

Making a Game Boy game in 2017: A "Sheep It Up!" Post-Mortem (part 2/2)



gamasutra.com/blogs/DoctorLudos/20180213/314554/Making_a_Game_Boy_game_in_2017_A_quotSheep_It_Upquot_PostMortem_part_22.php



by [Doctor Ludos](#) on 02/13/18 09:11:00 am

Featured Post



The following blog post, unless otherwise noted, was written by a member of Gamasutras community. The thoughts and opinions expressed are those of the writer and not Gamasutra or its parent company.

Everyone has childhood dreams. Mine was to make a game for my first console: the Nintendo Game Boy. A few months ago, I fulfilled this dream, by **releasing my first Game Boy game on a actual cartridge**: Sheep It Up!

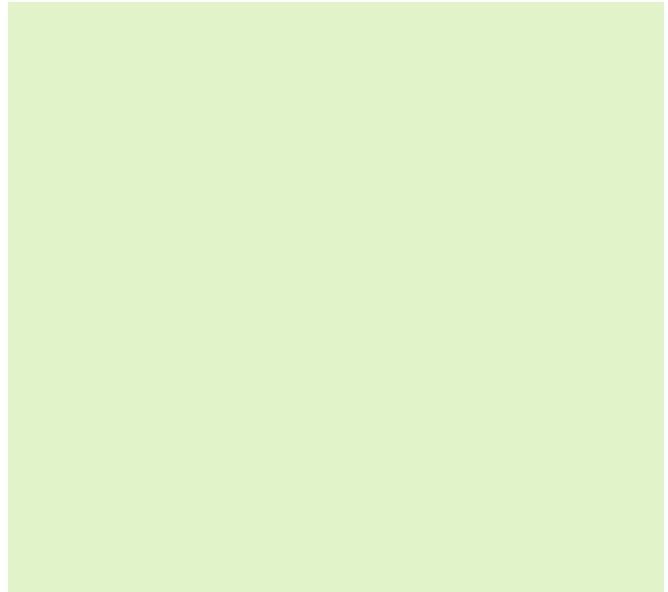
In this article, I'll present the tools I used, and some pitfalls a newcomer like me had to overcome to make this project a reality! Due to it's length, this article is split in two parts:

- Part 1 : [Tools used / Technical challenges / Art challenges](#)
- Part 2 : Audio challenges / Making Cartridges / Player Feedback

The game

"Sheep It Up!" is an arcade game where a sheep has to climb up by hanging himself to flying velcro straps. The concept is simple, but the game rapidly gets quite challenging: how high can you climb without falling down?

As a game collector myself, I wanted this game to be produced without sacrificing any vintage Game Boy titles. So everything is manufactured specifically for this game: the pcb, the rom, the shell, the protective case and even the sticker! We also tried to keep the price reasonable, so everyone can enjoy the game: **\$15 (+shipping)**. It will run on any Game Boy model, from the first one to the GBA SP, including the Super Game Boy and the Game Boy Player.



If you still own a Game Boy, you can buy a cartridge on the publisher website:

<https://catskullgames.com/sheep-it-up>

Sound constraints

As we've seen in [part 1](#), the Game Boy comes with some obvious limitations on the CPU power and graphics sides. But same can be said for audio. Indeed, unlike modern machines, the Game Boy can't play MP3 music or read WAV sound files. To make sounds and music with a Game Boy, you have to dynamically generate it using the 4 "sound channels" built inside the console. Each channel has a different function:

- **Channel 1 - Tone & Sweep:** can play a sound (a tone) with a pitch moving up or down.
- **Channel 2 - Tone:** plays a tone with no special effect.
- **Channel 3 - Waveform:** can play very simple waveforms (*encoded in 4bits*).
- **Channel 4 - Noise:** can generate explosion or engine sounds.

Let me say this again: **to make a sound with a Game Boy, you have to generate it manually by directly controlling these 4 sound channels**. Hopefully, this is actually quite easy to do: you simply need to modify the values of some hardware registers. Each sound channel is controlled by 4 or 5 hardware registers (*8bits variables*) that defines volume, pitch, special effects, etc. By setting different values on the registers that control a channel, you can make the Game Boy produce a large variety of sounds. For example, here are three sounds coming from *Sheep It Up!* that I created by using sound channel 1, alongside with the hardware registers values used to generate them:

Jump

```
NR10_REG = 0x15;  
NR11_REG = 0x96;  
NR12_REG = 0x73;  
NR13_REG = 0xBB;  
NR14_REG = 0x85;
```

Land on ground

```
NR10_REG = 0x79;  
NR11_REG = 0x8D;  
NR12_REG = 0x63;  
NR13_REG = 0xC8;  
NR14_REG = 0x80;
```

Falling down (game over)

```
NR10_REG = 0x4F;  
NR11_REG = 0x96;  
NR12_REG = 0xB7;  
NR13_REG = 0xBB;  
NR14_REG = 0x85;
```

As you can see, setting different values to 5 variables enables you to produce very different sounds effects. One could image that, to create sounds, you could go the trial and error route:

putting random values to the registers until you find a pleasing sound to hear. But that would be really tedious and not meaningful at all. So, hopefully, **modern tools are now available to make our life easier**. Namely, let me introduce you to **GB Sound Sample Generator**.



