Brandon Wright

[Company name]  [Company address]

Coursework 1 documentation

Table of Contents

[Create an appropriate sub-class of BaseEngine with an appropriate background which is different from the demos 3](#_Toc129961187)

[Create an appropriate new sub-class of the base engine. 3](#_Toc129961188)

[Code snippet (1-5): 3](#_Toc129961189)

[Give it an appropriate background and ensure that the background is different to the demos 3](#_Toc129961190)

[Name your class using your capitalised username followed by the text Engine. e.g. if your username was psxabc then your class would be called PsxabcEngine. 4](#_Toc129961191)

[You MUST create a new class – you must not just change/rename one of the existing demo classes. 4](#_Toc129961192)

[Show your ability to use the drawing functions: 4](#_Toc129961193)

[Draw your background ensuring that you use at least one of the shape drawing functions to draw on the background 4](#_Toc129961194)

[and that you draw at least one image to the background, 5](#_Toc129961195)

[which is different from the demos and shows your understanding. 5](#_Toc129961196)

[Be prepared to explain what you have done to the marker if asked. 5](#_Toc129961197)

[Draw some text on the background 5](#_Toc129961198)

[Draw some text onto the background (not foreground) and ensure that the text is not obliterated when moving objects move over it 5](#_Toc129961199)

[Have some changing text, refreshing/redrawing appropriately which is drawn to the foreground (not background), in front of moving objects 5](#_Toc129961200)

[This text may change over time (e.g. showing the current time, or a counter) or could show a score which changes, for example. 5](#_Toc129961201)

[It could also be drawn to the foreground as a part of an object (e.g. a moving label) if you wish, but does not need to move around with objects if you don’t want it to. 5](#_Toc129961202)

[When the text changes, the old text should be appropriately removed from the screen. 5](#_Toc129961203)

[Be prepared to explain how this works to the marker if asked. This shows your understanding of drawing text to the foreground. 5](#_Toc129961204)

[The text has to be drawn such that moving objects would move under it rather than on top of it though. i.e. not to the background, and basically it means it’ll be drawn after at least some of the objects 5](#_Toc129961205)

[For marking we will check the code where it is drawn if there is any doubt. E.g. whether the function which draws it is called before or after drawing objects. (Look at the different functions to see which to use – the point of this mark is to see whether you realised the difference between drawing changing text to foreground rather than background. 6](#_Toc129961206)

[Provide a user controlled moving object which is a sub-class of DisplayableObject and different to the demos: 6](#_Toc129961207)

[Have a moving object 6](#_Toc129961208)

[that the user can move around, 6](#_Toc129961209)

[using either the keyboard OR the mouse (or both) and is different to the demos. 6](#_Toc129961210)

[Ensure that both keyboard and mouse input are handled in some way and do something 6](#_Toc129961211)

[You handle both keyboard input AND mouse input and they both do something 6](#_Toc129961212)

[Something(s) should visibly change for both – e.g. some position of something or value of something or displayed image, or… 6](#_Toc129961213)

[Provide an automated moving object which is a sub-class of DisplayableObject and different from the one in requirement 5. 6](#_Toc129961214)

[Provide a second moving object (separate to the user-controlled one, with a different class) 6](#_Toc129961215)

[whose movement is not directly controlled by the player, which moves around, 6](#_Toc129961216)

[and which looks different to the objects in the demos and to your object in requirement 5. 6](#_Toc129961217)

[Include collision detection between a user-controlled (req. 5) and an automated (req. 7) moving object, so that they interact with each other. 7](#_Toc129961218)

[You need to check for a collision between the two objects 7](#_Toc129961219)

[Something should happen when they collide, and something should visibly change – e.g. something moves, direction of travel changes, or something is shown 7](#_Toc129961220)

[Collision detection should be at least as good as rectangle-rectangle interaction and should work properly 7](#_Toc129961221)

[Create your own subclass of TileManager. 7](#_Toc129961222)

[Create a subclass of the tile manager which has different behaviour (at least a little) to the demos, is drawn to the background, and is visible to the user when the program is run. 7](#_Toc129961223)

[Name your class using your username followed by the text TileManager. e.g. if your username was psxabc then your class would be called PsxabcTileManager. 7](#_Toc129961224)

[Be prepared to explain the difference(s) from the demo versions to the marker 7](#_Toc129961225)

[Have at least one moving object interacted correctly with the tile manager, changing a tile: 7](#_Toc129961226)

[Ensure that at least one of your moving objects visibly changes specific tiles when it passes over them – using the position checking appropriately 7](#_Toc129961227)

[The tile must be changed and redrawn correctly so that the user sees the change 7](#_Toc129961228)

# Create an appropriate sub-class of BaseEngine with an appropriate background which is different from the demos

## Create an appropriate new sub-class of the base engine.

This is seen in **ProjectSrc/cw1\_main.h**. The class psybw7Engine is the sub class of the baseEngine parent class.

