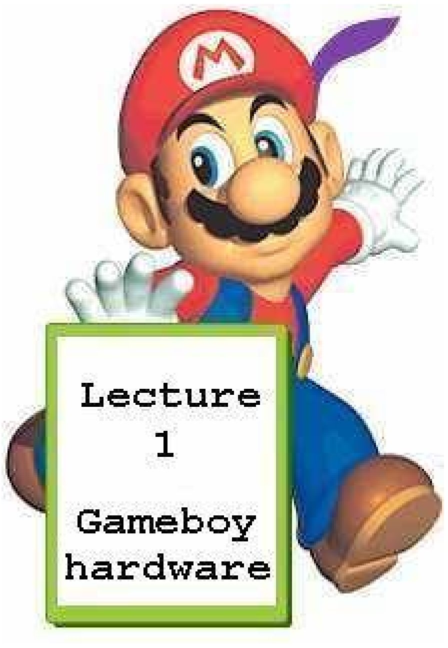


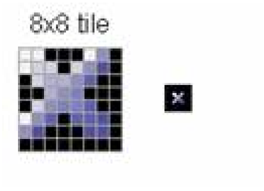
Gameboy Advance (GBA)

GBA has a 32 bit ARM7TDMI processor (RISC) with a cpu speed of about 16.67 MHz. It has also got its own graphics processor to support the main cpu . The gba includes a Z80 processor which is used to run games for the gameboy and gameboy color (this is not the subject of these lessons). GBA has 96Kb video memory (VRAM), 32Kb fast internal memory, and 256Kb external memory. The screen resolution is 240x160 pixels in all modes except mode 5, and can show a maximum of 32768 colors.

Graphic modes

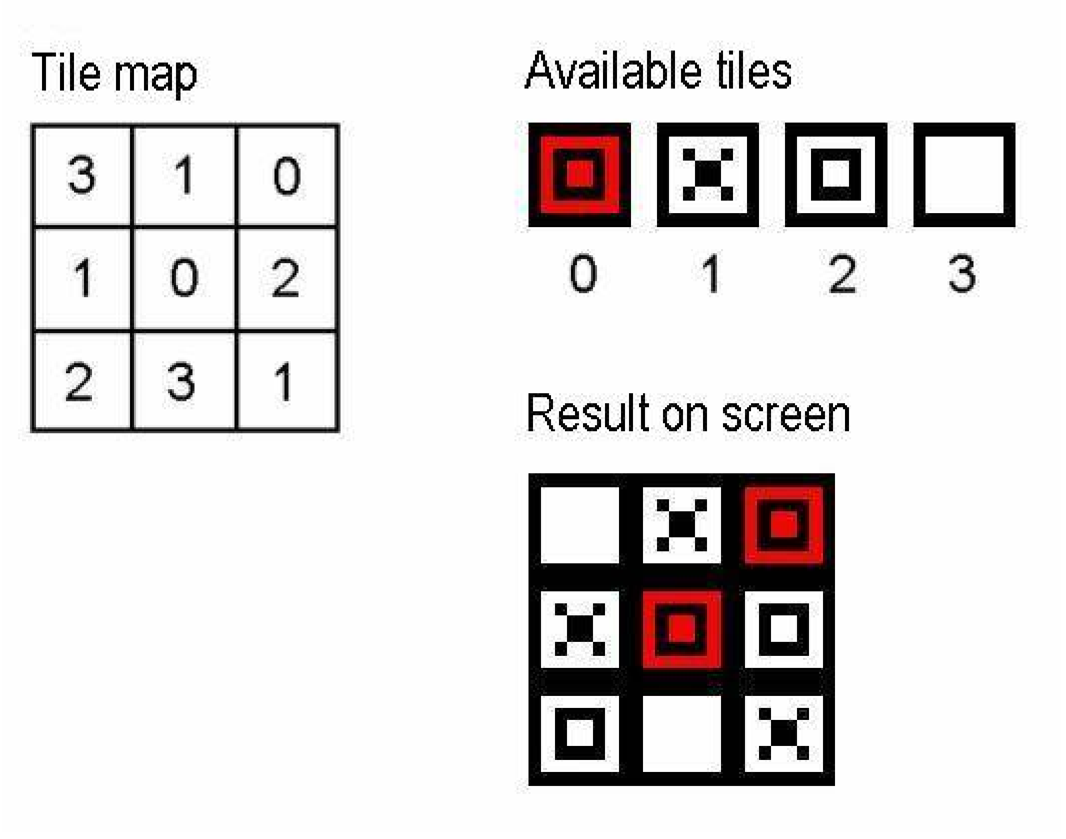
GBA has 6 different graphics modes. Here comes a description of each of these modes, but first a brief explanation. Mode 0, 1 and 2 are called tile modes, which means these modes are built up from small squares (tiles) of 8x8 pixels that make up the background(s) in the different modes. Mode 3, 4 and 5 are bitmapped modes, which means they give the programmer the opportunity to write directly to a pixel on the screen. This can be done in two ways: by writing a 15 bit color code to the address that represents the specific pixel on the screen, or input a reference to a color stored in a palette.

Tiles



Tile backgrounds are made up from tiles. Tiles are small, self made squares of 8x8 pixels size. These tiles can either be in 16 or 256 colors.

