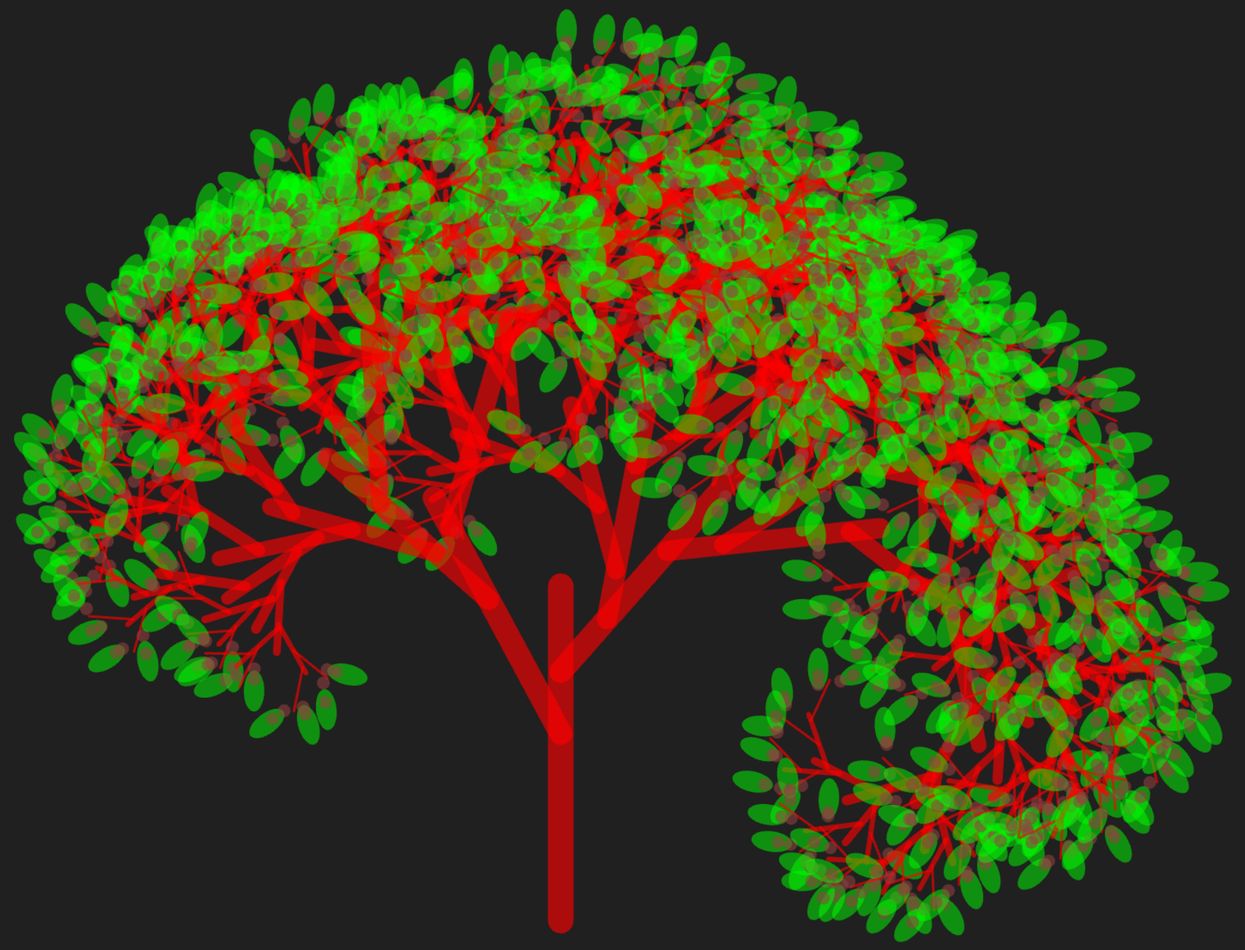
Recursion 🤯

CF1

Gabe Johnson | October 3 2024

Recursive Tree Sketch

https://editor.p5js.org/johnsogg/sketches/RHb2Wa_yT



What’s Recursion

And why are we doing it in CF1?

