

Computational Thinking

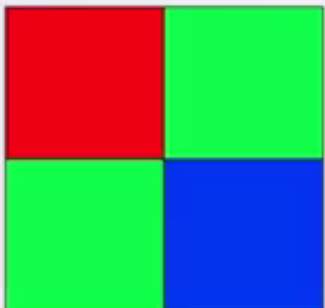
- ▶ Introduction2 min
- ▶ Everything Is a Number6 min
- ▶ How Is That a Number?2 min
- ▶ Developing an Algorithm6 min
- ▶ A Seven Step Approach to Solving Programming Problems7 min
- ★ Practice Quiz: Solving Programming Problems4 questions

Programming Fundamentals with JavaScript

Implementing the Green Screen Algorithm

Review

Write Down What We Just Did: Step-By-Step



fglImage

1 I started with the foreground image I wanted (fglImage)

I started with the foreground image I wanted, I'll call this fglImage and

Developing an Algorithm

Have a question? Discuss this lecture in the week forums.



Interactive Transcript

Search Transcript

English

0:03

The Green Screen problem also called a chromo key problem because it is keyed to the colors chromo or hue, like removing the red from eyes in a picture with a flash. Is common in videography as it allows actors to be recorded in a studio, and placed in front of a background or made from an image or video taken in another setting. This algorithm is what let Dru and I talk in outer space with dinosaurs. Now, we're going to walk through solving this problem as an example of how to approach programming problems in general.

0:38

The first thing we need to do before we solve a program problem is to figure out exactly how to solve that problem ourselves. You cannot write a program, explain to the computer what it needs to do until you completely understand yourself how to do the task in a precise step by step fashion. In fact, this is often the hardest part of programming.

1:01

Now, trying to do this particular example of the Green Screen proplem will be too difficult because these images have 2 million pixels each approximately. Trying to operate on them by hand would take an impossibly long time.

1:14

Instead, it is a good idea to work on a smaller instance of the problem by hand to gain a deep understanding of how to solve the problem. In this case, we are going to look at a 2 pixel by 2 pixel image. We will start by picking a 2 by 2 image to be the foreground. The image that will go on top, and we'll pick a 2 by 2 image to be the background. Also, we will need an image called the final result or output. This should also be 2 by 2. Now that the examples are chosen, let's go through and figure out what color to make each pixel of the output image.

1:51

Great! Now, we have solved an instance of a problem by hand. The next step is to write down exactly what we did in a step by step fashion. I started with the foreground image I wanted. I'll call this fglImage and with the background image I'll call it bglImage. I then made a blank image of the same size, I'll call it output. I looked at the first pixel in fglImage and it was red. So I set the output's corresponding pixel to be red as well. I looked at the second pixel in fglImage, it was green, so I looked at the same position in bglImage and set output's corresponding pixel to bglImage's pixel. I looked at the third pixel in fgimage, it was green. So, I looked at the corresponding pixel in bgimage. It's blue so I set the output to be that same color. I then looked at the fourth pixel in fgimage and it's blue, so I set the corresponding pixel to br blue in the output.

2:55

Great! Now, we have a step by step set of instructions to describe exactly how we solve the problem for this particular pair of images. But to write a program, we want to be able to solve this problem for any image of any size.

3:11

Now, if you look at these steps closely, you'll see that there is a lot of similarity. We're doing almost but not quite the same thing for each pixel in the image. When the fglImage's pixel is green, we use the the bglImage's pixel. And when the fglImage's pixel is not green, we use the fglImage's pixel directly. Going back to our step by step instructions, we will rewrite each step to be a little bit more general, taking into account this conditional behavior we just observed. You can do this for each pixel's step. And now each step is more general and more similar. They would work for any 2 by 2 images, but only for 2 by 2 images.

3:53

Here we've improved the step-by-step instructions to express the repetition over each pixel.

3:58

Now, the steps are general enough to work on any sized image. In fact, what we have done, is devise an algorithm, that we're going to implement. An algorithm is a clear step by step set of an instructions to solve any instance of the problem you want to solve. You can express an algorithm in English as we have done here or in code as we'll need to do to let the computer run it.

4:22

Everyone makes mistakes. When devising an algorithm there are many different ways to make mistakes. For example, you may not have seen the pattern correctly, or you may not generalize each step correctly. To try to protect against these types of mistakes, let's try our algorithm out on a different example to see if it works as expected. If it does, we'll be more confident that we did it right. If not, we've caught our mistakes early before we went to write code. Of course, once we write code, we will want to test them more thoroughly. But we'll talk about that later.

4:54

As you walk through your algorithm, you'll want to keep track of what step you're on. For us, we're going to draw a green arrow to show where we are in the algorithm. You then want to go through each step exactly as it is written. Here, I've picked the 3 by 1 image to be the foreground. And this 3 by 1 image to be the background. And the 3 by 1 image to be the output. Here, I will keep track of which pixel is the currentPixel where the blue arrow in the drawing above. This first pixel is green. So the algorithm says to look at the same pixel in the background image which is yellow.

5:28

And make the same pixel and the output image also yellow. Those are all the steps for that pixel, so now the algorithm says to go to the next one and repeat the process.

5:39

We will continue to follow these steps as they are written exactly as they are written until we finish all of them.

5:49

At this point, we want to see if the image came out as expected.

5:57

In this case, the output was right since our algorithm appears to work, it's time to write the code

Downloads

Lecture Video mp4

Subtitles (English) WebVTT

Transcript (English) txt

Lecture Slides pdf

Would you like to help us translate the transcript and subtitles into additional languages?