PSYBW7 – C++ CW2

Contents

[How to run 4](#_Toc133461760)

[Handling of program states 5](#_Toc133461761)

[a. a start-up state, a pause state and a running state, 5](#_Toc133461762)

[b. which differ in some way in both the appearance and behaviour 5](#_Toc133461763)

[c. at least five states d. including at least one win or lose state as well as those 5](#_Toc133461764)

[d. mentioned above. (Note: if you are not doing a game, then have a state which allows a reset, e.g. ‘load new document’ in a text editor.) 5](#_Toc133461765)

[e. There must be significant differences between the states in behaviour and/or appearance. The program must be able to correctly go back from the win/lost (or reset) state to the starting state to start a new game/document, correctly initialising any data. 5](#_Toc133461766)

[f. implement the state model (design pattern) using subtype polymorphism rather than if/switch statements in each function. Note: You may still need some if statements etc to enter the states, but the current state object is used to determine how to act in each function 5](#_Toc133461767)

[Input/output features 5](#_Toc133461768)

[a . save AND load at least one value to/from a file, e.g. a high score to a file (e.g. a single number to a file) 5](#_Toc133461769)

[b. It must be multiple read/write operations, of multiple values with some kind of structure to the file, rather than just a string of 3 numbers, for example. 5](#_Toc133461770)

[c. Something must visibly change depending upon what is loaded (e.g. set tiles according to data read and/or set positions of moving objects according to the data). 5](#_Toc133461771)

[d. A user should be able to save where they are (e.g. the current document that they are working on or the state of a game – saving all relevant variables for all objects) 6](#_Toc133461772)

[e. and they should be able to reload this state later, with everything acting correctly and continuing from where it was. 6](#_Toc133461773)

[f. Note: this means saving/reloading the positions/states of all moving objects as well as anything like changeable tiles, etc. g. You will need to provide some way to reload from the state as well – e.g. when the program starts (e.g. choosing to load or start a new game) or in response to some command from the user (e.g. pressing S for save and L for load?). 6](#_Toc133461774)

[Displayable object features 6](#_Toc133461775)

[Multiple displayable objects 6](#_Toc133461776)

[Create and destroy displayable objects during operation of a state 6](#_Toc133461777)

[Dynamically add one or more displayable objects to the program temporarily after a state has started (i.e. you add to the object list or make some object visible) 6](#_Toc133461778)

[when making the object appear during the running of one state, actually create the object and add it to the DisplayableObjectContainer (rather than just making it visible), 6](#_Toc133461779)

[when making the object disappear, within the running of the state, remove it from the object list in the DisplayableObjectContainer. 7](#_Toc133461780)

[notify the program when you change the contents of the DisplayableObjectContainer. If you change the object array you need to tell anything using it that you did so, hence why this is in as a separate requirement – it can be trickier to get right. Please see the drawableObjectsChanged() method and investigate how it works to get this right. 7](#_Toc133461781)

[correctly delete the object at that time (not at some later point, such as when the state changes). 7](#_Toc133461782)

[Complex intelligence on an automated moving object 7](#_Toc133461783)

[Path finding 7](#_Toc133461784)

[Can’t find player. 7](#_Toc133461785)

[Collision detection 8](#_Toc133461786)

[Pixel perfect 8](#_Toc133461787)

[Tile collision 8](#_Toc133461788)

[Animations, foreground scrolling and zooming 8](#_Toc133461789)

[Implement a scrolling background by manipulating the way that the background image is drawn 8](#_Toc133461790)

[Have an animated or changing background by utilising multiple images 8](#_Toc133461791)

[a. Have at least five drawing surfaces, set up with at least slightly different contents 8](#_Toc133461792)

[b. switch the appropriate surface in to be used as the background (noting that m\_pBackgroundSurface is accessible to subclasses and can be changed, deliberately for this purpose). 8](#_Toc133461793)

[c. Make it appear as if the background has been animated (at least slightly) using this process 8](#_Toc133461794)

[Animate moving objects 9](#_Toc133461795)

[a. show understanding of how to animate at least one object so that its appearance changes over time, however this could be a proof of concept instead of being smooth 9](#_Toc133461796)

