University of Nevada, Reno

CS 302 — Data Structures

Assignment 1

Students:
Joshua Gleason
Josiah Humphrey

Instructor:
Dr. George Bebis

February 15, 2010

Contents

1 Introduction

In this assigment, the students were asked to make an image class that manipulated a Portable Gray Map (PGM) image in various ways. This was to be accomplished using dynamic 2D arrays and other tools that were learned in previous CS courses.

One of the first problems faced was how to use the supplied read and write image files. These functions were provided to simplefy the build process as they allowed the students to concentrate on the manipulation of the images.

After reading and writing images was mastered, the students undertook the bulk of the assignment which was to manipulate the images that had been read in with the supplied read function. This required the knowledge of manipulating dynamic 2D arrays. Dynamic 2D arrays was a new tool for the students' C++ repatoire, but the students soon came to understand the benefits of dynamic 2D arrays.

The assignment also required the use of constructors, desctructors, copy contrusctors, and operator overloading, These topics were review from CS 202, but the use of them refreshed the students minds on how they work and to their purposes.

The students also extensively documented their program and made it as easy as possible to understand what was happening in the various algorithms that were implemented. The students realize that commenting and documenting the source code for projects is extremely important and is essential to the success of a powerful programmer.

2 Use of Code

3 Functions

3.1 Image.h

CONSTRUCTOR

Purpose
: default constructor allocates no memory and sets the size to zero

Input
: None

Output
: None

Assumptions
: Sets everything to zero and sets the pixelValue array to NULL

CONSTRUCTOR WITH PARAMETERS

Purpose
: change the dimenstions of the image, delete, and re-allocate memory if required

Input
: An N, M, and Q value to set the new image to

Output

None

Assumptions

Sets the image to a certain size and intializes the image as a grid

DESCTRUCTOR

Purpose

Deletes and memory that has been dynamically allocated

Input

None

Output

None

Assumptions

Checks to see if the pixelValue array has been set if so, deletes

COPY_CONSTRUCTOR

Purpose

Creates a new array absed on the thing to be copied then sets the pixelValue of the new object the same as the old image

Input

ImageType rhs is the old image to be copied over into the new array

Output

None

Assumptions

The old image must be passed as reference to prevent an infinate loop

OPERATOR=

Purpose

equal operator overload, this is basically the same as the copy constructor except it will likely have to de-allocate memory before copying values, all this is decided in setImage-Info however

Input

imageType rhs which is the old iamge to be copied over to the new image

Output

Returns the imageType obejct so that equal chaining can be implemented

Assumptions

Assumes that the user is not trying to copy the same object into itself

GETIMAGEINFO

Purpose

returns the width height and color depth to reference variables

Input

- rows
  This parameter grabs the number of rows in the imageType object
- cols
  This parameter grabs the number of cols in the imageType object
- levels
  This paremeter grabs the depth of the image in the imageType object

Output

None

Assumptions

Assumes nothing but it makes sense that the object being queried has been loaded with some image

setImageInfo

Purpose

Sets the image info, deleting and allocating memory as required, also creates a background grid

Input

- rows
  This parameter sets the number of rows in the imageType object
- cols
  This parameter sets the number of cols in the imageType object
- levels
  This paremeter sets the depth of the image in the imageType object

Output

None

Assumptions

Assumes nothing

Example

getPixelVal

Purpose

Returns the value of a pixel

Input

- i
  The row of the pixel
- j
  The column of the pixel

Output

The integer value of the pixel at pixelValue[i][j]

Assumptions

It is assumed that the image has been intialized

Example

SETPIXELVAL

Purpose

Sets the value of a pixel

Input

- i
  The row of the pixel to be changed
- j
  The column of the pixel to be changed

Output

None

Assumptions

Assumes the image has been intialized

Example

GETSUBIMAGE

Purpose

Obtain a sub-image from old. Uses the coordinates of the upper left corner and lower right corner to obtain image.

Input

- ULr
  The upper left row of the pixel to be x in (0,0) in the new image.
- ULc
  The upper left column of the pixel to be y in (0,0) in the new image
- LRr
  The lower right row of the pixel to be x in (max_x, max_y) in the new image
- LRC
  The lower right row of the pixel to be y in (max_x, max_y) in the new image

Output

None

Assumptions

Assumes that the UL{r,c} and LR{r,c} have been properly bounds and error checked before the function call

Example

MEANGRAY

Purpose

this calculates the average gray value in the picture, this is done by adding all of the pixels and dividing by the total number of pixels

