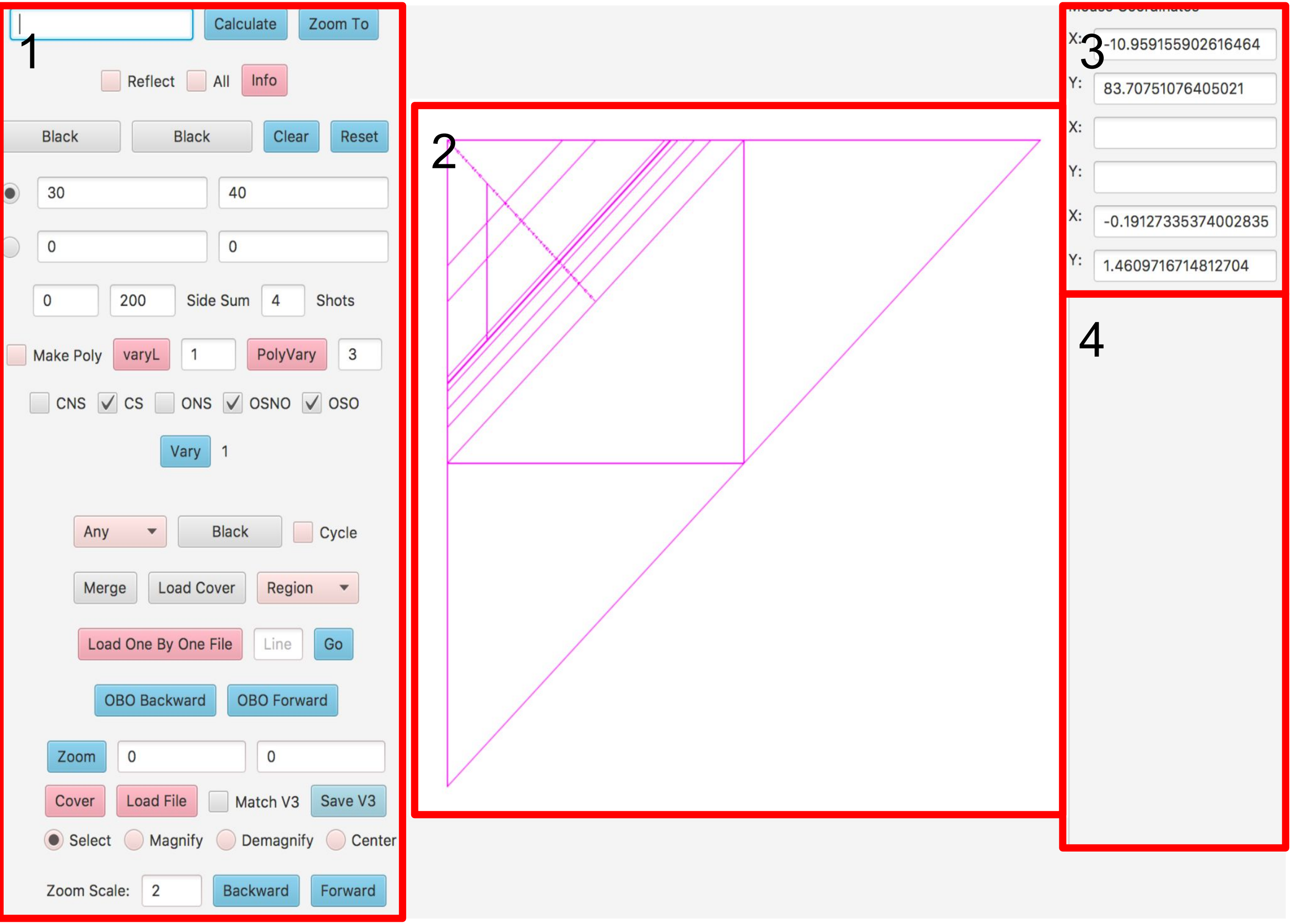
Billiard Viewer Everything



□ Reflect □ All

Black Black

● 30 40

○ 0 0

0 200 Side Sum 4 Shots

□ Make Poly 1 3

□ CNS ☑ CS □ ONS ☑ OSNO ☑ OSO

Vary 1

Any ▾ Black □ Cycle

Region ▾

Line

Zoom 0 0

□ Match V3

● Select ○ Magnify ○ Demagnify ○ Center

Zoom Scale: 2

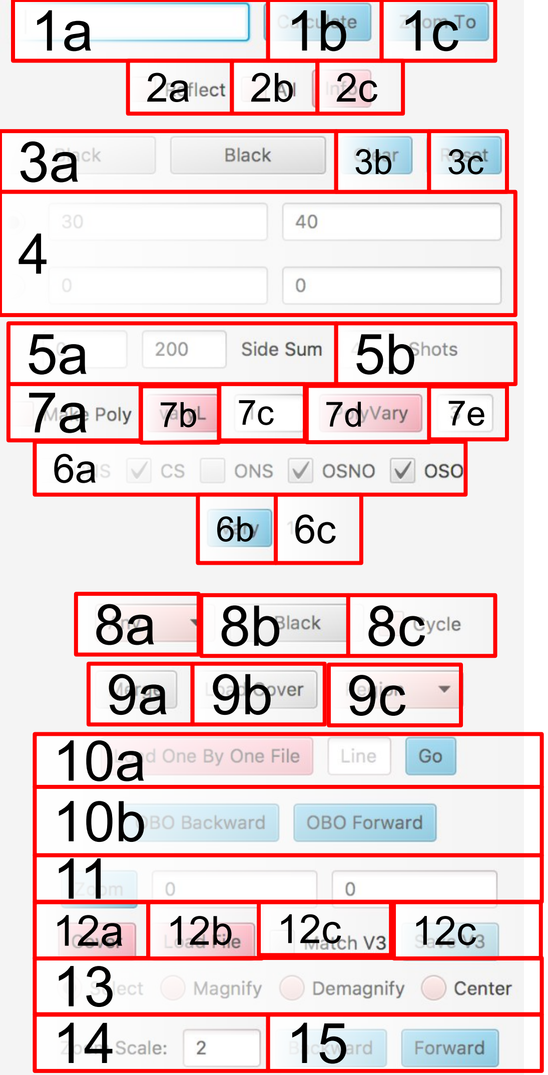
X: -10.959155902616464
Y: 83.70751076405021
X:
Y:
X: -0.19127335374002835
Y: 1.4609716714812704

1 - Tools

2 - Viewer

3 - Mouse Coordinates

4 - Code Sequences



1a - Code Sequence Box (triples require commas)

1b - Calculate Using Code Sequence

1c - Zoom The Code Sequence

2a - Reflect the Viewer

2b - Permutes x,y,z coordinates

2c - Information about the Code Inputted

3a - Colour of the Regions in Viewer

3b - Clears Viewer

3c - Resets to Default

4 - X Y Coordinates * (see instructions)

