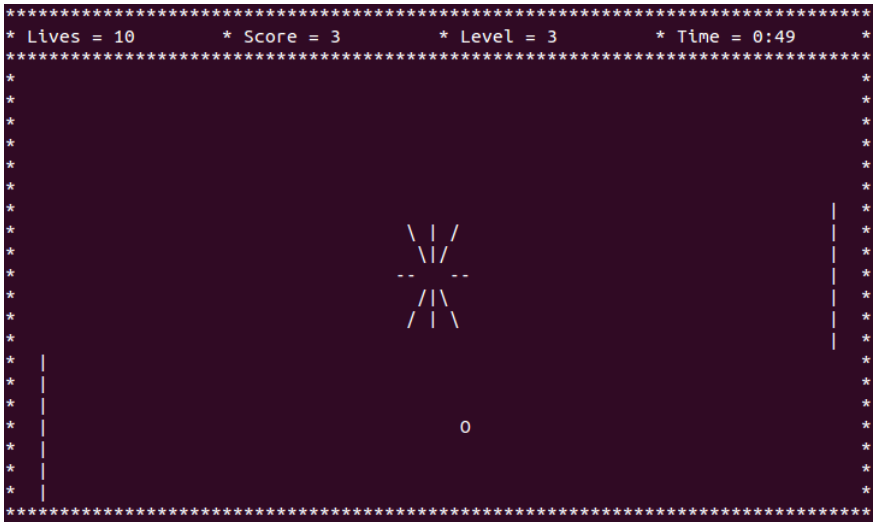
CAB202 Assignment 1: Pong

Due Date: **Friday 9 September, 2016 @ 11:59:59 PM**
Submission: via AMS
Marks: 30 (30% of your final mark)

Overview

The assignment is a variation on the Pong theme. The player controls a game paddle positioned near one edge of the terminal window which moves vertically but not horizontally. A ball flies within the playing area of the game, bouncing off floor and ceiling and one of the walls, but passing through the remaining wall. Released along with this document is the *CAB202_Assignment_1_marksheet.xls* document, which is a checklist of all the required functionality for this assignment.



Requirements

Your task is to implement the Pong game. Throughout the semester, you have been provided with a number of examples of skeleton code for implementing games with the CAB202 ZDK library. You have also been shown a demo version of the game (*see the Topic 3 lecture*), which implements some of the requirements for the assignment. You are not required to use the ZDK library, but you are strongly encouraged to do so.

WARNING: QUT takes plagiarism very seriously. If your assignment has material which has been copied from other class members, previous assignments, or code from any other source you will be penalised severely. If you cannot explain exactly what your code is doing, it is clearly not your own work, and you should not be submitting it as such!

Game Specification

The game has four levels which entail qualitatively distinct modes of play. Pressing the key labelled 'L' at any time during play advances the level by 1. After Level 4, the level cycles back to Level 1. There is no other way to progress between levels. For all levels, the following basic game functionalities need to be implemented in order for the game to work properly:

1. The game must adapt itself to occupy the full expanse of the terminal window at the time the program starts. **[1 mark]**

    a. You may assume that the width of the terminal window is at least 60 units and the height of the terminal window is at least 10 units.

    b. You may assume that the user will not resize the window while the game is running.

2. Help screen is displayed at commencement of game, prior to moving on to game itself. **[3 Marks]**

    a. Help screen must show the name and student number, plus a list of keyboard commands required to play the game.

    b. Help screen exits when user "Presses any key".

    c. Also displayed when user presses 'h'.

    d. Also displayed after the game over dialogue, if user has selected “Play again”, before moving on to restart the game.

    e. While the help screen is displayed, all other game dynamics are frozen, including the elapsed game time.

3. After help screen at the start (or restart) of game, borders and game status display is shown. The status display shows: **[2 Marks]**

    a. Lives, initially 10 (or 3, or something reasonable).

    b. Score, initially 0.

    c. Level, initially 1.

    d. Elapsed game time, measured in minutes and seconds, initially 00:00.

4. Ball is spawned at the beginning of play, and re-spawns each time it leaves the playing area, unless the game is over. Ball spawns in two stages. **[3 Marks]**

    a. Countdown (3, 2, 1...) for 0.3 seconds.

    b. During the countdown all other game dynamics are frozen, including the elapsed game time.

    c. Launch from centre of screen, travelling towards the player’s end of the game in random direction within ±45° of horizontal. The speed of the ball at spawning should be set to some fixed value that yields a playable game, neither too slow nor impossibly fast.