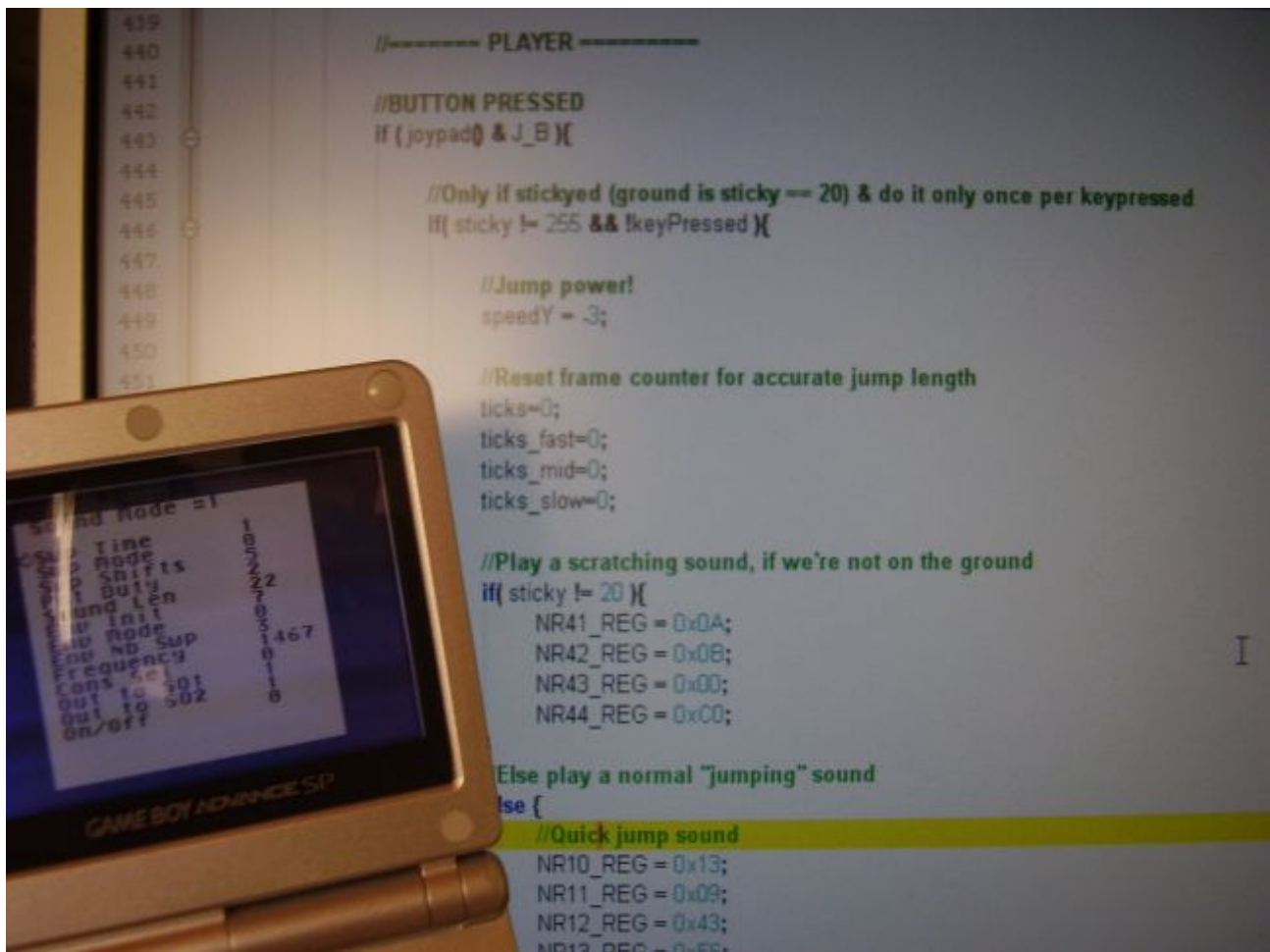
First, you tweak user-friendly parameters to find a sound you like (press START to test current sound)

It's a Game Boy program, fully working on real hardware, that allow you modify the values of the sound hardware registers in a user-friendly way. Instead of setting the raw value of "NR10_REG", you'll now be defining a sound frequency, a volume level, a sweep time, etc. You can think of this tool as SFXR for Game Boy. But to export your sound, instead of saving a WAV file, GB Sound Sample generator will display the corresponding hardware register values. So, to play a sound in your game, now you simply have to write down these values into your source code.



then, you hold A+SELECT to display the corresponding raw hardware register values

For Sheep It Up!, I used GB Sound Sample generator on a Game Boy Advance SP (*backlight screen is easier to read*), with the ROM running on a Everdrive GB. I spent quite some time tweaking parameters directly on the console, and when I found a nice combination, I could easily copy the values from the GB screen to the source code on my computer.



