指導教員：来住伸子　教授

Scratchプログラムの可視化による類似度推定
可視化によるScratchプログラムの理解支援

G13908　岩科智彩
G13924　森下汐美

平成29年1月13日

近年プログラミング教育の推進に伴い、義務教育化が進んでいる。その中で米国マサチューセッツ工科大学のメディアラボが開発したScratchは無償で提供されているグラフィックプログラミング環境である。プログラミングを行う際の命令を本ツールではブロックを組み合わせて作り上げる。初心者にとっては使いやすい構造となっているため米国では利用が増えているものの、日本のユーザーは全体の1%にも満たない。そこで実際に本ツールで公表をされているデータを利用してより教育に用いられるツールの解析を目指す。本ツールですでに

1. 各ブロックの種類が使われている全体での割合
2. あるプロジェクトが他ユーザーの引用関係を示したツリー構造

が公表されている。しかしこのデータでは全体図の把握が可能であるが1つのプログラムでブロックがどのように使われているか、引用していた場合引用元からどの程度変更させたかは不明である。従って本研究では1つのプログラムで使用されているブロックを解析し結果を出すと同時に引用元との比較を行い、関係性を導き出す。

目次

第I部

序論

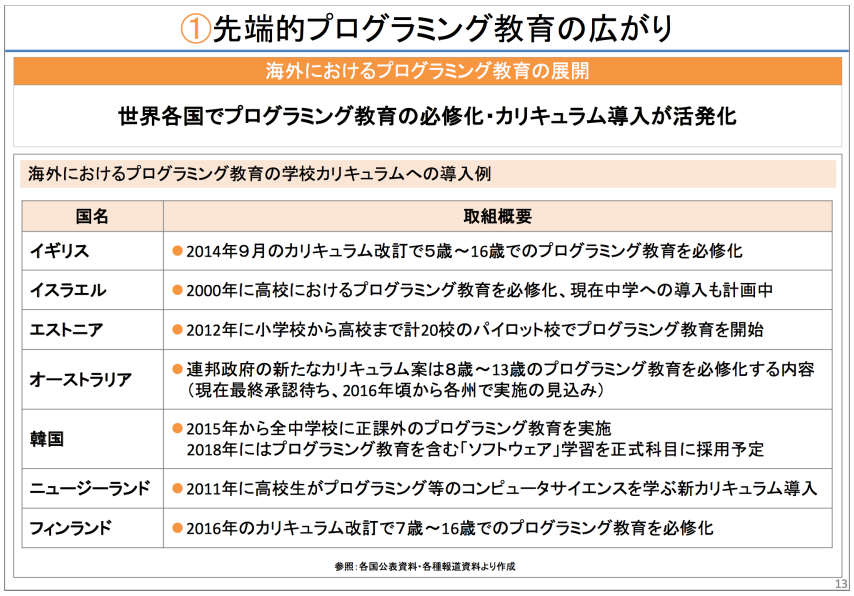
第1章

はじめに

1.1　本研究の背景

2006年に開発されたScratchというビジュアルプログラミング環境がある。本研究ではScratchのデータを用いて教育の場で役立てる方法を導き出す。Scratchとは米国マサチューセッツ工科大学のメディアラボが開発したフリーツールであり、キーボードでコードを打つのではなくパズルのようにブロックを組み合わせてプログラムの完成させる。近年、世界ではIT化が進み、物とインターネットを繋ぐIoTが次々と増えていく中エンジニア不足が叫ばれている。しかし世界的に見るとプログラミング分野では日本は遅れている。総務省平成２６年度「教育・学習分野の情報化に係る国内外の動向と先進事例」では世界各国でプログラミング教育が活発化していることが分かる(資料itedu)。日本の旧学習指導要領の必修科目では、パドコンでワードやエクセルの使い方の教育しか行っていないが、イスラエルではコンピュータの使い方よりも原理やプログラミングを教える教育が行われている。そのため文部科学省が2016年5月19日に発表をした成長戦略素案では第４次産業革命を支える人材を強化するため情報活用能力を伸ばそうと2020年度から小学校、2021年度に中学校、2022年度には高校でプログラミング教育での必修化の予定されている。それに先駆けて国内の学校では実際にプログラミングを行う授業も増えているという([2])。品川区立京陽小学校では2014年度よりScratchが導入され各教科の中で活用されている。例えば理科の授業で実験結果をシミュレーションするプログラムの作成にScratchが用いられているようだ。

Scratchでは全世界のユーザーのうち日本でのユーザーは現状で１％（資料1.2）である。Scratch公式では利用者の数、年齢、使っているブロック等の統計データの公表があるが、Scratch自体を題材にしている先行研究は日本では数が多くない。そこで義務教育化に伴い日本でも教育現場でScratchを導入しやすくするため公開されているプログラムやデータを利用して解析を行う。



①先端的プログラミング教育の広がり

海外におけるプログラミング教育の展開

世界各国でプログラミング教育の必修化・カリキュラム導入が活発化

海外におけるプログラミング教育の学校カリキュラムへの導入例

国名	取組概要
イギリス	●2014年９月のカリキュラム改訂で5歳～16歳でのプログラミング教育を必修化
イスラエル	●2000年に高校におけるプログラミング教育を必修化、現在中学への導入も計画中
エストニア	●2012年に小学校から高校まで計20校のパイロット校でプログラミング教育を開始
オーストラリア	●連邦政府の新たなカリキュラム案は8歳～13歳のプログラミング教育を必修化する内容（現在最終承認待ち、2016年頃から各州で実施の見込み）
韓国	●2015年から全中学校に正課外のプログラミング教育を実施 2018年にはプログラミング教育を含む「ソフトウェア」学習を正式科目に採用予定
ニュージーランド	●2011年に高校生がプログラミング等のコンピュータサイエンスを学ぶ新カリキュラム導入
フィンランド	●2016年のカリキュラム改訂で7歳～16歳でのプログラミング教育を必修化

参照：各国公表資料・各種報道資料より作成

13

図1.1　平成２６年度総務省 教育・学習分野の情報化に係る国内外の動向と先進事例

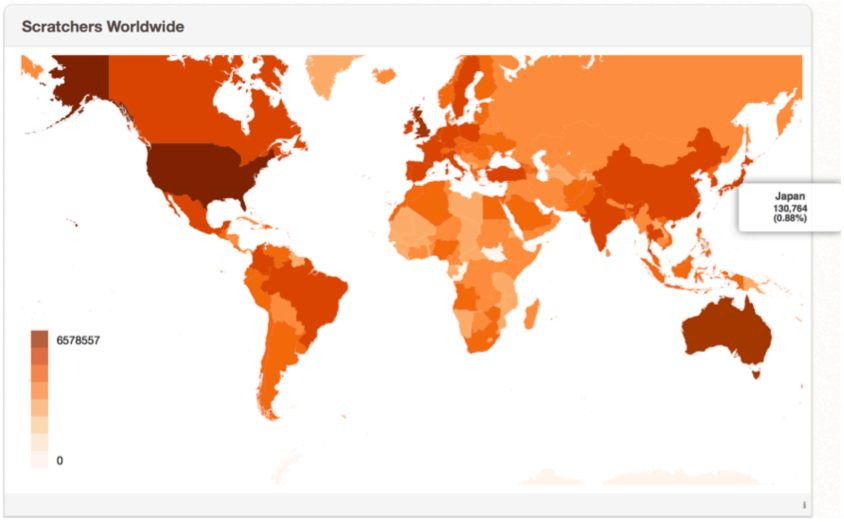


図1.2　全世界のScratchユーザー　日本　130,764人　0.88%

1.2　本研究の目的

本研究ではScratchで公開されているプログラム、統計データを利用して分析、実験を実施する。その中でも公開データであるリミックスツリーの精度を検証し、実際に使い易いデータの可視化であるかどうかを評価する。その結果を用いてプログラミング教育における課題点を解決を目指す。

