Coursework 4 submission guide, v1

Because we need to mark these courseworks from submission rather than demos this year, we will have some issues ensuring that you get all of the marks you deserve – particularly because the coursework is deliberately designed so that there are multiple ways to do this.

Changes from draft: combined two sections talking about documentation. Added appendix with a template you can use to avoid everyone creating their own. Moved text of what to remember to include in sections into the template in Appendix 2. Clarifications for what not to include in video were added. Hopefully it is clearer now. Thank you to those who fed back.

Note: If anyone finds errors I will release v2 and tell you the changes. It will only be corrections for errors though.

Summary information

Please submit all files to the same moodle submission entry, with the specified filenames.

All work must be your own work, as mentioned in the requirements.

As well as submitting your code, you will need to submit a short demo video of everything in action (without explanation), a document explaining how you met each requirement (referring to matching sections of the video for demonstration of each requirement, and to where in the code you do it, so we can see it). Specific details for each are below.

I am deliberately avoiding you needing to put voice-over on the video, because I know some people dislike that, but you can do if you wish. Please do not assume that we hear everything you say on the video.

Marking approach – so you can ensure you get all of the marks you deserve!

We will download, unzip and compile the code using the method mentioned below (in the zip file section), so you can test this yourself to avoid unexpected issues.

We will then work from your documentation file, and will look at the parts of the video that your documentation file prompts us to do so, running your program to check things when that is useful.

Your video will ideally NOT contain code or explanation (the documentation gives the explanation and points us to where to look in the code). Instead the video illustrates the points. Your document says that it does something and how, it points to the code where you do it (via the table), and tells us where in the video to see it happening (via the table).

Think of the video as your live demonstration (without the explanation, just showing us) and the documentation as your explanation and justification to go along with it. You can think of the documentation as some kind of commentary of the video if that helps, which may help you to keep both aligned , so the video illustrates the points in the document.

Assume that we can: pause your video, e.g. to read the documentation; skip sections, e.g. when we are happy you met a requirement to skip to the next one; go back and check something if we need to; check your code ourselves, so we do not need your video to show the code – just the running program. Feel free to highlight in the video when you change requirement you are explaining, just to make it easier to identify, but that is not necessary if it is a problem.

Submission overview

You will submit three things: a zip file of your project directory, including code and resources; a demo video; a documentation file.

A. Zip file of the code of your project

Please ensure that your project will build on windows – using the citrix flexible desktop is the easiest way to test it. Since with lockdown and remote working, we can only guarantee access to windows (through Citrix if necessary).

Please ensure that it works on Visual Studio 2017 (as on Citrix).

(If in doubt, download the original zip file from the web page, copy all of your source files into the src directory, open the project, add the files to the project ('add existing file') and build it. Then test it!)

Please use 'clean solution' (build menu) on your project before submission, to delete temporary files.

Please delete the 'map' file from the src directory. (It is a debug file and can get big.)

Please delete the .vs hidden directory, since it can be huge. It is usually in the main directory, the directory with the src directory in it.

Please then zip the project directory KEEPING relative file paths.

Please test your project as follows, since this is the approach we will use:

- Move the old copy of your project, so that any hard-coded path references to it don't work.
- Unzip the zip file to a NEW directory in a different place to the old directory, keeping directories within the zip
- Open the .sln file in visual studio 2017 (usually just double click it)
- Select rebuild solution from the build menu.
- Select to run it, from the debug menu.
- Test that it works, that images are visible, etc.

Please ensure that any extra SDL header files, lib files and dlls that you used (e.g. sdl_mixer if you needed it) are in the appropriate directories. i.e. the same as the files I provided.

This is all important. I remember in the past we had issues with:

> Some source code files missing
>
> Some resource files missing
>
> Hard coded paths to resource files, so when we tried to run it from a different directory it did not work.
>
> If your code does not compile and run so we can test it, you will not get those marks, so your mark will be severely reduced.

B. Documentation file (see appendices for example information & template!)

It is important that your documentation is clear so that you do not lose any marks through not making your work clear. The documentation should be short and clear, not verbose.

The aim of the structure is to help you to clearly explain things so we don't miss anything.

Your document will have 5 sections, with the last section being the longest by far.

Section 1: introduction.

Please just summarise in a few words what your program involves. Keep it to one paragraph of up to 5 lines maximum.

E.g.: "My program is a simple pong game, with a few enhancements such as a scrolling background which moves according to the ball direction and some animated …"

Section 2: Compilation and running.

Please explain any special considerations. brief summary of any compilation considerations and issues.

E.g. 1: "Nothing special to consider."