Input

None

Output

A double value that is the mean value of all the pixels in pixelValue

Assumptions

Assumes nothing and returns 0 if the image has not been intialized

Example

ENLARGEIMAGE

Purpose

This function enlarges an image by a magnitude of s, so for example if the original function was 100x100 and s is 10, then the new image is 1000x1000

Input

- S
  This is the magnitude of the enlargement
  The function is also overloaded to accept ints as well as doubles
- ImageType old
  This is the image to be enlarged
- cubic
  A bool value that decides which type of interpolation to use. If true, use cubic interpolation If false, use linear interpolation

Output

None

Assumptions

The method choosen to use was bicubic/linear interpolation which creates splines for each column(cubic or linear), then using those splines create an image which is a stretched version of the original image. The way this was achieved was to stretch the entire image only vertically, and then stretch that image horizontally. Then the same thing was done except reversed (stretched image horizontally first) and then the two image summed together. This gives an average value between both methods. Although it can handle S values less than 1, the shrinkImage function works better for this.

SHRINKIMAGE

Purpose

Shrink image, average all the values in the block to make the new pixel, this makes the shrink much less jagged looking in the end

Input

- s
  The inteter value of the shrink factor
- ImageType old
  The image to be shrunk

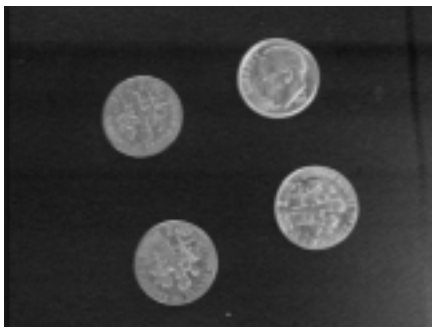
Output

None

Assumptions

Assumes the image has been intialized and that error checking has been done.

Example



reflectImage

Purpose

reflects image by moving the pixel to N or M minus the current row or column depending on the value of the flag (true being a horizontal reflection and false being a vertical reflection)

Input

- flag
  The flag that sets either vertical or horizontal reflection
- ImageType old
  The image to be reflected

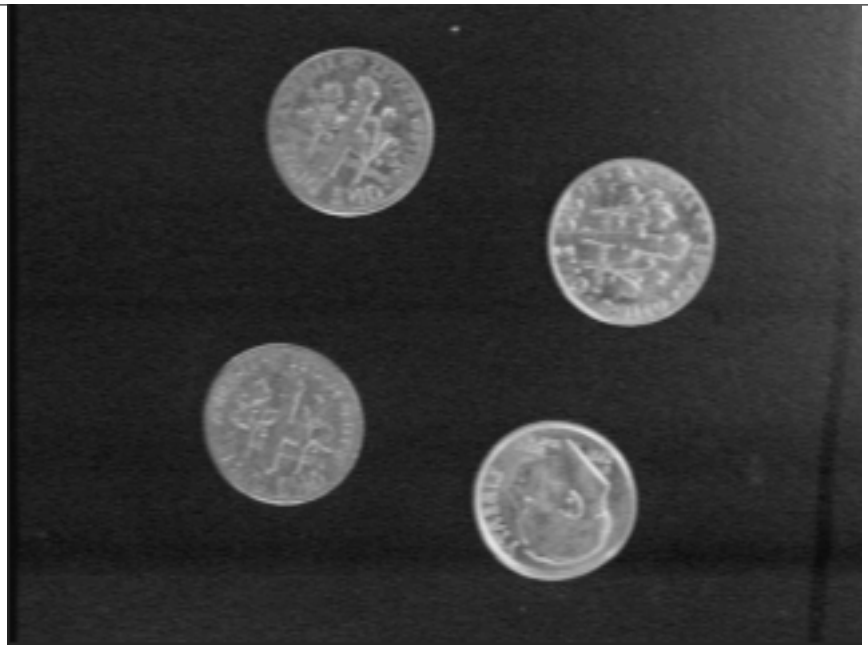
Output

None

Assumptions

Assumes nothing, but it makes sense to have an intialized image to reflect

Example



translateImage

Purpose

Translate the image down to the right, any part that goes out of the screen is not calculated. Checkered background from setImageInfo is retained.