What I refer to as a tile map (see top next page) is actually a 2D table that consists of numbers that reference the tiles that we want to place at the chosen location in the table. Tiles can be used more than once in the same tile map, and they can be turned up side down and sideways. Because of this we won't need that many different tiles to create a background. Tile maps can also be rotated and scaled, but not all modes have the same possibilities.



Sprites

Sprite characters are animations like this smurf that you will find in most games. These characters can move freely over the screen without destroying any of the background like any other graphics would have done. This is the reason we almost always want to use sprites when programming games.

The sprites are made by hardware (the graphics processor) so we don't need to think about what actually happens when we are using sprites. We just specify the size of the sprite, how many colors its going to have (16 or 256), if its going to be rotated, turned upside down, sideways or scaled.

Mode 0:

1 All 4 available backgrounds
2 No rotation or scaling
3 16 or 256 colors
4 Maximum 1024 tiles

Mode 1:

1 Only background 0, 1 and 2 are available
2 Background 2 supports rotation/scaling
3 256 colors
4 Maximum 256 tiles

Mode 2:

1 Only background 2 and 3 are available
2 Both backgrounds support rotation/scaling
3 256 colors
4 Maximum 256 tiles

Mode 3:

1 Only background 2 is available
2 240x160 pixels
3 32768 colors

Mode 4:

1 Only background 2 is available
2 240x160 pixels
3 Supports double buffering
4 256 different color entries from a palette made up from 32768 colors

Mode 5:

1 Only background 2 is available
2 160x128 pixels
3 Supports double buffering
4 32768 colors

How can you program the GBA ?

To be able to program the GBA you have to know C/C++ programming, and you need to have a special C compiler or assembler made for the gba cpu. Its probably easiest to use a C compiler for programming, but if you want to get the most out of the gba hardware you would be better of with the assembler. In these lectures I will only use a C compiler, and that's what I will suggest for anyone who is new to handheld development. And by the way I haven't heard of anyone who solely programs their games in assembler. Usually parts or functions are rewritten in assembler if they need to be very fast, but the rest of the program is written in C/C++. Nintendo is selling a C/C++ compiler, but I think it costs about $5000. A free unofficial compiler (DevKitAdv) is available at http://devkitadv.sourceforge.net/index.html

When you have downloaded DevKitAdv you can just unzip it to c:\.DevKitAdv To write your code you can use notepad, and save your program as 'myprogram.c'. Always remember to press enter after the last '}' in your code or the compiler will complain.

To be able to compile the C code you have written you would need to run a .bat file that starts the compiler.

Recipe: How to make a .bat file

1 Open notepad
2 Write this:

```
path=c:\devkitadv\bin
gcc  -o Test.elf Test.cpp  -lm
objcopy -O binary Test.elf Test.bin
```

3 Store this file as Make.bat

I have chosen to call my program Test.c, that is why I have the name Test in the Make.bat file. If you want to change the name of your C program to something else, you will have to change the name in the Make.bat file. Don't use space or foreign letters in the program names or path names, this can be a problem for the compiler. Its also very important that you write the correct path to the \bin folder in DevKitAdv, or the compiler will not work.

Testing your programs (emulators)

To test your programs, the fastest and most used method is to use emulators that emulate the gba hardware. There are lots of free emulators on the internet, VisualBoyAdvance and mappy are the most used.

If you put VisualBoyAdvance in the c:\devkitadv\bin folder, and put the line (VisualBoyAdvance test.bin) at the end of the Make.bat file, your program will compile and execute every time you run Make.bat.

Tools

Lots of tools have been made for converting pictures from different picture-formats used on computers to the formats that the gba understands. These tools can be specially made for making bitmapped pictures, sprites, tiles, or even tile maps. These are some of my favorites.

3 gifs2sprites
4 gfx2gba
5 bimbo (bitmap editor)
6 GBAMapEditor (tile map editor)

Tiles and sprites are made exactly the same way and can often use the same converter.

Transferring programs to hardware

If you have made a program/game that runs on the gba emulator its a lot more fun to se it run on real hardware than to see it on a big monitor on the computer. To accomplish this you need a flash cart

This is a flash cart.

You can connect the flash cart to a special box called a linker when you want to program it. The linker is connected to the PC and a program on the PC lets you send programs to the flash cart connected to the linker. You can then put the flash cart into your gba and see the result on your gba hardware. Some flash carts can be programmed while connected to the gba hardware, through a cable in the linker port of the gba.

A cheaper alternative is to buy an MBV II cable that is designed to transfer small programs directly into the gba memory. This cable must be connected to the linker port of the gba, and cannot be removed after you have transferred the program to the gba. This cable uses the internal memory of the gba and the biggest sized programs you can transfer with an MVB II is 256Kb (actually about 240Kb).

This is what the MBV II looks like. It connects to the parallel port on the computer, there is a micro controller inside the parallel connector, and it comes with special software to transfer programs from the computer to the gba.

Information on the internet

There is an IRC channel especially for gba development on EFnet, called #GBADEV. On this channel people are very helpful and have good knowledge of the gba hardware. Tutorials and lectures about gba programming are available online, have a look at www.gbajunkie.co.uk, or PERN project at http://www.thepernproject.com/index2.htm.