### Code snippet (1-5):

#pragma once

#include "../src/BaseEngine.h"

#include "TileMap.h"

class Psybw7Engine: public BaseEngine{

## Give it an appropriate background and ensure that the background is different to the demos

Chart

Description automatically generated with medium confidence

This background is rendered in the **void psybw7Engine::virtSetupBackgroundBuffer()** in cw1\_main.cpp lines(13-23)**.** The following code to render this is (see next page):

What draws the background Shape

Description automatically generated

    int colIndex = 0;

    int colour = 0x478aed;

    int iters = (getWindowHeight()/8)/0x5;

    for(int iY = 0; iY < getWindowHeight(); iY++){

        for(int iX = 0; iX < getWindowWidth(); iX++){

            setBackgroundPixel(iX,iY,colour);

        }

        if(iY % iters == 0){colour += 0x010100;}

    }

This works by setting each individual pixel on the screen and changing the colour of the row using the modulo operator %. I add 1 to the red and green for every modulo returning 0 to increase the gradient based on the height so no matter the height the gradient (should) stay the same. This background is different than the demos. I do not remember seeing any variation on gradients and even if they were some, my method to obtaining this result is way different.

## Name your class using your capitalised username followed by the text Engine. e.g. if your username was psxabc then your class would be called PsxabcEngine.

The class name can be seen [>>HERE<<](#_Code_snippet_(1-5):) , please refer to this.

## You MUST create a new class – you must not just change/rename one of the existing demo classes.

You can see that I made my own class that is a sub class of the BaseEngine parent class in **cw\_1main.h**. I did inherit functions from the parent class but added my own code such that there is a difference in functionality. I also added my own functions that did not derive from the parent class, called, **drawImage** and **setLabel**.

# Show your ability to use the drawing functions:

## Draw your background ensuring that you use at least one of the shape drawing functions to draw on the background

## and that you draw at least one image to the background,

## which is different from the demos and shows your understanding.

## Be prepared to explain what you have done to the marker if asked.

# Draw some text on the background

## Draw some text onto the background (not foreground) and ensure that the text is not obliterated when moving objects move over it

Ensure that moving objects can/do move over at least part of this text, so that the object appears in front of the text, and demonstrate that it is redrawn after the object has moved.

# Have some changing text, refreshing/redrawing appropriately which is drawn to the foreground (not background), in front of moving objects

## This text may change over time (e.g. showing the current time, or a counter) or could show a score which changes, for example.

## It could also be drawn to the foreground as a part of an object (e.g. a moving label) if you wish, but does not need to move around with objects if you don’t want it to.

## When the text changes, the old text should be appropriately removed from the screen.

## Be prepared to explain how this works to the marker if asked. This shows your understanding of drawing text to the foreground.

## The text has to be drawn such that moving objects would move under it rather than on top of it though. i.e. not to the background, and basically it means it’ll be drawn after at least some of the objects

## For marking we will check the code where it is drawn if there is any doubt. E.g. whether the function which draws it is called before or after drawing objects. (Look at the different functions to see which to use – the point of this mark is to see whether you realised the difference between drawing changing text to foreground rather than background.

# Provide a user controlled moving object which is a sub-class of DisplayableObject and different to the demos:

## Have a moving object

## that the user can move around,

## using either the keyboard OR the mouse (or both) and is different to the demos.

# Ensure that both keyboard and mouse input are handled in some way and do something

## You handle both keyboard input AND mouse input and they both do something

## Something(s) should visibly change for both – e.g. some position of something or value of something or displayed image, or…

# Provide an automated moving object which is a sub-class of DisplayableObject and different from the one in requirement 5.

## Provide a second moving object (separate to the user-controlled one, with a different class)

## whose movement is not directly controlled by the player, which moves around,

## and which looks different to the objects in the demos and to your object in requirement 5.

# Include collision detection between a user-controlled (req. 5) and an automated (req. 7) moving object, so that they interact with each other.

## You need to check for a collision between the two objects

## Something should happen when they collide, and something should visibly change – e.g. something moves, direction of travel changes, or something is shown

## Collision detection should be at least as good as rectangle-rectangle interaction and should work properly

# Create your own subclass of TileManager.

## Create a subclass of the tile manager which has different behaviour (at least a little) to the demos, is drawn to the background, and is visible to the user when the program is run.

## Name your class using your username followed by the text TileManager. e.g. if your username was psxabc then your class would be called PsxabcTileManager.

## Be prepared to explain the difference(s) from the demo versions to the marker

# Have at least one moving object interacted correctly with the tile manager, changing a tile:

## Ensure that at least one of your moving objects visibly changes specific tiles when it passes over them – using the position checking appropriately

## The tile must be changed and redrawn correctly so that the user sees the change