A possible explanation for the Board Cycle Bug of Microsoft Minesweeper

Rodrigo S. Camargo*

Department of Civil Engineering, Federal University of Espírito Santo, Brazil
minesweeperclone@yahoo.com

SUMMARY

The Microsoft version of the game Minesweeper has a known bug that makes sequences of boards appear again after generating thousands of boards, in a cycle, disturbing the random nature of the game. This article exposes the results of a research made in 2004, as well as the subsequent secondary discoveries, like the existence of boards made of combinations of mines from two consecutive boards of the cycle, the convergence of these boards towards the cycle, the fact that the mines of a given board can be put in a real order, and the relation between the locations of a same mine, if used to build beginner or intermediate boards. Finally, it gives a probable single explanation for all the phenomena observed, based on how pseudo-random number generators work, and on the probable conversion of these random numbers into boards.

1. Introduction

The Microsoft version of Minesweeper (the one that comes with Microsoft Windows) has a bug known as *Board Cycle Bug*. It is basically the fact that if a player starts a new game several thousands of times, the initial sequence of boards will eventually appear again, in an infinite cycle ([4]). However, this cycle is so huge that it is hardly noticed by most players.

On the other hand, this bug is in conflict with the random nature of the game, which assumes the boards to be truly random, or at least up to an unnoticeable degree. Also, this has been source of much controversy over the years, since any player with enough knowledge about this bug can take advantage of it to cheat, achieving unreachable scores.

The history of how this bug was discovered is really interesting, and has been well documented in some places on the internet ([1], [6]). It was possible mainly because of the appearance of a very easy intermediate board, which quickly became known as the *dreamboard*. As it was enormously easy, it was almost impossible not to get an excellent time, and many people broke their records on it. By comparison, those people noticed their records were achieved in the same board. Later, as videos started to become popular, people noticed that the boards preceding the *dreamboard* were also always the same, and then the idea of a cycle was fully formed.

Indeed, there is some evidence that some people have studied extensively this bug in the past, most of them active players. There is also evidence that some people did use this knowledge to cheat ([1], [5], [6]). So, not everything in this article is new content for some people, but certainly it is for the general public.

This article presents the results of a research made in 2004 and the discoveries acquired from it, as well as a possible explanation for the reason of this bug, based on the essential nature of pseudo-random number generators.

2. Board collecting process

2.1. Generating the database

The first step to start researching was to capture, or collect, all the boards of the cycle in a database. The approach of this was to click several squares on a board, until hitting a mine, revealing the positions of all of them. After that, these positions (columns and rows) are noted, and a new game is generated by pressing F2, or by clicking the smiley button. And the process is repeated thousands of times.

Of course a computer program was made for this purpose. It could emulate mouse clicks on the screen without having to physically move the mouse, or press its buttons. It was necessary to let the windows of both Minesweeper and this "collector" program (it was not properly named) completely visible on the screen. Then it was necessary to "initialize" it, providing the board width and height in squares, as well as the coordinates of the Minesweeper window, in pixels, relatively to the top-left corner of the screen.

It was programmed to perform the following sequence of actions repeatedly:

Accepted 23 September 2006

1. Click the smiley button to generate a new board;
2. Click every square of the board, in the order of normal reading, that is, starting from the top-left corner and going to the right, until clicking all the squares of the row, then going to the next row, until reaching the bottom-right square;
3. Emulate the pressing of the *PrintScreen* key;
4. Paste the copied image in its own window;
5. Analyze the captured screenshot, and extract the positions of all mines;
6. Save the board to the database.

The program was left running like that until stopped manually, some hours later. After stopping it and looking for repeated boards in the database, it was able to record the entire cycle more than twice. This information was used to adjust the time the program was left running in the following captures.

This process was repeated several times for both beginner and intermediate boards. It was possible to discover the existence of not only one, but two different cycles for each level, with the following lengths:

- Beginner cycle 1: 24320 boards
- Beginner cycle 2: 24304 boards
- Intermediate cycle 1: 12096 boards
- Intermediate cycle 2: 12064 boards

These numbers were already known by some people who have already studied board cycles in the past. Additionally, the *dreamboard* is present on the intermediate cycle 1, and was recorded in the database as the board number 4232 of the 12096 in it.

2.2. Correction for the *mine shift* effect

The Microsoft version has an effect known as *mine shift*: if the first click of the game is made on a square containing a mine, this mine is going to be moved to the first mine-free square starting from the top-left square and counting to the right ([2], [3]). This is done to make it impossible to hit a mine on the first click.

The method used previously to extract the location of all mines – clicking systematically all the squares of the board – certainly shifted the mines of the boards that had originally a mine in the top-left corner.

So, before starting any analysis, all the four databases had to go through a "correction" process. Initially, it was necessary to open Microsoft Minesweeper and manually click randomly until hitting a mine to reveal all of them. Then, the revealed board was identified and found in the database. This was a kind of synchronization between the game and the "collector" program.

Then, this program was slightly modified to perform one click in the first *opening* found (not simply in the first square) in the next board, counting from the bottom-right square, and moving to the left. This was only possible because it always knew the next board that would be generated, due to the previous synchronization with the game. After this opening was hit, the board was safe from being altered by the effect of *mine shift*. Being sure of it, the program worked as usually, clicking systematically all the squares of the board, to hit one and identify the position of all mines. Then the board was captured and saved to a new, corrected, database.

This "correction" process was a really tough part of the research, long and boring. Of course it could have been avoided, by modifying the board collecting process, in a number of ways.

