

# PSA5: Chromakey

**Read all instructions in this document before starting any coding!**

**START EARLY, START OFTEN!**

**Due: Sunday, November 12th, 11:59pm**

**Overview:** In this assignment you will be editing a picture of yourself and changing the background and your shirt color using the methods you have learned for editing pictures.

As described in class, **you may complete at most 4 of the quarter's programming assignment using pair programming.** If you have decided to work in pairs, please review the [guidelines for pair programming](#). In addition, note the following details about working with a partner:

- For this assignment, if you decided to work in pairs, find a partner before you start on the assignment. Or else, you have violated the policy on pair programming.
- You will submit only ONE version between the two of you though your group can submit as many times as you want to.

## Helpful Information:

[Online Communication: Using Piazza, Opening Regrade Requests](#)

[Getting Help from Tutor, TA, and Professor's Hours](#)

[Lab and Office Hours](#) (Always refer to this calendar before posting.)

[Academic Integrity: What You Can and Can't Do in CSE 8A](#)

[Remote Lab Access and File Transfer Guide](#) or [Yingjun's SSH and VNC post](#)

[Picture Documentation](#) and [Pixel Documentation](#)

## Table of Contents:

[Getting Started](#)

[Problem \[Total: 20 points\]](#)

[Part A: Chromakey Background \[7 points\]](#)

[Part B: Chromakey Shirt \[7 points\]](#)

[Chromakey.java file \[2 points\]](#)

[Program Description \[2 points\]](#)

[Commenting and Style \[2 points\]](#)

[Star point \[Optional\]](#)

[How to Turn in your Homework](#)

# Getting Started

## Instructions for working on a B230 lab machine:

- 1) Open a terminal
- 2) Copy the starter code from the public folder.

**Paste** the following lines of code one after the other in the terminal i.e. paste the first line, press enter and then the second line and so on:

```
cd ~/
mkdir psa5
cd psa5
cp ../../public/psa5/* .
ls
```

(Do you recall what each of the above commands mean. If you are not able to, review [PSA0](#), where each of the commands is explained.)

- 3) Verify that Picture.java, Chromakey.java is shown as the result of the last command.  
Following these procedures, your file will be named correctly and be located in the correct place. For every assignment you need to follow correct naming assignments in the correct locations to get credit for your work.

- 4) Proceed to start your programming assignment.

## Instructions for working on your Local Machine (Your Laptop/Computer):

- 1) Create a psa5 folder on your machine.
- 2) Download zip and unzip it inside the psa5 folder: [LINK HERE](#)

## Problem: Chromakey

In this project, you are using the colors of pixels in an image to allow you to substitute pixels from other images, in a process called Chroma Key. To do this project, you will need **3 source images (saved into your PSA5 folder)** and use them to create a new image:

1. A picture of yourself or yourself and your partner together standing/sitting/laying/jumping/etc in front of a green screen, located in B230. Person(s) appearing in this picture should be wearing a solid color shirt (almost any color but NOT GREEN, same color makes things easier, but it's not necessary). This is your first *source* image.
  - a. Name this pic0.jpg (referred to as *myselfSourceImage*),
2. You will then find/steal/make two additional *source* pictures.
  - a. One will be used as the background (to replace the green screen of your first image. Name this pic1.jpg (referred to as *myBackgroundSourceImage*)
  - b. The last source image can be anything that you like. The pattern on this image will replace the pixels on the solid color shirt that you are wearing in your first image. Name this pic2.jpg (referred to as *myShirtSourceImage*)
- You probably want to make sure that your images are not *too* large or your programs will take forever to run. You can use an image editor to scale them down after you take them, but note that this is the **only** processing of the images you're allowed to do outside your Java program. More about the sizes of your images in the "[Hints and Restrictions](#)" section.

