指導教員：来住伸子　教授

title

subtitle

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

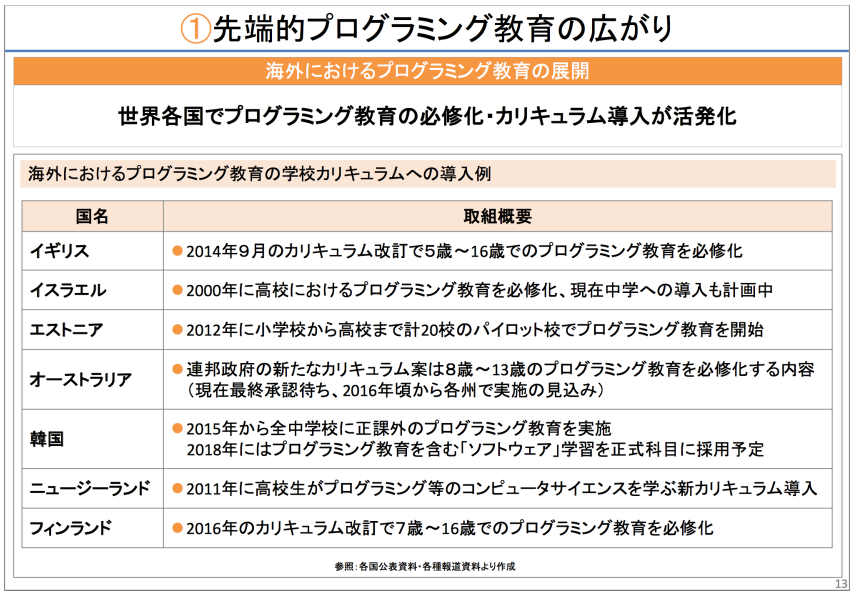
第1章

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（図1.1)。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。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％（図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

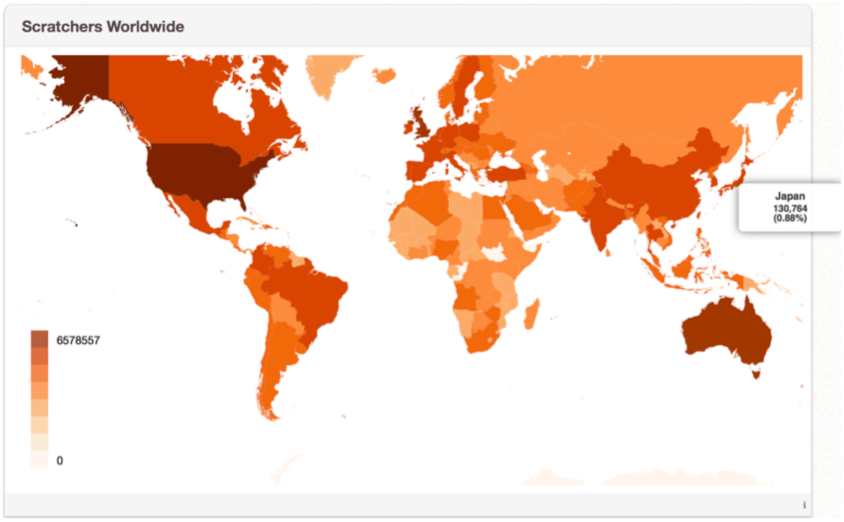


図1.2　全世界のScratchユーザー　日本　130,764人　0.88%

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教育現場で使いやすい新たなデータの創出を目的とする。Scratchの現状機能を調査し、プログラミング教育における課題点を解決に導く分析を目指す。

第II部

本論

第2章

Scratchについて

2.1　Scratchとは

2.2　Scratchの使い方

2.3　Scratchの利点、欠点

2.4　研究におけるScratchの利用意義

第3章

Scratch公式可視化データの現状

3.1　ブロックの使用頻度

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使用頻度を可視化してある。高い割合の項目ほど面積が大きい。項目をクリックすると使用頻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。この図ではScratch全体で使われているブロックの使用頻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使用頻度が分かるということではない。