2.3. Ethics

None of these databases, neither the "collector" program were published in any way, and will not be published. Doing it would ruin the fun of the game, and would make it very easy for cheaters to claim valid scores on invalid games.

Any possible public exposition of these databases or development of programs such as "trainers", based on what is in this article, is a strongly disencouraged and disapproved attitude.

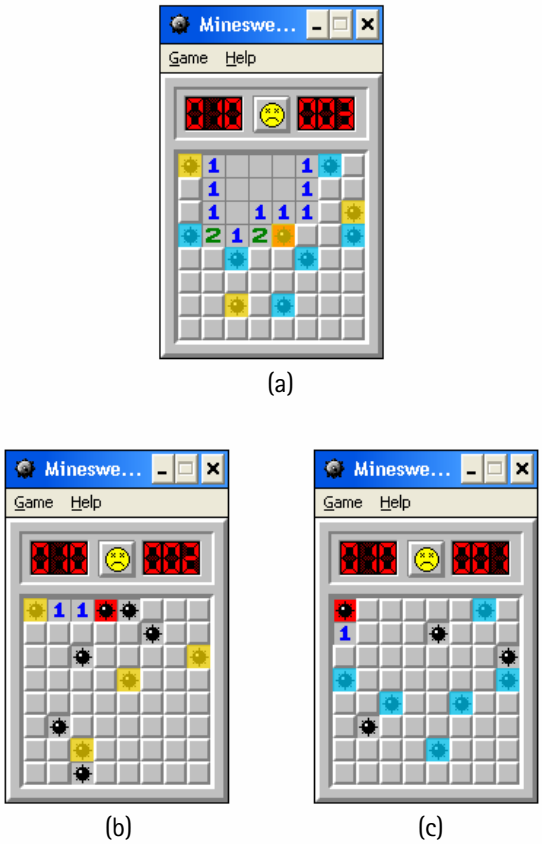


Figure 1. A "strange" board is formed by mines taken from two consecutive boards of the cycle.

3. Analysis of the boards

3.1. Convergence of the first boards generated

One of the first curious facts of the analysis appeared when trying to identify a random board of the game in the database. Sometimes, a board appearing on the game could not be found in the database, possibly indicating that a board outside the cycle was found.

However, after giving out searching for this "new" board, and generating some more boards, it became easy to find one of them in the database. After closing Minesweeper and opening it again, the same phenomenon showed up: the first boards generated by the game seemed not to exist in the database, but after generating a reasonable number of them, all of the subsequent existed.

After studying those "strange" initial boards in more details, it was possible to notice that they were not exactly new: some of its mines matched to a board in the database, and the other mines matched to the board following that one, in the database.

Figure 1 (a) shows one of these "strange" boards. Some mines are artificially colored yellow, and some are colored blue. Figure 1 (b) and (c) show two consecutive boards of the database of the beginner cycle 2. The first has 4 of its mines colored yellow, and the second one has 6 mines colored blue, matching the ones on (a). It is easy to see that the board in (a) is actually a mixture of the boards (b) and (c).

The next board generated after the one in figure 1 (a) also showed to be formed by the remaining 4 mines

