PRELIMINARY DRAFT DATED 2022-01-31 (marking scheme corrected)
PLEASE SEE LATEST VERSION ON CANVAS AS THIS ONE MAY CHANGE

CS-255 Computer Graphics Assignment (only 1 assignment, worth 20% of module)
Date set : 1/2/2022;
Deadline : Monday 28th February 2022 at 11:00
Viva booking: An Eventbrite link will be posted on Canvas closer to submission.

By submitting this coursework, electronically and/or hardcopy, you state that you fully understand and are complying with the university's policy on Academic Integrity and Academic Misconduct. The policy can be found at https://myuni.swansea.ac.uk/academic-life/academic-misconduct.

Guidance: **A lot of guidance for this assignment will be given in the lectures and support will be given in the Thursday assignment advisory classes**.

Unfair Practice: Do not copy code from colleagues, internet or other sources. You may discuss approaches together, but all coding must be your own. Presenting work other than your own at a viva is plagiarism and a recipe for disaster. The application which you demonstrate must be the code submitted to Canvas. To demonstrate code which is different to that submitted will count as Academic Misconduct.

Aims
Understand how an image is stored internally, and how to manipulate the image
Translate useful graphics algorithms into working code
Improve your programming skills through self-study and a challenging assignment
Understand that graphics can be useful to users (in this case within the medical context)
Work with a three-dimensional data set
Combine interaction with visual feedback
Practice presenting your work in a viva situation

Files:
The supporting framework is written in Java. You may build on this framework. If you wish to carry out the coursework in a different language you may do so, but there will be no provided framework. You will be required to demonstrate your working program on 3/3/2022 or 10/3/2022 using Zoom screen share. You will require several things to start the exercise:

1. A copy of the Java template – downloaded from Canvas. This demonstrates how to display and manipulate images, with functions that will help you with the exercise.
2. A data set to operate on – CThead.

[Note: *You should not redistribute this data set anywhere – you will not have permission to do that*]

Exercise:

1. Implement image resizing [40%]:

    a. Using the slider which gives values from 32 to 1024, update the large image so that it has the size indicated by the slider. Do this using Nearest Neighbour sampling. **[20 marks]**

    b. Also implement bilinear interpolation. In both cases, do not use Java provided libraries for image resizing. Do make your own functions that operate by sampling the CThead data (namely the provided “grey” array). **[20 marks]**

2. Implement a thumbnail view [40%]:

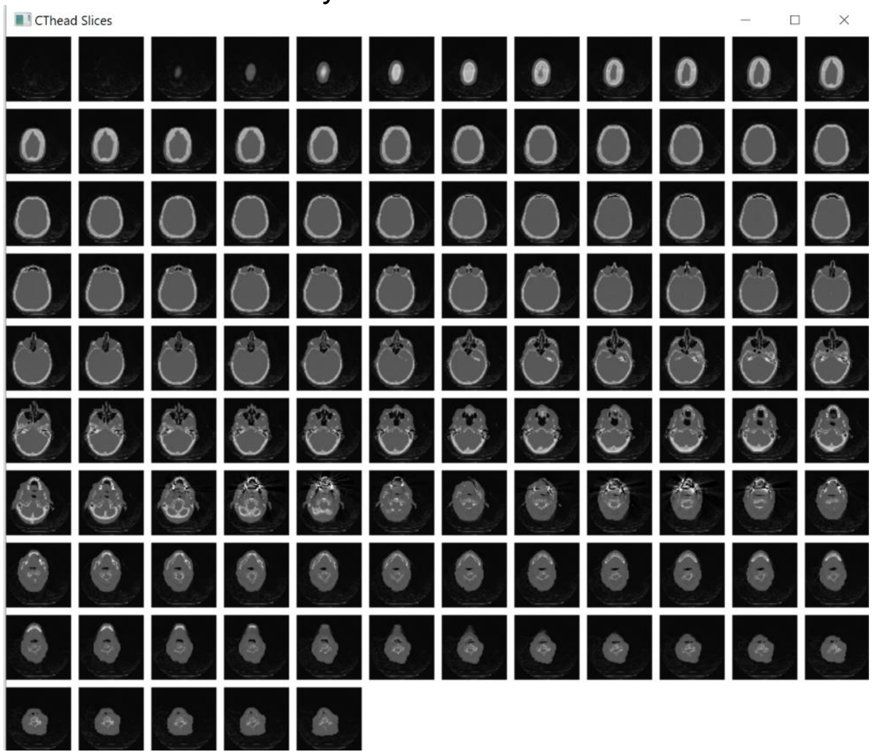
    a. Display a window with thumbnail images of all of the slices of the CThead using your own code – this is discussed later.