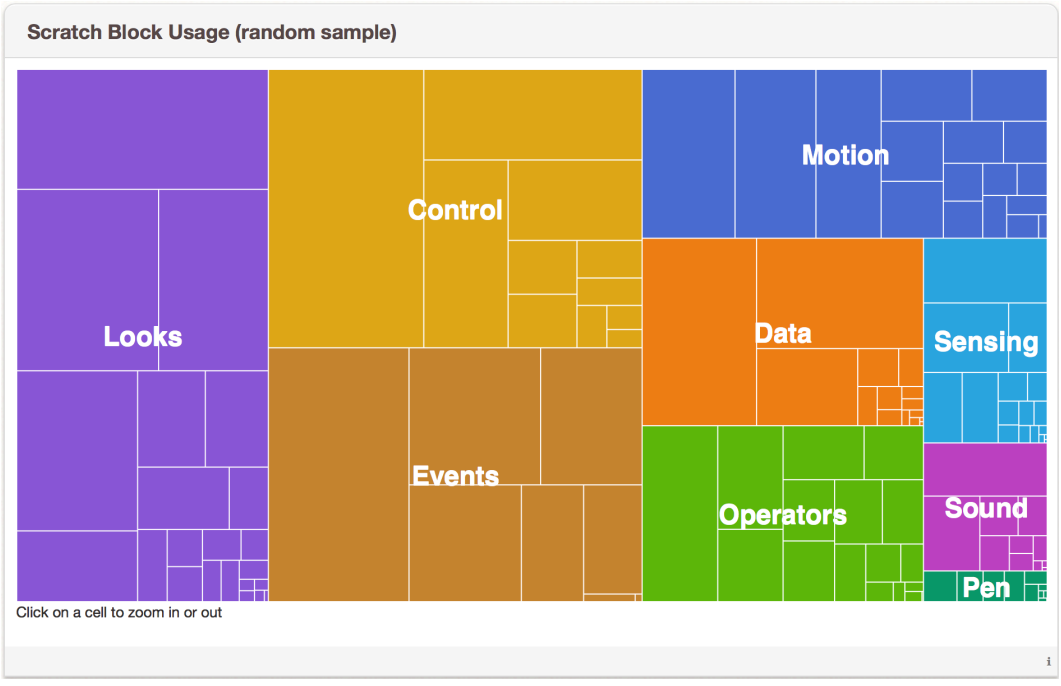


図3.1　全ユーザーブロック使用度

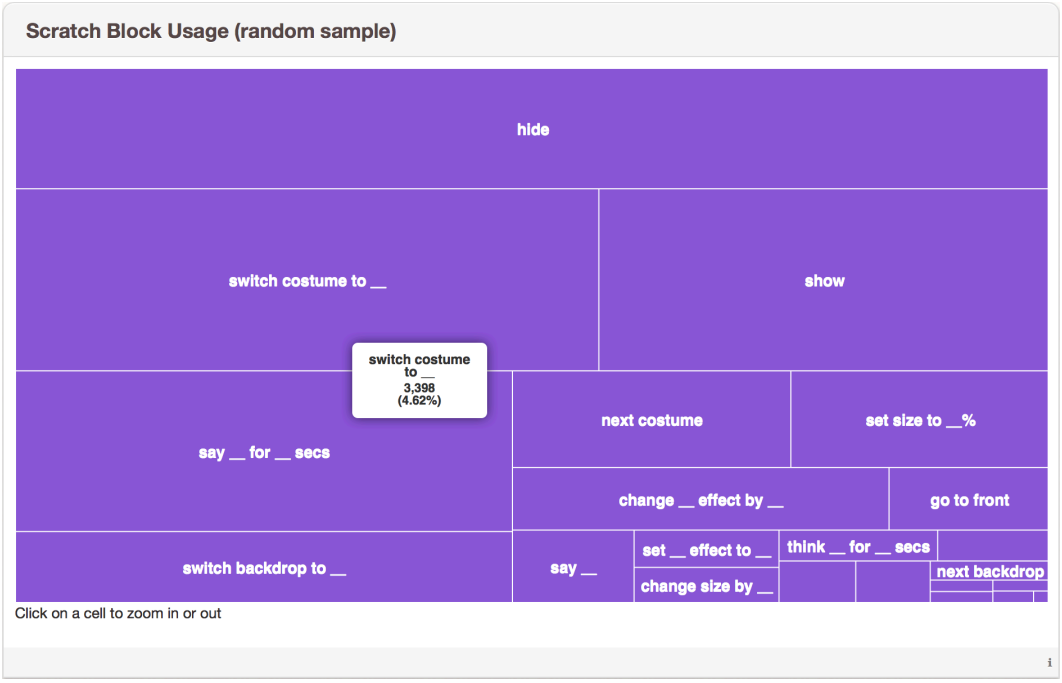


図3.2　全ユーザーブロック費用頻度　Locks（制御ブロック）をクリックした時

3.2　Remix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。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。この図のデメリット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不可能である。