第II部

本論

第2章

Scratchについて

2.1 Scratchとは

Scratchとは、プログラミングを学ぶ子どもや初心者が、構文の書き方や使い方を覚えることなく結果を得ることができる、プログラミング言語学習環境である。2006年にMITメディアラボのミッチェル・レズニックが主導するライフロング・キンダーガーデン・グループによって開発された。プログラミングを可能な限り簡単に学べるよう、視覚表現でキャラクターを動かすことができ、ビジュアルプログラミングとも言われている。

2.2 Scratchの使い方

Scratchの公式サイト([3])にアクセスしてオンライン上で使用することができる。(2013年5月にリリースされたバージョン2.0からウェブアプリケーションも出ている）またScratchのダウンロードサイト[4] よりパソコンにダウンロードすることで、オフラインでも楽しめる。ここではオンライン上での使い方を説明する。初めて使用するときはページ上部の「Scratchに参加しよう」というボタンをクリックし、ユーザ登録を行う。ユーザ登録を行うことでできあがった作品はオンラインコミュニティで他のユーザと共有することができる。　同じくページ上部の「作る」というボタンをクリックすると、開発環境(図2.1)が表示される。ページ左側が実行結果を表示し、右側がプログラムを書く部分である。左側の画面にいるネコはデフォルトで設定されているキャラクターで、スプライトと呼ばれている。このスプライト一つ一つにブロックを積んでいくことで実際に実行することができる。　プログラムの命令にあたるのが、ブロックである。ブロックは、動き・見た目・音・ペン・データ・イベント・制御・調べる・演算・その他の種類に分かれており、それぞれの機能に応じてブロックを組み合わせることができる。(図2.2)例えば、「画面上の緑の旗のマークが押されたときにスプライトが10歩動く」というプログラムを構成したいときは、イベントと動きのブロックを組み合わせることで、実行できる。　他のユーザの作品を見たいときは、「見る」というボタンをクリックする。アニメーションやゲームなど、さまざまなジャンルの作品を見ることができる。他のユーザの作品を見てみると、ページ下部の一番左に木のマークがあり、その横に数字が書かれている。

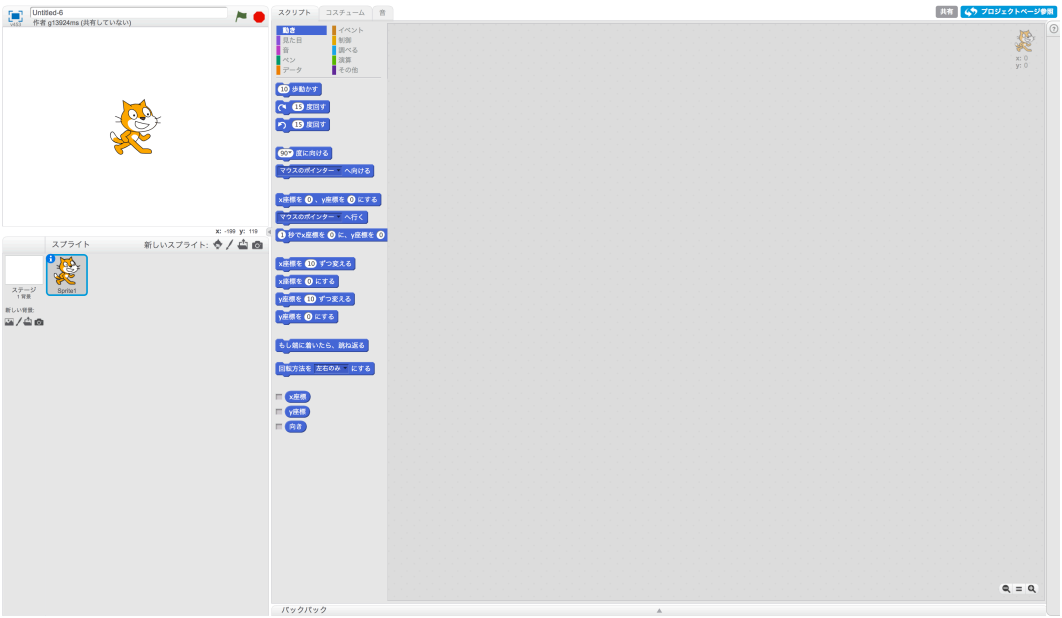


図2.1　Scratchプロジェクト作成画面

ここをクリックして開いてみると、ひとつの木が現れる。これはリミックスツリーというもので、このプログラムはどれだけのユーザが参考にしたのか、引用したのか、可視化されたものである。Scratchは簡単に他のユーザの作品をコピーすることができるが、それは誰かの作品を引用し作っていくことを大事にしているからである。このリミックスツリーはのちの研究の中で紹介する。

2.3　Scratchの特徴

Scratchの最大の特徴は視覚的な分かりやすさである。プログラムの構文や文法などを理解できなくても、ブロックの機能と適切なブロックを選び組み合わせることで簡単に実行させることができる。ブロックも機能ごとに色分けされ、役割を認識しながらプログラムを組み立てることができる。また繰り返しの構文や条件文などプログラム構文の基礎となるアルゴリズムの考え方もScratchを使用しているうちに少しずつ慣らしていくことができる。([5]) 作成できる作品の種類として、ゲームや動画制作、対話式のプログラムを得意とする。要因の一つとして、ビジュアルプログラミングであることが挙げられる。またScratchは一つのコミュニティサイトとしても見ることができる。公開した作品のユーザの反響の確認が可能であり、それを参考に改良したり、よりよいプログラムを見つけるとそれをそのままコピーするしたり、それを元にオリジナル作品を生み出すことができる。　一方、ブロックを使用することでプログラム構造の概念

図2.2　Pong Starterプロジェクト編集画面

は簡単に学ぶことができるが、実際のJavaやCなどのプログラミング言語の習得とは同様ではない。プログラム入門の一つの壁であるキーボード操作が分からない場合でも手軽に体験できるものであり、実際に言語を使って行うプログラミングとは異なる。またScratchを構成する元々要素が古く、加えてScratchの延長となるプログラミング言語がないことも理由に挙げられる。独立した言語であるために将来的な応用をつけるのは容易ではない。

2.4　Scratchの利用度

Scratchは元々8歳から16歳の子ども向けに作られたツールであるが、現在すべての年代のプログラミング初心者に使われている。Scratch公式サイト[6]によると150以上の国で使用されており、40カ国語以上に翻訳されている。ツール上で簡単に翻訳できるようになっている。また、学校での利用も多い。小学生から大学生まで幅広い年代の学生が情報の授業などで使用している。

