

RETRO ROOM

ANALOG

Instruction Manual

Copyright

©2022–2026 Matthew D. Steele
<https://mdsteele.games/>

Contents

The Story	1
Getting Started	2
Exploring the World	4
The Pause Screen	8
Programming Machines.	10
Cast of Characters	18
Credits.	24

The Story

Buried just beneath Anna’s medieval home village lie the crumbled ruins of a long-dead civilization that once advanced by leaps and bounds, until its knowledge outpaced its understanding. Now all memory of them is lost, and their forgotten machinery lies dormant underground, waiting for someone to revive and repurpose it. Find yourself needing an elevator while exploring the caves? A crane? A weapon? You’ll have to write the code that makes them go.

Bit by bit, you can reawaken all that ancient knowledge and power, and rediscover the history of how it once fell apart. And who knows?

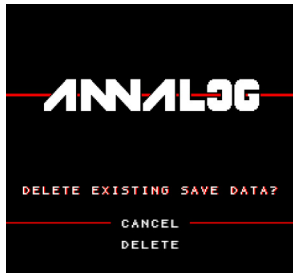
Maybe this time will be different.

Getting Started

Starting or continuing the game



To continue a previously saved game, if any, use the *D-pad* to select "CONTINUE" from the title menu, then press the **START button**. To start a new game from the beginning, select "NEW GAME" instead.



Only a single saved game file can be stored. If you try to start a new game while another saved game already exists, you will be prompted to confirm whether the old save data should be deleted.

Saving your progress

The game will automatically save your progress as you explore. When you power OFF the game, and later continue playing, you will start back at the last passage or door you passed through, or at the last machine console you used. Machine programs are saved automatically when you close the console window.



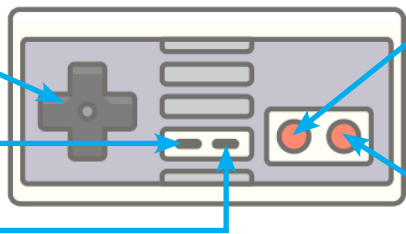
Machine consoles, doors, and passages between rooms all save the game when used.

Exploring the World

D-pad: Moves Anna left and right, or allows her to look up or down.

SELECT button: Hides or shows the head-up display (HUD) for the most recently used machine console.

START button: Pauses the game, allowing you to view the minimap or review collected papers.



B button: Press to use devices, pick up items, enter doors, read signs, talk to other characters, or (once it is collected) activate the B-remote.

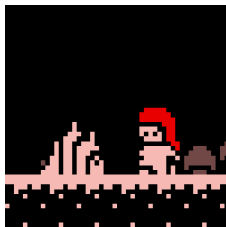
A button: Makes Anna jump.

Movement

Use the **D-pad** to run left or right, and press the **A button** to jump. When underwater, Anna will automatically float upwards, and can jump out of the water with the **A button** once she is at the surface. Anna can swim or hold her breath as long as she needs to, so you don't need to worry about drowning.

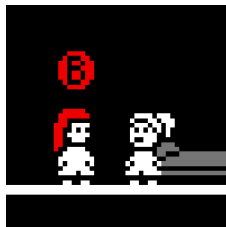
Injury and recovery

If Anna is hit by an enemy or lands on spikes, she will be injured, and the screen will turn slightly reddish. After a few seconds, she will



recover, and the screen will turn white again. However, if she is injured a second time before recovering, she will collapse and you will have to restart the room from the last passage or door you passed through, or the last machine console you used. There is no limit to the number of times you can restart and try again.

If Anna becomes crushed by a moving machine platform, or falls into lava, she will collapse instantly, and you will have to restart the room. Be careful!



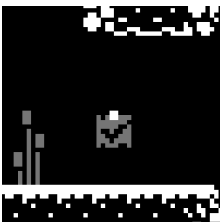
Conversation

Press the **B button** to talk to someone next to you, then press the **A button** to advance the conversation. If they ask you a question, use the **D-pad** to choose your answer, then press the **A button** to confirm.