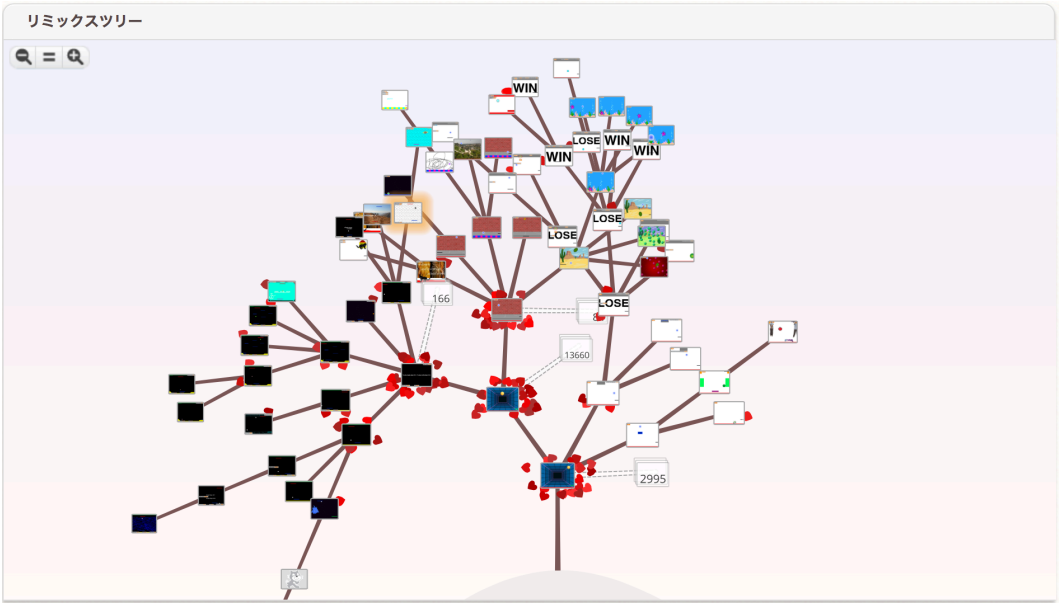


図3.3　Pong Starter のリミックスツリー

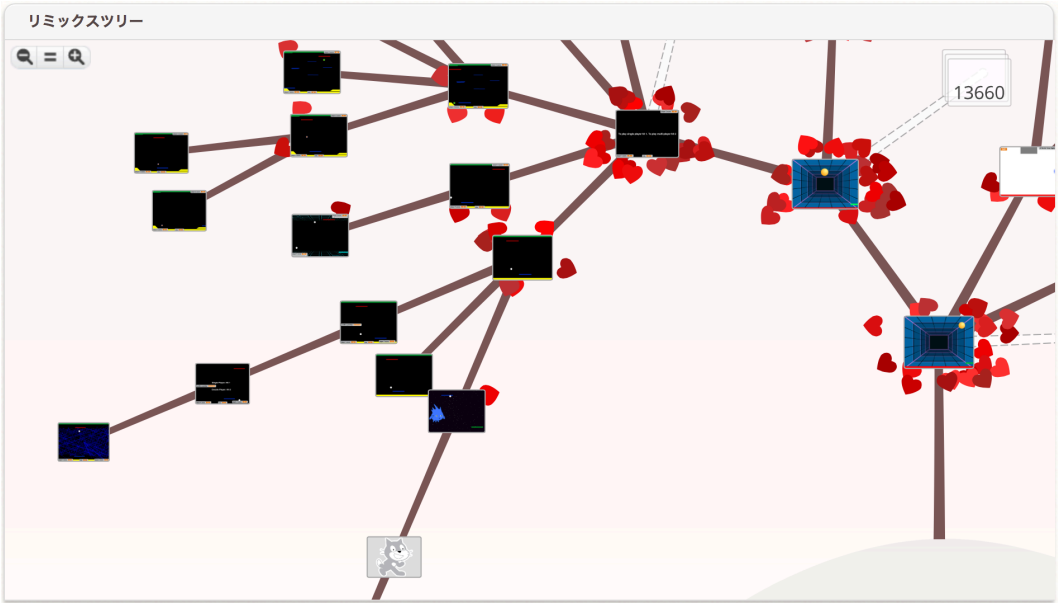


図3.4　Pong Starter のリミックスツリー　左下詳細

第4章

制作内容

4.1 分析内容

4.1.1 cos類似度の計算

ソースコード 4.1 cos類似度の計算

```
# −∗− coding: utf −8 −∗−

import json # jsonモジュールのインポート
import scratch_block #  scratchブロックの辞書のインポート
import urllib2 #  urllib2モジュールのインポート

#  cos類似度計算ファイルのインポート
from SimCalculator import SimCalculator

#
def printFirst(L):
    if isinstance(L, list)and len(L)>0 :
        first = L[0]
        if isinstance( first , unicode):
            for e in L:
                printFirst(e)
            if first in dict:
                dict[ first ] += 1
            else:
                dict[ first ] = 1
        for e in L:
            getFirst(e, dict)

def getFirst(L, dict):
    global count
    if isinstance(L, list)and len(L)>0 :
        first = L[0]
        if isinstance( first ,unicode):
            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)

#元のファイルの読み込み
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 = []

#全体のブロック数の合計
for i in range(len(y1)):
    if not a in y1[i].keys():
        continue
```

```
    s1 = s1 + y1[i][u'SCRIPTS']

#プログラムA
count = 0
x = scratch_block.block
getFirst(s1,x)

xx = x.keys()
xx = sorted(xx)
xxx = []
for ww in xx:
    xxx.append(x[ww])

#2つ目のファイルを読み込む
url2 = 'HTTP://PROJECTS.SCRATCH.MIT.EDU/INTERNALAPI/PROJECT/'+'10128431'+'/GET/'
r2 = urllib2.urlopen(url2)
root2 = json.loads(r2.read())
y2 = root2["CHILDREN"]
s2 = []

#全体のブロック数の合計