    [20 marks]

    b. As the user moves their mouse across the window, the large image view should update to show the slice underneath the mouse pointer.

    [20 marks]

A minimal solution may look like this.



3. Perform Gamma correction on the main image [20%]:

    You should implement gamma correction on the main image. The given slider allows values between 0.1 and 4.0 which is a reasonable range. Full marks are available if you use the look up table approach. Again, do not use any Java functions that do gamma correction. Do use your own function.

    [20 marks]

Submission Requirements:
You will demonstrate your working program via Zoom screenshare to me, or a post-graduate at times you will book using an Eventbrite link I will send week of the deadline or shortly after (these will be in the Thursday lab slots of 3rd March and 10th March). Submit your assignment through Canvas **by the deadline**. If you have several files, place them in a ZIP – **do not include the data set**).
The coursework is worth 20% of this module. There is only 1 coursework. It is marked at the viva.

Plagiarism:
It’s so important, I’m going to say it again. Each year a student will try to present code they don’t understand – don’t be that student. A simple question like “Why did you do that” or “What does that do” should not stump you.

Marking scheme

Note, if you cannot answer questions about your code (or have limited understanding of it), the marks will be reduced (sometimes down to zero).

1. **Resizing** [40 marks]
a. Does the image get bigger and smaller according to the slider without error? Has nearest neighbour sampling from the “grey” data set been used rather than some other function? [20 marks]
b. Does the image get bigger and smaller using bilinear interpolation? Has bilinear sampling from the grey data been implemented rather than using some other function? Is it well understood? [20 marks]
Understanding can be tested using questions like: How is this implemented, how are the other parts implemented, what does this bit of code do? Deduct appropriate marks from above if the student cannot describe their own code. **Just reading comments out is insufficient and marks should be deducted. All in code comments must be in English.**

2. **Thumbnails** [40 marks]
a. Does the window appear with all of the thumbnails? Has the student programmed the functions themselves rather than using Java image sizing library, or rather than adding lots of Java images/imageviews to a flowpane / scene? i.e., has a single image been created of the correct size based on the thumbnail size, number of slices, number of rows and desired gap size between images. [20 marks]
b. Does the mouse position over a slice change the main large image to be that slice? Is this a single event handler on the single large window image (that displays all the thumbnails)? Does that event handler include a function to go from mouse coordinates to slice via some computation? [20 marks]
If the student has used an implementation where individual images are added to a scene/flow and each image has its own event handler to update the large image, this can attract some marks (20), but not the full 40 marks.

3. **Gamma Correction** [20 marks]
Does changing the slider for gamma correction result in a dimmer or brighter image? (Slider to the right causes a brighter image)? Is a look up table used efficiently and correctly? (10 marks if no look up table is used). Is it the students own code and not a Java library call (0 marks if a Java function)?

Submission Procedure:
Submit the assignment through Canvas before the deadline. Demonstrate/viva your assignment after the deadline at times to be notified.

The college policy for late submission will be used. The timestamp from Canvas will be used. I will email students at their University account, so you must read this frequently during the term. You might not be able to demonstrate/viva if you submit late.

If you have extenuating circumstances (documentation must be provided), we will not have a problem with making alternative arrangements.

FAQ
I’ve submitted the wrong version.
You can submit multiple times – I will mark the last version (submitted before the deadline).

Feedback?
Feedback is provided at the demonstration/viva and very shortly **after all vivas are complete** in an email to your University number mail account.

Assignment Hints

Resize

The resize function creates an image of the correct size (obtained from the slider). It loops over every pixel and calculates where the CThead should be sampled. It calls a function with the double precision sample coordinates to get that sample. Until I give the lecture this year, this is covered in the 2021 lecture 18c. Background material is given first in the video, and specifically this paragraph refers to around 8 minutes into the video.

