

～西宮市の周囲長を測定しよう～

大阪教育大学附属池田中学校 2年岸岡友陽

1. 研究の動機

ある日、兵庫県の様々な市町村の統計を眺めているとふとあることに気が付きました。それは、市町村の周囲長に関する統計がないということです。都道府県の周囲長に関する統計は多くありましたが、市町村の周囲長に関する統計はありませんでした。そして、「ないのならば自分で測ってしまえ」という発想に至りました。

2. 研究の方法

さて、悩ましいのが周囲長を測る方法です。実際に現地に行って測るのは現実的に不可能であり、GoogleEarthの測定機能を使って測るのもできなくはないのですが途方もない時間がかかってしまいます。そこで僕は学生向けのプログラミング言語であるscratchを用いて、スプライトが線上を動くプログラムと、スプライトが動いた距離を求めるプログラムを作ることの思いつきました。次が今回の活動の流れです。

- ①scratchでスプライトが線上を動くプログラムと、スプライトが動いた距離を求めるプログラムを作る。
- ②GoogleEarthでそれぞれの市町村の境界を調べる。（この時のカメラの高さは全て同じにする。）
- ③それをChatGPTを活用して表にまとめて、後ほど紹介する「2点間の距離」の公式に当てはめて周囲長を求める全ての市町村の周囲長を求めるのは非常に時間がかかるため、今回は自分の住んでいる「西宮市」の周囲長を代表して求めようと思います。

①のプログラム→



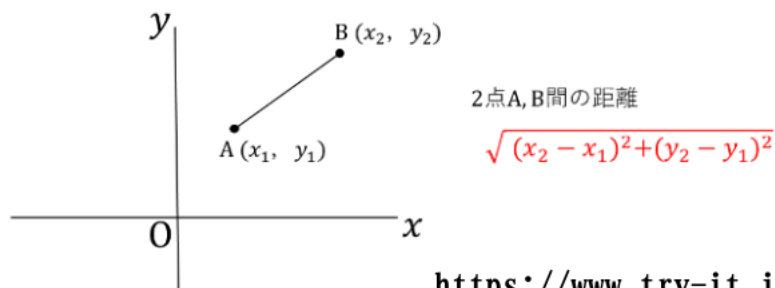
説明：このプログラムでは、スプライトが赤色の部分(GoogleEarthで市町村の境界が赤色で表示される)からはみ出さないようにするプログラムと、スプライトの座標を動きながら記録し続けるプログラムを組み合わせています。スプライトの座標を記録したところで、長さなんてわからないだろうと思う人もいるかもしれませんが、中3数学の「2点間の距離」(三平方の定理)を活用すれば求めることができます。具体的な方法は後のほど述べます。また、そのプログラムでは主に「変数」と「リストを作る」という記録を使って求めました。まず、現在のスプライトのx座標とy座標を示す「x座標」と「y座標」と

いう変数を作ります。そしてそのあとスプライトが赤色の部分(境界部分)をはみ出さないように進んでいくプログラムと結合します。それは赤色の部分に触れている最中は常時角を3度ずつ右に向けさせながら1歩ずつ歩かせ、もし赤色から離れたらもう一度赤色に触れるまでずっとその逆の動きをさせるという仕組みでできています。なぜ3や1などの低い値で設定されているかというと、大きい値だと赤色の部分からはみ出て、もう一度赤色にふれる時の動きが大げさになり、スプライトが無駄な動きをたくさんしてしまうようになるからです。その欠点として、スプライトが境界を一周するのに時間がかかってしまうことがあります。しかし、正確な長さを測るためには仕方のないことです。また、そのプログラムに沿ってスプライトが動いている最中に数秒おきに「リスト」にその時点でのx座標とy座標を記録しています。それは、次のプログラムで行います。



