CSI201Proj4 READ assg. TODAY!

3 Inputs

1 Output
CSI201
Proj4 READ assg. TODAY!
Chromakeyed image substitution
Project support lecture(s)
Lecture 11

Midterm: Snowed! Rescheduled from next week to Mar 27 (after Spring break and Proj4 are finished)
Proj4 Goal 1
Locate and get Pixels in a Picture by x y coordinate location and change their colors

Lab5: One Pixel at a time.
Lab6: One loop containing method does many Pixels
Goal 2
2 nested loops to process every Pixel in one Picture
Goal 3
Use an if conditional to change a Pixel's color *conditionally*
Lab6 followup
Goal 4
Copying part of one Picture into another Picture.
Goal 1
Locate and get Pixels in a Picture by x y coordinate location and change their colors

Lab5: One Pixel at a time.
Lab6: One loop containing method does many Pixels
public class EditablePicture extends Picture {

    public EditablePicture(String filenameParam )
    {
        super(filenameParam); }

    public void changeXYLocationsColor( int xParam, int yParam,
                                          java.awt.Color cParam )
    {
        //Purpose: Set up a Pixel reference
        //variable with a declaration.
        Pixel pixRef;
        //Purpose: Call the getPixel method to get a
        //reference to the Pixel to change,
        //and assign or copy that reference into a variable.
        pixRef = this.getPixel(xParam, yParam);
        //Purpose: Set the Color of the Pixel to change to the
        //value of the Color parameter.
        pixRef.setColor( cParam );
        //Purpose: Return control to the spot
        //where the method call was.
        return;
    }
}
public class EditablePicture extends Picture
{
    public void changeHLinesColor(int xParam, int yParam,
        int lenParam,
        java.awt.Color cParam )
    {

        WHOA DONT KNOW WHT TO DO..Make a PLAN!
    }
}

public class EditablePicture extends Picture
{
    public void changeHLinesColor(int xParam, int yParam,
            int lenParam,
            java.awt.Color cParam )
    {
        //Set up loop control variable whose value will be
        //how many Pixels we colored along the HLine.
        int nDone;
        //We didn't do any yet.
        nDone = 0;
        //Keep coloring as long as there are more to color.
        while( nDone < lenParam )
        {
            //Variable to refer Pixels..set it up
            Pixel pRef;
            //Get the Pixel to recolor
            pRef = this.getPixel(xParam+nDone, yParam);
            //Actually recolor it.
            pRef.setColor(cParam);
            //Count one more pixel recolored.
            nDone = nDone + 1;
        }
        //We are really done.
        return ;
    }
}
Lecture 12 starts here
Goal 2
2 nested loops to process every Pixel in one Picture
public class WhiteoutApp {
    public static void main(String[] a) {
        FileChooser.pickMediaPath();
        EditablePicture pRef =
            new EditablePicture(
                FileChooser.pickAFile()
            );

        pRef.explore();
        pRef.makeAllWhite();
        pRef.explore();
    }
}
public class EditablePicture
    extends Picture
{
    public EditablePicture
        (String filename)
    {
        super(filename);
    }

    public void makeAllWhite()
    {
    }
}

public class WhiteoutApp
{
    public static void main(String[] a)
    {
        FileChooser.pickMediaPath();
        EditablePicture pRef =
            new EditablePicture(
                FileChooser.pickAFile()
            );
        pRef.explore();
        pRef.makeAllWhite();
        pRef.explore();
    }
}
public void makeAllWhite()
{
    int x;    //set up a variable to track how many columns we finished whitening.
    x = 0;    //No columns were whitened when we begin.
    while( x < this.getWidth() )   //loop until we whitened all the columns
    {
        //whiten a whole column.
        int y;  //set up a var. to track how many pixels in column x we finished whitening
        y = 0; //None were whitened when we begin a column.
        while( y < this.getHeight() ) //loop until we whiten all the pixels in one column
        {
            Pixel pRef;   //Set up a var. to refer to the Pixel we will whiten.
            pRef = this.getPixel(x,y);   //Get the Pixel to whiten.
            y = y + 1;    //Count one more whitened Pixel in the column.
        }
        x = x + 1;   //Count one more finished column.
    }
    return ;    //Continue computing according to the caller's code.
}
public class EditablePicture extends Picture {

    public EditablePicture(String fnameParam) {
        super(fnameParam);
    }

    //In Java non-default constructor methods are not inherited.