For sampling, I wrote a function which takes as input the double precision (x,y) coordinate that we want to sample from a slice from the head. 0<=x,y<=255. It returns the grey value (0 to 1) that can be used as the pixel r,g,b channels to give a greyscale pixel.

To implement **nearest neighbour** sampling, I just needed to return the grey value based on the integer rounded values of x and y.

To implement **bilinear interpolation**, I needed to use linear interpolation three times as indicated in the course notes. Therefore, I wrote a linear interpolation function (lerp) that takes as input two values, their two positions, and the floating point position where we want to find the pixel value. This function implements the equation which (until I give the lecture this year) is in lecture 18c at 10 minutes into the 2021 lecture. I then wrote a bilinear interpolation function which has as input the 4 corner values, and the floating point position to sample. This calls lerp three times and returns the required sample.

This is my comment from my code with diagram to help:

```
	// v2 (px0,py1) +------+ v3 (px1,py1)
	//              |    X | v, X=(px,py)
	//              |      |
	// v0 (px0,py0) +------+ v1 (px1,py0)
	// v are the values, and px0 and px1 are the x coordinates, and py0 and py1 are 
the y coordinates
	// but since only the offset is important, we can assume px0=0, px1=1, py0=0, 
py1=1 and X=px,py is the fractional offset from px0,py0
```

(The above is a big hint – perhaps making this possibly too easy, so I wouldn't go further than this to help).

I did the resize function first, because the thumbnail part would reuse some of the code to make the images smaller.

Thumbnails

Here, I specifically want you to create a single large image in which you place all the thumbnails using your own sampling function and creating your own event handler to convert the larger image coordinates into the correct slice. This approach gets the full marks. An alternative approach (which will get only half marks) is to rely on Java libraries by creating lots of images (one for every slice) and adding them to, for example, a FlowPane, and also adding an event handler to every image so you know which image is "moused over". This latter approach requires less effort and removes the need for you to deal with pixels and pixel positions at a low level, and therefore attracts fewer marks.

I approached the problem by working out a few variables I need, which included, what size I wanted the thumbnails to be, what size the gap is to be around each thumbnail, and how many thumbnails are to be on each row.