2.5　本研究におけるScratchの利用意義

まず、学校教育でよく使われていることが挙げられる。日本ではまだ少ないが、世界的に見るとこのツールを使って授業をしているところが多く、これからの日本の情報教育の義務化にあたって、Scratchの機

能がこれからの日本の情報教育の方針に一番近いと考えられる。また、Scratchの特徴の一つに、他のユーザの作品を参考にするためにコピーが簡単にできるとあった。これは、他の作品で使われている新たなブロックの組み合わせ方や定義などを学び、自分の作品で使えるようにするためである。この仕組みが情報教育で役に立つと考える。そして、このツールは世界中のユーザが存在し、その分公開されているプログラムの数が多く、十分なデータを得ることができる、そしてそのデータがネット上で簡単に入手できるため本ツールを使用する。

第3章

Scratch公式可視化データの現状

3.1　ブロックの利用度

Scratch公式では無作為に選ばれたプログラムで使用されているブロックの種類を集計し、項目ごとに色を分けて利用度を可視化してある。高い割合の項目ほど面積が大きい。(図3.1) 項目をクリックすると利用度の高いブロックが同じように面積が大きく表示される。(図3.2) このデータではScratch全体で使われているブロックの利用度の把握が可能である。しかしユーザー個人個人が作成したプログラムのブロックの利用度が分かるということではない。

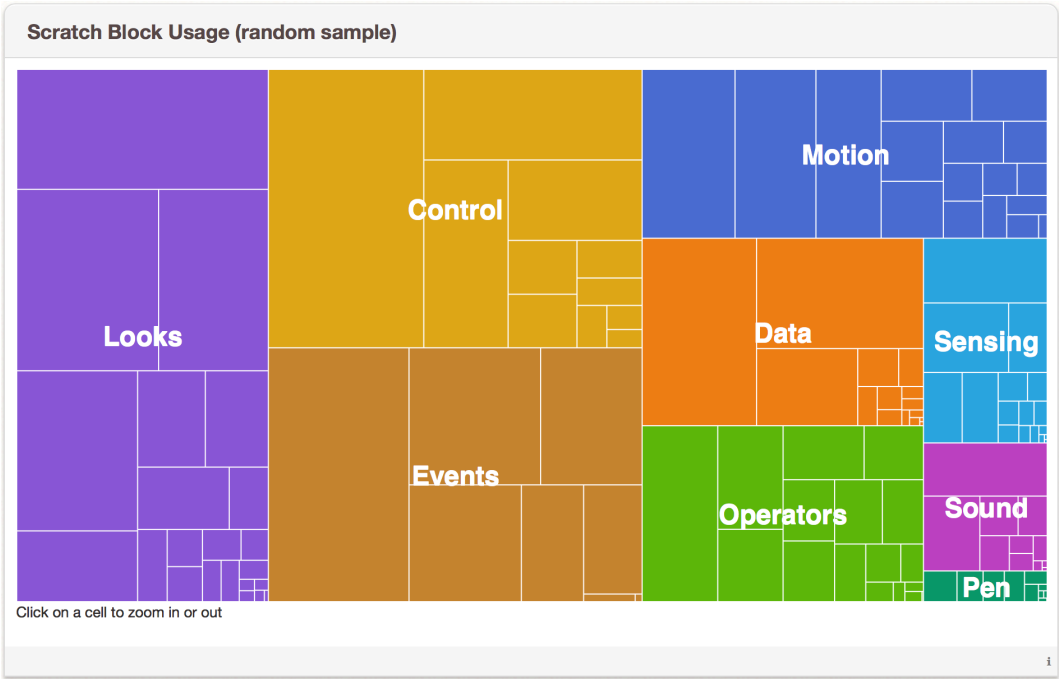


図3.1　全ユーザーブロック利用度

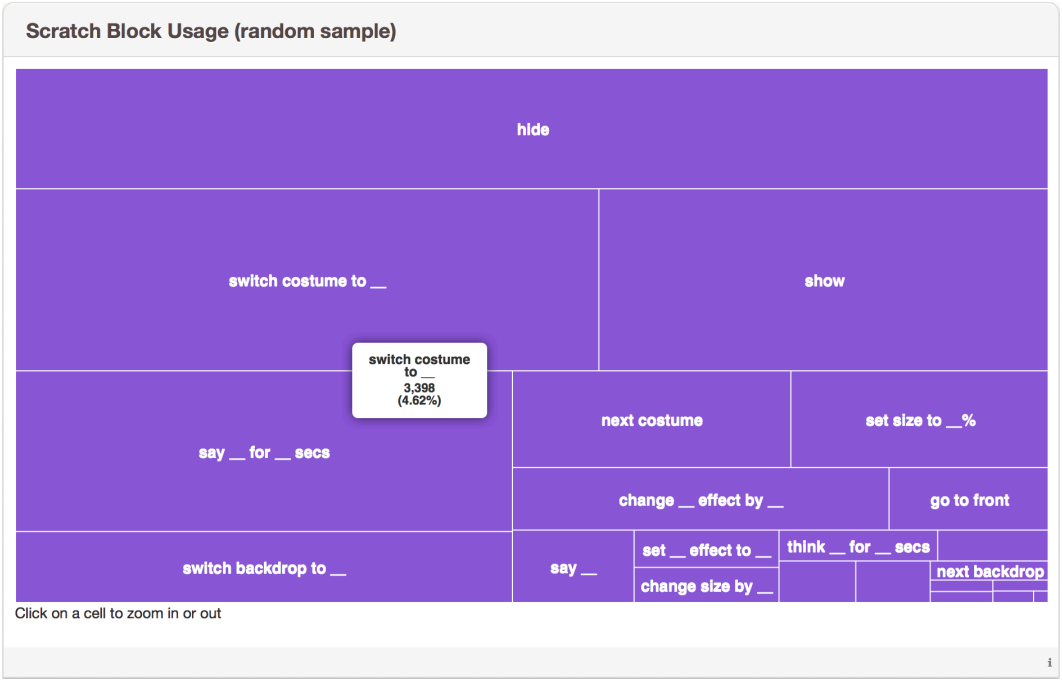


図3.2　全ユーザーブロック利用度　Locks（制御ブロック）をクリックした時

3.2　リミックスツリー

木構造になっているリミックスツリーは根元のプログラムを元として引用関係を表している。(図3.3)さらに引用されたプログラムを引用した場合も反映される。(図3.4)この図はどのくらい変更が加えられているかがわからない点から元のプログラムと先端のプログラムの差を把握することは難関である。