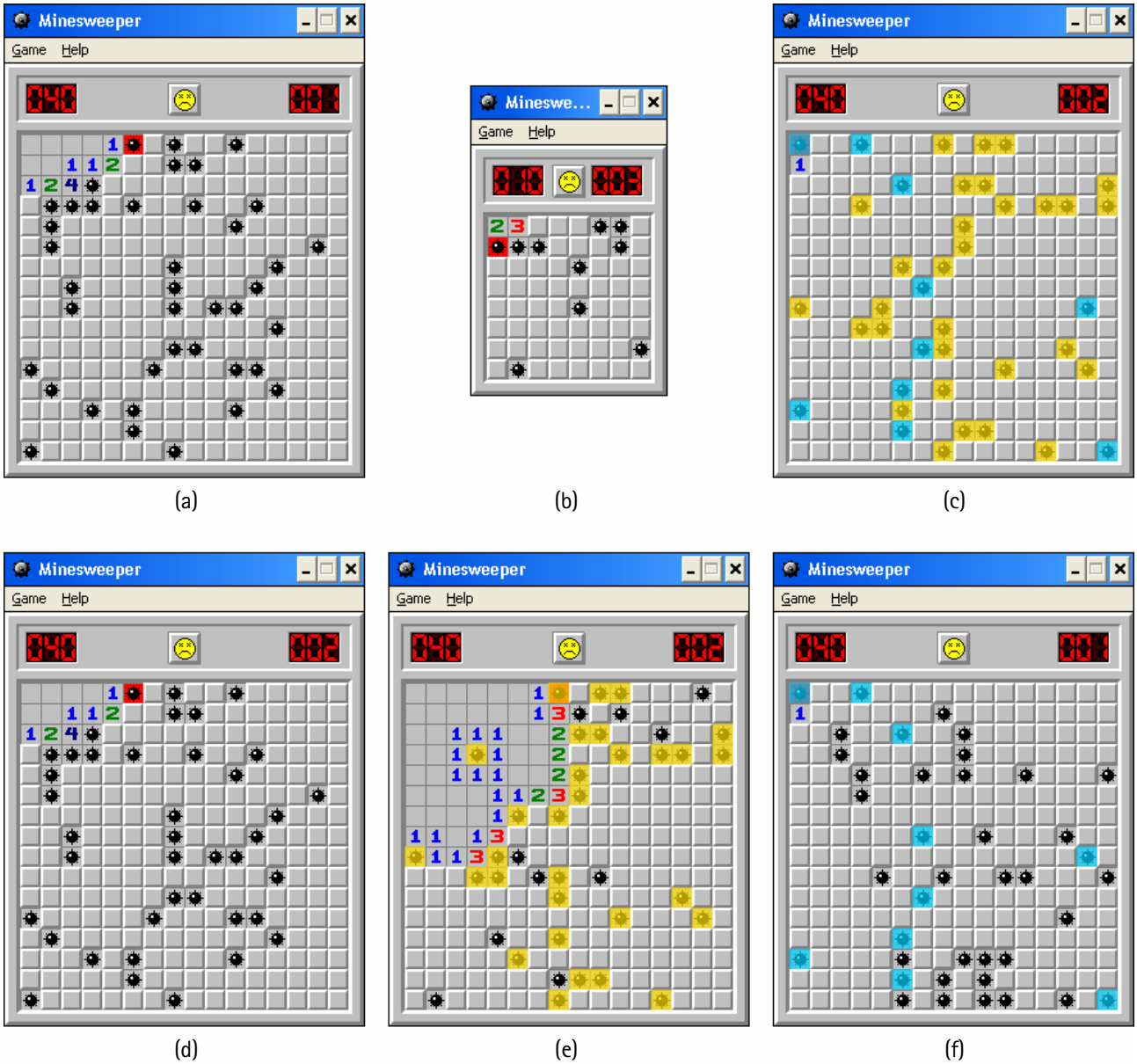


Figure 2. The sequence of boards that show up with the experiment of suddenly changing the level to beginner compared to the normal sequence of boards of the cycle.

of figure 1 (c) – the ones not "borrowed" – and by 6 mines of the board following that one in the database. And so on.

Also interesting is what happens with the number of mines "borrowed" from each board to make a "strange" board. The example of figure 1 used 4 mines from a board and 6 from the following one. As more boards are generated, this rate changes gradually. The number of mines used from the "previous" board tends to decrease, while the number of mines used from the "next" board tends to increase.

It seems that the way Microsoft Minesweeper generates boards leads to a kind of "convergence". The first boards generated after loading the game seem to be formed by a mixture of two consecutive boards of the database, but, as more and more boards are generated, they "converge" to the boards of the cycle, and once entering it, it never gets out.

3.2. Order of the mines in a board

One of the experiments made to check the behavior of the convergence to the cycles was the following:

After changing the level to intermediate and pressing F2, to generate new games, several hundreds of times (actually by keeping it pressed for some seconds), it can be assumed that the game will be inside one of the two cycles. Then, what would happen after changing the level to beginner and then right after back to intermediate again? Will the game continue in the cycle or not?

The answer, after performing this experiment, is *no*. Apparently, another "strange" board was formed as result. Then, pressing again F2 several times made the