これは、3秒おきに変数でおいているx座標とy座標をリストに記録するプログラムです。

先程述べた、中3数学の「2点間の距離」(三平方の定理)というのは、2つの座標からその間の距離を求めることができるというものであり、その公式は次のとおりです。



https://www.try-it.jp/keyword_articles/26/より引用

これは三平方の定理を利用しています。三平方の定理とは、直角三角形において斜辺の長さをc、ほかの2辺をaとbとした時に、 $a^2 + b^2 = c^2$ が成り立つという定理です。これを関数にあてはめて、(xを横辺・yを縦辺、xとyの間の距離を斜辺に見立てて)斜辺について解いた結果がこの公式です。この公式にプログラムで得た座標の具体値を当てはめていって長さを求めて、最後にそれを合計します。

西宮市→



リストを書き出したもの(txt形式で出力されたものをスクショして貼り合わせたもの)↓

```
-7. 44269556131249
-17. 424444246933746
-12. 444266911396126
-15. 38025883982502
-16. 814308422269065
-14. 205467638949466
-21. 376640509479195
-14. 126694847023789
-26. 185042236835248
-10. 142511242003481
-29. 90291040327159
-8. 569469223041665
-34. 34337800498174
-9. 154822557852622
-29. 340594167816
-7. 953628308498292
-43. 89556990177404
-7. 9283594835769
-48. 7159050065683
-8. 136482820583918
-53. 5277658308651
-6. 567843501480588
-58. 03428731828023
-3. 700540585380564
-62. 161976805539
-3. 3512703198795037
-66. 90005347118594
-3. 243070346101487
-71. 70189184095165
-4. 573959397539681
-76. 59437713354287
-6. 452811225494205
-81. 32132406022083
-7. 57382772988376
-85. 93589724664498
-9. 00160076219698

-67. 52564208446238
31. 020398274617897
-63. 52187368475243
28. 637628104093725
-59. 50197414038374
26. 181311177954797
-58. 38327232610146
20. 02705900554543
-53. 07974639063685
19. 820194716594223
-48. 56690067861623
18. 04223553952227
-43. 15362966123874
19. 9115217515811
-38. 00215584906517
22. 96949412157381
-31. 15663683404954
23. 61692923331102
-28. 6041924701662
27. 602078976813864
-25. 09616883287154
32. 90853976061712
-20. 41457000436107
33. 5928949150357
-16. 74450336272007
22. 52034955293365
-15. 5976297957449
15. 247541679082494
-15. 93316458696338
8. 49077004611923
-17. 903647895674001
2. 120002791331837
-17. 16540639078154
-2. 651518132098452
-16. 6312174212807

-67. 60996530545897
88. 7032024149589
-68. 00257024335359
83. 31123090683766
-77. 1492778822107
77. 77791663615277
-83. 02861638572084
74. 71342612304022
-87. 71864252898347
71. 96504636151899
-93. 61752437237602
70. 77625468440846
-99. 08299260004324
70. 00561835107091
-102. 75223454110862
67. 13480587934222
-107. 63466828854656
66. 07602069017742
-111. 51780661111337
63. 50361188414735
-109. 28048161540289
54. 87514055300798
-104. 8800898038551
47. 949386714443186
-95. 65192248625799
43. 70859966368123
-91. 29752756387367
40. 482256357623676
-37. 23996650505045
37. 34942944300525
-84. 7681927132006
33. 399841588746916
-77. 09804217934534
33. 89113951648967
-72. 09433800346554
32. 71044219069126

-113. 01765438624486
155. 4472936729621
-108. 28929109943981
158. 41673078690414
-103. 54375083913384
158. 83803000664443
-98. 78273822558826
158. 1718400282527
-94. 78319017360447
152. 87661047547397
-96. 68736697747221
141. 2905614204339
-93. 5614591616882
137. 9082383694253
-92. 1159896371829
133. 0607824398512
-91. 08141795983204
128. 16038365787057
-88. 53145012585733
124. 19953588001287
-85. 1053091142286
120. 88304259506215
-83. 41939588745018
116. 5371428587329
-83. 86061621761718
111. 11946650742406
-80. 3194264725776
107. 6284941088926
-77. 31047283085898
104. 48238130997623
-74. 92726100678848
100. 60187998013275
-73. 30504749313417
96. 15123059373143
-71. 03170623700892
92. 08491441691694

-203. 94762403024063
73. 52107626488733
-207. 21035901887257
76. 97004956372345
-120. 9521170303034
79. 6534959169784
-214. 01249586141947
83. 29156846644463
-214. 28034479092275
89. 35621534143684
-216. 16326545299012
93. 75859508002408
-217. 89336048250425
96. 20079972157049
-217. 3696762676171
103. 66122006307191
-218. 0747817732888
108. 4474825994235
-216. 28213178506414
113. 7762636882866
-215. 9951927804326
118. 44219184422828
-214. 43358079019535
123. 1512662040691
-213. 50188452142013
127. 24295840695432
-213. 45079120853424
131. 5875714912086
-212. 3630865616963
135. 8368758810023
-207. 01713815897907
139. 21514836052344
-200. 71446922783898
141. 09242452644048
-194. 79002077502636

-95. 192335939886
-12. 444384620621431
-98. 95832017884216
-14. 429478790978607
-103. 19976085165916
-15. 621574278545378
-107. 75668966713505
-17. 951034980985092
-111. 90746728846746
-14. 77052641948815
-117. 42442434879571
-24. 41241433712919
-118. 3112422527922
-27. 07119860230149
-122. 26435186322551
-30. 595924543105042
-125. 4089719974751
-35. 712346069179226
-126. 9564282029536
-44. 45460564612806
-119. 07838299917661
-48. 39260052977932
-110. 07148795018539
-43. 143848105045144
-96. 43789615846218
-44. 8191346746067
-91. 95931366746066
-57. 49905403361667
-102. 9261803261684
-61. 161856938494836
-98. 90179601459705
-62. 97405506055606
-92. 56903440064978
-65. 3526497498143
-88. 16350157722309
-73. 04497350084202
```