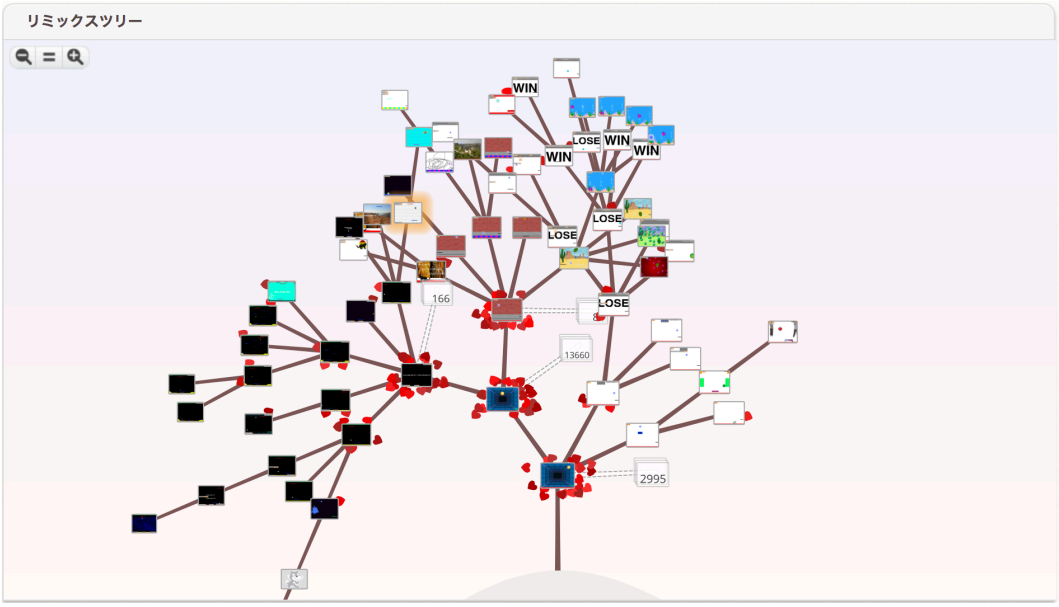


図3.3　Pong Starter のリミックスツリー

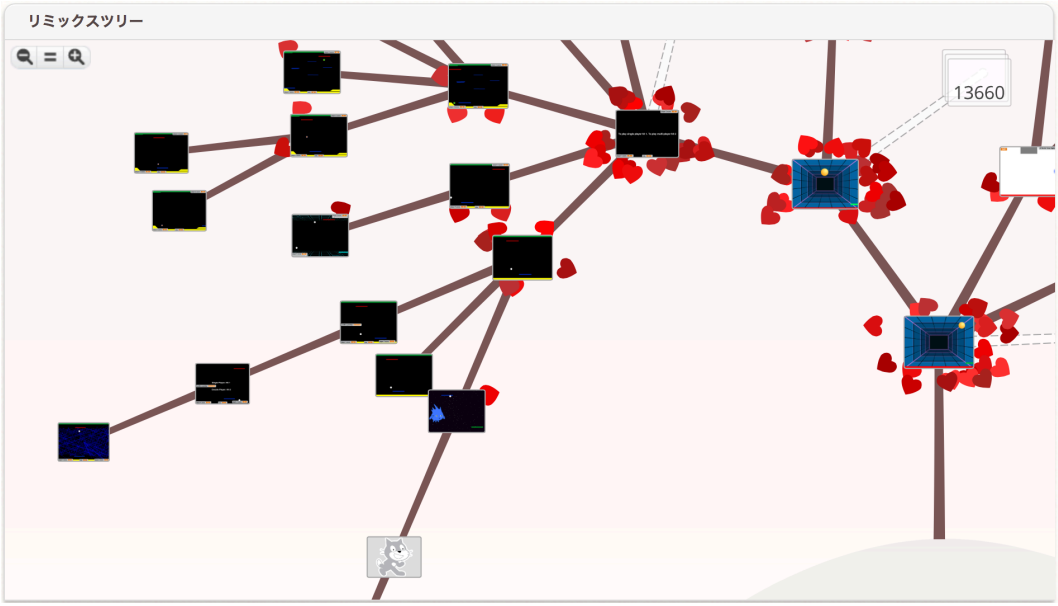


図3.4　Pong Starter のリミックスツリー　左下詳細

第4章

研究内容

4.1　問題提起

前章で挙げたリミックスツリーの可視化では元のプロジェクトから引用した新たなプロジェクトは引用したという結果が表されており、全て同じ距離になっている。(**??**) つまり何も変更が加えられてないプロジェクトとかなり改造されたプロジェクトの違いは可視化されない。学ぶ側はレベルの把握、教える側には評価をしやすいくするために変化の度合いの可視化を可能にする。

4.2　実験方法と仮説

本研究では 2 万弱の引用がされている”Pong Starter”プロジェクトのリミックスツリー([7])を利用する。まず元のプログラムから引用されたものを 1 段目とする。以後 1 段目のプロジェクトから引用されたものを 2 段目、2 段目から引用されたものを 3 段目とする。それらを順に元のプロジェクトと比較をしていき、使われたブロックの数、スプライト（オブジェクト）の数、類似度を表すcos類似度の数値化、グラフ化をする。リミックスツリーで表されているのは引用関係であるため、実際に 1 段目にあるものより 3 段目のものの方が元のプロジェクトと類似しているまた逆の結果が表れることが予測される。

4.3　技術の解説

4.3.1　使用するプログラミング言語

本研究ではScratch公式サイトよりプロジェクトをJSON形式でインポートし、Pythonを利用してデータを抽出、分析を行う。

Pythonとは

PythonはGuido van Rossum氏の作成した軽量プログラミング言語である。記述や取り回しが容易であり、コンパイルも不必要という特徴がある。([8])

ソースコード 4.1　C言語

```c
void main()
{
    printf("Hello World!");
    return 0;
}
```

ソースコード 4.2　Python

```python
print("Hello World!")
```

文法はC言語やJavaとは異なり、中括弧を使用せずインデントでのブロック構造になっている。これは誰がコードを書いても同じソースコードになることがわかり、可読性に優れている。このような習得のしやすさから教育の場では広く使用されてきていることも分かった。

JSONとは

JSONはJavaScript Object Notationの略であり、テキストベースのデータフォーマットである。表記の基はJavaScriptであるだけでそれ以外では、記述が容易であり可読性に優れているため他のプログラミング言語でも利用可能である。基本は数値とその名前（キー）をペアにしてコロンで対になっている。項目はコンマで区切られ、全体は中括弧で区切られる。([9])

PythonではjsonモジュールをインポートすることでJSON形式ファイルを読み込むことが可能であり、Pythonでは辞書型に保存される。([10])次節ではPython環境でのJSONデータの操作するプログラムの説明をする。

ソースコード 4.3　JSON

```
{
 "book1":{
"title": "Python Beginner",
 "year": 2013 ,
"page": 349
},
"book2":{
 "title": "JSON Beginner",
 "year": 2003 ,
"page": 640
 }
}
```

4.3.2　プログラムの解説

本研究では２つのプログラムと１つの辞書データを使用する。ソースコード4.5はブロック、スプライトの数、cos類似度を計算結果を出力する。ソースコード4.4はcos類似度を実施に計算するプログラムとなっている。共通するブロックの名前の個数である値の積を分子、値の自乗をブロックの種類分足してルートをかけたものをA,Bそれぞれ計算しかけたものを分母として、割り算を行い、cos類似度を求めている。

ソースコード 4.4　SimCalculator.py

```
1  # −∗− coding: utf −8 −∗−
2
3  # cos類似度
4  import math
5
6
7  class SimCalculator():
8      def sim_cos(self, v1, v2):
9          numerator = 0
10         #       v1とv2で共通するkeyがあったとき、その値の積を加算していく。2つのベクトルの内積になる。
11         i = 0
12         for n in v1:
13             if n in v2:
14                 numerator += v1[i]∗v2[i]
15                 i=i+1
16
17         ss1 = []
18         ss2 = []
19         for x in v1:
20             ss1.append(x∗x)
21         for x in v2:
22             ss2.append(x∗x)
23
24         denominator = math.sqrt(sum(ss1))∗math.sqrt(sum(ss2))
25
26         if denominator == 0:
27             return 0
28         return numerator / denominator
```