    public void makeAllWhite() {
        int x;    //set up a variable to track how many columns we finished whitening.
        x = 0;   //No columns were whitened when we begin.
        while( x < this.getWidth() )   //loop until we whitened all the columns
            {   //whiten a whole column.
                int y;  //set up a var. to track how many pixels in column x we finished whitening
                y = 0; //None were whitened when we begin a column.
                while( y < this.getHeight() ) //loop until we whiten all the pixels in one column
                    {   //whiten a whole column.
                        Pixel pRef;   //Set up a var. to refer to the Pixel we will whiten.
                        pRef = this.getPixel(x,y);   //Get the Pixel to whiten.
                        y = y + 1;    //Count one more whitened Pixel in the column.
                        x = x + 1;   //Count one more finished column.
                    }   //whiten a whole column.
        return ;    //Continue computing according to the caller's code.
    }

}

public class WhiteoutApp {

    public static void main(String[] a) {
        FileChooser.pickMediaPath();
        EditablePicture pRef =
            new EditablePicture(
                FileChooser.pickAFile()
            );
        pRef.explore();
        pRef.makeAllWhite();
        pRef.explore();
    }
}
public void makeAllWhite()
{
    int x;  //set up a variable to track how many columns we finished whitening.
    x = 0;  //No columns were whitened when we begin.
    while( x < this.getWidth() ) //loop until we whitened all the columns
    {
        //whiten a whole column.
        int y;  //set up a var. to track how many pixels in column x we finished whitening
        y = 0;  //None were whitened when we begin a column.
        while( y < this.getHeight() ) //loop until we whiten all the pixels in one column
        {
            Pixel pRef;  //Set up a var. to refer to the Pixel we will whiten.
            pRef = this.getPixel(x,y);  //Get the Pixel to whiten.
            y = y + 1;  //Count one more whitened Pixel in the column.
        }
        x = x + 1;  //Count one more finished column.
    }
    return ;  //Continue computing according to the caller's code.
}

What is this? The makeAllWhite method was called ON an EditablePicture, a picture of a person with red and green chromakeyed regions. this refers to that picture, so the computer can get its height and get its Pixels (one at a time.)
public void makeAllWhite()
{
    int x; //set up a variable to track how many columns we finished whitening.
    x = 0; //No columns were whitened when we begin.
    while( x < this.getWidth() ) //loop until we whitened all the columns
    {
        //whiten a whole column.
        int y; //set up a var. to track how many pixels in column x we finished whitening
        y = 0; //None were whitened when we begin a column.
        while( y < this.getHeight() ) //loop until we whiten all the pixels in one column
        {
            Pixel pRef; //Set up a var. to refer to the Pixel we will whiten.
            pRef = this.getPixel(x,y); //Get the Pixel to whiten.
            y = y + 1; //Count one more whitened Pixel in the column.
        }
        x = x + 1; //Count one more finished column.
    }
    return; //Continue computing according to the caller's code.
}
public void makeAllWhite()
{
    int x;    //set up a variable to track how many columns we finished whitening.
    x = 0;   //No columns were whitened when we begin.
    while( x < this.getWidth() )   //loop until we whitened all the columns
    {
        //whiten a whole column.
        int y;  //set up a var. to track how many pixels in column x were whitened
        y = 0; //None were whitened when we begin a column.
        while( y < this.getHeight() ) //loop until we whiten all the pixels in one column
        {
            Pixel pRef;   //Set up a var. to refer to the Pixel we will whiten.
            pRef = this.getPixel( x, y );   //Get the Pixel to whiten.
            y = y + 1;    //Count one more whitened Pixel in the column.
        }
        x = x + 1;   //Count one more finished column.
    }
    return ;    //Continue computing according to the caller's code.
}
4 applications of variables

• A parameter variable stores, holds, remembers a parameter value. (eg. `sizeParamVar`)

• A variable stores, holds, remembers some data so the method can use that data later. (eg. `xOrig, yOrig`)

• A variable stores, holds, remembers a reference to an object (where an object is inside the computer) so methods can be called ON that PARTICULAR OBJECT (eg. `tref` and `pixRef`)

• A loop control variable controls when a loop should stop and might repeatedly provide a useful numbers, like `count` in `this.getPixel(xParam+count, yParam);` (where `this` refers to the G&E Picture you are editing.)
In lecture exercise to practice using three PAPER variables (3 “tickets”) in a hand computation (done by a woman or a man). A VARIABLE is a piece of MEMORY.

