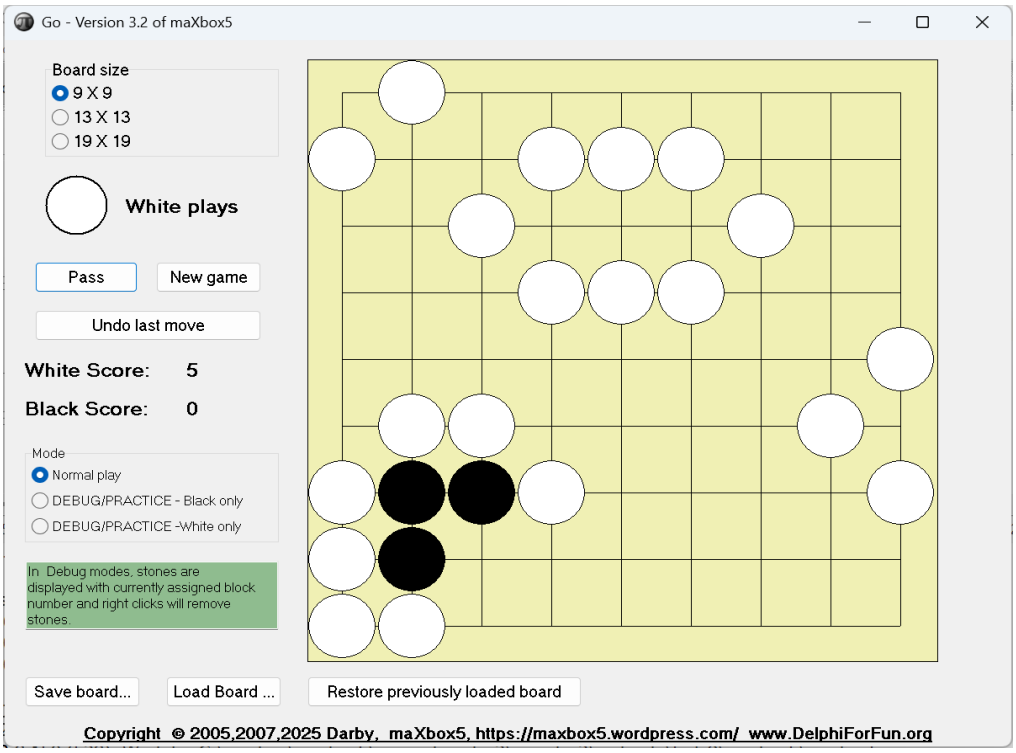
# Go Game 囲碁

## maXbox Starter 143 – Get a Board.

"Natura abhorret vacuum[1] – Spinoza.

Source: 1390_Sphenic_Numbers2TIO_12_py_uc.txt
**1406_U_Go3_1form2.pas**

Go originated in China, then spread to Japan and Korea—and has also been popular in Europe since the 20th century. However, of the estimated 27 million Go players worldwide, 22 million live in Asia. It's high time to change this ratio a little!




1406_1_GoScreenshot2025-06-162815.png

Go is one of those simple but complex board games. Two opponents take turns placing black and white stones on a square board. If you surround a group of opponents stones with yours, the stones are removed and you get credited with a point for each stone captured.

If you are not familiar with the game just do a web search on "Go game rules" or "Go game tutorial" and you'll find plenty of sites.