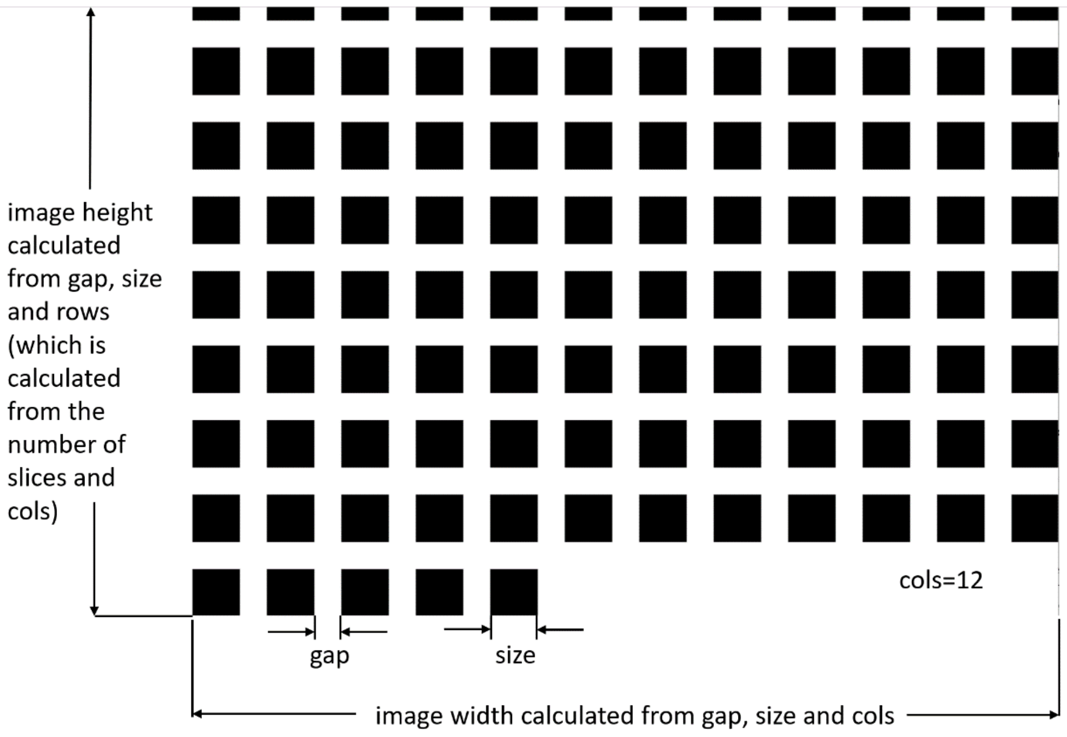
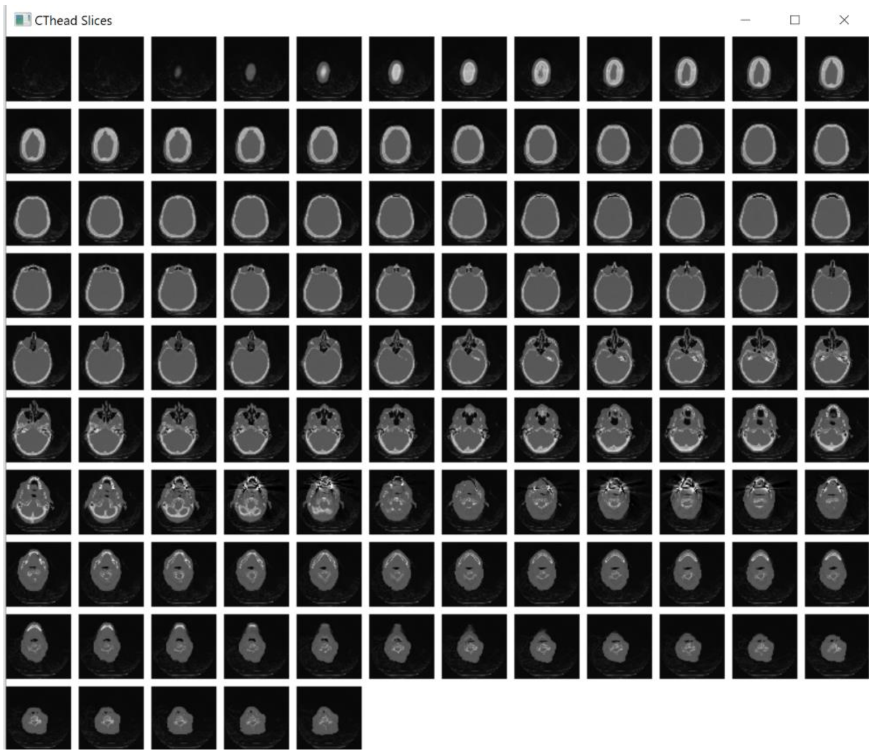


image height calculated from gap, size and rows (which is calculated from the number of slices and cols)
cols=12
gap
size
image width calculated from gap, size and cols

I created the image of the correct size. I looped through every pixel working out if it was in a gap (white) or a slice (black). This produced the debug image you can see above. I worked out for each pixel which slice it would be in, and its offset from the origin of that slice, and called my sampling function (from resize) to get the grey sample. This gave the thumbnails view. There are other ways, but this gives you one way of how to progress it.



For the mouse over event, I took the coordinates of the mouse and worked out the slice number. If you've worked out the image width and height from cols, gap and thumbnail size, then this is a similar process (in reverse). The last row could create an error and requires an extra check.

This is more involved than using Java libraries to add an image for each slice with its associated event handler. Therefore, this approach gets more marks.

Gamma Correction

I did Gamma Correction last. The grey value you return for resize can be gamma corrected using the gamma correction function. By observation, on a 1024 by 1024 image you would carry out gamma correction over 1 million times (which involves a floating point power function). Most displays and graphics drivers work with 256 different grey / colour levels. Therefore, you can create a 256 entry look up table which maps a value to its new gamma corrected value. Using the look up table gets the full marks on this part.

Changing the framework

Yes, you can change the framework. You can change the interface (hopefully to improve it). You can use IDE's to build an interface, so long as the key elements in the marking scheme are coded by you. i.e., you can style the sliders and reposition them in the interface (but make sure they still cover the range of values), but you need to write the resize functions as in the marking scheme (for example). You can use a filebrowser to open the data set (or a different data set). And so on. But none of this gets any marks in the above marking scheme. Many students do this because they want their software to look nicer and because they enjoy programming the assignment.