# Southampton Solent University

# Coursework Assessment Brief

# Assessment Details

|  |  |
| --- | --- |
| Unit Title: | Programming Fundamentals |
| Unit Code: | DAC416 |
| Unit Leader: | Mark Bennett |
| Level: | 4 |
| Assessment Title: | Project AE2 |
| Assessment Number: | AE2 |
| Assessment Type: | Software Project and Report |
| Restrictions on Time/Word Count: | 750 words maximum plus project files |
| Consequence of not meeting time/word count limit: | There is no penalty for submitting below the word/count limit, but students should be aware that there is a risk they may not maximise their potential mark. |
| Individual/Group: | Individual |
| Assessment Weighting: | 60% |
| Issue Date: | 11/11/2019 |
| Hand In Date: | 03/12/2019 |
| Planned Feedback Date: | January 2020 |
| Mode of Submission: | On-line through Solent Online Learning |
| Number of copies to be submitted: | n/a |
| Anonymous Marking | This assessment is exempt from anonymous marking |

# Assessment Task

Create a “Match-3” style game where players swap crystal positions to place like-coloured crystals together in order to destroy them and gain points.

Crystals are arranged in a grid formation. The player selects a crystal to and then selects an ADJACENT crystal. If the swap would result in a string of 3 or more crystals of the same colour, the swap is considered valid.

If the swap is invalid, the crystals return to their original position and the move is disregarded.

If a valid swap is made, all crystals in the resultant string of crystals are destroyed. The newly formed gap in the grid is closed by all non-destroyed crystals shunting along in the grid. Once all crystals are shunted, the grid is “topped up” with new crystals of random colours. This entire process is called a “cascade”.

If a cascade creates a new string of 3 or more like-coloured crystals, the string is destroyed and another cascade occurs. The cascade process repeats until no more strings are created.

***Game Project***

Project Requirements

These features require a good understanding of the coding concepts covered in the unit; solid implementation will demonstrate your ability to apply them in more complex problems as required in your next year of study.

1. The game board is an eight-by-eight grid.
2. Crystals can be one of up to SIX colours (different characters can be used instead) – Red, Green, Yellow, Blue, White, and Purple.
3. Top-up crystals are added from the TOP edge of the board.
4. The game ends when no more valid swaps can be made.
5. An intermediate scoring system is in place:

| **Action** | **Points scored** | **Example** |
| --- | --- | --- |
| 3-string crystal destroyed | 50 | 3-red-crystal-string destroyed:  **= 50 points** |
| 4 string crystal destroyed | 100 | 4-red-crystal-string destroyed:  **= 100 points** |
| 5-crystal string destroyed | 500 | 5-red-crystal-string destroyed:  **= 500 points** |
| 6-crystal string destroyed | 2500 | 6-red-crystal-string destroyed:  **= 2500 points** |
| T-shape string | + (value of largest string destroyed \* 3) | T-shape created from a 3-string & 3-string:  **+(50 \* 3)**  **= 150 points**  T-shape created from a 3-string & 5-string:  **+(2500 \*3)**  **= 7500 points.** |
| Cascade | + (50 \* (number of cascades to the power of 2)) | *Single cascade:*  **+(50 \* (1^2))**  **= 50 points**  *Triple cascade:*  **+(50 \* (3^2))**  **= 450 points**  *5 x cascade:*  **+(50 \* (5^2))**  **= 1250 points** |

1. High scores are displayed.

#### These minimum requirements must be met to achieve a passing mark

1. The Game Project will be written in C++ using Visual Studio 2015 as "Win32 Console Application". No additional engines or libraries are allowed.
2. The game board grid must be at least a single row of eight spaces.
3. Crystals can be any one of at least **THREE** colours (different characters may be used instead of colours). The following colours must be used: Red, Green, Yellow. It is recommended that if your game board is of a single row of eight spaces, you limit the number of colours to just three to allow for a playable game.
4. When a string of crystals is destroyed, the remaining crystals MUST cascade. Top-up crystals are added to the board from an edge (for basic requirements, we recommend the RIGHT edge).
5. The player must be allowed to exit the game.

Project Guidance

To get started, you may use the template project solution provided for you on Solent Online Learning and any program components created for class exercises, including pseudocode, flowcharts, or code – provided that it was created by you alone.

It is recommended that you do not focus on the creative/story potential of the project unless you have already built a functional game. You may choose to embellish the basic game, but be careful not to compromise the intent of the assessment brief. Additional marks may also be awarded for flair, finesse and imagination – but remember: combine your creativity with demonstrations of excellent coding practice.

