nesdoug

1. Getting Started

This is what you will need to program an NES game…

1. An assembler
2. A tile editor
3. Photoshop or GIMP (or similar)
4. Notepad++ (or similar)
5. A good NES emulator
6. A tile arranging program

For my examples I will exclusively work with cc65. It is part of a suite of software, and is one of the best compiler/assember programs available for 6502 (ie NES) programming. Although, the learning curve is a bit steep, I will help get you started.

http://cc65.github.io/cc65/ (http://cc65.github.io/cc65/)

(Click on Windows snapshot)

NOTE: the version that I use is V2.15 (type 'cc65 – -version' in command line to see your version). Files from different versions of cc65 are incompatible. If you have a different version, you will have to use the nes.lib included with your version, not the one I included.

Second, you need a tile editor to create the graphics. I personally prefer YY-CHR. You can get it here…

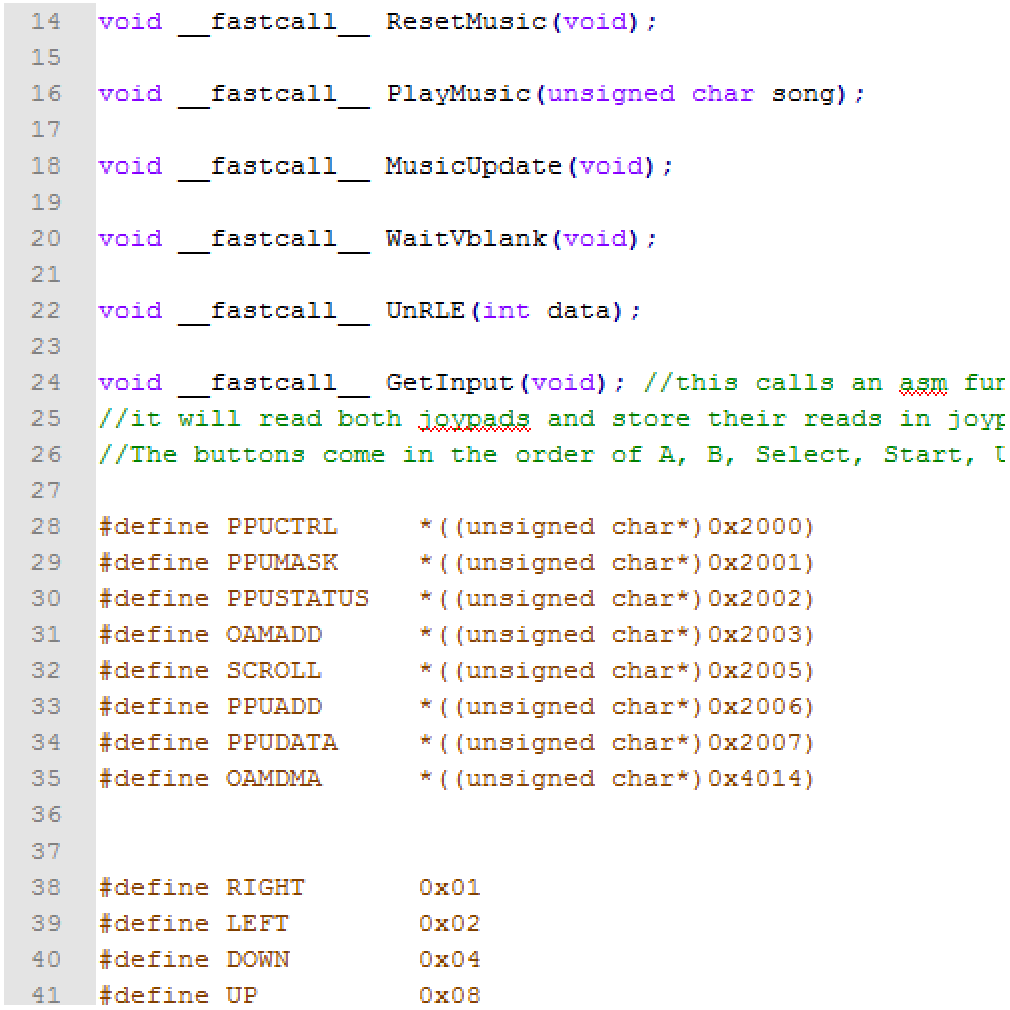
http://www.romhacking.net/utilities/119/ (http://www.romhacking.net/utilities/119/)

I prefer to work first in Photoshop, and then transfer over to YY-CHR later. There are numerous freeware programs that do the same things. I've heard good things about GIMP (but I haven't used it yet).