Doors

Press the **B button** to enter a door. Some doors are locked, and may require solving a puzzle or defeating a boss to unlock.

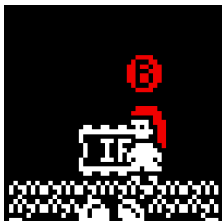




Papers

Press the **B button** to read and collect a paper from the wall. All papers you have collected can be reviewed later on the pause screen.

Press the **B button** to use a machine console. This will open the console window, allowing you to program that machine. Press the **B button** again to close the console window and resume exploring.

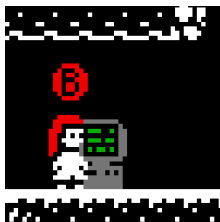
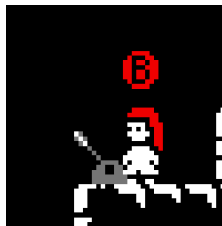


Whenever Anna enters a room, all machines in the room that have been programmed reinitialize their starting positions and registers, and then execute their programs.

Upgrades

Press the **B button** to pick up an upgrade. Each upgrade you collect provides a new capability when programming machines.

Consoles



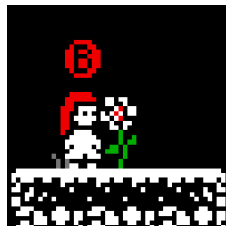
Levers

Press the **B button** to flip a lever. Most levers are connected to machines, and have no effect other than setting a register value in that machine—you must program the machine to respond to the lever position in some way. A few levers in the game have other effects instead, such as raising or lowering prison gates.

Press the **B button** to pick up a flower. If Anna is injured while carrying a flower, the flower will break, and must be collected again from wherever that flower was first found. Once a particular flower has been safely delivered to Selene in the mermaid village, it will no longer reappear where you found it.

There are twelve flowers in the game, some of which are quite challenging to find and deliver safely. However, none of them are required for completing the game.

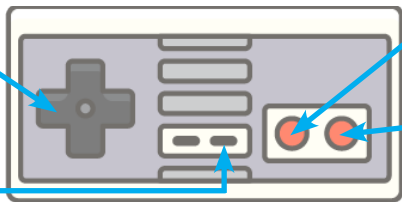
Flowers



The Pause Screen

D-pad: Moves the cursor within the collected papers window. Press **Down** to reveal the collected papers window while viewing the minimap.

START button: Exits the pause screen.



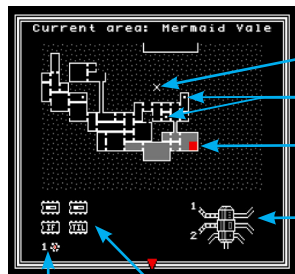
B button: Closes the collected papers window if open, or exits the pause screen.

A button: Re-reads the selected paper in the collected papers window.

The minimap

The pause screen minimap shows the parts of the game world that you have explored so far. The area of the game that you're in will be highlighted in gray, and your current location will blink red.

Any special goals given to Anna by other characters in the game are marked with an X on the minimap—going to these locations is required



Quest destination

Points of interest

Current location

Activated circuits

Delivered flowers

Collected upgrades

for advancing the story. Other points of interest, such as uncollected upgrades, undefeated bosses, and undelivered flowers, will be marked by dots on the minimap.

Collected papers



Press **Down** on the **D-pad** while viewing the pause screen minimap to open the collected papers window. In this window you can see and re-read all papers you have collected so far, as well as see how many papers are still to be collected within the current area. Use the **D-pad** to move the cursor between papers, then press the **A button** to re-read a paper. Press the **B button** to close the collected papers window and reveal the minimap again.