E.g. 2: "Although it looks like there is a memory leak, it is a case of the string in line x of file y.cpp being a static which is not cleared until after the memory leak detection code has executed."

E.g. 3: "I know about an issue where it will crash if you press a key while it is paused but I didn't manage to fix it in time." (Note that at least then we know that you are aware of the issue and it's not just a case of lack of testing. It also helps us to avoid accidentally crashing the program when we run it. Which can be annoying if we don't know why it crashes.)

Section 3: File list.

This is just a list of the files in your program. Sometimes it can be really hard for us to find the files which do various things to understand how your code works. In theory your requirements documentation for each requirement should be sufficient. In practice, it has not been at times in the past. You only need a few summary words for each and can group the .h and .cpp files.

Example table, for illustration:

Code .h/.cpp Filename	**What this file is for, e.g. what class is in it and what it does**
Pong.h and .cpp	My main BaseEngine subclass
Player.h and .cpp	DisplayableObject for the player
AIPlayer.h and .cpp	Displayable object for AI player
Ball.h and .cpp	Displayable object for the Ball. The custom collision detection class is also in the .cpp file and is only accessed from here.

Section 4: Resource file list

This should be a brief list of the various resources files we should expect to see and what they are for. One reason for this is in case for some reason you put them in the wrong directory or there is some path issue when we test it – it helps us identify the problem. It also means if, for example, you were using "/home/psstudent/myfile.jpg" in your program and there is no myfile.jpg in your submitted zip then we know roughly what you did wrong. Hopefully the instructions

E.g. "I use the following images: jason.png, background.jpg, bat.jp and ball.jpg. All are in the src directory." *You probably will not need to say more than something like this.*

Note: this should also prompt you to realise if you forgot to add any files to the submission zip. E.g. if you put them in odd directories. (See my notes on testing your zip file.)

Section 5: Requirements information

In this section you should go through each requirement, telling us what to do. Please see Appendix 1 for an example of how to fill in the information for one example. Please see Appendix 2 for a template to fill in for the documentation.

Important: Please ensure that you list the requirements in the order that they appear in the demo, so that we can go through and mark them one at a time while watching the video. In some cases that is not necessarily possible, in which case please put it in order of the first time it is mentioned in the video. (So we can note that, and come back to the other bits as we get to the later parts of the video.)

For each requirement you need to provide both the following table, to make it clear where to look in the video and code, and an explanation of the key points, how you implemented it, what mark you expect, why, and any requirement-specific information listed below. Please name the section according to the requirement number.

Item	Video start time	Video end time	Source files and line numbers

I have provided an example of a requirement explanation on the last page of this document. Please refer to it to get an idea of what sort of information to provide.

Some requirements will need specific extra information. To ensure you don't miss this, I summarise this below on a per-requirement basis. **I also suggest combining the descriptions for some requirements.**

Please ensure that your documentation includes the following key information for these requirements, **as well as the general information, such as the table, the expected mark and justification and a brief summary of what your code does**.

Please see Appendix A for an example of the amount of detail to add for a requirement.
Please see Appendix B for an outline of the full documentation file and information about what to add in each section. (I moved that information into the example so you can use it as a template and just add text if you wish, saving you time.) I suggest to take Appendix B as your initial document and leave the green text in until as late as possible. It is not a problem if you leave it in for the submission.

C. Video

You should submit one or more videos with a total length no more than 15 minutes long, **maximum,** and **ideally no more than 10 minutes in length**, which demo your various features.

Remember that we will watch this in combination with the documentation, so we may skip around it according to what the documentation tells us. You do not need to (and should not) put explanations in the video – put them in the documentation instead. Similarly, you do not need to put code samples in the video – instead the documentation should tell us which files and line numbers to look at. Think of the video as 'now see it running for evidence of it actually working as the documentation says it does'.

Your video should start with a brief demo/run through of the program, before potentially going into detail for the requirements, giving the visual support for the documentation. Doing this will make the program look better and give a better impression. However please make this relatively short and it may be that some requirements will be obvious already just from watching the program run and a few quick checks of code, so feel free to highlight those and put them first in your requirements section. Please remember that this is not a promotional video for a great new game – it is instead purely support and evidence for your documentation.

Your video should demonstrate each of the requirements, in an order which matches your documentation file (you will specify the details in your documentation file).

It may be that where two requirements are closely linked, you show them in one part of the video and just mention this in the requirements. Please highlight it in the first requirement if you do so, so that we know to be thinking about both requirements at the same time while watching it. E.g. 'This section of video also covers requirements D and F below'.

