

AutoBleem 2 User Manual

AutoBleem 2 15a36df

Contents

1. What you get

2. Installation

- 2.1 PlayStation Classic
- 2.2 Raspberry Pi
- 2.3 PC USB stick
- 2.4 Windows

3. Using AutoBleem

- 3.1 The launcher
- 3.2 Controls
- 3.3 The sets
- 3.4 The Quick menu
- 3.5 The system menu
- 3.6 Options
- 3.7 A game's settings
- 3.7 Memory cards and save states
- 3.8 Starting games, RetroArch and Apps
- 3.9 Adding games
- 3.10 The PS1 BIOS
- 3.11 Updates
- 3.12 Extensions and the AutoBleem Store
- 3.13 Scanner processors

4. Screens

- 4.1 Game Manager
- 4.2 Hardware Information
- 4.3 The button guide
- 4.4 The on-screen keyboard

5. On the PC

- 5.1 UpdateRoms - refreshing a console stick
- 5.2 LAN Share - your games and discs on the server on your network

6. The console tools (PlayStation Classic)

- 6.1 PSC-Bios
- 6.2 ABFlashKit - the AutoBleem kernel

7. If something goes wrong

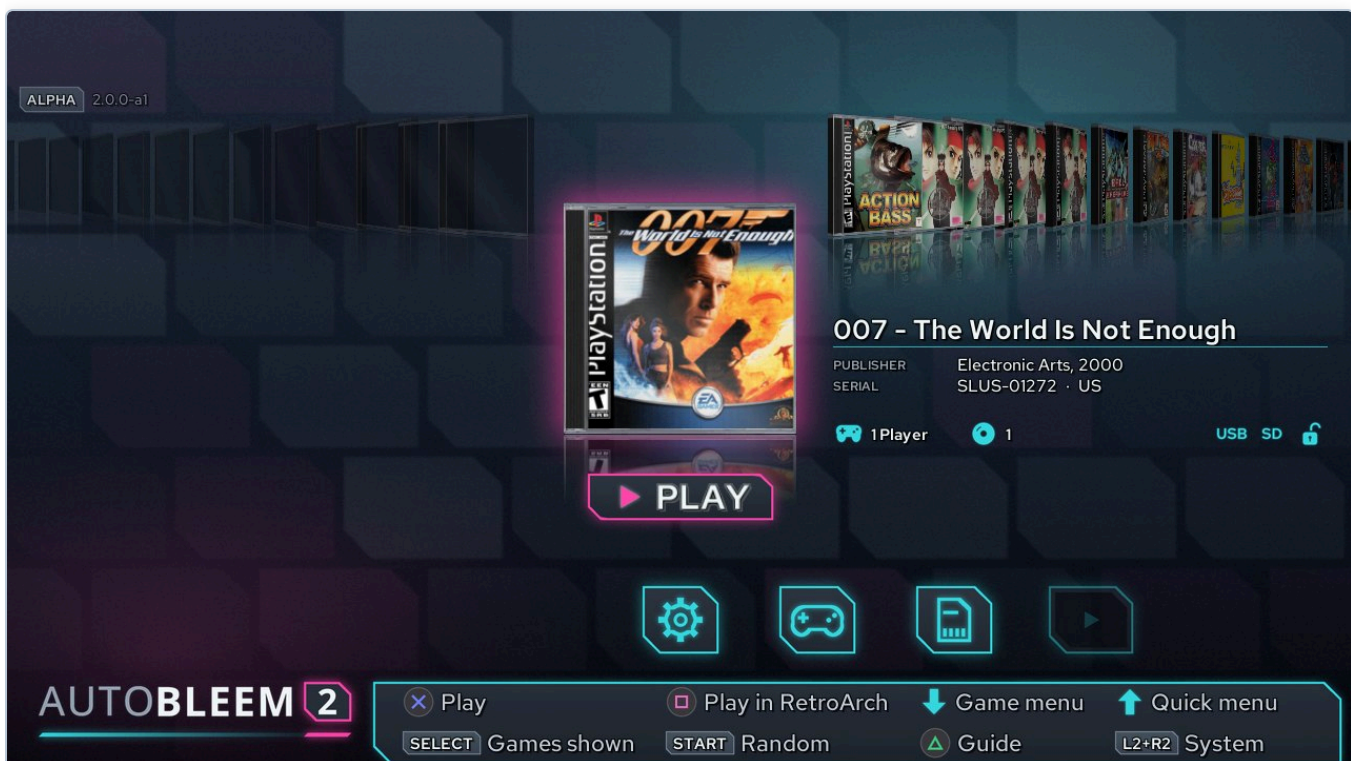
AutoBleem 2 User Manual

AutoBleem 2 is a game launcher for the **PlayStation Classic** - and, since version 2, for a **Raspberry Pi**, a **PC booted from a USB stick** and **Windows**. It shows your PS1 games as a shelf of covers with their box art and details, starts them in the bundled PCSX emulator, and, with RetroArch installed next to it, plays the games of other systems too. This manual covers installing it on each platform, everyday use, and the tools that come with it.

The downloads for every platform are at <https://autobleem.retromenele.pl/>. The page is organised by platform: the *Install* panel of each one is what you download; the *Build inputs* below it are what the installers fetch by themselves.

1. What you get

- **The launcher** - the carousel of covers, the sets (PlayStation, RetroArch, Apps), the game details, the Quick menu and the system menu, the options, the memory-card and save-state tools, and the Store for downloading apps and games. The same program on every platform.
- **Two PS1 emulators** - `pcsx-abnxt`, the current one (the default), and `pcsx-ab`, the classic one AutoBleem has always shipped. You pick one in the options; both use the same settings and memory cards.
- **RetroArch** (optional on every platform) for the other systems: NES, SNES, Mega Drive, Game Boy, arcade and many more. AutoBleem builds its RetroArch lists from the ROMs you copy in and starts each game with the right core.
- **The console tools**: *PSC-Bios* (shown in the menus as *Network & Controllers*) for WiFi, the clock, Bluetooth and gamepad mapping - on the console, and also on a Raspberry Pi and the PC stick - and *ABFlashKit* for installing the AutoBleem kernel (PlayStation Classic only).
- **UpdateRoms** for Windows: refreshes the RetroArch lists and box art of a console stick on a PC, because the console itself has no network.



The launcher: the shelf of covers, the selected game's details, the button hints

2. Installation

2.1 PlayStation Classic

You need a Windows PC, a USB stick (USB 2.0, 8 GB or more) and the stock console. The stick does not have to be empty and is **not formatted** unless you ask for it: AutoBleem is installed next to what is on it, and a stick you already use with AutoBleem 1.0 is upgraded in place (below). AutoBleem runs from the stick without any change to the console. The stick must be **FAT32** for a stock console - its kernel cannot read exFAT. Only a console with the AutoBleem kernel installed (ABFlashKit, chapter 6) also boots from an exFAT stick, which lifts FAT32's 4 GB file limit.

1. Download **AutoBleemInstaller-<version>.zip** from the PlayStation Classic panel of the site and unpack it anywhere. It holds `AutoBleemInstaller.exe` and its README; AutoBleem itself is downloaded while it installs, so the PC needs the internet.
2. Plug the stick in and start `AutoBleemInstaller.exe`. Pick the **channel** - *Release* (the tested version), *Testing* (the next version, being tested) or *Nightly* (the newest development build, which may not work) - and the drive. The line under them says which AutoBleem version the channel would install, and what is on the stick now: *a fresh install* (the stick's other files are left alone) or *AutoBleem <version> is on this stick: it will be updated* (games, saves, memory cards and settings stay). **Formatting is optional**: the **Format** button next to the drive is only for a new stick with no file system, one the console cannot read (NTFS, for example) or one you want to wipe - it asks first, then erases everything on the stick. Pick FAT32 there unless the console has the AutoBleem kernel. Tick what you want:
 - **Cover databases** - the box art and details of the PS1 library (ticked by default; about 300 MB).
 - **RetroArch** - RetroArch with its cores, the extra applications (Doom, Quake, Amiga, ...) and the libretro assets, for the other systems' games. Off by default; it can be added later by running the installer again.
 - **BIOS files** - the BIOS files the RetroArch cores need (needs RetroArch).
 - **Sample games** - a few free homebrew games so the shelf is not empty.
3. Press **Install** and wait. The progress bars and the log show every step; the stick is named `SONY` at the end, and `UpdateRoms` is put on it (see chapter 5).
4. Take the stick out safely, plug it into the console's **second USB port** (the right one, player 2) and switch the console on. AutoBleem starts instead of the stock menu.

Switching on and off. With the stick in, the console boots, blinks its light for a few seconds (AutoBleem is being picked up) and then goes to standby before anything is shown - that is the console's own way of staging an update, which is how AutoBleem gets to run. Press **Power** once and the launcher comes up. *Power off* in the system menu, or the console's Power button, puts the console into **AutoBleem's standby**: the stick is disconnected first, then the light turns **red** - the sign that AutoBleem is working as intended - and the next press of Power brings the launcher straight back, in a few seconds. **While the light is red the stick can be pulled** and put into a PC without Windows asking to check it; put it back before pressing Power. Unplugging the console's power goes through the boot standby again next time.

To **update** a stick, run the installer over it again (the button says *Update*): it installs the chosen channel's newest version, and your games, saves, settings and RetroArch content stay; only

AutoBleem's own files are replaced. A console with the AutoBleem kernel and WiFi can also update itself (section 3.11).

Upgrading from AutoBleem 1.0 (or AutoBleem-NG). The same installer, on the same stick - no format:

1. Copy the whole stick to the PC first, as a backup.
2. Plug it in, start `AutoBleemInstaller.exe`, pick the channel and the drive. Do **not** press Format.
3. Press **Install**. The installer brings the old layout up to date: RetroArch, the ROMs and the themes move to where AutoBleem 2 looks for them, and the old ROM folders (`nes`, `snes`, ...) are renamed to RetroArch's system names. Games, save states, memory cards, ROMs and RetroArch's saves stay; `config.ini` is kept, with the PS1 emulator set to pcsx-abnxt.
4. Put the stick back into the console. The first scan rebuilds the game list and the playlists (UpdateRoms on the PC fetches box art for the RetroArch games, chapter 5).

The stock console has no clock and no network: dates are only shown after the AutoBleem kernel is installed (chapter 6), and box art for RetroArch games comes from UpdateRoms on the PC (chapter 5).

Games go into the `Games` folder of the stick, one folder per game - see section 3.9 for the layout.

2.2 Raspberry Pi

AutoBleem turns a Pi into a small console: it boots straight into the launcher, with no desktop. Two ready images are on the site - 32-bit and 64-bit - plus a tarball for an existing Raspberry Pi OS Lite.

Model	32-bit image	64-bit image	Notes
Raspberry Pi 5	yes	yes	
Raspberry Pi 4 Model B, Pi 400	yes	yes	
Raspberry Pi 3 Model B / B+ / A+	yes	yes	fine for the launcher and PS1
Raspberry Pi Zero 2 W	yes	yes	512 MB of RAM: PS1 runs, the heavier RetroArch cores do not
Raspberry Pi 2 Model B	yes	v1.2 only	slow for anything 3D
Raspberry Pi 1, Zero, Zero W	no	no	ARMv6 - neither image runs

The **32-bit image is the recommended one** for PS1 games: `pcsx-ab`'s fast ARM recompiler is 32-bit only, so the 64-bit build runs PS1 games slower. The 64-bit image has the larger set of RetroArch cores.

With Raspberry Pi Imager:

1. Install Raspberry Pi Imager (raspberrypi.com/software). In *Choose OS* pick *Use custom* and the `autobleem-<version>-rpi-armhf.img.xz` (32-bit) or `-arm64.img.xz` (64-bit) you downloaded - or add a repository URL in the app's settings and pick AutoBleem from the list. There is one per

channel: `https://autobleem.retromenele.pl/rpi-imager/os_list.json` for the latest release, `.../os_list-testing.json` for the build being tested and `.../os_list-nightly.json` for the newest development build. The download page's Raspberry Pi tab lists the ones that exist, with a *Copy* button each.

2. Use Imager's customisation screen (the gear, or the question after *Next*) to set the ****user name and password, the WiFi network and country, and enable SSH****. AutoBleem needs a network on its first boot.
3. Write the card, put it in the Pi with a screen and a keyboard or pad connected, and power it on.

The first boot takes 5 to 25 minutes and shows what it does on the screen. Without a network it asks for one (a WiFi list, the password), then asks whether to install RetroArch (a minute without an answer means yes), grows the system partition, makes the `AUTOBLEEM` data partition from the rest of the card, installs RetroArch and its cores, the BIOS packs and the sample games, and reboots into the launcher.

The answers can be given in advance in `autobleem.txt` on the card's boot partition (editable on any PC before the first boot):

Key	Default	Meaning
<code>root_gib</code>	8	The system partition's size in GiB; the rest becomes the games partition.
<code>hdmi_mode</code>	1920x1080@60	The screen mode for the whole boot (<code>1280x720@60</code> for an older TV).
<code>retroarch</code>	(asked)	<code>yes / no</code> - RetroArch and the other systems, or PS1 only.
<code>thumbnails</code>	none	<code>boxarts</code> mirrors the whole box-art set for offline covers (~9000 files).
<code>bios</code> , <code>downloads</code> , <code>samples</code>	yes	Set to <code>no</code> to skip the BIOS packs, every download, or the sample games.

On an existing Raspberry Pi OS Lite (Bookworm or Trixie): copy `autobleem-rpi.tar.gz` (or the arm64 one) to the Pi, unpack it and run `sudo bash install.sh`. It asks the same questions, makes the data partition by shrinking the root at the next boot (`--shrink-root <GiB>`), and puts the launcher on the first console.

After the install the card's **AUTOBLEEM partition** (exFAT) is what you fill: pull the card and open it on any PC, or copy over the network (SSH is on). `Games/` for PS1 games, `RetroArch/roms/<system>/` for the other systems, `System/Bios/` for the PS1 BIOS (section 3.10), `Themes/` for themes.