このままでは非常に見にくいいためchatGPTを使って表にまとめました。次が自分とChatGPTとのやり取りです。(一部省略しています。)

自分：-203.94762403024063 73.52107626488733…(以下省略)数字を全て一の位で四捨五入して※正確には1の位を

GPT：-204 74 -207 77…(以下省略)

自分：-204 74 -207 77…(以下省略)これらの数字の中で、これらの数字の中で、奇数列の数を「x座標」偶数列の数を「y座標」という名前の箱に入れて表にしてください。

GPT：x座標（奇数列の数）：-204，-207，（以下省略）y座標（偶数列の数）：74，77，…（以下省略）

自分：表にして

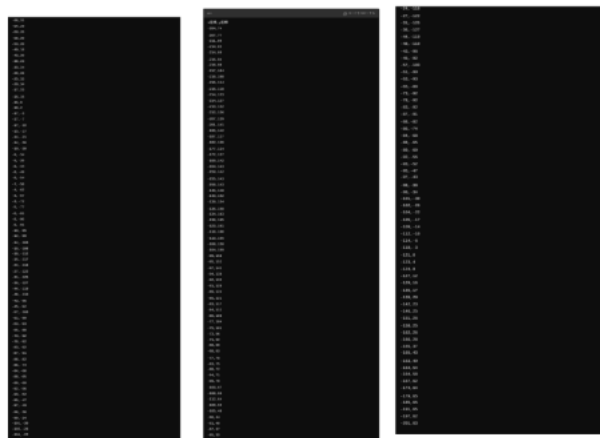
GPT：



自分：この表をダウンロードするにはどうすればいいですか

GPT：お手数をおかけして申し訳ありません。次の手順で自分のコンピュータでCSVファイルを作成することができます：

1. 以下の内容をコピーしてメモ帳などのテキストエディタに貼り付けます。



2. ファイルを保存する際に、ファイル名を `coordinates.csv` として保存してください。

3. これで、保存したCSVファイルを任意の表計算ソフト（Excelなど）で開くことができます。

この方法を用いてGoogleスプレッドシートで開いたのがこちらです

これらの値を「2点間の距離」の公式に当てはめます。

$$\text{距離} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

全て計算するのは大変なので、Chatgptを利用します。以下、自分とCHatGPTとのやり取りです。

自分： x座標y座標 -204 74 -207 77 -211…(以下省略)のx座標とy座標の表を上から2つずつ区切って、それぞれの2点の座標の距離を求めなさい。ただし、 $\text{sqrt}((x_2-x_1)^2 + (y_2-y_1)^2)$ の公式を必ず用いること。

GPT： 各2点間の距離を求めるために、次の手順を実行します：

1. 各ペアの2点の座標を取り出す。
2. 2点間の距離を計算する。

距離の計算には、次の公式を使用します： $\text{距離} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

すべてのペアについて計算を行いましょう。

```
import math
```

```
coordinates = [
```

```
    (-204, 74), (-207, 77), (-211, 80), (-214, 83), (-214, 89), (-216, 94), (-218, 98), (-217, 104), (-218, 108), (-216, 114), (-216, 118), (-214, 123), (-214, 127), (-213, 132), (-212, 136), (-207, 139), (-201, 141), (-195, 142), (-187, 137), (-182, 136), (-177, 134), (-172, 137), (-169, 142), (-164, 143),
```

```
    (-159, 142), (-155, 143), (-150, 143), (-146, 148), (-143, 152), (-139, 154), (-136, 158), (-134, 163),
```

```
    (-130, 165), (-123, 161), (-118, 160), (-113, 155), (-108, 158), (-104, 159), (-99, 158), (-95, 153),
```

```
    (-97, 141), (-94, 138), (-92, 133), (-91, 128), (-89, 124), (-85, 121), (-83, 117), (-84, 111),
```