Application 5: Variables to remember numbers that are changed during a computation so the changed values can be used accurately during the same calculation.
Your RED ticket. Use this ticket: To write a number on, to read a number to tell if it is 0, or to compare it to the number on another ticket, or to subtract it with another number from a ticket, and also, ERASE and replace with another number.

10 - 10 = 0

Write your name here: 

Job title: Computer. TA name 

Mission: Execute, run, perform, do, demonstrate Euclid's algorithm on parameter numbers 126, 102, 165

Your program, recipe, specific instructions, orders, directions, what to do EXACTLY:

1. Write parameter number 126 on your RED ticket.
2. Write parameter number 102 on your GREEN ticket.
3. Write parameter number 165 on your BLUE ticket.
4. Look over the numbers and pick any two that are NOT BOTH 0.
5. If step (4) failed because two tickets have 0's on them, then STOP, and shout out the answer by reading the number on the remaining ticket. Also write it here
6. See which of the two non-zero numbers you picked is bigger than the other. If they are the same, just pick one of the two tickets with the common number.
7. Put your thumb on the ticket whose number you picked.
8. Subtract the OTHER picked number from the number your thumb is on. (The difference should be 0 if they are the same, and positive if they are different.)
9. Erase (or cross out) the number under your thumb and overwrite it with the difference, the number you got from the subtraction you did in step 8.
10. Now, go back to step (4). You will repeat steps (4) – (9) over and over until you get two zeros.
Your RED ticket. Use this ticket:
To write a number on, to read a number to tell if it is 0, or to compare it to the number on another ticket, or to subtract it with another number from a ticket, and also, ERASE and replace with another number.

ACTUALLY DO AND SHOW CROSSING OUT or ERASING and then REWRITING values that you do on the VARIABLES.

Your GREEN ticket. Use this ticket:
To write a number on, to read a number to tell if it is 0, or to compare it to the number on another ticket, or to subtract it with another number from a ticket, and also, ERASE and replace with another number.

Your BLUE ticket. Use this ticket:
To write a number on, to read a number to tell if it is 0, or to compare it to the number on another ticket, or to subtract it with another number from a ticket, and also, ERASE and replace with another number.

Write your name here: [Name]
Job title: Computer. TA name: [Name]

Mission: Execute, run, perform, do, demonstrate Euclid’s algorithm on parameter numbers 126, 102, 165

Your program, recipe, specific instructions, orders, directions, what to do EXACTLY:

(1) Write parameter number 126 on your RED ticket.
(2) Write parameter number 102 on your GREEN ticket.
(3) Write parameter number 165 on your BLUE ticket.

Comment: The rest of the instructions are roughly "subtract like crazy" until you get two zeros.

(9) Erase (or cross out) the number under your thumb and overwrite it with the difference, the number you got from the subtraction you did in step 8.

(10) Now, go back to step (4). You will repeat steps (4) – (9) over and over until you get two zeros.
Back to an earlier example

Programming an ArtisticTurtle to draw a dome
YOU must ..... (4)

In

```java
int numberOfMoreTimes;
numberOfMoreTimes = 4;
while ( numberOfMoreTimes > 0 )
{
    this.forward(sizeParam);
    this.turn(72/2);
    numberOfMoreTimes = numberOfMoreTimes - 1;
    Subtract 1 from the memorized count.
}
```

MEMORIZE (4) How to pre-program so WHEN THE LOOP RUNS, the computer will subtract 1 from the memorized count after each repetition.
int numberOfMoreTimes;

Can do: Set up a variable.
numberOfMoreTimes = 4;

Copy 4 into that variable.

while ( numberOfMoreTimes > 0 )

Tell whether the count of more times is zero or not.
{
    this.forward( sizeParam );
    this.turn( 75/2 );
    numberOfMoreTimes = numberOfMoreTimes - 1;

    Subtract 1 from the memorized count.
}

Computer: Duh, sorry. I'm TOO DUMB
to know how to repeat 4 times. But, I can do what
you preprogrammed above with a variable.
4 step checklist for coding a loop