ソースコード 4.5　scratchAnalysis.py

```
# −*− coding: utf −8 −*−

'''
各モジュールをインポートする
JSONモジュール
URLLIB2モジュール
CSVモジュール
'''

import json
import urllib2
import csv

'''
SCRATCH_BLOCK.PYをインポートする
'''

import scratch_block


'''
COS類似度を計算するSIMCALCULATOR.PY_のインポート
'''
from SimCalculator import SimCalculator


'''
事前に作成した分析対象プロジェクトをリスト化した
CSVファイルの取得
'''
dataReader = csv.reader(open("PONG.CSV","RB"))


'''
ブロックの種類とその個数をカウントし、
SCRATCH_BLOCK.PYに保存をする
'''

def getFirst(L, dict):
    global count
    if isinstance(L, list)and len(L)>0 :
        first = L[0]
        if isinstance( first ,unicode):
            '''
␣␣␣␣␣␣␣␣␣␣␣例外ブロックの処理
␣␣␣␣␣␣␣␣␣␣␣'''
            if "TURNRIGHT:" in first:
                first = first.rstrip(":")
            if "TURNLEFT:" in first:
                first = first.rstrip(":")
            if "WAIT:ELAPSED:FROM:" in first:
                first = first.rstrip(":")

            if first in dict:
                dict[ first ] += 1
                count += 1
        for e in L:
            getFirst(e, dict)

'''
比較対象となる元のプロジェクトのデータを URLよりプロジェクト番号をして取得
  JSON形式のファイルを読み込み、 "CHILDREN"の持つリストに絞る
その中のブロックが使用されている␣"SCRIPTS"リストにさらに絞る
'''

url1 = 'HTTP://PROJECTS.SCRATCH.MIT.EDU/INTERNALAPI/PROJECT/'+'10000036'+'/GET/'
r1 = urllib2.urlopen(url1)
root1 = json.loads(r1.read())
y1 = root1["CHILDREN"]
a = "SCRIPTS"
s1 = []

'''
全体のブロック数を KEYの指定をしブロック数を数えるFOR文によりSCRATCH_BLOCK . PYの値を更新
'''

for i in range(len(y1)):
    if not a in y1[i].keys():
        continue
    s1 = s1 + y1[i][u'SCRIPTS']

'''
```

```
スプライト数が”objName”で指定されているためその個数を調べるfor文
'''

splite_count1 = 0
for i in range(len(y1)):
    if 'objName' in y1[i]:
        splite_count1 += 1


'''
元のプログラムの個数を求めていく
getFirstのメソッドを動かす
'''

count = 0
x = scratch_block.block
getFirst(s1,x)

'''
    getFirstで集計された個数をブロックの名前順にソートをし、
ブロック名を覗くことでベクターを取得する
'''
xx = x.keys()
xx = sorted(xx)
xxx = []
for ww in xx:
    xxx.append(x[ww])


'''
 for文でcsvファイルのURLを順番に取得し、計算を行う
'''

for row in dataReader:

    row = "".join(row)
    row = row.replace('https://scratch.mit.edu/projects/', '')
    print row


    '''
␣␣␣␣比較をする２つ目のプロジェクトを読み込む
␣␣␣␣以後元のプロジェクトに行った同様の処理を行う
␣␣␣␣'''

    url2 = 'http://projects.scratch.mit.edu/internalapi/project/'+row+'get/'
    r2 = urllib2.urlopen(url2)
    root2 = json.loads(r2.read())
    y2 = root2["children"]
    s2 = []

    for i in range(len(y2)):
        if not a in y2[i].keys():
            continue
        s2 = s2 + y2[i][u'scripts']

    count = 0
    y = scratch_block.block

    '''
␣␣␣␣１つ目のプロジェクトの個数がscratch_block.pyに残っているため初期化をする
␣␣␣␣'''

    for www in y:
        y[www] = 0

    getFirst(s2,y)

    splite_count2 = 0
    for i in range(len(y2)):
        if 'objName' in y2[i]:
            splite_count2 += 1

    yy = y.keys()
    yy = sorted(yy)
    yyy = []
    for ww in yy:
        yyy.append(y[ww])

    '''
␣␣␣␣１つ目、２つ目のプログラムよりcos類似度を計算するSimCalculatorの作動
␣␣␣␣'''

    if __name__ == '__main__':
```

```
167             sc = SimCalculator()
168             cos = sc.sim_cos(xxx,yyy)
169             print str(cos)
170
171         '''
172     ␣␣␣␣分析結果を保存するCSVファイルの作成
173     ␣␣␣␣リストヘッダーの指定
174     ␣␣␣␣'''
175
176         f = open("RESULT_PONG.CSV", "W")
177         writecsv = csv.writer(f)
178         header = ['BLOCK', 'SPLITE', 'COS']
179          csvlist = []
180
181         '''
182     ␣␣␣␣作成した結果をリスト化し、CSVファイルに書き込む
183     ␣␣␣␣'''
184
185         body = [count, splite_count2, cos]
186          csvlist.append(body)
187
188
189
190     writecsv.writerow(header)
191     writecsv.writerows(csvlist)
192
193     f.close()
```