To accomplish this task, you will **write at least the following two methods in Picture.java:**  
**chromakeyBackgroundChange** and **chromakeyShirtChange**

## Part A: Chromakey Background (7 points)

Now that you read about the problem that is being solved in this PSA. Take some time to think about the interactions of replacing the green sheet color with a background of your choice by answering the following question in the header of Chromakey.java:

1. What would happen if the tolerance for distinguishing the background and you is set too high? Too low?

**Method:** `Picture chromakeyBackgroundChange(Picture background, Color replaceColor, double tolerance)`

- This method must take in 3 parameters:
  - background: the Picture you will be using to obtain new background pixel values.
  - replaceColor: the Color on your source Picture to be replaced by the background Picture.
  - tolerance: determines whether the Color in the source Picture is close enough to replaceColor
- Since you are using a green screen, you should indicate that Color should be a color close to the green canvas. However, this should work for **any background canvas color!**
  - Use `colorDistance()` to see how far away a color is from another one. See the [documentation](#) of Pixel for more details.
  - We may test your method using different background colors, and see if it replaces those correctly!
- The method will copy the calling object and replace all of the Pixels in the copy (within a set tolerance) with Pixels from the background Picture.
- The **tolerance** amount is up to you to set in your main method. However, you should tune it so that it ONLY replaces the green screen Pixels! (May have to run it a few times to find a good value)
- **The method returns the modified copy!**
- See the [example for part A](#) for better understanding of the workflow for the program.

To see your chroma key background program in action. Test it out in in the main method of **Chromakey.java** by doing the following:

- You must use your own picture of you or your partner using the sheets in the lab as the original calling object.
- Open the picture using a relative filepath. Example:
  - `String fileName = "pic0.jpg";`
  - `Picture picture1 = new Picture(fileName);`
- In your methods, you need to **create a copy of the calling object and modify the copy, not the calling object itself.**
- Picture object of the original image (myselfSourceImage) displayed with `explore()`.

- A method call to chromakeyBackgroundChange to change the background):
  - This should replace most of the green pixels in the background of the image
  - The resulting Picture should be saved into a Picture Object, and displayed

See [ChromaKey.java portion](#) of the writeup for more details.

## Part B: Chromakey Shirt (7 points)

Before starting, think about how you can make sure that only the shirt's color is being changed. In the header of Chromakey.java answer the following question:

2. Why should the program make comparisons to the original picture with the solid green background when replacing the shirt's color with the shirt's new background instead of the resulting picture from part A (with the green background replaced)?

**Method:** `Picture chromakeyShirtChange(Picture shirt, Picture original, Color oldShirtColor, int startX, int startY, int width, int height, double tolerance)`

- The calling object of this method will be the Picture object that has it's background changed by the chromakeyBackgroundChange method.
- This method must take in the parameters shown!
  - Shirt: the source picture *myShirtSourceImage*, or pic2.jpg, used to obtain the new shirt Pixel values.
  - Original: the original picture (*myselfSourceImage*), assist the method with determining which Pixels of the shirt color are part of the shirt, rather than the changed background, since the background may now have many Pixels similar to the color of your shirt, which should not be changed.
  - oldShirtColor: the color of the old shirt that we wanted to change (from *myShirtSourceImage*)
  - startX, startY: top left of where the shirt starts
  - width, height: the width and height of the region with the shirt. (the size of the box this creates should only be slightly bigger than the size of the shirt)
  - tolerance: determines whether the Color in the calling object Picture is close enough to oldShirtColor
- The method will COPY the calling object (picture with changed background) and perform the shirt change on that copy. **Then this new image will be RETURNED by the method.**
- Example:



The method will COPY the calling object (picture with changed background) and perform the shirt change on that copy. Then this new image will be RETURNED by the method.

- See the [example for part B](#) for better understanding of the workflow for the program.

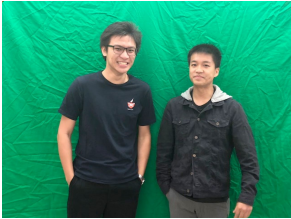


To see your chroma key shirt program in action. Test it out by adding to the main method of **ChromaKey.java** by doing the following:

- Use the original picture and the resulting picture with the changed background from part A.
- Make a method call to chromaKeyShirtChange to change the T-shirt color (part B):
  - Make sure to define the parameters (you can use explore to figure out the optimal points and set the values to constants)
  - The resulting picture should be saved into a new Picture Object, and displayed (this is your final Picture)
  - **Make sure to save each part as it's own image object! (since the methods return a modified copy, it is a new object, and can be stored into a new reference variable)**



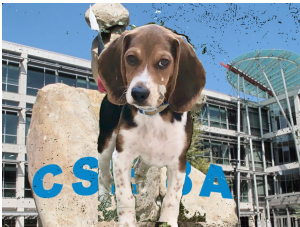
See [ChromaKey.java portion](#) of the writeup for more details.