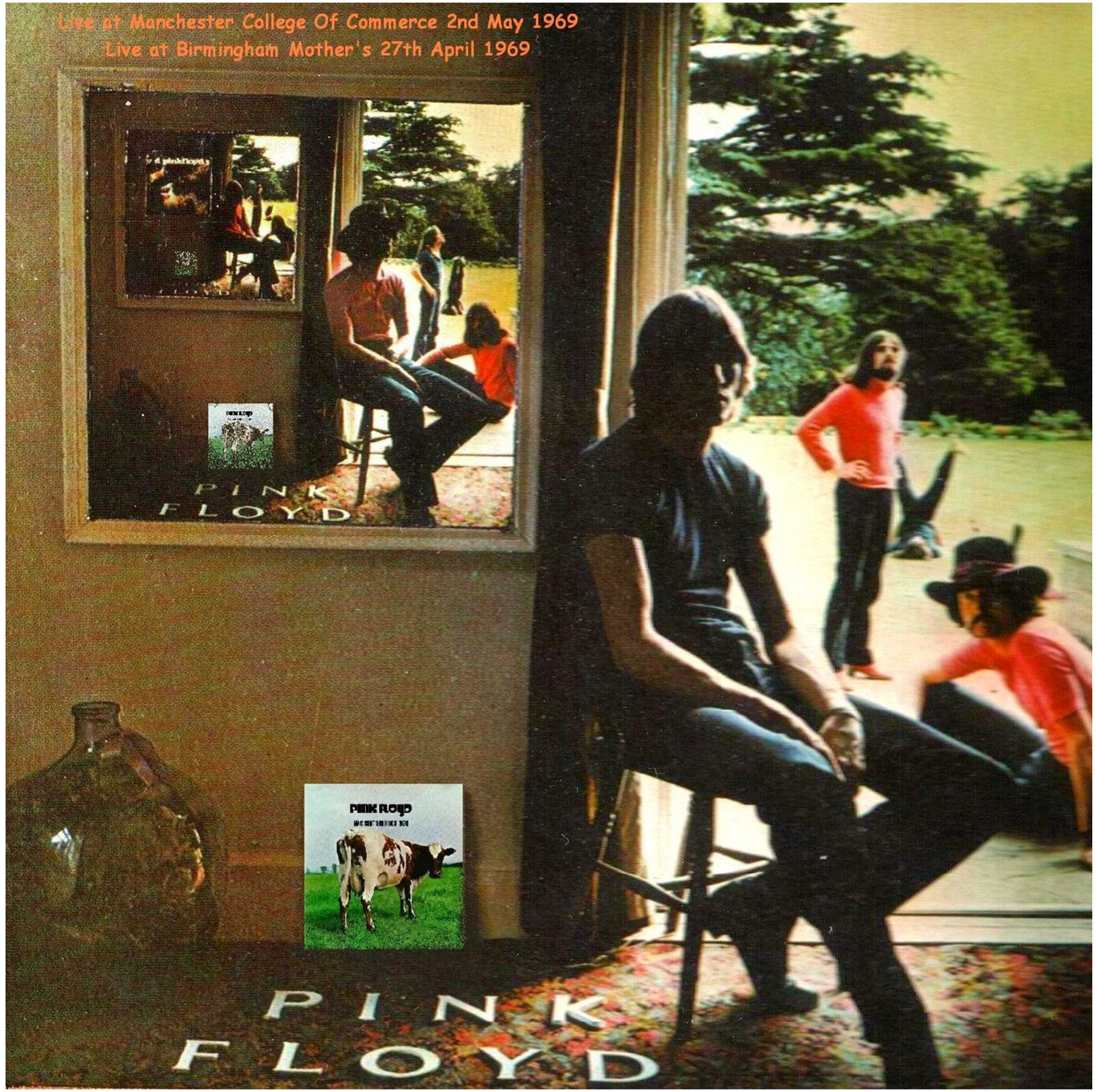
- Typically recursion is something you do in discrete mathematics and computer science
- I tell my CSCI 2270 students that recursion is one of the three topics that will likely melt their brains
- It isn’t *hard* in the sense that it isn’t hard to ride a bike.
  - Once you get it, you get it.
  - But until then, maybe wear leather if that’s your thing.

No Really, What’s Recursion

And why are we doing it in CF1?

- When a function invokes itself, that’s recursion!
- Use it when you have “self similarity”
- Be careful to provide a stopping condition
- … otherwise you’ll recurse forever, kinda like when you make an infinite loop


