LIVERPOOL JOHN MOORES UNIVERSITY

Faculty of Engineering and Technology

Coursework Title:	Implementation of an interactive 2D computer graphics application
Module Name:	Mathematics and 2D Computer Graphics
Module Code:	4109COMP
Level:	FHEQ-4
Year (Semester):	2016/17 (Semester 2)
Module Credits:	20 credits

Weighting of this coursework towards the module mark (%)	50%
Maximum mark available:	100

If you have any issues with this coursework you may contact your lecturer whose contact details are:

Lecturer:	Dr. Sud Sudirman
Email:	s.sudirman@ljmu.ac.uk
Office No:	BS/7.27

Hand-out Date:	29th January 2018 (Week 19)
Hand-in Date:	20th April 2018 (Week 30)
Checkpoint Date:	5th March 2018 (Week 24)
Hand-in Method:	Electronically via Canvas (see section on "What you should hand in" below)
Feedback Date:	7th May 2018 (Week 33)
Feedback Method:	Checkpoint Feedback on Checkpoint Date Summative Feedback on Feedback Date

Programmes:	BSc. (Hons) Computer Games Development MCOMP Computer Games Development

Introduction

In this assignment you are required to design and develop a 2D interactive graphical application that demonstrates a wide range of computer graphics features and processes. The application is expected to demonstrate the following graphical features:

1. Graphics API setup and render window creation.
2. 2D assets loading and management
3. Text rendering
4. Static texture and sprite rendering
5. Texture and sprite rendering effects (colouring, blending, and transparency, etc.)
6. Animated sprite rendering
7. 2D transforms including, scaling, rotation and translation.
8. Texture scrolling and parallax scrolling
9. Collision detection and response.

This work must be completed individually.

Learning Outcome to be assessed:

3. Implement logical expressions and arithmetic models to represent the decisions and actions that form the mechanics an interactive 2D graphical application.
4. Construct mathematical models and apply them programmatically to control graphical primitives.
5. Use a modern graphics 2D API in conjunction with a high-level programming language to develop an interactive graphical 2D application.

Detail of the task:

You are required to develop a 2D computer game with the following specification:

The graphics setup

The game will use SFML as the graphics API. The game should run at 1920x1080 in full

screen. An FPS counter should be visible on the screen as a text message and this could be toggled on/off by pressing a certain key. The game loop should limit how often the game states are updated to 60 times per second. On the other hand, no limitation should be applied to how often the screen is updated.

The graphics assets
The game should use a wide range of graphical assets including textures, sprites, fonts, and asset metadata files. The game should be able to utilise the features of the graphics API as well as C++ library to load these assets and manage these assets in appropriate data structures. Whenever the assets are no longer needed, their space in the memory (RAM or VRAM) should be properly and promptly released.

You can find the asset files to be used in the Blackboard. You may use different assets, but you are advised not to spend significant amount of time to create the assets yourself. Any assets from third-party should be use according their terms of use (if any) and their original author properly credited.

The graphical user interface
The game should start by displaying a title screen. The game's main menu should include at least three options to a) Start a Game, b) Display Credits, and c) Quit the Game. When the game starts for the first time, on-screen texts should be displayed to show the background story of the game before continuing to the gameplaying state.

You have complete freedom in deciding the text of the game's background story. It should however consist at least 500 words in two paragraphs. The display of this story should be animated by making the texts appear letter by letter as if they are being typed.

The application will proceed to the gameplaying state if the user presses the Escape key or when the text typing animation is completed.

While in the gameplaying state, if the user presses the Escape key, the game should pause and show the same game menu as before, with one difference. The first option should display "Resume" instead of "Start a Game". Choosing to resume the game from this menu should not display the game story again but directly resumes the gameplaying from the last time it was left off.

When the Display Credits option is selected, the application should show the Credits Screen. The Credit Screen should display the following (in order):

- Game company logo (you have freedom on what images to use)
- Game company name (you have freedom on what to name it as)
- Lead programmer's name (your name)
- Lead artist's name (an made up name)
- Programmers' names (two made up names)
- Artists' names (two made up names)
- Acknowledgements (see acknowledgment section below) plus any others who have credits to any assets you use.