```
    (-80, 108), (-77, 104), (-75, 101), (-73, 96), (-71, 92), (-68, 89), (-68, 83), (-77, 78),
```

```
    (-83, 75), (-88, 72), (-94, 71), (-99, 70), (-103, 67), (-108, 66), (-112, 64), (-109, 55),
```

```
    (-105, 48), (-96, 44), (-91, 40), (-87, 37), (-85, 33), (-77, 34), (-72, 33), (-68, 31),
```

```
(-64, 29), (-60, 26), (-58, 20), (-53, 20), (-49, 18), (-43, 20), (-38, 23), (-33, 24),
(-29, 28), (-25, 33), (-20, 34), (-17, 23), (-16, 15), (-16, 8), (-18, 2), (-17, -3),
(-17, -7), (-17, -12), (-15, -17), (-14, -21), (-14, -26), (-10, -30), (-9, -34), (-9, -39),
(-8, -44), (-8, -49), (-8, -54), (-7, -58), (-4, -62), (-3, -67), (-3, -72), (-5, -77),
(-6, -81), (-8, -86), (-9, -91), (-10, -95), (-12, -99), (-14, -103), (-16, -108), (-15, -117),
(-24, -118), (-27, -122), (-31, -125), (-36, -127), (-44, -119), (-48, -110), (-43, -96), (-45, -92),
(-57, -103), (-61, -99), (-63, -93), (-65, -88), (-73, -92), (-78, -92), (-83, -92), (-87, -91),
(-88, -82), (-86, -74), (-84, -68), (-88, -65), (-89, -60), (-92, -56), (-93, -52), (-95, -47),
(-97, -43), (-98, -38), (-99, -34), (-101, -), (-102, -26), (-104, -22), (-105, -17), (-109, -14),
(-112, -10), (-114, -6), (-118, -3), (-121, 0), (-123, 4), (-124, 8), (-127, 12), (-129, 16),
(-135, 17), (-139, 20), (-142, 23), (-146, 25), (-151, 26), (-158, 25), (-162, 26), (-166, 28),
(-165, 37), (-165, 43), (-163, 49), (-164, 54), (-164, 58), (-167, 62), (-174, 64), (-179, 65),
(-185, 65), (-191, 65), (-197, 62), (-201, 63)
```

```
]distances = []
for i in range(0, len(coordinates), 2):
    x1, y1 = coordinates[i]
    x2, y2 = coordinates[i + 1]
    distance = math.sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2) ← 距離 =  $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$ 
    distances.append(distance)
distances
```