PINK
FLOYD
GIGI
PINK
FLOYD

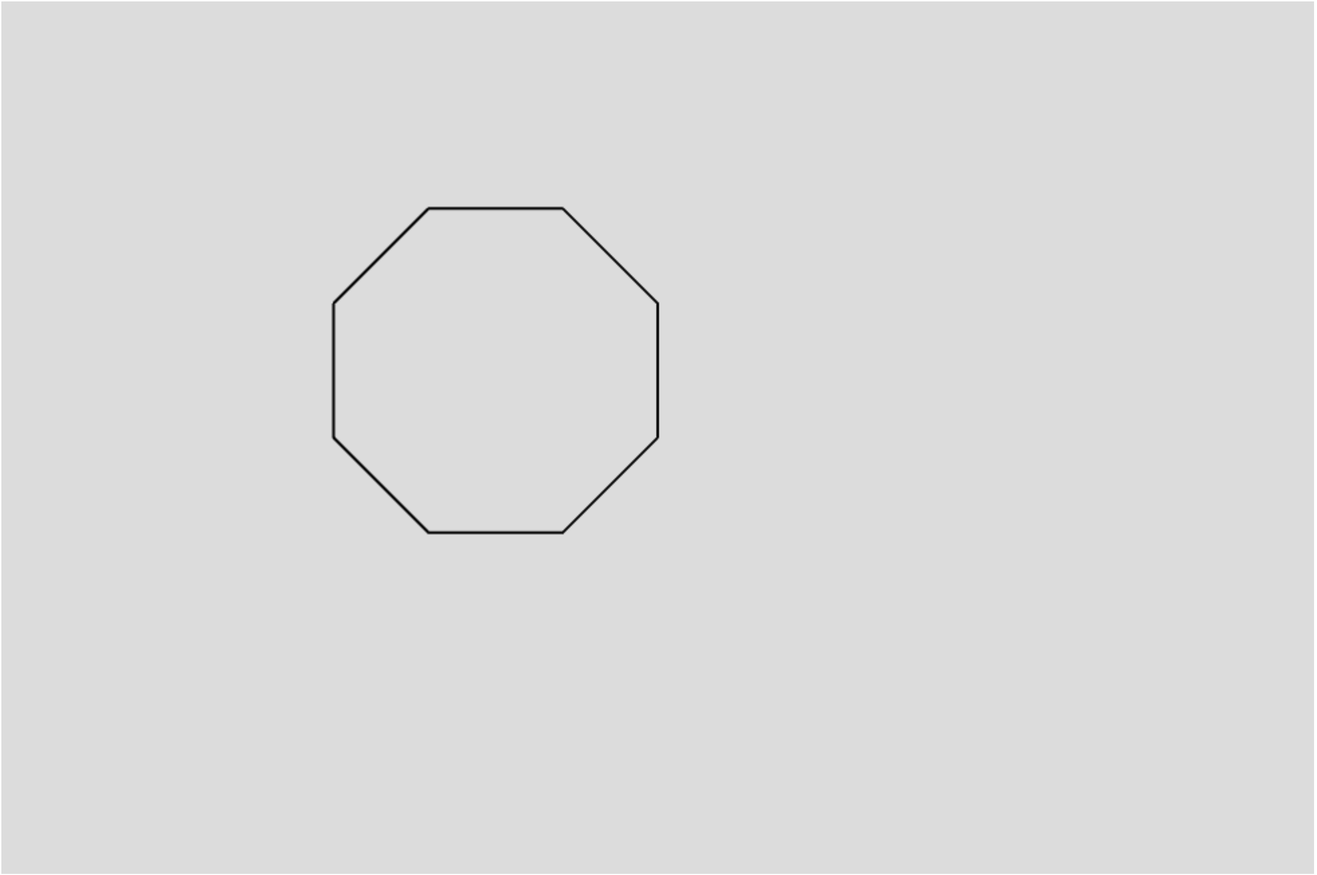

Live at Manchester College Of Commerce 2nd May 1969
Live at Birmingham Mother's 27th April 1969
PINK
FLOYD
PINK FLOYD
PINK
FLOYD


gigi
PINK
FLOYD

Some Actual Recursive Code

```
function weirdShape(depth, dist, ang) {
  if (depth > numSides) {
    return;
  }
  rotate(ang);
  line(0, 0, 0, dist);
  translate(0, dist);
  weirdShape(depth + 1, dist, ang);
}
```

```
function weirdShape(depth, dist, ang) {
  if (depth > numSides) {
    return;
  }
  rotate(ang);
  line(0, 0, 0, dist);
  translate(0, dist);
  weirdShape(depth + 1, dist, ang);
}
```



```
function weirdShape(depth, dist, ang) {
  if (depth > numSides) {
    return;
  }
  rotate(ang);
  line(0, 0, 0, dist);
  translate(0, dist);
  weirdShape(depth + 1, dist, ang);
}
```

Takes arguments as normal

```
function weirdShape(depth, dist, ang) {
  if (depth > numSides) {
    return;
  }
  rotate(ang);
  line(0, 0, 0, dist);
  translate(0, dist);
  weirdShape(depth + 1, dist, ang);
}
```

Returns when a stopping condition has been reached.

If you don’t do this, it will recurse forever. This is never what you want.

```
function weirdShape(depth, dist, ang) {
  if (depth > numSides) {
    return;
  }
  rotate(ang);
  line(0, 0, 0, dist);
  translate(0, dist);
  weirdShape(depth + 1, dist, ang);
}
```

If you didn’t return early, then do some work. Here we rotate a bit, draw a line, and then translate to the end of that line.

```
function weirdShape(depth, dist, ang) {
  if (depth > numSides) {
    return;
  }
  rotate(ang);
  line(0, 0, 0, dist);
  translate(0, dist);
  weirdShape(depth + 1, dist, ang);
}
```

Last, issue the recursive call.

Notice that we increase the depth by one, which is necessary for the recursion to eventually end.

```
function weirdShape(depth, dist, ang) {
  if (depth > numSides) {
    return;
  }
  rotate(ang);
  line(0, 0, 0, dist);
  translate(0, dist);
  weirdShape(depth + 1, dist, ang);
}
```