ソースコード 4.6　scratch_block.py

```
# −*− coding: utf −8 −*−
#動き
block = {
    'forward:':0,
    'turnRight':0,
    'turnLeft':0,
    'heading:':0,
    'pointTowards:':0,
    'gotoX:y:':0,
    'gotoSpriteOrMouse':0,
    'glideSecs:toX:y:elapsed:from:':0,
    'changeXposBy:':0,
    'xpos:':0,
    'changeYposBy:':0,
    'ypos:':0,
    'bounceOffEdge':0,
    'setRotationStyle':0,
    'xpos':0,
    'ypos':0,
    'heading':0,
#見た目
    'say:duration:elapsed:from:':0,
    'say:':0,
    'think:duration:elapsed:from:':0,
    'think:':0,
    'show':0,
    'hide':0,
    'lookLike:':0,
    'nextCostume':0,
    'startScene':0,
    'changeGraphicEffect:by:':0,
    'setGraphicEffect:to:':0,
    'filterReset':0,
    'changeSizeBy:':0,
    'setSizeTo:':0,
    'comeToFront':0,
    'goBackByLayers:':0,
    #' costumeIndex ':0,
    #' sceneName ':0,
    #' scale ':0
#音
    'playSound':0,
    'doPlaySoundAndWaid':0,
    'stopAllSounds':0,
    'playDrum':0,
    'rest:elapsed:from:':0,
    'noteOn:duration;elapsed;from;':0,
    'instrument':0,
    'changeVolumeBy:':0,
    'setVolumeTo:':0,
    #' volume ':0,
    #' changeTempo :':0,
    'setTempoTo:':0,
    #' tempo ':0
#ペン
    'clearPenTrails':0,
    'stampCostume':0,
    'putPenDown':0,
    'putPenUp':0,
    'penColor:':0,
    'changePenHueBy:':0,
    'setPenHueTo:':0,
    'changePenShadeBy:':0,
    'setPenShadeTo:':0,
    'changePenSizeBy:':0,
    'penSize:':0,
#データ
    'readVariable':0,
    'setVar:to:':0,
    'changeVar:by:':0,
    'showVariable:':0,
    'hideVariable:':0,
    'contentsOfList:':0,
    'append:toList':0,
    'deleteLine:ofList:':0,
    'insert:at:ofList:':0,
    'setLine:ofList:to:':0,
    'getLine:ofList':0,
    'lineCountOfList:':0,
    'list:contains:':0,
    'showList:':0,
    'hideList:':0,
```

```
#イベント
    'whenGreenFlag':0,
    'whenKeyPressed':0,
    'whenClicked':0,
    'whenSceneStarts':0,
    'whenSensorGreaterThan':0,
    'whenIReceive':0,
    'broadcast':0,
    'doBroadcastAndWait':0,
#制御
    'wait:elapsed:from':0,
    'doRepeat':0,
    'doForever':0,
    'doIf':0,
    'doIfElse':0,
    'doWaitUntil':0,
    'doUntil':0,
    'stopScripts':0,
    'whenCloned':0,
    'createCloneOf':0,
    'deleteClone':0,
#調べる
    'touching':0,
    'touchingColor':0,
    'color:sees:':0,
    'distanceTo:':0,
    'doAsk':0,
    'answer':0,
    'keyPressed':0,
    'mousePressed':0,
    'mouseX':0,
    'mouseY':0,
    'soundLevel':0,
    'senseVideoMotion':0,
    'setVideoState':0,
    'setVideoTransparency':0,
    'timer':0,
    'timerReset':0,
    'getAttribute:of':0,
    'timeAndDate':0,
    'timestamp':0,
    'getUserName':0,
#演算
    '+':0,
    '-':0,
    '*':0,
    '/':0,
    'randomFrom:to:':0,
    '<':0,
    '=':0,
    '>':0,
    '&':0,
    '|':0,
    'not':0,
    'concatenate:with':0,
    'letter:of':0,
    'stringLength':0,
    '%':0,
    'rounded':0,
    'computeFunction:of':0,
#その他
    'procDef':0
}

block2 = sorted(block)
# print   block2
```

cos類似度とは

cos類似度とはベクトル空間モデルにおいて、文書同士を比較する際に用いられる類似度計算方法である。

> コサイン類似度は、そのままベクトル同士の成す角度の近さを表現するため、三角関数の普通のコサインの通り、1に近ければ類似しており、0に近ければ似ていないことになる。([11])

【数式】

$$\cos(A,B) = \frac{\vec{A} * \vec{B}}{|\vec{A}||\vec{B}|} = \frac{\vec{A}}{|\vec{B}|} * \frac{\vec{A}}{|\vec{B}|} = \frac{\sqrt{\sum_{i=1}^{|V|}}}{\sqrt{\sum_{i=1}^{|V|} A_i^2 * \sum_{i=1}^{|V|} B_i^2}} \tag{4.1}$$

【例】（[12]）

文章A：「赤と緑、緑と青」

文章B：「赤と黄、紫と赤」

２つの文章のベクターは赤、緑、青、黄、紫のうちそれぞれ何個使われているかを表す。

ベクターA：(赤、緑、黄、青、紫) = (1, 2, 0, 1, 0)

ベクターB：(赤、緑、黄、青、紫) = (2, 0, 1, 0 ,1)

２つのベクターを数式に当てはめる。

$$A * B = 1 * 2 + 2 * 0 + 0 * 1 + 1 * 0 + 0 * 1 = 2 \tag{4.2}$$

$$|A| = \sqrt{(1 * 1 + 2 * 2 + 0 * 0 + 1 * 1 + 0 * 0)} = \sqrt{6} \tag{4.3}$$

$$|B| = \sqrt{(2 * 2 + 0 * 0 + 1 * 1 + 0 * 0 + 1 * 1)} = \sqrt{6} \tag{4.4}$$

$$\cos\theta = A * B \div |A||B| = \frac{3}{\sqrt{6}\sqrt{6}} = \frac{1}{2} = 0.5 \tag{4.5}$$

本研究では例”色”に当たる部分を、”ブロックの種類”にして計算を行った。Scratchでは数多くのブロックが用意されており、どのブロックを使用するかによって、まったく異なるプログラムを作成することができる。それぞれのプログラムのブロック数を集計した後、ソートしベクターに直し計算式に適用する。

SimCalculatorの解説

cos類似度の計算は次のプログラムで行う。（サンプルプログラム：[13]）

ソースコード4.4では２つのベクトルの内積を計算するsim_cosを作る。

sim_cosではソースコード4.5のベクターリストxxxとyyyを引数とし内積の計算をする。v1(xxx)のn番目のブロックと同じものがv2(yyy)にも存在をするとき、変数numeratorに２つを掛け合わせたものを加える。次にそれぞれのベクターの値を二乗していったものの合計の平方根を掛け合わせたものが分母となるため変数denominatorと置く。最後にnumeratorをdenominatorで割ったものを返す。

scratchAnalysisの解説

プログラム4.5では始めに使用するモジュールの読み込みをする。jsonデータを使用するためのモジュール（L10）、主にHTTPでURLを開くための関数およびクラスのurllib2モジュール（L11, [14]）、csv形式（データをカンマで区切ったもの）のデータを扱うことができるcsvモジュール（L12）を読み込む。またscratch_block（ソースコード4.6）を同様に読み込む。このデータファイルはScratchに登場するブロックを辞書型にしたデータファイルscratch_block.py(4.6)であり、キーがブロックの名前、その値は初期値0を指定したものである。最後にcos類似度の計算を行うSimCalculator（ソースコード4.4）を読み込む。
次に比較するプログラムURLのリスト化したものを読み込む。L31ではバイナリーモードで開いたファイルオブジェクトを変数dataReaderと置く。ファイルを開く際には組み込み関数open()を使用する。([16]) open()とは開いたファイルをファイルオブジェクトを返す。その際にmodeを２つ目の引数で指定でき、本プログラムでは読み込み専用、バイナリーモードを指定している。このファイルはL141以降のfor文内で改め呼び出す。

表4.1　【例】URLリストファイル

https://scratch.mit.edu/projects/100275945/
https://scratch.mit.edu/projects/100492744/
https://scratch.mit.edu/projects/100742859/
https://scratch.mit.edu/projects/100821454/
https://scratch.mit.edu/projects/101706154/
https://scratch.mit.edu/projects/101833379/
https://scratch.mit.edu/projects/10190891/
https://scratch.mit.edu/projects/10191205/
https://scratch.mit.edu/projects/10191673/
https://scratch.mit.edu/projects/102090434/
https://scratch.mit.edu/projects/10214982/
https://scratch.mit.edu/projects/102313969/
https://scratch.mit.edu/projects/102381385/
https://scratch.mit.edu/projects/10245234/
https://scratch.mit.edu/projects/102562822/
https://scratch.mit.edu/projects/102589407/
https://scratch.mit.edu/projects/102718914/
https://scratch.mit.edu/projects/102933941/

L39 メソッドgetFirstでは読み込んだプロジェクトで使用されているブロックごとの個数を数え、ソースコード4.6の値の上書きを行う。ブロックの総個数を変数countと置く。L41のif文では組み込み関数isinstance()を使用しリストであるかを確認する。([15]) 組み込み関数のisinstance()とは２つの引数を指定