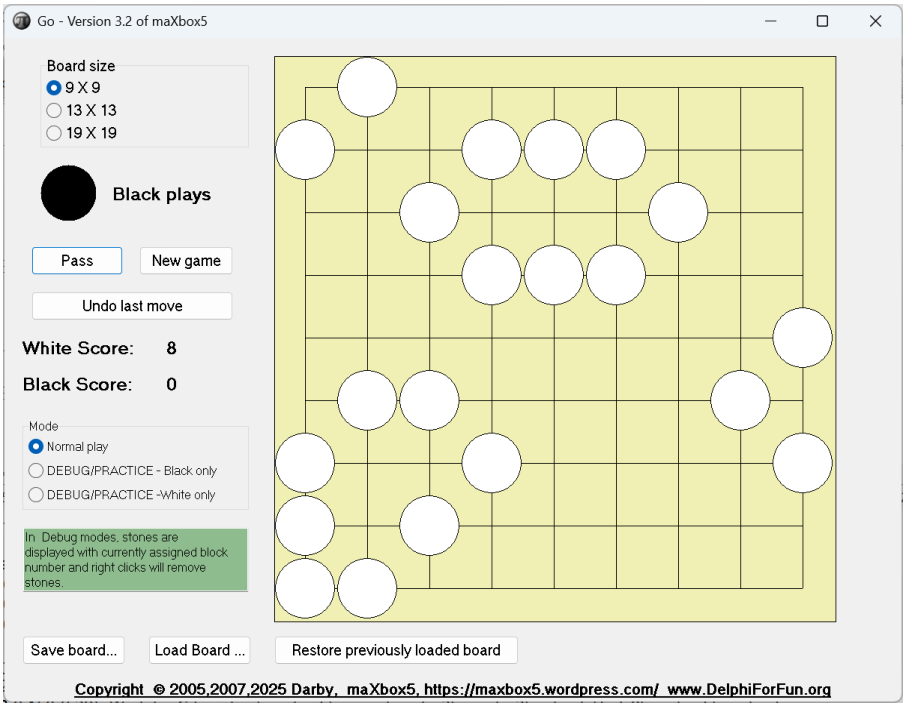
Advocates describe it as "the greatest board game ever" and "more difficult to master than Chess". I'm not, and never will be, a Go expert. In fact, I don't completely understand the intricacies of the rules of play. A student who was looking for an implementation as part of a senior

1 Nature recoils from emptiness -

project wrote to me, and I got caught up in how to code it. I decided I had better post this first effort for a couple of reasons.

The professor will certainly do an online search after he submits his project, so he can determine how much of the code is mine and how much is the student's. Secondly, I'm hoping that some knowledgeable Go players out there will be willing to tell me what important features are missing and what is plain wrong in this version.

Three board sizes are available: 9x9, 13x13, and the official 19x19.