for i in range(len(y2)):
    if not a in y2[i].keys():
        continue
    s2 = s2 + y2[i][u'SCRIPTS']

#プログラムB
count = 0
y = scratch_block.block
for www in y:
    y[www] = 0

getFirst(s2,y)

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


#  SimCalculatorを使って計算
if __name__ == '__MAIN__':
    sc = SimCalculator()
    cos = sc.sim_cos(xxx,yyy)
    print str(cos)
```

4.1.2　ブロックとスプライトの合計数

ソースコード 4.2　ブロックの合計

```
# −*− coding: utf −8 −*−

import json # jsonモジュールのインポート
import scratch_block #  scratchブロックの辞書のインポート
import urllib2 #  urllib2モジュールのインポート

def printFirst(L):
    if isinstance(L, list)and len(L)>0 :
        first = L[0]
        if isinstance( first , unicode):
            for e in L:
                printFirst(e)
            if first in dict:
                dict[ first ] += 1
            else:
                dict[ first ] = 1
        for e in L:
            getFirst(e, dict)

def getFirst(L, dict):
    global count
    if isinstance(L, list)and len(L)>0 :
        first = L[0]
        if isinstance( first ,unicode):
            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)

#元のファイルの読み込み
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 = []

#全体のブロック数の合計
for i in range(len(y1)):
    if not a in y1[i].keys():
        continue
    s1 = s1 + y1[i][u'SCRIPTS']

#スプライトの数
count1 = 0
for i in range(len(y1)):
    if 'OBJNAME' in y1[i]:
        count1 += 1

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

print "ブロック数:"+ str(count)
print "スプライト数:"+ str(count1)
```