[b. your objects should have animated rather than fixed appearances 9](#_Toc133461797)

[c. the animation should be smooth 9](#_Toc133461798)

[d. and visually impressive. Please do note the ‘impressive’. So please don’t just rotate objects using facilities from a demo as that would not get you the marks. 9](#_Toc133461799)

[e. Be able to explain in the demo what you have done and how it works. 9](#_Toc133461800)

[Tile manager usage 9](#_Toc133461801)

[Other features 9](#_Toc133461802)

[Allow user to enter text which appears on the graphical display 9](#_Toc133461803)

[• Capture the key presses for letters/characters. 10](#_Toc133461804)

[• Store the key presses somewhere 10](#_Toc133461805)

[• Capture delete key press and handle it appropriately 10](#_Toc133461806)

[• Display the text on the screen 10](#_Toc133461807)

[Show your understanding of templates, operator overloading OR smart pointers 10](#_Toc133461808)

[Additional complexity 10](#_Toc133461809)

[Ternary 10](#_Toc133461810)

[Overall impact/impression 10](#_Toc133461811)

[• you made some effort with the graphical appearance, 10](#_Toc133461812)

[• the background is relevant and not plain or the same as any of the demos, 10](#_Toc133461813)

[• the background includes at least some use of relevant shapes (e.g. separating off a score by putting it in a box and labelling it) and/or images, 10](#_Toc133461814)

[• moving objects are not just plain circles or squares. 10](#_Toc133461815)

[• have at least three moving objects 10](#_Toc133461816)

[• at least one is user controlled 10](#_Toc133461817)

[• you accept both mouse and keyboard input in some way 10](#_Toc133461818)

[• you use both images and drawing primitives appropriately 10](#_Toc133461819)

[• you have appropriate states (at least 2) 10](#_Toc133461820)

[• have something beyond the minimum required for the other marks (please say what) 11](#_Toc133461821)

[• your program should work smoothly 11](#_Toc133461822)

[• everything looks good with no problems 11](#_Toc133461823)

# How to run

This coursework, unlike the previous, is intended for windows platforms, this is to avoid any memory leak ambiguities linked with SDL2 + Linux. To run this program, you need to

Compiling

IGNORE THESE STEPS IF YOU WANT TO RUN THE PRECOMPILED BINARY STRAIGHT AWAY

1. Open the solution provide “CourseworkBase2021\_vs2019.sln”
2. Click “Local windows debugger” – Note: use Debug to see debugging information and release for no console window.
3. The program will compile and run.

Running

Double click the **“CourseworkBase2020.exe”** file. If this doesn’t seem to launch, its because the **resources** didn’t get copied over during compilation. To fix this, simply copy the **resources** folder from the **“src”** folder and paste it into the same directory as the **“CourseworkBase2020.exe”** file.

# Handling of program states

## a start-up state, a pause state and a running state,

## which differ in some way in both the appearance and behaviour

## at least five states d. including at least one win or lose state as well as those

## mentioned above. (Note: if you are not doing a game, then have a state which allows a reset, e.g. ‘load new document’ in a text editor.)

## There must be significant differences between the states in behaviour and/or appearance. The program must be able to correctly go back from the win/lost (or reset) state to the starting state to start a new game/document, correctly initialising any data.

## implement the state model (design pattern) using subtype polymorphism rather than if/switch statements in each function. Note: You may still need some if statements etc to enter the states, but the current state object is used to determine how to act in each function

# Input/output features

## a . save AND load at least one value to/from a file, e.g. a high score to a file (e.g. a single number to a file)

## b. It must be multiple read/write operations, of multiple values with some kind of structure to the file, rather than just a string of 3 numbers, for example.

## c. Something must visibly change depending upon what is loaded (e.g. set tiles according to data read and/or set positions of moving objects according to the data).

## d. A user should be able to save where they are (e.g. the current document that they are working on or the state of a game – saving all relevant variables for all objects)

## e. and they should be able to reload this state later, with everything acting correctly and continuing from where it was.

## f. Note: this means saving/reloading the positions/states of all moving objects as well as anything like changeable tiles, etc. g. You will need to provide some way to reload from the state as well – e.g. when the program starts (e.g. choosing to load or start a new game) or in response to some command from the user (e.g. pressing S for save and L for load?).

