CS 91S: Games Systems

Assignment 7: NES Pong Remake

DUE November 15th at 11:59 PM

In this lab we will remake **PONG** for the NES (demo).

1 Pre-lab: Close-read `controller.c`

1. What's the difference between `byte` and `sbyte`?
2. What are the 4 parameters to `oam_meta_spr`?
3. What is `DEF_METASPRITE_2x2` (e.g., variable, function, macro)?
4. What is `OAM_FLIP_H` (list its type, value, purpose)?
5. What is `PAD_RIGHT` (list its type, value, purpose)?
6. Why are the `actor_` arrays declared globally?
7. What is `runseq` used for?
8. What do you like about the movement/controls?
9. How could we make it more Mario/Platformer like?
10. Name the character!

2 Bouncing Ball

Bounce a ball around the screen using `oam_spr` and pattern `0xB6`. Your ball should keep track of its position (`x` and `y`) and also its speed (`dx` and `dy`)—*what types should those variables be?* When the ball hits the edge of the screen it should bounce. At the start the ball should be centered in the screen, and when the `A` button is pressed, you should make the ball pick a random speed (`rand8`) and start moving.

3 Paddle Meta-sprites

Draw two paddles using using `oam_meta_spr`, for example a 1x3 meta-sprite composed of pattern `0x85`. Move the paddles based on the gamepads. If you go with 1x3 meta-sprites, and the ball is the first sprite, figuring out the correct sprite offset is a little tricky, but you might do something like:

```
oam_meta_spr(paddles_x[i], paddles_y[i], 4 + i*4*3, paddle);
```

Each sprite is 4 bytes, and if we have a 1x3, that's 12 bytes per paddle meta-sprite. Alternatively, you can use the return value of `oam_meta_spr` to figure out which sprite offset to use.

```
oam_id = oam_meta_spr(paddles_x[i], paddles_y[i], oam_id, paddle);
```

4 Collisions

Modify the ball bouncing so that it only bounces on the left and right if the ball touches a paddle. When doing this kind of check, it might be useful to move the ball right next to the paddle (i.e., collision ejection) so it doesn't get stuck when reversing the direction (more on collisions here):

```
ball_dx *= -1;
ball_x = paddles_x[i] + paddle_width;
```

5 Score

Add scores and display them; first player to 9 wins and the game resets.

6 Additional challenges:

0. Add some sound (see the climber example using the `sfx_play`).
1. Change the sprites of the ball and paddle.
2. Change the speed of the ball as time passes or based on the collision location with the paddle.
3. Allow a score of above 9.

7 Deliverables

1. Commit the c-source file to the repo (`pong.c`).
2. Write a small reflection (as a markdown document) about what you were able to accomplish in this mini-lab and the answers from the first section.
   - include a link to your 8bitworkshop project in the markdown document (link can be retrieved in the `share` menu).
3. Write a little about where you are in your demake.