The credit should be displayed as scrolling text and images (from bottom to top) and centred justified on the middle of the window.

While in the gameplaying state, the game world is viewed from top-down directly perpendicular to the game world planes. In addition to showing the view of the game world, the application should also display relevant information about the player character in an overlaid Head Up Display (HUD). The HUD should show

- Character portrait
- Character health bar
- Image of the currently equipped weapon

The control system
The player controls a player character using a combination of keyboard and mouse controls. The WASD keys are used to make the character move forward, strafe left, move

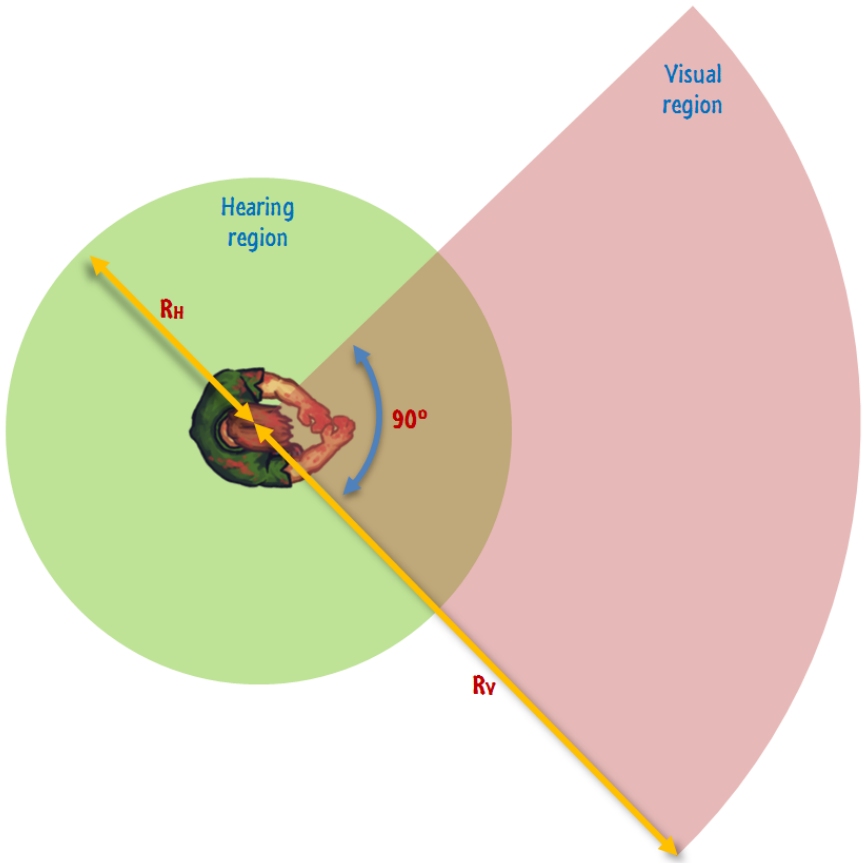
back, and strafe right respectively. When the right-mouse button is pressed, the character rotates to face the mouse pointer and when the left-mouse button is pressed, the character performs actions depending on the currently equipped weapon. The player can change the currently equipped weapon by pressing shortcut keys.

The camera system
The game world will be viewed using a virtual camera positioned directly above the player character looking straight down. Whenever the character moves the camera should move accordingly following the character. Changes in the viewable parts of the game world should be implemented as texture scrolling. In addition, there should be also at least two additional partially transparent layers of texture rendered after the character. The first one should scrolls independently to the character movement while the other should be fixed. These layers should visually be emulating objects at higher position and be implemented as parallax scrolling. The application should also emulate camera zooming via keyboard key presses.

The gameplay elements
In addition to the player character, there should also be a number of non-playable characters (NPCs). The NPCs should have the following behaviour:

1. Move to randomly in either idle when the player character is not sensed
2. Move towards the location of the player character and into attacking state if they sense its presence.