1406_2_GoScreenshot2025-06-02162846.png

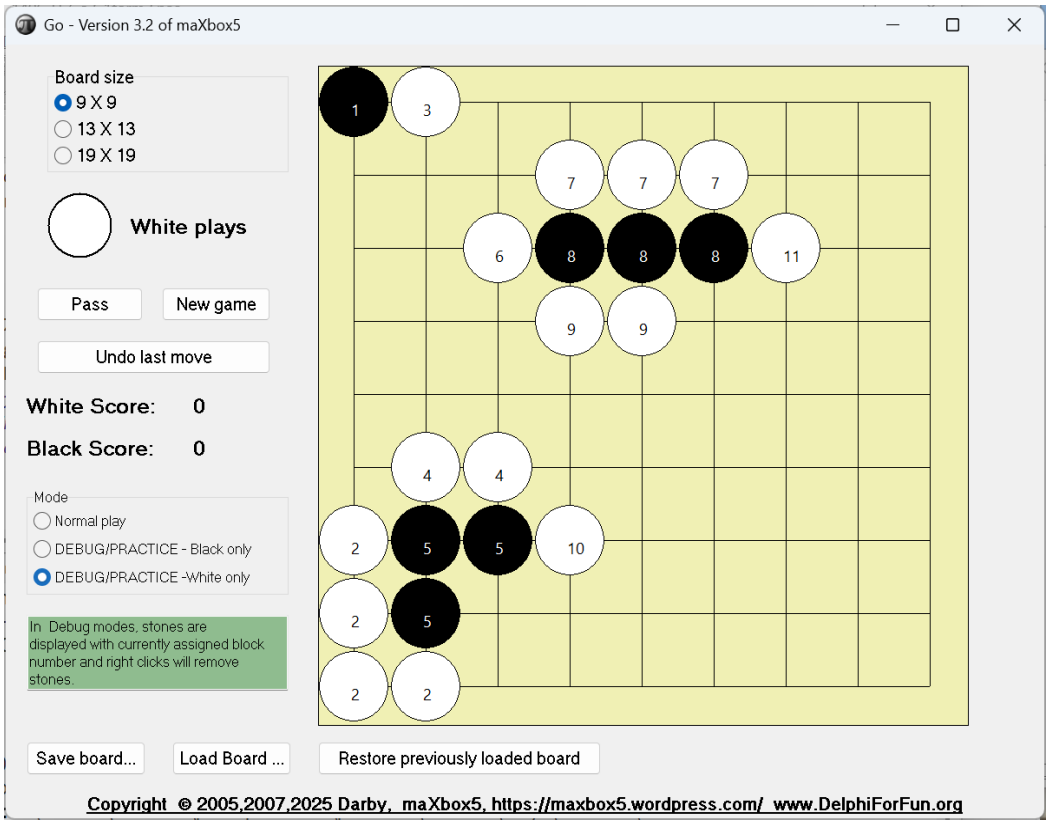
For scoring, I just credit the taker with captured stones. I've seen other information about "live" and "dead" empty spaces. As I understand it, dead spaces could be filled by one player but would be suicidal (therefore forbidden) for the other. I think that "dead" spaces count for the player who could play there, but I'm not sure about that or whether it's worth the effort to classify them.

```
function Sameboard({const} b1,b2:TGoBoard):boolean;
{tests two boards for any difference }
var i,j: integer;
begin
  result:=true;
  for i:=0 to vlines-1 do begin
    for j:= 0 to hlines-1 do begin
      if b1[i][j].occupiedby<>b2[i][j].occupiedby then begin
        result:=false;
        break;
      end;
    end;
    if not result then break;
  end;
end;
```

