****

**Faculty of Science and Technology**

**2015/2016**

|  |  |  |
| --- | --- | --- |
| **Assignment Brief** | | |
| Programme/s:  BSc Games Programming  BSc Game Technology | Unit Name:  Software Programming | |
| Assignment set by:  Andrew Watson | C:\Users\Cooper\Desktop\ScreenHunter_01 Apr. 17 18.23.jpgQA:  Dr. Kendra Cooper | |
| Assignment Issued:  **11/01/2016** | Recommended time to complete this assignment:  50 hours | |
| **Date Due:**  **07/03/2016** | **Unit Weighting:**  **50%** | **Assignment number:**  **2** |
| **PLEASE DELETE ANY THAT DO NOT APPLY:**  ☐ Assignment is to be electronically submitted via myBU by 12:30pm on the due date (please allow sufficient time to upload files before the deadline) | | |
| **Assessment Task:** | | |
| **ASSESSMENT TASK:**  This is an individual assignment.  To demonstrate your understanding of both program design concepts and the C++ programming language you are required to design and implement a simple 2D game programmed in C++ which uses the Allegro 4 v4.2.3 game programming graphics library.  The game can be anything the designer wants it to be and can be of any genre.  However, the game, **at a minimum**, must contain the following elements.  The game must have an end goal  The game has to be in graphics mode (No text/console only games)  The game must contain bitmap images (i.e. not just constructed from graphic primitives).  The game must contain a user controllable object (for example tank, space ship).  The game must contain at least one additional non-user controlled object (for example, asteroids)  The game must allow user input / control by both keyboard and mouse  The game must provide the user with feedback both audio and visual (explosions, sound etc.)  The game must contain a background sound track  The game must provide the current score  The game should be single player only  The game must provide the user with simple instructions  The game must begin and end in a controlled manner (for example; Menu, Hot keys)  The game must implemented be using at a minimum two meaningful C++ classes  The student will demonstrate their game and undergo an oral examination which will be held during the laboratory sessions of the week starting the 07/03 /2016. (Due to the large student numbers the presentation will be conducted over two weeks).  *Note: Extra bonus marks may be awarded for originality, playability and humour!*  **DELIVERABLES:**     1. Design and Implement the game application using Microsoft Visual Studio 2012 C++ and Allegro 4 v4.2.3 Library to meet the above requirements. 2. Complete and submit the supplied project synopsis document which includes sections for a brief description of the game, it’s inspiration, a summary of any similar games, all references ( including documents ,source code and web sites) and the self-assessment. 3. The student must provide a recorded video of the game being played. The recording should capture all the main features of the game The video should be recorded mp4. 4. Undertake 15 minute demonstration / and oral examination of the system during which you will be required to defend your work. 5. The submission is an Online Submission through myBU. Submit a zipped file of the complete game project. The zipped file will include all files, i.e. all C++ source and header files, game assets files, music files, the project synopsis and any supporting documents. and the video recording file.   **Note :**  Please make sure that the submission contains all files needed to run your application without having to recompile the project this should be checked/tested before you submit.  . **MARKINGSCHEME:** Oral Examination Details *The student should be able defend their work and be able discuss all aspects of the application development stages: the idea, analysis, design, implementation and testing.*  *The student* ***may*** *provide/use prepared notes / documentation / diagrams to support their defence of their work during their oral examination The following list details some content that the student may find useful*   1. System block diagram, top level design and key implementation features. 2. Testing strategy, test results and problems found.   3. Discussion and conclusions  4. The description of the game program  5. The analysis of the problem, design and modular decomposition  6. Program structure presented using structure chart / or class diagrams  7. Detailed modules designs presented using pseudo code or flow charts  8. Complete Source Code Listing.   * Ensure that assignment is complete in the 3 sections * **Analysis** * **Design** * **Implementation**   Analysis   * This section is to do with the analysis of the problem. * Break down of the problem into distinct modules, indicating the inputs, processes and outputs. * What will the program do? What is the logic of the program and how it will function? * Focus on identifying modules and program structure / Operation, (not how they are coded). * IPO diagram and structure diagram. (Top level Flowchart or Top Level Pseudocode).   Design  The design section is to with the detailed design of each of the modules that have been identified in the analysis section.  The modules design will be expressed in terms of input, process and output with respect to the required functionality.  The modules functionality will be presented using an Flow Charts ,supporting design description and or pseudo code.  Implementation  The implementation section is to do all aspects of the applications implementation  This section will contain commented source code for the application. The application’s source code will be reviewed with respect to, coding standards (layout formatting, comments) , its structure and its logical separation of the program into modules .  The executable will be demonstrated and measured against the defined marking criteria.  The implementation section will contain the testing procedures used during the testing of the application  **LEARNING OUTCOMES**  Having completed this assignment the student is expected to be able to demonstrate:   1. Write and test a software system in a comprehensive and unambiguous manner; 2. Develop understanding of, and skills in, the planning and executing of the structural and functional testing and debugging of a software application; 3. Understand the use of computer algorithms in software programs; 4. Understand the use of data structures and formats used in software programs. | | |
| **Confirmation that this assignment assesses the relevant ILOs: Yes/No** | | |
| **ALL Written assignments must be submitted before 12:30pm on the date due.**  **Electronic submission time will be 12:30pm on the due date following the above assignment detail, note this deadline is the time for the upload of the assignment to be completed, you are advised to begin submission AT LEAST 1 hour before.**  **For submission of hard copies (where required) the assignments submission room is on the first floor of Christchurch House, in the corridor near C114. Assignments must be submitted here before 12:30pm**  **Please note that as per the Standard Assessment Regulations any coursework submitted after the deadline will be capped at 0%.**  **Capped assignments will be considered by the Board of Examiners and cannot be retrospectively uncapped by Academic Staff.**  You must keep a copy of your assignment – the university will not take responsibility for lost assignment.  If you are unable to submit on time due to medical or other circumstances you MUST obtain an approved extension PRIOR to the submission deadline. Extension Request Forms are available from C237 or on the university website under Student Policies, Regulations and Procedures / Assessment then Mitigating Circumstances <http://portal.bournemouth.ac.uk/C11/Mitigating%20Circumstances/default.aspx>).  **Plagiarism**  Plagiarism is the act of copying the work or ideas of others without proper acknowledgement of this work.  Plagiarism also includes self-plagiarism or duplication: the inclusion in coursework, or a dissertation, or project, of any material which is identical or substantially similar to material which has already been submitted for any other individual assessment within the University or elsewhere.  Avoiding plagiarism is best achieved through the use of proper academic referencing and minimising direct quotations (i.e. re-write others’ ideas in your own words, but still provide the reference of where these ideas came from).  Further information can be found here:  <http://www.bournemouth.ac.uk/library/how-to/plagiarism.html>  and  <http://www.bournemouth.ac.uk/library/how-to/academic-offences.html> | | |