To play this sound in your game, you simply need to copy the hardware register values in your source code

Chiptune music!

Despite limited audio capabilities, nowadays the Game Boy is used as a musical instrument. In the chiptune scene, many talented artists are making energizing tracks on Game Boy, by using programs like [LSDJ](#) or [Nanoloop](#). For example, you may have heard of Irish chiptune legend [Chipzel](#), who composed music for [Super Hexagon](#), and French wizard [Sidabitball](#) who remixed many [pop music hits from the 90's](#).



Chipzel



Sidabitball

Sadly for us game developers, LSDJ, the tool used to compose these outstanding songs, use almost 100% of the Game Boy hardware resources. So you can make outstanding music with it, but you can't run a game while playing this music on the console. Hopefully, there are less resource-hungry alternatives to create and play Game Boy music.

1) Custom music player

The most bare bone solution is to make your own "music player". Remember, the only way to make sounds on a Game Boy is to set values to the hardware registers that controls the 4 sound generation channels. So you can quite easily make your own "music player" by storing "notes" in an array. Each "note" is a particular combination of values for each hardware register. Then, if you define a tempo at which you run through this array to play these notes, you can have a very simple way to make music. And **it's actually how I did it in Sheep It Up!** Here is a short gameplay video with sound and music so you can hear the result:

To be more specific, my "notes" are only played on the sound channel 2. Indeed, the 4 sound channels are shared for both sound effects and music. So, **if you are playing a note on a channel, you can't use it to play a sound effect at the same time.** And, as each channel is somewhat "specific", sometimes it may be preferable to use some channels for music only, while others channels may be reserved for sound effects. For example, the "scratching" sound that you hear whenever the sheep get attached to a flying strap is played on channel 4 (noise). As it's a very common sound in my game, I decided to avoid using this channel for music. Same goes for channel 1 that I used for all the other sound effects. As my music skills are very limited, I choose not to use the waveforms of channel 3, and to focus on the tones of channel 2 instead.

2) GBT Player & Carillon Player

Many homebrew games use the “custom music player” solution. Hopefully, this is not the only one available. There is a library called GBT Player that can convert music modules (*in Impulse Tracker “.it” format*) to play them back on a Game Boy. Of course, there are a lot of limitation regarding the sample sizes, the number of channels and the notes effects that you can use. But still, it allows you to compose real music in an actual music tracker, such as OpenMPT, and play it back on a Game Boy. The cherry on the cake is that this library only consumes about 7-8% of the Game Boy CPU resources, leaving a lot of processing power to run the rest of the game. Moreover, it’s very easy to integrate it into your code, whether you choose to program your game in C with GBDK or in assembly with RGBDS. Carillon Player is a good alternative to GBT Player, although you’ll have to use Carillon Editor to compose music instead of your favorite tracker program.

Geometrix is an homebrew game by AntonioND, creator of GBT Player. The music uses this library.

Let’s get physical: putting the game on cartridges

After a lot of work on code, graphics and audio, the game was finally complete. But, at this stage, all I had was a ROM file. Sure, I could (*and did*) test it a lot on the various Gameboy models in my collection (*all of them except for the AGS-101*). But it didn’t feel like the project was complete: to me, **the game needed to have its own dedicated cartridge to be considered as “finished”**.

As I mentioned in [part 1](#), the whole *Sheep It Up!* game ROM weights only 32KB, like the early Game Boy titles such as *Tetris* and *Alleyway*. But most of the Game Boy games are actually way larger, ranging from 128KB to 512KB, with *Pokemon Blue and Red* being the largest games at a whopping 1024KB (1MB). In the 90’s, the larger the game, the more costly were the cartridges to produce: you had to use more expensive chips to store large games. But today, they all cost almost the same. **So why did I limit myself to a 32KB game?**

The answer is that I wanted to be able to release the game on newly produced cartridges. **As you can imagine, Nintendo have discontinued manufacturing Game Boy cartridges (*err.. I mean Game Paks*) since a long time ago. So, two alternative solutions exists to make your own game cartridge today:**

