CS 2110 Timed Lab 3

Due Date and Time

Day: Thursday, July 14th

Time: Before the end of lab (5:45pm)

Warning

Please double check over your submission after you submit it. Please download what you submit and check over it to ensure you have submitted the correct file. Failure to do so will result in an automatic zero (in the case of an empty file) on the timed lab.

Policy

Submission

TURN IN THIS ASSIGNMENT ELECTRONICALLY USING T-SQUARE. SUBMISSIONS WHICH ARE LATE WILL NOT BE ACCEPTED. EMAIL SUBMISSIONS WILL NOT BE ACCEPTED UNDER ANY CIRCUMSTANCES! IN ADDITION IF YOU FORGET TO HIT THE SUBMIT BUTTON YOU WILL GET A ZERO.

Questions

If you are unsure of what questions mean, the TA's will clarify them to the best of their ability. In the end you are solely responsible for what you submit. We will not be able to answer any questions about how to reach a solution to the lab questions.

What's Allowed

- The assignment files
- Your Previous Homework submission
- Your mind
- Blank paper for scratch work

What's Not Allowed

- The Internet (except the T-Square Assignment page to submit)
- Any resource on T-Square that is not given in the assignment.
- Dropbox (if your harddrive crashes we will let you retake it!)
- Notes on paper or saved on your computer.
- Textbook
- Email
- IM

·Contact in any form with any other person besides TAs
·If you have any questions on what you may not use then assume you can't use it and ask a TA.

Other Restrictions

1. You may not leave the classroom until we have verified that you have submitted the lab. If you leave the classroom without submitting you will receive a zero.

2. **YOU MUST SUBMIT BY THE END OF YOUR LAB PERIOD**. Bear in mind that the clock on your computer may be a few minutes slow. You are supposed to have a full class period to work, and we are letting you use the 10 minutes between classes to make sure you have submitted your work. **WE WILL NOT ACCEPT LATE SUBMISSIONS**, be they 1 second or 1 hour late.

3. The timed lab has been configured to accept one submission. If you accidentally submit or submit the wrong version flag one of the TAs and we will reopen submission for you.

Violations

Failure to follow these rules will be in violation of the Georgia Tech Honor Code. **AND YOU WILL RECIEVE A ZERO** and you will be reported to Bill and the Office of Student Integrity.

We take cheating and using of unauthorized resources **VERY SERIOUSLY** and you will be in serious trouble if you are caught.

Remember

1. There is a lot of partial credit given and most of it is following the directions.

2. We allow you to use your homework assignment.

3. Please don’t get stressed out during a timed lab. You have plenty of time however use your time effectively

4. Remember don’t get stressed partial credit will be given. Do the best you can!

5. If you don’t know something at least **TRY** do not just walk out of the lab or submit an empty file Partial credit!

6. Remember what you can and can’t use if you don’t know then don’t use it and ask a TA if you can use it. If we catch you with unauthorized resources we will give you a zero, so better to be safe than sorry.

The assignment

You will be coding ONE algorithm. The ONE algorithm you must implement is explained below:

Decompressing an RLE encoded image via drawDecompressedImage

You will write an algorithm to take a compressed image and draw the decompressed version of the image onto the screen. To make things easier for you here the image you will be decompressing will be 240x160. To explain the format of RLE and the principles behind it, read this paragraph.

> **Run-length encoding** (**RLE**) is a very simple form of data compression in which *runs* of data (that is, sequences in which the same data value occurs in many consecutive data elements) are stored as a single data value and count, rather than as the original run. This is most useful on data that contains many such runs: for example, simple graphic images such as icons, line drawings, and animations. It is not useful with files that don't have many runs as it could greatly increase the file size.

RLE gives good compression if your image has a value that is repeated in contiguous elements. Instead of storing all of these elements you store the number of times the element occurred then the element that was repeated. As an example, assume you have this array:

```
u16 array[16] = {0x0000, 0x0001, 0x0001, 0x0001, 0x0001, 0x0001,
0x0001,
0x0000, 0x0003, 0x0004, 0x0004, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF};
```

An RLE encoded version of this data would look like so.

```
u16 array[12] = {1, 0x0000, 6, 0x0001, 1, 0x0000, 1, 0x0003, 2,
0x0004, 5,
0xFFFF};
```

Here we save 4 elements which means a total savings of 8 bytes! The numbers in decimal and are **bolded** in the above array are called *runs*. Notice how 0x0000 appeared once therefore its run is 1, the 0x0001 appeared 6 times so its run is 6. The run is stored before the data.

The format of how the data will be given to you is as follows:

run, rundata, run, rundata, …

Your implementation should be able to draw an image given this compressed format. **You should use DMA to draw any run that is greater than 2.**

Notes:

· Remember that the image width and height for drawDecompressedImage is 240 by 160. You know if you have drawn this many pixels that you are done. This is why the lengths of the compressed arrays are not given, you don't need to know them.

· You aren't given the length of the array containing compressed image data. You don't need this. You know you're done when you have drawn 240*160, 38400 pixels. **Don't bother trying to use sizeof** because it won't work properly.