**MARKINGSCHEME: Generic Criteria**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Deliverable | 0% -> 39% | 40% -> 49% | 50% -> 59% | 60% -> 69% | 70%+ |
| **Software Artefact.** | 1. No evidence of modularity. 2. No evidence of programming style. | 1. Little evidence of modularity. 2. Little evidence of programming style. | 1. Evidence of modularity. 2. Evidence of the use of programming style. | 1. Good modularity. 2. Good programming practices. | 1. Excellent modularity. 2. Excellent programming practices. |
| **Demonstration & Viva.** | 1. No real professionalism. 2. Demonstration provided no real evidence of understanding or application of appropriate theory and principles. 3. No real evidence of meeting system requirements. 4. Slow, unconfident responses. 5. No real evidence of understanding of technical issues. 6. No real knowledge of system operation. | 1. Little professionalism. 2. Demonstration provided some evidence of understanding and application of appropriate theory and principles. 3. Some evidence of meeting system requirements. 4. Slow responses. 5. Some evidence of understanding of technical issues. 6. Some knowledge of system operation. | 1. Reasonable professional approach taken. 2. Demonstration provided a reasonable evidence of understanding of appropriate theory and principles. 3. Reasonable evidence of meeting system requirements. 4. Reasonably confident and quick responses. 5. Reasonable evidence of understanding of technical issues. 6. Reasonable knowledge of system operation. | 1. Good professional approach taken. 2. Demonstration provided a good evidence of understanding of appropriate theory and principles. 3. Good evidence of meeting system requirements. 4. Good confidence and quick responses. 5. Good evidence of understanding of technical issues. 6. Good knowledge of system operation. | 1. Very professional Demonstration. 2. Demonstration provided excellent evidence of understanding and application of appropriate theory and principles. 3. Fully demonstrated all system requirements. 4. Very confident and quick responses. 5. Excellent evidence of understanding of technical issues. 6. Excellent knowledge of system operation. |
| **Design and Implementation.** | 1. No evidence of modularity. 2. No evidence of programming style. 3. No Real Test Strategy or results. 4. No Discussion & Conclusion. | 1. Little evidence of modularity. 2. Little evidence of programming style. 3. Little shown on testing, limited test results shown. 4. Limited discussion and conclusion. | 1. Evidence of modularity. 2. Evidence of the use of programming style. 3. Evidence of a test strategy and results. 4. Some discussion and Conclusions. | 1. Good evidence of modularity. 2. Good evidence of the use of programming style. 3. Good evidence of a test strategy and results. 4. Discussion and Conclusions addressed the main points. | 1. Excellent modularity. 2. Excellent programming practices. 3. Excellent test strategy and results shown. 4. Comprehensive discussion with comprehensive supporting conclusions. |

**MARK DISTRIBUTION: Specific Criteria**

|  |  |
| --- | --- |
| **Element** | **Contribution.** |
| **Software Artefact.** | **30%** |
| **Demonstration &Viva.** | **45%** |
| **Design and Implementation** | **25%** |

**Appendix 1**

**Games Synopsis**

**References**

**BSc Games Programing and Game Technology - Level C**

**Software Programming Assignment Part 2– Self-Assessment of Performance**

**Issued :**

**Hand-in :**

**Tutor :** Andrew Watson

|  |  |
| --- | --- |
| **Student’s Name** |  |

Circle the appropriate response:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Did I submit the assignment on time?** | **Yes** | **No** |  |  |
| **Did I complete the assignment?** | **Yes** | **No** |  |  |
| **If no, approx. how much did I complete** | **%** |  |  |  |
| **How happy am I with what I submitted?** | **Very happy** | **Satisfied** | **Disappointed** | **Ashamed** |
| **What mark do I expect?** | **%** |  |  |  |
| **Did I spend enough time on the assignment** | **Yes** | **No** |  |  |
| **Did I get it proof-read by someone else?** | **Yes** | **No** |  |  |
| **Have I properly ‘referenced’ it?** | **Yes** | **No** |  |  |
| **Could I improve the presentation?** | **Yes** | **No** |  |  |

Answer the following questions:

|  |  |
| --- | --- |
| **The best part of my performance was:** |  |
| **The worst part of my performance was:** |  |
| **One way in which I could improve the content of my assignment is:** |  |
| **One way in which I could improve the presentation of my assignment is:** |  |
| **One thing I will do to improve my performance in my next assignment is:** |  |
| **Another thing I will do to improve my performance in my next assignment is:** |  |