5a - Min Max Side Sum

5b - Number of Shots *

6a - Code Type

6b - Finds Codes

6c - Number of Codes Found

7a - Creates Polygon Vertices Coordinates *

7b - List of Coordinates

7c - Maximum Number of Regions Drawn From Coordinates of VaryL

7d - Polygon Coordinates

7e - Maximum Number of Subdivisions

8a - Total Number of Selected Polygons

8b - Colour of Cover Squares

8c - Cycles the Colours in the Cover

9a - Merges Abutting Covers

9b - Loads a Previous Cover

9c - Draws Selected Bounding Polygon

10a - Load One By One File *

10b - Direction of One By One

11 - Center at the Coordinates

12a - Cover a Polygon with Stables and Triples

12b - Draws Codes From a Selected File

12c - Find Intersecting Code From Clicked Location

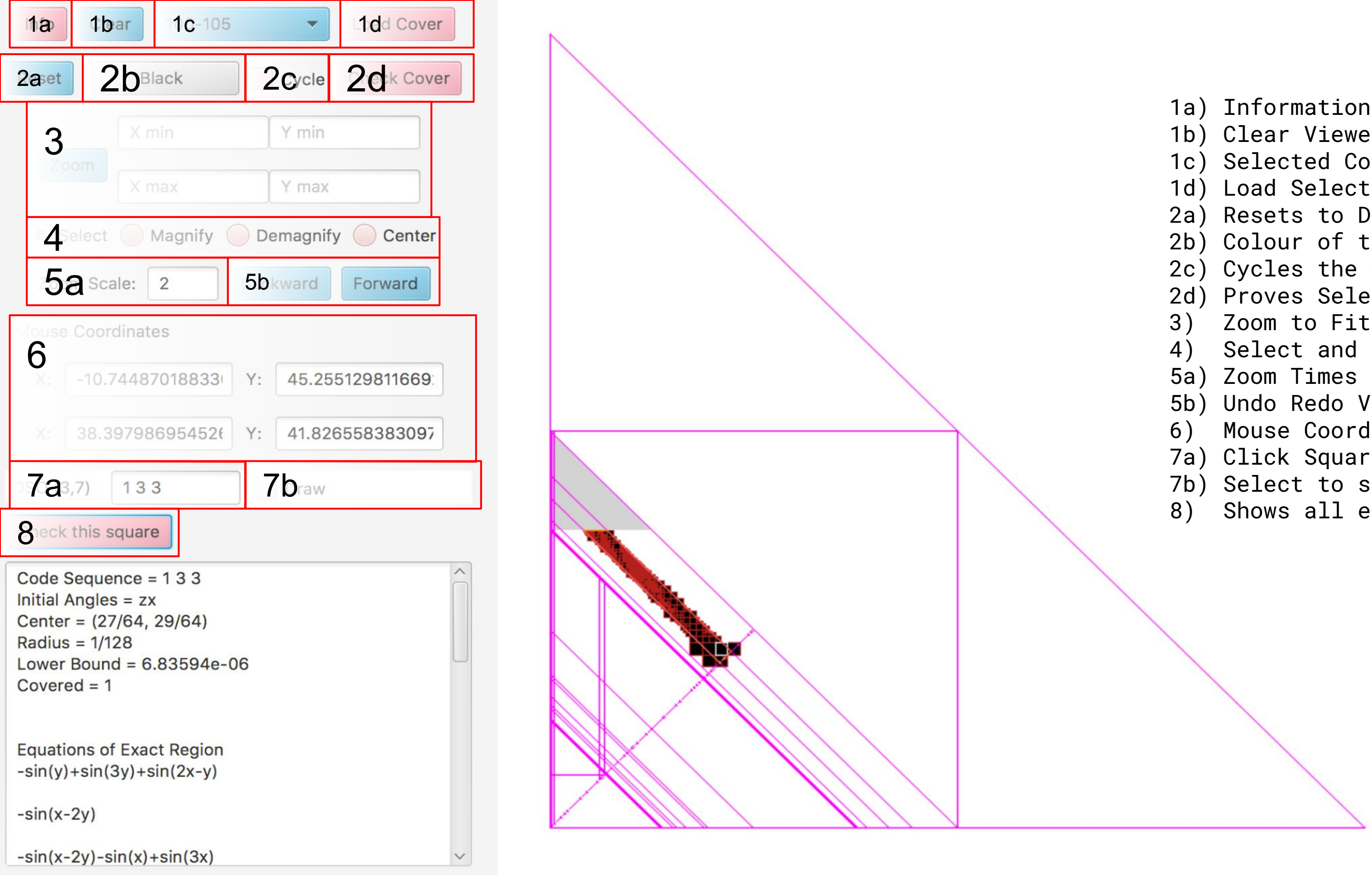
12d - Save Code

13 - Select and Press on Viewer

14 - Zoom Times Size

15 - Undo Redo Viewer

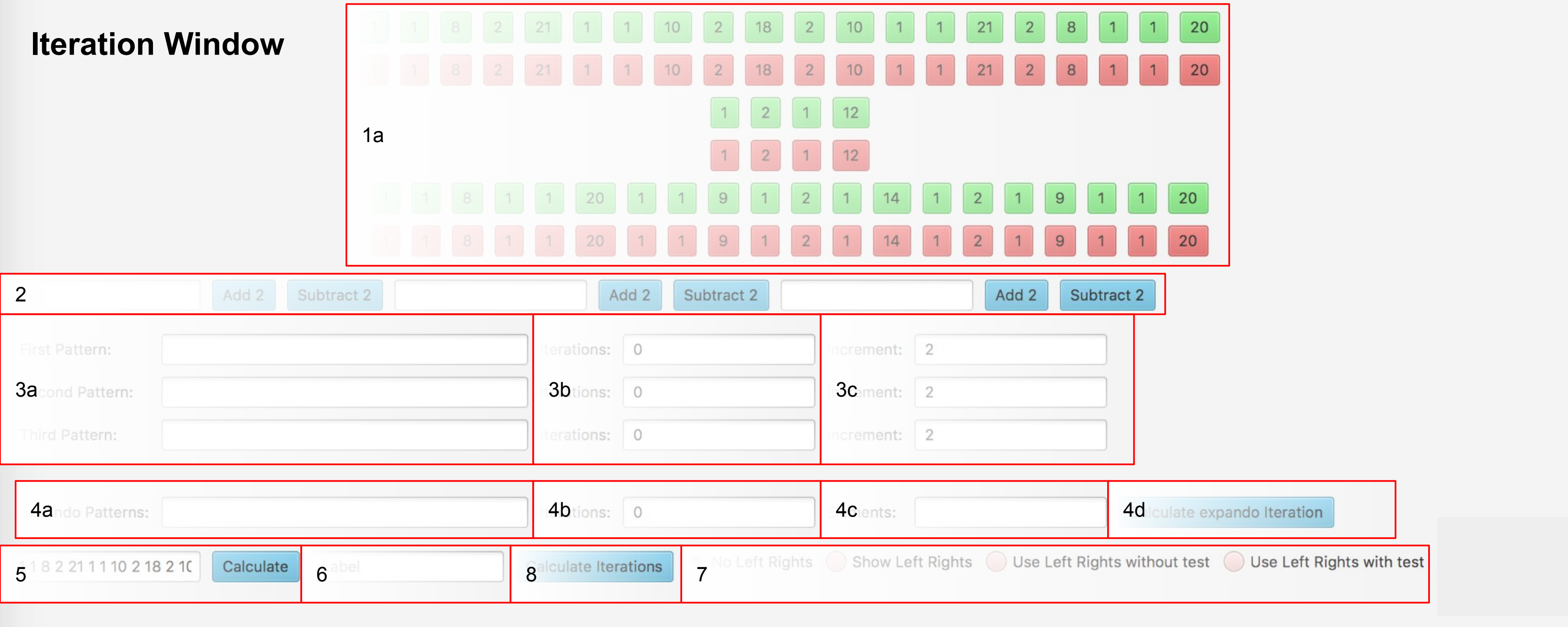
Billiard Viewer Covers



```
Code Sequence = 1 3 3
Initial Angles = zx
Center = (27/64, 29/64)
Radius = 1/128
Lower Bound = 6.83594e-06
Covered = 1


Equations of Exact Region
-sin(y)+sin(3y)+sin(2x-y)

-sin(x-2y)

-sin(x-2y)-sin(x)+sin(3x)
```

1a) Information about the Code Inputted
1b) Clear Viewer
1c) Selected Cover
1d) Load Selected Cover
2a) Resets to Default
2b) Colour of the Regions in Viewer
2c) Cycles the Colours in the Cover
2d) Proves Selected Cover
3) Zoom to Fit Coordinates into Square
4) Select and Press on Viewer
5a) Zoom Times Size