The Go board consists of a grid of horizontal and vertical lines. The most popular is a 19x19 board with 361 intersection points. Smaller boards (9x9 to 15x15) are also common for practice games and quick games.

Go Browser Game COSUMI - It's Free!

Identifying groups of stones and determining when a group was surrounded by stones of a the opposite color was harder to code than expected. The game is played by placing stones on the intersections of gridlines. A group is a set of one or more stones of the same color connected by horizontal and vertical grid lines.
A group is captured by surrounding it with stones of the opposite color. "Surrounding" means that each horizontal and vertical grid line extending from the edges of a group is filled with a stone of the opposite color.

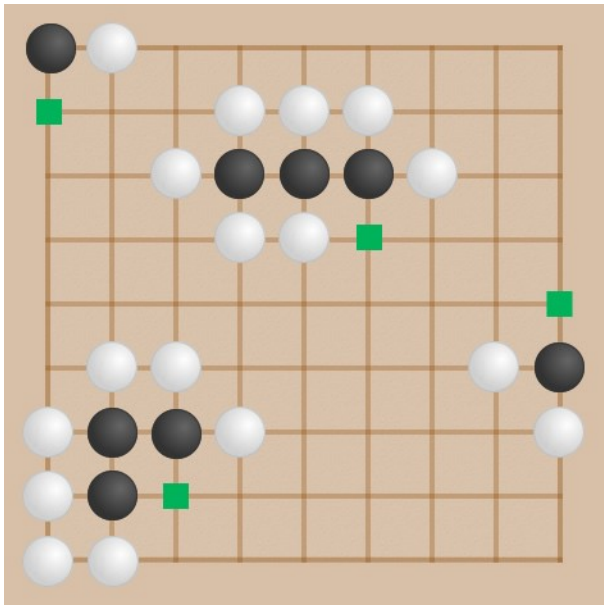


1406_3_GoScreenshot2025-06-02165757.png

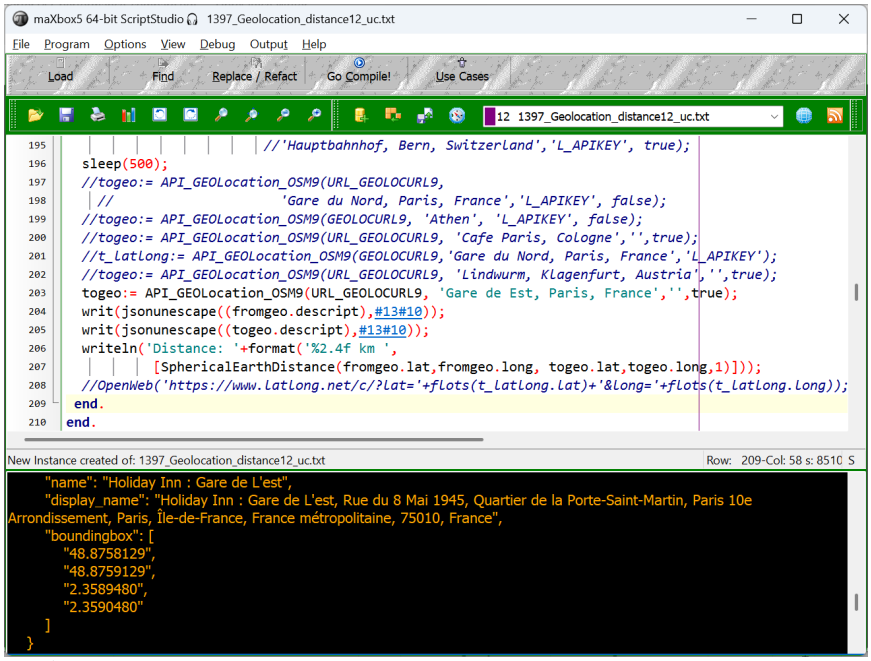
After a few false starts, my solution was to implement routines that clear all blocks and identify them from scratch after each stone is placed. **Findblocks()** initiates the process and the **FindBlocksFrom()** function recursively completes the search for each block. Function **BlockCount** is called to count the blocks in each group and also returns the number of open edges. Everything could have been done more efficiently, but there are only a few hundred stones at most and only a few hundred moves, so the time to start from scratch for each stone played is not significant.

By occupying all the liberties of an opponent's piece or chain, you can capture or capture pieces. If you succeed, the piece or the entire chain

is removed from the board and stored. Each captured piece becomes a prisoner and counts as one point in the final score.



In September, Brian Kleiner represented Switzerland in the 17th Korea Prime Minister Cup (KPMC), in Gwangju. There were players from 53 countries present, with 6 rounds over three days of play. The organisation from the hosts was excellent from start to finish, the late summer weather was ideal, and a good spirit reigned throughout.
The long bus rides were rewarded with attractive views of the countryside, explanations from the guides, and several memorable visits – first to a beautiful bamboo forest park, and then to a Baduk (Go) factory, museum, and shop for souvenirs.




MaxMatrix Time/Space:

The multiplication of past x future is a vector with the function:= known = f(changeable) [y=f(x)] as distance over time, so **distance** is a function of time: **d=f(t)**.

| Past | Present | | known |
|---|---|---|---|
| Big Bang | Future | | unknown |

unchangeable changeable **Time**

## Conclusion

As the game draws to a close, every move must be carefully considered. There are only a few free spaces left, and you don't just have to consider what will score you the most points. The more important question is: What will harm your opponent the most? If a move only scores you one point but causes significant damage to your opponent, you should do it. Even if another move could score you more points.
The BlockCount functions were rolled into FindBlock and a new TBlock object keeps stone count, open edge count, and detail stone location information for each block. The Blocks array keeps track of all blocks on the board.

**Script:**
https://sourceforge.net/projects/maxbox5/files/examples/1406_U_Go3_1form2.pas/download

**References:**
Game of Go
Home - Geocoding API Documentation
Schweizer Go Verband
Go Spielregeln - Einfach erklärt


Doc and Tool: maXbox5 - Manage Files at SourceForge.net

**Max Kleiner 05/06/2025**

maXbox4exe
Digitally signed by maXbox4exe
Date: 2025.06.05 17:21:24 +02'00'