



MMbg PC3 – Backgammon for MMBasic

Manual · Version 1.10.22 · 19 October 2026

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

Important: This version runs **only on the Pico Computer 3 (PC3)** with firmware **PicoMitePC3 V7.00.00 RC2 or later** (as of today; framebuffer graphics, sound and all tests on RC2). It does not run on other PicoMite boards or with older firmware. There is a separate edition for the PicoCalc (MMbg PiCa).

Contents

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. Notes and limits
16. Licence and thanks
17. A flight through the source code

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 over HDMI (640 × 480) with smoothly animated moves. At the same time a serial terminal shows an ASCII board with move list, evaluations and log.
- **Controls:** Play with keyboard or USB 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
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
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 (optional)	Drive B: for program, logs and <code>.mat</code> files

The big net needs about 600 KB of memory, the animation another 150 KB for a framebuffer. The PC3 has enough PSRAM for this, which is why this edition runs only there.

3. Installation

3.1 Files

All files belong **in the same folder** as the program, either on the flash drive **A:** or in a folder on the SD card **B:** (chapter 3.3).

File	Size	Purpose	If missing ...
mmbg_pc3.bas	approx. 190 KB	the program	–
bgnet345.dat	approx. 60 KB	small nets 3, 4, 5 (gnubg)	engines 0–2 only
bgnet02.dat	text	output weights of the big nets	no engines 4 and 5
bgnet0.bin, bgnet2.bin	256 000 bytes each	big nets contact and crashed	no engines 4 and 5
bgesc.bin	small	escape tables (gnubg)	no engines 4 and 5
bgbo15.dat	approx. 1.6 MB	bear-off database up to 15 checkers	race with net 5 only

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

3.2 Transfer and start

Step 1: Transfer each file from the PC to the PC3 with `XMODEM RECEIVE "name"`, or copy it via the SD card.

Step 2: Load and start the program:

```
LOAD "mmbg_pc3.bas", C
RUN
```

With `, C` MMBasic leaves out the comments. The published version has only few comments and also loads without `, C`.

Step 3: 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).

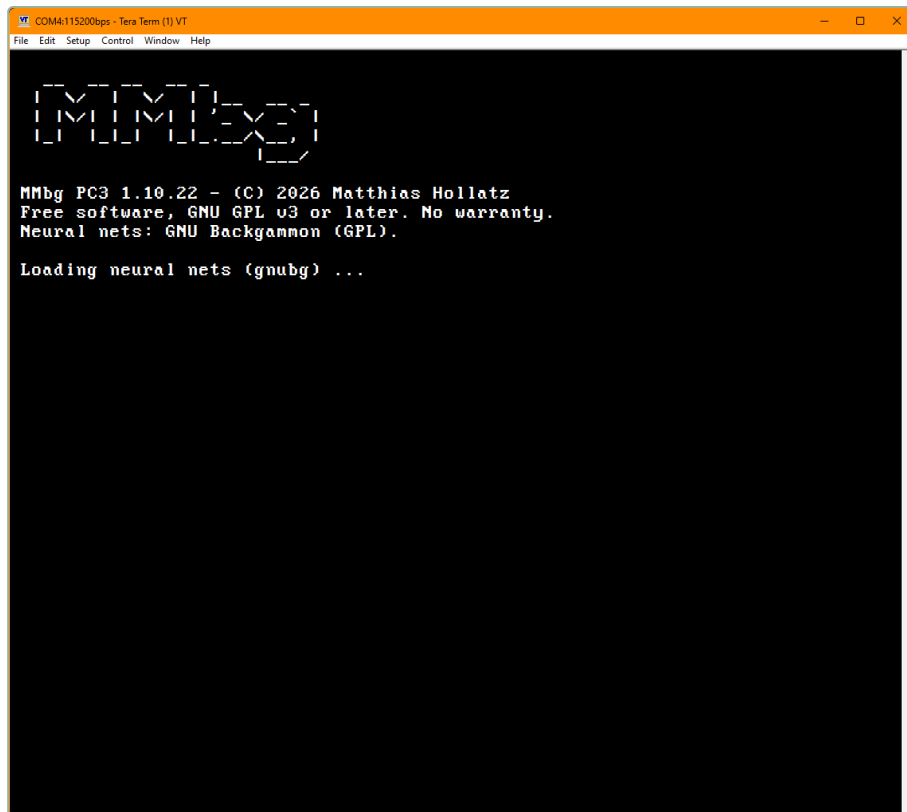


Figure 1: Start in the terminal – logo, licence, the nets are being loaded.