I am asking you to submit the video in the same format as was requested for the group project since you should be able to do that already:

"The demonstration video should be recorded using a screen capture tool and must be encoded and submitted as ~~1280x720~~ H264 video in an MP4 wrapper. This is a standard format that can be generated from most video software on all platforms. Videos submitted in other formats will be rejected."

I deleted the size, since you may want to go with a larger or smaller size depending upon the size of the window you are capturing. Please be aware that we may not all have access to large screen sizes when watching though. I suggest that your mark consideration on width and height of the video should be making it clear to us when we watch.

If you have problems with video format, Handbrake has been suggested to me as a free software which can convert the video format for you.

Appendix A: Example of documentation for requirement 2:

Requirement 2. Save and load some non-trivial data

I have filled in an example for the saving/loading requirement 2, giving an idea of level of detail:

Item	Video start time	Video end time	Source files and line numbers
Loading map	0:32 1:52		Pong.cpp line 12, loadMap(). The map is loaded. The map is reloaded, using same function as above
Saving state	0:52	1:10	Pong.cpp line 312, saveState() method Pong.cpp line 112, handle key press to call saveState() Ball.cpp line 12, saveMyself() saves this object etc
Loading state	2:32	2:51	Pong.cpp line 381 loadState() method Pong.cpp line 131, user chooses to load rather than restart, calling loadState() Ball.cpp line 31, loadMyself() reloads the state from the file etc

I believe that I deserve three marks for this. I load a complex level map structure, and all of the changes to the map are stored in the save file. Also I save the state of all of the objects, which includes the objects which are created when needed, such as extra balls which bounce around. The file format is fairly complex as you can see in savegame.txt.

My save/load saves the state of all objects to a file called … . When a user presses s in the running state it will take a snapshot of the state of each object and save them to a file, by opening a file and asking each object in turn to write its information to the file.

From the starting menu, a user can choose to load the saved state rather than start a new game. In that case each object is created then calls loadState() to load its state from the file.

Note: there is not a lot of information here, but I provide enough information to ensure that I get all of the marks. I do not repeat information from the table in the text below. The text tells the marker what mark I expect and why. I then explain how the features work. The marker can look at the lines I mention in the table to see where I implement it, and at the section of the file to see where to see it working.

I chose this example because it is a bit complex in that it will probably be in multiple places in the video.

Note the separate rows for loading map, loading state and saving state, to highlight to the marker that there are three distinct things happening here, so that he/she can't miss it.

Appendix B, Example structure of documentation file

This appendix is designed to be a template where you can fill in the missing information, avoiding everyone having to make their own. You may delete the GREEN text once you read it.

1. Introduction:

Include a brief summary of what your program is about. You can include a screenshot if you wish, but it is not necessary since we have the video. See main requirements.

2. Compilation and running:

Give a brief summary of any compilation or execution considerations and issues that you know about. See main requirements.

3. File list:

Code .h/.cpp Filename	What this file is for, e.g. what class is in it and what it does

Fill in a list of the files you added to the project and what they are for. This can help us find things.

4. Resource file list:

Include a brief list of the various resources files we should expect to see and what they are for.

E.g. "I use the following images: <insert image list>, which are all in the src directory. "

5. Requirements

This is REALLY important and is the key to us marking your coursework and giving you all of the marks you deserve. List the requirements in the table in the order that you demo them in the video, so that we can mark them in that order. Assume that we will work through your documentation in order – regardless of the order in the video, so it will be clearer if you make the video order match this documentation.

Note: Extra information then appears for each of these requirements under a heading below.

Requirement 1: Add states to your program

Item	*Video start time*	*Video end time*	*Source files and line numbers*

Ensure that your explanation lists the states, how to get between them.

If you do the state model version please explain how you implemented the state model, including how you change state and where you create and destroy any objects.

Requirement 2: Save and load some non-trivial data

Item	*Video start time*	*Video end time*	*Source files and line numbers*

Ensure you mention everything that you load/save. You may want to clarify what the complexity is and where the marker can look for an example. Ensure that it is obvious how this is done if multiple classes are involved. Note: JigsawDemo is not a good example because I deliberately made it simple.

Requirement 3. Interesting and impressive automated objects

Item	*Video start time*	*Video end time*	*Source files and line numbers*

Why are your automated objects interesting and impressive? The video will be more useful probably for demonstrating complex behaviour than words, so you can mention specific times where it does specific things. Also ensure that you explain clearly how they do them.

Mention which object uses and image and where.

Consider explaining requirements D or F around the same time.

Requirement 4. Impact/Impression (and requirement L: Sellable quality)

Item	*Video start time*	*Video end time*	*Source files and line numbers*

This is why it is useful to have a runthrough at the start of your video – it's your chance to make a positive impression first. Consider making this the first requirement you explain, for this purpose. There is a saying that 'first impressions last'.

