

College of Engineering, Construction and Living Sciences Bachelor of Information Technology

IN628 Programming 4

Level 6, Credits 15

## Assessment Task

Assessment 1: Roguelike



# Assessment 1: Roguelike

**Due Date:** Friday, 1st November, 5.00 pm – code freeze.

**Value:** 45% of your final mark.

**Group Size:** Group of two.

## Learning Outcomes Met By This Assessment:

At the successful completion of this course, learners will be able to:

1. Program effectively in an industrially relevant programming language
2. Implement a wide range of intermediate data structures and algorithms to act as modules of larger programs
3. Use an appropriate integrated development environment to create robust applications

## Assessment Overview

For this assessment, you will use **Visual C++** with **Visual Studio 2017** to build **a 2D Roguelike** game (a dungeon crawler with procedurally generated dungeons).

**Assessment Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Assessment Activity** | **Weighting** | **Learning Outcomes** | **Assessment Grading Scheme** | **Completion Requirements** |
| Roguelike | 45% | 1-3 | Percentage/CRA | 50% |

## Functional requirements. The application must:

|  |  |
| --- | --- |
| **System:** | |
| **1** | Open without modification in Visual Studio 2017. |
| **2** | Display well at 1920x1080. |
| **Game world:** | |
| **1** | Implement procedurally generated dungeons containing multiple rooms, connected by corridors and containing a randomly located stair (or equivalent portal) to the next dungeon. |
| **2** | Represent and display dungeons as tile-maps. A "dead zone" of 1/2 the dimension of  the viewable area is permitted at each edge of the world. |
| **3** | Implement “fog of war”. That is, the dungeon must be progressively revealed as the  player character moves through the world. |
| **Entities:** | |
| **1** | Contain one or more player characters under user (keyboard) control. |
| **2** | Contain at least two types of animated enemies placed randomly in the dungeon. Enemies may be confined to experience levels (or similar game play rule). Enemies  must have distinct visual representations and behavior statistics. |
| **3** | Contain at least one type of item that directly impacts the game score (i.e. gold,  treasure, etc.) |
| **4** | Contain at least two types of item which affect the player’s condition (e.g. increase or  decrease health, increase or decrease attack strength, etc.) upon contact. |
| **5** | Demonstrate correct sprite to terrain collision detection. Players and enemies may not  walk through walls. |
| **6** | Demonstrate correct sprite to sprite collision detection. Collision between player and item  affects the player’s condition and/or score. Collision between player and enemy  initiates battle. |
| **7** | Implement a battle system. Turn-based, to-the-death is acceptable. |
| **8** | Implement at least one enemy that exhibits complex programmed behaviour (i.e. AI). |
| **9** | Contain at least one game element whose behaviour involves trigonometric  computation, as discussed in class (i.e. trajectory, rotation, and/or orientation). This |

|  |  |  |
| --- | --- | --- |
|  | | computation may be part of the AI requirement, #8 above. |
| **10** | | Use a Finite State Machine to control the behaviour of one or more entities. |
|  | **Game play:** | |
| **1** | | Have a clearly defined and displayed scoring system. |
| **2** | | Have a clearly defined and displayed loss condition. |
| **3** | | Provide appropriate user feedback. |
| **4** | | Be visually attractive, with a coherent graphical theme and style. Please include correct citations for all externally-sourced graphic elements. All media must be royalty free (or  legally purchased) for educational use. |
| **5** | | Provide an interesting game play experience. |

**Marking Rubric**

Attached at the end of this document.

## Group Contribution

All git commit messages must identify which member (or members) of the two participated in the associated work session. Proportional contribution will be determined by inspection of the commit logs. If the commit logs show evidence of significantly uneven contribution proportion, the lecturer may choose to adjust the mark of the lesser contributor downward by an amount derived from the individual contributions.