Input

- t
  The integer value of the translation. The translation will occur down and to the right 't' pixels
- ImageType old
  The image to be translated

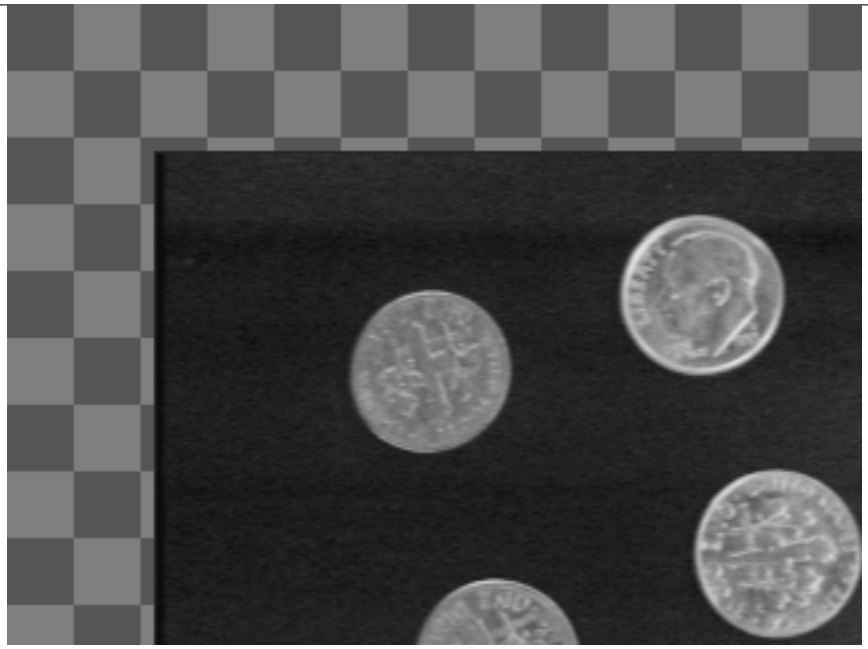
Output

None

Assumptions

No assumptions are made, but it makes sense to have an intialized image

Example



rotateImage

Purpose

Rotate the image clockwise using bilinear interpolation, basically traversing the entire image going from the destination to the source by using the in reverse (which is why its clockwise). Once a location is determined the surrounding pixels are used to calculate intermediate values between the pixels, this gives a pretty smooth rotate.

Input

- theta
  The degrees to rotate. This is converted to radians inside the function
- ImnageType old
  The image to be rotated

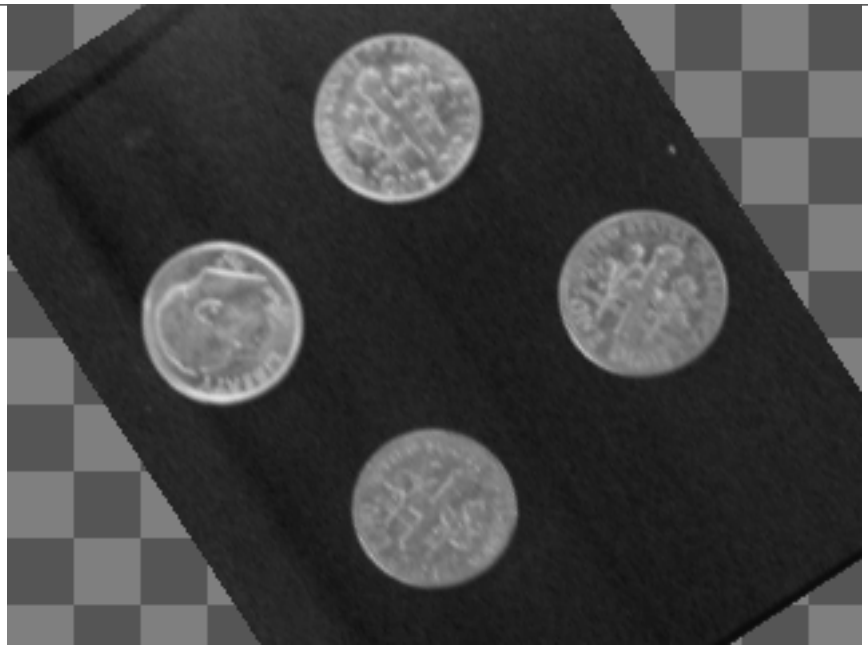
Output

None

Assumptions

Assumes that theta is in degrees because theta is converted to radians from degrees inside the function for the use of the trig functions. It is also assumed that the image has been intialized before the function call. It is also assumed that theta is between 0 and 360.