# Examples

## Example 1: background change (part A)

 A	 B	 C
myselfSourceImage	myBackgroundSourceImage	myChangedBackgroundImage

## Example 2: background change (part A)

 A	 B	 C
myselfSourceImage	myBackgroundSourceImage	myChangedBackgroundImage

The following example refers to Example 1:

Suppose that we have two Picture object references: `myselfSourceImage`, and `myBackgroundSourceImage` that have been defined previously in the main method. These pictures are shown in the figure above (picture A and B). We also have a Color variable `bgColor` that stores a green color (like the background color in `myselfSourceImage`).




Now imagine we call

```
Picture myBackgroundSourceImage =
myselfSourceImage.chromaKeyBackgroundChange(myChangeBackgroundImage, bgColor,
tolerance);
```

This method call will return a NEW Picture, which will get the reference `myChangedBackgroundImage`. The Picture will look like the one shown in the figure above (picture C). Notice that the other two images will NOT be modified.



## Example 2: T-shirt change (after changing the overall background color first, part B)

 <p>C</p>	 <p>D</p>	 <p>E</p>
myChangedBackgroundImage	myShirtSourceImage	myFinalImage

After completing the first example, we have a `Picture` object myChangedBackgroundImage (reference to picture C). The next step is to change the shirt Pixels to the pattern shown in Picture myShirtSourceImage (reference to D). You will use the method `chromakeyShirtChange` on the reference to C (`myChangedBackgroundImage`) with the line:

```
Picture myFinalImage=
myChangedBackgroundImage.chromakeyShirtChange(myShirtSourceImage,
myselfSourceImage, shirtColor, startX, startY, width, height, tolerance)
```

Where `shirtColor` is close to dark blue in this case, and `startX`, `startY`, `width`, `height`, and `tolerance` are what you will have defined.

This method will return a NEW picture, which will be referenced by `myFinalImage` (a reference to picture E). Notice that it replaces the dark blue shirt with the new background (the Star Wars characters in this case). It does NOT change the other images (calling object or parameter objects). Similarly, these steps were repeated again to replace the second person's jacket with the new background.

**Why does this method take the original `myselfSourceImage` (a reference to picture A)?** The reason is that your image with the new background (picture C) might have pixels that are the same color as the shirt, even within the red-box region of the T-shirt specified by `startX`, `startY`, `width` and `height`. So you should use the *original* image to determine which pixels to change, but change those pixels in the copy of the calling object.

## Hints & restrictions:

- Note: the two abovementioned examples intend to offer you some hints on how to test your methods, and only give some barebone implementations. Please make sure you write your testing codes thoroughly.
- If size of the picture that you took is very large, you can resize it using any photo editing software like Microsoft Paint to an appropriate size. It is recommended that you resize your background image to

be the same size as the original image with the green background. As for the T-shirt image, its size is recommended to be the same as the T-shirt region in the original image.

- Other than the sizes of the images, you are **not allowed** to digitally modify the color of the picture you have taken, using Photoshop or any other editing software. If you feel the need, you can use your own different colored sheet or have a contrasting color of the T-Shirt. The color of the pictures should only be modified by your program.
- You can define additional methods to "improve" the lighting or color contrast of your images (similar filters to what you've done for previous PSAs).

## Chromakey.java (2 points)

- Open Chromakey.java. You will write an application to call your methods to that implement chromakey.
- **REQUIREMENT:** You must use the picture that you and your partner took using the sheets in the lab as the original calling object and in your methods, you need to **create a copy of the calling object and modify the copy, not the calling object itself.**
- **IMPORTANT:** Your code should work for different image paths. To ensure this, do not use the file path or MediaPath functions in your method. **Copy the images you want to use for your final submission into the same folder as your source code. Then specify the image file path by mentioning file name. For example, if you want to use pic0.jpg, and then use**

```
String fileName = "pic0.jpg";
Picture picture1 = new Picture(fileName);
```