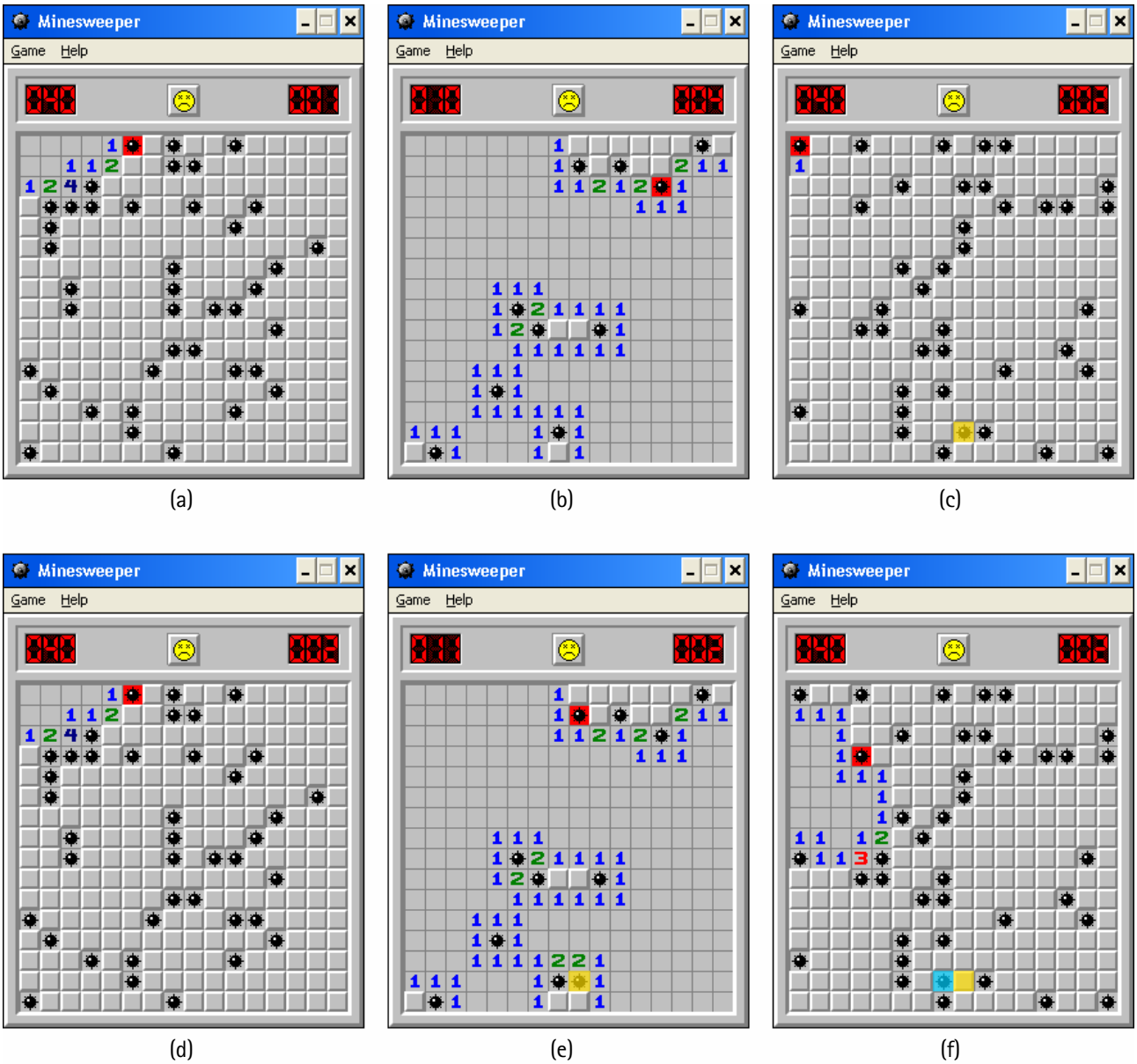


Figure 3. The sequence of boards that show up with the experiment of suddenly generating custom intermediate boards with 10 and 11 mines.

game converge back to the cycle, as if the game was closed and loaded.

Figures 2 (a), (b) and (c) show the sequence of boards obtained for the experiment. Figures 2 (d), (e) and (f) show the sequence of boards that would appear if F2 was simply pressed, without changing the level to beginner. So, (d), (e) and (f) are three consecutive boards of the cycle.

The interesting thing is that the out-of-the-cycle board on (c) has 30 mines and 10 mines in common with the inside-the-cycle boards on (e) and (f), respectively. These mines are artificially colored yellow and blue, also respectively.

From this, it can be deducted that, after changing the level to beginner, the "first 10" mines of the next "normal" board (e) were used to build the beginner board (b). Also, after changing the level back to intermediate, the "next 30" mines of board (e) and the "first 10" mines of board (f) were used to build the "strange" board (c).

This idea of "first" and "next" mines of a board means that the mines of every board can be put in a kind of order.

Another possible experiment is to repeat the previous one, but, instead of suddenly changing the level to beginner, generate a custom intermediate board with 10 mines. A similar result will occur, shown in figure 3 (a), (b) and (c): the next board generated with 40 mines (c), will also be formed by the last 30 and the first 10 mines of the boards in figures 2 (e) and (f), respectively.

Another interesting thing is that the custom board on figure 3 (b) is formed exactly by the mines not colored yellow of figure 2 (e). This confirms that these mines were really the "first" mines of the latter board.

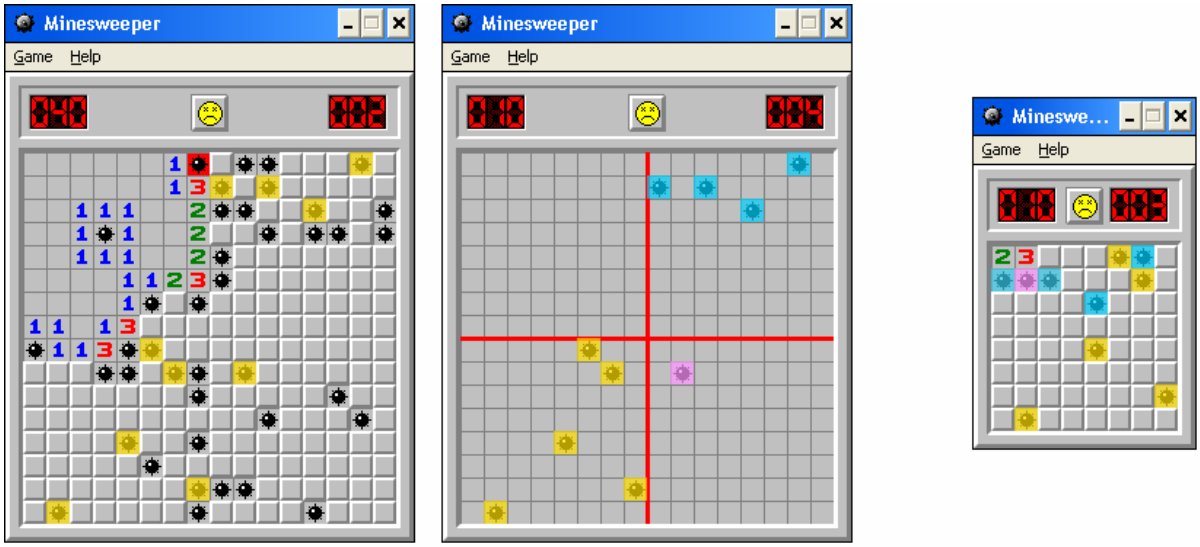
Figure 3 (d), (e) and (f) show the resulting sequence of generating a custom board with 11 mines. The board of figure 3 (e) is generated by the first 11 mines of figure 2 (e). The next board generated with 40 mines is, then, formed by the last 29 and the first 11 mines of the boards in figures 2 (e) and (f), respectively.

The difference between the figures 3 (b) and (e) is only one mine, colored yellow in (e). This mine can be, then, labeled as the "eleventh" mine of the original board on figure 2 (e). It is interesting to note that this same mine (also colored yellow) appeared on figure 3 (c), but did not appear on (f), because it was "taken" from it to complete the 11 mines of (e). Finally, the "eleventh" mine of figure 2 (f) was taken to be the last mine (colored blue) of the board of figure 3 (f).