2.3 PC USB stick

The same appliance for any PC that boots from USB - a 32-bit system, so old machines work too:

1. Write the image to a stick of 8 GB or more. **On Windows** use **AutoBleemFlasher** (PC panel of the site - unzip it and run `AutoBleemFlasher.exe` ; it asks for administrator rights, since it writes a whole disk): pick the channel and the stick, press *Write* and confirm twice - it downloads the image, checks it, writes it and reads it back. Only USB sticks and SD cards are offered, never the disk Windows runs from, and **everything on the chosen stick is erased**. On Linux or macOS,

download `autobleem-<version>-pcusb-i386.img.xz` from the PC panel and write it with `xzcat autobleem-*.img.xz | sudo dd of=/dev/sdX bs=4M status=progress` (or balenaEtcher).

2. Boot the PC from the stick (the boot menu key of your PC - F12, F8, Esc...). Both BIOS and UEFI boot work; **Secure Boot must be off**.
3. The first boot is the Pi's: a network question if there is no cable, the RetroArch question, then the install - about eight minutes with a wired network - and a reboot into the launcher.

The stick then has an `AUTOBLEEM` partition for your games, visible on Windows 10 (1903 and newer) as a second drive when you plug the stick into a running PC. `autobleem.txt` is on the first partition, with the same keys as on the Pi (no `hdmi_mode` - the PC uses the screen's native mode).

2.4 Windows

AutoBleem as a Windows program: full screen, the emulators and RetroArch started as programs.

1. Download **AutoBleemSetup-<version>.exe** and run it. It installs per user, without administrator rights: the program under `%LOCALAPPDATA%\Programs\AutoBleem`, the data (games, settings, themes, RetroArch) in a folder you choose - `Documents\AutoBleem` by default.
2. Tick the components - the cover databases, RetroArch (the official Windows build and its cores), the BIOS files, the sample games - and let the setup helper download them.
3. Start AutoBleem from the Start Menu or the Desktop. On a PC the keyboard works as a pad (section 3.2).

Running a newer setup over it updates the program and keeps the data folder. The launcher also checks the site once a day and offers an update when there is one (section 3.11).

3. Using AutoBleem

3.1 The launcher

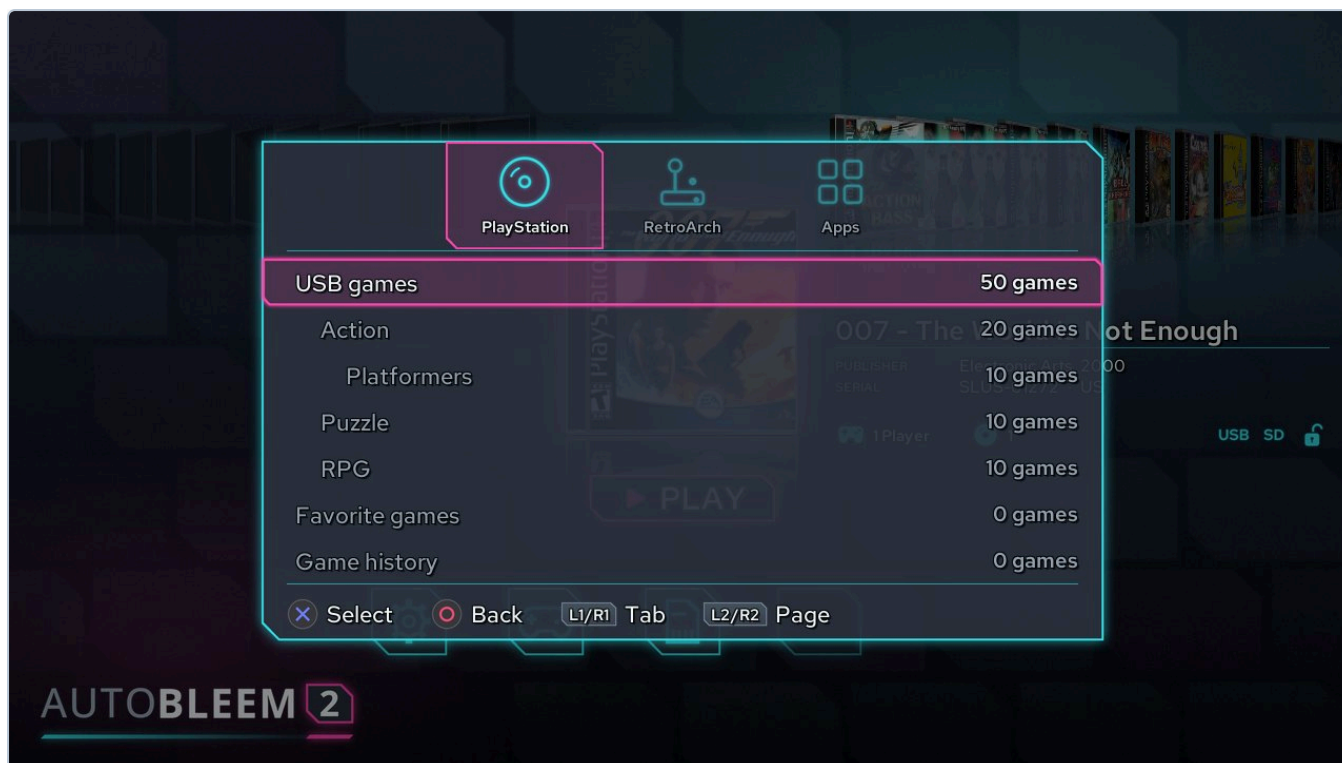
The launcher opens on the shelf: the covers of the current set, the selected one in the middle with a soft reflection under it, its details next to it as a compact grid - publisher, year, serial, region, players, when it was last played (a fact a game does not have is left out) - and a play button. The default look is the **ab2.0.0** theme; a fresh install and an update that brings it switch to it once. The hint bar at the bottom has two lines of four slots. The first line is what the buttons do for the selected game (play, play in RetroArch, open the icon row, the Quick menu); the second always shows Select (the set), Start (a random game), Triangle (the guide) and L2 + R2 (the system menu), dimmed in a state where one does nothing. A scan of the games folder runs in the background on every start; while it runs, a bubble at the top right shows its progress, and new games appear on the shelf as they are found.

A fresh install has no games yet: instead of an empty shelf the launcher shows a welcome card - *Hi, and welcome to AutoBleem!* - telling you to drop games into **Games** and press *Re-scan games*, and naming the place for your platform: on your stick (PlayStation Classic, PC stick), on your SD card (Raspberry Pi) or in your AutoBleem folder (Windows). The card goes as soon as a scan finds the first game.

Notifications appear as bubbles at the top right: the scan's progress, the name of the set you switched to (*Showing: ...*, for as long as Options → *Notification timeout* says), a low pad battery, a note after a crash, a scanner processor at work, and the Store's download in progress. The download's bubble shows its speed and the time left, e.g. **1.4 MB/s · 0:42**.

The channel tag. A build that is not a final release shows a small tag under the pad battery plate in the top-left corner: a chip with the channel - **ALPHA**, **BETA** or **RC** for a pre-release, **TESTING** for any other pre-release, **NIGHTLY** for a nightly build, **DEV** for a build made by hand - and the short version beside it (for **DEV**, the commit it was built from). A release shows no tag.

A wireless pad with a battery reading - the console, a Pi or the PC stick, not Windows - shows as a small icon with its percentage, stacked from the top-left corner on its own plate. A pad matched to Player 1 or Player 2 (following the Swap Player 1 / Player 2 option) is tagged P1/P2; an unmatched pad, or a third one, has no tag. When a pad's battery runs low, a notification line reports it once, by name and percentage.



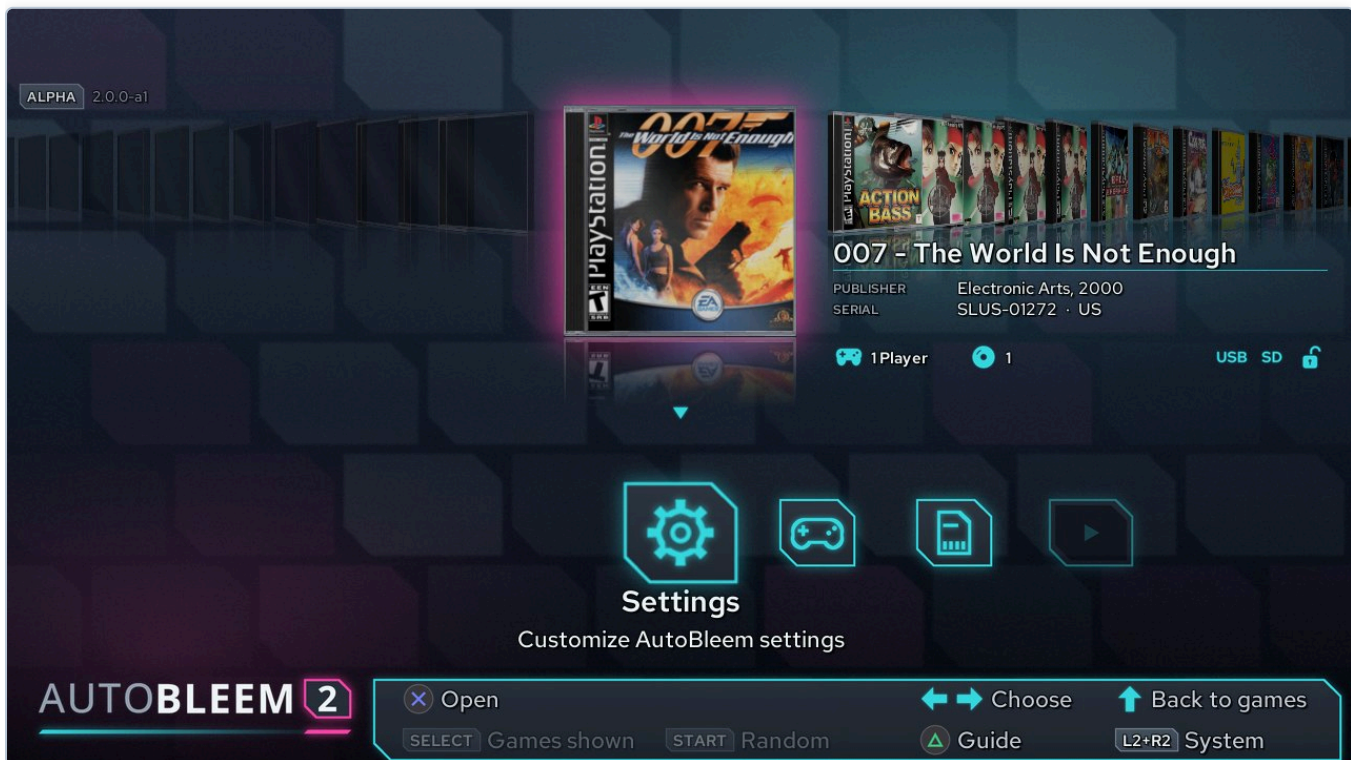
The set picker: three tabs, and the groups of the current one with their game counts

3.2 Controls

Button	On the shelf
Left / Right	Previous / next game. Holding scrolls on.
L1 / R1	Jump to the previous / next first letter of the titles.
Cross	Start the selected game (a PS1 game in the PS1 emulator; a RetroArch game in its core; an App after its read-me).
Square	Start the selected PS1 game in RetroArch instead.
Triangle	The button guide.
Start	A random game from the current set.
Select	The set picker: PlayStation / RetroArch / Apps tabs (L1 / R1), the groups of the tab (Up / Down, L2 / R2 a page), Cross picks.
Up	The Quick menu (section 3.4).
Down	Open the icon row under the game (Settings, Game, Memory Card, Resume). Up closes it.
L2 + R2	The system menu (section 3.5).

With a keyboard (a PC without a pad, or a USB keyboard on the console, a Pi or the PC stick) the keys stand in: **Arrow keys** = d-pad, **Enter** = Cross, **Esc or Backspace** = Circle, **Tab** = Triangle, **Space** = Square, **F1 / F2** = Select / Start, **Page Up / Page Down** = L1 / R1, **Home / End** = L2 / R2, **F10** = the system menu. On a development machine Esc closes the program and Space is Start.

In every list and menu: Up / Down move, **L2 / R2 turn pages**, L1 / R1 jump to the first / last row, **Cross selects**, **Circle goes back**. A screen with settings saves them when you leave it with Circle.