# Displayable object features

## Multiple displayable objects

Multiple displayable objects from at least three displayable object classes, with different appearances and behaviour from each other and you should have an intermediate class of your own between the framework class and your end class, the intermediate class adds some non-trivial behaviour.

## Create and destroy displayable objects during operation of a state

## Dynamically add one or more displayable objects to the program temporarily after a state has started (i.e. you add to the object list or make some object visible)

ensure that the object cannot interact with anything else before it is added or after it is ‘destroyed’. E.g. if you make the object invisible then it is not collision detecting with anything while it is not visible. You must not just re-create the entire object array contents to do this, although you could add to the end of it. (Please see the methods on DisplayableObjectContainer.) This requirement is not met by the re-creation of objects when you change states. The idea behind this requirement is that moving objects appear and disappear while using the program within the state.

### when making the object appear during the running of one state, actually create the object and add it to the DisplayableObjectContainer (rather than just making it visible),

### when making the object disappear, within the running of the state, remove it from the object list in the DisplayableObjectContainer.

### notify the program when you change the contents of the DisplayableObjectContainer. If you change the object array you need to tell anything using it that you did so, hence why this is in as a separate requirement – it can be trickier to get right. Please see the drawableObjectsChanged() method and investigate how it works to get this right.

A picture containing text

Description automatically generated

### correctly delete the object at that time (not at some later point, such as when the state changes).

Making sure that you destroy/delete the object at the right time is important for this – it is trying to get you to understand how to get objects to destroy themselves safely without causing issues with using the pointer after the object was destroyed. This is an important thing to understand in C++.

## Complex intelligence on an automated moving object

### Path finding

### Can’t find player.

# Collision detection

## Pixel perfect

Talk about rect to rect then to pixel itself

## Tile collision

Talk about the canMove function on a high level

# Animations, foreground scrolling and zooming

## Implement a scrolling background by manipulating the way that the background image is drawn

## Have an animated or changing background by utilising multiple images

### Have at least five drawing surfaces, set up with at least slightly different contents

### switch the appropriate surface in to be used as the background (noting that m\_pBackgroundSurface is accessible to subclasses and can be changed, deliberately for this purpose).

### Make it appear as if the background has been animated (at least slightly) using this process

## Animate moving objects

### show understanding of how to animate at least one object so that its appearance changes over time, however this could be a proof of concept instead of being smooth

### your objects should have animated rather than fixed appearances

### the animation should be smooth

### and visually impressive. Please do note the ‘impressive’. So please don’t just rotate objects using facilities from a demo as that would not get you the marks.

### Be able to explain in the demo what you have done and how it works.

# Tile manager usage

To get this mark you must be using a tile manager, and must have multiple displayable objects. - Your tile manager must draw a number of appropriate and different pictures (either using images or the drawing primitives) for the different tile types, which are not just different colours. - You should have at least 5 different tile types (not just different sizes ovals/rectangles). - At least one tile must be drawn using an image. You should load the image only once and keep it in a relevant object, NOT keep reloading it each frame if you want these marks. Note: If you use multiple images then you could also potentially meet the animated appearance of automated objects criterion, and/or the animated background if you do it appropriately.

Collision with tiles (just link to collision detection)

# Other features

## Allow user to enter text which appears on the graphical display

### • Capture the key presses for letters/characters.

### • Store the key presses somewhere

### • Capture delete key press and handle it appropriately

### • Display the text on the screen

## Show your understanding of templates, operator overloading OR smart pointers

## Additional complexity

### Ternary

# Overall impact/impression

As a minimum this means:

## • you made some effort with the graphical appearance,

## • the background is relevant and not plain or the same as any of the demos,

## • the background includes at least some use of relevant shapes (e.g. separating off a score by putting it in a box and labelling it) and/or images,

## • moving objects are not just plain circles or squares.

## • have at least three moving objects

## • at least one is user controlled

## • you accept both mouse and keyboard input in some way

## • you use both images and drawing primitives appropriately

## • you have appropriate states (at least 2)

## • have something beyond the minimum required for the other marks (please say what)

## • your program should work smoothly

## • everything looks good with no problems