1) Cartmodding

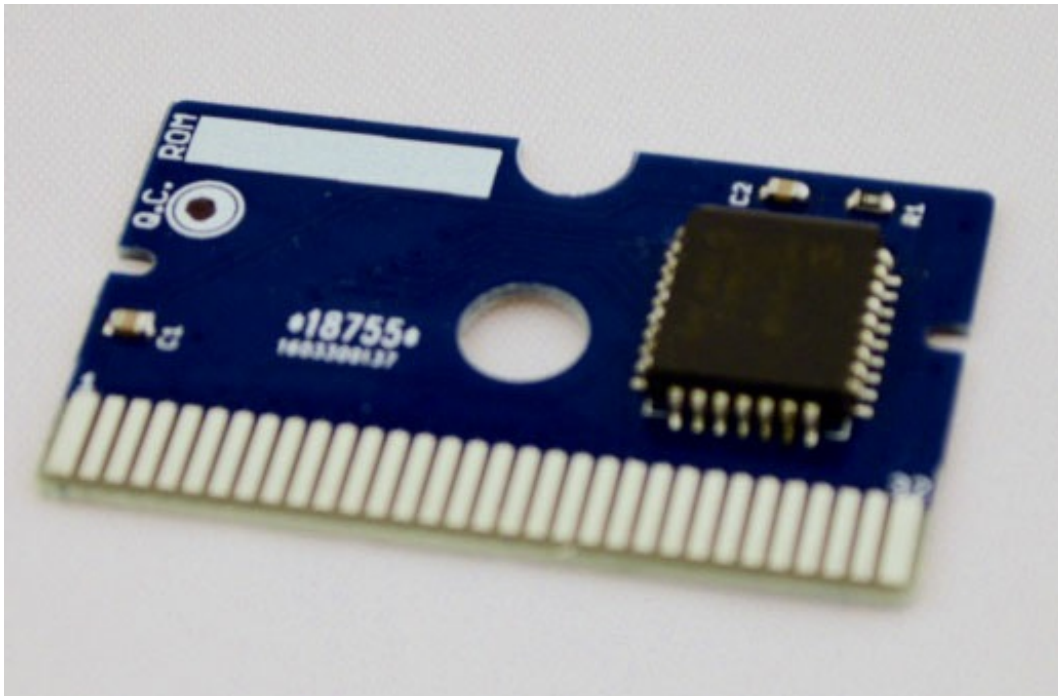
The first one, named “cartmodding”, consists in recycling vintage Game Boy games. The games released during the Game Boy commercial era are stored on non-rewritable cartridges. So, to replace the game stored on it, you have to open it, desolder the memory chip, and replace it by another memory chip where you uploaded your own game ROM file. This solution allows you to create games of any size. But, as a game collector myself, I didn’t want to destroy any Game Boy games to produce mine.

An example of cartmodding. A new memory chip containing a homebrew game was soldered inside a vintage cartridge



2) Custom built cartridge

So, the second solution is to build a new cartridge from scratch. As I know nothing about electronics, I started to search online for people able to build Game Boy cartridges. After several failed attempts, I was lucky enough to get in touch with Catskull. He's a brilliant electronic wizard who build and sells a lot of Game Boy related hardware on his shop [Catskull Electronics](#). Among others things, he designed a device to connect the Gameboy to a MIDI interface, enabling musicians to use Game Boy with actual instruments and synthesizers. He also designed his own Game Boy cartridge. For his cartridge, **everything is manufactured anew**: the PCB, the chips, the shells, and even the stickers are new parts - nothing is recycled from previous games. That was the perfect solution for me.



The original cartridge PCB designed by Catskull, used for Sheep It Up!

It gets even better: after a few email exchanges, Catskull told me he wanted to start a homebrew games publishing label since a long time ago, and that my game was the perfect opportunity for him to launch it! So, we teamed up: *I make the software, and he makes the hardware*. The only limitation with this solution is that **Catskull's Game Boy cartridge can only store a 32KB game**. Why?

Originally, the Game Boy console was designed to be able to access to 32KB of ROM data only. In order to have games with more than 32KB of data, you need to "browse" through all the data in your cartridge, by accessing only a subsection of the whole ROM data at a time. This method is called **"Bank Switching"**: the whole ROM data is segmented in banks of 16KB each, and you can swap them out anytime you want (with some limitations). However, to be able to do bank switching, **the cartridge need to have an additional chip on it** the "Memory Bank Controller" (MBC). Several chips exists (*MBC1*, *MBC5*, etc.), allowing for larger ROM sizes, battery-powered RAM for savegames, etc. Unfortunately, for now, there is no real solution to produce new cartridges from scratch with such a chip on it.

So people who are creating homebrew games larger than 32KB are usually doing cartmodding, in order to reuse the MBC chip of the vintage Game Boy cartridge they are "recycling." Another solution is to buy counterfeit copies of actual Game Boy games, and to reflash them. Indeed, since several years, Chinese manufacturers have been able to "clone" the MBC chip, in order to produce pirate copies of large games like *Pokemon*. And, as these pirate carts use EEPROM chips instead of the ROM chips found on vintage games, it's usually possible to "reflash" them to modify the game stored on it. For that, you'll need to use a cartridge flasher, such as the BennVenn's Joey. Several homebrew games larger than 32KB have been released using this solution.



