LAB 07:
Sprites

Provided Files

- `main.c`
- `gba.c, gba.h`
- `game.c, game.h`
- `mode4.c, mode4.h`
- `sprites.c, sprites.h`
- `background.c, background.h`
- `print.c, print.h`
- `spritesheet.bmp`
- `Makefile`

Files to Edit / Add

- `main.c`
- `game.c`
- `sprites.c`
- `spritesheet.c, spritesheet.h`
- `.vscode with tasks.json`

Instructions

In this lab you will set up multiple sprites to create a simple scene in Mode 4. Your code may not compile until you complete an entire TODO block, at which point the game should compile with a new component of the final outcome (unless otherwise specified).

TODO 1: Setting up sprites

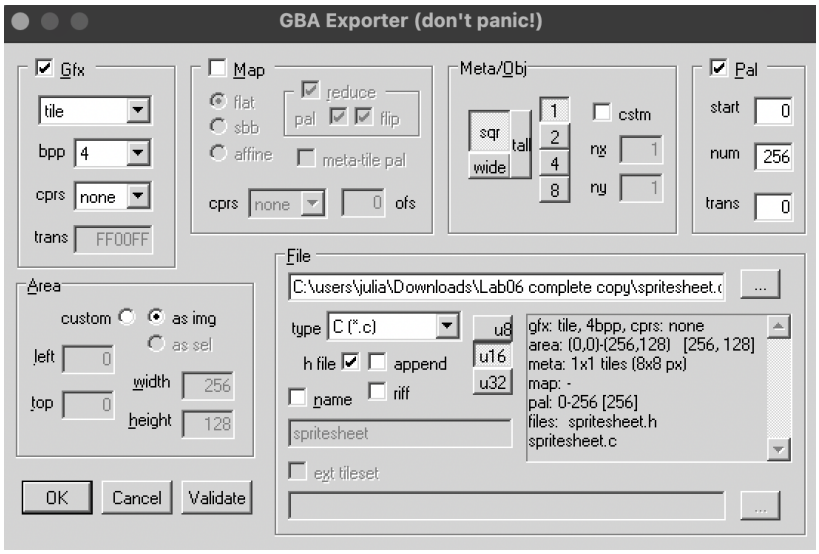
TODO 1.0

- Open the provided `spritesheet.bmp` in Usenti.
    - This is a 256x128p image since we're going to be using sprites in a bitmapped mode (Mode 4) and we're going to be using 4bpp sprites.
- The palette has already been set up for you. If you press the "16" button on the top left corner of the palette, you can switch to 4bpp mode for the palette. Then,

you can click on each palette row to see what all of your sprites would look like using that palette row.

TODO 1.1

- Next, let's export this as something the GBA understands.
- Export your image (**Image > Export**).
- You should save it as a ***GBA source file*** in the Lab06 folder.
- You want your export settings to look something like this:



TODO 1.2

- At the top of `main.c`, include the `spritesheet.h` that Usenti just created.

TODO 1.3

- In `goToGame()`, use `DMANow` to load the tiles that Usenti created for you into the appropriate location in memory.
    - *Hint:* What Mode are we using for the lab? How does that affect our destination?
    - *Hint:* spritesheetTilesLen gives the length of the array in bytes. What do we know about the type of data DMA transfers? Consider this for the number of transfers you want to complete.

TODO 1.4

- Next, use `DMANow` to load the palette that Usenti created into `SPRITE_PALETTE`.
    - *Hint:* spritesheetPalLen gives the length of the array in bytes. What do we know about the type of data DMA transfers? Consider this for the number of transfers you want to complete.

TODO 1.5

- Navigate to `sprites.c` and complete `hideSprites()`.

    - This function should set *all* sprites in the shadowOAM to be hidden.
        - *Hint:* Which OAM attribute holds the rendering mode?
    - Looking at (and using) the macros in `sprites.h` should help with this!

TODO 1.6

- Back in goToGame in `main.c`, use `DMANow` to copy the shadowOAM into the OAM once we've waited for the vBlank.
    - *Hint:* Consider the maximum number of sprites, how many attributes each has, and the size of each attribute to determine the number of transfers.

TODO 1.7

- In `initialize()`, enable sprites in the display control register.

Build and run your program. You should see just the background, no sprites yet.