The sensing is emulated as two checks, one (to emulate hearing) checks the distance against hearing distance (R_H) and the other (to emulate seeing) checks the distance against visual distance (R_V) and a 90° view field as illustrated in the figure below. If any one of the checks is satisfied then the NPC will go into attacking state.



The player should be able to attack these NPCs at melee or range (via shooting bullets). Collision detection algorithm should be applied to work out if collision has occurred

between each sprite's bounding boxes. Appropriate changes in the game states should be applied accordingly if a collision has occurred. Visual cues should also be applied to the sprite to provide feedback to the player. This could be implemented, for example, as flashing the sprite with red tint or by applying blood splatter animation.

The game world should be tinted black to simulate night time. The player will be able to see areas immediately surrounding the character and in front of it as lit up by a torch. This effect could be achieved using additional SFML RenderTexture and by appropriate applying texture transparency and blending effects. A light mask image is provided on the Blackboard for this purpose.

The sprite animation
All characters should be properly animated depending on their states and currently equipped weapon. In total, 16 main character sprite sheets and 3 NPC sprite sheets are provided and can be downloaded from the Blackboard. You are expected to produce sprite info files for each of them.

The application should be developed using Object Oriented C++ programming language as a Microsoft Visual Studio C++ project.

Required resources:

The main source of information to complete this coursework is the lecture and workshop materials. The workshop in particular will provide starting points to some of the tasks that you are required to complete. Therefore, completing them timely and correctly is essential and should provide you with a very good grounding on completing the more complex tasks. To complete the more complex tasks, you will need to carry out problem solving steps (analysis, design, implementation and evaluation) independently by drawing on the knowledge you learn from the lecture and workshop sessions.

To complete this coursework you will need to have access to:

- Lecture and workshop materials, which are provided on the Blackboard.
- The 2D graphics assets (texture, fonts, sprites) which are provided on the Blackboard.
- A personal computer with modern CPU and GPU running Microsoft Windows (version 7 or later).
- Microsoft Visual Studio 2013 or 2017 with Visual C++ compiler.
- Simple and Fast Multimedia Library (SFML) 2.4.x with C++ language binding.
- Image editing software, e.g., GIMP or Adobe Photoshop.
- Textbooks and online resources (see the *Recommended reading* section below)

These resources are all available across all computing labs on the 6th and 7th floors of the James Parsons Building, Byrom Street or on HC/1.61 and HC/1.73 in Henry Cotton building. Textbooks are available via the library (electronic and prints).

Important note: *Any assets from third-party should be use according their terms of use (if any) and their original author properly credited.*

What you should hand in:

You are required to submit the following:

1. The Visual Studio project and solution files, and all the necessary files to build the project. The project's properties should be set properly to use SFML in both Debug and Release configurations.
2. The SFML dynamically link library (.dll) files, as required by features of SFML that you use, stored in the relevant directory.
3. The 2D graphics assets files as required by your application, stored in the relevant directory.

You should zip all files into one archive and submitted electronically via the Canvas assessment handler by the submission deadline. Remember that you are required to keep a personal copy of your submitted work in case there is a problem retrieving your work from the Canvas.

Important note: *If you are unable to meet the deadline due to an acceptable mitigating circumstance please inform the module leader before the deadline date. Any submissions*

after the deadline will be awarded an automatic zero mark for this component. If you did not submit anything for this assessment, you will fail the module.

Acknowledgment:

A number of 2D assets are included in this assignment for you to use. These assets are obtained from a number of sources and used within the term allowable by the original owners:

- wire.png (http://dementiarunner.deviantart.com/art/wire-seamless-png-549508647)
- dirt.png (http://hhh316.deviantart.com/art/Seamless-Cracked-Dirt-Texture-347828451)
- fog.png (http://www.freeiconspng.com/png-images/)
- Player character sprite images (http://opengameart.org/content/animated-top-down-survivor-player)
- NPC sprite images (http://opengameart.org/content/animated-top-down-zombie)
- Blood sprite images (http://opengameart.org/content/blood-effect-sprite-sheet)
- game-logo.png (http://dogtimes.wps60.org/?p=2866)
- menu.png (https://wall.alphacoders.com/by_sub_category.php?id=81399)
- SegoeMarker.ttf (https://www.azfonts.net/families/segoe-marker.html)