Example



OPERATOR+

Purpose

Sum two images together, basically just finding the average pixel value of every pixel between two images. Throws an exception if dimesions of both images don't match

Input

- ImageType rhs
  This is the image to be added to 'this' image

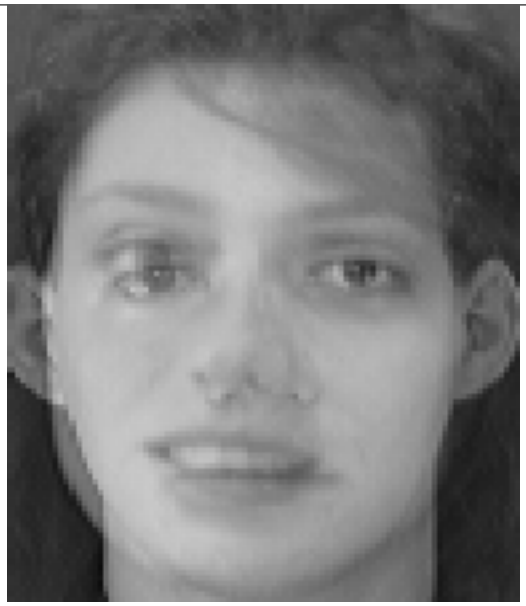
Output

ImageType object to chain additions

Assumptions

It is assumed that each image have the same dimensions. However, if the images do not have the same dimensions, then a string is thrown stating that the images do not have the same dimensions. It is not neccesary to have each image initialized, but it makes senses that they would each be initialized.

Example



OPERATOR-

Purpose

subtract two images from each other to see the differences, if the magnitude of the difference is less then Q/6 then the pixel is replaced with black, otherwise white is used. This seems to help reduce the amount of noise in the pictures

