



MMbg – Backgammon for MMBasic

Manual · Version 1.10.34 · 25 October 2026

© 2026 Matthias Hollatz · Free software under the GNU GPL v3 or later

One package, three computers. The program `mmbg_pc3.bas` (version **1.10.34**) runs on the **Pico Computer 3 (PC3)** with firmware **PicoMitePC3 V7.00.00 RC2 or later** and on the **Colour Maximite 2** with firmware **MMBasic 7.00.00b3 or later** (chapter 15). For the **PicoCalc** there is the separate edition **MMbg PiCa** (`mmbg_pica.bas`, version **1.10.33**, Part 2). All of them play the same game with the same engines and the same files.

Memory figures (marked with ~) were measured with the V7.00.00 release candidates and may change a little with later firmware.

Contents

Part 1 - MMbg PC3, Colour Maximite 2 and the common basics

1. What is MMbg?
2. Requirements
3. Installation
4. The screen and the terminal
5. Playing a match
6. Undo, saving and resuming
7. Recording, analysis and screenshots
8. Engines and playing strength
9. The neural nets
10. Playing against gnubg over WLAN
11. Long test runs
12. Options
13. Command summary
14. Files
15. MMbg on the Colour Maximite 2
16. Notes and limits
17. A flight through the source code

Part 2 - MMbg PiCa (PicoCalc), version 1.10.33

- P1. MMbg PiCa at a glance
- P2. Requirements and firmware set-up
- P3. Installation and start
- P4. Screen, terminal and controls
- P5. Differences from the PC3
- P6. Nets and memory on the PicoCalc
- P7. The 16-bit nets: rounding and playing strength
- P8. Files
- P9. Notes and limits

Licence and thanks

New since the last release (1.10.33)

- **Colour Maximite 2:** `mmbg_pc3.bas` now also runs on the CMM2 (chapter 15).
- **PS/2 mouse:** besides a USB mouse, a PS/2 mouse is found as well (CMM2, and PicoMite with `OPTION MOUSE`).
- **Stopping the computer:** the Enter key no longer stops a running computer by accident.
- **MMbg PiCa** stays unchanged at version 1.10.33.

Part 1 - MMbg PC3, Colour Maximite 2 and the common basics

1. What is MMbg?

MMbg is a complete backgammon program written in MMBasic. It plays **matches** from 1 to 15 points, with doubling cube, Crawford rule and everything that goes with it.

- **Playing strength:** With the *big* engine MMbg plays at about the level of GNU Backgammon "Expert" (0-ply). Against gnubg "World class" (2-ply) it wins about **47 % of 5-point matches** (577 of 1236 measured matches).
- **Graphics and terminal:** The board is shown in colour (PC3: HDMI 640 × 480, CMM2: VGA 640 × 480, PicoCalc: LCD 320 × 320) with smoothly animated moves. At the same time a serial terminal shows an ASCII board with move list, evaluations and log.
- **Controls:** Play with the keyboard or, on the PC3 and CMM2, with the mouse. Sounds for dice and moves are available on request.
- **Comfort:** Multi-level undo, automatic saving and resuming. Matches are recorded as `.mat` files for analysis in gnubg. Positions can be exchanged via **GNUbg IDs**.
- **For tinkerers:** gnubg on a PC can act as opponent over WLAN. On top come overnight test runs, duplicate matches and detailed logs.

2. Requirements

What	Requirement
Computer	Pico Computer 3 (PC3): RP2350B with PSRAM, HDMI, USB (Colour Maximite 2: chapter 15, PicoCalc: Part 2)
Firmware	PicoMitePC3 V7.00.00 RC2 or later. Check with <code>?MM.VER</code> : at least 7.000052
Display	HDMI, MODE 3 (640 × 480, 16 colours)
Keyboard	USB keyboard on the PC3 and/or serial terminal
Terminal (recommended)	VT100 capable (e.g. Tera Term, PuTTY), 80 × 40 characters , colours on
Mouse (optional)	USB mouse; a PS/2 mouse is found as well (chapter 15)
Sound (optional)	<code>OPTION AUDIO</code> set up (I ² S on the PC3)
WLAN (optional)	only for gnubg as opponent (chapter 10)
SD card (recommended)	drive B: for the program folder, logs and <code>.mat</code> files

On the PC3 the big nets are kept as floating-point numbers in PSRAM (~512 KB), the animation uses another ~150 KB for a framebuffer.

3. Installation

3.1 Files

All files belong **in one folder** together with the program (chapter 3.2). The package contains the files for all editions; each edition simply ignores the files it does not need (table in chapter 14 and in Part 2, chapter P8).

File	Size	Purpose	If missing ...
<code>mmbg_pc3.bas</code>	~190 KB	the program (PC3 and CMM2)	-
<code>bgnet345.dat</code>	~96 KB	small nets 3, 4, 5 (gnubg)	engines 0-2 only
<code>bgnet02.dat</code>	text	output weights of the big nets	no engines 4 and 5
<code>bgnet0.bin</code> , <code>bgnet2.bin</code>	256 000 bytes each	big nets contact and crashed	no engines 4 and 5
<code>bgesc.bin</code>	64 KB	escape tables (gnubg)	no engines 4 and 5
<code>bgbo15.dat</code>	~1.6 MB	bear-off database up to 15 checkers	race with net 5 only
<code>bgnet35.dat</code> , <code>bgbo10.dat</code>		smaller fallbacks, used only if <code>bgnet345.dat</code> or <code>bgbo15.dat</code> is missing	-

MMbg creates the settings and saved-game files (`mmbg_pc3.ini` , `bgext.ini` , `mmbg_pc3.sav`) itself in the program folder, and the folder `log/` below it.

3.2 Everything in one folder on the SD card

MMbg looks for all its files **relative to the program folder**: nets, database, settings, saved match and the log folder below it. Nothing is tied to a fixed drive. The folder can therefore **have any name and lie on any drive at any depth** - which is also why the same package runs on every computer, whether the SD card is called `B:` there (PC3, PicoCalc) or `A:` (Colour Maximite 2).

Recommended:

Folder	Contents
<code>/mmbg</code>	program, nets, database, ini files, saved match
<code>/mmbg/log</code>	logs, <code>.mat</code> files and screenshots (created by MMbg itself)

To set this up, unpack the package and copy the folder `mmbg` to the root of the SD card. Alternatively transfer each file with `XMODEM RECEIVE "name"` into a folder you have created with `MKDIR "B:/mmbg"` .

3.3 Start

Always start **from the program folder**, so that MMbg finds its files there:

```
CHDIR "B:/mmbg"  
LOAD "mmbg_pc3.bas", C  
RUN
```

With `, C` MMBasic leaves out the comments and so saves program memory; this is recommended. `mmbg_pc3.bas` has only short comments and also loads without `, C`. On the Colour Maximite 2 the SD card is `A:`, so use `CHDIR "A:/mmbg"`.

At start-up the logo, the licence and the loading lines for nets, database and WLAN appear. MMbg then waits on the start screen for **N** (new match).

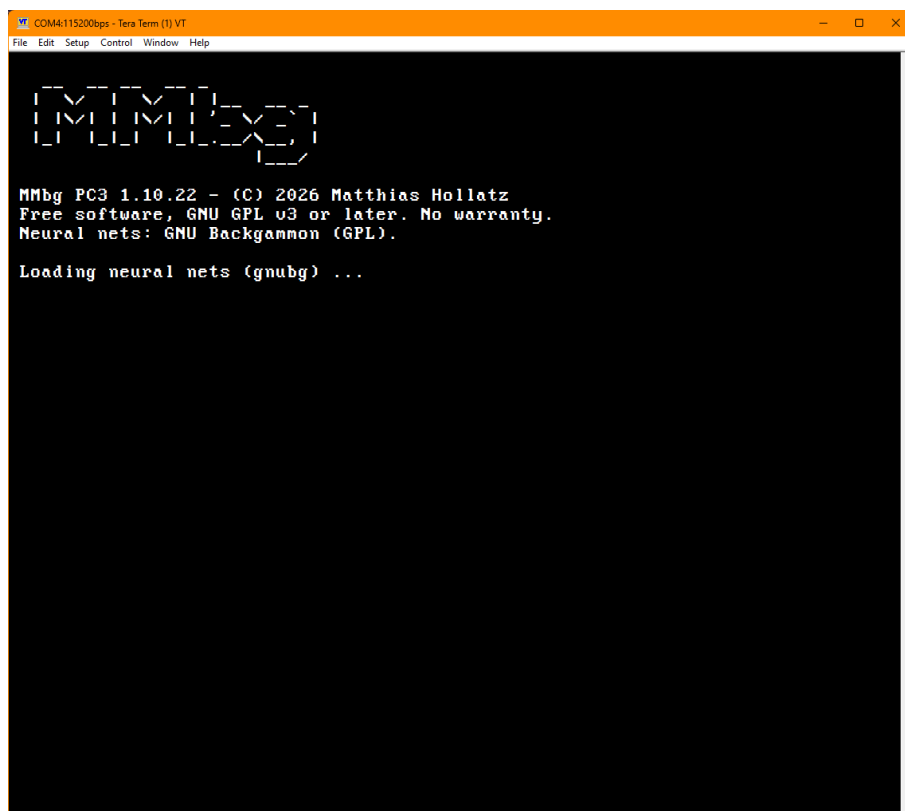


Figure 1: Start in the terminal – logo, licence, the nets are being loaded.