A pirate cartridge made in China. This one can be reflashed to put your game on it, but others are not reflashable ([more info](#))

Regarding Sheep It Up!, I wanted to produce 100% newly manufactured cartridges to avoid destroying original Game Boy games. So I was very lucky to be able to partner with Catskull, who produces his own cartridges, even if it meant that I was limited to 32KB of total storage.

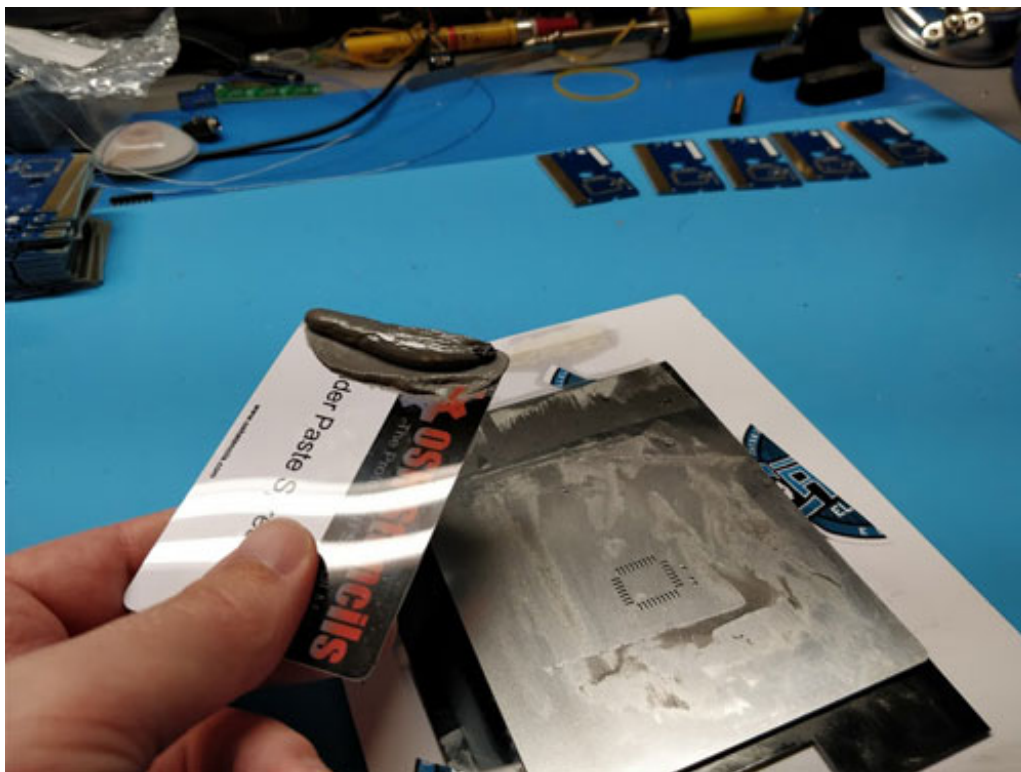
Baking cartridges!

So, how are “custom-built” cartridges made? I’ve asked Catskull to take pictures of the different steps of his own cartridge assembly process. As you’ll see, each cartridge is assembled by hand:

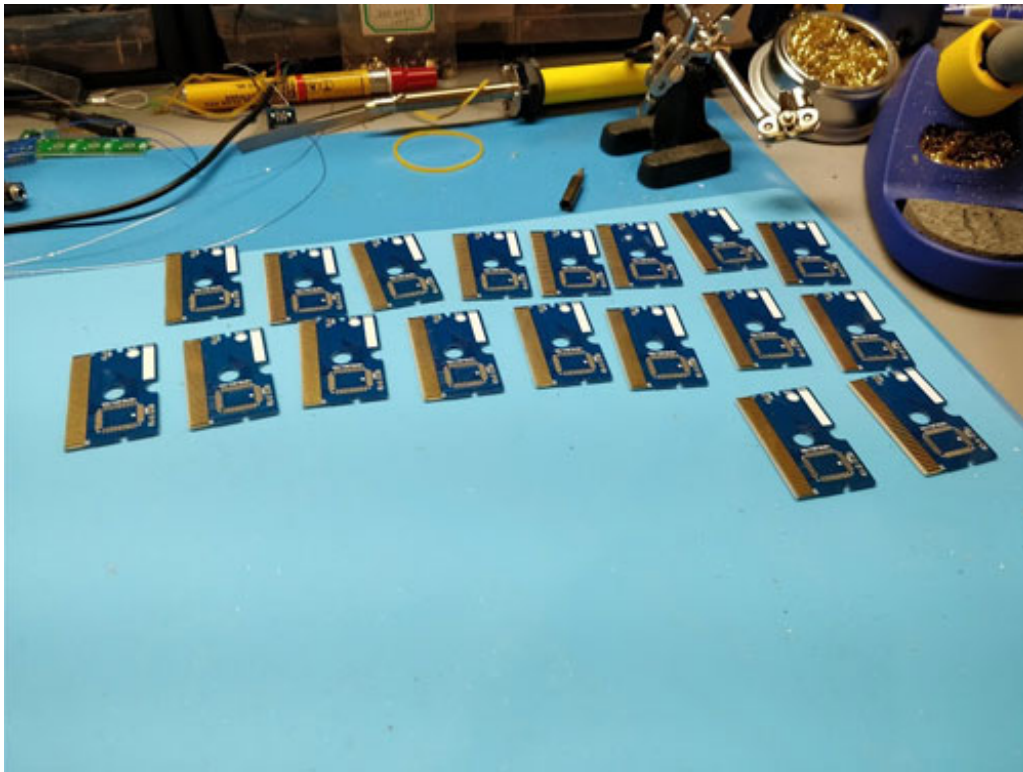
1) The PCB is inserted into a fixture.