The icon row under the selected game

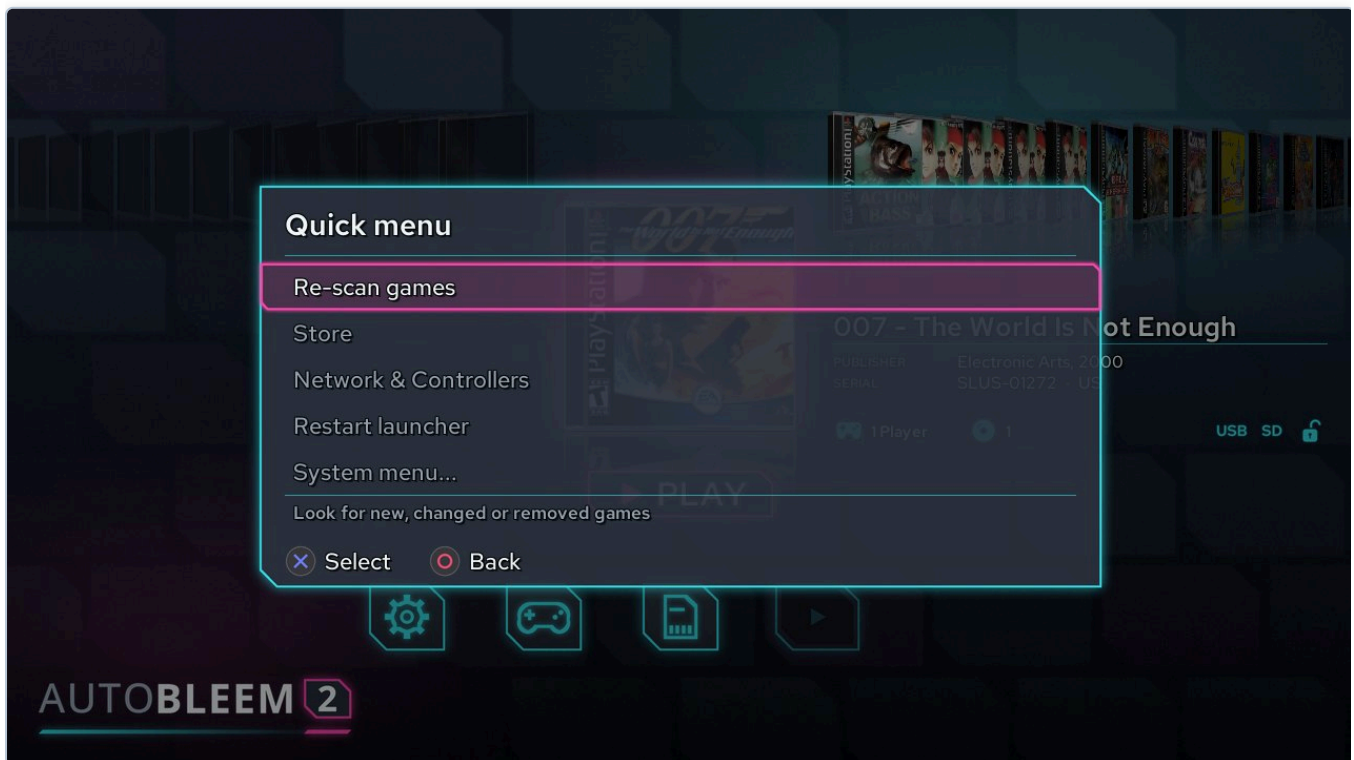
3.3 The sets

Select opens the set picker. The PlayStation tab lists, on a PlayStation Classic, *All games* and *Internal games* (the console's built-in twenty), then *USB games* (everything in `Games/`) and every folder you made under it (a game in a sub-folder belongs to that group), then *Favorite games*, *Game history* and, when any game is flagged as one, *Lightgun games*. On a Raspberry Pi, a PC stick and Windows there are no internal games, so the list starts at *USB games*, which is the whole library. The RetroArch tab (only where RetroArch is installed) lists one group per system that has games, plus RetroArch's own Favorites and History. The Apps tab groups applications by type: *All apps*, then *Games*, *Emulators*, *Tools*, *Media* and *Other* (the category is set in each app's `app.ini` file). Each row shows how many items it holds; a group with none opens on an empty shelf with the icon row showing Settings only. The footer shows the keys: L1 / R1 for the tabs, L2 / R2 for a page, Cross to pick, Circle for *Back*.

3.4 The Quick menu

Up in the launcher, or the **gear icon** in the icon row (where Settings / Game / Memory Card / Resume are): the Quick menu for actions you reach for from the carousel. A short list: *Re-scan games* (starts a scan now), *Store* (browse and install games, apps and extensions), *Network & Controllers* (only where an installed extension provides the `network` entry - PSC-Bios on the console, a Pi and the PC stick: Wi-Fi, Bluetooth pairing, the gamepad mapping wizard - see section 6; greyed out with "enable it in Extensions" when that extension is disabled - Cross opens the Extensions list), *Restart launcher* (closes AutoBleem and starts it again; on the console, a Pi and the PC stick only), and *System menu...* (everything else: Options, Game Manager, Power off and more - the full menu below). Up / Down move

(wrapping), Cross picks, Circle back. Each item has a one-line description on its row. Apart from the Store and Restart launcher, every item is also in the system menu.

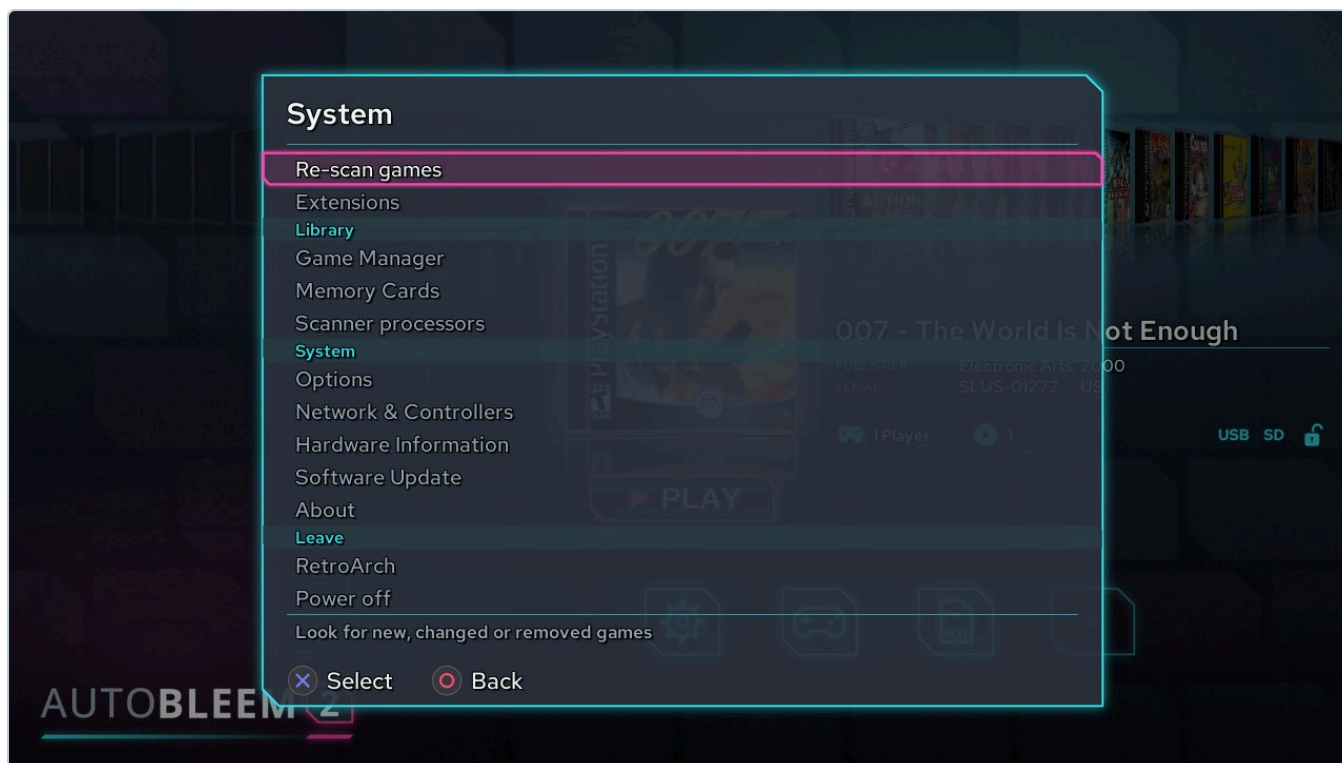


The Quick menu

3.5 The system menu

L2 + R2 (together, in either order) opens the system menu over the shelf. Each row has a one-line description, and the menu is grouped into sections:

Section	Item	What it does
(top)	Re-scan games	Looks for new, changed or removed games now (the scan also watches the folder by itself).
	Extensions	The extensions on the stick - the AutoBleem Store and others (section 3.12).
Library	Game Manager	The PS1 games as a list with their folders: delete a game, flush the covers. Disabled while a scan is running.
	Memory Cards	Your memory card sets (section 3.7).
	Scanner processors	The programs every scan runs first - their order, on or off (section 3.13). Disabled while a scan is running.
System	Options	AutoBleem's settings (section 3.6).
	Network & Controllers	Only where an installed extension provides the <code>network</code> entry (Provides=network in its <code>extension.ini</code> - PSC-Bios on the console, a Pi and the PC stick) - Wi-Fi, Bluetooth controller pairing, DualShock 3 setup, and the gamepad mapping wizard - see chapter 6. When that extension is installed but disabled, this item stays greyed with a note "enable it in Extensions" - Cross opens the Extensions list at it.
	Hardware Information	The machine's facts: system, CPU, storage, network interfaces, time zone, display, the pads and their mappings - the same page on every platform (section 4.2).
	Software Update	(Raspberry Pi and PC) Check the site for a newer AutoBleem or RetroArch now; the row says <i>Update available</i> when the launcher already knows of one.
	About	Credits and licence.
Leave	RetroArch	(Only where RetroArch is installed.) Leaves the launcher for RetroArch's own menu. Closing RetroArch comes back.
	Power off	After a confirmation: on the console AutoBleem's standby - the stick disconnected, the light red, Power brings the launcher back (section 2.1); on a Pi or PC the machine shuts down.



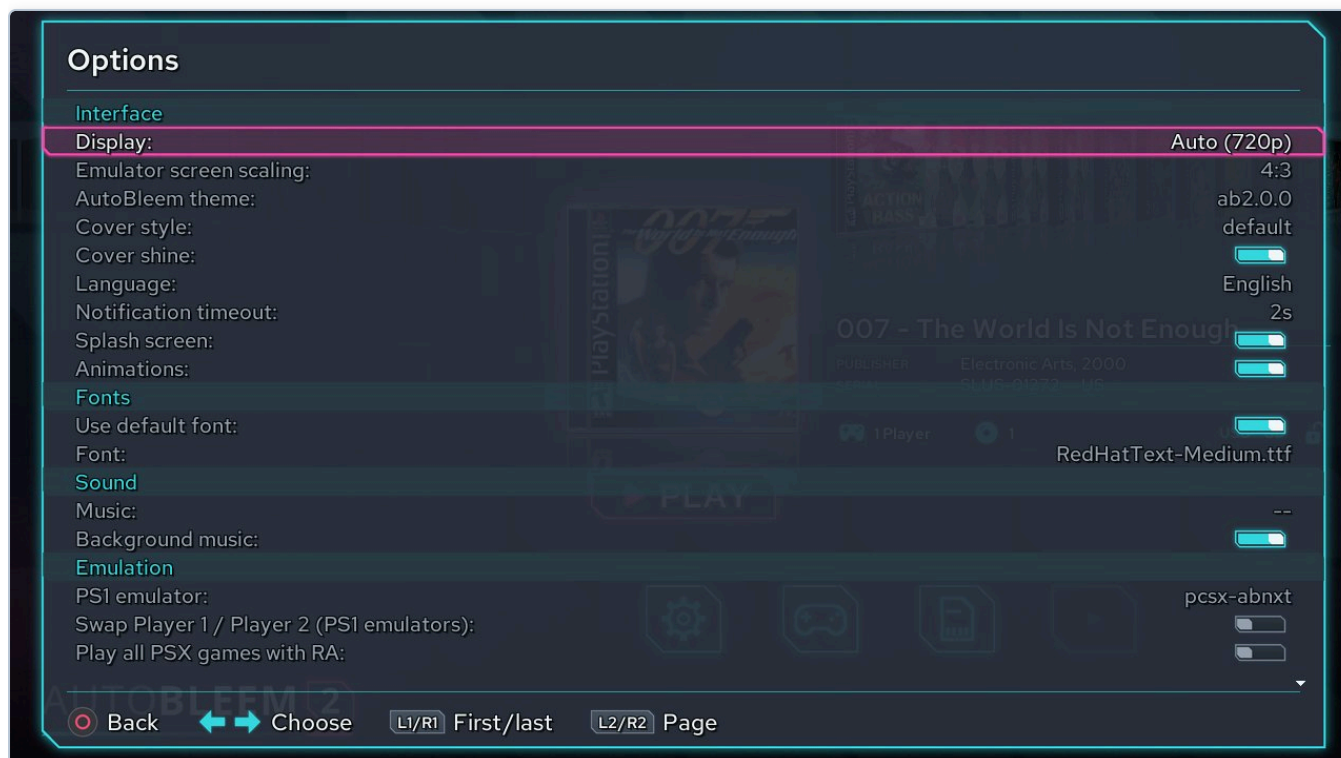
The system menu

3.6 Options

The settings are in groups, each under a heading; Up / Down move between rows, Left / Right change a value (a tap is one step, a hold scrolls), L1 / R1 jump to the first / last row, L2 / R2 page, Circle leaves and saves. Every change is applied at once. On/off values read **ON** / **OFF**.

Group / setting	What it does
Interface: Display	The screen's resolution, for the launcher and the PS1 emulator: <i>Auto</i> (the screen's own mode, shown as <i>Auto (1920x1080)</i>) or any mode the screen lists; the console offers 720p and 1080p. A new mode is asked about: <i>Keep this display mode?</i> - if you do not confirm, it goes back after a countdown. Not on a development window.
Emulator screen scaling	How the PS1 emulator fits a game's picture to the screen: <i>1x1</i> (the PlayStation's own pixels), <i>2x (integer)</i> , <i>4:3, 4:3 (integer)</i> or <i>Full screen</i> . Integer scaling uses whole multiples only (the sharpest). It replaces the old Widescreen switch; the classic <code>pcsx-ab</code> and RetroArch know only full screen and 4:3.
AutoBleem theme	The look. Themes live in <code>Themes/</code> ; a theme zip dropped there is listed by name too. The themes AutoBleem ships are refreshed with every update - to customise one, copy it under a new name first. The default is ab2.0.0 .
Cover style	The jewel-case frame drawn around PS1 covers.
Cover shine	A shine that crosses the selected cover when the shelf comes to rest.
Language	The launcher's language, applied at once (17 languages).
Notification timeout	How long the information bubbles ("Showing: ...", the scan's summary) stay, 0 to 20 seconds; 0 shows <i>Off</i> . Errors keep their own fixed time.
Splash screen	The AutoBleem picture when the launcher starts; off goes straight to the shelf.
Animations	The movement between screens; off makes every screen change instant.
Fonts: Use default font	The launcher uses its default font (Red Hat Text), or - switched off - the font chosen below.
Font	Any <code>.ttf / .otf</code> from <code>resources/fonts</code> , <code>RetroArch/fonts</code> or the theme's folder; the row names the font in use.
Sound: Music, Background music	Which track plays under the launcher (the theme's, or a file from <code>resources/music</code>), and whether one plays at all.
Emulation: PS1 emulator	<code>pcsx-abnxt</code> (the default: current PCSX-ReARMed with AutoBleem's additions) or <code>pcsx-ab</code> (the classic). A resume point saved by one continues in the other, unless the game ran without a BIOS file.
Swap Player 1 / Player 2 (PS1 emulators)	Swaps which of the first two pads is Player 1 and which is Player 2, in both PS1 emulators (<code>pcsx-abnxt</code> and the classic <code>pcsx-ab</code>). It only takes effect with two or more pads connected; with one pad, play is always Player 1. RetroArch is not affected.
Play all PSX games with RA, Update RA config, Persist RetroArch config	(Only where RetroArch is installed.) Every PS1 game starts in RetroArch's PS1 core; AutoBleem writes its settings into RetroArch's config when it starts a game there; a change made in RetroArch's own menu is kept when RetroArch quits.
Library: Show internal games	The console's built-in games in the PlayStation lists (PlayStation Classic only).
Fetch box art online	The scan fetches missing covers from libreto's servers (Raspberry Pi, PC, Windows).