5b) Undo Redo Viewer
6) Mouse Coordinates
7a) Click Square to see Code Sequence
7b) Select to see type of Polygon
8) Shows all equation

Iteration Window



First Pattern:
Second Pattern:
Third Pattern:

Iterations: 0
Iterations: 0
Iterations: 0

Increment: 2
Increment: 2
Increment: 2

Expando Patterns:
Iterations: 0
Elements:

1 1 8 2 21 1 1 10 2 18 2 1C

Label

1) Increase/Decrease code Individually By 2
2) Add 2/Subtract 2 at specified positions (for triples separate by comma)
3a) Put in specified positions (for triples separate by comma)
3b) Put in specified number of iterations
3c) Put in specified increment
4a) Put in valid horizontal pattern code
4b) Put in specified number of iterations
4c) Put in specified pattern of each element
4d) Press Button to calculate expando iterations

5) Code Sequence Box (triples require commas)
6) Label (Not required)
7) Options for the iterations (read instructions)
8) Calculate iterations using 3 and 7

Prereqs: IntellJ or Eclipse (That can run gradle.), sqlite, javaFX, macOS

#RUNNING BILLIARDS EVERYTHING#
You need to use a mac and have Java 8 on it otherwise it may not work.
Once you have gotten the zip file
1. Unzip the zip file and in a new folder, a jar file will appear.
2. Drag that folder into the terminal and press enter.
3. Type java -jar followed by a space.
4. Open the folder and drag the jar into the terminal again and the terminal becomes your console and you can create or select your database.