2) A stainless steel stencil is put over the PCB, in order to apply solder paste. This paste will allow to solder the memory chip onto the card.

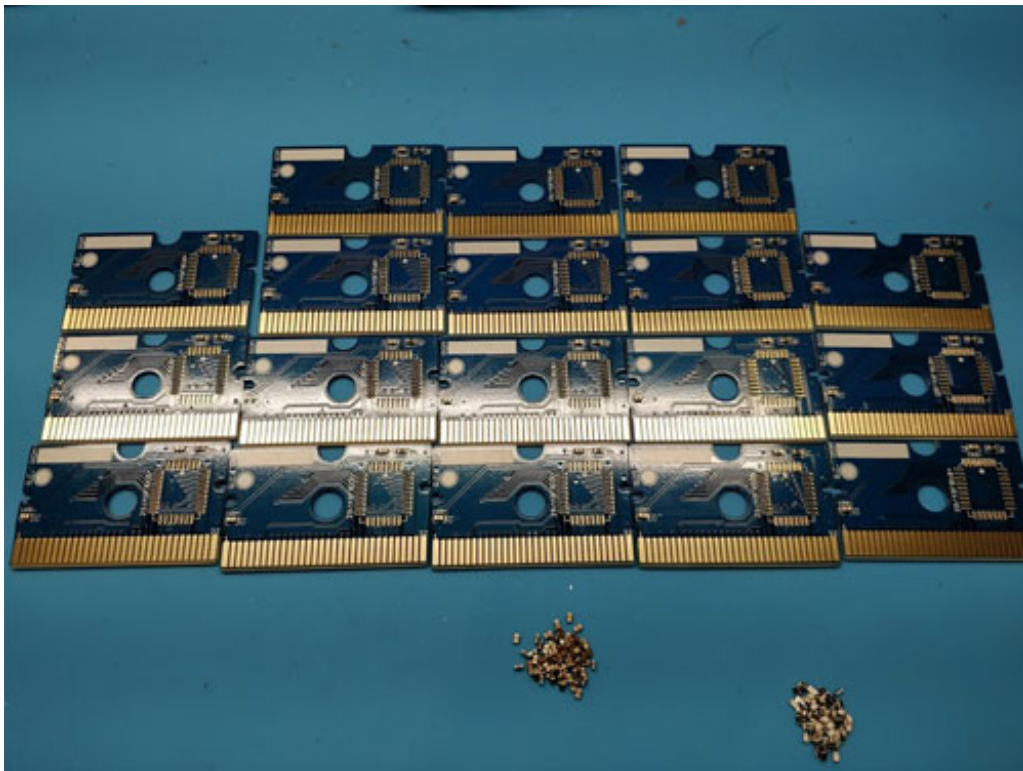


3) I usually do batches of 18-20 cards at a time. Solder paste is applied to this batch.



4) I know have to manually pick the capacitors and resistors, and place them on the cards.

I then place the Flash Memory chips onto each card.



5) The cards are inserted into a oven rack to do the reflow.

We literally “bake” each card to solder each chip on the cards.

I can now pick each PCB, and flash the game ROM onto the memory chip.



6) The PCB is then placed into an aftermarket game cartridge shell, with a beautiful label made by a professional printer.



7) After a thorough testing to check that the cartridge works perfectly on actual hardware,
the game is now ready to get shipped to the player who ordered it!

Achieving your childhood dreams

After all these technical information, I'd like to give you a **more personal feedback** about what this project means to me. For context, the most inspirational talk I've ever seen is "Really achieving your childhood dreams" by Randy Pausch. I highly recommend you to watch it. It's a brilliant lesson on how to take life with a sense of wonder, even in the darkest days. To have a meaningful life, Randy Pausch invites us to try to achieve our childhood dreams and to enable the dreams of others.

On the top section of my personal list of childhood dreams sits *"To create a video game for the Nintendo Game Boy."* The Game Boy was my first video game console, and it will forever hold a dear place in my heart. For years, I told myself that someday I would create a game for it.