し、１つ目が２つ目のインスタンスであるかどうかを返す。また読み込んだデータリストLが長さ0より大きいリストであれば最初の値を変数firstと置く。L43のif文では同様にisinstance()を使用しリストのキーがUnicode文字列であれば処理を進める。Unicode文字列はバイト単位である通常の文字列に対して、文字型に構わず 1 文字を 1 文字として扱う。([17]) if文内では始めに対象データ内の例外処理を行う。ソースコード4.6に記録されているブロック名に一致するように”turnRight:”、”turnLeft:”、”wait:elapsed:from:”の値より末尾のセミコロンを覗く。その際組み込み型のrstripを利用する。rstripはstr.rstripの引数で指定した文字もしくは文字集合を文字列の末尾から覗く。([18]) L54ではdict内([**?**]内の辞書）にfirstと一致するものがあればdictの値を 1 つ増やす。同時にcountを 1 つ増やす。L57ではデータリスト内の値をgetFirstの引数に繰り返し置くことで再帰メソッドにする。

次にurllib2モジュールを使用し、Scratchプロジェクトを呼び出す。変数r1でurlを開き、r1.read()でファイルを読み出したものをjson.loadsを使用しjson型のデータをpythonオブジェクトへの脱直列化する。([19]) root1より”children”を親とするリスト(y1)を取り出す。L77では取り出したリストy1の長さ分のfor文を作成する。事前に”scripts”を変数aとし（L70)、もしaが含まれていれば、使用されているブロックをL71で定義したリストs1に加える。s1はL99でgetFirstの引数に指定する。L98でソースコード4.6のキーであるblockをxと置く。次にcos類似度の計算をするためのベクターを作る。L106ではxをソートし、xxと置く。初期化したリスト（xxx）にはfor文でキーの値を取得し、書き込んでいく。

次にスプライト数を数える。L86ではスプライト数を数える変数splite_count1を初期化する。L77のfor文と同様にリストの長さ分、”objName”を持つリストを探し存在すればsplite_count1の数を 1 増やす。

L116のfor文では始めに読み込んだcsvファイルに存在するデータURL分繰り返すことにする。ここでは最初に読み込んだプロジェクトと順番に比較していく新たなプロジェクトの計算を同様の方法で行う。for文の最後（L166）では作成したベクターリストxxxとyyyを使い、SimCalculatorより呼び出したメソッドsim_cosで計算を行う。L176では結果を書き込むcsvファイルを作成する。L177の変数writecsvにwriter()メソッドを指定し、与えられたリストをcsv形式に変換し、返す。L185のbodyリストにはブロックの総数、スプライトの個数、cos類似度をいれ、csv書き込み用のリストcsvlistに格納。L190とL191でL178で指定したcsvの列のラベル名とL186のcsvlistをL176のファイルに書き込む。

表4.2　【例】結果が書き込まれたファイル

block	splite	cos
57	2	0.622501651
84	2	0.685970521
116	3	0.727309832
38	2	0.788050924
144	3	0.558885703
108	2	0.61049762
28	3	0.939108193
20	2	0.986013297
19	2	1
47	3	0.877613961
69	2	0.654319919
62	2	0.662733496
93	2	0.845819931
61	2	0.674736483
44	3	0.917117834
53	2	0.592156525
35	3	0.916030755
148	10	0.910134671
74	3	0.39669503
93	4	0.758264769
19	2	1
21	2	0.972597525

第III部

結論

第5章

結論

5.1　実験結果

出力されたデータを用いて散布図に表した。縦軸にcos類似度の値、横軸に出力されたブロック数（スプライト数）を元のプログラム（”Pong Starter”）の数値で割った値の対数値でとる。数値では（1,1）が最も類似しているプロジェクトであるため似ているものから青、赤、緑、紫、水色で色を区別してプロットをする。