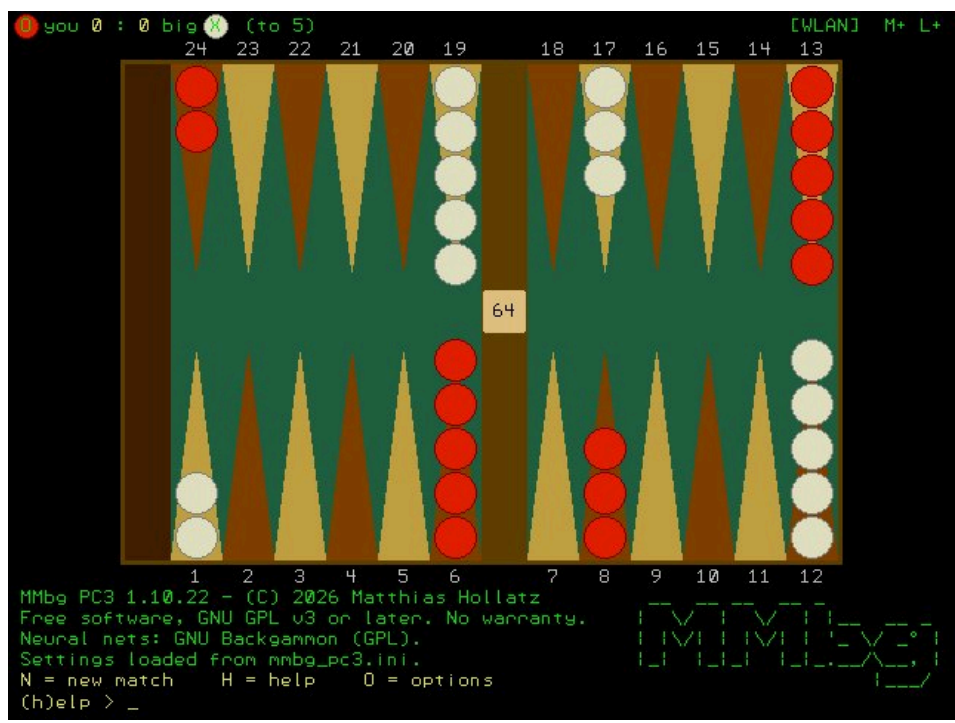


Figure 2: Start screen on HDMI. Bottom left version, licence and messages, on the right the logo, below the most important keys.

Error messages with a line number refer to the program without comment lines when it was loaded with `, C`.

3.4 The log folder

Logs, `.mat` files and screenshots go to the **log folder**. By default this is `log/` below the program folder (`B:/mmbg/log`), and MMbg creates it at start-up if it does not exist. Nothing needs to be set up.

To use another folder, open the options page (`O → 7`), press `L` and enter the folder:

Input	Log folder
<code>log</code>	<code>log/</code> below the program folder (default)
<code>B:/games/bglog</code>	any existing folder, absolute
<code>-</code>	the program folder itself

Save with `S`. In `bgext.ini` the setting is the line `logdir=`: empty means `log/`, `.` means the program folder.

4. The screen and the terminal

4.1 HDMI screen

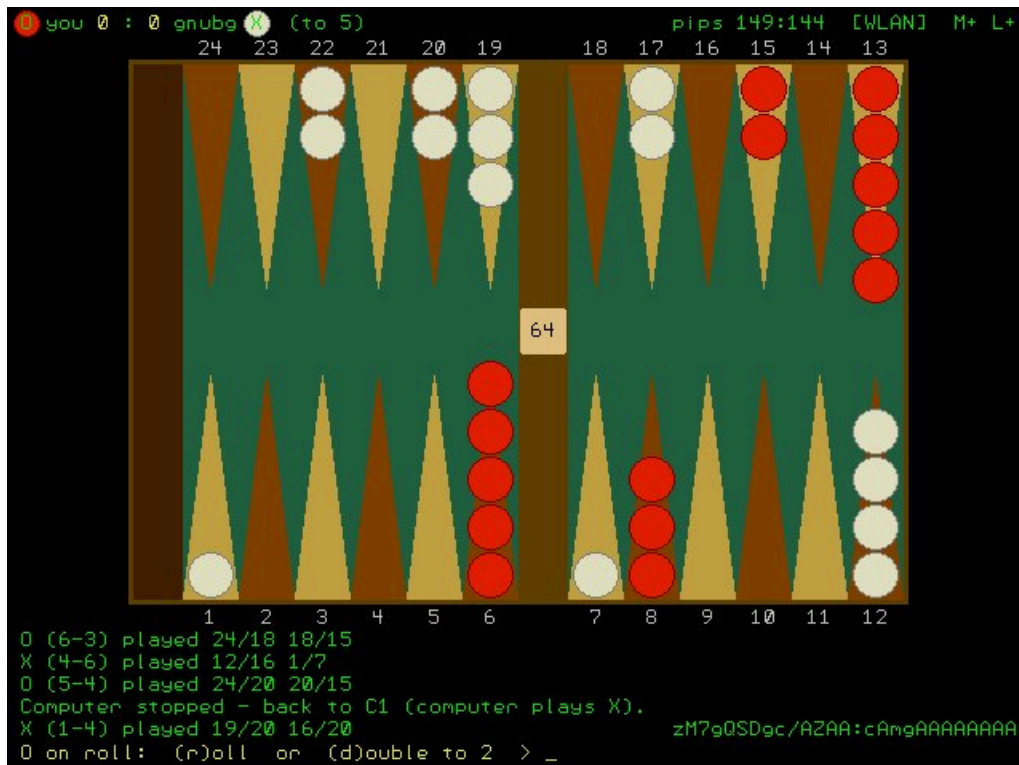


Figure 3: A game in progress. Title with players, score, pips and switches, below it the board with the cube in the bar, the five-line log, the GnuBg ID bottom right and the prompt at the very bottom (here the question before the roll).

- **Title line**

- O and X appear as small checkers in the player's colour, next to them the player: `you` for a human, otherwise the name of the engine.
- The **score is yellow**, followed by the match length, e.g. `(to 5)`.
- `M 4:2` counts the matches won in this session. It appears in computer runs and as soon as a match is over.
- In the middle: `Crawford` or `Post-Crawford`.
- On the right: the pips (option), `[WLAN]`, `M±` for `.mat` recording and `L±` for the log.

- **Board:** 24 points, bar, tray. The dice always lie **to the right of the player** as seen from his side, so O's dice on the right and X's dice on the left.

- The cube sits in the middle of the bar or with its owner. In the Crawford game it is grey with "64"; doubling is not allowed then.
- An **offered double** appears as a single cube on the board, on the side of the player who has to decide.

- **Log:** The last five messages, moves and hints are shown above the prompt.

- **GnuBg ID:** The ID of the current position is shown in green at the right of the bottom log line (chapter 7.4).

- **Colours:** Text is green; keys, input and changing values are yellow.

On the Colour Maximite 2 the VGA screen looks exactly the same.

- The opening roll decides who starts.

5.2 Moving

Input	Meaning
13/7 8/7	moves in gnuBG notation, several partial moves separated by spaces
bar/22 , 6/off	from the bar, bearing off (short: <code>b</code> , <code>o</code> ; <code>-</code> instead of <code>/</code> also works)
13 24	start points only: the bigger die first
b 6	from the bar and from the 6
#3	move no. 3 of the move list
E	end the turn if no die can be played any more
U	undo (chapter 6)

A partial move is only valid if the largest possible number of dice can still be played afterwards. If a player cannot move at all, MMbg passes automatically and shows the dice briefly.

With the mouse (PC3 and CMM2):

Action	Meaning
Left click on an own checker (or the bar)	move it with the bigger die; if that is not possible, with the smaller
Right click on an own checker (or the bar)	move it with the smaller die; if that is not possible, with the bigger
Click on your own dice	undo
Before the roll: click on the cube	double
Before the roll: click on the board	roll
Click on the offered cube	take. Pass only with the key <code>P</code>

Example: a checker is on the bar and the roll is 3-1. A **right click** on the bar enters with the 1, and the 3 is left for another checker.

5.3 Doubling cube

Before every roll MMbg asks the player on roll, if he may double: `(r)oll` or `(d)ouble` . If the opponent offers a double, the answer is `(t)ake` or `(p)ass` .

- With the option *hint* the log shows the cube recommendation and the match winning chances.
- In the **Crawford game** doubling is not allowed. Post-Crawford follows.
- Match rules apply. Money-game rules like Jacoby or beaver are not available.

5.4 End of game and match

The winner scores one point (single), two (gammon) or three (backgammon), each times the cube value. The next game starts after three seconds, a new match with ENTER.

6. Undo, saving and resuming

6.1 Undo

- **U** during a turn takes back your own partial moves.
- **U** without a moved checker jumps back to **your last turn start**, with the same dice. This also works after the computer has already moved, for example after a mis-click.
- **Another U** goes back one more of your own turns, and so on, up to the start of the game.
- After an undo the computer rolls again, so nobody can react to its dice. Undo only works within the current game.

It is based on the **move history**: for every turn start MMbg keeps the GNUbg ID, i.e. board, score, cube and dice.

6.2 Saving and resuming

- MMbg saves the current match **automatically** to `mmbg_pc3.sav` (PicoCalc: `mmbg_pica.sav`):
 - at every turn start of a human,
 - at the questions roll/double and take/pass,
 - in computer runs C2 and C3 at every turn. C4 does not save.
- **SG** saves immediately, **LG** loads the saved match.
- At **start-up** MMbg asks: *"Saved match found - continue it? (y/n)"*.
- Saved are the whole move history (undo still works after loading), computer mode, engines and the running `.mat` recording.
- The file is deleted at the end of the match.