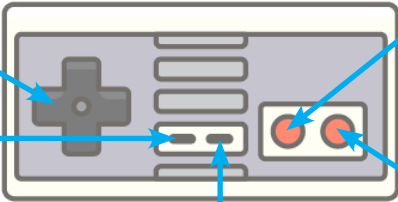
There are forty-five papers in all to collect in the game: nine CPU field manual pages, which give tips on programming machines, and thirty-six journal entries, which collectively explain some of the backstory from before the start of the game. However, none of the papers are required for completing the game.

Programming Machines

D-pad: Moves the cursor within the program or side menu.

SELECT button: Inserts a new instruction just above the currently-selected one.

START button: Opens the debugger menu, or resets the debugger.

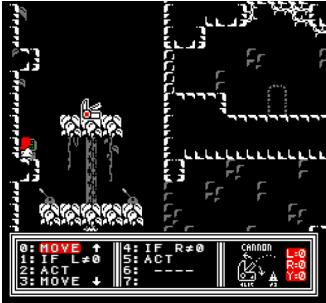


B button: Closes the console window or exits the side menu, or (once it is collected) activates the B-remote while debugging.

A button: Selects a program element to change, or selects an item on the side menu, or advances the program while debugging.

Each machine can have its own program, consisting of one or more instructions. At first, your programs can only have up to eight instructions, but collecting certain upgrades increases that limit.

The console window has three columns. The instructions of the program (if any) are listed in the left-hand and middle columns. The names and values of the machine's registers are shown in the right-hand column, along with other information specific to that machine.



Selecting from the side menu

When it is necessary to select a command, instruction, or instruction field, a pop-up menu (the “side menu”) will appear in the right-hand column. Within the side menu, use the **D-pad** to highlight the item you want, then press the **A button** to confirm (or the **B button** to cancel).

Writing a program

Each instruction in a machine's program has an operation (such as “MOVE” or “IF”), and may also have one or more fields (such as the direction to move in, or the values to compare). To change the operation of an instruction, or to change any of its fields, use the **D-pad** to move the cursor until the operation or field is highlighted, then press the **A button** and make your selection from the side menu.



To add a new instruction to the end of your program, move the cursor to the dashed line, and press the **A button**, then select an operation. To insert a new instruction just before another one, move the cursor to the instruction you'd like to insert before, press the **SELECT button**, then select an operation. To delete an instruction from the program, move the cursor to highlight that instruction's operation, press the **A button**, then select “delete”. To erase the whole program, press the **START button** to open the debugger menu, then select “ERASE”.

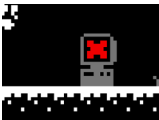
Execution

Each machine executes its program one instruction at a time, from first to last.

After executing (or skipping) the last instruction, the machine automatically loops back to the first instruction and continues. In this way, the machine keeps running forever, unless it reaches an END instruction or encounters an error.

Errors

If the machine tries to MOVE and cannot (because it is blocked in that direction), or tries to ACT and cannot (because the action isn't currently possible), the program will halt and the machine's console screen will flash a red "X" until you reset the machine by using its console.



You can avoid errors by crafting your program carefully; on the other hand, sometimes an intentional error is the simplest way to halt the machine until you're ready to reset it manually, so don't be afraid to take advantage of them.

Registers

Each machine has one or more registers, each of which has a value that can change over time. For example, a machine might have a Y register whose value is always equal to the machine's vertical position, or an L register whose value is controlled by a nearby lever. Some instruc-

tions can test these values to help control the machine; for example, an "IF L=1" instruction skips the next instruction unless the nearby lever has been flipped to the "1" position.

When a machine's console window is open, the names and values of its registers are displayed in the right-hand column. When the console window is then closed, the registers for that machine are displayed in a head-up display (HUD) in the top-left corner of the game screen until Anna leaves the room or uses another machine's console. If this HUD becomes a distraction for you, you can toggle it on and off with the **SELECT button**.

