Assumptions:

You have read Craits tutorial.

Discussion of game

The endless runner.

Discussion of Sprites

Sprites and masks

Encoding sprites

Encoding masks

Drawing a sprite

Example 1 ground, 1 dinosaur drawOverwrite showing how it erases the ground

Example 2 ground, dinosaur and mask showing how it does not erase the ground.

Making the dinosaur run.

Sprite and frames

Different size sprites.

Making the ground move.

Flintstones

Sprites

A sprite is simply an image or graphic that represents a player, an enemy or even background elements of your game and can be drawn or moved around as a single object. The Arduboy library provides a number of functions to render sprites to the screen. In this tutorial, we will concentrate on two functions drawOverwrite() and drawExternalMask().

But before we get into those, let’s recap how we define a sprite. The image below shows Steve in his upright, ready-to-run position.

The array definition for this sprite is shown below.

byte myFirstSprite[] = {

8, 8,

126, 231, 231, 129, 129, 231, 231, 126,

};

I have formatted the array to make it a little more readable. The first line contains the width and height of the array, in this case 8 pixels by 8 pixels. The remaining 8 bytes contain the pixel data for each column of the sprite and are calculated using a simple formula shown below.

Notice how I have labelled the side of the graphic with 1, 2, 4 and so on. To calculate that the first column’s value is 126, I simply added up all of the values adjacent to the pixels I want to be turned on (white). 2 + 4 + 8 + 16 + 32 + 64 = 126. The remaining columns are calculated in exactly the same way.

The sprite is drawn by simply overwriting what was already there. A bit set to 1 in the frame will set the pixel to 1 in the buffer, and a 0 in the array will set a 0 in the buffer. In the example below, the black corners of the ball are visible as the ball passes into the white area.

When rendering a sprite, bits set to 1 in the mask indicate that the pixel will be set to the value of the corresponding image bit. Bits set to 0 in the mask will be left unchanged. This can be seen clearly as the ball moves into the right hand side of the background. The top-left and bottom-right corners of the image are rendered as black as the mask is set to 1 in these areas which in turn ensures that the images pixels (both zeroes and ones) are rendered on the background.

* Dino sprite
* drawOverwrite()
* drawErase()
* drawExternalMask()
* drawPlusMask()
* drawSelfMasked()