Group / setting	What it does
Updates	The update channel: <code>release</code> (the tested version), <code>testing</code> (the next version, being tested), <code>nightly</code> (the newest development build) or <code>off</code> . The default follows the version installed. Not shown on a development host.
Diagnostics: Keep logs on the stick	Keep every log on the stick from the next start, not only after a crash (chapter 7).
Show performance	An overlay in the bottom-left corner: frame rate, CPU load, threads and memory; the emulator shows its FPS and CPU in the game too.



The options, in groups

3.7 A game's settings

With a game selected, **Down** opens its icon row: **Settings** (the options above), **Game** (the game's own settings), **Memory Card** (its memory card) and **Resume** (its save states). Cross opens the one under the cursor.

The **game editor** shows the game's details on the right (title, publisher, year, players, folder, memory card) and its settings on the left, in four groups:

- **Game:** *Favorite* (in the Favorite games group), *Lightgun game* and *Play using RA* (only where RetroArch is installed: a light-gun game joins the Lightgun group and always runs in RetroArch, whose PS1 core has the GunCon; *Play using RA* runs this game in RetroArch), *Lock data* (the scanner leaves the game's title, serial and disc list as you set them).
- **Display:** *Resolution* (1x or 2x, on the built-in GPU), *Remove seams* (only with 2x), *Dithering* (Off, On, Always), *Smoothing*, the *Filter* - how the picture is scaled: Nearest (plain pixels), Linear (smoothed), Sharp or Sharp (simple) (crisp pixels without shimmer), Quilez, or the CRT filters CRT (fast) and CRT-Pi (they draw their own scanlines, so the scanline rows grey out) - and *Scanlines*

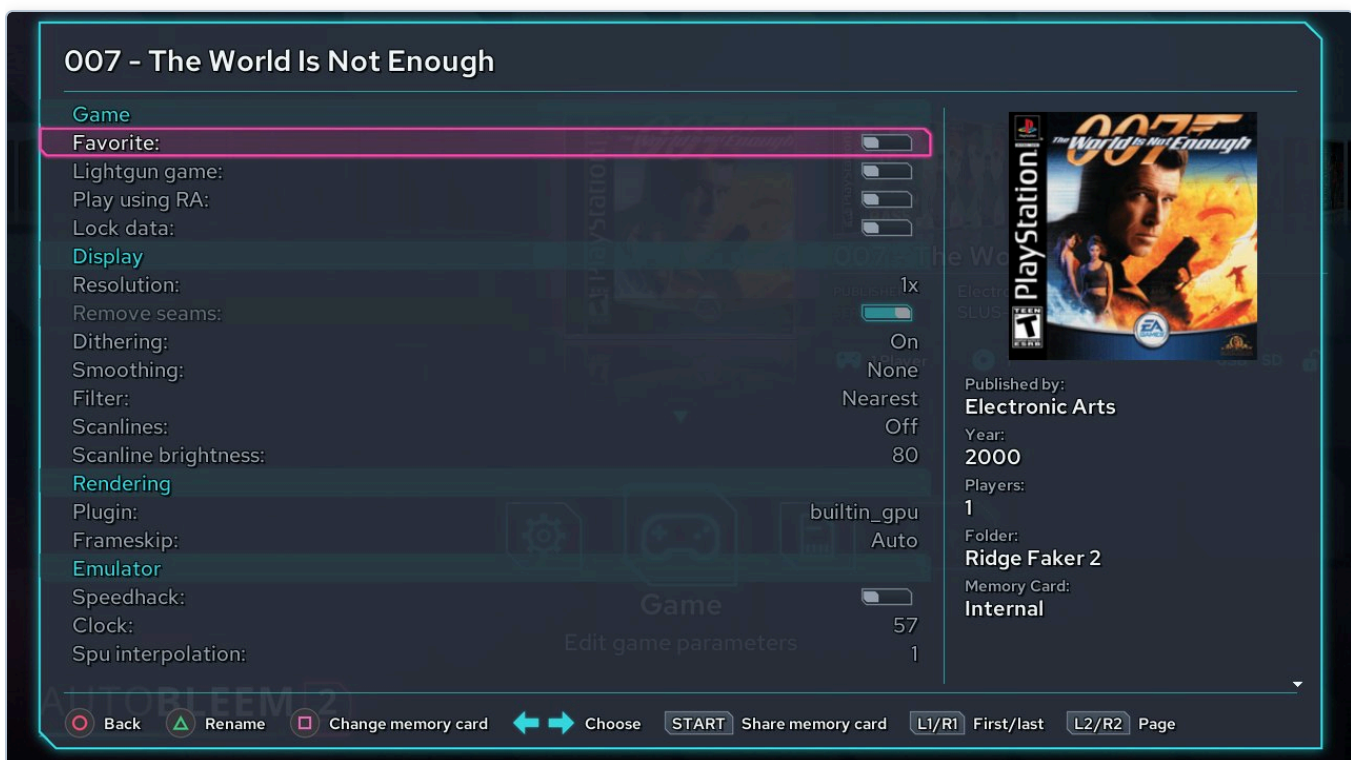
with their *Scanline brightness*. Resolution, remove seams, dithering, smoothing and the filters other than Linear and Nearest are for `pcsx-abnxt` ; the classic `pcsx-ab` and RetroArch show the rest as Nearest.

- **Rendering:** the GPU *Plugin* and the *Frameskip* (Auto, Off, 1 to 3).
- **Emulator:** *Speedhack*, the CPU *Clock*, *Spu interpolation*, the *Boot logo* (off skips the BIOS shell - for a homebrew disc whose custom logo breaks the boot), and with `pcsx-abnxt` the *Sony hacks* toggle.

The picture shape and the screen's resolution are global (Options → *Emulator screen scaling* and *Display*). A game with no title in its data is shown under its folder's name.

Triangle renames the game, Square changes its memory card, Start shares a new card. Circle saves and leaves.

Settings saved in the emulator. The emulator's own menu has *Save settings for this game*. Once a game has settings saved there, they are the ones it plays with, and the game editor shows its Display, Rendering and Emulator rows greyed out, with those values, under the heading *Saved in the emulator*. To go back to the game editor's settings, pick **Unlock the settings** and confirm: this deletes the settings the emulator saved, and the rows can be changed again. Both emulators, `pcsx-ab` and `pcsx-abnxt` , read and write the same saved settings.



The game editor

3.7 Memory cards and save states

Every PS1 game has its own memory card by default (kept with its save states in `Games/!SaveStates/<game folder>/`). **Memory Cards** in the system menu manages **shared sets** - a card several games use, kept in `Games/!MemCards/` : create one (Square, with the on-screen keyboard), rename (Cross), delete (Triangle). A game is put on a set with *Change memory card* in its editor, or from its Memory Card icon.

The **memory card editor** (the Memory Card icon) shows the game's card and a second card side by side, with every save's icon and title: copy a save between the two (Square), delete one (Triangle), defragment a card (Select). Start swaps the card on the right for another set.



The memory card editor

Resume points: when you leave a PS1 game with the console's Reset button (or the emulator's menu on a Pi or PC), AutoBleem keeps a save state of where you were and offers it under the **Resume** icon - four slots, shown as framed cards, each with a picture of the moment, its slot number and date; the newest is marked **NEWEST** and an unused slot says *No resume point*. Cross continues from the slot, Triangle deletes it. A game with a resume point shows a small picture on its Resume icon; a game without any has the Resume icon greyed out. While the resume point is being written on the way out of a game, the emulator shows *Please wait...*

3.8 Starting games, RetroArch and Apps

Cross starts the selected game. A PS1 game runs in the chosen PS1 emulator (section 3.6), full screen, until you leave it - on the console with the front **Reset** button (back to the launcher with a resume point; it works from inside the in-game menu too) or **Power** (the console turns off); on a Pi or PC through the emulator's in-game menu (below). **Square** starts a PS1 game in RetroArch instead.

The in-game menu (pcsx-abnxt). Press the menu button - the pad's Home, **Select + Start** on a pad without one, or **Esc** on a keyboard - and the game stops behind a menu with the game's last picture.

Holding the menu button for 2 seconds is the same as Reset: it leaves the game. L1 / R1 switch between its three tabs, and the menu opens on the tab and row it was left on:

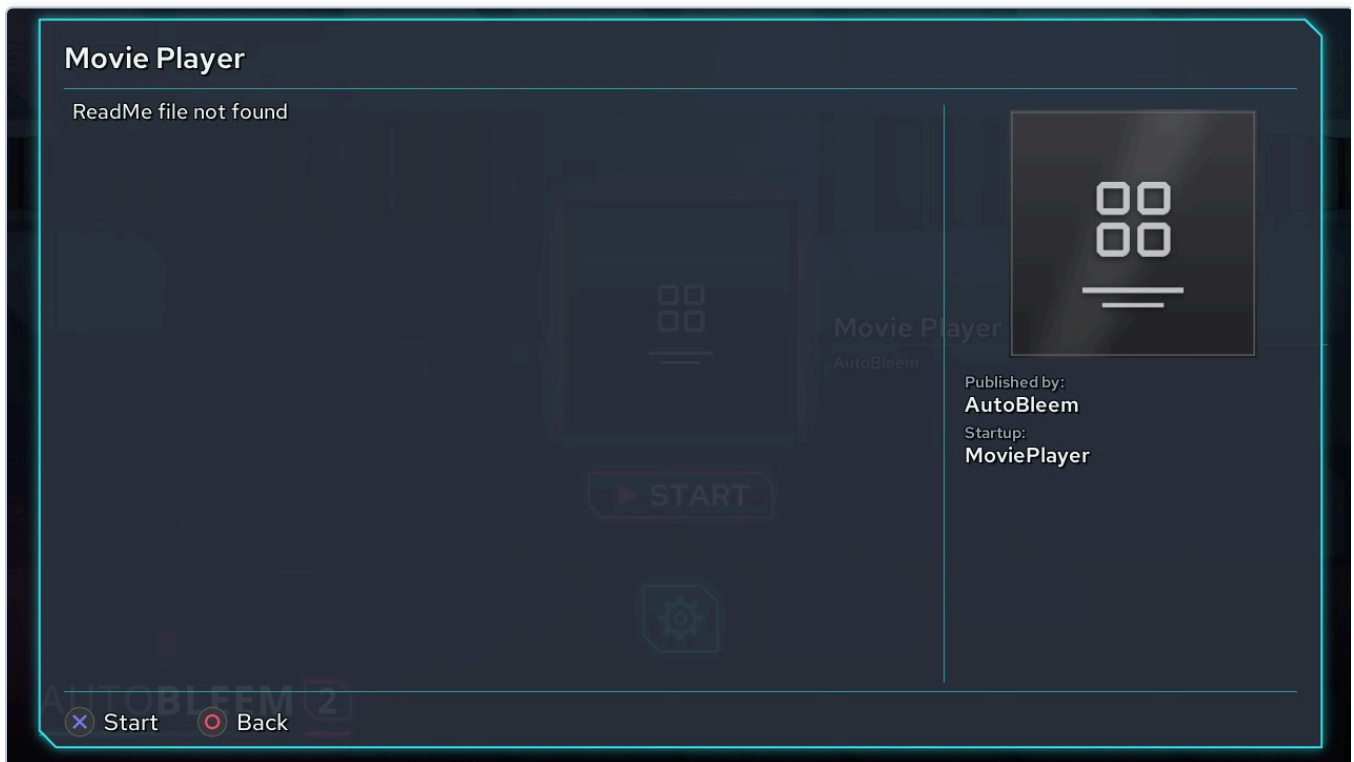
- **Game:** Resume game; under Saves: Quick save, Quick load and Load autosave (the game as it was up to 30 seconds ago - the emulator saves it in memory while you play); under CD disc: Change disc and Reset game (starts it over); Save settings for this game (see section 3.7), PCSX menu (PCSX-ReARMed's own pages: options, cheats, About) and Exit (back to AutoBleem).

- **Picture:** *Display* (the screen's resolution - on the console it is chosen in Options and only shown here), *Resolution* (1x or 2x), *Remove seams*, *Dithering*, *Scaling*, *Smoothing*, *Filter*, *Scanlines* and *Scanline brightness*. Each row has a help line on the right. CRT-Pi is too heavy for the console at 1080p. A row that does not apply is greyed, and its help says why.
- **Controllers:** *Controller 1* and *Controller 2*: standard (digital), analog (DualShock), a gun or none; takes effect when the game goes on.

The menu is drawn in the launcher's ab2.0.0 look, with the pad batteries and the last quick save's picture.

A **RetroArch** game starts in RetroArch with the core the launcher chose for its system; *Close Content* or *Quit RetroArch* in its menu comes back to the launcher. The RetroArch item in the system menu opens RetroArch's own menu (XMB) with nothing loaded, for its settings and its own content lists.

An **App** (the Apps set: the console tools, and on a console the extra applications RetroArch's package brings - Doom, Quake, Amiga, ...) shows its read-me first; Cross starts it, Circle goes back.