You can mention here other requirements which you met which you think make it especially impressive. E.g. I spent a lot of time making it look really nice, but it is also a really tricky program to play against. In requirement F I made a really clever AI which learns how to beat you, and I added some really useful sound effects (requirement I) at various times.

You do NOT need to repeat anything said in other requirements – just point the marker to them.

You should be thinking to only summarise key points, not to write a huge essay about why this will revolutionise the gaming industry.

Note: we may also be thinking about requirement L when we read this, so you could make it a really good argument here and highlight anything you think makes this especially wonderful and sellable. Note my comment about requirement L is probably not one to explicitly aim for, but you could use this opportunity to highlight things like number of levels, features, configuration, appearance, etc, by referring to other requirements and mention to us that you are hoping to get requirement L as well.

You should not add a separate requirement L section, so please make the argument here if applicable.

Requirement A: Correctly implement scrolling and zooming using the framework's FilterPoints class

Item	*Video start time*	*Video end time*	*Source files and line numbers*

Just explain how you did it. The lines in the table should tell us where to look in the code.

Requirement B: Have advanced animation for background and moving objects

Item	*Video start time*	*Video end time*	*Source files and line numbers*

If you had to do anything special to ensure it is smooth, please mention it. Note the 'impressive' comment on the 2 mark criteria. Something simple and obvious is unlikely to be impressive.

Requirement C: Interesting and impressive tile manager usage

Item	*Video start time*	*Video end time*	*Source files and line numbers*

Explain why you think it is impressive. What do you want to draw the marker's attention to, to ensure that they do not miss it?

Requirement D. Creating new displayable objects during the game

Item	*Video start time*	*Video end time*	*Source files and line numbers*

Are the objects there all of the time and just invisible, or are they created as needed? How do you create them? When are they destroyed? Are there load/save issues? Consider explaining this close to your explanation of requirement 3.

Requirement E. Allow user to enter text which appears on the graphical display

Item	*Video start time*	*Video end time*	*Source files and line numbers*

Remember the need to handle delete/backspace!

Did you need to add any extra handling for special cases or overflows?

Requirement F. Complex intelligence on an automated moving object

Item	*Video start time*	*Video end time*	*Source files and line numbers*

Explain the basic ideas and ensure that we know where in the code to look for the implementation, how it works, and where in the demo video to see the behaviour.

Please see the notes on requirement 3, and consider explaining this straight after requirement 3 explanation, since it may be easier for you to explain as well as for us to mark.

Requirement G. Non-trivial pixel-perfect collision detection

Item	*Video start time*	*Video end time*	*Source files and line numbers*

Basic rectangle or circular collision detection was in CW3. You need to have something that is more complex than that, so explain why it is non-trivial. Most people seem to be checking pixels or bitmasks for this, so explain how you do it and how you make sure it is fast enough if you need to.

Requirement H. Image rotation/manipulation using the CoordinateMapping object

Item	*Video start time*	*Video end time*	*Source files and line numbers*

Ensure that the explanation makes it clear that you understand the principles behind this – how does it work – rather than just referencing code which does it. In theory some people could modify the code without really understanding it.

Requirement I. Integrate sound using SDL

Item	*Video start time*	*Video end time*	*Source files and line numbers*

What functions or SDL libraries did you use? Did you have any problems (I admit I had some issues at first with a dodgy function). Where is the sound happening? (background music, sound effects?)

Requirement J. Show your understanding of templates and/or operator overloading

Item	*Video start time*	*Video end time*	*Source files and line numbers*

This is quite open but please not the comment about appropriate manner and ensure that your explanation shows why what you did was appropriate in the situation.

Requirement K. Use your own smart pointers appropriately

Item	*Video start time*	*Video end time*	*Source files and line numbers*

This will be hard because I leave this lecture until late in the module, so you will have had to look things up yourself. Please show your level of understanding about what these are and how you used them. Please be clear about how you implement them or what standard classes you use. My main criteria are your proven level of understanding and interesting appropriate use. i.e. not just for the sake of it.

Your table may want to have separate rows for the classes, where you create objects, where you use them, etc.

Requirement M. An advanced feature I didn't think of but you had pre-approved

Item	*Video start time*	*Video end time*	*Source files and line numbers*

I am not sure that I approved any explicitly yet. Please do not leave it until the last minute to ask me! Where things were approved, please reference any email discussion we had as evidence (i.e. email date & time). In general I suggested for people to use this only if they did something anyway, rather than aiming at it.

There is no need to add requirement L. Sellable quality!

Please see requirement 4. You should NOT add a requirement L section, and should instead make the argument in Requirement 4.