このPythonコードを実行して、2点間の距離を求めます。計算結果を提供します。

各ペアの2点間の距離は次の通りです：

```
(-204, 74) と (-207, 77) の距離： 4.24 (-211, 80) と (-214, 83) の距離： 4.24 (-214, 89) と (-216, 94) の距離： 5.39
(-218, 98) と (-217, 104) の距離： 6.08 (-218, 108) と (-216, 114) の距離： 6.32(-216, 118) と (-214, 123) の距離： 5.39
(-214, 127) と (-213, 132) の距離： 5.10(-212, 136) と (-207, 139) の距離： 5.83(-201, 141) と (-195, 142) の距離： 6.08
(-187, 137) と (-182, 136) の距離： 5.10(-177, 134) と (-172, 137) の距離： 5.83(-169, 142) と (-164, 143) の距離： 5.10
(-159, 142) と (-155, 143) の距離： 4.12 (-150, 143) と (-146, 148) の距離： 6.40(-143, 152) と (-139, 154) の距離： 4.47
(-136, 158) と (-134, 163) の距離： 5.39(-130, 165) と (-123, 161) の距離： 8.06(-118, 160) と (-113, 155) の距離： 7.07
(-108, 158) と (-104, 159) の距離： 4.12(-99, 158) と (-95, 153) の距離： 6.40(-97, 141) と (-94, 138) の距離： 4.24
(-92, 133) と (-91, 128) の距離： 5.10(-89, 124) と (-85, 121) の距離： 5.00(-83, 117) と (-84, 111) の距離： 6.08
```

(-80, 108) と (-77, 104) の距離: 5.00(-75, 101) と (-73, 96) の距離: 5.39(-71, 92) と (-68, 89) の距離: 4.24

(-68, 83) と (-77, 78) の距離: 10.30(-83, 75) と (-88, 72) の距離: 5.83(-94, 71) と (-99, 70) の距離: 5.10

(-103, 67) と (-108, 66) の距離: 5.10(-112, 64) と (-109, 55) の距離: 9.49(-105, 48) と (-96, 44) の距離: 9.85

(-91, 40) と (-87, 37) の距離: 5.00(-85, 33) と (-77, 34) の距離: 8.06(-72, 33) と (-68, 31) の距離: 4.47

(-64, 29) と (-60, 26) の距離: 5.00(-58, 20) と (-53, 20) の距離: 5.00(-49, 18) と (-43, 20) の距離: 6.32

(-38, 23) と (-33, 24) の距離: 5.10(-29, 28) と (-25, 33) の距離: 6.40(-20, 34) と (-17, 23) の距離: 11.40

(-16, 15) と (-16, 8) の距離: 7.00(-18, 2) と (-17, -3) の距離: 5.10(-17, -7) と (-17, -12) の距離: 5.00

(-15, -17) と (-14, -21) の距離: 4.12(-14, -26) と (-10, -30) の距離: 5.66(-9, -34) と (-9, -39) の距離: 5.00

(-8, -44) と (-8, -49) の距離: 5.00(-8, -54) と (-7, -58) の距離: 4.12(-4, -62) と (-3, -67) の距離: 5.10