5. Game border and info panel during play **[4 Marks]**

    a. Lives are reduced by 1 every time the ball passes out of the play area at the player’s end.

    b. Score increments by 1 every time the player’s paddle strikes the ball (or vice versa).

    c. Current level accurately reflects state of game.

    d. Ball, paddles and other sprites never pass onto or over the boundaries.

e. Elapsed game time is updated in a timely manner once per second.

f. Elapsed game time records time spent in game. While any dialog (help screen, spawn countdown, or game over) is being displayed, the clock is stopped. After a dialog is cleared, elapsed game time resumes from the value it had at the time the dialog opened. That is, the clock is stopped while the game is paused.

6. The player's game paddle: **[3 Marks]**

a. The player controls a paddle which is constrained to move vertically at one side of the playing area.

b. The paddle is placed so that there are exactly two columns of empty space between the paddle and the nearest vertical wall of the playing area.

c. The paddle must never protrude beyond the playing area, overlap the boundary, be obscured (in part or whole) by the boundary, or in any other way not fit properly inside the playing area.

d. The paddle must have a width of 1 screen unit and the height must be the minimum of the two values listed below:

1. 7 screen units; or

2. $(h - h_s - 1)/2$ screen units, where h is the height of the screen and h_s is the height of the status display.

When the ball hits the player's paddle, or the computer-controlled paddle (when in play, see Level 2) **(NOTE: This section outlines the physics required for criteria on row 42 of the marking spreadsheet)**:

a. If the point of contact is not one of the endpoints of the paddle, the ball is reflected horizontally.

b. If the point of contact is the top endpoint of the paddle:

i. If ball is moving downward, AND distance from top of paddle to ceiling is greater than height of the ball – ball is reflected *vertically*, that is, it bounces off the top of the paddle and continues moving with its current horizontal velocity. If necessary, the ball may be shifted to the screen location immediately above the paddle to ensure it does not get stuck.

ii. Otherwise – ball is reflected *horizontally*, that is, it bounces off the side of the paddle, reversing its horizontal velocity.

c. If the point of contact is the bottom endpoint of the paddle:

i. If ball is moving upward, AND distance from bottom of paddle to floor is greater than height of the ball – ball is reflected *vertically*, that is, it bounces off the bottom of the paddle and continues moving with its current horizontal velocity. If necessary, the ball may be shifted to the screen location immediately below the paddle to ensure it does not get stuck.

ii. Otherwise – ball is reflected *horizontally*, that is, it bounces off the side of the paddle, reversing its horizontal velocity.

d. We are aware that this may occasionally produce slightly unrealistic physics: this is by design.

Level 1: Handball [16 marks total]

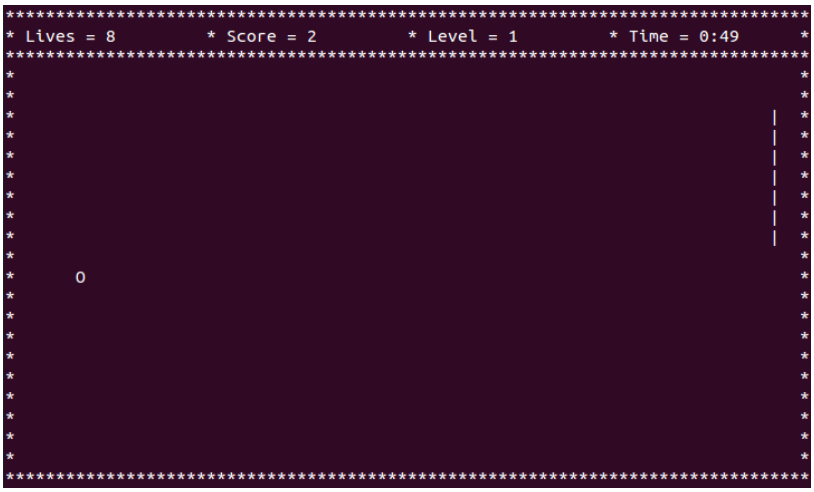


Figure 1: Level 1 - Handball

On this level, there is a single game paddle controlled by the player. The wall at the opposite end of the playing area is just that: a wall. The ball should bounce off the wall and return. The wall is not a paddle, so special rules about paddle collision listed below do not apply when the ball hits a wall. If the computer-controlled paddle was present on the previous level (due to cycling) it is removed from play.