Input

- ImageType rhs
  This is the image to be subtracted from 'this' image

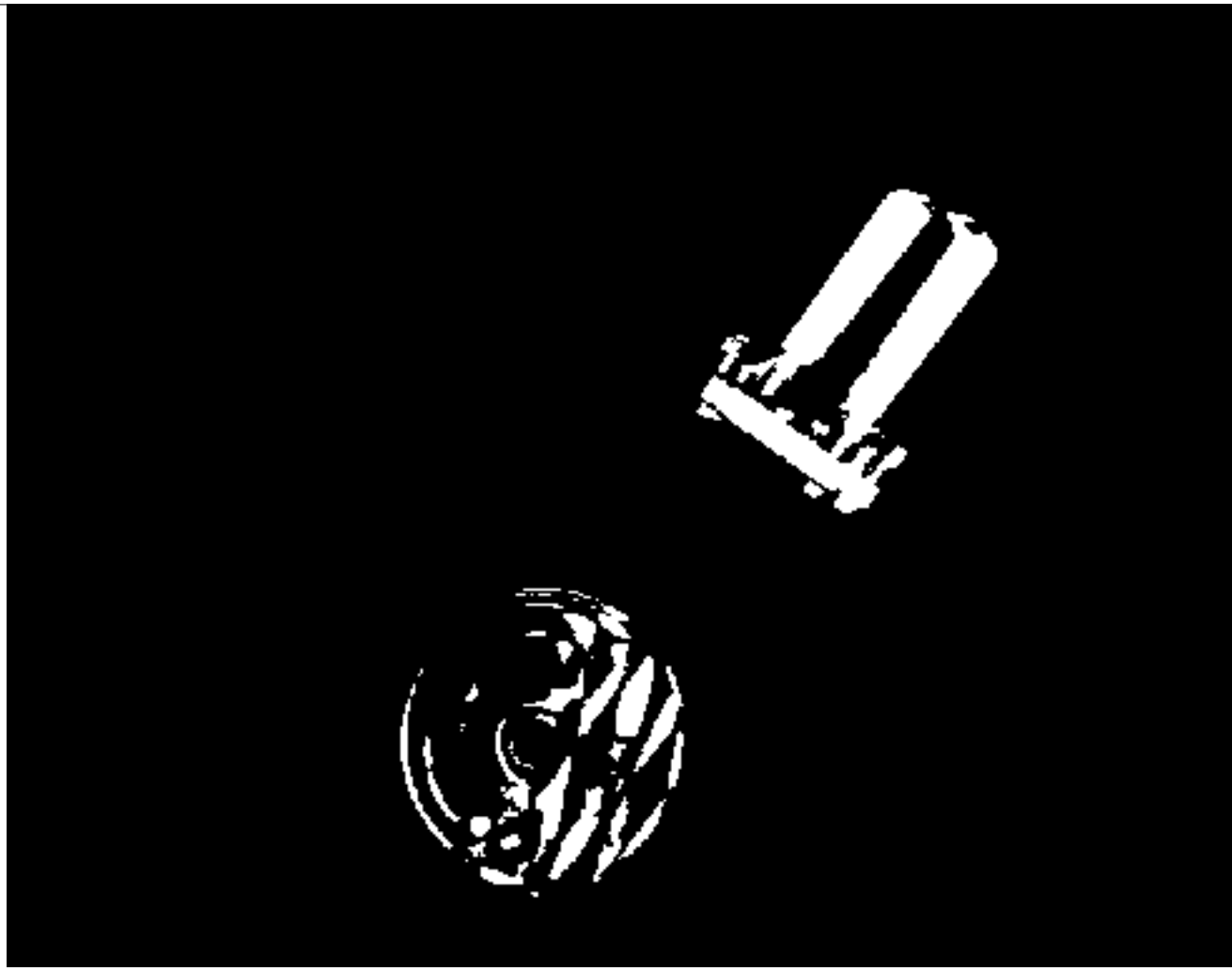
Output

ImageType is returned to allow chaining of subtraction

Assumptions

It is assumed that each image have the same dimensions. However, if the images do not have the same dimensions, then a string is thrown stating that the images do not have the same dimensions. It is not neccesary to have each image initialized, but it makes senses that they would each be initialized.

Example



NEGATEIMAGE

3.2 driver.cpp

SHOWMENU

Purpose

This is the function which builds the scrolling menu system, this simply creates a curses window and puts all the options stored in menuStr onto the window, it then waits for the user to press UP, DOWN, or RETURN before reacting. The parameters allow menus to be different widths, heights, and locations. A few constants can be changed to change the colors of the window.

Input

An un-initialized window pointer, a string to be the title, height, width, xLoc, yLoc of the window, list of c-style strings to be used in the menu, the number of menu options, and a bool value which says weather the last choice is left highlighted.

Output

Display a window with menu options, let user choose and return the index of that choice.

Assumptions

Assumes that window is un-intialized and will be destructed by calling function.

SHOWREGS

Purpose

Display a window of registers next to the main menu (or wherever the constants dictate).

Input

An un-initialized window pointer, a title string, and a list of register names.

Output

Displays a window next to main of all the registers.

Assumptions

This allocates memeory for a WINDOW but it doesn't delete it.

DRAWWINDOW

Purpose

This function simply draws an empty window with a given title, height, width, x, and y locations. The colors have default values but can be changed if oddly colored windows are wanted.

Input

An un-intialized window pointer, a title string, and then the window height, width, yLoc, and xLoc, plus the colors for the background and foreground which are defaulted to some constants.

Output

Displays a empty window with a border and title using the given parameters.

Assumptions

This allocates memeory for a WINDOW but it doesn't delete it.

DELETEMENU

Purpose

This basically clears the entire screen after deleting the window that is passed.

Input A WINDOW pointer that is has been initialized.

Output

De-allocate memory for the window pointer and refresh the main screen.

Assumptions

Assumes WINDOW object is intialized before calling.

PROCESSENTRY

Purpose

This is the function that decides where to go depending on the choice in the main menu. The reason it has all the parameters is for passing to the subsequent functions that will be using them.

Input List of register images, list of register bools, list of register names, and a value assumed to be choosen by user.

Output

Depending on the value of choice, call a function to do some image manipulation.

Assumptions

Assumes value ¿= 0 and ¡ MENU_OPTIONS, not that anything will crash if its not true, but nothing will happen, also assumes that names contain valid c strings.

STDWINDOW

Purpose

This just builds the window used for message box, this function is just to simplify the plethora of other functions that use this.

Input

An un-initalized window and a title string.

Output

Displays a window in the standard text box location with the title and a border.

Assumptions

The window object is initalized here but not deleted, this is left up to the calling function.

PROMPTFORREG

Purpose

This is the function that calls the menu for the register prompt, it can be called in different locations (like in addImg and subImg) but has a default defined by some global constants. The function creates a list of registers and adds the "Back" option as the final option, this way the user has the option to cancel choosing a register. Although in the program it looks like the register display and register choosing window are the same, this menu overlaps the other menu to make it seem like control is transfering to another window.

Input

List of register bools, list of regist names, a flag that indicates if registers that have not been loaded can be choosen, the yLoc, and xLoc of the menu.

Output

Display a menu with the registers in it, allowing user to choose a register.

Assumptions

Assumes that names are already set to valid c strings.

PROMPTFORFILENAME

Purpose

Create a message box and prompt the user for a string value with given prompt.

Input

Title string, prompt string and char used to store user input.

Output

Sets the final parameter equal to the filename the user chooses and returns the length.

Assumptions

Assumes first 2 parameters are valid c strings and that the final parameter is a string of at least length 16 plus the length of the file path declared as a constant.

PROMPTFORLOC

Purpose

This function prompts the user for a location (both row and column) and sets the valid points equal to row or col. If -1 is returned in either location it means user choose to cancel the prompt. The validity of the points is calculated by the image object it is passed. The image properties are calculated and then used to determine the bounds of row and column.

