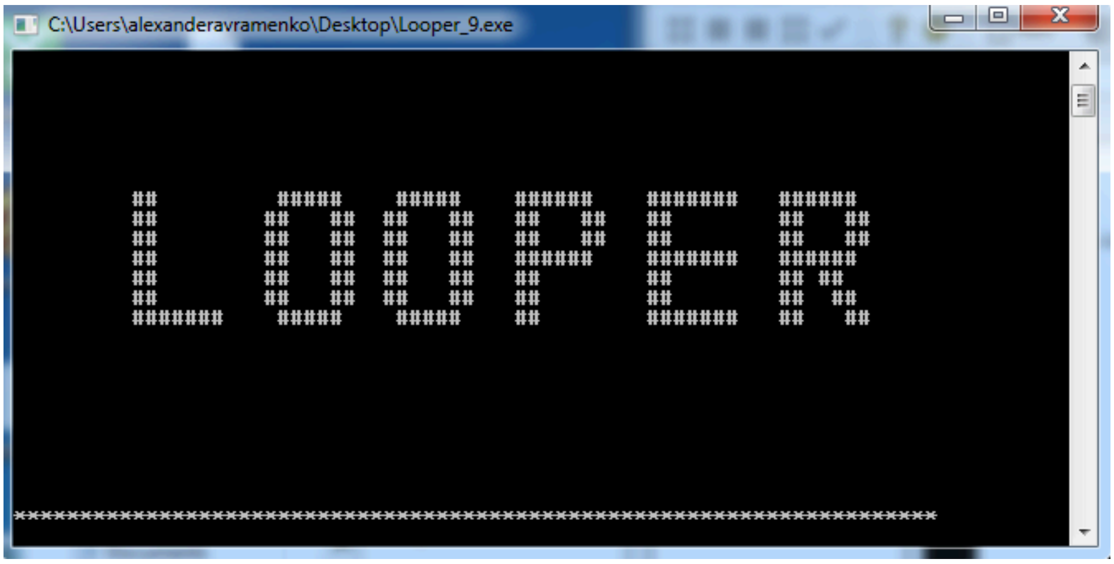
Looper (v.8) program description

Written by Alex Avramenko on 5th of August 2012. King's College London.
aaavramenko@gmail.com   +447411565631.



```
##         #####    #####    ######    #######   ######
##        ##   ##  ##   ##   ##   ##   ##        ##   ##
##        ##   ##  ##   ##   ##   ##   ##        ##   ##
##        ##   ##  ##   ##   ######    #######   ######
##        ##   ##  ##   ##   ##        ##        ## ##
##        ##   ##  ##   ##   ##        ##        ##  ##
#######    #####    #####    ##        #######   ##   ##

***************************************************************************
```

Introduction.

The purpose of the program is to use a ready input text file, to permute it, and to feed it in to Ray/Reflec. An input text file is simply the text that one would type using the keyboard, ad verbatim, in to the Ray/Reflec program, saved as a text file.

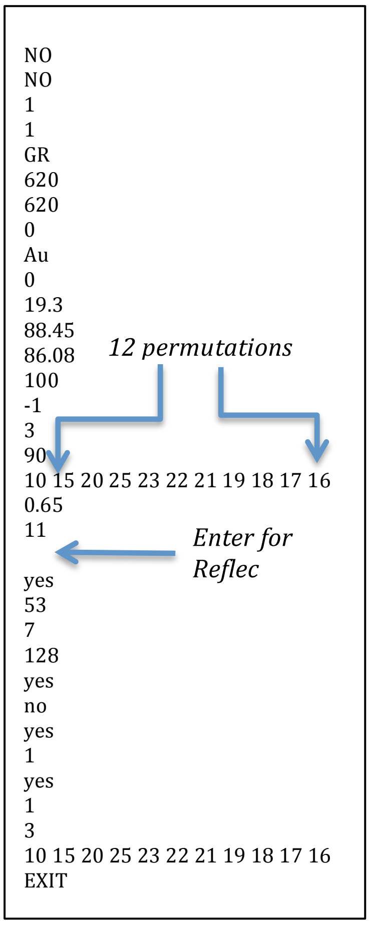
Key features.