TODO 2: Kitty sprite

TODO 2.0

- Go to `game.c`, and in `initKitty()` set the sprite's oamIndex, or where the sprite's information will be stored within the shadowOAM.
    - You can set the oamIndex to any number, but for this lab the kitty **needs to use index 0**. However, it's good practice to start from 0 in any case.

TODO 2.1

- Also in `game.c`, in `updateKitty()` update the sprite's direction when the left or right buttons are pressed.
    - `kitty.direction` is initialized to `RIGHT`, and should always be either `LEFT` or `RIGHT`.

TODO 2.2

- Go to `drawKitty()` and set up the kitty sprite in the shadowOAM.
    - Use the oamIndex struct member you initialized as the index!
    - **Look at `sprites.h` to see all the macros we created to help you with setting the attributes!**
    - Attribute 0: the kitty's y position, shape, and whether it's hidden or regular.
        - Look at the spritesheet to determine shape and size for your sprites.
        - **Do not include the shadow underneath the kitty!** We will be using a separate sprite for this.
    - Attribute 1: the kitty's x position, size, and horizontal flipping enabled *only if the kitty is facing right*.

- Attribute 2: the palette row using `kittyPalette` as the index, the tile ID, and priority 1.
  - *Hint:* What needs to be done to the tile ID when we're using sprites in a bitmapped mode?
  - We're using priority 1 so that this sprite appears behind another sprite.

TODO 2.3

- Still in `drawKitty()`, set up the kitty's shadow sprite in the shadowOAM.
  - Use 1 as the shadowOAM index since it must be higher than the kitty.
  - Look at `sprites.h` to see all the macros we created to help you with setting the attributes!
  - Attribute 0: the shadow's y position and shape.
    - The shadow's y position should always be `kitty.y+kitty.height-4`.
  - Attribute 1: the shadow's x position (which is the same as the kitty's), size, and horizontal flipping enabled *only if the kitty is facing right*.
  - Attribute 2: the palette row using `kittyPalette` as the index, the tile ID, and priority 1.
    - *Hint:* What needs to be added to the tile ID when we're using sprites in a bitmapped mode? Think about the OVRAM.
    - We're using priority 1 so that this sprite appears behind another sprite.

Build and run. You should now see the kitty and its shadow, and they should always move together. Pressing the left/right buttons should make the kitty face the appropriate direction.

TODO 3: Statue sprite

TODO 3.0

- In `drawStatue()`, set up the statue's sprite in the shadowOAM.
  - statue.oamIndex has already been initialized to use as the index of the shadowOAM.
  - Attribute 0: the statue's y position and shape.
  - Attribute 1: the statue's x position and size.
  - Attribute 2: palette row 3 and the statue's tile ID.
    - By default, priority is 0 which is what we want here. This is so the kitty can go "behind" the statue!

Build and run. You should now see the statue in the center of the screen. It should never move, and it should be drawn on top of the kitty's sprite.

TODO 4: Paw sprite

TODO 4.0

- In `drawPaw()`, set up the paw's sprite in the shadowOAM.
    - paw.oamIndex has already been initialized to use as the index of the shadowOAM.
    - Attribute 0: the paw's y position and shape.
    - Attribute 1: the paw's x position and size.
    - Attribute 2: palette row 4, the paw's tile ID, and priority 1.
- Look at how we're using the `collided`, `kittyPalette`, and `kitty.framesPassed` variables to cycle through palette rows when there is a collision between the paw and the kitty.

Build and run. You should now see the paw by the bushes on the bottom right of the screen. When the kitty collides with the paw, it should change color every 30 frames. If it collides with the statue while already changing colors, the kitty will stop at whatever color it was when the collision occurred.

Submission Instructions

Ensure that **cleaning** and building/running your project still gives the expected results. **Please reference the last page of this document for instructions on how to perform a "clean" command.**

Zip up your entire project folder, including all source files, the Makefile, and everything produced during compilation **(including the .gba file)**. Submit this zip on Canvas. Name your submission Lab07_LastnameFirstname. For example:

> "Lab07_KittyHello.zip"

It is *your* responsibility to ensure that:

- The ZIP file is named correctly,
- All the appropriate files have been submitted, and
- Your submitted ZIP can be opened and everything cleans, builds, and runs.