Using the debugger

While editing a program in the console window, press the **START button** to open the debugger menu, then select the "DEBUG" option to begin debugging. This will reset all machines in the room.

While the debugger is active, you can step through the program one instruction at a time by pressing the **A button**, and observe the machine's execution step by step. Press the **START button** again to reset the machine and exit debugging mode.

While debugging, certain machine actions will only perform a "dry run"—for example, rocket launchers won't fire real rockets while debugging. Once you're ready to try your program for real, exit the debugger and close the console window to allow the machine to run normally.



Instructions

MOVE direction [available in every machine that can move]

The MOVE instruction causes the machine to move by one step in the specified direction. Executing MOVE when the machine cannot move any farther in that direction is an error (see page 12).

ACT [available in every machine that can perform a special action]

The ACT instruction causes the machine to perform a special action specific to that machine, such as firing a projectile or dropping its load. Executing ACT when the machine is unable to do so (for example, trying to fire a rocket while the launcher barrel is blocked) is an error (see page 12).

END [available in every machine]

The END instruction causes the program to halt. The machine will emit a chiming noise when this happens. No more instructions will be executed until you reset the machine by using its console.

TIP: Most programs don't need to include an END instruction, as it's usually more useful for the program to keep looping forever. However, it is sometimes useful to make a machine move into position or perform a single task, and then halt until you are ready to manually reset it.

In the following instruction descriptions, an instruction field that is described as a “value” may be either a numerical constant (from 0 to 9) or the name of a register.

IF value1 comparison-operator value2 [available in every machine once the IF upgrade has been collected]

The IF instruction compares two values and skips the next instruction if the comparison condition isn't met. For example, an “IF X=0” instruction allows execution of the following instruction when

the X register is 0, but skips over the following instruction when the X register holds any other value.

TIP: An IF instruction skips over exactly one instruction—the second instruction after the IF will not be skipped.

TIL value1 comparison-operator value2 [available in every machine once the TIL upgrade has been collected]

The TIL instruction compares two values and goes back to the previous instruction if the comparison condition isn't met. For example, a “TIL X=0” instruction keeps going back to the previous instruction over and over until that instruction causes the X register to become equal to 0.

TIP: A TIL instruction repeats only the immediately previous instruction. Moreover, it will not by itself prevent that instruction from being executed the first time—it can only control whether or not the instruction will be repeated a second time or more.

GOTO instruction-number [available in every machine once the GOTO upgrade has been collected]

The GOTO instruction causes program execution to jump to the specified instruction.

TIP: When combined with IF instructions, GOTO instructions allow building more complex logic into your programs. They are also useful for creating longer instruction loops than are possible with the TIL instruction.

REST value [available in every machine once the REST upgrade has been collected]

The REST instruction causes program execution to pause for a length of time specified by the value before continuing to the next instruction. For example, REST 4 pauses for the amount of time it would typically take for a machine to execute four MOVE instructions.

TIP: REST instructions allow you to slow machines down at critical points in the program, for example to give yourself more time to jump onto or off of a platform before it moves away.

COPY register $\leftarrow$ value [available in every machine once the COPY upgrade has been collected]

The COPY instruction stores the specified value into the specified register.

TIP: Changing the value of a register may have a visible effect on the machine. For example, writing to a lever register L will flip the lever; writing to a valve register V will rotate the valve. Some machines later in the game can only be usefully operated by writing values to their registers.

When the COPY upgrade is collected, every machine gains a new A register. Writing to the A register does not have a visible effect on the machine, but can be used to store a numerical value for later use. For example, a program could copy some other register into the A register, and then read that value from the A register later on, even if that other register has since changed.

ADD register $\leftarrow$ value1 + value2 [available in every machine once the ADD / SUB upgrade has been collected]

SUB register $\leftarrow$ value1 - value2 [available in every machine once the ADD / SUB upgrade has been collected]

MUL register $\leftarrow$ value1 x value2 [available in every machine once the MUL upgrade has been collected]

These instructions add, subtract, or multiply two values, then store the result into the specified register, just like the COPY instruction. If the resulting value would be less than 0, then 0 is stored instead; if the resulting value would be greater than 9, then 9 is stored instead.