Input

A prompt string, image object, and 2 integers passed by reference.

Output

Sets two reference parameters equal to row and column of users choice.

Assumptions

Assumes image is intialized and has a valid height and width also that first parameter is a valid c string.

PROMPTFORPIXVALUE

Purpose

Prompt for a pixel value which is from 0 to maxVal, if not display message box and re-prompt user until valid choice is made.

Input

A title string, prompt string and max input value.

Output

Prompts user in message window and returns the value when the user inputs a valid value(-1 indicates cancel).

Assumptions

Assumes that first 2 parameters are valid c strings.

PROMPTFORSCALEVALUE

Purpose

This function prompts the user for a scale value and checks to make sure it is not greater than maxVal and not less than 2. This is used in the enlarge and shrink functions.

Input

Title string, prompt string and max input value.

Output

Prompts user in message window and returns the value when the user inputs a valid value(-1 indicates cancel).

Assumptions

Assumes that first 2 parameters are valid c strings.

PROMPTFORMIRROW

Purpose

Prompt the user for the characters h, v, or c (not case sensitive) and return the value as soon as one of the 3 is pressed.

Input

A title, prompt string.

Output

Returns users choice as a char.

Assumptions

Both parameters are valid c strings.

PROMPTFORANGLE

Purpose

Prompt user for a valid angle using a message box, make sure input is between 0 and 360, if not display a message box and then re-prompt.

Input

A title and prompt string

Output

Returns the user angle choice.

Assumptions

Both parameters are valid c strings.

MESSAGEBOX

Purpose

Displays a message box in the center of the screen with the message displayed in it.

Input

A title and message string.

Output

Displays a message box in the center of the screen with the message displayed in it, then waits for the user to press return before continuing.

Assumptions

Assumes both parameters are valid c strings.

FILLREGS

Purpose

This displays a simple message box to the screen with the given title and msg inside of it, it waits for the user to press RETURN before returning to calling function.

Input

List of images, bools, and c strings all of length REGS also the number of arguments passed to main and the array of strings passed to main.

Output

Sets valid arguments to registers (loading images) and clears the rest of the registers.

Assumptions

Assumes that char** is a valid list of strings with int rows.

CLEARREGISTERS

Purpose

Prompt for a register that is filled and then clear it.

Input

List of images of length REGS, list of bools of length REGS, and list of c strings of length REGS.

Output

Prompt the user for which register to use, then if nessessary prompt for additional information and call the function in ImageType that will allow the image in that register to be manipulated.

Assumptions

Assumes all names in the c string list are valid c strings and the bools coincide with the image types that reference the same index.

LOADIMAGE

Purpose

Prompt the user for a register to load to, then let them choose from a list of the .pgm files in the local images directory (defined as a constant).

Input

List of images of length REGS, list of bools of length REGS, and list of c strings of length REGS.

Output

Prompt the user for which register to use, then if nessessary prompt for additional information and call the function in ImageType that will allow the image in that register to be manipulated.

Assumptions

Assumes all names in the c string list are valid c strings and the bools coincide with the image types that reference the same index.

SAVEIMAGE

Purpose

Save image from a register to the local images directory, prompting user for register and file name.

Input

List of images of length REGS, list of bools of length REGS, and list of c strings of length REGS.

Output

Prompt the user for which register to use, then if nessessary prompt for additional information and call the function in ImageType that will allow the image in that register to be manipulated.

Assumptions

Assumes all names in the c string list are valid c strings and the bools coincide with the image types that reference the same index.

GETIMAGEINFO

Purpose

Simply retrieve image information and display to a window below the registers The data being displayed is the Register number, Image Height, Width, Q value, and average gray value.

Input

List of images of length REGS, list of bools of length REGS, and list of c strings of length REGS.

Output

Prompt the user for which register to use, then if nessessary prompt for additional information and call the function in ImageType that will allow the image in that register to be manipulated.

Assumptions

Assumes all names in the c string list are valid c strings and the bools coincide with the image types that reference the same index.

SETPIXEL

Purpose

Prompt user for a register then a pixel location (row, col) and then the pixel value to change that pixel to.

Input

List of images of length REGS, list of bools of length REGS, and list of c strings of length REGS.

Output

Prompt the user for which register to use, then if nessessary prompt for additional information and call the function in ImageType that will allow the image in that register to be manipulated.

Assumptions

Assumes all names in the c string list are valid c strings and the bools coincide with the image types that reference the same index.

GETPIXEL

Purpose

Return the value of a pixel in a selected image to the user.

