせ始める

これらを確認したら、別添の「授業案日本語版」を使って、授業を始めましょう！！