Notepad++ is a tool for writing programming code. You could use Wordpad or some other program, if you like. Notepad++ is available here…

https://notepad-plus-plus.org/download/ (https://notepad-plus-plus.org/download/)

The great thing about Notepad++ is you can set it to highlight your code, which makes it easier to read. And it has the line # on the left. If you get error messages while compiling, it will tell you the line of the error, which you can easily see here…



```
void __fastcall__ ResetMusic(void);

void __fastcall__ PlayMusic(unsigned char song);

void __fastcall__ MusicUpdate(void);

void __fastcall__ WaitVblank(void);

void __fastcall__ UnRLE(int data);

void __fastcall__ GetInput(void); //this calls an asm fur
//it will read both joypads and store their reads in joyp
//The buttons come in the order of A, B, Select, Start, U

#define PPUCTRL     *((unsigned char*)0x2000)
#define PPUMASK     *((unsigned char*)0x2001)
#define PPUSTATUS   *((unsigned char*)0x2002)
#define OAMADD      *((unsigned char*)0x2003)
#define SCROLL      *((unsigned char*)0x2005)
#define PPUADD      *((unsigned char*)0x2006)
#define PPUDATA     *((unsigned char*)0x2007)
#define OAMDMA      *((unsigned char*)0x4014)


#define RIGHT       0x01
#define LEFT        0x02
#define DOWN        0x04
#define UP          0x08
```

And, next is an NES emulator. I use FCEUX 90% of the time because it has excellent debugging tools, PPU viewer, Nametable viewer, Hex Editor, etc. However, it is not the most accurate emulator. You may wish to test your game on multiple emulators, to assure that other people will be able to play your game without issues. (I've heard that Nintendulator, Nestopia, and puNES are more accurate.) For one thing, the default palette is way off. Go to Config/Palette…and set to NTSC emulation (or load a custom palette).

FCEUX is here…

http://www.fceux.com/web/download.html (http://www.fceux.com/web/download.html)

* here's where you can get a custom palette — FirebrandX has been working on making a better NES palette. What's wrong with FCEUX's default palette? Too bright and too saturated — not accurate to what it would look like on an actual NES. (FCEUX / Config / Palette / Load Palette).

http://www.firebrandx.com/nespalette.html (http://www.firebrandx.com/nespalette.html)

I don't see the exact palette I mentioned anymore, but I think it was this one…

http://dl.dropboxusercontent.com/s/y3yeaqc87dnhqel/FBX-NES-Unsaturated.pal (http://dl.dropboxusercontent.com/s/y3yeaqc87dnhqel/FBX-NES-Unsaturated.pal)

And, lastly, the tile arranging program. We can make a game without it, but it will definitely be helpful. Since we are working on an NES game, I highly recommend, NES Screen Tool. It shows the NES color limitations very well, and is good for making single screen games. It also gives you nametable addresses and attribute table addresses, which comes in handy. Get it here…

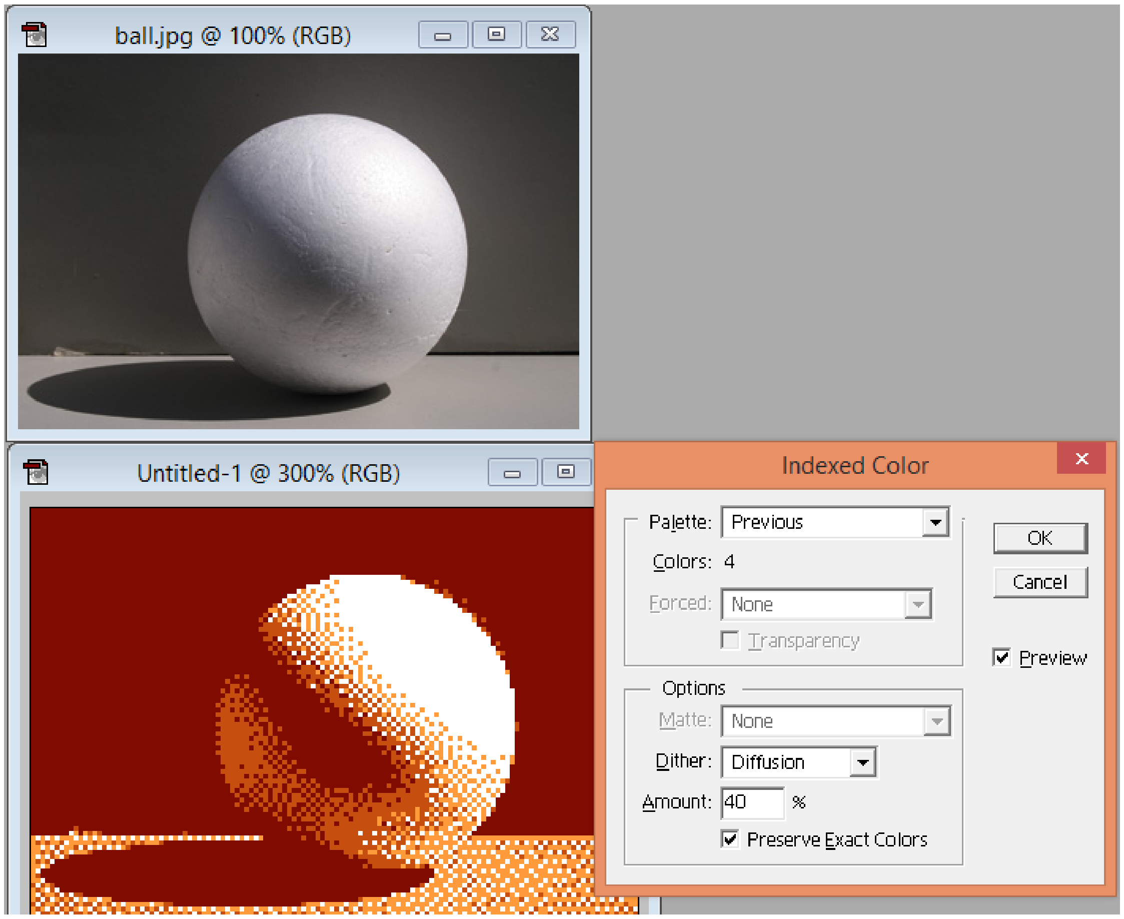
https://shiru.untergrund.net/software.shtml (https://shiru.untergrund.net/software.shtml)