But I always had “more important things to do”, such as working on a project that can actually bring in money to pay the rent. Then, one day I’ve stumbled upon the [BitBitJam](#), a game jam to create games running on retro hardware. That’s when I remembered the words of the wise Randy Pausch: **what if, for one time, I’ll try to do a project with no other expectations than fulfilling one of my childhood dreams?**

So, about a month before the Jam started, I began to look into the [various tools](#) to create Game Boy games. Then, during the Jam, I spent an entire week (*I was on vacation*) to make a [prototype version of Sheep It Up!](#) After the Jam, I kept working on the game, polishing it, refining it, optimizing it, redrawing it... until I was satisfied with the result. Meanwhile, I was also looking for a way to create actual cartridges for the game. There, I was lucky enough to meet with Catskull who were able to build them, and also had the secret desire to start a homebrew games publishing label. In the end, I now have created and released an actual Game Boy game, available on physical cartridges. But why did the cartridges matter so much? Couldn’t I have stopped at distributing an electronic version of the game online, like many other homebrewers do?

Remember, my real goal wasn’t to release a game, but to achieve a childhood dream. And **in my dream, I can play the game cartridge I made on my Game Boy**, like I did with all the other games I bought when I was a kid. So, here’s a video of me testing the game on the actual Game Boy my parents got me for Christmas when I was 9 years old:

As you can see, my childhood console is quite beaten up: many pixels columns are no longer displaying, the buttons are clearly worn, and there is a big vertical scratch on the screen. But it still works. **I can’t put words on the feeling of playing a game YOU created on the actual console you had so many good times with as a kid.** All I can I say is that, at this very moment, the 9-years-old version of myself was really proud of the grown-up he became. And that feeling alone is the best outcome I could have excepted from such a project.



Still working after 25 years, and running a game manufactured in 2017!