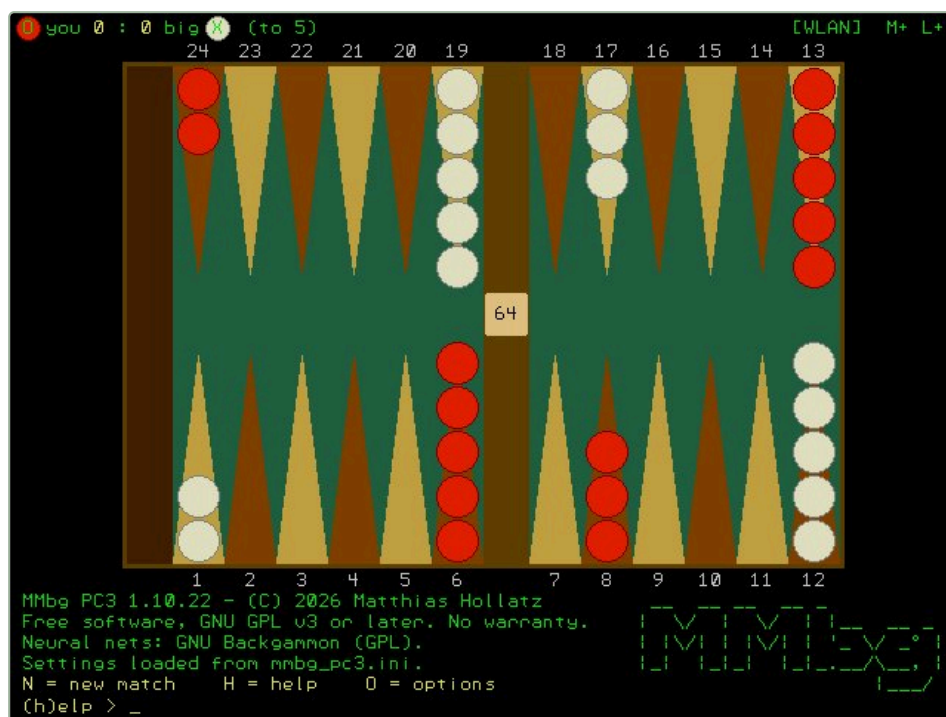


Figure 2: Start screen on HDMI. Bottom left version, licence and messages, on the right the logo, below the most important keys.

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

3.3 Everything in one folder on the SD card

Program, nets and database can also live together in one folder on **B:**. The flash drive A: then stays free for other things. Recommended:

Folder	Contents
B:/mmbg	program, nets, database, ini files, saved match
B:/mmbg/log	logs, .mat files and screenshots

Set up once:

```
MKDIR "B:/mmbg"  
MKDIR "B:/mmbg/log"
```

Then copy all files from chapter 3.1 to B:/mmbg. Start from this folder, so that MMbg finds its files there and creates the ini files there:

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

Finally set the log folder: **O** → **7**, then **L** and enter B:/mmbg/log, save with **S**. So the folder for logs, .mat files and screenshots is independent of the program folder and can be somewhere else entirely. It must exist beforehand.