1. DECLARE (which means set up) the loop control variable, giving it a name.
2. ASSIGN with = (which means COPY) the initial value that the loop variable should have.
3. Code a while statement's HEAD and BODY so you write as the HEAD \texttt{while( TEST )} a \texttt{TEST} for the computer to do each time, on the loop variable.
4. Code inside the BODY both
   
   \begin{enumerate}
   
   \item[(4a)] what to repeat e.g. \texttt{forward(); turn();}
   \item[(4b)] to CHANGE the loop variable's value e.g. \texttt{numberOfMoreTimes=numberOfMoreTimes-1;}
   
   \end{enumerate}
iClicker What if you omit step 4B?

```java
int numberOfMoreTimes;
numberOfMoreTimes = 4;
while ( (numberOfMoreTimes > 0 ) ) {
    this.forward(sizeParam);
    this.turn(72/2);
    numberOfMoreTimes = numberOfMoreTimes - 1;
    Subtract 1 from the memorized count.
}
```

A) It still does 4 lines, turns and stops.
B) It crashes.
C) It keeps doing lines and turns until you click DrJava's RESET button.
D) It does nothing.  E) Nothing, and gets stuck.
for Prof. Hurd..
How many college semesters did you complete?

A) Less than 1
B) 1
C) 2
D) 3
E) More than 3
public void makeAllWhite()
{
    int x;    //set up a variable to track how many columns we finished whitening.
    x = 0;   //No columns were whitened when we begin.
    while( x < this.getWidth() )   //loop until we whitened all the columns
    {
        //whiten a whole column.
        int y;  //set up a var. to track how many pixels in column x we finished whitening
        y = 0; //None were whitened when we begin a column.
        while( y < this.getHeight() ) //loop until we whiten all the pixels in one column
        {
            Pixel pRef;   //Set up a var. to refer to the Pixel we will whiten.
            pRef = this.getPixel(x,y);   //Get the Pixel to whiten.
            y = y + 1;    //Count one more whitened Pixel in the column.
        }
        x = x + 1;   //Count one more finished column.
    }
    return ;    //Continue computing according to the caller's code.
}

Study it without the clutter.
Study HOW the computer is commanded to
whiten a whole column—It runs a loop similar
to the loop you just studied.
Goal 3
Use an if conditional to change a Pixel's color conditionally
Lab6 followup
public void makeRedWhite()
{
    int x;
    x = 0;
    while( x < this.getWidth() )
    {
        //whiten a whole column..
        int y;
        y = 0;
        while( y < this.getHeight() )
        {
            Pixel pRef;
            pRef = this.getPixel(x,y);
            if (pRef.getRed() > (pRef.getGreen() + pRef.getBlue()))
            {
                pRef.setColor(new java.awt.Color(255,255,255));
            }
            y = y + 1;
        }
        x = x + 1;
    }
    return;
}
public void makeRedWhite()
{
    int x;
    x = 0;
    while( x < this.getWidth() )
    {
        //whiten a whole column..
        int y;
        y = 0;
        while( y < this.getHeight() )
        {
            Pixel pRef;
            pRef = this.getPixel(x,y);
            if ( pRef.getRed() > (pRef.getGreen() + pRef.getBlue()) )
            {
                pRef.setColor(new java.awt.Color(255,255,255));
            }
            y = y + 1;
        }
        x = x + 1;
    }
    return ;
}
Goal 3
Use an if conditional to change a Pixel's color *conditionally*
Lab6 followup

Mini-goal: Start saving your old versions of your work...directions are IN Project 4. (Due 3/25 in 4 more school days!)
(Optional: Google GitHub and find out what people do with it.)
public void makeRedWhite() {
    int x;
    x = 0;
    while (x < this.getWidth()) {
        // whiten a whole column..
        int y;
        y = 0;
        while (y < this.getHeight()) {
            Pixel pRef;
            pRef = this.getPixel(x, y);
            if (pRef.getRed() > (pRef.getGreen() + pRef.getBlue())) {
                pRef.setColor(new java.awt.Color(255, 255, 255));
            }
            y = y + 1;
        }
        x = x + 1;
    }
    return;
}
public void makeRedWhite()
{
    this.makeRedAGivenColor( new java.awt.Color(255,255,255) );
    return;
}
public void makeRedAGivenColor( java.awt.Color cParam)
    x = 0;
    while( x < this.getWidth() )
    {
        //whiten a whole column..
        int y;
        y = 0;
        while( y < this.getHeight() )
        {
            Pixel pRef;
            pRef = this.getPixel(x,y);
            if ( pRef.getRed() > (pRef.getGreen() + pRef.getBlue()) )
            {
                pRef.setColor( cParam );
            }
            y = y + 1;
        }
        x = x + 1;
    }
    return ;
}
Goal 4
Copying part of one Picture into another Picture (with G&E's API)
(This week's Lab7)
REPEAT (use one loop nested into a another loop)