5.1.1　ブロック数とcos類似度のグラフ

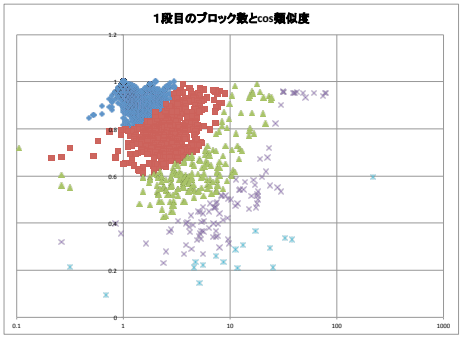


図5.1　１段目のグラフ

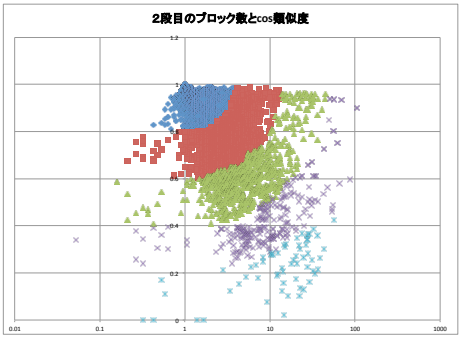


図5.2　２段目のグラフ

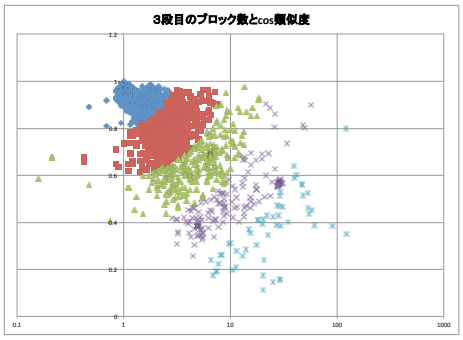


図5.3　３段目のグラフ

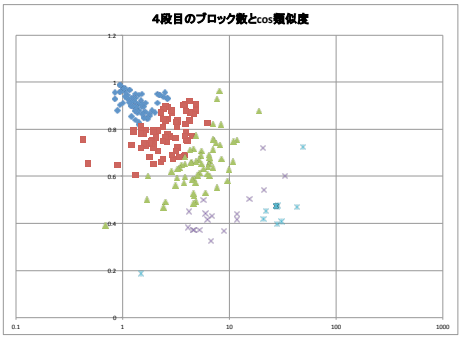


図5.4　４段目のグラフ

1段目のグラフ

- 最も似ているもの

  https://scratch.mit.edu/projects/48772890/

  https://scratch.mit.edu/projects/136740878/

- 最も離れているもの

  https://scratch.mit.edu/projects/19276295/

- 中間

  https://scratch.mit.edu/projects/14377547/

2段目のグラフ

- 最も似ているもの

  https://scratch.mit.edu/projects/10630533/

  https://scratch.mit.edu/projects/20087979/

- 最も離れているもの

  https://scratch.mit.edu/projects/73001742/

- 中間

  https://scratch.mit.edu/projects/31314852/

3段目のグラフ

- 最も似ているもの

  https://scratch.mit.edu/projects/97570310/

- 最も離れているもの

  https://scratch.mit.edu/projects/71289464/

- 中間

  https://scratch.mit.edu/projects/70556576/

4段目のグラフ

- 最も似ているもの

  https://scratch.mit.edu/projects/11534736/

  https://scratch.mit.edu/projects/23516253/

- 最も離れているもの

  https://scratch.mit.edu/projects/44740270/

- 中間

  https://scratch.mit.edu/projects/118943024/

5.1.2　スプライト数とcos類似度のグラフ

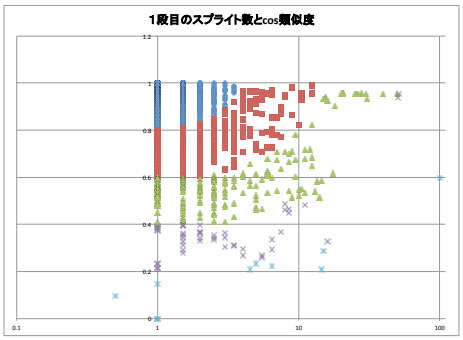


図5.5　1段目のグラフ

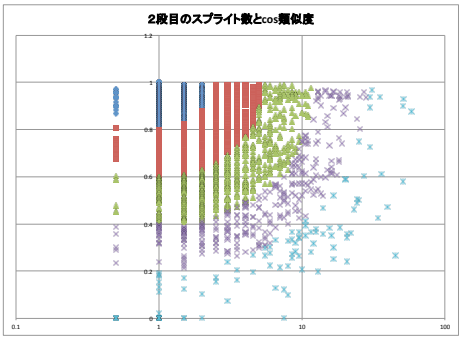


図5.6　2段目のグラフ

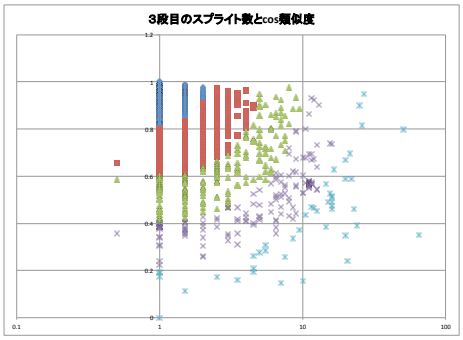


図5.7　3段目のグラフ

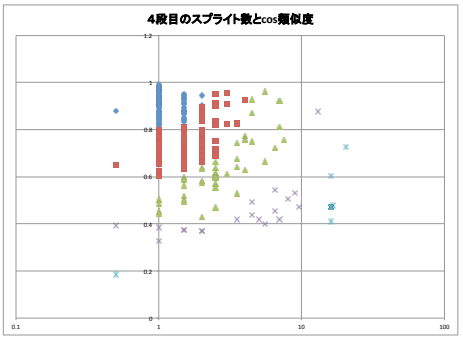


図5.8　4段目のグラフ

1段目のグラフ

- 最も似ているもの

  https://scratch.mit.edu/projects/48772890/

  https://scratch.mit.edu/projects/136740878/

- 最も離れているもの

  https://scratch.mit.edu/projects/19276295/

- 中間

  https://scratch.mit.edu/projects/90709078/

2段目のグラフ

- 最も似ているもの

  https://scratch.mit.edu/projects/10630533/

https://scratch.mit.edu/projects/20087979/

- 最も離れているもの

  https://scratch.mit.edu/projects/73001742/

- 中間

  https://scratch.mit.edu/projects/70556576/

３段目のグラフ

- 最も似ているもの

  https://scratch.mit.edu/projects/97570310/

- 最も離れているもの

  https://scratch.mit.edu/projects/71289464/

- 中間

  https://scratch.mit.edu/projects/33903574/

４段目のグラフ

- 最も似ているもの

  https://scratch.mit.edu/projects/11534736/

  https://scratch.mit.edu/projects/23516253/

- 最も離れているもの

  https://scratch.mit.edu/projects/44740270/

- 中間

  https://scratch.mit.edu/projects/134337312/

第6章

評価

6.1　類似研究者

6.1.1　来住伸子教授

6.2　自己評価

　実験結果が

新奇性

有益性があったかどうかを評価する。

グラフがリミックスツリーと比べてどのようなことがわかり、どのようなメリットがあるのか。

グラフは実際のプロジェクト同士の類似関係と比べて性格であるかどうか。

今後このデータ結果をどのように利用をすれば教育機関で生かされるか。

さらに改善点を示す。

第7章

謝辞

本研究を進めるにあたり、日々丁寧かつ熱心なご指導を来住伸子教授から賜りました。また青山学院大学の吉田葵助教授にも貴重な時間を割いていただきアドバイスをいただきましたここに感謝の意を表します。

参考文献

[1] 教育・学習分野の情報化に係る国内外の動向と先進事例, 総務省 http://www.kknews.co.jp/maruti/news/2014/0804_4a.html

[2] 小学校でプログラミング　アイデアを形にする力をつける (2014年8月4日), 教育課程新聞 http://www.kknews.co.jp/maruti/news/2014/0804_4a.html

[3] Scratch - Imagine, Program, Share, MIT Media Lab https://scratch.mit.edu/

[4] Scratch Offline Editor, MIT Media Lab https://scratch.mit.edu/scratch2download/

[5] Scratchで始めるプログラミング教育（1）:プログラミングを学習する意義、Scratchの基本的な使い方超入門 (2016年3月21日), 薬師寺国安 http://www.atmarkit.co.jp/ait/articles/1603/21/news012.html

[6] Scratch - Imagine, Program, Share, Scratchについて, MIT Media Lab https://scratch.mit.edu/about

[7] Remix tree for Pong Starter https://scratch.mit.edu/projects/10000036/remixtree/

[8] はじめに—学生のためのPython講座 http://python4study.9isnine.com/first

[9] JSONってなにもの？(2008), 竹添直樹 https://thinkit.co.jp/article/70/1

[10] 【Python入門】JSON形式データの扱い方 (2016年12月12日), Morio http://qiita.com/Morio/items/7538a939cc441367070d

[11] コサイン類似度 http://www.cse.kyoto-su.ac.jp/ g0846020/keywords/cosinSimilarity.html

[12] TF-IDF Cos類似度推定法 - Qiita (2015年8月24日), nmbakfm http://qiita.com/nmbakfm/items/6bb91b89571dd68fcea6

[13] Python3.3で実装したナイーブベイズ分類器を利用して文章と文字列中の語の共起頻度から、類似度を計算する (2013年12月26日), katryo http://qiita.com/katryo/items/b6962facf744e93735bb

[14] 20.6. urllib2 ー URLを開くための拡張可能なライブラリ (原文) (2017年1月2日), Python Software Foundation http://docs.python.jp/2/library/urllib2.html

[15] 2. 組み込み関数 ー Python2.7.x ドキュメント (2017年1月2日), Python Software Foundation http://docs.python.jp/2/library/functions.html#isinstance

[16] 2. 組み込み関数 ー Python2.7.x ドキュメント (2017年1月2日),

http://docs.python.jp/2/library/functions.html#open

[17] Unicode 文字列(ユニコード文字列) - 文字列 - PythonWeb, TATSUO IKURA, http://www.pythonweb.jp/tutorial/string/index5.html

[18] 5. 組み込み型 ー Python2.7.x ドキュメント (2017 年 1 月 2 日), http://docs.python.jp/2/library/stdtypes.html?highlight=rstrip#str.rstrip

[19] 18.2. json ー JSON エンコーダおよびデコーダ (原文) ー Python2.7.x ドキュメント (2017年1月2日), http://docs.python.jp/2/library/json.html