## Submission Details

* Visual Studio project files, media files, and completed planning document must be submitted via GitHub Classroom. Here is the link to the repository you will be using for submission - <https://classroom.github.com/a/5xAisr7->

## Passing Criteria

To pass this assessment, you must gain a mark of 50%.

## Authenticity

All parts of your submitted assessment must be completely your own work and any references must be cited appropriately using APA 6th edition style referencing.

## Policy on Submissions, Resubmission and Resits

The School process in relation to submissions, extensions, resubmissions and resits complies with Otago Polytechnic Policies. Students can view policies on the Otago Polytechnic Website located at [http://www.otagopolytechnic.ac.nz/.](http://www.otagopolytechnic.ac.nz/) Students may be requested to resubmit an assessment following a rework of part/s of the original assessment.

Resubmissions are completed within a short time frame (usually no more than 5 working days) and usually must be completed within the timing of the course to which the assessment relates. Resubmissions will be available to students who have made a genuine attempt at the first assessment opportunity. The maximum grade awarded for a resubmission will be C-.

## Extensions

Please familiarise yourself with the assessment due date. If you need an extension, please contact your Course Coordinator prior to the due date. If you require more than a week's extension, a medical certificate or support letter from you manager may be needed

# Assessment 1: Roguelike Assessment Rubric

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **10-9** | **8-7** | **6-5** | **4-0** |
| **Planning Document** | Planning document submitted before the due date. The provided set of questions are thoroughly answered in detail.  System design thoroughly planned and no changes to the submitted planning document. | Planning document submitted before the due date. The provided set of questions are mostly answered in detail.  System design mostly planned and a few changes to the submitted planning document. | Planning document submitted after the due date. The provided set of questions are answered in some detail.  System design planned and several changes to the submitted planning document. | Planning document submitted after the due date or not submitted. The provided set of questions are answered, though in minimal or no detail.  System design poorly planned or not planned. |
| **Code Commenting** | All header comments thoroughly explain the input, output, effect and computational logic of each class and method.  All inline comments thoroughly explain the logic of construct of each computational statement. | Most header comments explain the input, output, effect and computational logic of each class and method.  Most inline comments explain the logic of construct of each computational statement. | Some header comments explain the input, output, effect and computational logic of each class and method.  Some inline comments explain the logic of construct of each computational statement. | Minimal or no header comments explain the input, output, effect and computational logic of each class and method.  Minimal or no inline comments explain the logic of construct of each computational statement. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Code Elegance** | All class files contain no integer literals except for 0, 1 and 2.  Application demonstrates thorough elegance on all of the following:   * Correct use of intermediate variables, e.g., no method calls as arguments * Idiomatic use of control flow and data structures * Sufficient modularity, e.g., classes, methods have a single purpose * Efficient algorithmic approach | Most class files contain no integer literals except for 0, 1 and 2.  Application demonstrates clear elegance on most of the following:   * Correct use of intermediate variables, e.g., no method calls as arguments * Idiomatic use of control flow and data structures * Sufficient modularity, e.g., classes, methods have a single purpose * Efficient algorithmic approach | Some class files contain no integer literals except for 0, 1 and 2.  Application demonstrates elegance on some of the following:   * Correct use of intermediate variables, e.g., no method calls as arguments * Idiomatic use of control flow and data structures * Sufficient modularity, e.g., classes, methods have a single purpose * Efficient algorithmic approach | Class files contain frequent integer literals.  Application does not demonstrate elegance on any of the following:   * Correct use of intermediate variables, e.g., no method calls as arguments * Idiomatic use of control flow and data structures * Sufficient modularity, e.g., classes, methods have a single purpose * Efficient algorithmic approach |