And, if you are making a scrolling game, I would also pick up Tiled map editor.

http://www.mapeditor.org/ (http://www.mapeditor.org/)

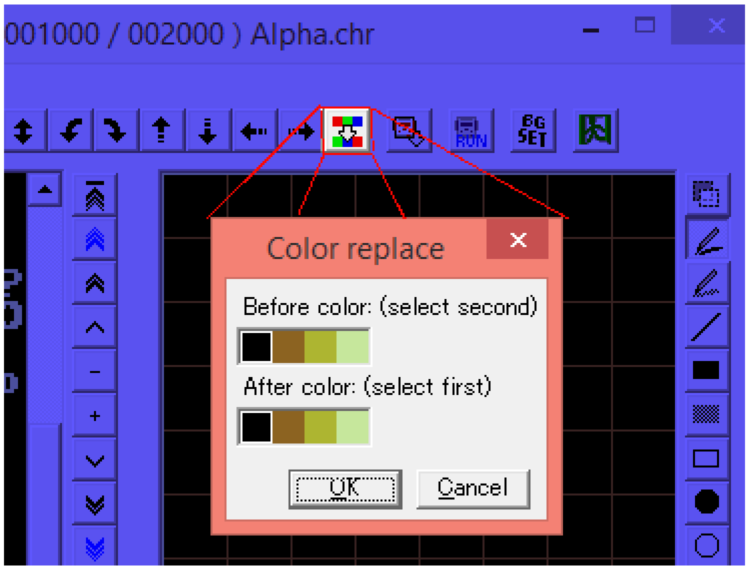
Now that that's done…how do these things work?

Photoshop – to prepare files to go to YY-CHR. First, resize to some reasonable NES size, here I'm using 128 pixels wide (use nearest neighbor for resizing). Then reduce to 4 colors, by Image/Mode/Indexed… Palette:Custom, reduce to 4 colors. (I made a custom 4 color swatch set, that can be loaded here.) You may have to touch up the image with the pencil tool. Cut and paste into YY-CHR.



Palette: Previous
Colors: 4
Forced: None
Transparency
Preview
Options
Matte: None
Dither: Diffusion
Amount: 40 %
Preserve Exact Colors

YY-CHR – make sure it's set on 2bpp (NES). You may have to use the color replacement tool, in YY-CHR, if it got the color indexing wrong…


001000 / 002000) Alpha.chr
Color replace
Before color: (select second)
After color: (select first)
OK
Cancel

It doesn't matter what palette settings you give it here. YY-CHR can show you various color options, but doesn't save the palette. You will have to program the palette into your game.

cc65 – will take some explaining…next time.

November 15, 2015April 16, 2017 dougfraker

2 thoughts on "1. Getting Started"

1. **Matt** says:
   April 23, 2018 at 3:45 am Edit
   I downloaded CC65-master, but the package is just a bunch of folders and files – No application that I would use to type in and compile code. I'm using windows 7. Am I doing something wrong?

   Reply

   dougfraker says:
   April 23, 2018 at 11:27 am Edit
   You should have a 'bin' folder with 12 exe files (ar65, ca65, cc65,…). These are console applications, they don't have an interface. You have to type your program in a text editor, like notepad.

   Reply