Similarly, but generating custom boards with 12, 13, or any other number of boards allows to "number" all the mines of the board of figure 2 (e). And this can be extrapolated to number all the mines of *every* board of the cycle. The infinite cycle of boards can be then understood as actually an infinite cycle of *mines*, which can be grouped in any sequence, in the order they appear, to form boards with any number of mines.

3.3. How each mine of the cycle is shown on boards with beginner and intermediate sizes

The board of figure 2 (d) can have all of its mines numbered. If, instead of being used to generate a board with 40 mines, these mines were used to generate a custom intermediate board with 10 mines, it would be simple to know how it would look: it would be formed by the first 10 mines out of these 40 mines, as can be



(a) (b) (c)

Figure 4. The apparent way the first 10 mines of an intermediate board are used to generate a beginner board.

seen in figure 3 (b).

But what would happen if these mines were used to form a *beginner* board, also with 10 mines? How would the first 10 mines of that intermediate board be arranged in a beginner board, since they have different sizes?

Actually, this has been done on figure 2 (b), and the answer to this question comes quickly, after using these 10 mines to generate custom beginner boards with increasing number of mines from 1 to 10, and observing their locations.

The result is that the mines originally located in the first 8 rows and 8 columns of the intermediate board appeared in the beginner board in the same places. And, curiously, the mines originally located outside the top-left 8x8 squares of the intermediate board appeared in the corresponding squares of the beginner board, as if the original board was divided in 4 "quadrants", and as if these quadrants were superposed.

Figure 4 illustrates that. Shown in (a) is the original board of the cycle, equal to the one on figure 2 (e), but with the first 10 mines colored yellow. These 10 mines are shown isolated on (b). Still on (b), the board is divided in 4 quadrants of 8x8 squares, and the mines on each quadrant are colored differently. If these 4 quadrants are superposed, the board of (c) is formed (the colors are preserved to maintain their relative positions). Then, comparing this to figure 2 (b), it is possible to see that they are equal.

This is a somewhat curious effect, because it is so well structured that, at first sight, seems to have been made on purpose by the author of the game. However, it is very improbable that this behavior would be coded inside the game, because beginner and intermediate are not the only levels of the game, and doing a code to deal with such a particular case of the game is not a common practice for programmers. On the contrary, this behavior *suggests* that it comes naturally, as a result of some part of the program of the game.

3.4. Determinant factor for existence of cycles

Although beginner and intermediate levels do have cycles, it is still not clear why cycles do not seem to exist on expert level. While the board size of beginner and intermediate levels are 8x8 and 16x16 squares (multiples), expert size is 16x30. Also, while beginner and intermediate levels have 10 and 40 mines (also multiples), expert has 99.

The two main suspects for that can be then the board size and number of mines. In an attempt to check if the number of mines is the responsible, the "collector" program was used again. However, attempts were made to find cycles on custom boards with different number of mines, and the results are shown on table 1.

Table 1. Relation between different number of mines and cycle length.

Board (height x width)	Mines	Cycle length
8x8	10	24320
8x8	20	11016
8x8	40	4256
16x16	10	51496
16x16	20	25224
16x16	40	12096
16x16	80	5504
16x16	99	4216
16x16	120	3272
16x16	139	2640
16x16	160	2088

Apparently, the number of mines does not seem to be determinant for the existence or not of cycles, since all of them did show cycles.

Although not extensively studied, some tests were also made for variable board sizes. The results are shown in table 2.

Table 2. Relation between different board sizes and cycle length.

Board (height x width)	Mines	Cycle length
16x8	40	11016
16x8	80	4232
8x16	40	11016
8x16	80	4216
12x12	40	-
24x24	90	-
16x24	200	-
24x16	200	-

These results deserve special attention. Rectangular 16x8 and 8x16 boards did show cycles. So, the fact that the expert board is not square can not be a reason for the inexistence of expert cycles. Also, the square 12x12 and 24x24 boards did not show cycles. So, the fact that the beginner and intermediate boards are squares can not be a reason for the existence of beginner and intermediate cycles.

Interestingly, rectangular 16x24 and 24x16 boards did *not* show cycles. These sizes were chosen to test if the dimensions of a board must be multiples of 8 for the presence of cycles, and apparently this is not true. The next test would be if board dimensions must be powers of two for the presence of cycles. So, the next tests would involve boards with 8x32, 32x8, 16x32, 32x16 or 32x32 squares, but any dimension greater than 30 mines seems to be unreachable for Microsoft Minesweeper, even using tricks, like editing the registry, etc.

So, these results, although very inconclusive, only *suggest* that the board size determines the existence or inexistence of cycles, not the number of mines. Probably this explains also why beginner boards of the Windows XP version of Microsoft Minesweeper (which have 9x9 squares, instead of 8x8) do not have any known cycle.

4. A possible explanation

4.1. Pseudo-random number generators

It is first necessary to understand how pseudo-random number generators work. These numbers are not truly random. They are just results of calculations that depend on one or more previously generated numbers.

This dependence results in two known effects that occur in all PRNG's: the need for a kind of "initialization", or seed; and the occurrence of cycles.

Typically, computers use the CPU internal time as the seed. This is sufficiently good for most applications. The real "problem" is the occurrence of cycles, because this can be potentially dangerous for programs that depend on good randomization, like Minesweeper.

This is basically the fact that after generating millions of numbers, all PRNG will eventually repeat the same sequence, entering an infinite loop. With luck, the amount of numbers in the cycle "length" is huge enough that users of programs will not notice the existence of a cycle. Unfortunately, this was not the case concerning Microsoft Minesweeper.