***Project Report***

The Project Report accompanies the programming project and contains the written documentation of your approaches to identifying and solving the assessment, and how your program implemented the solution. It must contain at least these 3 sections which correspond to the assessment marking criteria:

Problem Analysis and Design

To design this program you need to thoroughly analyse the requirements and decompose the overall problem into smaller problems which are individually solvable. You will need to consider exactly what the player will input to the program and what information the program will output to the player as well as what information it needs to store in order to function correctly.

Designs for your solutions must be demonstrated using appropriate tools, such as pseudo-code and flow diagrams, with stepwise refinement used to keep each at a reasonable level of detail. Compare your designs against the assignment brief and if you notice errors or omissions then make appropriate changes.

It is not unusual for the complete solution to develop incrementally as you alternate between planning and implementing. It is important that some program design is in place before implementation begins, and that you show how your design evolved over time to completely solve the problems. Keep track of changes to your design so they can be mentioned in the project report.

Things to consider:

* Storing/Updating the game board – how you represent the board internally may differ greatly from how it should be displayed
* Displaying instructions and game status, score, move remaining, etc.
* Capturing and validating input from the player
* Determining win/lose conditions and declaring such
* How can you minimise duplicated code?
* How to test that each action is functioning correctly

Implementation

Use the design to implement your solution in C++, with well-structured code that follows best practices. You should describe how your implementation proceeded from planning and design, and how any difficult obstacles were encountered were overcome.

The implementation should closely follow the design. If the design is found to be flawed once implementation begins, then a redesign should be thought out carefully and documented, describing what you had to change and why.

It is worth additionally checking that the final implementation fulfils the original assignment brief.

You DO NOT need to include source code in the body of the report, but you may want to include it in the appendix, especially if you want to refer to specific parts in the report.

Test Planning and Testing

Implementation should be tested *as you develop the project*, using the debugger, output values, test code, etc. Good testing does not mean just making sure the end product is free of obvious bugs – it means testing throughout development to ensure that as each component is completed, it and all previous code remains solid and functional. Use a range of tests that try to explore all the potential paths that the program can go down to confirm that it is valid.

Testing should include a broad Test Plan that shows the functionality to be tested, the expected result, and what result was actually obtained. Incorrect or unexpected results should be analysed, and tests and fixes should be documented. If the design must be updated because of a program error, the Design section in your report should reflect the changes.

Providing evidence that supports your testing is essential as it shows that you have followed through with testing, and that you understand how to correctly interpret and respond to test results. Evidence can include screenshots of the running program, screenshots of the debugger, error messages and references to any test code.

***Hand-in Details***

The Game Project and Project report must be submitted online before the deadline to receive full marks. Your submission should contain two files:

1. Your Project Report document, containing the design, testing, conclusions and any supporting diagrams, tables or charts.
   * The front page of this report should be a title page that contains at least the information "**[your full name] DAC416\_AE2-R\_Report".**
   * The document should preferably be in MS Word format.
   * Your source code may optionally be included as an appendix in the report.
2. A compressed/zipped file containing your Game Project
   * Name the file "**[your full name] DAC416\_AE2-R\_Project"**
   * Include all of solution and source code files: .h and .cpp files and any additional files you have created as resources for you project.
   * Make sure you include all source and resource files necessary to build and run your program, including any files supplied to you.
   * You do not need to include temporary or intermediate files created during the build process

Due to the volume of network traffic especially near deadlines, the online submission system may be slower than you expect. Extensions for connection problems will not be granted unless university-wide. Give yourself plenty of time!

You **MUST** keep an exact copy of your project stored on your U: drive as backup in case of submission failure.

Ensure that the PC used to submit your work or create your submission package is free of viruses or malware. Submitting digital media containing any form of malware will result in significant penalty or failure.