Get a source Pixel.
Get a destination Pixel.
Get the Color from the destination Pixel.
Set the Color of the source Pixel TO the Color gotten from the destination Pixel.

clicker question coming up...
A) Get a source Pixel must be done BEFORE Get a destination Pixel.
B) Get a destination Pixel must be done BEFORE Get a source Pixel.
C) DOESN'T MATTER, as long as the other steps are done in the order marked by the arrows below.

Get a source Pixel. 
Get the Color from the destination Pixel. 
Set the Color of the source Pixel TO the Color gotten from the destination Pixel.
public void chromakeyR( Picture pictForRed )
{
    x = 0;
    while( x < this.getWidth() )
    {
        //process a whole column..
        int y;
        y = 0;
        while( y < this.getHeight() )
        {
            //Purpose: Set up to remember a Pixel from this portrait.
            Pixel pRef;
            //P: Get the Pixel from this portrait to analyze and
            //MAYBE change its color.
            pRef = this.getPixel(x,y);
            //P: Analyze the color and decide whether to replace it
            if ( pRef.getRed() > (pRef.getGreen() + pRef.getBlue()) )
            {
                //The code below is for replacing the color in a
                //portrait's Pixel with the color from the corresponding
                //Pixel of pictForRed
                //????????????????????????
            }
            y = y + 1; //Count one more pixel of a column done.
        }
        x = x + 1; //Count one more whole column done.
    }
    return ;
}
public void chromakeyR( Picture pictForRed )
{
    int x;
    x = 0;
    while( x < this.getWidth() )
    {
        //process a whole column ..
        int y;
        y = 0;
        while( y < this.getHeight() )
        {
            //Purpose: Set up to remember a Pixel from this portrait.
            Pixel pRef;
            //P: Get the Pixel from this portrait to analyze and
            //MAYBE change its color.
            pRef = this.getPixel(x,y);

            //P: Analyze the color and decide whether to replace it
            if ( pRef.getRed() > (pRef.getGreen() + pRef.getBlue()) )
            {
                //The code below is for replacing the color in a
                //portrait's Pixel with the color from the corresponding
                //Pixel of pictForRed
                //?????????????????????

                y = y + 1; //Count one more pixel of a column done.
            }
        }
        x = x + 1; //Count one more whole column done.
    }
    return ;
}
//P: Analyze the color and decide whether to replace it
if ( pRef.getRed() > (pRef.getGreen() + pRef.getBlue()) )
{
    //The code below is for replacing the color in a
    //portrait's Pixel with the color from the corresponding
    //Pixel of pictForRed

    //???????????????????

}
//P: Analyze the color and decide whether to replace it
if ( pRef.getRed() > (pRef.getGreen() + pRef.getBlue()) )
{
  // The code below is for replacing the color in a portrait's Pixel with the color from the corresponding Pixel of pictForRed

  //P: Set up memory for a reference to the Pixel that will provide the replacement color.
  Pixel pixFromCostume;
}

//P: Analyze the color and decide whether to replace it
if ( pRef.getRed() > (pRef.getGreen() + pRef.getBlue()) )
{
    //The code below is for replacing the color in a
    //portrait's Pixel with the color from the corresponding
    //Pixel of pictForRed

    //P: Set up memory for a reference to the
    //Pixel that will provide the replacement color.
    Pixel pixFromCostume;

    //P: Get the CORRESPONDING Pixel from the constume
    pixFromConstume = pictForRed( x, y );
}

//P: Analyze the color and decide whether to replace it
if ( pRef.getRed() > (pRef.getGreen() + pRef.getBlue()) )
{
    //The code below is for replacing the color in a
    //portrait's Pixel with the color from the corresponding
    //Pixel of pictForRed

    //P: Set up memory for a reference to the
    //Pixel that will provide the replacement color.
    Pixel pixFromConstume;

    //P: Get the CORRESPONDING Pixel from the constume
    pixFromConstume = pictForRed( x, y );