-Can run up to 180 loops.
-Stores up to 350 lines of feed.
-Can permute in 2 Dimentions.
-Can write a simple sequence between specified values.

Disclamer.

I am a novice programmer and also had very little time to write the program. There are bound to be many bugs and errors. None the less I hope in the current state the program will still be useful for a wide range of uses.

Sample input file.


12 permutations
Enter for Reflec

```
NO
NO
1
1
GR
620
620
0
Au
0
19.3
88.45
86.08
100
-1
3
90
10 15 20 25 23 22 21 19 18 17 16
0.65
11

yes
53
7
128
yes
no
yes
1
yes
1
3
10 15 20 25 23 22 21 19 18 17 16
EXIT
```

To the left is a Reflec input text file (.txt). The permutations are entered on the same line, and the next input is put in on the following line. In this case one might select the number of runs as 12.
If an enter is required to skip something in Ray/Reflec, an enter can be put in the text file.

Process of the program.

The program starts with asking the user for the number of runs of the program.
It will then create a matrix where each line is a row of that matrix, and each run is a column.
This matrix will then be fed with the input file.
Once the user is done editing the matrix in editor mode, the program crates Run text files (Run1, Run2,…). These run text files are then fed to Ray/Reflec using the simple Terminal command:
Ray.exe < Run1.txt , *Ray.exe < Run2.txt …* and so on.
Once all the run files have been run, the program will then delete them. (It will pause just before to give you time to have a look at what the program created to catch any errors).

Editor.

If (yes) has been selected to the question of whether to edit the file, then the editor mode will be entered. If enter is pressed the editor goes to next line. There are five options here:

Value (v)

A value that will be the same for all runs.

Jump (j)

Allows to jump to another line. For example if on line 14: entering 3 will skip to 17, entering -5 will go back to 9, entering -15 will cause the program to crash, entering a number larger than the last line will cause to skip the editor.

Sequence (s)

Here one will be prompted to put in the first number and the last number of wanted sequence. For example if runs= 5, the first number is 1 and the last number is 3, the sequence will be:

First number	**in-between**	**last number**
1	1.5 2 2.5	3.

Manual (m)

Will give the option to put in values one by one for each run. Does not yet work well for large amount of runs.

File (f) (2 dimensional uploading)

Allows uploading a file with parameters. The file should be in the same directory as Ray/Reflec. Cannot read enter as a new line.
Here the user will have three options for the *amount* of values to put in. One can put the amount as: positive number, negative number or zero. This will affect the way the program treats the number.

Positive number

This will read the specified amount of values and repeat them in a string way.
For example the following is entered: (runs= 15), (amount=5) (file reads: **a, b,c d, e, f, g**...) --- *instead of commas the file should have each value on a new line*.
The program will then read:
a b c d e
and will dispatch it to Ray/Reflec as follows:
a b c d e a b c d e a b c d e .

Zero

This will just read the file for the reps required. For example if (runs=15) (file reads: **a, b,c d, e, f, g**...)
The program will then read:
a b c d e f g h i j k l m n o

Negative number

This will read the amount of values specified and will repeat each value *reps* times.
For example if the following is entered: (runs= 15), (amount= -5), (reps= 3) (file reads: **a, b,c d, e, f, g**...)
The program reads:
a b c d e
and will dispatch it to Ray/Reflec as follows:
a a a b b b c c c d d d e e e

2 Dimensional matching.

If 2 dimentional variation is required (lets say for a grating 5 energies and for each energy 10 profiles), then the File option can be used in the Editor.
One would first create two text files each with the parameters, then one would input then in the editor mode as follows (selecting option f):

INPUT FFILE (Spaces are real entries)

```
gr
150
150


-1

88.4907
86.0780
177.25



15




53
7
3



3
15
9
```

Energy eV

500
600
700
800
900

Profile nm

1
2
3
4
5
6
7
8
9
10

abc.txt

The Energy is entered as amount=5 and Profile is entered as amount=-10, reps=5. Thus one must also enter the runs as 50 (10*5).

The abc.txt file should also be provided with Looper and contains 1000 permutation of three alphabetic letters, to make sure the name of the saved file always varies.