An App's read-me before it starts

3.9 Adding games

PS1 games go into the `Games` folder, **one folder per game**, named after the game:

```
Games/
  Crash Bandicoot/      Crash Bandicoot.cue + Crash Bandicoot.bin
  Final Fantasy VII/    Final Fantasy VII (Disc 1).chd, (Disc 2).chd, (Disc 3).chd
  Platformers/         a folder of games: a group of its own in the set pick
    Klonoa/             Klonoa.pbp
```

- Formats: `.cue` + `.bin` (or `.img`), `.pbp`, `.chd` (zstd too), `.ecm` (decoded by the scan), `.iso`. A zipped game works too: the **Unzip** processor unpacks it before the scan (section 3.13).
- A multi-disc game is one folder with every disc in it; folders named `Game (Disc 1)`, `Game (Disc 2)` ... are merged into one `Game` folder by the scan.
- Games dropped straight into `Games/` (loose files) are sorted into folders by the scan.
- A **cover** is a PNG next to the game's image, named like it. Without one, the box art comes from the cover databases, or - with RetroArch installed - from libretro's thumbnail set; on a Pi, a PC or Windows a missing one is fetched online (Options → *Fetch box art online*).
- The scan reads each disc's serial number and takes the title, publisher, year, players and region from RetroArch's PlayStation database or the cover databases. Change anything in the game editor and tick *Lock data* to keep it.

Other systems go under `RetroArch/roms/`, **one folder per system, named as RetroArch's databases are** (the folder is made for you): `Nintendo - Nintendo Entertainment System`, `Nintendo - Super Nintendo Entertainment System`, `Sega - Mega Drive - Genesis`, `Nintendo - Game Boy Advance`, `FBNeo - Arcade Games (or Arcade)`, ... ROMs may stay zipped. On a Pi, a PC or Windows the scan reads them by itself and writes RetroArch's play lists; on a console stick, run **UpdateRoms** on the PC (chapter 5).

Apps go under `Apps/<name>/` with an `app.ini` (the name, the icon, what to run) and a `run.sh`.

Themes go under `Themes/<name>/` (`theme.json` and the images) - or drop the theme's zip into `Themes/`.

3.10 The PS1 BIOS

On a **PlayStation Classic** the emulator uses the console's own BIOS. On a **Raspberry Pi, a PC and Windows** put your own PS1 BIOS into `System/Bios/`: `romw.bin` (the US/European SCPH-5501/5502) and `romJP.bin` (the Japanese SCPH-5500). The installers fill them from the RetroArch BIOS packs unless your own files are already there. Without them the emulator runs on its built-in HLE BIOS, which many games tolerate and some do not.

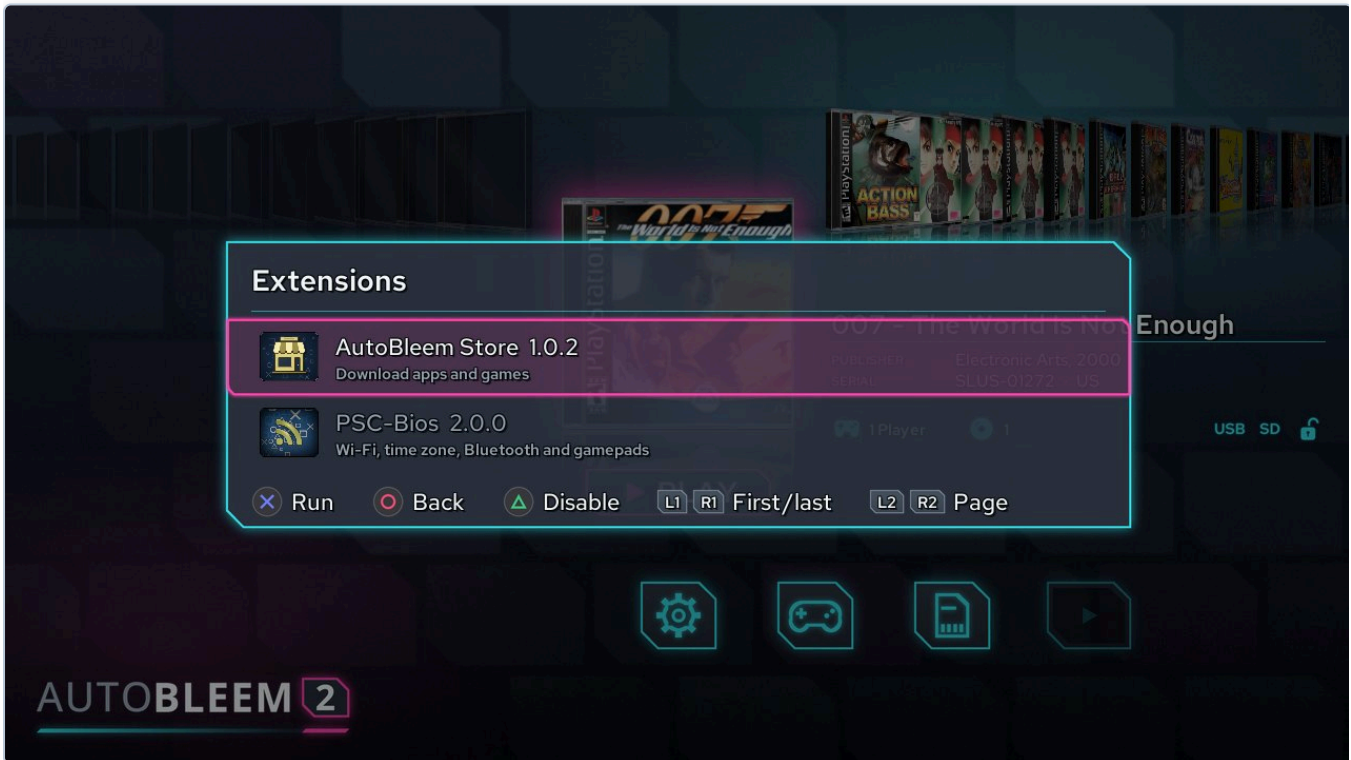
3.11 Updates

The launcher checks the site at start and once a day for a new version on its **channel** (Options → *Updates*: `release`, `testing`, `nightly` or `off`); *Software Update* in the system menu checks now. When there is one it asks: *Update now* downloads it, *Remind me tomorrow* and *Skip this version* are the other answers. Your games, saves and settings always stay; the launcher rescans once after an update.

- **Raspberry Pi, PC stick**: after the download the installer runs again with its first-boot progress screen (a newer RetroArch is offered too), then the launcher is back.
- **Windows**: the downloaded setup opens, installs, and starts the new launcher.
- **PlayStation Classic**: only a console with the **AutoBleem kernel** (ABFlashKit, section 6.2) and a **WiFi network** set up in PSC-Bios (section 6.1) checks - a stock console has no network, and the launcher does not look. After *Update now* the launcher closes, the AutoBleem picture stays on screen while the stick is updated (a few minutes), and the new launcher starts. Without a network *Software Update* says *Not connected*. Any console stick can also be updated from a PC with `AutoBleemInstaller.exe` (section 2.1).

3.12 Extensions and the AutoBleem Store

Extensions add their own screens to the launcher. They live in `Extensions/<name>/` on the stick (on a Raspberry Pi its data partition, on Windows the data folder); to install one, unpack its zip there. **L2 + R2** → **Extensions** lists them: Cross runs one, Triangle turns it off or on again. An extension that needs the network is not started without one, and one that stopped the launcher is turned off - the list says so.

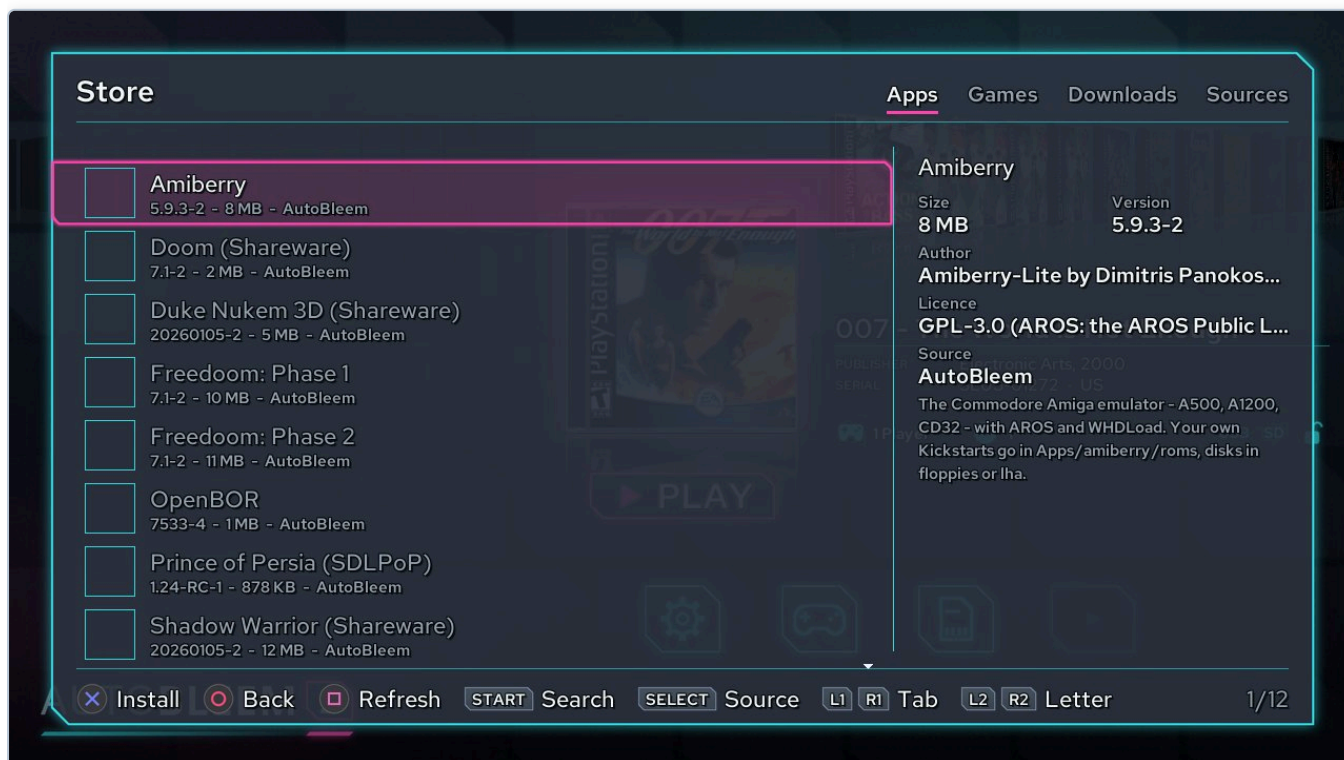


The Extensions list

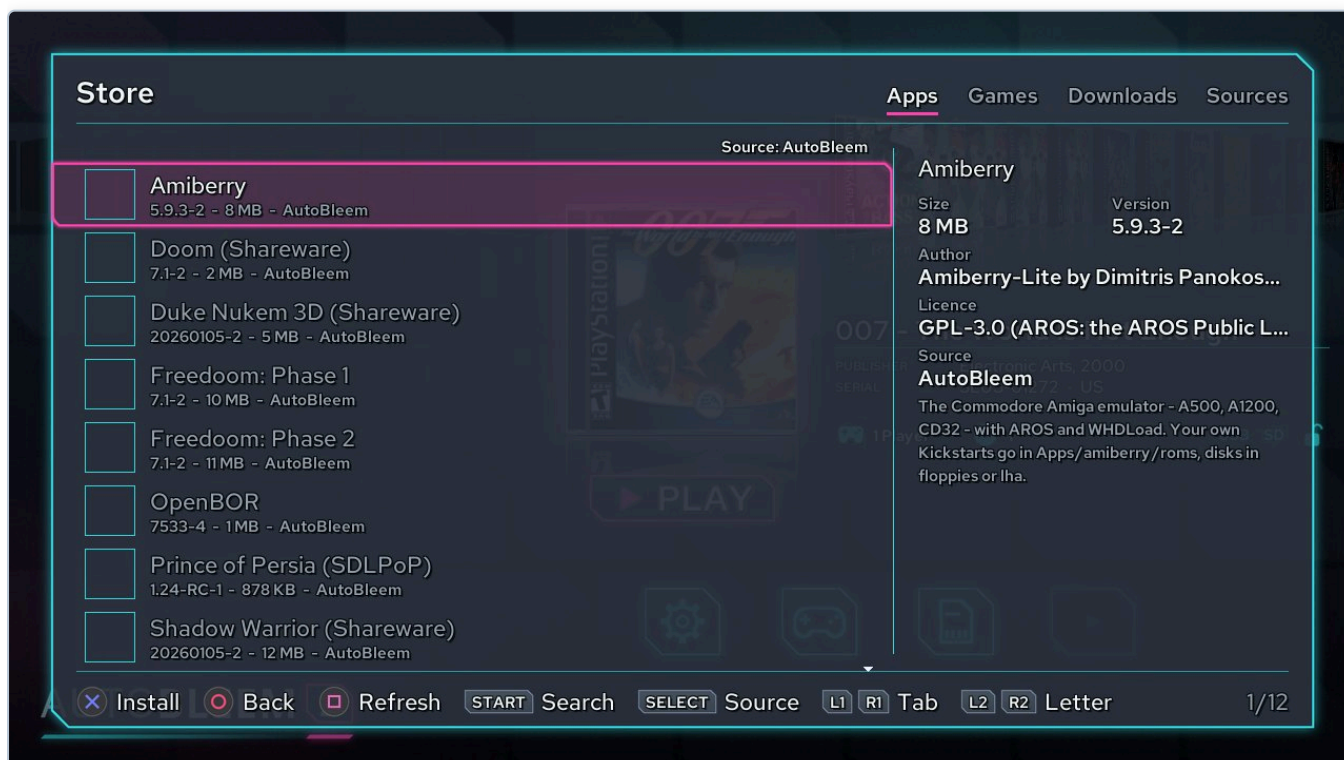
The AutoBleem Store is the first extension: Apps and games to install with one press, on every system AutoBleem runs on (a PlayStation Classic needs the AutoBleem kernel's WiFi). Its four tabs, L1 / R1 between them:

- **Apps** and **Games**: what the sources offer, each with its picture, version, size and source favicon. Installed items carry an *Installed* badge. Cross installs (or updates, or tries again after a failure, or cancels a download that is queued or running), Triangle removes what the Store installed, Square refreshes the lists. L2 / R2 jump by letter, **Select** shows one source at a time, **Start** searches the titles. The footer shows the keys that apply to the selected row. Item pictures are cached and can be retried if they fail to load.
- **Downloads**: what is downloading, waiting, failed or installed. The progress bar updates steadily, and while you are elsewhere in the launcher a bubble shows the running download with its speed and the time left (`1.4 MB/s · 0:42`). Downloads go on in the background, also after you leave the Store; starting a game or powering off only pauses them, and a stopped download resumes where it stopped. If the network drops, the item says *Waiting for the network* and carries on from where it stopped when the network is back (it gives up after 30 minutes). An installed game appears on the shelf after the next scan, with the Store's picture as its cover. Downloads over 2 GB work on all platforms, including 32-bit builds.
- **Sources**: where the lists come from - AutoBleem's own catalog, a TSV list dropped into `System/Extensions/store/sources/` , and the addresses you add with **Add a source URL**.