ソースコード 4.3　scratchAnalysis.py

```
# -*- coding: utf-8 -*-

import json # jsonモジュールのインポート
import scratch_block #  scratchブロックの辞書のインポート
import urllib2 #  urllib2モジュールのインポート
import csv  #  csvモジュールをインポートする

# csvファイルの読み込み
dataReader = csv.reader(open("PONG.CSV","RU"))
f = open("RESULT_PONG1.CSV", "W")
writecsv = csv.writer(f)
header = ['BLOCK', 'SPLITE', 'COS']
csvlist = []

#  cos類似度計算ファイルのインポート
from SimCalculator import SimCalculator

#
def printFirst(L):
    if isinstance(L, list)and len(L)>0 :
        first = L[0]
        if isinstance( first , unicode):
            for e in L:
                printFirst(e)
            if first in dict:
                dict[first] += 1
            else:
                dict[first] = 1
        for e in L:
            getFirst(e, dict)

def getFirst(L, dict):
    global count
    if isinstance(L, list)and len(L)>0 :
        first = L[0]
        if isinstance( first ,unicode):
            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)

#元のファイルの読み込み
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 = []

#全体のブロック数の合計
for i in range(len(y1)):
    if not a in y1[i].keys():
        continue
    s1 = s1 + y1[i][u'scripts']

#スプライトの数
count1 = 0
for i in range(len(y1)):
    if 'objName' in y1[i]:
        count1 += 1
        ###

#プログラムA
count = 0
x = scratch_block.block
getFirst(s1,x)

print "プログラムA"
print "ブロック数␣:"+ str(count)
print "スプライト数␣:"+ str(count1)

xx = x.keys()
xx = sorted(xx)
xxx = []
for ww in xx:
    xxx.append(x[ww])


# csvファイルを読み込み
for row in dataReader:

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


#2つ目のファイルを読み込む
    url2 = 'http://projects.scratch.mit.edu/internalapi/project/'+row+'get/'
    r2 = urllib2.urlopen(url2)
    root2 = json.loads(r2.read())
    y2 = root2["children"]
    s2 = []

    #全体のブロック数の合計
    for i in range(len(y2)):
        if not a in y2[i].keys():
            continue
        s2 = s2 + y2[i][u'scripts']

    #プログラムB
    count = 0
    y = scratch_block.block
    for www in y:
        y[www] = 0

    getFirst(s2,y)

    #スプライトの数
    count2 = 0
    for i in range(len(y2)):
        if 'objName' in y2[i]:
            count2 += 1

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

#  SimCalculatorを使って計算
```

```
129     if __name__ == '__MAIN__':
130         sc = SimCalculator()
131         cos = sc.sim_cos(xxx,yyy)
132         print str(cos)
133 # csvファイルへのリスト
134     body = [count, count2, cos]
135     csvlist.append(body)
136 # csvファイルの書き込み
137 writecsv.writerow(header)
138 writecsv.writerows(csvlist)
139
140 f.close()
```