4. The screen and the terminal

4.1 HDMI screen

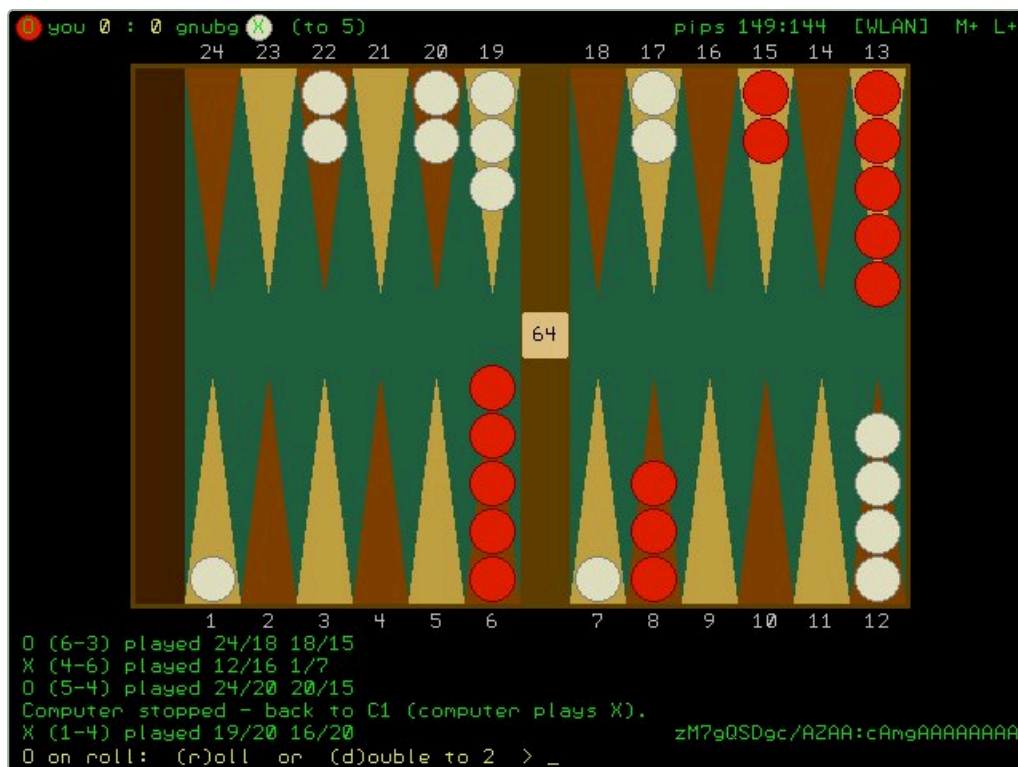


Figure 3: A game in progress. Title with players, score, pips and switches, below it the board with the cube in the bar, the five-line log, the GNUbg ID bottom right and the prompt at the very bottom (here the question before the roll).

- **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.

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: b , o ; - instead of / 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:

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	move it with the smaller die
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 P

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`:
- 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 `mmbg_pc3.sav`: 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=1.10.22
date=19-10-2026 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=B:/mmbg/log/mmbg_20261019_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 gnubg

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

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

- The format is the same one gnubg 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 gnubg X is player 0 (top), O is player 1 (bottom).

Analysis in gnubg:

1. Transfer the file to the PC, e.g. via XMODEM or SD card. XMODEM pads the end of the file with zero bytes; gnubg does not mind.
2. In gnubg: *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 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 gnubg'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 gnubg, the response time. There are also lines with `---` (cube, engines, settings, undo), `***` (end of game), `###` (end of match and running total) and `!!!` (errors).