# Assessment criteria

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CRITERIA** | **A1 – A4** | **B1 – B3** | **C1 – C3** | **D1 – D3** | **F1 – F3** |
| **DESIGN**  **Suitability and quality of problem solving techniques, appropriateness of design.**  **(25%)** | **An excellent and sustained use of problem solving techniques and documentation, along with consideration of relevant multiple potential solutions and justification for best one.**  **Elegant design, which may usefully extend the brief.**  **Well documented evidence of iterative design.** | **Thorough and consistent use of problem solving techniques, with a detailed analysis of the problem.**  **Evidence of appropriate redesign.**  **Design fully solves problem.** | **Clear use of appropriate problem solving techniques.**  **A full analysis of the problem, but lacking in detail.**  **Design may have minor errors and omissions.**  **Design fully solves problem, but in a basic fashion.** | **Some use of basic problem solving techniques.**  **Partial analysis of problem with some inaccuracies.**  **Design is appropriate, but may not fully solve problem.** | **Inadequate understanding of theory leading to poor or non-existent problem solving attempt.**  **Analysis is simplistic with inaccuracies and omissions.**  **Design not appropriate or does not solve problem.** |
| **IMPLEMENTATION**  **Understanding and proper use of programming syntax to create application that fulfils design.**  **(60%)** | **Excellent and wide ranging knowledge of programming.**  **Fully functional application may incorporate advanced techniques.**  **Application fulfils design in an elegant and/or efficient way.** | **Shows a clear understanding of programming syntax and constructs, used extensively and appropriately to create fully functional application.**  **Application fully realises design.** | **Basic level of competence in all programming syntax and constructs.**  **Application fully functional, but may not fulfil entire design.** | **Basic understanding of some syntax shown, but may not be complete or fully appropriate.**  **Occasional minor errors in application, some parts may fulfil design.** | **Significant deficiencies in understanding of program syntax.**  **Application missing or non-functional, or clearly does not fulfil design.** |
| **TESTING**  **Suitability and quality of application testing.**    **(15%)** | **Excellent and sustained use of many testing techniques with a full and detailed analysis of the application.**  **In-depth analysis of test results and well documented resolutions to problems encountered, with useful reference to code involved.** | **Thorough and consistent use of several testing techniques, with a full analysis of the application.**  **Comprehensive analysis of test results, with documented reference to problem resolution.** | **Clear use of some appropriate testing techniques**  **A full analysis of the application, though lacking in detail.**  **Tests cover all program paths, but may have minor errors and omissions.**  **Moderate analysis of test results with some reference to problem resolution.** | **Some use of basic testing techniques.**  **Some evidence of the use of the debugger and/or program output to test the program.**  **May have incorrect results, or misinterpretations of results.**  **Little analysis of results or diagnosis of problems found.** | **Little or no testing carried out, or inappropriate or unrelated testing reported.**  **No analysis of results or discussion of test results.** |

# Learning Outcomes

This assessment will enable students to demonstrate in full or in part the learning outcomes identified in the unit descriptors.

# Late Submissions

Students are reminded that:

1. If this assessment is submitted late i.e. within 5 working days of the submission deadline, the mark will be capped at 40% if a pass mark is achieved;
2. If this assessment is submitted later than 5 working days after the submission deadline, the work will be regarded as a non-submission and will be awarded a zero;
3. If this assessment is being submitted as a referred piece of work (second or third attempt) then it must be submitted by the deadline date; any Refer assessment submitted late will be regarded as a non-submission and will be awarded a zero.

<http://portal.solent.ac.uk/documents/academic-services/academic-handbook/section-2/2o-assessment-policy-annex-1-assessment-regulations.pdf?t=1411116004479>

# Extenuating Circumstances

The University’s Extenuating Circumstances procedure is in place if there are genuine circumstances that may prevent a student submitting an assessment. If students are not 'fit to study’, they can either request an extension to the submission deadline of 5 working days or they can request to submit the assessment at the next opportunity (Defer). In both instances students must submit an EC application with relevant evidence. If accepted by the EC Panel there will be no academic penalty for late submission or non-submission dependent on what is requested. Students are reminded that EC covers only short term issues (20 working days) and that if they experience longer term matters that impact on learning then they must contact a Student Achievement Officer for advice.

A summary of guidance notes for students is given below:

<http://portal.solent.ac.uk/documents/academic-services/academic-handbook/section-4/4p-extenuating-circumstances-procedures-for-students.pdf?t=1472716668952>

# Academic Misconduct

Any submission must be students’ own work and, where facts or ideas have been used from other sources, these sources must be appropriately referenced. The University’s Academic Handbook includes the definitions of all practices that will be deemed to constitute academic misconduct. Students should check this link before submitting their work.

Procedures relating to student academic misconduct are given below:

<http://portal.solent.ac.uk/support/official-documents/information-for-students/complaints-conduct/student-academic-misconduct.aspx>

**Ethics Policy**

The work being carried out by students must be in compliance with the Ethics Policy. Where there is an ethical issue, as specified within the Ethics Policy, then students will need an ethics release or an ethical approval prior to the start of the project.

The Ethics Policy is contained within Section 2S of the Academic Handbook:

<http://portal.solent.ac.uk/documents/academic-services/academic-handbook/section-2/2s-university-ethics-policy.pdf>

**Grade marking**

The University uses a letter grade scale for the marking of assessments. Unless students have been specifically informed otherwise their marked assignment will be awarded a letter grade. More detailed information on grade marking and the grade scale can be found on the portal and in the Student Handbook.

Policy: <http://portal.solent.ac.uk/documents/academic-services/academic-handbook/section-2/2o-assessment-policy.pdf>

**Guidance for online submission through Solent Online Learning (SOL)**

<http://learn.solent.ac.uk/onlinesubmission>