    //P: Set up memory for the replacement Color
    java.awt.Color color;
}

//P: Analyze the color and decide whether to replace it
if ( pRef.getRed() > (pRef.getGreen() + pRef.getBlue()) )
{
    //The code below is for replacing the color in a
    //portrait's Pixel with the color from the corresponding
    //Pixel of pictForRed

    //P: Set up memory for a reference to the
    //Pixel that will provide the replacement color.
    Pixel pixFromCostume;

    //P: Get the CORRESPONDING Pixel from the costume
    pixFromConstume = pictForRed( x, y );

    //P: Set up memory for the replacement Color
    java.awt.Color color;

    //P: GET the replacement color into memory
    color = /*COPY FROM!!*/  pixFromCostume.getColor();
}

//P: Analyze the color and decide whether to replace it if ( pRef.getRed() > (pRef.getGreen() + pRef.getBlue()) ) {

    //The code below is for replacing the color in a portrait's Pixel with the color from the corresponding Pixel of pictForRed

    //P: Set up memory for a reference to the Pixel that will provide the replacement color.
    Pixel pixFromCostume;

    //P: Get the CORRESPONDING Pixel from the constume
    pixFromConstume = pictForRed( x, y );

    //P: Set up memory for the replacement Color
    java.awt.Color color;

    //P: GET the replacement color into memory
    color = /*COPY FROM!!*/  pixFromCostume.getColor();

    //P: ACTUALLY CHANGE the red portrait's pixel's color
    pRef.setColor( color );

    //Now what ???
}
clicker..
We're in the if part of the conditional.
We just changed to color of one Pixel.
The if ( ... ) { ... } is inside a loop.
What do we code next INSIDE THE if part?

A) return  -- return to where in the App (main) the chromakeyR method was called from.
B) somehow stop the loop
C) NOTHING
D) x = x + 1;
E) y = y + 1;
public void chromakeyR( Picture pictForRed )
{
    x = 0;
    while( x < this.getWidth() )
    {
        //process a whole column ..
        int y;
        y = 0;
        while( y < this.getHeight() )
        {
            //Purpose: Set up to remember a Pixel from this portrait.
            Pixel pRef;
            //P: Get the Pixel from this portrait to analyze and
            //MAYBE change its color.
            pRef = this.getPixel(x,y);
            //P: Analyze the color and decide whether to replace it
            if ( pRef.getRed() > (pRef.getGreen() + pRef.getBlue()) )
            {
                //The code below is for replacing the color in a
                //portrait's Pixel with the color from the corresponding
                //Pixel of pictForRed
                //????????????????????
            }
            y = y + 1; //Count one more pixel of a column done.
        }
        x = x + 1; //Count one more whole column done.
    }
    return ;
}
//P: Analyze the color and decide whether to replace it
if ( pRef.getRed() > (pRef.getGreen() +
pRef.getBlue()) )
{
    //The code below is for replacing the color in a
    //portrait's Pixel with the color from the corresponding
    //Pixel of pictForRed

    //P: Set up memory for a reference to the
    //Pixel that will provide the replacement color.
    Pixel pixFromCostume;

    //P: Get the CORRESPONDING Pixel from the constume
    pixFromConstume = pictForRed( x, y );

    //P: Set up memory for the replacement Color
    java.awt.Color color;

    //P: GET the replacement color into memory
    color = /*COPY FROM!!*/ pixFromCostume.getColor();

    //P: ACTUALLY CHANGE the red portrait's pixel's color
    pRef.setColor( color );

    //Now what ???
}
We're in the if part of the conditional. We just changed to color of one Pixel.

The if ( ... ) { ... } is inside a loop. What do we code next INSIDE THE if part?

A) return -- return to where in the App (main) the chromakeyR method was called from.

B) somehow stop the loop

C) NOTHING

D) x = x + 1;

E) y = y + 1;
public void chromakeyR( Picture pictForRed )
    x = 0;
    while( x < this.getWidth() )
    {
        //whiten a whole column..
        int y;
        y = 0;
        while( y < this.getHeight() )
        {
            Pixel pRef;
            pRef = this.getPixel(x,y);
            if ( pRef.getRed() > (pRef.getGreen() + pRef.getBlue()) )
            {
                Pixel pixFromCostume;
                pixFromCostume = pictForRed.getPixel( x, y );
                java.awt.Color color;
                color = pixFromCostume.getColor( );
                pRef.setColor( color );
            }
            y = y + 1;
        }    
        x = x + 1;
    }
    return ;
}