A good analogy is to compare the numbers generated with the ring of dots shown in figure 5. The arrow indicates the number that will be returned by the PRNG in the next time it is called. As a number is returned, the arrow advances one position in the clock-wise direction, and stays waiting for the next call. Eventually, the arrow will complete a full rotation, and the same sequence of numbers will be returned.

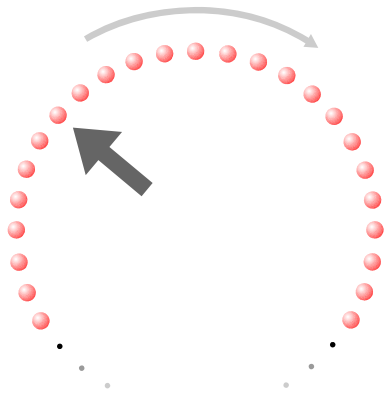


Figure 5. Each number generated by a PRNG can be compared in analogy to a dot placed in a ring.

The "initialization" of a PRNG basically changes the arrow position to a new position in the ring, determined by the seed. This new position can generally be considered better randomized, since the seed depends on a physical value (the CPU internal time, normally). This works well, and some programs do it only once, when loaded. This way, every time the program is loaded, the arrow starts at a given position, and the sequence of numbers can be taken from the position of the arrow, and ahead.

This analogy is good to state that the quantity of numbers that a given PRNG can generate is *finite*. Also, the chance of getting the same number twice, after initializing a PRNG several times, is reduced if this quantity is very big. This quantity is called the *period* of the PRNG.

Typically, those numbers generated also lie within a range, and usually it is between 0 and 1, including 0 and excluding 1. An example of a possible output for several requests for random numbers can be, for example:

0.705548
0.533424
0.579519
0.289563
0.301948
0.774740
0.014018
0.760724
0.814490
0.709038
(...)

4.2. Using random numbers to place mines

One possible way to acquire mine coordinates from these numbers, for an intermediate board, would be calculating $1+\text{int}(16 \cdot x)$, where x is each random number. Grouping these values at every two, they can be used as the coordinates of a mine in the board, as the example of table 3.

Table 3. Conversion of random numbers into coordinates for the mines.

x	$1+\text{int}(16 \cdot x)$	Coordinates (column, row)
0.705548	12	(12, 9)
0.533424	9	
0.579519	10	(10, 5)
0.289563	5	
0.301948	5	(5, 13)
0.774740	13	
0.014018	1	(1, 13)
0.760724	13	
0.814490	14	(14, 12)
0.709038	12	
(...)	(...)	(...)

In the same way, calculating $1+\text{int}(8\cdot x)$ would convert the random numbers to the range from 1 to 8, making the resulting coordinates suitable for filling a beginner board, for example.

This algorithm for placement of mines uses two random numbers to place one mine. Of course there are other algorithms that can place one mine using only one number, or even algorithms that can place several mines with one number.

However, the usage of two numbers for each mine explains the existence of *exactly* two cycles for each level, and not only one, or even more. In other words, this algorithm is able to explain why the number of cycles for each level is exactly two.

The understanding of it is simple: if the initialization of the PRNG places the "arrow" over, for example, the number in the second line of table 3 (0.533424), then the coordinates of the generated mines will be (9, 10), (5, 5), (13, 1), (13, 14), and so on. More generally, if the "arrow" is placed over the number of any odd line of table 3, one sequence of mines will be generated; if over the number of any even line, another sequence will be generated. These two different sequences of mines will generate two different sequences of boards.

4.3. The convergence effect and "mixed" boards

It will be easier to understand the convergence effect by analyzing a super-smaller-scale example. A good PRNG has a huge period, and is capable of generating an enormous amount of uniquely different numbers – even though these are not really random, and are cycled.

However, it is possible to imagine a very poor PRNG, with a period of only 40 numbers. This PRNG is able to generate only 40 unique numbers between 0 and 1. If it is called for the 41st time, it will return the same number it returned in the first time, entering a cycle of only 40 numbers.

If this poor PRNG is put to generate boards with 8x8 squares and 5 mines, it would be necessary to calculate $1+\text{int}(8\cdot x)$ for each random number x. This would convert the 40 numbers into 40 coordinates. After this conversion, it is possible to suppose that the 40 coordinates that the PRNG is able to generate are:

6–3–8–6–3–2–4–5–7–2–
3–6–4–1–1–8–7–2–8–3–
4–1–7–2–8–5–6–7–3–4–
7–1–5–1–5–8–4–2–6–5–

The last "–" sign in the end of the sequence is to indicate that it starts again by returning to the first number. It is important to notice that this sequence is well-distributed, since there are exactly five occurrences of each number from 1 to 8.

Now, supposing that the "arrow" starts at the first number of the period, it would be necessary to take the first 10 numbers to place the 5 mines of the board. After generating the first board, the "arrow" would be at the 11th number, waiting for the next call for a random number. Similarly, generating another board would require the sequence from the 11th to the 20th random number to be used. And generating a third and a fourth board would use the remaining numbers. These four boards are shown in figure 6.

Notice that if a new board is generated after the fourth board, the first one would appear again. So, the four boards of figure 6 are in an infinite cycle.

Now, what would happen if the initialization of the PRNG placed the "arrow" over the 7th number of the sequence? The same process would be used, and the new sequence of boards is displayed in figure 7. It is important to mention that when the random numbers place a mine over a square that already has a mine, it is simply skipped.

