

# Slit Scan Notes

Fred Mitchell

May 5, 2022

## Contents

<b>1</b>	<b>Slit Scan Notes</b>	<b>1</b>
1.1	Disclaimer . . . . .	1
1.2	Loading Images . . . . .	1
1.2.1	Pipelining . . . . .	2
1.3	Scanning the images . . . . .	2
1.3.1	Orign of Graphics.Image images . . . . .	2
1.4	Outputting the video . . . . .	2
1.5	Future planned enhancements . . . . .	2

## 1 Slit Scan Notes

### 1.1 Disclaimer

These are my basic notes on this project, and are not meant for general consumption, and therefore are not gauranteed to be accurate or even useful for anyone other than myself. Please see the README.org or the README.pdf for that.

### 1.2 Loading Images

- We could resize one of the images to be the exact same size/dimensions of the other. Would simplify the math quite a bit. But might cause some wierdness in some circumstances. Of course, the user can correct the issue and make sure both images are the same dimensions, etc.
- Trying to determine how to transverse the images slit-wise. In mapping the images, we are simply given the x,y coordinates. Of course, we can work backwards from there to determine the "slit", and should cover this in the math description in the README.

### **1.2.1 Pipelining**

We can manipulate the source images to be the same size, and rotate them to do a horizontal scan. For 2 images, we create each canvas separately, then combine them during the compositing phase.

Indeed, we shall take a pipelining approach, so we can run all of this in parallel to utilize all the available cores.

## **1.3 Scanning the images**

It just occurred to me that we should scan from the basis of the canvas, since that is the target, and simply do the math transforms to the source images. This way, we can handle blending / dithering / smoothing between pixels on the destination image (canvas) very naturally.

In fact, we can utilize the normal mapping function of the destination, and simply address the pixels via the index function from the sources.

### **1.3.1 Origin of Graphics.Image images**

The origin is in the upper-left corner and descends down and right. This is typical of computer graphics going all the way back to the raster scan days which almost invariably started from the upper left and scanned to the right and down.

From a mathematical perspective, I have always found this annoying, but for purposes of keeping things "simple", I will embrace that in my math here.

## **1.4 Outputting the video**

## **1.5 Future planned enhancements**