SYNC [available in every machine once the SYNC upgrade has been collected]

The SYNC instruction causes program execution to pause until all machines in the room have each reached a SYNC instruction, at which point all of them resume executing simultaneously.

TIP: SYNC instructions help allow multiple machines to work together, aligning their positions or timing with each other before proceeding. Sometimes it is helpful to also place a REST after each machine's SYNC, so that they pause together at the same time.

TIP: Using two SYNC instructions in a row can help a pair of machines to take turns with each other. For example, one machine can SYNC to signal to the other to proceed, then immediately SYNC again to wait for the other machine to finish.

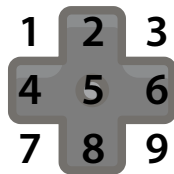
SKIP value [available in every machine once the SKIP upgrade has been collected]

The SKIP instruction causes the program to skip past the number of subsequent instructions indicated by the value. For example, SKIP 3 skips the next three instructions.

TIP: SKIP instructions are most useful when skipping over a variable number of instructions, by specifying a register instead of a constant value. This advanced instruction should be considered optional; you may not find that you need to use it.

The B-Remote

All the machines that Anna encounters in the game were built to respond to this master remote control. When the B-Remote upgrade is collected, all machines gain a B register that responds to the **B button**. Whenever the **B button** is being held down, the B register will hold some value from 1 to 9, depending on which directions are being held down on the **D-pad**; whenever the **B button** is not being held down, the B register will hold the value 0.



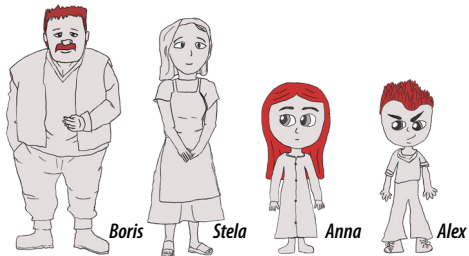
Cast of Characters

Human townsfolk

Bartik Town is home to Anna and all her family and friends.

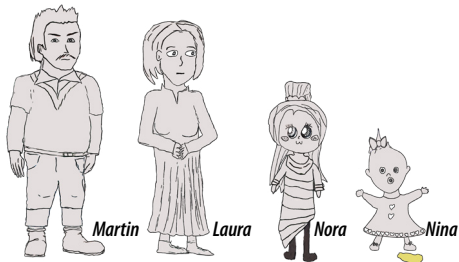
Boris, Stela, Anna, and Alex

Anna's twin brother Alex is courageous and precocious, and is currently apprenticed to Smith Dominic. Anna is thoughtful, and quieter than her brother, but no less brave. Their mother, Stela, cares for them while their father, Boris, is away on an expedition across the sea.



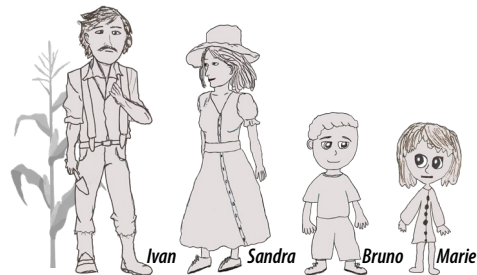
Martin, Laura, Nora, and Nina

Martin and Laura are uncle and aunt to Alex and Anna, and work hard to help keep the town running. Their elder daughter, Nora, often gets stuck babysitting for her little sister Nina while her parents are working.



Ivan, Sandra, Bruno, and Marie

Ivan and Sandra work hard in the fields to provide food for the townsfolk, but it hasn't been easy lately. Their kids, Bruno and Marie, are fast friends with Alex and Anna.

