Flappy Bird, implemented in a “something-something-Script” language, but one you may not be so familiar with…

…it’s PostScript!

PostScript was developed to drive this machine—the very first laser printer. Until this time, printers printed ASCII characters. This machine was going to print high definition graphics.

But how are we going to describe those graphics?

In a dedicated programming language that is a very convenient compilation target.

But maybe not as convenient to write…

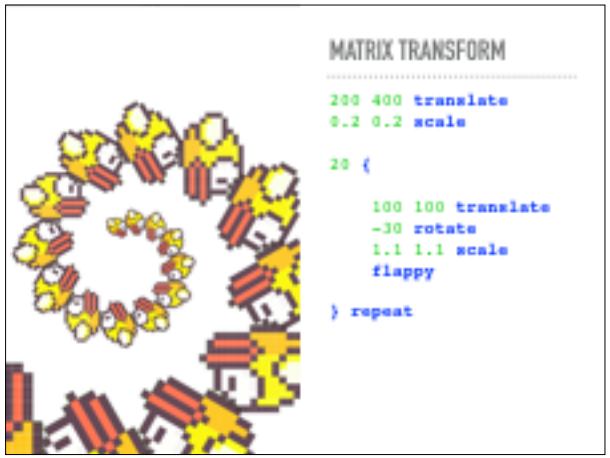
```
1 2 add 3 mul
```

$(1 + 2) * 3$

You've seen this in Uni. The operands go onto a stack, and every operator pops the top N items off the stack, and pushing the result.

PostScript completely works this way.

What is cool about PostScript?



It's made for graphics, and it shows! Stacked matrix transforms make it super easy to do any kind of graphics manipulation. And this works on all types of graphics, bitmaps, rectangles, text!

Some things are not so great…

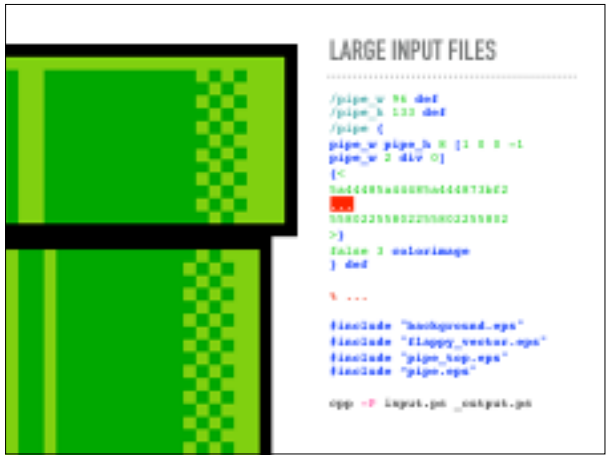
This might have been me, and there's probably a better way to do this. But I had a function here with 2 mutable local variables (on the stack, of course), which called another function to see if my Flappy was hitting a pipe.

You can see I added lots of comments to keep reminding myself of the form of the stack, so I knew where variables were at what point in time.

And finally…

Well yes, obviously PostScript doesn't take any input, because how would a printer take input?

So our input is predetermined at execution time. Our jump frames are stored in an array, and we'll jump whenever the current frame is in this array.

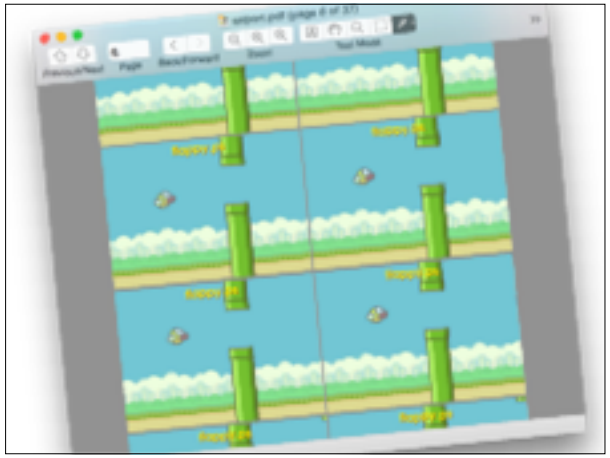


Bitmaps need to be encoded literally as a giant hex blob of pixel data.

My editor wasn't very happy with 40k lines of hex garble, so I needed to find another way. In this case, I used the C preprocessor (CPP) to pretend my PostScript was C source and piece the final source file together from fragments.

Another thorn in my side: bitmap data can't have transparency! But obviously Flappy has an irregular shape and it should have transparent pixels at the edges.

So I wrote a Python script that converts a PNG file with transparency to a command that draws a whole bunch of rectangles. One per pixel. Voila, an irregularly shaped sprite in PostScript! :)



The end result: a run of the flappy.ps script, displaying one frame per page!