Assessment Criteria:

Assessment Criteria	**Weights (%)**
Graphics setup and initialisation	5
Render window creation	*(2)*
Game loop update and render timing	*(3)*
Handling of the graphics assets	5
Asset loading	*(2)*
Asset management	*(3)*
Implementation of graphical user interface	25
Title screen	*(2)*
Game menu	*(3)*
Background story screen	*(5)*
Credit screen	*(5)*
Heads Up Display	*(10)*
Implementation of control system	5
Character movement	*(2)*
Character facing	*(2)*
Character attack	*(1)*
Implementation of camera system	10
Camera setup and update	*(2)*
Texture scrolling	*(3)*
Parallax scrolling	*(3)*
Camera zoom	*(2)*
Implementation of gameplay elements	25
Implementation of various NPC states and behaviours	*(5)*
Implementation of shooting action	*(5)*
Collision detection	*(3)*
Collision response and visual cue	*(7)*
Night time and flashlight lighting simulation	*(5)*
Implementation of sprite animation	20
Sprite sheet preparation	*(5)*
Animation frames (Rectangles and Centres)	*(5)*
Animation playback, transition and control	*(5)*
Animation quantities	*(5)*
Structure of program	5

Setup of Visual Studio project properties to use SFML	*(2)*
Good use of object oriented paradigm	*(3)*

Recommended reading:

Course Material	Official Website
Author	Laurent Gomilla
Publishing Year	2016
Title	Simple and Fast Multimedia Library (SFML)
URL	http://www.sfml-dev.org/learn.php

Course Material	Book
Author	Jan Haller, Henrik Vogelius Hansson, and Artur Moreira
Publishing Year	2013
Title	SFML Game Development
Subtitle	Learn how to use SFML 2.0 to develop your own feature-packed game
Publisher	PACKT Publishing
ISBN	9781849696845

Course Material	Book
Author	Raimondas Pupius
Publishing Year	2015
Title	SFML Game Development By Example
Subtitle	Create and develop exciting games from start to finish using SFML
Publisher	PACKT Publishing
ISBN	9781785287343

Course Material	Book
Author	Tomas Akenine-Möller, Eric Haines, Naty Hoffman.
Publishing Year	2008
Title	Real-time rendering
Subtitle	
Edition	3rd Edition
Publisher	A.K. Peters
ISBN	9781568814247

Course Material	Book
Author	Jason Zink
Publishing Year	2011
Title	Practical rendering and computation with Direct3D 11
Subtitle	
Edition	
Publisher	CRC Press
ISBN	9781568817200

Course Material	Book
Author	James M. Van Verth and Lars M. Bishop
Publishing Year	2016
Title	Essential Mathematics for Games and Interactive Applications
Subtitle	
Edition	3rd Edition
Publisher	Apple Academic Press
ISBN	9781482250923

Extenuating Circumstances:

If something serious happens that means that you will not be able to complete this assignment, you need to contact the module leader as soon as possible. There are a number of things that can be done to help, such as extensions, waivers and alternative assessments, but we can only arrange this if you tell us. To ensure that the system is not abused, you will need to provide some evidence of the problem.

	More guidance is available at https://www.ljmu.ac.uk/about-us/public-information/student-regulations/guidance-policy-and-process **Any coursework submitted late without the prior agreement of the module leader will receive 0 marks.**
Academic Misconduct	The University defines Academic Misconduct as 'any case of deliberate, premeditated cheating, collusion, plagiarism or falsification of information, in an attempt to deceive and gain an unfair advantage in assessment'. This includes attempting to gain marks as part of a team without making a contribution. The Faculty takes Academic Misconduct very seriously and any suspected cases will be investigated through the University's standard policy (https://www.ljmu.ac.uk/about-us/public-information/student-regulations/appeals-and-complaints). If you are found guilty, you may be expelled from the University with no award.

- End of Assignment Specification -