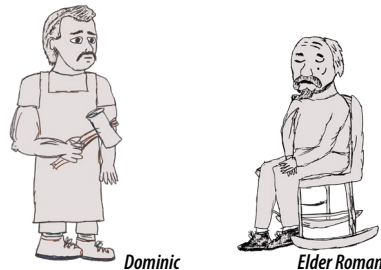


Smith Dominic

Dominic is the town blacksmith, and works to maintain the farming tools that keep the town going. He is a patient teacher to his apprentice, Alex.

Elder Roman

Roman is the town elder and a source of wise counsel, but age has been wearing on him lately. He is prone to sleepiness and forgetfulness.

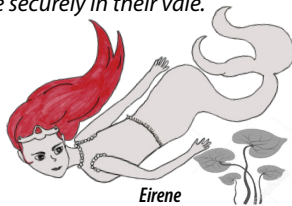


Mermaid villagers

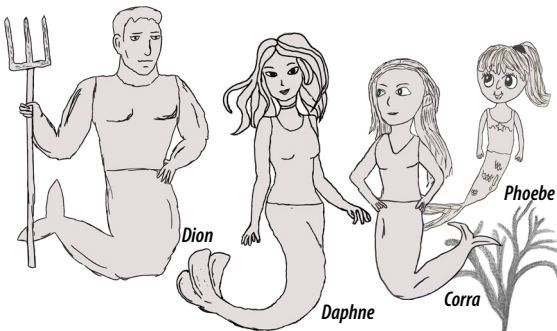
Hidden deep underground, far from the humans, the mermaids live securely in their vale.

Queen Eirene

As queen of the mermaids, Eirene's first priority is the welfare of her people. However, she is dedicated to the cause of peace, and will never turn away those in need, even when she has her doubts about them.



Eirene



Dion, Daphne, Corra, and Phoebe

Dion and Daphne work as farmers for their village. Their elder daughter, Corra, loves to learn and explore, and is discovering her own path in life as she verges on adulthood. Their younger daughter, Phoebe, wishes she could go exploring too, but her parents won't let her yet.

Selene

Selene is fond of flowers, and often serves as an advisor to Queen Eirene. She knows a great deal of history, far more than most mermaids (or humans or orcs), and is known to share her wisdom with those willing to help her in return.



Selene



Guards

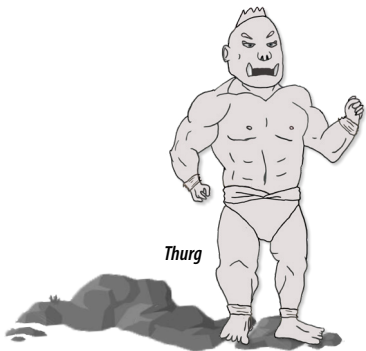
Although the mermaids have embraced peace as a way of life, they remain ever vigilant to protect their domain from orcs and other dangers. Under Queen Eirene's leadership, and the natural protection of their vale, the mermaids have remained safe and prosperous for many years.

Orc raiders

Hailing from over the mountains, the orcs were no more than a frightening bedtime story for most humans, until now.

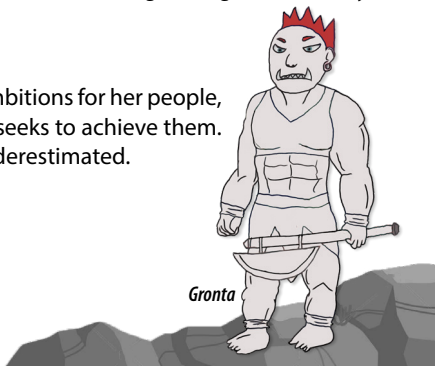
Chief Gronta

The fearless leader of the orcs, Gronta, has ambitions for her people, and will let nothing stand in her way as she seeks to achieve them. Cunning and dangerous, she is not to be underestimated.