Hints:

· Pointers! Pointers! Pointers! for drawDecompressedImage, you need to know where you are in the image and videoBuffer, you may alternatively keep track of the offsets, but your knowledge of pointers is what we are testing here.

· Remember the videoBuffer is a linear 1D array acting as a 2D array here this means that rows are stored next to each other in memory. This means that videoBuffer + 240 is the starting address of the second row and videoBuffer + 239 is the last pixel on the first row.

· Volatile. Remember that declaring a variable volatile forces the optimizer not to optimize access to that variable, by putting it in a register. It forces the variable to live in memory and not in a register.

· DMA operates on memory address and not values themselves.

· You are free to manually set the videoBuffer like an array without calling setPixel if the run is <= 2.

Super Hints:

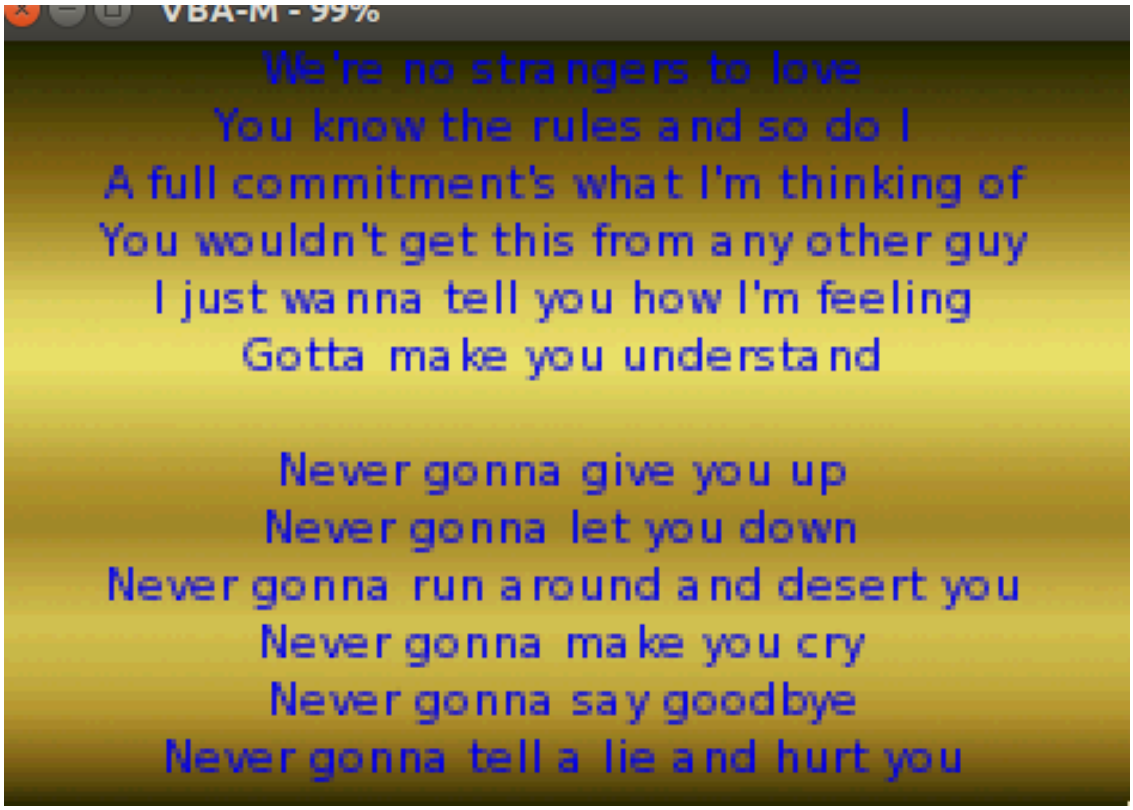
· Volatile

Restrictions:

· **For runs of 3 or more pixels you must use DMA** to draw the run onto the screen.

· **You may not use DMA to do 1 or 2 pixel copies**. It defeats the purpose of using DMA!


VBA-M - 99%
ONLY
AT
TECH


VBA-M - 99%
We're no strangers to love
You know the rules and so do I
A full commitment's what I'm thinking of
You wouldn't get this from any other guy
I just wanna tell you how I'm feeling
Gotta make you understand
Never gonna give you up
Never gonna let you down
Never gonna run around and desert you
Never gonna make you cry
Never gonna say goodbye
Never gonna tell a lie and hurt you

Deliverables

Warning

Please double check over your submission after you submit it please download what you submit and check over it to ensure you have submitted the correct file. Failure to do so will result in an automatic zero (in the case of an empty file) on the timed lab.

1) Everything; PLEASE do not submit .o files. Run make clean and then zip up all of the files. Please do not zip up a directory and submit it.

You may submit only the files listed above. We will not accept any internet links, we want the files above and only these files! Check over your submission after you submit it. If you submit the wrong file and leave the lab, I will not be happy, and we will grade what you submit, so please check over what you submitted after you submit it! Check your email afterward to see if you get an email from T-Square. ***Note I have given zeroes to those who failed to submit the correct file, so please don't let this happen to you. I really don't like saying you get a zero.***

Have fun and good luck!