What is in the file: The file is plain text, one entry per line. The core are the `turn=` lines: for **every turn start** of the game they hold the **GNUbg ID** of the position (board, score, cube, dice), plus the turn number and how many `.mat` entries had been written up to then. When loading, MMbg rebuilds the position from the last ID; the earlier IDs make undo possible after loading.

```
' MMbg PC3 saved match
version=1.10.34
date=25-10-2026 20:26:53
computer=1                computer mode (C1)
engine_o=0                engines 0 and X
engine_x=4
game=7                    game number in this session
turn=4HPwATDgc/ABMA:cAmgAAAAAAAA 0 0    turn start: GNUbg ID, turn, .mat count
turn=4HPwATDgc/ABMA:QqnrABAAGAAA 1 1
...
matfile=log/mmbg_20261025_202011.mat
mgame=3                   game number in the match (.mat)
act=2|52: 13/8 24/22      actions of the game recorded so far
now=4HPwATDgc/ABMA:cAmgAAAAAAAA          position before the roll (at roll/double)
```

Every ID can also be pasted into gnuBG on its own to look at a position from a saved match. The explanations on the right are only in this manual, not in the file.

7. Recording, analysis and screenshots

7.1 Matches as `.mat` for gnu**bg**

With the option **mat** (`M+` in the title) MM**bg** writes every match to its own file:

```
[log folder]mmbg_YYYYMMDD_HHMMSS.mat
```

- The format is the same one gnu**bg** writes with `export match mat`.
- The header is written when the match starts, each game is appended at its end.
- New matches never overwrite old files.
- Moves taken back with undo do not appear in the file.
- A resumed match continues in its old file.
- In gnu**bg** X is player 0 (top), O is player 1 (bottom).

Analysis in gnu**bg**:

1. Transfer the file to the PC, e.g. via SD card or XMODEM. XMODEM pads the end of the file with zero bytes; gnu**bg** does not mind.
2. In gnu**bg**: *File* → *Import* → *.mat* or `import mat file.mat`.
3. *Analyse* → *Analyse match* or `analyse match`, then the statistics with `show statistics match`.

7.2 Log file

Switch on or off with **G** (`L±` in the title). The setting is saved with the other options. The file `bg_YYYYMMDD_HHMMSS.log` in the log folder contains:

- every move with time, dice, rank in the move list and evaluation,
- the computing time, the pips and gnu**bg**'s response time,
- cube decisions with winning chances, and game and match results.

A new file starts at about 1 MB. **GN** starts a new one immediately.

A move line looks like this (in the log it is a single line):

```
2105.3s G12 T 14 X 6-4 big 24/18 13/9 #1/18 +0.123 best +0.123
gen 512ms pip 0 140 X 150 ext 70ms
```

That means: seconds since start, game 12, turn 14, X rolled 6-4, engine big, the move played, its **rank in the move list** (1 of 18 legal moves), its evaluation and the best evaluation, the computing time of the move generator, the pips of both sides and, with gnu**bg**, the response time. There are also lines with `---` (cube, engines, settings, undo), `***` (end of game), `###` (end of match and running total), `!!!` (errors) and, on the PicoCalc, `D` (board dump with key D).

7.3 `.mat` or log - which one for what?

	<code>.mat</code> recording	log file
Purpose	replay and analyse the match	watch the engines , mainly in test runs
Format	standard format, read by gnuBg (and other programs)	own text format for humans and evaluation scripts
Contents	only the game: rolls, moves, cube, results	in addition rank in the move list, evaluations, computing times, pips, gnuBg response times, errors, settings
Size	one small file per match	one running file, a new one from approx. 1 MB
Typical use	your own games against the computer	overnight long runs (chapter 11)

Both can be on at the same time. Players who only want to analyse their games in gnuBg need only `.mat`.

7.4 GNUbg ID

The ID describes a position completely: board, match length, score, cube, Crawford, player on roll and dice. An example:

```
4HPwATDgc/ABMA:cAmgAAAAAAAA
```

- **MMbg → gnuBg**: The ID is shown at the bottom right and on the dump page (key **D**). Copy it in the terminal and paste it into the *GNUbg ID* field in gnuBg, or type `set gnuBgid ...`. This way any position can be examined in gnuBg straight away.
- **gnuBg → MMbg**: Simply paste an ID at the prompt, also at roll/double or take/pass. MMbg asks "*Take over the position...? (y/n)*" and sets up the position. Money game, a match longer than 15 points and a cube above 64 are rejected.
- The IDs have been cross-checked with gnuBg 1.08.003. An offered double is not part of the ID, because gnuBg drops it when reading.

7.5 Hint option

With **hint** (option 3) MMbg writes the best move to the log when a human is on turn, and the recommendation at the cube questions. Good for learning.

7.6 Board dump (D)

D shows the position as text: checkers per point, bar, borne-off checkers, the rolled dice, the dice still to play and the **GNUbg ID**. This is the easiest way to note or report a position, for example together with a bug report. On the PicoCalc the board and the ID are also written to the log (lines starting with `D`).

7.7 Screenshot (Ctrl+P, PC3)

At any input **Ctrl+P** saves the whole HDMI screen as a BMP file (24 bit, ~900 KB) to the log folder:

```
[log folder]mmbg_YYYYMMDD_HHMMSS.bmp
```

The log reports the file name. The pictures in this manual were made this way. (The image viewer in the PC3's file manager shows the colours wrongly; on the PC they are correct.)

8. Engines and playing strength

8.1 The engines

No.	Name	How it works	Strength (rough)
0	base	evaluation formula: pips, blots, points, bar	beginner
1	book	base plus opening book	beginner+
2	tuned	weights tuned in the lab	casual player
3	net	small gnubg nets (0-ply)	good club player
4	big	big gnubg nets, pre-filter with the small nets	≈ gnubg "Expert" (0-ply)
5	deep	big with 1 roll look-ahead (1-ply)	stronger than big, approx. 18 times slower
E	gnubg	gnubg on the PC over WLAN	as strong as gnubg is set

Choose with **A45**: engine for O is 4, for X is 5. For each side one character from 0 to 5 or E.

Measurements for big:

Measurement	Result
matches against gnubg 2-ply (to 5), PC3	577 of 1236 = 46.7 %
matches against gnubg 2-ply (to 5), PicoCalc (16 bit)	199 of 450 = 44.2 % ± 4.6 % (Part 2, P7)
error per move, measured against gnubg 2-ply	big 2.6 mEq, gnubg 0-ply 2.4 mEq
best move found	big 82 %, gnubg 0-ply 83 %

- About 90 % of the gap to gnubg 2-ply comes from move choice, about 10 % from the cube.
- The computing time averages about 0.5 to 0.6 s per move on all computers, about 10 s with *deep*.

8.2 Cube model

For the cube MMbg calculates like gnubg at 0-ply: from the win, gammon and backgammon chances of the net, the **Kazaross-XG2 match equity table** gives the chances for no double, double/take and double/pass.

8.3 Computer modes

Command	Mode
C0	two humans
C1	computer plays X (default)
C2	computer plays both sides (to watch, with pauses)
C3	both, without pauses and without animation
C4	turbo : both, no board drawing, endless matches (test runs)

In C2 to C4 **any key** stops the computer.

9. The neural nets

9.1 How does such a net work? (explained very simply)

At its core a neural net is a **big calculation with many fixed numbers**. It gets a position and returns how good it is. Nothing more – it does not think ahead and knows no rules.

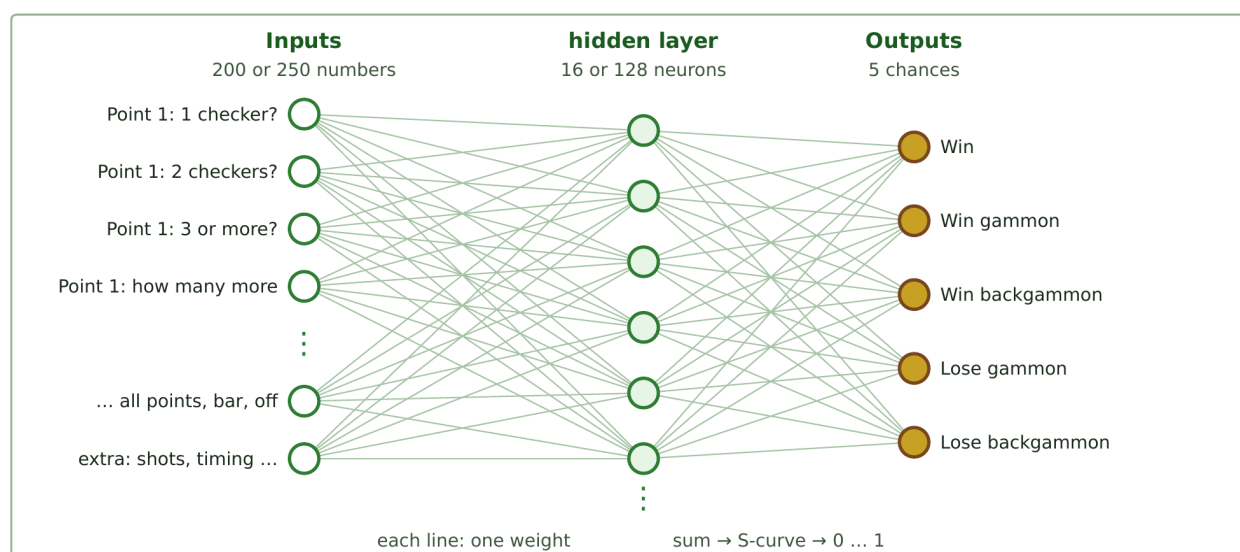


Figure 5: Structure of a gnubg net (shown reduced). On the left the position as numbers, in the middle the hidden layer, on the right the five chances.