Each source shows its favicon in the list. Cross on one you added renames it, changes its address, switches it between `http://` and `https://`, or removes it.



The Store's Apps tab



A source's menu

What AutoBleem's catalog offers is also listed on the download site, <https://autobleem.retromenele.pl/store/>. **You are responsible for what the sources you add contain.**

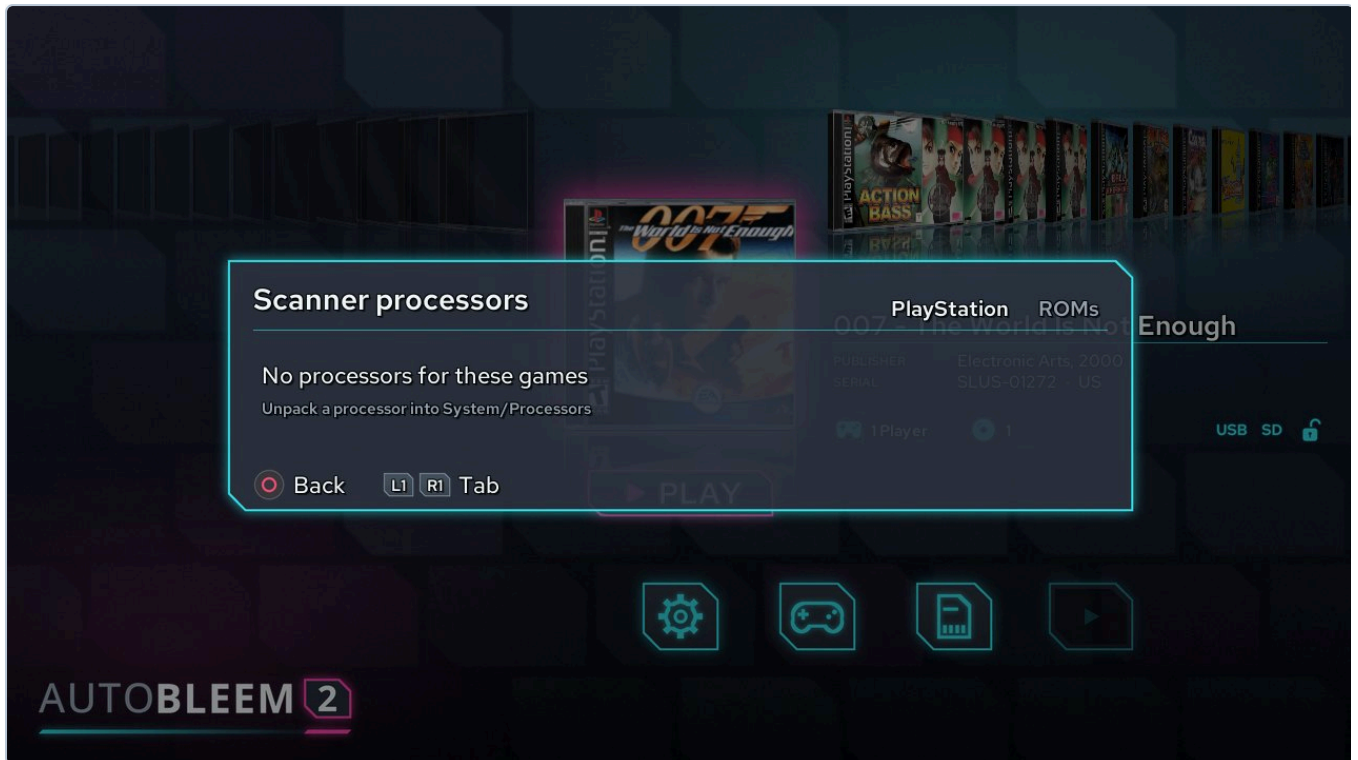
Your own games on your network: `abstored` , the Store's LAN server, serves a folder of PS1 games to the Store on the same network. It runs on any Linux machine - a Raspberry Pi, a home server - and only reads the folder. Start it with `abstored <games folder>` , open `http://<that machine>:8124/` in a browser to see what it serves and any problems it found, and add `http://<that machine>:8124/store.tsv` as a source. Ready programs for Linux and Windows are on the Store's page, in its **LAN server** tab; setting it up as a service is `INSTALL-linux.md` (`ext_store/server/` in the source). **LAN Share** (section 5.2) puts games and discs from a PC on such a server.

3.13 Scanner processors

Scanner processors are small programs that every scan runs before it reads your games. One can turn a format AutoBleem does not read into one it does - a zipped game, for example - or change a game's data, such as a translation patch. They live in `System/Processors/<name>/` on the stick (on a Raspberry Pi its data partition, on Windows the data folder); to install one, unpack its folder there. The next scan runs it.

- **Unzip comes with AutoBleem:** it unpacks zipped PS1 games in `Games/` before the scan reads them, and zipped ROMs one at a time (arcade sets stay zipped). Updating AutoBleem updates it too, and leaves it switched off if you switched it off.
- A processor that has already dealt with a game is not run on it again until the game changes.
- While a processor works, the bubble at the top right shows what it is doing; a warning or a failure appears on the line under it. `processors.log` in the logs folder has the details.
- Starting a game or RetroArch stops a processor that changes files; the next scan finishes its work.

L2 + R2 → **Scanner processors** shows them in the order they run, one tab for the PS1 games and one for the ROMs (L1 / R1). **Square** picks a processor up and Up / Down move it - the order matters: a processor that unpacks has to come before one that patches what was unpacked. **Cross** switches one off or on, **Triangle** has it look at every game again at the next scan, **Circle** goes back and starts a scan if you changed anything. A processor built for another machine stays on the list, greyed.



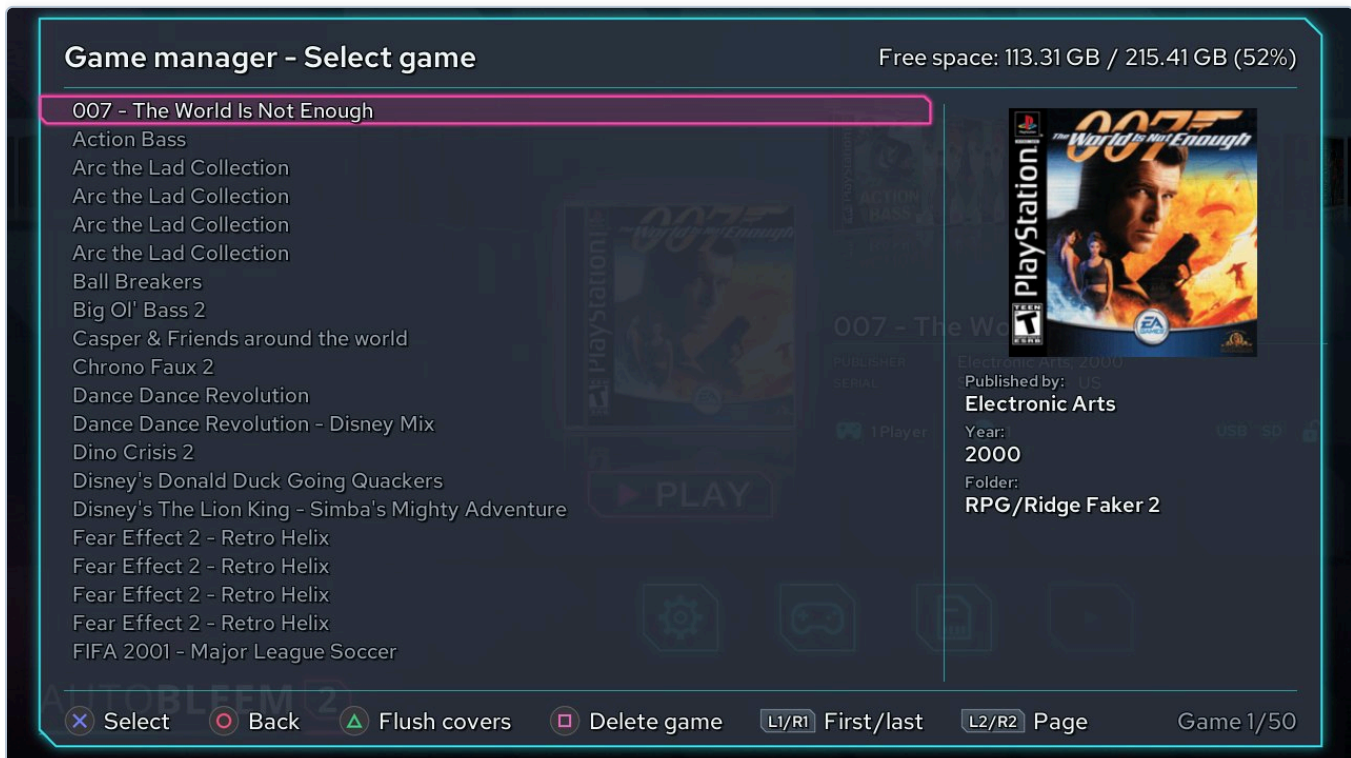
Scanner processors

Writing your own: Unzip's page, https://github.com/autobleem2/proc_unzip, explains everything a processor has to do, and `tools/proc_check.py` in AutoBleem's source checks one before you share it.

4. Screens

4.1 Game Manager

The PS1 games as a list of their titles (the selected game's folder is in its details) and the selected one's cover. Cross opens the game editor, **Square deletes the game** (its folder and, after a second question, its save states), Triangle deletes every cover PNG next to the games (the scan takes them from the databases again), L2 / R2 page. The free space of the drive is at the top right. The Game Manager waits while a scan runs.

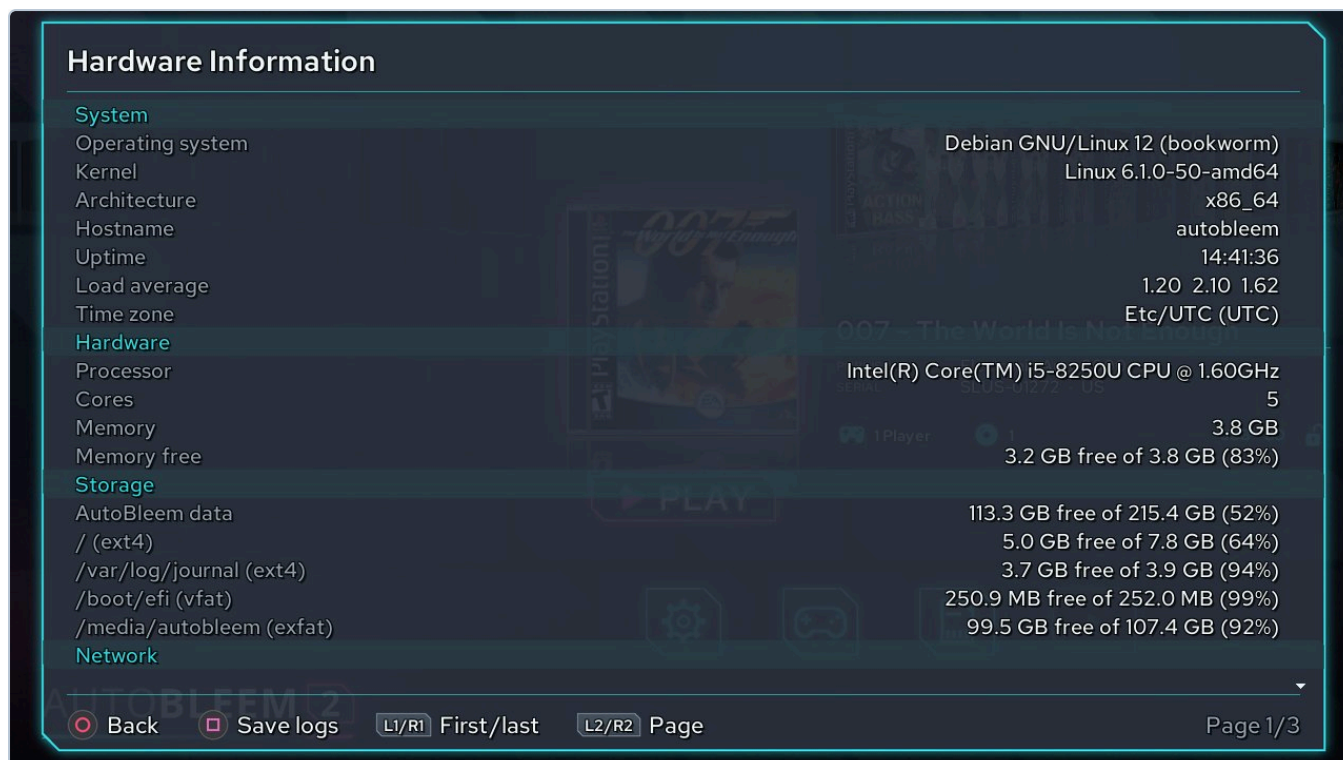


The Game Manager

4.2 Hardware Information

The machine's facts - system, hardware, storage with its free space, network addresses, the display and audio drivers, the connected pads - re-read every second. It is the same page on every platform, the console included; the network and controller setup screens are **Network & Controllers** (PSC-Bios, chapter 6).