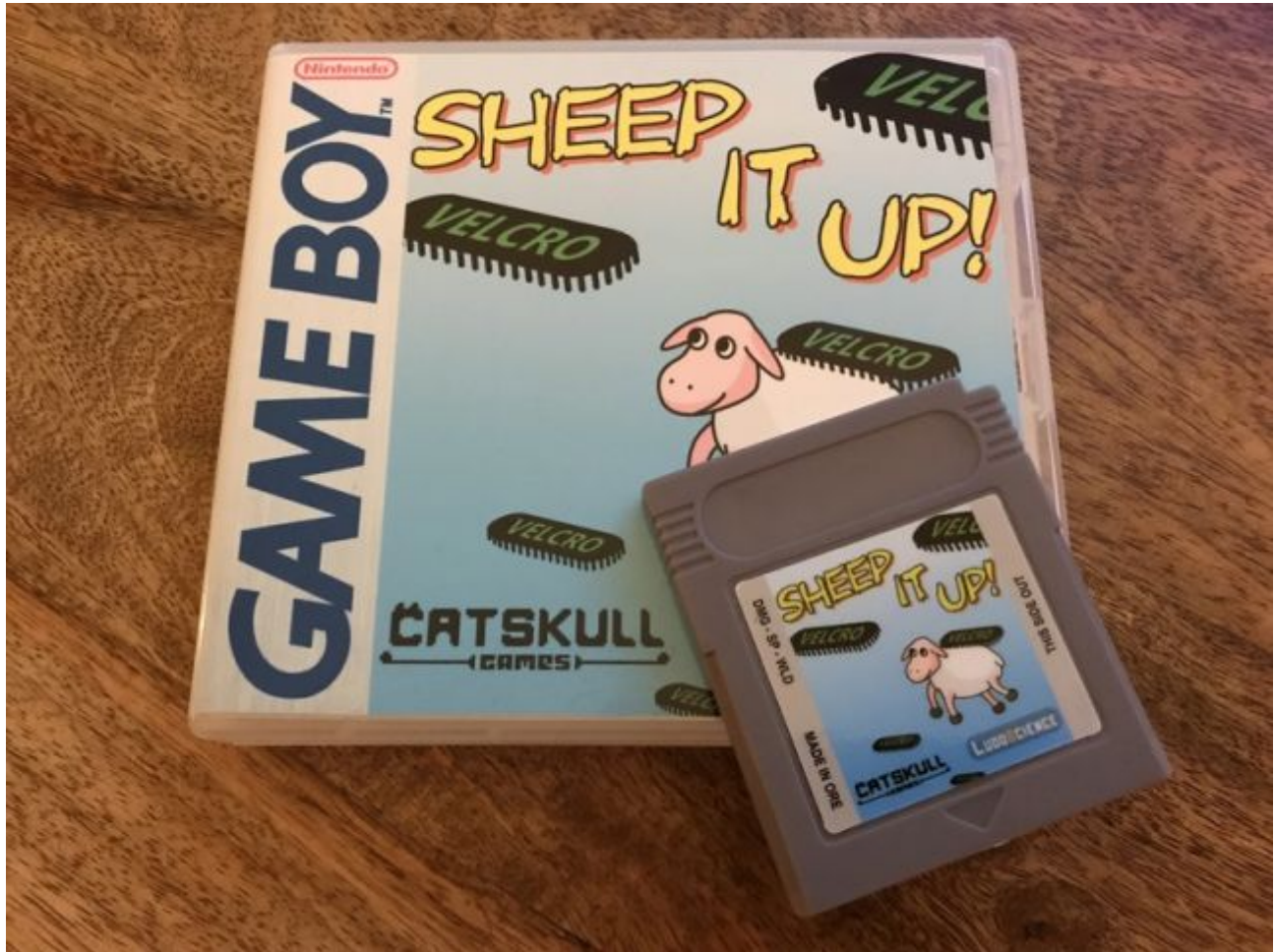
Sharing the dream : players feedback

So, to create a game primarily for your own enjoyment is the best. But actually, it can still **become even better if other people are enjoying it too**. At the end of the project, I knew that *Sheep It Up!* was a simple game, so my expectations weren't really high in this area. But the player feedback I received did manage to surprise me anyway!

Indeed, despite (or thanks to) this simplicity, **players who tried the game really enjoyed it**. The gameplay is simple enough to understand how to play in a second, but then you'll quickly get hooked to do "one last run" and try to beat the highscore. By-the-way, **my current highscore is 147!** But what I love most about the retro gaming community, is that the key drive for most players is passion. And that can lead to some really nice surprises.

For starter, several people complained that the game wasn't available in a "boxed version." As I collect retro games in loose cartridges (*both for money and shelf space constraints*), a boxed

version didn't cross my mind at first. But now, I realize that it would have been a good idea to offer it to people who asked for it. But one fan of the game, Cyph, went the extra mile, and built his own custom-made game box by recycling a Nintendo DS jewel case. I don't know for you, but I find the result totally awesome. For those who are interested, you can make your own box at home too. You simply need to pick a Nintendo DS jewel case, remove two tiny plastic pieces inside it, and insert this beautiful cover designed by Cyph



A beautiful custom box for Sheep It Up! created by Cyph, a player of the game, using a Nintendo DS case!

Another funny fact about the game is that it was **showcased during one of the largest video game convention in the world**: the Paris Game Week (300.000 visitors, more than E3 and Tokyo Game Show). So, how did an amateur game released on Game Boy get showcased in the same event as the latest AAA games in the industry? Well, for this opportunity all my gratitude goes to OrdiRetro. It's a French association dedicated to video game history, and more specifically to the brand new games released for old consoles and computers. They had a booth at this event to showcase a selection of games. Thanks to a friend of mine, Julian Alvarez, they heard about the game and proposed me to display it during the event.



Sheep It Up! being played on a Super Game Boy during the Paris Game Week 2017 - thanks [OrdiRetro!](#)

But such an opportunity can also happen without you knowing anyone in particular. Right after release, the game was bought by two Game Boy enthusiasts from Germany, who are running the [DMGpage](#) website. But instead of simply buying the game, they actually showed it during a retrogaming convention, the Retro-Börse in Hessen. So players were actually able to play my game on a Super Game Boy in a German convention, thanks to two Game Boy fans - again, I'm very grateful for that!



Sheep It Up! demonstrated on a Super Game Boy during the Retro-Börse in Hessen - thanks [DMGpage!](#)

Conclusion

Making homebrew games for retro consoles isn't about money nor popularity. Very few copies of your game will be produced. You can usually hope for an audience of several dozens of people, or at best a couple of hundreds if your game is really good. But you can be certain that these players will have a genuine interest for your game. They will be driven by the same passion for retro hardware that you have. Sometimes they will even be more passionate about it than you. So, **if you enjoyed this two-parts article about making a Game Boy game today, why not give it a try yourself?**

As you'll see in [part 1](#), there are many easy to use tools to create a game for the most iconic handheld console in history. Sure, it won't pay your rent, but you'll have a lot of fun doing it. And maybe, you'll also offer an equally fun experience to the people who will play your game!



Well, I really hope that you enjoyed this post-mortem, and that it will motivate you to try your hand at retro game development. Meanwhile, if you still have a Game Boy laying around, don't forget that you can **buy a beautiful Sheep It Up cartridge for a mere \$15** from Catskull games. Each cartridge will be assembled by hand before being sent to you!

And if you want to be informed when I release my next game, or my next article, you can sign up to my newsletter! To give you a taste of things to come, my favorite console ever is the Super Nintendo. So, who knows, maybe next time we will discuss how to “*play with Super Power!*”

Related Jobs

Galvanic Games, Inc — Seattle, Washington, United States

[06.22.18]

Multiplayer Game Engineer

Armature Studio — AUSTIN, Texas, United States

[06.22.18]

SENIOR GAMEPLAY PROGRAMMER

Square Enix Co., Ltd. — Tokyo, Japan

[06.21.18]

Experienced Game Developer

innogames — Hamburg, Germany

[06.21.18]

QA Engineer/ Software Engineer in Test

[View All Jobs]