When you make your first cover, new folders will be generated.

#RUNNING BILLIARDS COVERS#
You need to use a mac and have Java 8 on it otherwise it may not work.
Once you have gotten the zip file

1. Unzip it.
2. Run the jar inside the new folder by double-clicking since we do not need a terminal.
3. Try different covers using “Load cover”.

#SELECTING AND CREATING DATABASE#
A small menu will pop up to allow you to pick a database on sqlite to store your code sequences.

If a database has not been created yet, then click on new and add a name. Then click on "OK" to create. After creation, you will create a database and you can click on and and click on select. (All databases are stored in the "billiard-databases" folder.
This will open up the main viewer with tools/buttons on the left side and mouse coordinates on the right side.

#IDENTIFYING YOUR GIVEN OPEN SPACE#
Your polygon (or quadrilateral) coordinates will be given as below.

Ex:
8 58
9 57
9 55
8 56

Now go to the tools menu and click on "Cover" in the bottom left region.
These polygon coordinates should then be copied and pasted into the first big input box with the hint "Polygon". This is the first of three big input boxes
titled polygon, stables, triples.

Look towards the bottom of the Cover pop-up and make sure MRR is selected and use these suggested numbers as follows 22 20 [Calculate] 50 respectively.
(Calculate is used to outline your polygon and to find uncovered holes in the viewer. The 3 numbers around the calculator button are specifications.
First small box is the number of decimal places. //Do not need to change.
Second small box is the number of subdivisions. //May need to change/increase towards the end of covering the polygon.
Third small box is the list of point coordinates displayed where those points are not covered. //Generally 50-100 points is good enough for a single session to start.
Eventually you will need to use thousands of points and leave the program run over night or longer.)

#FILLING IN SPACES/HOLES (Stables)#
There are many ways of filling empty space. The possible ways of filling spaces will be loosely ordered by the size of the holes to fill.
Most values, such as subdivisions and side sum or code sum are dependent on your processor specs.
Loose ranges and example values will be called
low, medium or high and will be specified in the steps below.

1. Menu - PolyVary

Generally used for a new polygon. Subdivision should be set to a low-medium range (3 - 5). Side sum should be of medium range (800 - 1200). CS should be selected first.
Do not forget to change your number of cores beside "Shots"
Click on PolyVary and put in the given polygon coordinates appearing there and press the Vary button seen inside.
Copy all code sequences that show up in the console.
Paste these code sequences into the second big input box under Cover (if it is empty then there will be a hint text that says stables.)
Increasing the subdivisions, side sum or code sum can help find smaller holes, as well as zooming to enlarge the viewer with empty space which can help to find more
code sequences but in return will take longer.

2. Menu - Side Sum
When the side sum is over 1000 or so, it could be time to increase your point coordinates into the thousands. Then put your coordinate list inside varyL and press the varyL button inside.
Make sure that you check the box on Draw. Note the max boxes are in code sums not side sums. A typical start is CS max 333 and increase that as needed up to the 500's.
For the OSO max use a range of 122-166 or so and for the OSNO use a range of 50-136 or so. Caution: The checkboxes in the Tools Menu should be on for the CS first, then with the CS and OSO on and then lastly with the CS,OSO and OSNO on.