| **OO Architecture** | All classes adhere to a general OO architecture, e.g., classes, methods, concise naming and methods assigned to the correct classes.  Inheritance fully and carefully implemented in classes, e.g., player sprite inherits from sprite.  Finite State Machine (FSM) implemented fully and stores three states and actions. | Most classes adhere to a general OO architecture, e.g., classes, methods, concise naming and methods assigned to the correct classes.  Inheritance mostly implemented in classes, e.g., most classes are deriving from base classes.  Finite State Machine (FSM) mostly implemented and stores two states and actions. | Some classes adhere to a general OO architecture, e.g., classes, methods, concise naming and methods assigned to the correct classes.  Some inheritance implemented in classes, e.g., some classes are deriving from base classes, though some are incorrectly implemented in the wrong classes.  Some Finite State Machine (FSM) implemented and stores one state and action. | Classes adhere to minimal or no general architecture, e.g., classes, methods, concise naming and methods assigned to the correct classes.  Minimal inheritance implemented or not attempted.  Minimal Finite State Machine (FSM) implemented or not attempted. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Functionality & Robustness** | Application opens in Visual Studio 2017 without errors and does not need to be modified to be run.  Application demonstrates thorough functionality & robustness on all the following:   * Displayed at the correct screen size of 1920x1080 * Dungeon represented as a tile map * Edge of the world has a dead zone which is ½ the dimension of the viewable area * Dungeon procedurally generated at each new level, e.g., multiple non-overlapping rooms, walls, corridors and portal tiles correctly placed * Fog of war reveals the dungeon as the player character progressively navigates through the dungeon * One or more player characters are controlled by user keyword * Two distinct animated enemies * Careful sprite and terrain collision detection, e.g., sprite to enemy, sprite to wall collision detection * Careful collision detection that affects the score and condition, e.g., sprite to coin, sprite to health potion * Working battle system, e.g., turn-based or/and to-the-death * Immediate gameplay feedback including battle system feedback, score, win and loss * One enemy that exhibits artificial intelligence behaviour. This may be implemented using trigonometry | Application does open in Visual Studio 2017, though needs to be modified to be run.  Application demonstrates most functionality & robustness on all the following:   * Displayed at the correct screen size of 1920x1080 * Dungeon represented as a tile map * Edge of the world has a dead zone which is ½ the dimension of the viewable area * Dungeon procedurally generated at each new level, e.g., multiple non-overlapping rooms, walls, corridors and portal tiles correctly placed * Fog of war reveals the dungeon as the player character progressively navigates through the dungeon * One or more player characters are controlled by user keyword * Two distinct animated enemies * Careful sprite and terrain collision detection, e.g., sprite to enemy, sprite to wall collision detection * Careful collision detection that affects the score and condition, e.g., sprite to coin, sprite to health potion * Working battle system, e.g., turn-based or/and to-the-death * Immediate gameplay feedback including battle system feedback, score, win and loss * One enemy that exhibits artificial intelligence behaviour. This may be implemented using trigonometry | Application needs to be modified to be open and run in Visual Studio 2017.  Application demonstrates some functionality & robustness on all the following:   * Displayed at the correct screen size of 1920x1080 * Dungeon represented as a tile map * Edge of the world has a dead zone which is ½ the dimension of the viewable area * Dungeon procedurally generated at each new level, e.g., multiple non-overlapping rooms, walls, corridors and portal tiles correctly placed * Fog of war reveals the dungeon as the player character progressively navigates through the dungeon * One or more player characters are controlled by user keyword * Two distinct animated enemies * Careful sprite and terrain collision detection, e.g., sprite to enemy, sprite to wall collision detection * Careful collision detection that affects the score and condition, e.g., sprite to coin, sprite to health potion * Working battle system, e.g., turn-based or/and to-the-death * Immediate gameplay feedback including battle system feedback, score, win and loss * One enemy that exhibits artificial intelligence behaviour. This may be implemented using trigonometry | Application cannot be opened in Visual Studio 2017 or application is empty.  Application does not demonstrate functionality & robustness on any of the following:   * Displayed at the correct screen size of 1920x1080 * Dungeon represented as a tile map * Edge of the world has a dead zone which is ½ the dimension of the viewable area * Dungeon procedurally generated at each new level, e.g., multiple non-overlapping rooms, walls, corridors and portal tiles correctly placed * Fog of war reveals the dungeon as the player character progressively navigates through the dungeon * One or more player characters are controlled by user keyword * Two distinct animated enemies * Careful sprite and terrain collision detection, e.g., sprite to enemy, sprite to wall collision detection * Careful collision detection that affects the score and condition, e.g., sprite to coin, sprite to health potion * Working battle system, e.g., turn-based or/and to-the-death * Immediate gameplay feedback including battle system feedback, score, win and loss * One enemy that exhibits artificial intelligence behaviour. This may be implemented using trigonometry |
| **Player Experience** | Highly attractive, with a coherent graphical theme and style  Application is highly appealing and has an engaging game play experience | Mostly attractive, with a coherent graphical theme and style  Application is mostly appealing and has an engaging game play experience | Somewhat attractive, with a graphical theme or style  Application is somewhat appealing and has an engaging game play experience | Minimal attempt or no coherent graphical and style  Application is not appealing or engaging, e.g., no game play |