1. The position becomes numbers. For each point and each side there are four inputs: is there exactly 1 checker (a blot)? Exactly 2? 3 or more? And how many beyond that? Bar and borne-off checkers come on top. This gives 200 numbers, mostly 0 or 1. The big net gets 50 extra features a human would "see": How often can I be hit? How easily do my back checkers escape? How much time do I have left?

2. The hidden layer. Each neuron in the middle forms a **weighted sum** of all inputs: each input times its own weight, all added up. The sum goes through an **S-curve** that squeezes it to a value between 0 and 1. In this way each neuron becomes a small "detector" for a pattern, such as "good home board" or "exposed blot" – which pattern, the net has learned by itself.

3. The outputs. The five output neurons do the same again with the values from the middle. Out come five chances between 0 and 1: win, win with gammon, win with backgammon, lose with

gammon, lose with backgammon. From these one single value is formed, the **equity**:

$$\text{Equity} = 2 \times \text{win} - 1 + \text{gammon wins} + \text{backgammon wins} - \text{gammon losses} - \text{backgammon losses}$$

4. Where do the weights come from? They are the actual knowledge. The gnubg authors trained them with a very large number of played-out positions: the net estimates, compares with the real result and shifts the weights a tiny bit in the right direction – millions of times. MMbg only calculates with the finished numbers; it does not learn by itself.

5. How does this become a move? The move generator finds all legal moves (often 20 to 50, with doubles sometimes several hundred). For each move the **resulting position** is sent through the net. The move with the best equity is played. Engine *big* first rates all moves with the small net and recalculates only the best 12 with the big one; *deep* additionally looks ahead at all 21 possible rolls of the opponent.

The big net needs about 32 000 multiplications per position ($250 \times 128 + 128 \times 5$). In MMBasic on the PC3 this takes about 19 ms. Since most of the 250 inputs are zero in a real position, MMbg only adds up the rows of the non-zero inputs (about 50).

9.2 The nets in MMbg

MMbg uses the **neural net weights of GNU Backgammon** (gnubg 1.08, GPL). MMbg evaluates the nets itself in MMBasic. The input encoding is the same as in gnubg.

Net	Structure	Used for
3 contact (small)	200 inputs, 16 hidden neurons, 5 outputs	engine net, pre-filter for big
4 crashed (small)	200-16-5	positions with a collapsed home board
5 race (small)	200-8-5	races, when there is no contact any more
0 contact (big)	250 inputs (200 + 50 extra features), 128 neurons, 5 outputs	engines big and deep
2 crashed (big)	250-128-5	engines big and deep

- **The 5 outputs** are: win, win with gammon, win with backgammon, lose with gammon, lose with backgammon.
- **The 50 extra features** of the big net, such as hit chances, escapes, containment and timing, are calculated by MMbg following gnubg.
- **Bear-off database `bgbo15.dat`** : computed by ourselves, it contains no gnubg data. It gives exact values in the endgame: one side, 6 points, up to 15 checkers, 54 264 positions, with gammon shares.
- **On the PicoCalc** the same weights are stored packed (32 or 16 bits per weight) – see Part 2, chapters P6 and P7.

Which nets each engine needs:

Engine	needs
0 to 2	nothing, only the program
3 net	bgnet345.dat
4 big, 5 deep	additionally bgnet02.dat and the big nets: PC3 and CMM2 bgnet0.bin , bgnet2.bin , bgesc.bin ; PicoCalc see P8
all	bgbo15.dat makes the endgame exact

10. Playing against gnubg over WLAN

The PC3 (and the PicoCalc with a Pico 2 W) can use **GNU Backgammon on a PC** in the home network as opponent. MMbg asks the questions, gnubg does the calculating. The Colour Maximite 2 has no WLAN.

10.1 Setting up

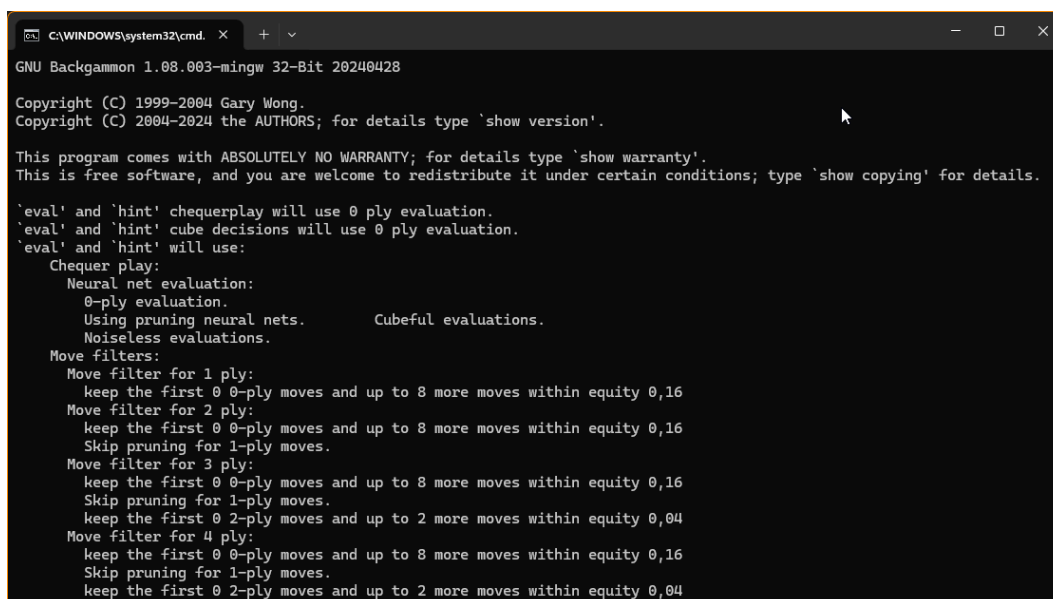
Step 1 - WLAN on the PC3: set up once and restart:

```
OPTION WIFI "MyNetwork", "Password"
```

Step 2 - gnubg on the PC: start it (e.g. gnubg-cli) and enter:

```
external :41000
```

gnubg then waits on port 41000. The playing strength is set in gnubg itself, e.g. 0-ply or 2-ply. The package contains ready-made batch files for Windows in `gnubg_start_PC` (0-, 1- and 2-ply).



```
C:\WINDOWS\system32\cmd. x  +  v
GNU Backgammon 1.08.003-mingw 32-Bit 20240428

Copyright (C) 1999-2004 Gary Wong.
Copyright (C) 2004-2024 the AUTHORS; for details type 'show version'.

This program comes with ABSOLUTELY NO WARRANTY; for details type 'show warranty'.
This is free software, and you are welcome to redistribute it under certain conditions; type 'show copying' for details.

'eval' and 'hint' chequerplay will use 0 ply evaluation.
'eval' and 'hint' cube decisions will use 0 ply evaluation.
'eval' and 'hint' will use:
  Chequer play:
    Neural net evaluation:
      0-ply evaluation.
      Using pruning neural nets.      Cubeful evaluations.
      Noiseless evaluations.
  Move filters:
    Move filter for 1 ply:
      keep the first 0 0-ply moves and up to 8 more moves within equity 0,16
    Move filter for 2 ply:
      keep the first 0 0-ply moves and up to 8 more moves within equity 0,16
      Skip pruning for 1-ply moves.
    Move filter for 3 ply:
      keep the first 0 0-ply moves and up to 8 more moves within equity 0,16
      Skip pruning for 1-ply moves.
      keep the first 0 2-ply moves and up to 2 more moves within equity 0,04
    Move filter for 4 ply:
      keep the first 0 0-ply moves and up to 8 more moves within equity 0,16
      Skip pruning for 1-ply moves.
      keep the first 0 2-ply moves and up to 2 more moves within equity 0,04
```

Figure 6: gnubg 1.08.003 on the PC (command line), set to 0-ply. Next comes `external :41000`.

Step 3 - in MMbg: open the options page (**O** → **7**) and set:

- **H** host: the IP address of the PC

- **P** port: e.g. 41000
- **W** timeout in seconds
- **F** fallback: the engine that steps in if gnubg does not answer (e.g. 4 big)
- **T** tests the connection (version and 10 pings)
- **S** saves everything to the ini files

When `[WLAN]` appears in the title, the WLAN is up.

```
MMbg PC3 1.10.22 - OPTIONS (C) 2026 Matthias Hollat
key + ENTER = change ENTER alone = back to the game
-----
BOARD 1 animation + 4 flip board + home board left
      2 pips in title + 5 swap colours + 0 red, X light
      3 hint - 8 mouse + USB mouse found
      0 sound - dice, moves, hits (PLAY BBC SOUND)
-----
RECORD 9 match file .mat + G log file +
      L folder b:/bg/f/ for log and .mat files, must exist
      now: log bg_20261019_210528.log match -
-----
GNUBG external engine E = gnubg on a PC over WLAN (TCP)
      on the PC start gnubg and type: external :41000
      H host 192.168.1.132 P port 41000
      W timeout 30 s F fallback 4 big
      D debug log - C check gnubg's moves + (slower)
      V level (text for the log) 2-ply
      T test connection (version + 10 x ping)
-----
TEST RUNS U duplicate - each match again, same dice, sides swapped
          R dice seed 0 = from the clock (now 1717527)
-----
STATUS WLAN: connected, IP 192.168.1.150 gnubg at 192.168.1.132:41000
      S save all settings (mmbg_pc3.ini + bgext.ini)
-----

Connected in 7 ms. gnubg 1.08.003-mingw 32-Bit Answer: Interface: 2!RFBF:
0!Engine: 1.01!Software: 1.08.003-mingw 32-Bit Ping 10x: 10 ok, avg 29 ms,
max 102 ms

gnubg level for the log (e.g. 0-ply, 2-ply) > 0-ply_
```