Level 2: RoboPong [4 marks total]

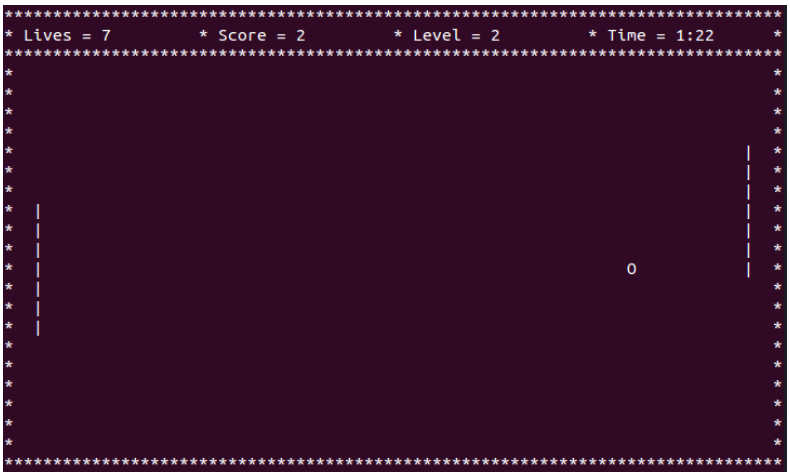


Figure 2: Level 2 - RoboPong

In level two, a computer-controlled paddle is introduced at the end of the playing area opposite the player's paddle, separated by two blank columns from the boundary. The paddle, which is the same size and shape as the player's paddle, moves so that it always blocks the path between the ball and the wall. At all times the computer-controlled paddle remains entirely within the bounds of the playing area. The computer-controlled paddle is retained when leaving this level.

Level 3: Singularity [4 marks total]

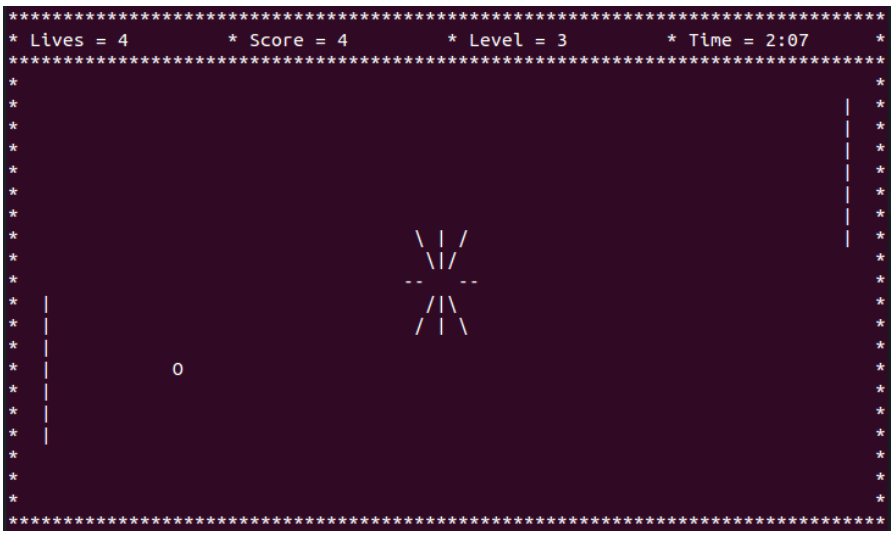


Figure 3: Level 3 - Singularity

A gravitational anomaly (which may be depicted by a suitable character such as a star) appears at the centre of the playing area 5 seconds after spawning or entering the level. The anomaly exerts a visible gravitational force on the ball, causing the ball to accelerate toward the centre of the playing area. Regardless of the acceleration, the maximum velocity of the ball must never exceed the speed of light, which in the game universe is defined to be one screen unit per time-step.

The fundamental forces of nature must be balanced such that the ball cannot become trapped in orbit around the singularity. Passing through the singularity has no effect on the ball other than acceleration. The anomaly disappears and becomes inactive when the ball spawns, reappearing 5 seconds later as noted above. The anomaly is removed when leaving the level.