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 _absolute(self, vector):
9          # ベクトル vの長さつまり絶対値を返す
10         squared_distance = sum([vector[n] ** 2 for n in vector])
11         distance = math.sqrt(squared_distance)
12         return distance
13
14     def sim_cos(self, v1, v2):
15         numerator = 0
16         #       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。
17         i = 0
18         for n in v1:
19             numerator += v1[i]*v2[i]
20             # if n in v2:
21             #  numerator += v1[n] * v2[n]
22             # print i, numerator, v1[i], v2[i]
23             i=i+1
24
25             ss1 = []
26             ss2 = []
27         for x in v1:
28             ss1.append(x*x)
29         for x in v2:
30             ss2.append(x*x)
31
32         denominator = math.sqrt(sum(ss1))*math.sqrt(sum(ss2))
33
34         if denominator == 0:
35             return 0
36         return numerator / denominator
```

4.2　グラフ化

4.2.1　cos類似度とブロック数のグラフ

4.2.2　cos類似度とスプライト数のグラフ

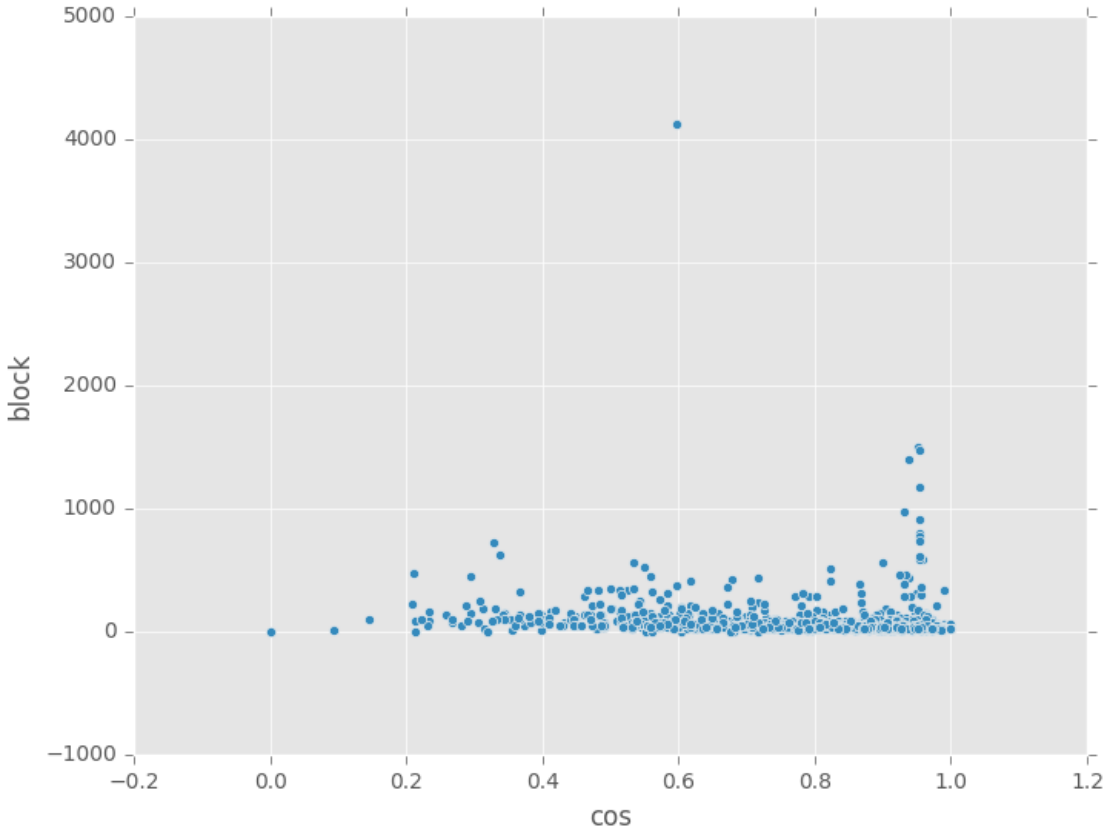


図4.1　cos類似度とブロック数のグラフ

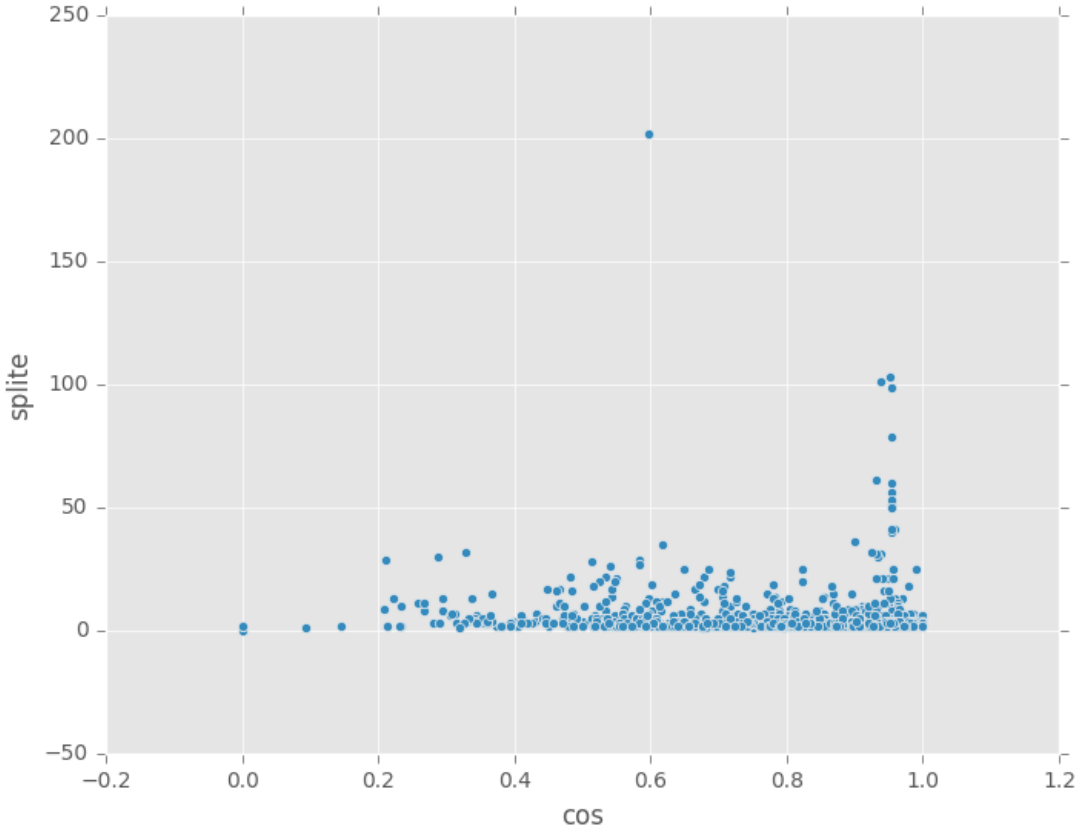


図4.2　cos類似度とスプライト数のグラフ

第III部

結論

第5章

評価

5.1　類似研究者

5.1.1　吉田葵先生

5.1.2　来住伸子先生

第6章

結論

第7章

謝辞

第8章

参考文献

参考文献

[1] D.E. クヌース，改訂新版 TEX ブック，アスキー出版局，東京，1992.

[2] 磯崎秀樹，LATEX 自由自在，サイエンス社，東京，1992.