Multiple ASCI files can be most easily read with Origins(the really old version) and then for ease of use can simply be copied in to excel.

Looping many things at once.

I have noticed that the Looper program does not use the full capacity of the computer when running. Since running the full 1000 loops can take almost an hour (on my computer, running Reflec), one can actually just copy and paste a few Ray/Reflec folders and run a few Loopers at the same time. For I managed to run 6 Loopers at one time without slowing down my computer too much, but it really depends on the machine.

Example of use.

Below is described one example of use.

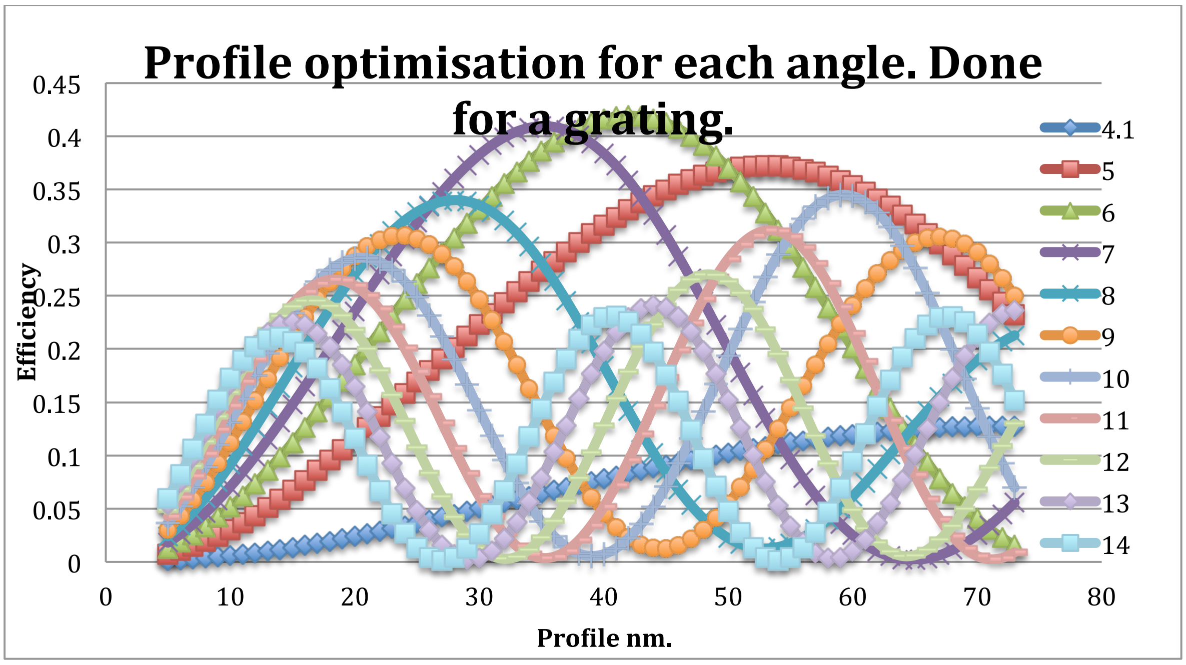


Fig. 1 An example of use of Looper program. Each point on the diagram is a run of Reflec. In total 759 runs. (24, 288 enter clicks if done manually)

As can be on Fig. 1 is an example of the use of Looper program. The aim was to find the optimal angle. For each angle however only the optimal profile was required. This meant that for several angles the graphs of their efficiency dependence on profile, had to be constructed. This can be done with 5 runs of Looper. It took around half an hour for the whole run on my machine as I ran them in separate folders at the same time.

Memory management.

Unfortunately due to the unforeseen memory management problem, the program uses one matrix to store all its values. This means that it is limited to 1,000,000 units of code.
This means that it is limited to:
180 runs.
16 characters per unit.
350 lines of input.
This will be the first thing I will try to update in this program and if need be you can freely set this parameters in the source code for your own use.

Just remember that **runs * character per line * lines of input = 1,000,000** .

```
// rows* columns * senence_sizew = 1,000,000
#define ARRAY_SIZE_rows 350
#define ARRAY_SIZE_columns 180
#define SENTENCE_SIZE 16
```