(-3, -72) と (-5, -77) の距離: 5.39(-6, -81) と (-8, -86) の距離: 5.39(-9, -91) と (-10, -95) の距離: 4.12

(-12, -99) と (-14, -103) の距離: 4.47(-16, -108) と (-15, -117) の距離: 9.06(-24, -118) と (-27, -122) の距離: 5.00

(-31, -125) と (-36, -127) の距離: 5.39(-44, -119) と (-48, -110) の距離: 9.85(-43, -96) と (-45, -92) の距離: 4.47

(-57, -103) と (-61, -99) の距離: 5.66(-63, -93) と (-65, -88) の距離: 5.39(-73, -92) と (-78, -92) の距離: 5.00

(-83, -92) と (-87, -91) の距離: 4.12(-88, -82) と (-86, -74) の距離: 8.25(-84, -68) と (-88, -65) の距離: 5.00

(-89, -60) と (-92, -56) の距離: 5.00(-93, -52) と (-95, -47) の距離: 5.39(-97, -43) と (-98, -38) の距離: 5.10

(-99, -34) と (-101, -30) の距離: 4.47(-102, -26) と (-104, -22) の距離: 4.47(-105, -17) と (-109, -14) の距離: 5.00

(-112, -10) と (-114, -6) の距離: 4.47(-118, -3) と (-121, 0) の距離: 4.24(-123, 4) と (-124.8) の距離: 4.12

(-127, 12) と (-129, 16) の距離: 4.47(-135, 17) と (-139, 20) の距離: 5.00(-142, 23) と (-146, 25) の距離: 4.47

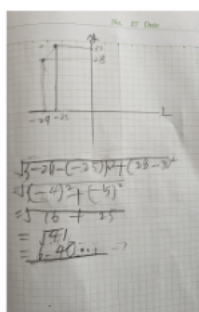
(-151, 26) と (-158, 25) の距離: 7.07(-162, 26) と (-166, 28) の距離: 4.47(-165, 37) と (-165, 43) の距離: 6.00

(-163, 49) と (-164, 54) の距離: 5.10(-164, 58) と (-167, 62) の距離: 5.00(-174, 64) と (-179, 65) の距離: 5.10

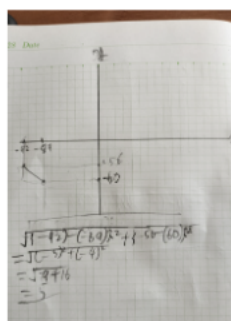
(-185, 65) と (-191, 65) の距離: 6.00(-197, 62) と (-201, 63) の距離: 4.12

念の為この結果が本当に正しいかどうか検証します。3つランダムに選んで自分で計算してみます。

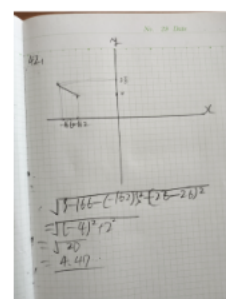
41, (-29, 28) と (-25, 33)



67, (-89, -60) と (-92, -56)



80, (-80, 108) と (-77, 104)



GPTの答えと一致しました。正しいですね

そして座標の値と実際の距離の関係性を求めます。(-204, 74)～(-214, 83)の距離は座標上では $\sqrt{(-214 - (-204))^2 + (83 - 74)^2}$

$$= \sqrt{(-10)^2 + 9^2}$$

$$= \sqrt{100 + 81}$$

$$= \sqrt{181}$$

$$= 13.45\cdots$$

ですが、それが現実においてどのくらいの長さに相当するかをGoogleEarthで確かめます。

図1はスプライトがスクラッチで使用した西宮の境界線の(-204, 74)と(-214, 83)にそれぞれいる時をスクリーンショットしたものと、実際のGoogleEarthの西宮の衛星写真をスクリーンショットしたものを重ねたものです。少々見づらいですが、青線の一部が(-204, 74)～(-214, 83)です。そして図2はその距離をGoogleEarthの測定機能で測ったものです。1052.75mとなりました。1の位を四捨五入すると1053mとなる。これをもとに周囲長を求めます。