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

	.mat 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:cAmgAAAAAAAAA
```

- **MMbg → gnuBg**: The ID is shown at the bottom right. 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 Screenshot (Ctrl+P)

At any input **Ctrl+P** saves the whole HDMI screen as a BMP file (24 bit, approx. 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)	577 of 1236 = 46.7 %
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 s per move.

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 screen, 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.

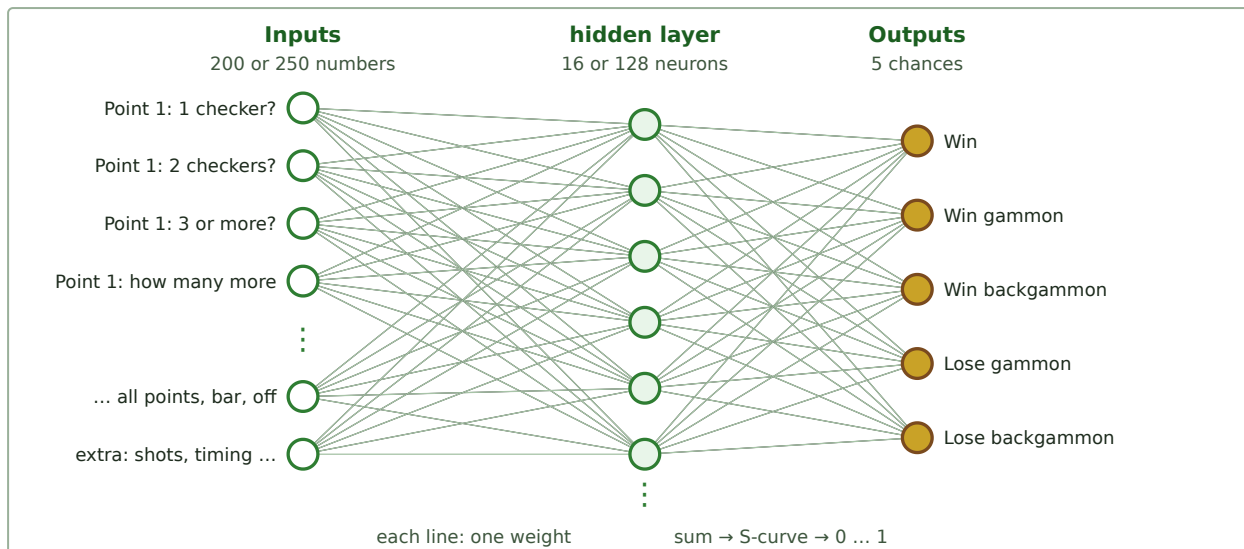


Figure 11: Structure of a gnubg net (shown reduced). On the left the position as numbers, in the middle the hidden layer, on the right the five chances.

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.

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.

Which nets each engine needs:

Engine	needs
0 to 2	nothing, only the program
3 net	<code>bgnet345.dat</code>
4 big, 5 deep	additionally <code>bgnet02.dat</code> , <code>bgnet0.bin</code> , <code>bgnet2.bin</code> , <code>bgesc.bin</code> and enough memory (PSRAM)
all	<code>bgbo15.dat</code> makes the endgame exact

10. Playing against gnubg over WLAN

The PC3 can use **GNU Backgammon on a PC** in the home network as opponent. MMbg asks the questions, gnubg does the calculating.

10.1 Setting up

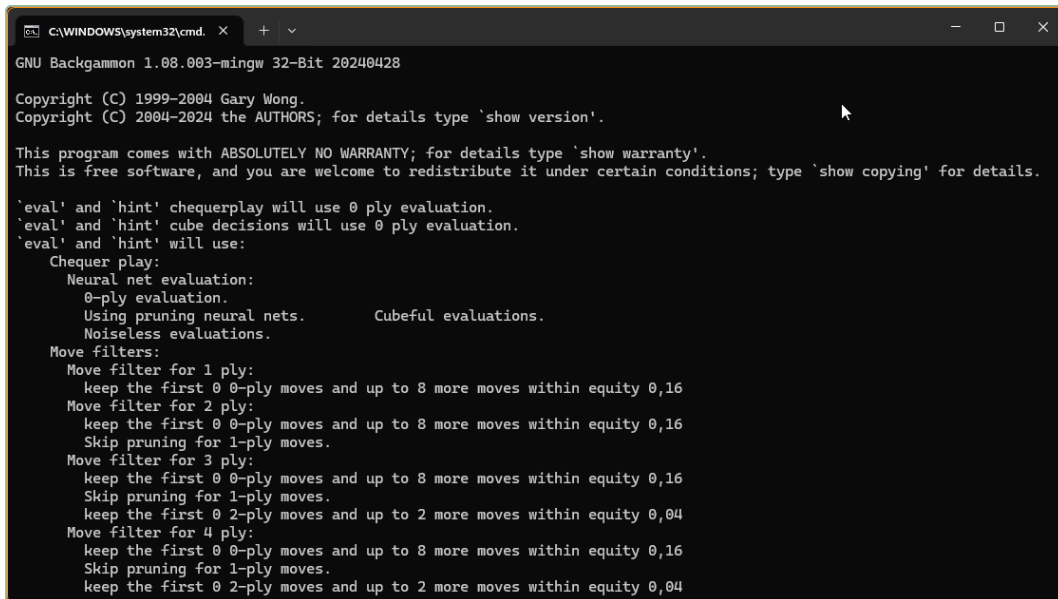
Step 1 - WLAN on the PC3: set up once and restart:

```
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.



```
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 5: `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 = gnuhg on a PC over WLAN (TCP)
      on the PC start gnuhg 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 gnuhg'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 gnuhg at 192.168.1.132:41000
      S save all settings (mmbg_pc3.ini + bgext.ini)
-----

Connected in 7 ms. gnuhg 1.08.003-mingw 32-Bit Answer: Interface: 2!RFBF:
0!Engine: 1.0!Software: 1.08.003-mingw 32-Bit Ping 10x: 10 ok, avg 29 ms,
max 102 ms

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

