[CAB202 AMS](https://sefams01.qut.edu.au/CAB202/)

* [Home](https://sefams01.qut.edu.au/CAB202/)

Hello, QUTAD\n9808281 !

CAB202 Assignment 1, Exercise 1: Assignment 1

**Requirements:**

CAB202 Assignment 1

Due Date: 23:59:59 (AEST) Monday 10th of September 2018

Marks: 30 (30% of your final mark)

For this assignment, you will be writing a computer program that emulates a player moving through a series of static and moving blocks. The aim of the player is to move/jump between blocks while gaining points and lives.

Assignment Specifications

This assignment is divided into different sections, however you may attempt the sections in any order you see fit. You may attempt other sections even if you have not fully completed the Basic Functionality.

You will submit documentation as per the instructions given below and you will be marked on a combination of your documentation and your program. You will need to do both well to get the full marks. If you submit code without the specified documentation you will be marked down, conversely, if you submit documentation for aspects you have not implemented or documentation that does not accurately reflect your program, you will be marked down.

If your program crashes, you will be marked for the sections your marker was able to test before the program crashed. Please test your code regularly to minimise your chance of this happening. You can make multiple submissions to AMS and you are encouraged to do so.

Basic Functionality (15 Marks)

1. ~~The display screen is 5 characters high, located at the top of the screen, is entirely visible at all times (i.e. never occluded in any way) and displays:~~
   1. ~~Student number,~~
   2. ~~Lives remaining,~~
   3. ~~Playing time in minutes and seconds format as mm:ss,~~
   4. ~~Score,~~
   5. ~~Border that separates the display screen from the game area.~~
2. ~~The player initially appears on a 'starting block' in the top row. The player’s sprite is at least 3 characters high and 3 characters wide.~~
3. ~~All blocks are:~~
   1. ~~2 characters high,~~
   2. ~~At least 7 characters wide (unless Advanced Functionality is implemented) and are clearly distinguished from each other (through characters in bitmap or spacing between sprites),~~
   3. ~~Always at least~~**~~sprite height + 2~~**~~characters vertically away from other blocks.~~
4. ~~Columns are calculated as~~**~~visible screen width / max block width~~**~~+ horizontal spacing (if used to distinguish between sprites) and there are:~~
   1. ~~The number of blocks on screen is proportional to the number of columns, to a maximum of 160 safe blocks and 40 forbidden blocks,~~
   2. ~~At least 1 safe block per column denoted by '=' on the screen at all times,~~
   3. ~~At least 2 forbidden block on the screen denoted by 'x' on the screen at all times.~~
5. ~~Blocks have no consistent observable pattern and each block:~~
   1. ~~Appears in a randomly selected row,~~
   2. ~~Appears in a randomly selected column (after there is 1 safe block per column),~~
   3. ~~Does not overlap other blocks or the display screen.~~
6. ~~Player movement behaves such that:~~
   1. ~~Pressing the ‘a’ or ‘d’ key moves the character left and right respectively,~~
   2. ~~The player can not overlap the display screen.~~
   3. ~~When off a block, the player free falls downwards at an appropriate speed,~~
   4. ~~No lateral motion occurs during downward motion (unless Advanced Functionality is implemented).~~
7. ~~There is a treasure sprite that:~~
   1. ~~Is no larger than the player's sprite and has a clearly distinct image,~~
   2. ~~Spawns at the bottom of the screen and moves back and forward along the bottom, changing horizontal direction when it reaches the edges of the screen,~~
   3. ~~Stops moving when 't' is pressed and starts moving again if 't' is pressed again (i.e. motion toggles on and off when 't' is pressed),~~
   4. ~~Disappears when the player collides with it and gives the player~~**~~2~~**~~more lives and returns the player to the 'starting block'.~~
8. ~~The following game mechanics are present:~~
   1. ~~The player starts with~~**~~10~~**~~lives.~~
   2. ~~A point is scored every time the player lands on a safe block.~~
   3. ~~The player dies when they fall off the bottom of the screen or hit a forbidden block. On death, the player starts from the ‘starting block’ (unless Advanced Functionality is implemented).~~
   4. ~~When the player loses all their lives, the game over screen is displayed which displays a game over message, total score, and game play time.~~
   5. ~~The game over screen allows the player to restart by pressing ‘r’ (score, lives, time, and screen all reset) or quit with the ‘q’ key which ends the game gracefully.~~