Figure 7: Connection test with `T` successful: gnubg version, 10 pings with response times. MMbg then asks for the text for the log (`V` level).

```
MMbg PC3 1.10.22 - OPTIONS (C) 2026 Matthias Hollat
key + ENTER = change ENTER alone = back to the game
-----
BOARD 1 animation + 4 flip board + home board left
      2 pips in title + 5 swap colours + 0 red, X light
      3 hint - 8 mouse + USB mouse found
      0 sound - dice, moves, hits (PLAY BBC SOUND)
-----
RECORD 9 match file .mat + G log file +
      L folder b:/bg/f/ for log and .mat files, must exist
      now: log bg_20261019_210528.log match -
-----
GNUBG external engine E = gnubg on a PC over WLAN (TCP)
      on the PC start gnubg and type: external :41000
      H host 192.168.1.132 P port 41000
      W timeout 30 s F fallback 4 big
      D debug log - C check gnubg's moves + (slower)
      V level (text for the log) 2-ply
      T test connection (version + 10 x ping)
-----
TEST RUNS U duplicate - each match again, same dice, sides swapped
          R dice seed 0 = from the clock (now 1717527)
-----
STATUS WLAN: connected, IP 192.168.1.150 gnubg at 192.168.1.132:41000
      S save all settings (mmbg_pc3.ini + bgext.ini)
-----

Connect failed: connect 192.168.1.132:41000 failed: No response from
client Is gnubg waiting? (gnubg: external :41000)

connect 192.168.1.132:41000 failed: No response from client

option > _
```

Figure 8: Connection test failed: gnubg is not running or not yet waiting (`external :41000` forgotten), or host/port are wrong.

10.2 Playing

Command	Effect
A4E	O = big, X = gnuBg
C1	the human plays O, gnuBg plays X: a game against gnuBg
C2 or C3	big against gnuBg to watch
C4E	engines O 4, X gnuBg and turbo

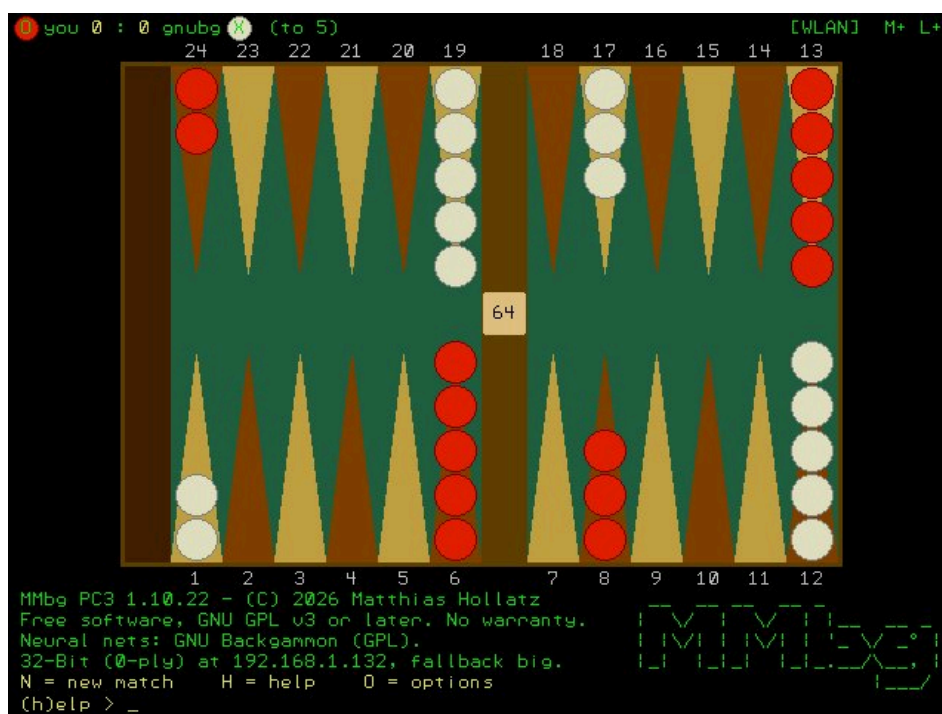


Figure 9: X is gnuBg. The log reports version, level (0-ply), address and the fallback engine (big).

Further switches on the options page:

- **D debug log:** gnuBg's requests and answers go into the log.
- **C check:** MMbg also evaluates gnuBg's moves. Slower, but good for comparisons.
- **V level:** only a text for the log, e.g. "2-ply". MMbg **cannot query** gnuBg's playing strength: it is set in gnuBg itself (e.g. `set evaluation chequerplay evaluation plies 2`). To record in the log which level was played against, enter it here by hand to match and save with S.

gnuBg's move is always played. MMbg looks up gnuBg's answer in its own move list. With some doubles there are several hundred legal moves and the list can be full (PC3 500, PicoCalc 300 moves). If gnuBg's move is missing for this reason, MMbg adds it at the top and plays it; the log notes `--- ext move not in the full list, taken: ...`. Only an answer that cannot be read or played lets the fallback engine decide (`!!!` line in the log).

gnuBg 1.08.003 swaps the score in `external` mode for move and double requests. MMbg compensates for this. The bug has been reported to the gnuBg developers.

11. Long test runs

For strength comparisons the computer plays any number of matches against itself or against gnubg.

1. Choose engines, e.g. `A4E` (big against gnubg) or `A45` (big against deep).
2. Start mode **C4** (turbo): no board drawing, endless matches, one status line with intermediate results. **C3** runs with board but without pauses.
3. With the **log** on, every move goes into the log file in the log folder.
4. Stop with **any key**.

Aids for fair comparisons (options page):

- **U duplicate:** every match is played again with the same dice and the sides swapped. This evens out dice luck.
- **R dice seed:** a fixed seed for reproducible dice. 0 takes the seed from the clock.

Evaluation:

- At the end of a match the log writes `### MATCH` and `### total: games ... matches ...`.
- The title shows the match counter `M a:b`.
- Examples: on the PC3 an overnight run of 13 hours gave 117 matches big against gnubg 2-ply (46.2 %). On the PicoCalc with a Pico 2 W 450 matches ran in 8.6 hours (Part 2, P7). On the Colour Maximite 2, 404 matches big against big ran in 10.4 hours (chapter 15).

12. Options

12.1 Quick menu (key O)

```
1 anim+ 2 pips+ 3 hint- 4 flip+ 5 swap+ 8 mouse+ 9 mat+ 0 sound- 6 save 7 more
```

A key toggles the option; `+` means on, `-` off.

Key	Option
1	animation of the moves
2	pips in the title
3	hint (best move, cube recommendation in the log)
4	mirror the board (home board on the left)
5	swap colours (O red, X light)
8	mouse on/off (PC3, CMM2)
9	<code>.mat</code> recording (takes effect from the next match)
0	sounds: dice, moving, landing, hitting (only with animation, needs <code>OPTION AUDIO</code>)
6	save settings (<code>mmbg_pc3.ini</code>)
7	options page

12.2 Options page (0 → 7)

The page shows **all** settings with their values. A key plus ENTER changes a setting, ENTER alone returns to the game.

```
MMbg PC3 1.10.22 - OPTIONS (C) 2026 Matthias Hollat
key + ENTER = change      ENTER alone = back to the game
-----
BOARD   1 animation      +      4 flip board      +      home board left
        2 pips in title  +      5 swap colours  +      0 red, X light
        3 hint          -      8 mouse          +      USB mouse found
        0 sound         -      dice, moves, hits (PLAY BBC SOUND)
-----
RECORD  9 match file .mat +      G log file      +
        L folder b:/bg/f/ for log and .mat files, must exist
        now: log bg_20261019_210528.log match -
-----
GNUBG   external engine E = gnuBG on a PC over WLAN (TCP)
        on the PC start gnuBG and type: external :41000
        H host 192.168.1.132 P port 41000
        W timeout 30 s F fallback 4 big
        D debug log - C check gnuBG's moves + (slower)
        V level (text for the log) 2-ply
        T test connection (version + 10 x ping)
-----
TEST RUNS U duplicate - each match again, same dice, sides swapped
          R dice seed 0 = from the clock (now 1717527)
-----
STATUS  WLAN: connected, IP 192.168.1.150 gnuBG at 192.168.1.132:41000
        S save all settings (mmbg_pc3.ini + bgext.ini)
-----

option > _
```

Figure 10: The options page (PC3). Here logs and `.mat` files are kept in a folder of their own on the SD card, gnuBG runs on the PC 192.168.1.132.