```

Figure 6: Connection test with T successful: gnuhg 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 = gnuhg on a PC over WLAN (TCP)
      on the PC start gnuhg 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 gnuhg'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 gnuhg 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 gnuhg waiting? (gnuhg: external :41000)

connect 192.168.1.132:41000 failed: No response from client

option > _

```

Figure 7: Connection test failed: gnuhg 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

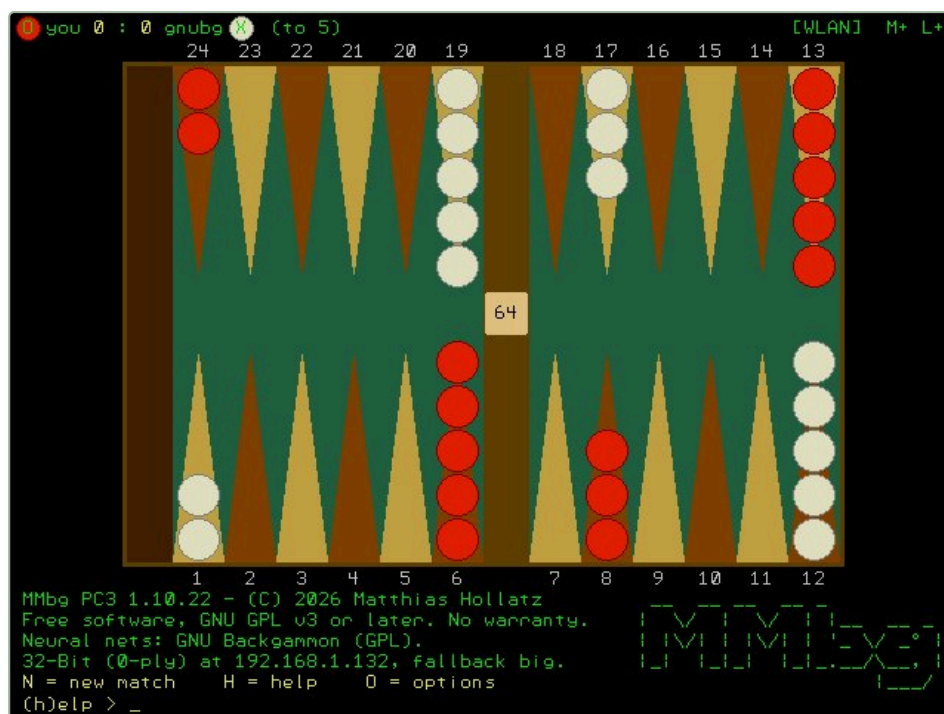


Figure 8: X is gnubg. The log reports version, level (0-ply), address and the fallback engine (big).

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 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 screen, endless matches, one status line with intermediate results. **C3** runs with screen but without pauses.
3. With the **log** on, every move goes into the log file. For long runs the log folder should be on the SD card, e.g. B:/mmbg/log (chapter 3.3).
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.
- Example: an overnight run of 13 hours gave 117 matches big against gnubg 2-ply, 46.2 %, on average 6.5 minutes per match.

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
9	.mat recording (takes effect from the next match)
0	sounds: dice, moving, landing, hitting (only with animation, needs OPTION AUDIO)
6	save settings (mmbg_pc3.ini)
7	options page

12.2 Options page (O → 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 Hollar
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)
-----

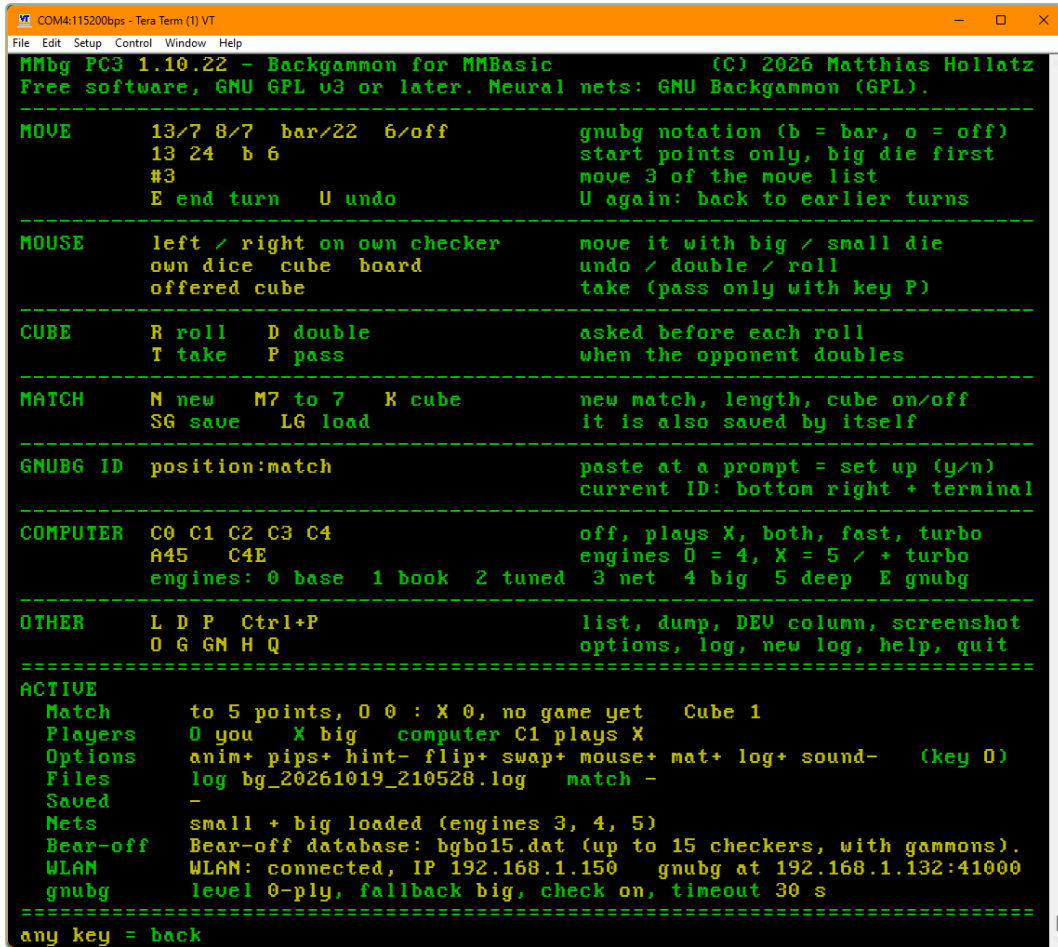
option > _
```