### What we want to see in the main method:

- Picture object of the original image (myselfSourceImage) displayed with explore().
- A method call to chromakeyBackgroundChange to change the background (part A):
  - This should replace most of the green pixels in the background of the image
  - The resulting Picture should be saved into a Picture Object, and displayed
- A method call to chromakeyShirtChange to change the T-shirt color (part B):
  - Make sure to define the parameters (you can use explore to figure out the optimal points and set the values to constants)
  - The resulting picture should be saved into a new Picture Object, and displayed (this is your final Picture)
  - You can post your resulting image on Piazza to show off!
  - **Make sure to save each part as it's own image object! (since the methods return a modified copy, it is a new object, and can be stored into a new reference variable)**

## Program Description (2 points)

Describe what your program does as if it was intended for a 5 year old or your grandmother. Do not assume your reader is a computer science major. The programs you should comment on in this segment include PSA5 and Picture.java. **Write this as comments at top of Picture.java file.**

## Comments and Style (2 points)

**Remember include file header comments with your and your partners Name, login id and date in all your files.** Comments and Style are worth **2 points**.

We will be looking for the following additional comment/style points in your code:

- Meaningful variable names (i.e. don't name a pixel "n", "i", or "bob" or "hello". Name them something pertinent to their role, such as "sourcePixel" or "targetPixel")
- Proper indentation of code. Every subsequent coding/loop block must be further indented properly. Every single code example in your textbook follows this convention, so see your book for examples
  - In Dr Java, you can highlight everything and press tab, it will automatically indent everything for you. Remember for future projects (cse8b, etc.), you won't have Dr.Java as your editor, and will have to do this indenting manually!
- Method header comments at beginning of methods indicating what they do at a high level. Example: "`///method flips the image upside down`", or "`///method mirrors the image along the center vertical line`". You should also include a description of the parameters and return values in the header comments. (google javadoc styling if you're not sure)
- **Each code line should be no longer than ~80 characters.**
- **Do not copy-paste formatted text into your code! Apostrophes and certain characters being inside of your code will break the code! We will take points off if your code does not compile because of this!**

## Star point (Optional)

For the star point this week, you can do either one of the following:

1. The background image for the chromakeyBackgroundChange method may not be of the same size as your original picture with the green background. Resize the background picture, without introducing compressing or stretching effects, to fit the background of your original image by java only, **i.e without using any photo editing software** like Microsoft Paint or any other picture editing software.
  2. Similarly resize the image for T-shirt to fit your shirt size by java, **without resizing the picture using any photo editing software** like Microsoft Paint or any other picture editing software.
- NOTE: CROPPING DOES NOT COUNT!!!

### Other Notes

- After you dynamically resize the background images for either the entire background or the T-shirt background, you need to call the appropriate methods you wrote for this PSA (either chromakeyBackgroundChange or chromakeyShirtChange) to evaluate how well your resizing has worked. Your resizing method should consider both shrinking and enlarging.
- The resized image shouldn't have any compressed or stretched artifacts. One potential algorithm you can try is illustrated in this youtube video: <https://www.youtube.com/watch?v=6NclJXTlucg>.
- For this star point, you should create a new method in Picture.java that you can call in a new file called **StarPointPSA5.java**. Please name your new star point method **resize** and add a comment in both Picture.java and StarPointPSA5.java headers. Your resize method need to take in the desired size (i.e. width and length) as its parameters. You can include other parameters for your resize method as you deem necessary. **Don't forget your header comment in that file which includes all the usual information.**
- You should turn in your method resize in Picture.java, the StarPointPSA5.java, the original image (StarPointOriginal.jpg), and the resized image (StarPointResized.jpg).

## How to Turn in Your Homework

1. Place your psa5 folder in the home directory of your student account (cs8afx).
2. Ensure that the psa5 folder contains at least all the following files: **Picture.java**, **Chromakey.java**, **pic0.jpg**, **pic1.jpg**, **pic2.jpg**
3. Run the command: `cse8aturnin psa5`
4. Follow the prompts.
5. Verify that your homework was turned in with the command: `cse8averify psa5`

Refer back to [psa0](#) turnin instructions for additional details.