Lieutenant Thurg

Thurg is fully loyal to Chief Gronta, even if he is perhaps not as clever as she is. He despises humans and mermaids.



Grunts

All the orcs look to Gronta as their leader, and respect her for her strength. They see humans and mermaids as weak, and most of them are keen to demonstrate it.

War rhinos

These beasts can always be found within orc-occupied territory. Bred by the orcs for their ferocity, they will charge at high speed when provoked.



Credits

Design/Programming: Matthew Steele

Music: Jon Moran, Matthew Steele

Additional artwork: Stephanie Steele, Holly Steele, Emily Steele

Creative Commons assets: Adam Saltsman, CobraLad, Daivuk, drotzruhn, Exuin, gpag1, Hyell, KJose, Luke Sharples, qubodup, SamsterBirdies, Walter Odington

Beta feedback: Cam Tenny, Costas Akrivoulis, Christopher DuBois, David Couzelis, Eric Chang, Gus Prevas, Jon Moran, verylowsodium, walkingeye

Manual design: Barbara Steele

Copy editor: Guy Steele

Publishing: The Retro Room

Ultimate Kickstarter backers: Aaron Schmitt; Aaron Smith; Adam Gashlin (hcs); Alex Beene; alwaysdoit; Amy Weinstock; Andi McLean; Andrea “idillicus” Piccardi; Andrew Cove; Andrew Saxe; Anna Rachel Brooks; Anonymous; Anonymous; Anthony Sontag; BaghiraMusic; Bogomil76; Borch187; Brayden Waltz; Brayden White; Carlos Noguera; Cezar “ikari” Pokorski; Chris Kee; Christian Deitering, aka “Ferris Bueller”; Civilian; Colby Haag; Coursin Hill; DPS2004 (Daphne S.); Damien Darmandé; Daniel Hui; Dave Holets; David Kerns; Diana L.; DoubleEwe01; Dylan Hesthaven; Dylan Turtle; Dylan ‘Clan’ Possamaï; Eric Missimer; HeatGuyStu; JSH1up; Jacuk Sulinskai; James DeLuca; James Maurice; Jeffrey Geske, MD; Jesse Thorn; jgkspssx; John Falkevitz; Jonathan Sato; Justin Orenich; Justyn “Honk” Dooley; Jérémie Théolier; Keenan Hecht; Lee Hathcock; Maciej Hopek; Matthew Devine;

Matthew Lotti; Matthias Hildebrandt; Mike Coln; mmmndonut; Mohammed Al-Saai; natgreystar; Nicholas Ostrom; Nicholas Paul Wilson; non; Oliver Haslinger; PatCDR; Patrick McIntosh; rabideau; retrocity5; Robert fireturtle Oliveira; Rocco Buffalino; RoxasAmor; RpgCollector; RumeB; Ryan “Faub” Fauble; Scott Lowe; Seiromem; SkinnyLegs; SkoveDog; Steve Fowler; Steven M. Basile Jr.; Tero Lahti; Terry ‘Doc’ Jordan; The Burner; The Great Gildersleeve; Tilly Chamberlain; Tim Phelps; Tom Rybka; TrevorO; Ty; Uros “Mucky” Colovic; walt6502; Will O.; Zac Cochran

Special thanks

Thank you to the NES homebrew/emulation community, especially NesDev.org, for producing all of the documentation, wiki articles, forum posts, and software tools that I relied on while making this game. This project would have been impossible without the tireless work over the decades—reverse engineering, extensive testing, meticulous documentation, and emulator writing—of countless other NES fans, whose shoulders I am standing upon. For anyone who’s ever wanted to make their own homebrew game, there’s never been a better time than now.

Thank you to my family for their encouragement and patience with me as I sunk hours and hours of free time into willing my weird ideas into existence. This game is dedicated to them.

And thank you to you, the player, for giving Analog a try. This has been my passion project for more than four years now, and it means so much to me to be able to share it with you.

—mdsteele