Figure 9: The options page. Here logs and `.mat` files are kept in their own folder 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 folder for log, <code>.mat</code> and screenshots
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:



```
COM4:115200bps - Tera Term (1) VT
File Edit Setup Control Window Help
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      gnu bg 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 gnu bg
=====
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 O)
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 gnu bg at 192.168.1.132:41000
gnu bg    level 0-ply, fallback big, check on, timeout 30 s
=====
any key = back
```

Figure 10: 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 (terminal)
P	DEV column (terminal)
O	options
G / GN	log on/off / new log file
Ctrl+P	screenshot (BMP)
H	help with ACTIVE overview
Q	quit

14. Files

File	Location	Written by	Contents
mmbg_pc3.ini	program folder	O → 6 or S	interface: anim, pips, hint, flip, swap, mouse, mat, sound, match, cube, computer, engines
bgext.ini	program folder	options page S	gnubg/WLAN, log folder, duplicate, seed
mmbg_pc3.sav	program folder	automatic, SG	current match (move history, .mat 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
bglog_err.txt	program folder	on problems	write errors of the log

15. Notes and limits

- **PC3 only, firmware V7.00.00 RC2 or later only** (see cover).
- **Large programs in RC2:** a program that exceeds the 256 KB program memory when loaded without `, C` can damage the flash ("Program damaged: symbol table" in an endless loop). The published version is small enough; development versions with all comments must be loaded with `, C`.
- **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 as before; 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 and are a first attempt. Anyone who finds better settings, such as other envelopes or volumes, is welcome to report them in the TheBackShed forum; they will go into the next version. The values are in the source code in `GuSnd` and `GuSnd0n`.
- **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.

16. 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
Neural network weights derived from GNU Backgammon,
(C) the GNU Backgammon authors (GPL v3 or later).
```

Thanks to

- **Geoff Graham**, the creator of MMBasic,
- **Peter Mather** for the PicoMite firmware, the PC3 firmware 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!

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 6 200 lines and 260 routines. 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...	.mat 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`, `GuPal` (colours), `BgLogo` (start screen).
3. **Loading**: `NnLoad` (small nets), `BoOpen` (bear-off database), `ExIni` (gnubg settings), then `ZLoad` (big nets, only if the memory is available in one piece).
4. `GuInit` (framebuffer, colours), `GuCfgLoad` (`mmbg_pc3.ini`), `GuMsInit` (mouse), 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 PicoMite 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.
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 mmbg_pc3.sav
BgMatNew, BgMatTurn, BgMatGame	.mat in gnuhg's format
BgLog, BgLogTurn	log file, one line per move
ExAsk\$, ExBoard\$, ExCube	conversation with gnuhg 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