The interesting thing if that the board generated on figure 7 (f) is equal to the board on figure 6 (c). So, the next one would be equal to the one on figure 6 (d), and the cycle depicted in figure 6 would restart. This means

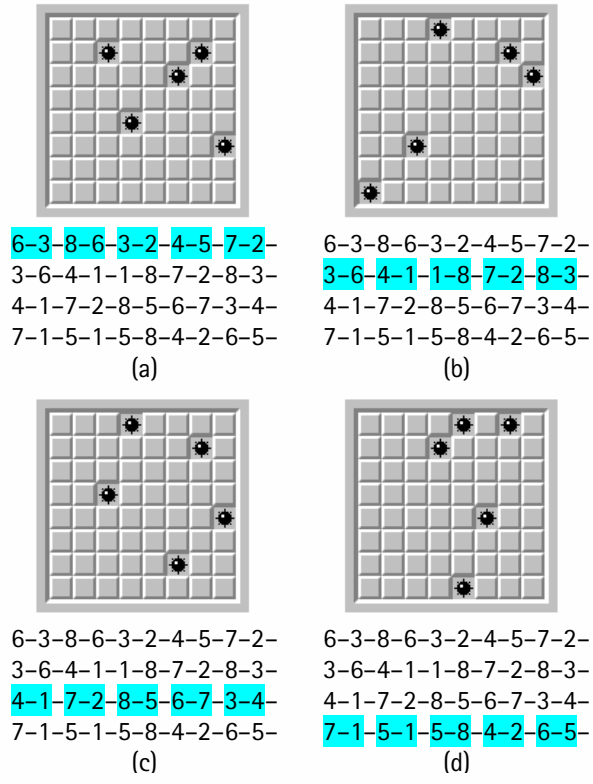
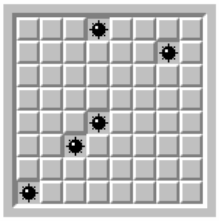


Figure 6. A sequence of 40 coordinates that leads to a cycle of 4 custom beginner boards with 5 mines.

that figure 7 shows a sequence of boards that *converge* towards the cycle of figure 6.

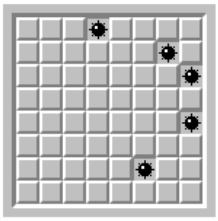
Also, the boards on figures 7 (a) to (e) are mixtures of the boards on figure 6. The board on figure 7 (a) is composed of 2 mines taken from the first board of the cycle and 3 mines taken from the second. The sequence of figure 7 starts to converge, and at a given point, on figure 7 (c), 1 and 4 mines are taken from two consecutive boards of the cycle. Finally, on figure 7 (f), all mines of a board of the cycle are taken to form a single board, and the convergence is complete.

Similarly, it is possible to start generating boards with the "arrow" in any of the 40 coordinates generated by this poor PRNG. Doing it with the "arrow" initially on an odd position will always converge to the cycle of figure 6. However, doing it with the "arrow" initially on an even position will converge to a second cycle, different from the one on figure 6, but also made of 4 boards. And, of course, in the same way, this second cycle can also be achieved through convergence.



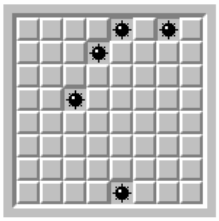
6-3-8-6-3-2-4-5-7-2-
3-6-4-1-1-8-7-2-8-3-
4-1-7-2-8-5-6-7-3-4-
7-1-5-1-5-8-4-2-6-5-

(a)



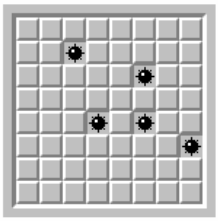
6-3-8-6-3-2-4-5-7-2-
3-6-4-1-1-8-7-2-8-3-
4-1-7-2-8-5-6-7-3-4-
7-1-5-1-5-8-4-2-6-5-

(b)



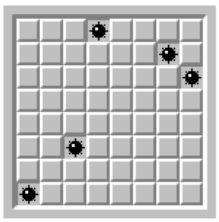
6-3-8-6-3-2-4-5-7-2-
3-6-4-1-1-8-7-2-8-3-
4-1-7-2-8-5-6-7-3-4-
7-1-5-1-5-8-4-2-6-5-

(c)



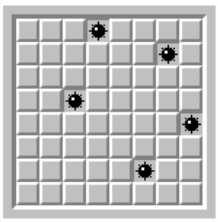
6-3-8-6-3-2-4-5-7-2-
3-6-4-1-1-8-7-2-8-3-
4-1-7-2-8-5-6-7-3-4-
7-1-5-1-5-8-4-2-6-5-

(d)



6-3-8-6-3-2-4-5-7-2-
3-6-4-1-1-8-7-2-8-3-
4-1-7-2-8-5-6-7-3-4-
7-1-5-1-5-8-4-2-6-5-

(e)



6-3-8-6-3-2-4-5-7-2-
3-6-4-1-1-8-7-2-8-3-
4-1-7-2-8-5-6-7-3-4-
7-1-5-1-5-8-4-2-6-5-

(f)

Figure 7. A sequence of boards that can be generated with these 40 coordinates, converging to the cycle.

In this example, the largest sequence of boards that converge to the "odd cycle" is 11 boards, if the "arrow" is initially placed in the 23rd or 25th position. The largest sequence of boards that converge to the "even cycle" is 12 boards, if the "arrow" is initially placed in the 36th position.