Input

List of images of length REGS, list of bools of length REGS, and list of c strings of length REGS.

Output

Prompt the user for which register to use, then if nessessary prompt for additional information and call the function in ImageType that will allow the image in that register to be manipulated.

Assumptions

Assumes all names in the c string list are valid c strings and the bools coincide with the image types that reference the same index.

EXTRACTSUB

Purpose

After getting the image to manipulate, prompt for two corners to make a subimage out of, if the lower right corner is above or left of the upper right corner re-prompt for valid points

Input

List of images of length REGS, list of bools of length REGS, and list of c strings of length REGS.

Output

Prompt the user for which register to use, then if nessessary prompt for additional information and call the function in ImageType that will allow the image in that register to be manipulated.

Assumptions

Assumes all names in the c string list are valid c strings and the bools coincide with the image types that reference the same index.

ENLARGEIMG

Purpose

This function prompts the user for a scale value to enlarge an image by, it makes sure the scale value does not make the image larger than MAX_IMG value because it may cause a stack overflow.

Input

List of images of length REGS, list of bools of length REGS, and list of c strings of length REGS.

Output

Prompt the user for which register to use, then if nessessary prompt for additional information and call the function in ImageType that will allow the image in that register to be manipulated.

Assumptions

Assumes all names in the c string list are valid c strings and the bools coincide with the image types that reference the same index.

SHRINKIMG

Purpose

The same as enlarge except it shrinks the image making sure it never gets smaller than MIN_IMG. This is because some image viewers won't open images as small as 2x2 (xv for example).

Input

List of images of length REGS, list of bools of length REGS, and list of c strings of length REGS.

Output

Prompt the user for which register to use, then if nessessary prompt for additional information and call the function in ImageType that will allow the image in that register to be manipulated.

Assumptions

Assumes all names in the c string list are valid c strings and the bools coincide with the image types that reference the same index.

REFLECTIMG

Purpose

Prompt user for a direction to reflect an image then reflect the image and store it back in the original register image.

Input

List of images of length REGS, list of bools of length REGS, and list of c strings of length REGS.

Output

Prompt the user for which register to use, then if nessessary prompt for additional information and call the function in ImageType that will allow the image in that register to be manipulated.

Assumptions

Assumes all names in the c string list are valid c strings and the bools coincide with the image types that reference the same index.

TRANSLATEIMG

Purpose

This prompts the user for how far to translate the image, then calls the translate function which moves the image down to the right 't' number of pixels. Also Won't let user choose t value that would move image totaly off the screen.

Input

List of images of length REGS, list of bools of length REGS, and list of c strings of length REGS.

Output

Prompt the user for which register to use, then if nessessary prompt for additional information and call the function in ImageType that will allow the image in that register to be manipulated.

Assumptions

Assumes all names in the c string list are valid c strings and the bools coincide with the image types that reference the same index.

ROTATEIMG

Purpose

This prompts the user for an angle theta which will rotate the image counter clockwise by theta degrees. The input is only valid from 0 to 360 which should cover all possibilities.

Input

List of images of length REGS, list of bools of length REGS, and list of c strings of length REGS.

Output

Prompt the user for which register to use, then if nessessary prompt for additional information and call the function in ImageType that will allow the image in that register to be manipulated.

Assumptions

Assumes all names in the c string list are valid c strings and the bools coincide with the image types that reference the same index.

SUMIMG

Purpose

Prompt for 2 images and attempt to sum them, there is no size checking because operator+ will throw a string which will be handeled by main if sizes of the two images are different.

Input

List of images of length REGS, list of bools of length REGS, and list of c strings of length REGS.

Output

Prompt the user for which register to use, then if nessessary prompt for additional information and call the function in ImageType that will allow the image in that register to be manipulated.

Assumptions

Assumes all names in the c string list are valid c strings and the bools coincide with the image types that reference the same index.

SUBTRACTIMG

Purpose

Prompt for 2 images and attempt to calculate the difference, there's no size checking here for the same reason sumImg doesn't do size checking

Input

List of images of length REGS, list of bools of length REGS, and list of c strings of length REGS.

Output

Prompt the user for which register to use, then if nessessary prompt for additional information and call the function in ImageType that will allow the image in that register to be manipulated.

Assumptions

Assumes all names in the c string list are valid c strings and the bools coincide with the image types that reference the same index.

NEGATEIMG

Purpose

Prompt user for which image to negate and negate it, pretty simple function.

Input

List of images of length REGS, list of bools of length REGS, and list of c strings of length REGS.

Output

Prompt the user for which register to use, then if nessessary prompt for additional information and call the function in ImageType that will allow the image in that register to be manipulated.

