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

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 setImageInfo 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

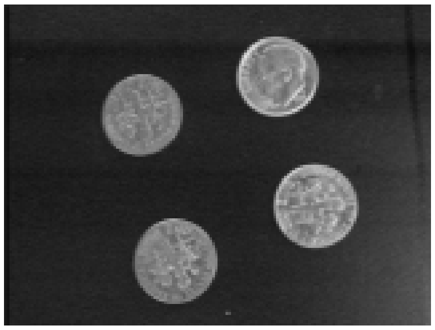
Output

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

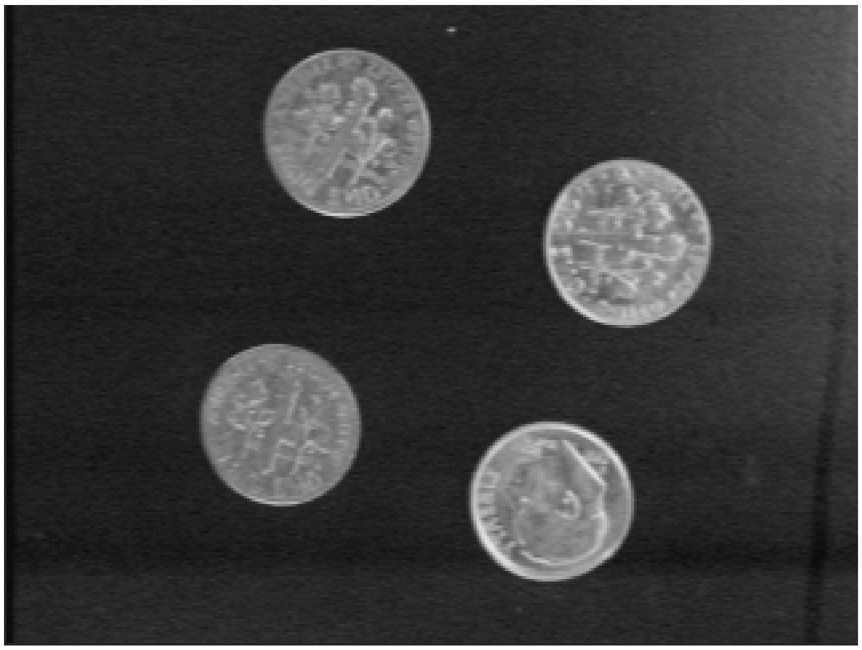
Output

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

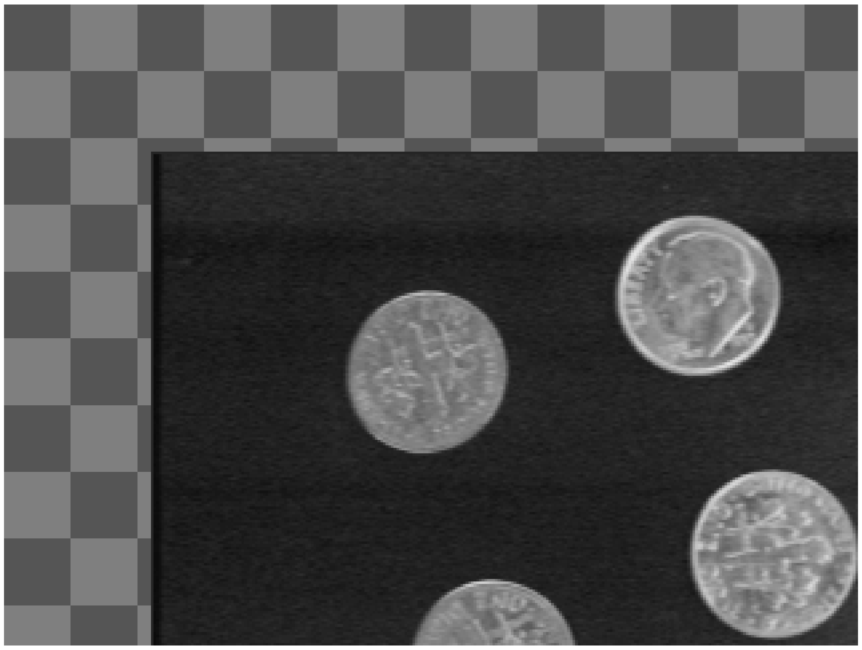
Output

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

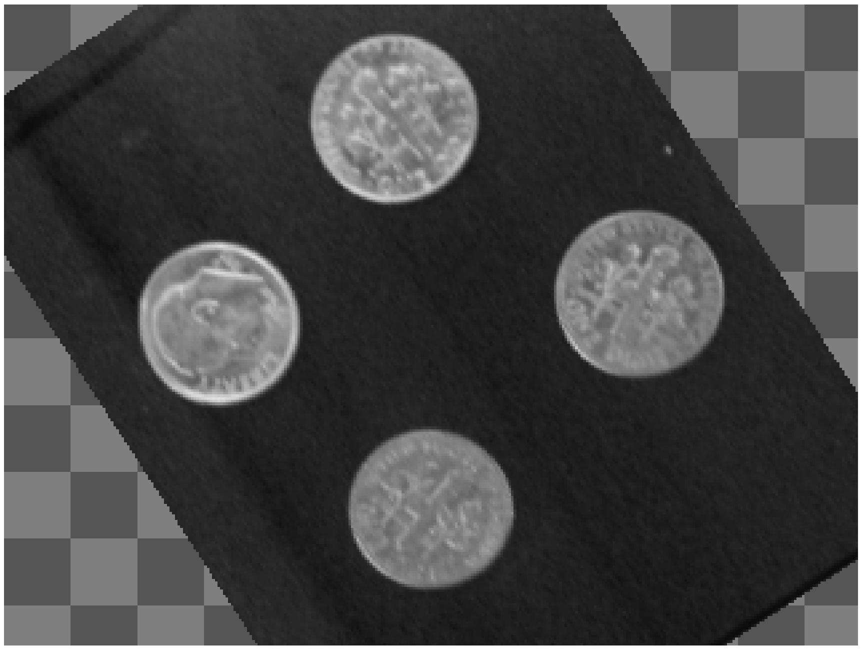
Output

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

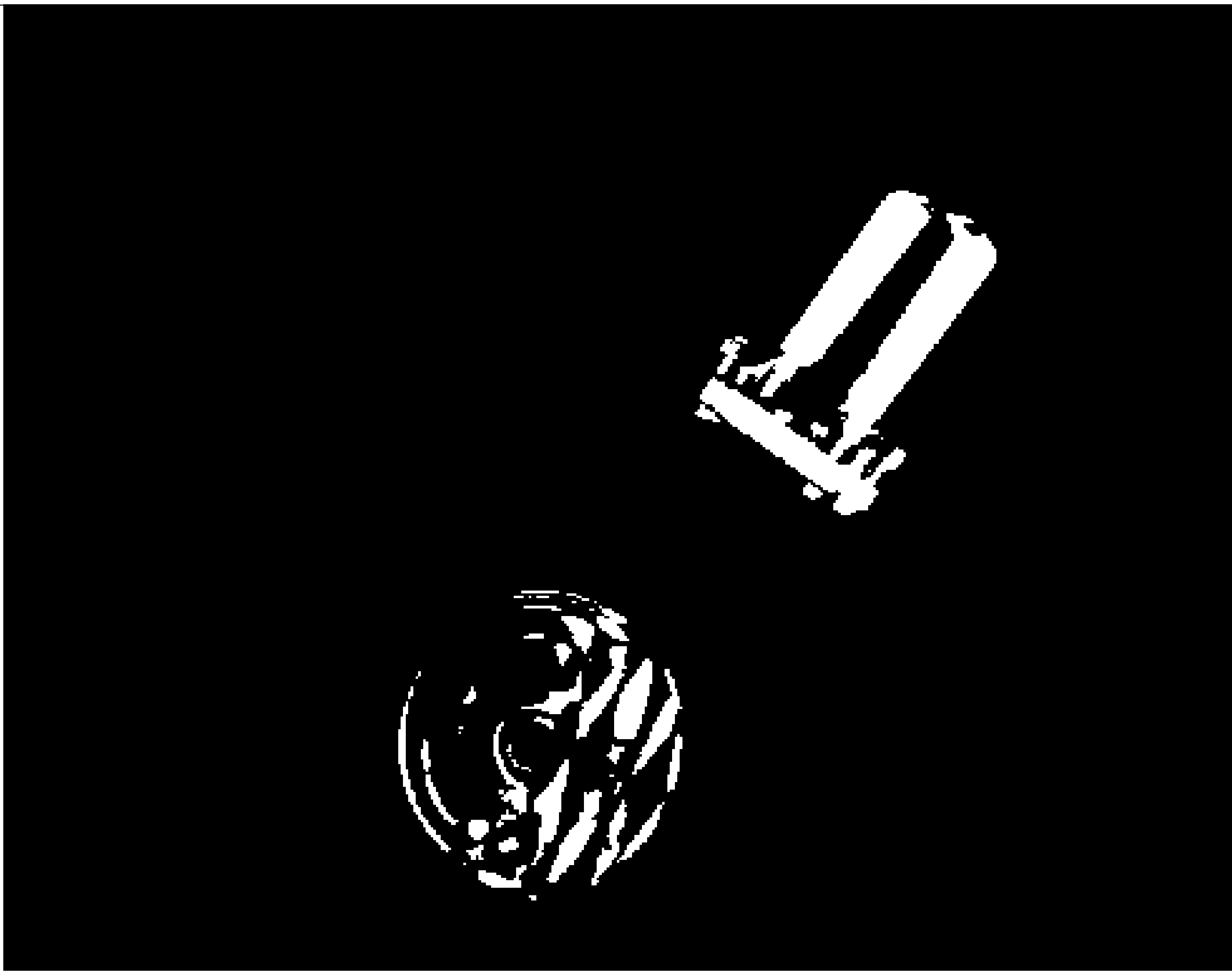
Output

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

Input

Output

Assumptions

showRegs

Purpose

Input

Output

Assumptions