3. Menu - Match V3/Save V3
Used to find holes individually.

Should be used when almost finished with the cover but can be used whenever the user likes to.
Side sum should be medium to high (2000 - 5000), CS should only be selected.
Zoom in enough to see the hole.
Make sure Match V3's checkbox is checked. Click Select as the radio option.
Click on the corners of the given empty space on the viewer. Match V3 will find the common sequences of all the coordinates clicked on the viewer.
Check the console for something similar:

```
//-------------------------- Matching Code Sequences --------------------------//

CS   (56, 384) 1 1 6 2 20 2 9 1 2 1 19 2 8 2 19 1 2 1 9 2 20 2 6 1 1 19 2 8 2 20 2 8 2 20 2 7 1 3 20 2 8 2 20
3 1 7 2 20 2 8 2 20 2 8 2 19
CS   (88, 412) 1 1 4 1 1 18 1 1 4 1 1 18 1 1 4 1 1 18 1 1 4 1 1 18 1 1 4 1 1 18 1 2 1 10 1 2 1 19 3 1 7 2 19
1 2 1 10 1 2 1 18 1 2 1 10 1 2 1 18 1 2 1 10 1 2 1 18 1 2 1 10 1 2 1 19 2 7 1 3 19 1 2 1 10 1 2 1 18
...
```

Copy a single code sequence and put it in the sequence drawer and click on "Calculate".
If the code sequence covers what you wish it to cover, then put it in the "Stables" input box in Cover. (second input box)
If it does not cover what you wish, you can either try the next matching sequence or you can keep the sequences and repeat this process for the smaller empty space.
Alternatively you can also save them all together in any file you choose by pressing Save V3 and load them all when ready by pressing the load button.

If the hole looks like a line, you can click on both sides of the line to see if there is any matching code sequences. You can also try using a triple below.

#FILLING IN SPACES/HOLES (TRIPLES)#

These holes are very long rectangles that look like lines.
A triple is just a combination of stable, unstable, stable codes.
Example of a triple with commas:

1 1 10 2 23, 1 2 1 13 2 20 2 13, 1 1 12 2 20 2 13 1 2 1 13 2 20 2 12 1 1 23 2 12 2 20 2 12 2 23

To find the two stables, click one side of the line and then the other side and you should get the two stables as long as they intersect in a common boundary line.
To find the unstable involves loading low side sum (100 should be good enough) CNS and ONS and connecting the existing sequences to the left and right of the CNS/ONS sequence found.

To find the unstable use the two radio buttons next to the two rows of coordinates near the top in the Tools Menu.

1. Click on the first radio button and click on one end of the line (a close approximation is all you need) and its coordinates will appear.
2. Click on the second radio button and click on the other end of the line and its coordinates will appear.
3. Change the Side Sum to 144 or so and only have the CNS and ONS check boxes selected.
4. Press Vary and you will see a list of unstables in the console.
5. Copy all the code sequences found into a text file which we will call lines.txt
6. Click "Load File". Find and click on lines.txt. (some will say empty and others don't work).

7. Go back to the viewer and click anywhere on the given line. Find any CNS or ONS that appears in the Coordinate/Codes Column underneath the coordinates.
If none appears, try again with a bigger Side Sum and more shots. Hopefully you will find one and you can take the first unstable in the list.

Now go to the Tools Menu under Cover, paste the sequence stable, unstable, stable in the third input box as with the example and with the hint triples.
To double check you can copy the triple sequence into the box next to Calculate in the Tools Menu to see if the line has been filled or partially.

Alternatively if using a triple, you can try using a stable OSO or OSNO to cover that line.

#FINDING MORE HOLES TO FILL#

Go to "Cover" in the bottom left of the tools menu and click on the Calculate button after following
#IDENTIFYING YOUR GIVEN OPEN SPACE#
Go to the console. Within the generated text should be something similar:

```
1653711 stable squares used in the cover
0 triple squares used in the cover
2310 stables used in the cover
0 triples used in the cover
MRR at 22 decimals, deepest magnification 27
Total stable cost: 831208131
485334 squares were not filled in
8.2754087448120117 57.724614143371582
8.2933473587036133 57.561535835266113

...

Not Covered

Time elapsed: 22m 20.514s
```

Copy all pairs of coordinates and paste into an accessible text file.

From the Tools Menu, go to "Load One By One File" and find the text file that contains all the copied coordinates. The viewer will automatically move to the first coordinate in the text file.

To go to the next pair of coordinates, click on "OBO Forward". Repeat filling the holes until the given polygon is covered.