図1

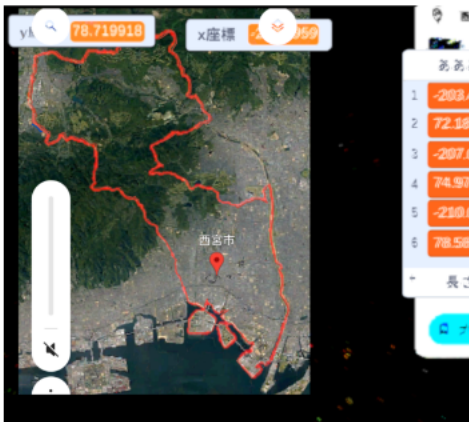
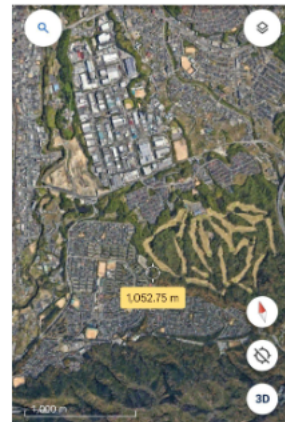


図2



先ほどの作業で座標上における13.44の距離が1053mに当たると分かりました。そして次に、先ほどGPTに求めてもらった距離を合計します。

4.24+4.24+5.39+6.08+6.32+5.39+5.10+5.83+6.08+5.10+5.83+5.10+4.12+6.40+4.47+5.39+8.06+7.07+4.12+6.40+4.24+5.10+5.00+6.08+5.00+5.39+4.24+10.30+5.83+5.10+5.10+9.49+9.85+5.00+8.06+4.47+5.00+5.00+6.32+5.10+6.40+11.40+7.00+5.10+5.00+4.12+5.66+5.00+5.00+4.12+5.10+5.39+5.39+4.12+4.47+9.06+5.00+5.39+9.85+4.47+5.

$66+5.39+5.00+4.12+8.25+5.00+5.00+5.39+5.10+4.47+4.47+5.00+4.47+4.24+4.12+4.47+5.00+4.47+7.07+4.47+6.00+5.10+5.00+5.10+6.00+4.12=479.92$

関数電卓を使用して求めました。↓



座標上における距離の長さを実際の距離の長さは比例するので、次のような比例式を立てる。

$13.44 : 1053 = 479.52 : x$ (x=西宮市の周囲長)

$x = 1053 \times 479.52 / 13.44$

$x = 37569.535 \dots$

→約37570m→約37.57km

したがって西宮市の周囲長は約37.57kmと予想されます

感想・今後の課題

今回のレポート作成を通じて、スクラッチやChatGPTをより使いこなせるようになったと思います。また、西宮市の周囲長と最初に思いついた時は、どのようにして調べようか頭を悩ませていました。しかし、その方法を見つけ実際に計算や測定を続けていたときはとても爽快でした。また、いつか他の市の周囲長も求めてみたいと思います。また、今回行った作業で最も難しかったのは、スクラッチでスプライトが西宮市の境界線上を動くようなプログラムを立てることです。何歩動かすかや何度回すかの塩梅が大変でした。スプライトが変な方向に動いてしまうことが何度もありました。また、ChatGPTに2点間の距離の計算をさせたときに、「Python」というプログラミング言語を使用していたことが何度かありました。Pythonは他のプログラミング言語に比べて比較的難易度が低いそうなので、また勉強してみたいなと思いました。

使用したツール

- ・ ChatGPT
- ・ 関数計算desmos (<https://www.desmos.com/scientific?lang=ja>)
- ・ Google earth