IN628 Programming 4

Semester 2, 2019

# Assessment 1: Roguelike Planning Document

**Due Date:** Wednesday, 25th September, 5.00 pm

Fill in the following questions in detail before you begin to code your game. Please use a digital copy of the document, not a hard copy. For each question, justify your answer. If during implementation you make any changes to your originally articulated plan, amend the document, specifying the changes, and explaining your rationale.

Submit the completed document with your source code via **GitHub Classroom**. To receive full credit, the completed document **must** be uploaded to the repository before any code files.

|  |  |
| --- | --- |
| **1** | Are your player, items and enemies the same class, different classes in the same family,  or completely different classes? |
| **2** | What logic will you put into your Form class? What logic will you put into your Game  Manager class? |
| **3** | What class (es) do you need to implement the dungeon? Briefly explain the job of each  class, list the data members it must hold, and the methods it must expose. How do the Dungeon and the TileMap communicate? |
| **4** | What data structure(s) do you need to hold collections of enemies and items? |
| **5** | Does the dungeon need pointers to its sprites? Why or why not? |
| **6** | Does the sprite class need a pointer to its dungeon? Why or why not? |
| **7** | What enumeration types (if any) do you need? |
| **8** | Does the player sprite need access to the collection(s) of enemy sprites? |
| **9** | What class is responsible for creating the collections of enemies and items? |
| **10** | If you are using an FSM, what class calls the FSM methods of the sprites? |
| **11** | At each game cycle, you need to perform collision detection between the player character and each enemy and item in the dungeon. What class or classes hold a method to compare the areas of two entities to check for collision? What is the function header of this method? What other classes are involved in the collision detection logic? |
| **12** | Describe the AI you are going to include.   1. Describe the behaviour 2. Describe the implementation logic |
| **13** | Describe the trigonometry you are going to include (if not already contained in the AI  from #12 above). |
| **14** | Describe in detail, the logic of your battle algorithm and computations. |
| **15** | Sketch the screen layout with controls that you will use to provide feedback during  battle. |

**Marking Cover Sheet**



Assessment 1: Roguelike

**IN628 Programming 4**

Level 6, Credits 15

**Bachelor of Information Technology**

Name: Date:

Learner ID:

Assessor’s Name:

Assessor’s Signature:

|  |  |  |  |
| --- | --- | --- | --- |
| **Criteria** | **Out Of** | **Weighting** | **Final Result** |
| Planning Document | **10** | **10** |  |
| Code Commenting | **10** | **10** |  |
| Code Elegance | **10** | **25** |  |
| OO Architecture | **10** | **20** |  |
| Functionality & Robustness | **10** | **25** |  |
| Player Experience | **10** | **10** |  |
| **Final Result** | | | **/100** |
| This assessment is worth 45% of the final mark for the Programming 4 course. | | | |