When done the console should say something like this.

```
// 0 squares were not filled in
// 1111 stable squares used in the cover
// 24 triple squares used in the cover
// 31 stables used in the cover
// 2 triples used in the cover
// MRR at 7 decimals, 1x1 number of squares, deepest magnification 17
// Total stable cost: 65809
// Covered
// Time elapsed: 0.128s
```

#WHAT SHOULD A GREAT PERIODIC HUNTER DO NEXT#

Go to the jar called Billiards Everything and you will find a folder called Cover. Send that to me and I will check it and run it to the All proof that goes through all the equations and we will add your name, date and cover to the Great Hunt. You can do that yourself if you wish by checking the All check box instead of the MRR check box. If you do, be prepared that it could take 20 times or more and days or weeks or months or years. To ensure that the polygon assigned is full look for two things:

// 0 squares were not filled in
// 1111 stable squares used in the cover
// 24 triple squares used in the cover
// 31 stables used in the cover
// 2 triples used in the cover
// MRR at 7 decimals, 1x1 number of squares, deepest magnification 17
// Total stable cost: 65809
// Covered
// Time elapsed: 0.128s

This shows that your polygon is covered and I will run the proof to ensure that the polygon has been fully covered.

#BONUS FUN ON ITERATIONS#

Iterations:

When you put in a code which can be a stable, a non-stable or a triple and press Calculate, the iterations bar comes up.

If you use a triple, you will see six rows of boxes, three green and three pink. Otherwise you will see two rows of boxes. Pressing a green increases the code by two and pressing the pink decreases the code by two.

Alternatively you can use the Add 2/Subtract 2 boxes to change a given code. In the empty box to the left, if you put in for example 3 6 10 then the 3rd, 6th and 10th code number will add 2 or subtract 2 to those spots. Similarly for triples if you put in 3, 2, 3 5 and press add 2 it will add 2 to the 3rd spot of the first stable, add 2 to the 2nd spot of the unstable and add 2 to the 3rd and 5th spots of the other stable.

You can also create patterns using the next three horizontal lines of boxes. For example using the First Pattern box and putting in 3 6 10 for those spots, and putting 12 in the second box will give the number of iterations and putting 6 in the third box will give the increment to all of those spots. Then press the button Calculate Iterations and you will see in the viewer all those regions belonging to all of those codes.

Furthermore, there are also four options you can choose from for calculating iterations:

First is the ‘no left rights’ option which means it doesn't use any previous left rights equations in the calculation for the following iterations, and then each time it calculates and uses the current left rights equations which will take the longest time.

The second is ‘show left rights' which does the same as the first but will also print out the left rights for each code in the iterations in the console.

The third option is ‘use left rights without test’ which means using the left rights found in the previous iterations in the calculation for the following iterations and thus it does NOT always give you the correct result. The trade off is if you trust the pattern, then this is the fastest option.

The last option is ‘use left rights with test’ which means using left rights found in the previous iteration in the calculation for the following iterations with a test. This test can filter out many wrong results and then use the normal method to calculate instead. It is the second-fastest option but again is not foolproof.

NOTE: the third and the fourth option works with only the stable codes and doesn't work for the unstable codes.

Expando Iterations:

Expando iterations are used when the code sequences expand horizontally in a pattern and where the left rights of the code sequences also have a pattern. Here is an example.

1 1 2 2 1 1 3 3
1 1 2 2 2 2 1 1 3 2 2 3
1 1 2 2 2 2 2 2 1 1 3 2 2 2 2 3
...

Left Right
(2, 0, 7, 0)
(8, 0, 5, 0)
(8, 0, 1, 0)
(4, 0, 1, 0)

Left Right
(2, 0, 9, 0)
(10, 0, 7, 0)
(10, 0, 1, 0)
(4, 0, 1, 0)

Left Right
(2, 0, 11, 0)
(12, 0, 9, 0)
(12, 0, 1, 0)
(4, 0, 1, 0)

...

To use and press the Calculate expando iteration button, put in the expando pattern box a pattern substituted with the capital letters.

For example input 1 1 2 2 A 1 1 2 2 B 1 1 2 2 A in the Expando pattern box and put in the elements box for example 2 2,2 2 separated by commas with A becomes 2 2 and B becomes 2 2. This also allows expanding patterns involving A,B,C,...

The Expando iteration has only two options to use:

a) use 'left rights with test’ means all the code sequences in the iterations are calculated the normal way which is the slower method and this acts as the test.

b) use 'left rights without test’ means using the left rights found in the previous iteration in the calculation for the following iterations. Again this is not foolproof.