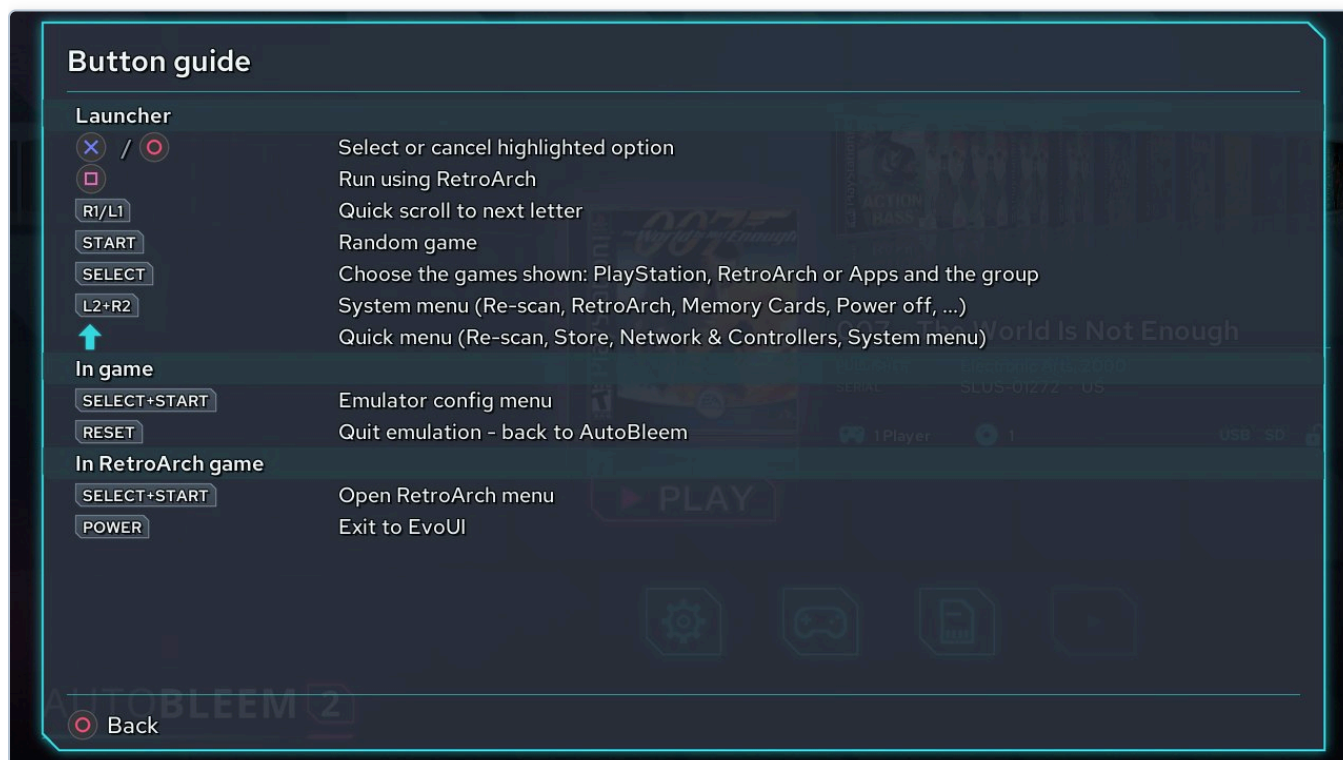
The first two controllers are shown as Player 1 and Player 2 - the ports the PS1 emulator gives them. Any further controller is shown as not used by the PS1 emulator. RetroArch assigns controllers by its own settings and may order them differently. When a controller is plugged in or pulled out, the launcher briefly shows which pad is Player 1 and Player 2.



Hardware Information

4.3 The button guide

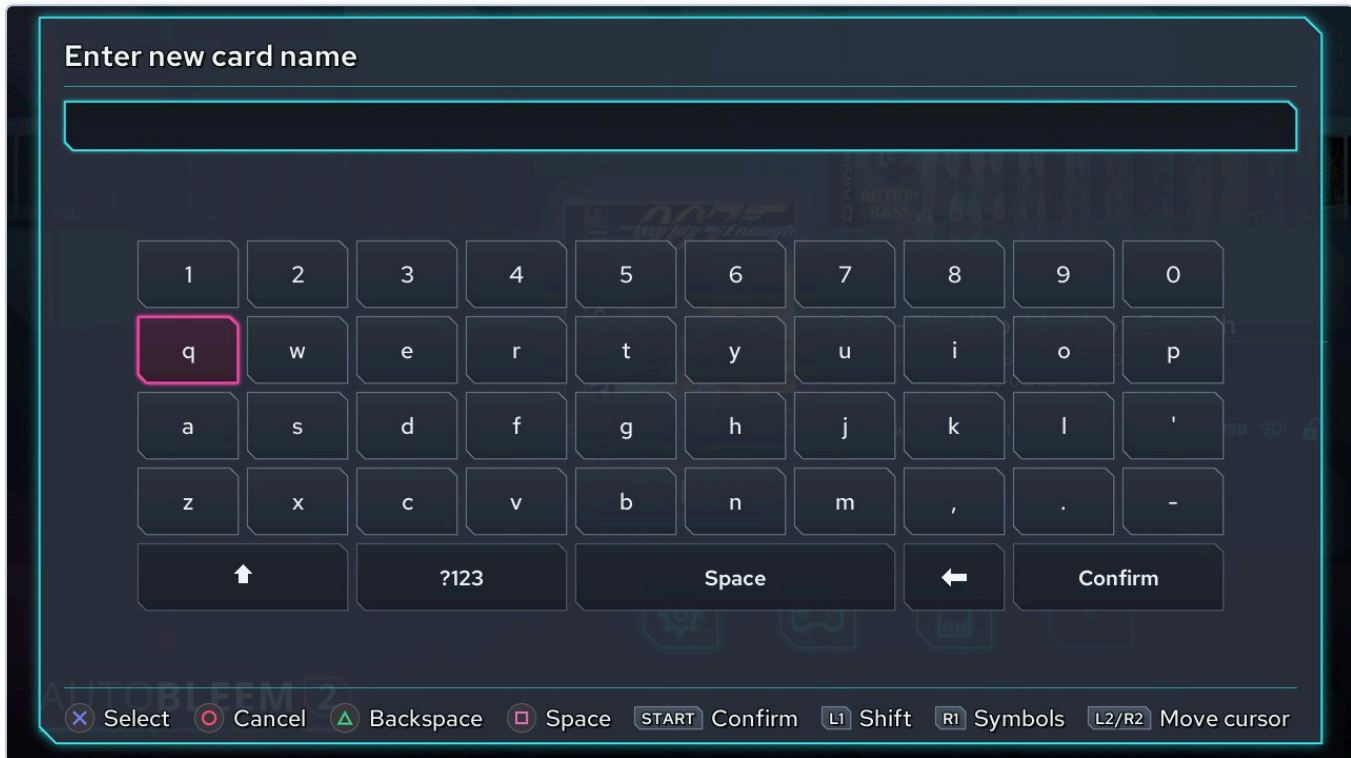
Triangle on the shelf: every button of every screen on one page. When a USB keyboard is connected or has been used, a Keyboard column shows the keys alongside the pad buttons.



The button guide

4.4 The on-screen keyboard

Wherever text is typed - a memory card set, a game's title, a WiFi password, a source's address - the same keyboard, laid out like a phone's: letters, a page of symbols (/ \ : ? & = % @ # and the rest an address or a password needs) and two pages of accented letters, with Shift, the page key, Space, Backspace and Confirm on the bottom row. The directions move, Cross types, Triangle deletes, Square is a space, **L1** is Shift (twice for caps lock), **R1** the next page, **L2 / R2** move the cursor, Start confirms, Circle cancels. A USB keyboard types at any time: Enter confirms, Esc cancels.



The on-screen keyboard

5. On the PC

5.1 UpdateRoms - refreshing a console stick

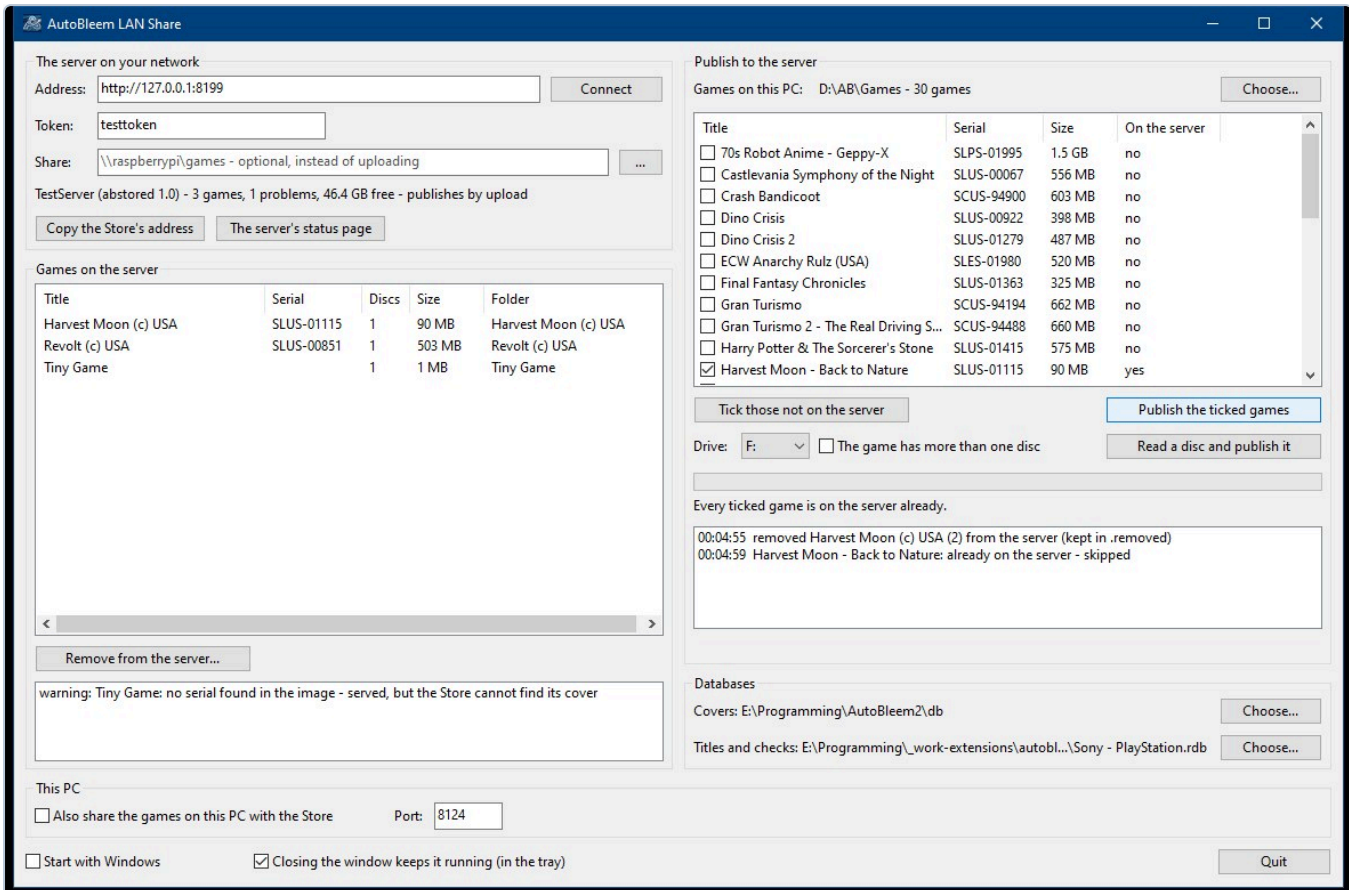
The PlayStation Classic has no network, so the RetroArch lists and box art of its stick are made on the PC: **UpdateRoms** does on the PC what the launcher's scan does on a Pi, with the PC's network and the console's paths, so the console boots and finds everything in place.

1. Copy your ROMs onto the stick under `RetroArch/roms/<system>/` (section 3.9). The folder names must be RetroArch's database names; the installer makes the common ones.
2. Start **UpdateRoms\UpdateRoms.exe** from the stick (the installer put it there). It finds the stick from where it sits, shows a stage line, a progress bar and a log, and:
 - downloads RetroArch's database bundle when the stick has none, and identifies every ROM by it - a game the database knows gets its proper name;
 - writes one play list per system into `RetroArch/bin/playlists/` with the console's paths, keeping whatever RetroArch itself added there;
 - fetches the box art of every ROM that has none from libretro's thumbnail servers into `RetroArch/bin/thumbnails/`.
3. Take the stick out safely and put it back into the console. The RetroArch tab of the set picker lists every system that has games.

Run it again after every change to the ROM folders; a folder nothing changed in is skipped, so a re-run is quick. The log is `System/Logs/updateroms.log`. A Raspberry Pi card in a card reader can be refreshed the same way (`UpdateRoms.exe <drive> --target rpi`), though a Pi does it by itself when it has a network.

5.2 LAN Share - your games and discs on the server on your network

LAN Share (`LanShare.exe`, on the Store's page in its **LAN server** tab) puts your PS1 games on the Store's server on your home network - an `abstored` on a Raspberry Pi, a NAS or another PC - and reads a PS1 disc in the PC's CD/DVD drive for it. The Store on the console, the Pi or the PC then installs them from there. Nothing to install; the settings are kept in `%LOCALAPPDATA%\AutoBleem LAN Share\`.