Assumptions

Assumes all names in the c string list are valid c strings and the bools coincide with the image types that reference the same index.

FINDLOCALPGM

Purpose

A somewhat brittle function that reads all the .pgm files from a local directory (defined as a constant) and places them into a dynamically allocated c style string array.

Input

One un-intialized double pointer of chars

Output

Allocates enough memory for a list of all the ".pgm" files in the local path specified by the FILELOC constant. It then copys the file names to the array and returns the number of rows in the array.

Assumptions

Pointer parameter is not initialized, but will be in the function this means it needs to be de-allocated later.

3.3 cubicspline.h

CONSTRUCTOR

Purpose

Input

Output

Assumptions

COPY CONSTRUCTOR

Purpose

Input

Output

Assumptions

CONSTRUCTOR WITH PARAMETERS

Purpose

Input

Output

Assumptions

DESTRUCTOR

Purpose

Input

Output

Assumptions

CREATE

Purpose

Input

Output

Assumptions

CREATECUBIC

Purpose

Input

Output

Assumptions

GETVAL

Purpose

Input

Output

Assumptions

GETCUBICVAL

Purpose

Input

Output

Assumptions

3.4 imageIO.h

READIMAGEHEADER

Purpose

Reads the iamge header amd puts them into values that are passed by reference

Input

- fname[]
  This is the name of the file stored as a C-style string
- N
  This is the number of rows in the image
- M
  This is the number of columns in the image
- Q
  This is the depth of the image
- type
  This makes sure that the file type is .pgm and not some other format

Output

None

Assumptions

Assumes that a file exists and is in pgm format

READIMAGE

Purpose

Reads the image into the image object from a file

Input

- fname[]
  The C-style string to hold the image file name
- ImageType image
  The image object that holds the image data

Output

None

Assumptions

Assumes there is a file to be read and that the user has read access

WRITEIMAGE

Purpose

Writes the image to disk

Input

- fname[]
  The C-style string to hold the image file name
- ImageType image
  The image object that holds the image data

Output

None

Assumptions

Assumes the user has write access to the destination folder

3.5 comp_curses.h

STARTCURSES

Purpose

This initializes the curses screen and its functions

Input

None

Output

None

Assumptions

No assumptions are made besides have a terminal capable of displaying curses correctly.

ENDCURSES

Purpose

This ends the curses screen and its functions

Input

None

Output

None

Assumptions

This assumes that curses has ibeen initialized with startCurses()

SETCOLOR

Purpose

This sets the colors for stdscr

Input

- *somewin
  This is the window pointer to set the colors to a specific window
- cf
  This is the first color (foreground) for the color pair to set in the window
- cb
  This is the second colod (background) for the color pair to set in the window

Output

None

Assumptions

Assumes that screen has been initialized

SCREENWIDTH

Purpose

Returns the max screen x value

Input

None

Output

The int value of the max x value for the entire terminal

Assumptions

Assumes startCurses() has been run

SCREENHEIGTH

Purpose

Returns the max screen y value

Input

None

Output

The int value of the max y value for the entire terminal

Assumptions

Assumes startCurses() has been run

PROMPTFORINT

Purpose

Prompts for an int at some int at some (x,y) cordinate

Input

- *somewin
  Some window to prompt for the int in
- y
  The y cordinate at which to prompt for the int
- x
  The x cordinate at which to prompt for the int
- promptString[]
  The string to display when prompting for the int

Output

The integer value of the user's input

Assumptions

It is assumed that startCurses() has been run. The function has built in error checking to prevent bad data from being input

PROMPTFORDOUBLE

Purpose

Prompts for a double at some int at some (x,y) cordinate

Input

- *somewin
  Some window to prompt for the double in
- y
  The y cordinate at which to prompt for the double
- x
  The x cordinate at which to prompt for the double
- promptString[]
  The string to display when prompting for the double

Output

The double value of the user's input

Assumptions

It is assumed that startCurses() has been run. The function has built in error checking to prevent bad data from being input (such as multiple periods)

PROMPTFORSTRING

Purpose

Prompts for a string at some (x,y) cordinate

Input

- *somewin
  The window at which to prompt for the string
- y
  The y cordinate at which to prompt
- x
  The rxy cordinate at which to prompt
- promptstring
  The string to display when prompting for the string
- str[]
  The array for the string that is typed in by the user
- len
  The length of the string stored

Output

None

Assumptions

It is assumed that startCurses() has been run. The function also accounts for backspaces and makes sure that only valid input is entered.

4 Bugs and Errors

hmm what goes here

5 What was Learned

lol

6 Division of Labor

ok!