[3] S. von Bechtolsheim, TEX in Practice, Springer-Verlag, New York, 1993.

[4] 藤田眞作，化学者・生化学者のためのLATEX—パソコンによる論文作成の手引，東京化学同人，東京，1993.

[5] 阿瀬はる美，てくてくTEX，アスキー出版局，東京，1994.

[6] N. Walsh, Making TEX Work, O'Reilly & Associates, Sebastopol, 1994.

[7] D. Salomon, The Advanced TEX book, Springer-Verlag, New York, 1995.

[8] 藤田眞作，LATEX マクロの八衢，アジソン・ウェスレイ・パブリッシャーズ・ジャパン，東京，1995.

[9] 中野賢，日本語 LATEX 2_ε ブック，アスキー出版局，東京，1996.

[10] 藤田眞作，LATEX 2_ε 階梯，アジソン・ウェスレイ・パブリッシャーズ・ジャパン，東京，1996.

[11] 乙部巌己，江口庄英，pLATEX 2_ε for Windows Another Manual，ソフトバンク パブリッシング，東京，1996–1997.

[12] ポール W. エイブラハム，明快 TEX，アジソン・ウェスレイ・パブリッシャーズ・ジャパン，東京，1997.

[13] 江口庄英，Ghostscript Another Manual，ソフトバンク パブリッシング，東京，1997.

[14] マイケル・グーセンス，フランク・ミッテルバッハ，アレキサンダー・サマリン，LATEX コンパニオン，アスキー出版局，東京，1998.

[15] ビクター・エイコー，TEX by Topic—TEX をよく深く知るための39章，アスキー出版局，東京，1999.

[16] レスリー・ランポート，文書処理システム LATEX 2_ε，ピアソンエデュケーション，東京，1999.

[17] 奥村晴彦，［改訂版］LATEX 2_ε 美文書作成入門，技術評論社，東京，2000.

[18] マイケル・グーセンス，セバスチャン・ラッツ，フランク・ミッテルバッハ，LATEX グラフィックスコンパニオン，アスキー出版局，東京，2000.

[19] マイケル・グーセンス，セバスチャン・ラッツ，LATEX Web コンパニオン—TEX とHTML/XML の統合，アスキー出版局，東京，2001.

[20] ページ・エンタープライゼス(株)，LATEX 2_ε マクロ & クラスプログラミング基礎解説，技術評論社，

東京，2002.

[21] 藤田眞作，LaTeX 2$_\varepsilon$ コマンドブック，ソフトバンク パブリッシング，東京，2003.

[22] 吉永徹美，LaTeX 2$_\varepsilon$ マクロ & クラスプログラミング実践解説，技術評論社，東京，2003.