Section	Keys
BOARD	1-5, 8, 0 as in the quick menu
RECORD	9 <code>.mat</code> , G log on/off, L log folder (chapter 3.4)
GNUBG	H host, P port, W timeout, F fallback, D debug log, C check, V level, T test
TEST RUNS	U duplicate, R dice seed
STATUS	WLAN state; S saves everything (<code>mmbg_pc3.ini</code> and <code>bgext.ini</code>)

13. Command summary

The key **H** shows all commands and, below under ACTIVE, what is currently set:

```
MMbg PC3 1.10.22 - Backgammon for MMBasic (C) 2026 Matthias Hollatz
Free software, GNU GPL v3 or later. Neural nets: GNU Backgammon (GPL).

-----
MOVE      13/7 8/7  bar/22  6/off      gnuBG notation (b = bar, o = off)
          13 24  b 6                  start points only, big die first
          #3                          move 3 of the move list
          E end turn    U undo          U again: back to earlier turns
-----
MOUSE     left / right on own checker  move it with big / small die
          own dice cube board          undo / double / roll
          offered cube                  take (pass only with key P)
-----
CUBE      R roll    D double            asked before each roll
          T take    P pass              when the opponent doubles
-----
MATCH     M new     M7 to 7  K cube     new match, length, cube on/off
          SG save   LG load             it is also saved by itself
-----
GNUBG ID  position:match                paste at a prompt = set up (y/n)
                                          current ID: bottom right + terminal
-----
COMPUTER  C0 C1 C2 C3 C4                off, plays X, both, fast, turbo
          A45  C4E                      engines 0 = 4, X = 5 / + turbo
          engines: 0 base 1 book 2 tuned 3 net 4 big 5 deep E gnuBG
-----
OTHER     L D P  Ctrl+P                  list, dump, DEV column, screenshot
          O G GN H Q                    options, log, new log, help, quit
=====
ACTIVE
Match     to 5 points, 0 0 : X 0, no game yet    Cube 1
Players   0 you  X big  computer C1 plays X
Options   anim+ pips+ hint- flip+ swap+ mouse+  mat+ log+ sound- (key 0)
Files     log bg_20261019_210528.log  match -
Saved     -
Nets       small + big loaded (engines 3, 4, 5)
Bear-off   Bear-off database: bgbo15.dat (up to 15 checkers, with gammons).
WLAN       WLAN: connected, IP 192.168.1.150  gnuBG at 192.168.1.132:41000
gnuBG      level 0-ply, fallback big, check on, timeout 30 s
=====
any key = back
```

Figure 11: The help page (here in the terminal; on the HDMI screen it looks the same).

Command	Effect
13/7 8/7 , 13 24 , #3	move (chapter 5.2)
E / U	end turn / undo
R D T P	roll, double, take, pass (at the cube questions)
N / M7	new match / match to 7
K	cube on/off
SG / LG	save / load match
paste a GNUbg ID	take over the position
C0 ... C4	computer mode
A45 , A4E , C4E	engines per side (plus turbo)
L	full move list
D	board dump with GNUbg ID (chapter 7.6)
P	DEV column (terminal)
O	options
G / GN	log on/off / new log file
Ctrl+P	screenshot (BMP, PC3)
H	help with ACTIVE overview
Q	quit

14. Files

All files are in the program folder; the log folder is `log/` below it unless set otherwise (chapter 3.4).

File	Location	Written by	Contents
mmbg_pc3.ini	program folder	O → 6 or S	interface: anim, pips, hint, flip, swap, mouse, mat, sound, log, match, cube, computer, engines
bgext.ini	program folder	options page S	gnubg/WLAN, log folder (<code>logdir=</code>), duplicate, seed
mmbg_pc3.sav	program folder	automatic, SG	current match (move history, <code>.mat</code> state)
bg_YYYYMMDD_HHMMSS.log	log folder	log on	every move, cube, results
mmbg_YYYYMMDD_HHMMSS.mat	log folder	mat on	match for gnubg
mmbg_YYYYMMDD_HHMMSS.bmp	log folder	Ctrl+P	screenshot (PC3)
bglog_err.txt	program folder	on problems	write errors of the log

Contents of the package

Path	Contents
mmbg/	both programs, nets, database, ini files – copy this folder to the SD card
mmbg/log/	empty log folder
gnubg_start_PC/	batch files to start gnubg on a Windows PC (0-, 1-, 2-ply)
MMbg_Manual.pdf	this manual
LICENSE.txt , README.txt	licence, short instructions

15. MMbg on the Colour Maximite 2

Since version 1.10.34 **the same program** `mmbg_pc3.bas` also runs on the **Colour Maximite 2 (CMM2)**. This is made possible by the CMM2's firmware 7: it brings MMBasic up to the level of the PicoMite and, with `OPTION PICOMITE`, can behave like a PicoMite with HDMI. MMbg has been tested on a Generation 2 CMM2.

15.1 Requirements

What	Requirement
Computer	Colour Maximite 2 (tested: Generation 2)
Firmware	MMBasic 7.00.00b3 or later (<code>OPTION PICOMITE</code> exists only from b3)
Compatibility	<code>OPTION PICOMITE HDMI</code>
Display	VGA, 640 × 480 (MODE 3 as on the PC3)
Keyboard	USB keyboard in the upper USB socket at the back
Mouse (optional)	dual-mode mouse (USB/PS/2) in the lower USB socket at the back, with <code>OPTION MOUSE 0</code>
Terminal (optional)	USB type B socket (power and console), as on the PC3 80 × 40
WLAN	none – engine E (gnubg) is not available on the CMM2

The lower USB socket of the CMM2 is not a real USB socket: the mouse runs there in **PS/2 mode**. It therefore has to be a mouse that supports both modes, for example the Microsoft Basic Optical Mouse.

15.2 Setting up

Once at the prompt:

```
OPTION PICOMITE HDMI
OPTION MOUSE 0           (only with a mouse in the lower socket)
```

`OPTION LIST` then shows, among other things:

```
480MHz Colour Maximite 2 G2
MMBasic 7.00.00b3
OPTION PICOMITE PicoMiteHDMIUSB RP2350B, 640x480
```

The CMM2 firmware is installed via the upper USB socket and a USB A-to-A cable; this is described in the CMM2 manual (Appendix K).

15.3 Start

As on the PC3, the program folder is on the SD card, on the CMM2 at `A:/mmbg` (chapter 3.2). Start as in chapter 3.3 with `CHDIR "A:/mmbg"`, or conveniently from the **file manager FM**: change to the program folder and run `mmbg_pc3.bas`.

15.4 What is different

- **Enter key:** the CMM2 sends two characters for Enter (CR and LF), also over the terminal. MMBg sets `OPTION CRLF CR` at start-up and skips a left-over LF, so a running computer is not stopped by it.
- **Mouse:** MMBg finds the PS/2 mouse by itself (mouse channel 2). Without a mouse set up, you simply play with the keyboard.
- **No WLAN:** `[WLAN]` does not appear in the title; engine E and the connection test are missing. Everything else – engines 0 to 5, cube, undo, saving, `.mat`, log, test runs – is the same.
- **Speed:** the CMM2 (480 MHz) calculates about as fast as the PC3 (315 MHz): *big* needs ~0.6 s per move on average, *deep* ~10 s. A turbo run C4 big against big over 10.4 hours played 404 matches (~92 s per 5-point match) without a single error.

16. Notes and limits

- `mmbg_pc3.bas` runs on the PC3 with firmware V7.00.00 RC2 or later and on the Colour Maximite 2 with firmware 7.00.00b3 or later (chapter 15). For the PicoCalc: Part 2.
- **Large programs:** a program that exceeds the program memory when loaded without `, C` can damage the flash in the release candidates ("Program damaged: symbol table"). The published programs are small enough; loading with `, C` leaves extra room.
- **Stopping with Ctrl+C:** the colour palette stays changed; `MAP RESET` restores it. On a normal exit with `Q` this happens automatically.
- **Animation:** MMBg restores the background of the flying checker from a framebuffer (~150 KB PSRAM). Without the buffer it redraws the area piece by piece; that is slower but looks the same.
- **Sounds** are only available with animation (C0 to C2). Without `OPTION AUDIO` the option stays off and the log reports "no audio output". The sounds (dice, moving, landing, hit) are built from noise. Anyone who finds better settings, such as other envelopes or volumes, is welcome to report them in the TheBackShed forum. The values are in the source code in `GuSnd` and `GuSndOn`.
- **Mouse and dice:** a click on your own dice takes the move back at once (undo). After a mis-click, simply move again.
- **Match play only** up to 15 points, no money game. Undo only works within the current game.
- **A pasted GNUbg ID** ends the `.mat` recording for this match, because an arbitrary position cannot be represented in the `.mat` format.

17. A flight through the source code

For everyone who wants to look into `mmbg_pc3.bas` or change it: a short tour. The program has about 6 300 lines and 264 routines (the PicoCalc edition is derived from it automatically, Part 2). The names tell the area:

Prefix	Area
Bg...	game: rules, input, move generator, evaluation, cube, match, log, saving
Nn..., Z..., Bo...	small nets, big nets, bear-off database
Ex...	gnubg over WLAN (TCP), options page
Gu...	graphics, mouse, options, sounds
Gid..., BgId...	GNUbg ID
BgMat...	<code>.mat</code> recording

17.1 Start-up

1. **Global variables** (`DIM ...`): board `bd(0..25)` (+n = O checkers, -n = X checkers, `bd(0)` bar of O, `bd(25)` bar of X), dice `dw()`, move lists `gm$()` / `gv()`, net weights, screen state `yb()`.
2. `MODE 3`, `GuPa1` (colours), `BgLogo` (start screen), `OPTION CRLF CR` (CMM2 only, skipped elsewhere).
3. **Loading**: `NnLoad` (small nets), `BoOpen` (bear-off database), `ExIni` (gnubg settings), `BgLogDir` (create the log folder), then `ZLoad` (big nets, only if the memory is available in one piece).
4. `GuInit` (framebuffer, colours), `GuCfgLoad` (`mmbg_pc3.ini`), `GuMsInit` (mouse: USB first, otherwise PS/2 on channel 2), question about the saved match (`BgLoadG`).