DRAWWINDOW

Purpose

Input

Output

Assumptions

DELETEMENU

Purpose

Input

Output

Assumptions

DELETEWINDOW

Purpose

Input

Output

Assumptions

PROCESS ENTRY

Purpose

Input

Output

Assumptions

STDWINDOW

Purpose

Input

Output

Assumptions

PROMPTFORREG

Purpose

Input

Output

Assumptions

PROMPTFORFILENAME

Purpose

Input

Output

Assumptions

PROMPTFORLOC

Purpose

Input

Output

Assumptions

PROMPTFORPIXVALUE

Purpose

Input

Output

Assumptions

PROMPTFORSCALEVALUE

Purpose

Input

Output

Assumptions

PROMPTFORMIRROW

Purpose

Input

Output

Assumptions

PROMPTFORANGLE

Purpose

Input

Output

Assumptions

MESSAGEBOX

Purpose

Input

Output

Assumptions

FILLREGS

Purpose

Input

Output

Assumptions

CLEARREGISTERS

Purpose

Input

Output

Assumptions

LAODIMAGE

Purpose

Input

Output

Assumptions

SAVEIMAGE

Purpose

Input

Output

Assumptions

GETIMAGE

Purpose

Input

Output

Assumptions

SETPIXEL

Purpose

Input

Output

Assumptions

GETPIXEL

Purpose

Input

Output

Assumptions

EXTRACTSUB

Purpose

Input

Output

Assumptions

ENLARGEIMG

Purpose

Input

Output

Assumptions

SHRINKIMG

Purpose

Input

Output

Assumptions

REFLECTIMG

Purpose

Input

Output

Assumptions

TRANSLATEIMG

Purpose

Input

Output

Assumptions

ROTATEIMG

Purpose

Input

Output

Assumptions

SUMIMG

Purpose

Input

Output

Assumptions

SUBTRACTIMG

Purpose

Input

Output

Assumptions

NEGATEIMG

Purpose

Input

Output

Assumptions

FINDLOCALPGM

Purpose

Input

Output

Assumptions

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!