It is possible to use these same 40 numbers to generate boards with different number of mines. For example, if used to generate boards with 10 mines, this PRNG would generate two cycles of 9 boards, one for "odd" and one for "even" initial positions, and also with the convergence effect.

Now it is interesting to forget this super-smaller-scale example, and consider what happens with real dimensions. Instead of using only 10 random numbers to place mines on a board, a real beginner board requires at least 20, and a real intermediate board requires at least 80 random numbers. Also, the PRNG of Microsoft Minesweeper is obviously not capable of generating only 40 unique numbers, but probably several millions. Finally, skipping mines, as was the case twice on figure 7, becomes enormously more frequent. It is not difficult to realize that the skipping of mines explains the difference in the lengths of both cycles.

4.4. Possible correction for the conversion of random numbers into coordinates

Up to now, the division of an intermediate board into four quadrants to form a corresponding beginner board is not explained.

Assuming that this effect was not made on purpose by the author of the game, then it must appear naturally. In other words, the same way the game handles the mines should be applied to boards with both beginner and intermediate sizes, and *still* result in the "quadrant superposition" effect.

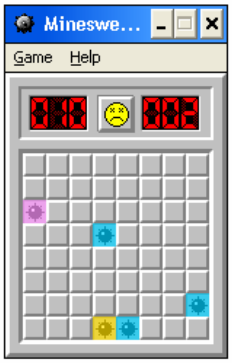
A hypothesis would be to slightly modify the formula $1+\text{int}(D \cdot x)$ to convert random numbers into coordinates, where D is the width or the height of the board, and x is a random number given by the PRNG.

One possible formula would be $1+\text{int}(N \cdot x) \bmod D$, where D and x are already explained and N is any number sufficiently greater than D. Actually, the greater, the better. The word "mod" stands for the *rest of integer division* operator. For example, $7 \bmod 3$ equals 1, because the integer division of 7 by 3 gives 2 as result and 1 as rest.

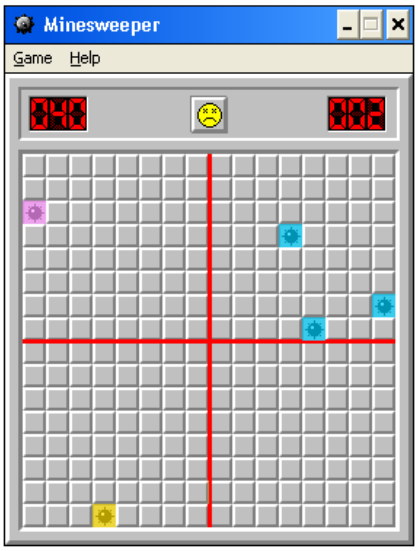
Table 4 gives the conversions of the same random numbers on the first column of table 3 to beginner and intermediate coordinates (with $D = 8$ and $D = 16$, respectively). Also, as N can be any number, it was arbitrarily chosen as 10000.

Table 4. Conversion of random numbers into coordinates using a corrected formula.

x	$1 + \text{int}(N \cdot x) \bmod 8$	$1 + \text{int}(N \cdot x) \bmod 16$
0.705548	8	16
0.533424	7	7
0.579519	4	4
0.289563	8	16
0.301948	4	12
0.774740	4	4
0.014018	5	13
0.760724	8	8
0.814490	1	1
0.709038	3	3
(...)	(...)	(...)



(a)



(b)

Figure 8. The simulated boards that reproduce the "quadrant superposition" effect.

Grouping these coordinates at every two, it is possible to build a beginner board and an intermediate board, shown in figure 8 (a) and (b). As indicated by the colors, this "corrected" version of the formula seems to produce the same effect observed on Microsoft Minesweeper.

This is not absolutely conclusive. It should be observed that there is no way to prove that this *proposed* modified formula is really being used. However, the fact that the *exact* effect observed on the actual game can be reproduced is evidence that at least the operator *mod* seems to be used.

5. Conclusion

Of course there are still questions without answer, even worthy of much more research, like, for example:

- What is the *real* reason for the *board shift* effect?
- How does the PRNG of Microsoft Minesweeper work *exactly*? What are the minimum and maximum values for the numbers returned? What is its period?
- How *exactly* is the formula that converts random numbers into coordinates?

However, much can be taken from this research. Also, many tests were done since 2004, when this research was done, and all of them seem to validate all hypothesizes presented here.

REFERENCES

[1] Barry, D. (2004): *David Barry*. (personal profile) http://www.planet-minesweeper.com/active-ranking/David_profile.htm

[2] Bright, C. (2004): *Minesweeper Bugs: 1 second theory*. http://www.geocities.com/brightprojects/minesweeper/1sectheory.html

[3] McGinley, M. (2002): *1 Second Theory*. http://inthalloffame.8m.com/1sectheory.html

[4] McGinley, M. (2002?): *Same Boards*. http://inthalloffame.8m.com/sameboards.html

[5] McGinley, M. (2004): *Matt McGinley*. (personal profile) http://www.planet-minesweeper.com/active-ranking/Matt_profile.htm

[6] Moore, D. (2002): *The board cycle and shifted boards*. http://www.metanoodle.com/minesweeper/DBcycle.html

[7] *Other internet links:* http://www.planet-minesweeper.com/ http://www.metanoodle.com/minesweeper/ http://www.dotti.at/ms/ http://en.wikipedia.org/wiki/Minesweeper_%28computer_game%29

* The author is also author of *Minesweeper Clone*, a known version of Minesweeper, acceptable for records worldwide, and, of course, free of board cycles.