Advanced Functionality (7 Marks)

1. Each block has a random width between 5 and 10 and:
   1. The top and bottom rows of blocks remain stationery, but the blocks in other rows move at appropriate speeds (different rows may have different speeds),
   2. Blocks in the same row move at the same speed and direction,
   3. Blocks reappear on the other side of the screen as they go off the screen,
   4. Each row of blocks moves in opposite directions or speeds to the row directly above (e.g. if row 2 moves left, row 3 could move right),
2. Player motion is controlled as follows:
   1. An 'a' or 'd' key press gives the player a horizontal velocity so that if 'a' or 'd' is pressed on a moving block, the player moves along the block accordingly,
   2. ~~When on a block, a 'w' key press gives the player a vertical velocity upwards (pressing 'w' again does not do anything until player lands again),~~
   3. When the player lands on a block their vertical velocity changes to zero and they adopt the same horizontal velocity as the block,
   4. If the player is on a block and any part of the block goes off screen, the player loses a life and respawns.
3. A constant vertical acceleration (i.e. gravity) acts downwards at all times, generating a reasonably realistic ballistic motion effect when the player is in flight after stepping or jumping from a block. Gravity generates the following visually observable behaviour:
   1. ~~When on a block, the effects of gravity are eliminated – the player does not fall through the block,~~
   2. ~~When jumping, the player’s vertical velocity is slowest at the peak jump height,~~
   3. When jumping with a horizontal velocity, the player traces a parabolic path. The horizontal velocity is constant, while the vertical velocity increases at a constant rate with passage of time,
   4. When falling with a non-zero horizontal velocity, the player also traces a parabolic path but starts with zero vertical velocity,
   5. When hitting the side of a block, the player’s horizontal velocity becomes zero.
4. Player randomly respawns on any of the top row safe blocks after dying (doesn't just reappear on the 'starting block').

Advanced Graphics (4 Marks)

1. ~~The treasure has some sort of animation.~~
2. ~~The player's image changes when:~~
   1. ~~The player falls or when they jump,~~
   2. ~~The the player is moving left,~~
   3. ~~The the player is moving right,~~
   4. ~~The player is stationery or only moving with the blocks.~~
3. The player has some sort of animation every time the player respawns.

Advanced Coding Practices (4 marks)

1. ~~Your program has minimal code duplication and is cleanly structured for a human reader.~~
2. ~~You have used arrays or pointers where appropriate.~~
3. ~~Your program is composed of functions. A function should only perform one task and not exceed 20 lines of code.~~
4. ~~You have used parameters where appropriate and minimised the number of global variables.~~

Documentation

**You must document all the functionality you have implemented for full marks in each criteria.** Your documentation must be clearly written, formatted appropriately, follow the structure in the template, and include:

* A cover page with your full name, student number and assignment date,
* A contents page,
* A list of all the features outlined above and which ones you have attempted as per **Table 1**,
* A heading, linked to the contents page, for each section,
* Test plans for all functionality implemented from the Basic Functionality and Advanced Functionality sections as per **Table 2**. You do not need to write a test plan for items you have not implemented,
* A brief description of **what** you have implemented from the Advanced Graphics section and **how** it can be observed,
* A written explanation of **why** and **how** you implemented what you have from the Advanced Coding Practices.

Use the templates below, and/or the Word document provided on Blackboard, for your documentation.

|  |  |  |
| --- | --- | --- |
| Table 1 | | |
| **Assignment Implementation Summary** | | |
| **Item Number** | **Item Description** | **Implementation Level** |
| 1 | Display screen | Partially implemented |
| 2 | Player size and initial position | Fully implemented |
| 3 | Block size and spacing | Not implemented |
| 4 | Blocks are randomly positioned. |  |
| 5 | Columns and block types |  |
| 6 | Player movement |  |
| 7 | Treasure |  |
| 8 | Basic game mechanics |  |
| 9 | Block width and movement |  |
| 10 | Advanced player movement |  |
| 11 | Ballistic motion |  |
| 12 | Player random respawning |  |
| 13 | Treasure animation |  |
| 14 | Player movement animation |  |
| 15 | Player respawn animation |  |
| 16 | Minimal duplication |  |
| 17 | Arrays or pointers |  |
| 18 | Functions |  |
| 19 | Parameters |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Table 2: Test Plan Example | | | |
| **Display Screen** | | | |
|  |  |  |  |
| **Test of Specific Functionality** | **Test Setup and Actions** | **Expected Result** | **Actual Result** |
| Display screen is 5 pixels high, at top of screen and is entirely visible at all times. | Run game and press 'w' for player to jump. | Player and blocks should never cross the display screen. | Player can jump into display screen. |
| Lives remaining are displayed in the status display area. | Run game and die. | Lives should be displayed towards the right of the display screen area and decrease on death. | As expected. |
| Student number is displayed in the status display area. | Run game | Student number should be displayed in the left of the display screen. | As expected. |