17.2 The main loop

```
D0
BgShow          draw terminal and screen (computes the move list)
Input:          computer -> best move gm$(1) | human -> GuIn$ (keyboard, mouse)
Command?       H, O, N, M7, C1, A45, SG, LG, U, GNUbg ID ... -> execute
Move?          BgInput -> BgTry -> BgMove          (check and move)
                GuAnim -> GuMv -> GuFly           (animation, sounds)
                BgNext -> BgAuto -> BgTurn         (change sides, cube, roll)
Game over?     BgEndGame                          (points, log, .mat, next game)
LOOP
```

17.3 How the computer finds its move

1. `BgTurn`: first `BgCubeTurn` (cube before the roll; `BgCube` computes the match winning chances like gnubg 0-ply), then `BgRoll`.
2. `BgPrep` calls the **move generator** `BgGen`: it tries all dice orders and start points with its own stack instead of recursion (the stack only allows a few levels). `BgMx` first checks how many dice can be played at most.
3. Every finished move lands in `BgLeaf`; duplicate final positions are dropped. `BgEval` rates the position: formula (engines 0-2) or net via `NnEval` → `NnFwd` (small), `ZFwd` (big), `NnBo` (bear-off

database).

4. BgSort sorts, ZBest recalculates the best 12 with the big net, ZThink looks one roll ahead for *deep*, BgBook puts the book move on top when needed, ExMove asks gnubg (ExMvKey\$ plays gnubg's answer on trial and finds it in the list, or adds it if the list is full).
5. Place 1 of the list is played.

17.4 Further building blocks

Routines	Task
BgGid\$, BgIdGet, BgIdPut	create, check and take over a GNUbg ID
BgHist, BgBack	move history (one ID per turn start) and undo
BgSaveG, BgLoadG	write and read the saved match
BgMatNew, BgMatTurn, BgMatGame	.mat in gnubg's format
BgLog, BgLogTurn, BgLogDir	log file, one line per move, log folder
ExAsk\$, ExBoard\$, ExCube, ExMove	conversation with gnubg over TCP (FIBS board line)
GuFull, GuDraw, GuPt, GuDice	redraw the screen or only what changed
GuFly, GuNear, GuRep	flying checker, mouse pointer, fallback without framebuffer
GuHit\$, GuShort\$	translate a mouse click or short input into a move
GuSnd, GuSndOn	sounds with PLAY BBC SOUND
BgHelp, ExMenu, GuOpt	help, options page, quick menu

Part 2 - MMbg PiCa (PicoCalc)

MMbg PiCa has version 1.10.33. The changes of 1.10.34 only concern the PC3 and the Colour Maximite 2 (PS/2 mouse, Enter key); there is nothing new for the PicoCalc.

Part 2 describes only **what is different on the PicoCalc**. For the game itself – moving, cube, undo, saving, `.mat`, log, GNUbg ID, engines, gnuhg over WLAN and test runs – Part 1 applies unchanged; the references (e.g. "→ 5.2") point to the chapters there.

P1. MMbg PiCa at a glance

- **One program** (`mmbg_pica.bas`) for the PicoCalc with a **Pico 2** (firmware PicoMite RP2350) or a **Pico 2 W** (firmware WebMite RP2350). It recognises the firmware at start-up by itself.
- **The same game as on the PC3:**
 - all engines up to *big* and *deep* (→ 8.1),
 - doubling cube, Crawford, saving and resuming, multi-level undo, GNUbg IDs, `.mat` recording, log, sounds (→ 5-7),
 - computer modes C2 to C4 and long test runs (→ 8.3, 11),
 - gnuhg as opponent over WLAN and the terminal over telnet (Pico 2 W only, → 10).
- **The same playing strength as the PC3:** the nets are the same gnuhg weights, stored packed. The 16-bit nets of the Pico 2 W lose practically nothing (P7).
- Runs with **RAM and SD card only**. No PSRAM, no flash slots and no `OPTION SYMBOLS OFF` are needed.

P2. Requirements and firmware set-up

What	Requirement
Computer	PicoCalc with Pico 2 (RP2350, without PSRAM) or Pico 2 W
Firmware	PicoMite RP2350 or WebMite RP2350, V7.00.00 RC3 or later
SD card	drive B: for the program folder (program, nets, logs)
Terminal (optional)	VT100, 80 × 40, over USB serial or telnet (WebMite)
WLAN (optional)	Pico 2 W only: gnuhg as opponent (→ 10), terminal over telnet

Set up the firmware once at the prompt:

```
OPTION RESET PICOALC      (LCD, keyboard, SD card, sound)
OPTION COMPILE ON         (important for speed)
OPTION WIFI "MyNetwork", "Password" (Pico 2 W only)
OPTION TELNET CONSOLE ON  (Pico 2 W only, if wanted)
```

P3. Installation and start

P3.1 One folder for everything

As on the PC3 (→ 3.1), all files live in **one folder**, for example `B:/mmbg`; name, drive and depth are free. The package contains the files for all systems. Each edition takes only what it needs (P8), so the same folder works for a PC3, a CMM2, a Pico 2 and a Pico 2 W.

Folder	Contents
<code>mmbg</code> (any name)	programs, nets, database, ini files, saved match
<code>mmbg/log</code>	logs and <code>.mat</code> files; MMbg creates the folder at start-up (→ 3.4)

P3.2 Start

```
CHDIR "B:/mmbg"
LOAD "mmbg_pica.bas", C
RUN
```

The start screen shows, one after another:

Line	Meaning
Nets: bgnet345.dat (...ms).	small nets loaded
Big net 16 bit: 188/250 RAM, rest SD.	16 bit on the Pico 2 W, 32 bit on the Pico 2, and how many of the 250 rows of the big net fit into the RAM buffer (P6.3); with <code>250/250</code> RAM. a whole net is in RAM
Bear-off db: bgbo15.dat (with gammons).	endgame database
Settings: mmbg_pica.ini.	settings loaded
RAM free: ... KB.	free memory after start-up

MMbg then waits for **N** (new match). If a match was saved, it asks whether to continue it (→ 6.2).

P4. Screen, terminal and controls

P4.1 LCD (320 × 320)

- **Title line** (40 characters): players, score, match length and the switches as on the PC3 (→ 4.1): `W` WLAN, `M±` `.mat`, `L±` log.
- **Board**: point 23 px, checker 19 px, bar and tray. As on the PC3, the dice lie to the right of the player as seen from his side.
- **Log**: the last 4 messages of 40 characters.
- **Prompt**: at the very bottom, 40 characters. All questions are at most 39 characters long, e.g. `Engine 0 (0-5, E) >`. The meaning of the numbers is on the help page.

P4.2 Terminal

Over USB serial or telnet the terminal shows **the same ASCII board as on the PC3** (→ 4.2), 80 × 40, with move list, DEV column and GNUbg ID. MMbg switches the terminal to 80 × 40 at start-up if the terminal program allows it (Tera Term, PuTTY). Input works from the PicoCalc keyboard and from the terminal alike.

P4.3 Text pages (help, options, list, dump)

All text pages have **40 columns in Font 1 (8 × 12)**, 26 lines. Longer pages are split into **two pages**:

Page	Contents	Turning pages
H help 1/2	all commands (→ 13)	arrow down or any key → page 2, Q back
H status 2/2	the current state: match, cube, players, computer mode, options, log folder and file, <code>.mat</code> file, saved match, nets (rows in RAM), bear-off database, WLAN, gnuBG, GNUbg ID	arrow up → page 1, any other key back
O → 7 options 1/2	board and recording: 1-5, 0, 9, G, and L with the log folder in a line of its own	arrow down → page 2
O → 7 options 2/2	gnubg and test runs: H, P, W, F, D, C, V, T, U, R and the WLAN status	arrow up → page 1
L move list	20 moves per page	any key → more, Q back
D dump	board in two columns, dice, GNUbg ID , rows of the big net in RAM, SD reads and net loads	any key → back

On the options page **all keys work on both pages**; after a change the page with the changed value is shown. **S** saves everything (`mmbg_pica.ini` and `bgext.ini`). The terminal shows the same pages, with the input line directly below them.

Copying a GNUbg ID by hand: key **D** and status page 2/2 show the ID on the LCD. In the terminal it is at the bottom right as on the PC3. An ID is pasted at the prompt as usual (→ 7.4).

P4.4 Quick menu (key O)

As on the PC3 (→ 12.1), shown in the log area of the LCD and without the mouse switch: `1 - 5 , 9 , 0 , 6 save , 7 more .`

P4.5 Turbo mode C4 on the LCD

In turbo mode (→ 8.3, 11) MMbg switches the LCD **black** and the backlight to the **lowest level**, so that nothing stands still on the screen for hours. Any key stops the run; the backlight returns to its previous level and the board is redrawn.

The backlight is set by the keyboard controller of the PicoCalc, which accepts values from 16 to 240; MMbg uses `OPTION BACKLIGHT LCD` , which also stores the value in the flash. If a turbo run is

ended with **Ctrl+C**, the restore is skipped: set the backlight again with `OPTION BACKLIGHT LCD n` (n from 16 to 240).

P5. Differences from the PC3

	PC3	PiCa
Screen	HDMI 640 × 480	LCD 320 × 320
Text pages	80 columns	40 columns, help and options on 2 pages each
Mouse, screenshot (Ctrl+P)	yes	no (the LCD cannot be read back)
Animation	with framebuffer (PSRAM)	without framebuffer, the area is redrawn; it looks the same
GNUbg ID	bottom right in the log	in the terminal, with D and on status page 2/2
Move list	500 moves	300 moves (only a few doubles have more, → 10.2)
Undo	300 turn starts	160 (Pico 2), 80 (Pico 2 W)
<code>.mat</code>	400 actions per game	250
Log rank in the move list	all moves	the first 40
Big nets	floating point (8 bytes) in PSRAM	packed: 32 bit (Pico 2) or 16 bit (Pico 2 W), row buffer in RAM, rest from the SD card
Small nets	floating point	first layer packed (32 or 16 bit)
Turbo C4	screen without board drawing	LCD dark, backlight lowest (P4.5)
WLAN / telnet	yes	Pico 2 W only

Everything else – game, engines, cube, saving, `.mat`, log and gnuBG – is **the same**.