The LAN Share window

1. **The server:** enter its address (`http://<its address>:<port>` , as the Store has it) and press **Connect**. Its games and any problems its scan found are listed on the left. To put games on it, give one of:
 - **Share** - the server's games folder as it is shared on the network (Samba), e.g. `\\raspberrypi\games` : LAN Share copies the games there and asks the server to scan. The server itself stays read only.
 - **Token** - when the server was started with `--allow-uploads` : its token (the server prints it at start and keeps it in `<state>/upload-token`). LAN Share uploads over HTTP, and a stopped upload goes on where it stopped.
2. **Games on this PC:** choose a folder of games (one folder per game), tick games and press ****Publish the ticked games. On the server**** says whether the server has a game already (by its serial, else by its title); such a game is never sent twice. **Tick those not on the server** ticks the rest.
3. **A disc:** put a PS1 disc in the drive and press **Read a disc and publish it**. The disc is read whole into a `.bin` + `.cue` (and an `.sbi` for a LibCrypt game, when the drive gives the subchannel), named after its title, checked against the known good dump (when the databases are chosen) and published. For a game on several discs tick **The game has more than one disc**: LAN Share asks for each next disc and publishes them together as one game.
4. **Remove from the server...** takes the selected games off the server. Nothing is deleted: each is moved into a `.removed` folder next to the server's games, and moving it back puts it back.

The **Databases** - AutoBleem's covers folder (`coversU/P/J.db`) and RetroArch's `Sony - PlayStation.rdb` - give the titles and the check of a read disc; both are optional. **Also share the**

games on this PC with the Store (off by default) serves the folder on this PC to the Store directly. The first time, Windows asks about its firewall: allow private networks only.

6. The console tools (PlayStation Classic)

Two tools for a PlayStation Classic stick. Both draw in the launcher's theme and language, and both are driven by the pad - and, in the gamepad wizard, by the console's front buttons. **PSC-Bios** is an extension that comes with the console package: the *Network & Controllers* item of the Quick menu and the system menu opens it, and it is in the Extensions list. **ABFlashKit** is an App in the Apps set.

6.1 PSC-Bios

An extension that comes with the console package, also available on a Raspberry Pi and the PC stick. It is opened from the System menu's *Network & Controllers* item (or from the Extensions list). When that extension is installed but disabled, the *Network & Controllers* item in the Quick menu and System menu is greyed out with a note "enable it in Extensions" - Cross there opens the Extensions list at it.

The opening screen shows machine facts: time, timezone, WiFi/Ethernet/Bluetooth network adapters with their addresses, and every connected controller with whether it has a button mapping. The network and Bluetooth parts need the AutoBleem kernel on the console (section 6.2) or system tools on a Raspberry Pi / PC stick; the gamepad wizard works on any system.

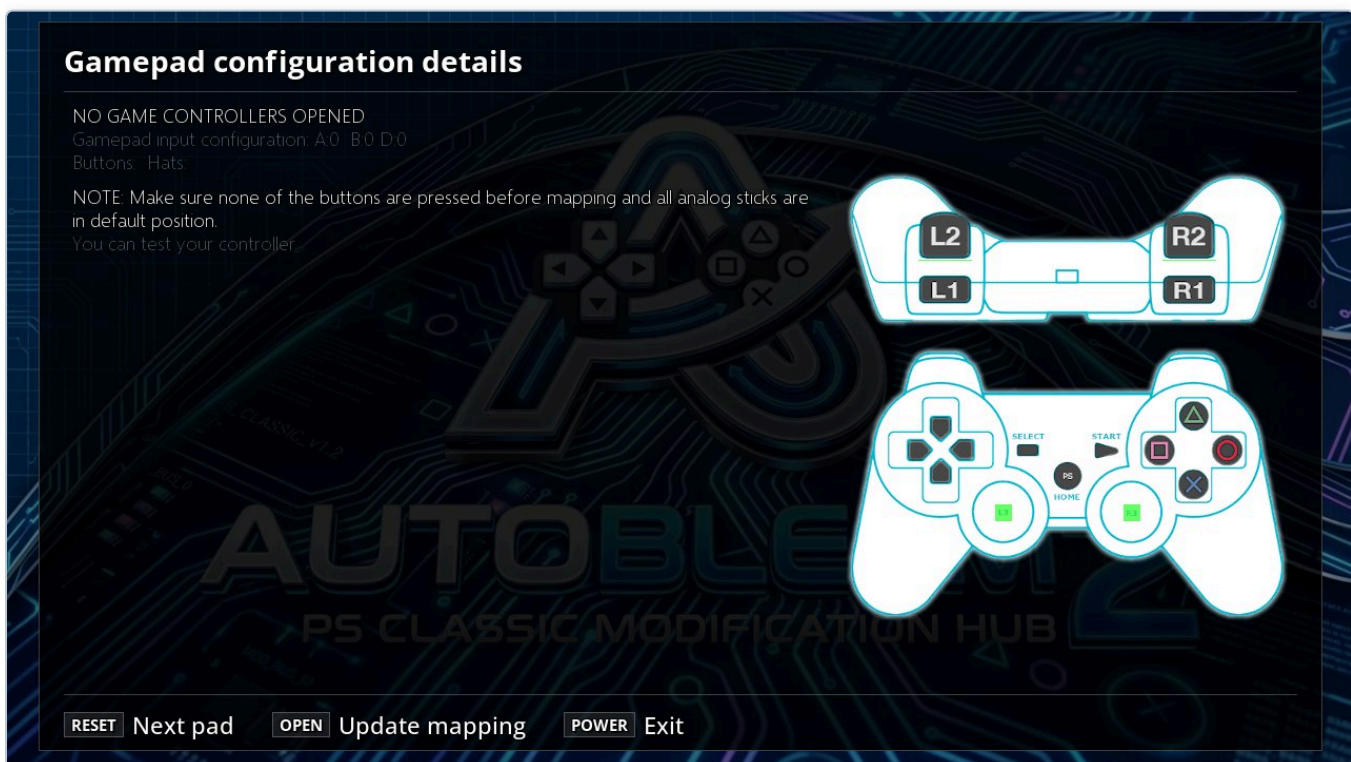


PSC-Bios: the Network & Controllers hub

- **Select - Wi-Fi Network** (kernel or NetworkManager): the network name (typed, or picked from a scan), password, driver mode, and *Apply / Restart Network*. The time zone is set here too. The console's IP address is shown once connected.
- **Square - Bluetooth Controllers**: a scan for Bluetooth gamepads (DualShock 4, etc.), to pair or remove.

- **L1 - DualShock 3 Pairing:** USB-only connection for the first DualShock 3, through the kernel's sixaxis plugin.
- **R1 - Controller Mapping:** the mapping wizard (below).
- **Triangle - About, Circle - back** to the launcher.

The **gamepad wizard** shows the connected pad raw - every axis, button and hat as numbers, and a DualShock picture that lights up as you press. Because the pad under test cannot be trusted, the wizard is driven by the console's **front buttons**: **RESET** switches to the next pad, **OPEN** starts the mapping (then answers each question - press the button lit on the picture, or OPEN when the pad has no such button), **POWER** cancels or leaves. Holding Circle on the pad for 2 seconds leaves the wizard (a bar fills and the footer hint says "Hold 2 s: Exit"). While the pad has no mapping yet, holding any button for 2 seconds does it ("Hold any button 2 s: Exit"). A short press is mapped as usual. On a keyboard, Esc / Space / Enter stand in for POWER / RESET / OPEN. At the end the new mapping is added for a test and OPEN saves it under a name of your choice; the launcher loads it from then on.

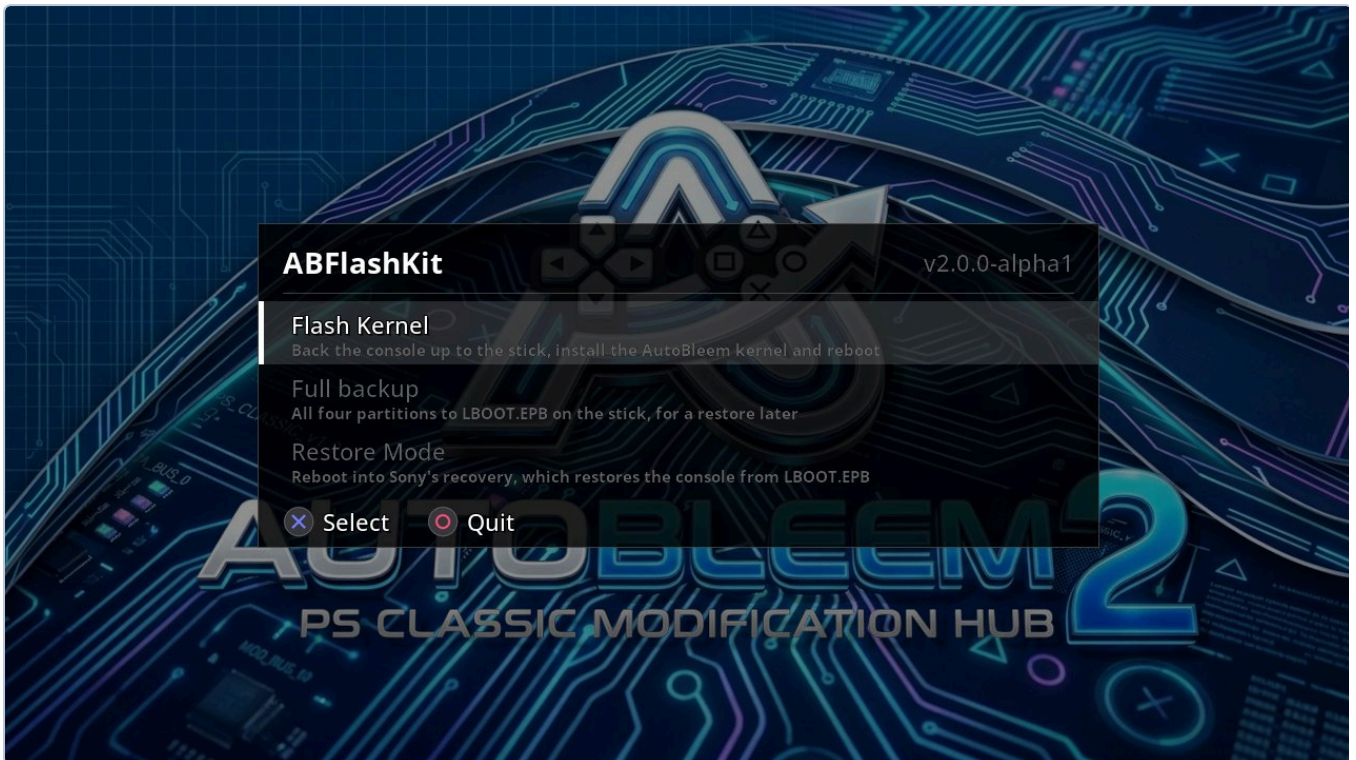


PSC-Bios: the controller mapping wizard

6.2 ABFlashKit - the AutoBleem kernel

The AutoBleem kernel is an optional replacement of the console's Linux kernel: it brings a working clock, USB WiFi and Bluetooth dongles (for PSC-Bios and Bluetooth pads) and the front-button support the emulator uses for resume points. ABFlashKit installs it, backs the console up first, and can put the console back to stock through Sony's own recovery.

This tool writes to the console's flash memory. A flash that is interrupted - the power cut, the stick pulled - can leave the console unable to start, and installing a custom kernel voids its warranty. Keep the console powered and the stick in until it reboots by itself. ABFlashKit opens on this warning; *I understand* goes on, *Quit* leaves.



ABFlashKit's menu

- **Flash Kernel:** makes a recovery backup of the console's partitions on the stick (`LBOOT.EPB`) if there is none yet, checks it and the kernel image, writes the kernel and AutoBleem's system files, and reboots. *All done - when the screen goes black replace power cord:* pull the console's power and plug it back in.
- **Full backup:** all four partitions to `LBOOT.EPB` , for a restore later (the previous backup is overwritten after a question).
- **Restore Mode:** checks the backup is a stock one, sets the recovery flag and reboots into Sony's recovery, which restores the console from `LBOOT.EPB` on the stick - the way back to the stock firmware.

A progress bar under every step shows how far the action is. The tool refuses to flash a console that runs another custom firmware (BleemSync, Project Eris): restore it to stock first.

7. If something goes wrong

- **Logs:** AutoBleem keeps its logs in memory, so the stick is not written to all the time - they reach `System/Logs/` on the stick, card or data folder only when something went wrong: a crash of the launcher, of a PS1 game or of RetroArch saves them to `System/Logs/crash-<n>/` (the last three are kept), and the launcher says so once when it comes back. To keep every log, switch on **Options -> Diagnostics -> Keep logs on the stick** (from the next start), or create an empty file `System/Logs/keep` on a PC. On a Pi or a PC, *Hardware Information* shows where the logs are and Square saves them to `System/Logs/saved-<n>/`. The files: `autobleem.log` (the launcher), `launch.log` and `pcsx.log` (a PS1 game's start and the emulator's output), `retroarch.log`, and - always on the stick - `update.log` (an online update) and `updateroms.log` (UpdateRoms).
- **A game is not on the shelf:** check the folder layout (one folder per game, the image formats of section 3.9). The *Game Manager* lists the folders the scan refused after the games, marked *Not added*, with the reason; Square deletes such a folder. *Re-scan games* in the system menu runs the scan again.
- **No covers:** the cover databases were not installed (run the installer again with them ticked), or, for RetroArch games on a console, UpdateRoms has not been run on the PC.
- **A pad does nothing or has its buttons mixed up:** PSC-Bios's gamepad wizard (a console) maps it; on a Pi or PC the Hardware Information page lists what SDL sees.
- **The console shows a black screen after a game:** AutoBleem rebuilds its window by itself (up to three times); if it stays black, hold the power button and switch the console on again.
- **Raspberry Pi:** `Alt+F2` gives a login prompt on the second console; SSH is enabled from the first boot. `sudo journalctl -u autobleem` shows the launcher's service; `sudo systemctl restart autobleem` restarts it. A first boot that could not finish (no network) retries on the next boot.
- **Windows:** `Esc` leaves the launcher; the data folder is the one chosen in the setup (`Documents\AutoBleem` by default), the logs are in its `System\Logs`.

AutoBleem is free software (GNU GPL v3 or later), with no warranty. Support and news: the Discord server linked on the About screen, and <https://autobleem.retromenele.pl/>.