Submission Information

Important!

1. The assignment will be graded by strict reference to the criteria listed below.
2. Assessment is evidence-based. You must submit a written report via Blackboard which provides evidence of completion, supported by working source code submitted via AMS. Both items must be submitted by the due date. Absence of either or both will result in a score of 0.
3. **This is not a group assignment.** While we encourage you to discuss the assignment and brain-storm with your associates, you must ensure that your submission is your own individual work.

*Share ideas, not code!*

1. A high standard of academic integrity is required. Breaches of academic integrity, including plagiarism or any other action taken to subvert, circumvent, or distort the assessment process, will not be tolerated. QUT policy regarding academic conduct is available in the [QUT MOPP Section C/5.3 Academic Integrity](http://www.mopp.qut.edu.au/C/C_05_03.jsp). In particular, under the provisions of [MOPP statement C/5.3.7](http://www.mopp.qut.edu.au/C/C_05_03.jsp#C_05_03.07.mdoc), Part (e), we reserve the right to require you to authenticate your learning. You may be required to show evidence of materials used in the production of the assignment such as notes, drafts, sketches or historical backups. You may also be required to undertake a viva or complete a supervised practical exercise.
2. Use of any third-party code library (other than the ZDK and those supplied with a standard distribution of GCC and ncurses) is strictly prohibited. Use of code downloaded from the internet without correct attribution to the original author(s) is strictly prohibited. Subcontracting, outsourcing, off-shoring, purchasing, borrowing, stealing, copying, or obtaining source code or report content by any means other than through an act of original creation by yourself, is strictly prohibited.
3. Abundant code samples, demonstrations, and exercises have been made available to support your effort toward this programming task. Written permission must be obtained from the Unit Coordinator if you want to use technology other than the ZDK to implement your game. Permission will only be granted if there are compelling special circumstances that make it impossible for you to use the ZDK. Without this permission, a game implemented with some other graphical framework will receive a score of 0. Direct use of the ncurses library to render graphics or text is expressly prohibited.

Deliverables

1. Report, submitted via Blackboard

Submit a succinct and well-presented written report which accurately describes your program and how you have tested it. You may refer to [QUT Cite Write](https://www.citewrite.qut.edu.au/write/report.jsp) for information on the general guidelines for writing a report. For convenience we provide a word template.

The report is your primary assessable item.

The report is the medium through with you will convince your tutor that the program you have written is in fact a solution to the task that has been set. It **is not** a copy of the source code of your program. It needs to contain enough information to convince an intelligent reader (who may not necessarily have a background in software engineering) that you have done what you claim to have done. The reader should be able to see easily in the report how you have tested the respective features, and whether they work as claimed.

The report should contain one section with the summary of the features that you implement and one section with the test plan and results. Refer to the report structure template in the specification in AMS and the Word document on Blackboard.

Implementation Summary.

* + - Provide indication whether the requested feature was fully, partially or not implemented.
    - *Do not include a copy of the source code in your report.*

Test plan and results.

* + - The most appropriate format for this is the template provided. Do not submit screenshots of the program or code.
    - Some features may require several tests.

Your mark for each feature will be awarded based what you document that you have done. Make sure you cover all relevant features in your documentation and implementation, and supply evidence of comprehensive testing to back it up.

1. Source code submitted via AMS

Submit full source code for your implementation into AMS using this form.

A working implementation of your program which can be executed by your tutor running the CAB202 Portable Cygwin software is required. Therefore, your program must compile in AMS and the resulting executable program must work as stated in your report.