P6. Nets and memory on the PicoCalc

P6.1 The task

The big gnuBG net has 250 inputs × 128 neurons = **32 000 weights**, and there are two of them (contact and crashed, → 9.2).

- **As floating-point numbers** (8 bytes each, as on the PC3) one net needs **256 KB**, both together 512 KB. The PC3 has PSRAM for this.
- **The PicoCalc** has no PSRAM. With firmware V7.00.00 RC, about **~270 KB** of RAM are free after loading MMBg on a Pico 2 W; the Pico 2 has ~50 KB more. MMBg's other variables take ~140 KB, and during play MMBasic needs another ~31 KB for local variables and temporary values.

So the weights must be stored more compactly, and not all of them have to be in RAM at the same time.

P6.2 Packed weights

Each weight is stored as an unsigned whole number `u` and converted back when calculating:

$$\text{weight} = l_0 + u \times q$$

	Pico 2 (32 bit)	Pico 2 W (16 bit)
File per net	<code>bgnet0.pk</code> , <code>bgnet2.pk</code> , 128 000 bytes each	<code>bgnet0.p16</code> , <code>bgnet2.p16</code> , 64 000 bytes each
Scale	one per net: $l_0 = -256 / -512$, $q = 2^{-23} / 2^{-22}$	one per input row: l_0 = smallest weight of the row, q = (largest – smallest) / 65 535; files <code>bgnet0.s16</code> , <code>bgnet2.s16</code> , 4 000 bytes each
Rounding error per weight	at most $q/2 \approx 0.00000012$	at most $q/2$: typically 0.0003 to 0.0006, at most 0.0036 (P7)
Check sums	<code>bgpk.dat</code>	<code>bgpk16.dat</code>

MMbg checks the check sums **once at start-up**. A damaged file is reported on the start screen, and the engines fall back to the small nets.

P6.3 One row per input, and the row buffer

The weights are stored **row by row per input**: one row holds the 128 weights from one input to all neurons (512 bytes with 32 bit, 256 bytes with 16 bit). Of the 250 inputs only about 50 are non-zero in a real position, so each evaluation needs only these ~50 rows. Each row is processed with three fast firmware commands:

```
MEMORY UNPACK (unpack the row) -> MATH SCALE ( $\times$  input  $\times$  q) -> MATH C_ADD (add up)
```

At start-up MMbg creates a **row buffer** `pk%()` as large as the free RAM allows, leaving a **reserve of 44 KB** for the game (the measured need is ~31 KB, plus a safety margin).

- **All 250 rows fit:** a whole net is in RAM. When a position changes between contact and crashed, MMbg loads the other net from the SD card. The pre-filter sorts the candidates so that this happens at most once per move list.
- **Otherwise (partial buffer):** the buffer holds as many rows as possible, mixed from both nets. A missing row is read from the SD card (`SEEK` + `MEMORY INPUT` , under 1 ms per row). Free places are found with the **clock method**: rows that were used recently get a second chance before they are overwritten.

With firmware V7.00.00 RC both PicoCalc versions use a partial buffer of ~170 to ~190 of the 250 rows (the Pico 2 has more RAM, but needs twice the bytes per row). Measured on a Pico 2 W:

Rows in RAM	SD reads	computing time per move (move list and evaluation)
25 (test only)	~87 per second	~0.6 s on average (34 games)
~188	~3 per second	~0.5 s on average (450 matches)

With the normal buffer the SD card hardly slows down the play; the computing time is the same as on the PC3.

P6.4 Small nets

The first layer of the small nets (contact, crashed, race: 200×16 and 200×8 weights) is packed as well: 32 bit on the Pico 2, 16 bit with one scale per net on the Pico 2 W. This saves 32 or 48 KB. The second layer is small and stays floating point.

P7. The 16-bit nets: rounding and playing strength

P7.1 Where the rounding comes from

With 16 bits a weight can take 65 536 different values between the smallest and the largest weight **of its row**. Because every row has its own scale, the steps are fine: a typical row spans about 45 (contact net) to 80 (crashed net) units, so one step is about 0.0007 to 0.0012, and the rounding error of a weight is at most half a step – typically 0.0003 to 0.0006, in the widest row 0.0036. In one evaluation about 50 rows $\times$ 128 neurons are added up; the errors are random in sign and largely cancel out, and the S-curves of the hidden layer damp them further.

P7.2 How much does a position value change?

The 16-bit evaluation was checked against the full-precision evaluation of the PC3 with a Python model of MMbg's evaluation (inputs, both net sizes, move generator, pre-filter) and the logs of real runs on the PicoCalc:

Check	Result
1 014 positions of a run against gnutbg 2-ply	deviation of the equity at most 0.0011
2 274 positions of the overnight run	deviation at most 0.0012
680 move decisions from 34 games	679 identical moves ; once two moves were practically equal (difference 0.00002) and swapped places

A deviation of 0.001 equity only changes the move when two moves are almost exactly equal – and then it hardly matters which of the two is played.

P7.3 Estimated loss of playing strength

From the check above: one swapped decision in 680, costing 0.00002 equity, gives an average loss of about **0.00000003 equity per move** (3×10^{-8}). For comparison, the big net itself makes an average error of **0.0026 equity per move** (2.6 mEq) against gnutbg 2-ply ($\rightarrow$ 8.1). The rounding loss is therefore about **one hundred-thousandth** of the net's own error – far below anything that can be measured in games.

P7.4 Measured in play

An overnight run on a Pico 2 W (16 bit, big against gnutbg 2-ply, 5-point matches, turbo C4):

	Pico 2 W, 16 bit	PC3, floating point
Matches won	199 of 450 = 44.2 % ± 4.6 %	577 of 1236 = 46.7 %
Games won	778 of 1631 = 47.7 %	-
Run time	8.6 h, ~69 s per match	-
Computing time per move	~0.5 s	~0.5 s

The difference of 2.5 percentage points is well within the statistical spread (about one standard deviation); with 450 matches a difference of this size is expected by chance alone. Together with P7.3: **the PicoCalc plays as strongly as the PC3.**

P8. Files

File	PC3 / CMM2	Pico 2	Pico 2 W	Contents
mmbg_pc3.bas	✓			program PC3 and CMM2
mmbg_pica.bas		✓	✓	program PiCa
bgnet345.dat	✓	✓	✓	small nets (text)
bgnet02.dat	✓	✓	✓	output layer of the big nets
bgnet0.bin , bgnet2.bin , bgesc.bin	✓			big nets, floating point
bgnet0.pk , bgnet2.pk , bgpk.dat		✓		big nets, 32 bit
bgnet0.p16 , bgnet2.p16 , bgnet0.s16 , bgnet2.s16 , bgpk16.dat			✓	big nets, 16 bit with row scales
bgesc.pk		✓	✓	escape tables (bytes)
bgbo15.dat	✓	✓	✓	bear-off database
bgnet35.dat , bgbo10.dat	✓	✓	✓	fallbacks (→ 3.1)

Created by MMbg itself:

- in the program folder: mmbg_pica.ini (interface, with log=), bgext.ini (gnubg, logdir= , test runs; shared with the PC3), mmbg_pica.sav (saved match);
- in log/ : bg_YYYYMMDD_HHMMSS.log and mmbg_YYYYMMDD_HHMMSS.mat .

P9. Notes and limits

- **Image retention on the LCD:** the PicoCalc panel can keep a faint after-image of a picture that stood unchanged for a long time, such as text at the prompt. It shows later as a shadow on dark colours, for example on the felt of the board. It fades again: switch off, or show a bright area for a while (BOX 0,0,320,320,0,RGB(WHITE),RGB(WHITE)). Turbo mode C4 darkens the screen for this reason (P4.5).

- **Dice:** in normal play they come from `RND` ; on the RP2350 the firmware keeps reseeding it from the hardware random number generator. Only with **duplicate** (option U, → 11) are the dice deliberately repeatable from a seed. Without an own value (option R) the seed comes from the clock, which on the Pico 2 starts at 01-01-2024 after every power-on. For real test runs, set a seed.
- **Terminal size:** the automatic switch to 80 × 40 only works if the terminal program allows it. Otherwise set it by hand.
- **Telnet:** according to the firmware release notes, RC4 fixes errors in the telnet output. If characters are missing in a telnet terminal, this is the likely reason.
- **Ctrl+C in turbo mode:** see P4.5 (backlight).
- **Error messages with a line number** refer to the program without comment lines when it was loaded with `, C` .

Licence and thanks

MMbg is **free software** under the **GNU General Public License v3 or later**: it may be used, passed on and modified, also commercially. Whoever passes it on must do so under the GPL again, with source code. There is no warranty.

```
MMbg - Backgammon for MMBasic
Copyright (C) 2026 Matthias Hollatz - Soli Deo gloria
Neural network weights derived from GNU Backgammon,
(C) the GNU Backgammon authors (GPL v3 or later).
```

Thanks to

- **Geoff Graham**, the creator of MMBasic and the Maximite,
- **Peter Mather** for the PicoMite and WebMite firmware, the firmware for the PC3, the PicoCalc and the Colour Maximite 2, and the BBC sound engine,
- the authors of **GNU Backgammon** for nets, methods and testing via [external](#),
- the community in the **TheBackShed forum**.

Have fun playing!