Level 4: Rails [6 marks total]

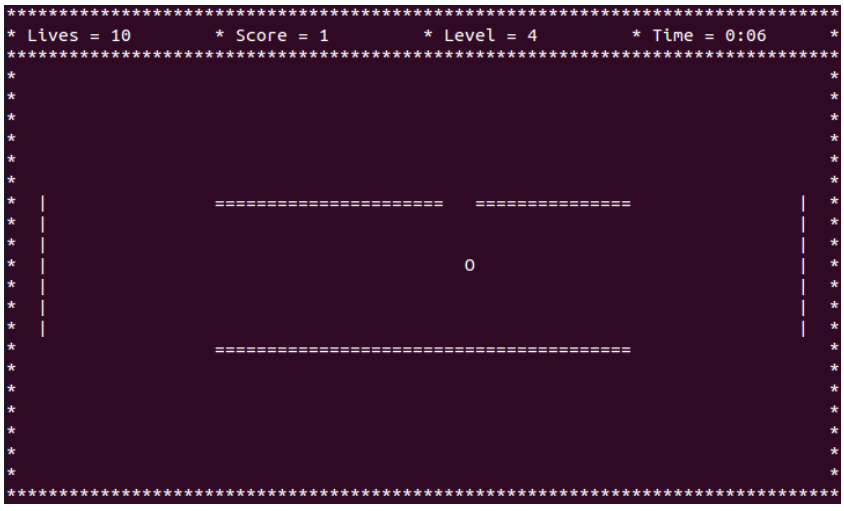


Figure 4: Level 4 - Rails

Two horizontal rails appear – the rails have thickness of one screen unit, and length equal to half the screen width. The rails are horizontally centred within the playing area and placed vertically so that they split the playing area as evenly as possible into three horizontal strips. When the ball hits either rail, it bounces as it would bounce off the floor or ceiling. However, each time the ball hits a rail it punches a hole in the rail, knocking out the unit of the rail that it hit, and one unit either side. Subsequently, the ball will pass through any holes in the rail without interacting with the rail.

Remarks

- You may choose to implement any of the functionality described in this specification sheet in any order (i.e.: You do not have to finish level 2 to implement features for level 3) ***so long as the functionality is implemented in the correct level.***

- Absolutely no extra marks will be given for implementing functionality outside of the features defined in this specification sheet. In order to receive marks for the functionality implemented, it must be easy for the marker to demonstrate it. In other words, in order for your assignment to get the best marks possible ***it must be easily playable***.

- Player may be at either end at discretion of implementer.

- Paddle may be controlled by any keys at discretion of implementer (just make sure to properly describe the game mechanics in the help menu)

- At all times, movement of the ball and paddles must be smooth. For the purpose of this assignment, smooth means that the ball is never observed to move more than one screen unit in x or y direction between successive frames.

- When moving between levels the game may continue to function as it was apart from changes required to introduce and/or remove assets as appropriate for the level. However it is also OK to re-spawn. Life counter and score should be preserved.

Marking

The assignment will be out of 30 marks and will be worth 30% of your total grade in this subject. The breakdown of marks is outlined in the task specification above. The following should be noted about marking:

- **If your code does not compile, you will get 0 marks for the entire assignment.**
- If segmentation faults occur, you will receive marks for what was displayed but lose marks for the segmentation fault. No more of your assignment will be marked. We will not debug your code to make it compile or run.
- Your game must be easily playable. If timings, settings, or controls are set in a manner that makes it difficult to play (e.g. not using the key inputs specified, ridiculously fast movement, etc.), you **will receive 0 for the assignment**.
- Mark penalties will be applied if the code exhibits general defects or undesirable behaviour. This includes, but is not limited to, errors in object motion, such as objects jumping more than one unit per frame, or sprites overlapping when they are not required to by specification.
- Your marker will not spend longer marking your assignment if they cannot easily demonstrate a specific feature. It is **your responsibility** to demonstrate that a feature works, not the marker's to prove it is there.

Submission

Your assignment is due on **Friday 9 September, 2016 @ 11:59:59 PM**. Submission will be online through the AMS used in the tutorials. You must submit through the submission page (currently not available), and follow all of the instructions. When submitting to the AMS, it is your responsibility to make sure that your assignment compiled correctly. The AMS will compile your code and return any errors that occur. As mentioned above, if you do not submit a compiled assignment, you will receive 0 marks. You will

have unlimited submissions of the assignment, so there are no excuses for not resolving any compilation issues.