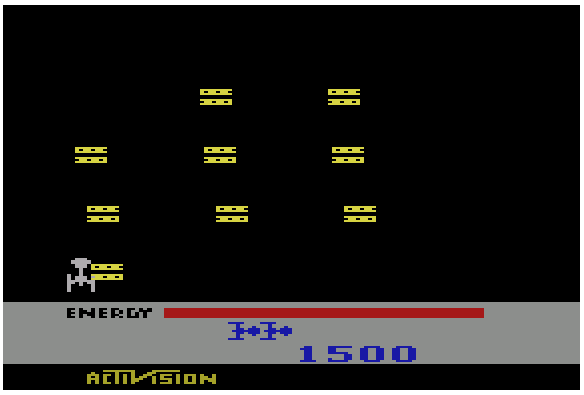
CAB202 Assignment 1: Megamaniac

Due Date: 11:59PM Sunday 6 September 2015
Submission: to AMS
Marks: 30 (30% of your final mark)

For this assignment, you will be writing your own version of a game called Megamaniac. This game is based on the arcade games Space Invaders and Megamania (see below for a screenshot and https://www.youtube.com/watch?v=BhK8h6iBr-A for example gameplay).



Requirements

Your task is to implement the Megamaniac 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 week 4 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!

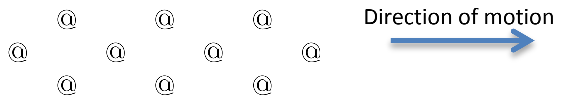
Game Specification

The basic aim of the game is to outlast all of the enemy aliens. Aliens appear on the screen moving in a set pattern (this pattern depends on what level is currently activated). The player starts with 3 lives and there are 10 enemy aliens alive. The player can shoot a single bullet at a time to try and destroy the aliens. The score in the game should increase by 30 points each time the bullet hits an alien. When all aliens are cleared from the screen, the player is awarded 500 points and every alien respawns (the game is never won). A life is lost when a bomb dropped by aliens hits your spacecraft. When the player has no lives left, a Game Over screen should be shown, and the Player should have the option to restart or quit. If you have completed more than level 1, you must be able to toggle between levels in game with the 'l' key to receive any marks for the extra levels.

Level 1: Basic functionality [15 marks total]

All versions require the following basic functionality (you will not be eligible for any more than 15 marks unless everything in this section is completed):

- The game must start immediately.
- A group of aliens should be generated and they should start moving from left to right in the top half of the screen.
- Motion is cyclic. This means that once each alien disappears from the right of the screen, the same alien should appear again from the left and continue its movement.
- The game should start with 10 aliens and they should each be represented by a single '@' character.
- The shape of the group of aliens must look like below:



- A spacecraft (the player) must be at the bottom of the screen (1 character above the score panel) and move left and right on 'a' and 'd' key presses respectively.
- The spacecraft should shoot a bullet towards the aliens when the 's' key is pressed.
- Every time a bullet from the spacecraft hits an alien, the alien should disappear and the player should get 30 points.
- Every 3 seconds a bullet should be dropped from an alien within the group (the alien that drops the bullet should change with each bullet).
- No more than four bullets from the aliens should be visible at one time.
- Every time a dropped bullet from the group of aliens or an individual alien hits the spacecraft, the player loses 1 life. The player starts with 3 lives.
- An information panel across the bottom of the screen should display the score, lives, your student name and number, and the current level being played.
- The score and lives should both be updated with the corresponding events in the gameplay.
- When every alien has been destroyed, the player should receive 500 bonus points and all of the aliens should respawn. The lives must **not** reset.
- A game over screen should be shown when all lives have been lost, giving the player the option to quit ('q' key) or restart ('r' key).
- The player must be able to quit ('q' key) and restart ('r' key) at any point in the game.

Level 2: Harmonic motion [2 marks total]

The alien behaviour is the same as level 1, but the aliens now have two directions of motion. You cannot receive any marks for this unless level 1 is completed. The behaviour must be as follows:

- Horizontal motion must be from left to right, wrapping at the right edge.
- Vertical motion must be up and down in an oscillatory motion (like a 'sin' or 'cos' wave).
- The aliens' average vertical position must not change (i.e. the aliens should never reach the ship).

Level 3: Falling motion [2 marks total]

The alien behaviour is the same as level 2, but the vertical motion is now from top to bottom. You cannot receive any marks for this unless levels 1 and 2 are completed. The behaviour must be as follows:

- Horizontal motion must be from left to right, wrapping at the right edge.

- Vertical motion must fall from the top of the screen to the bottom, wrapping at the bottom (i.e. top of score panel).
- If an alien hits the player's ship, the player must lose a life and the alien also must disappear.

Level 4: Drunken motion [3 marks total]

The alien behaviour is the same as level 3, but the horizontal motion is now a random displacement for each individual alien. You cannot receive any marks for this unless levels 1, 2, and 3 are completed. If you do not correctly use random number generation in your alien motion then you will not receive any marks. The random number generator must be seeded differently every time your program is run. The behaviour must be as follows:

- Each alien must have its own individual horizontal motion. This motion must be controlled by a sequence of randomly generated numbers.
- No alien can ever move into a position that is directly adjacent to another alien. As a result, the random number generation must be biased in a way that allows the aliens to expand and spread out (i.e. if they are stuck in a 'left and right jittering motion' you will receive 0 marks).
- Vertical motion must fall from the top of the screen to the bottom, wrapping at the bottom (i.e. top of score panel).

Level 5: Aggressive motion [5 marks total]

The final level has the same motion behaviours as level 3 (left to right, top to bottom, and wrapping back around). You cannot receive any marks for this unless levels 1, 2, 3, and 4 are completed. The behaviour must be as follows:

- Horizontal motion must be from left to right, wrapping at the right edge.
- Vertical motion must fall from the top of the screen to the bottom, wrapping at the bottom (i.e. top of score panel).
- After a random delay (with an average of 3 seconds), a single alien amongst the group must change behaviour. It should do the following:
    1) Travel a path out of the group of aliens. It cannot collide with any other alien.
    2) Once the alien reaches the edge of the group, it should travel a parabolic path from its current position to the player's current position. The horizontal displacement in the parabola should be randomly generated.
    3) Once the alien travels off screen, it should 'magically' respawn at its original position within the alien group (i.e. not where it started its movement, but where it would be if it had stayed within the group).
- The player must be able to shoot bullets that curve to the left and right with the 'z' and 'c' keys respectively. For full marks, the amount of curve in the bullet should be decided by how long the key is held down for (longer hold equals a more severe curve).

Bonus: Game runs on any screen dimensions [3 marks total]

Whenever your program is run, the game must fill the terminal it is running in. You can receive part marks for this feature if you have attempted any of levels 2-5. You can only receive full marks if every level in the game is resizable. It is safe to assume that the game will not be run at a size that doesn't allow you to fit all of the features from Level 1. Your game **does not** have to dynamically resize during the middle of a game (the libraries do not facilitate this behaviour). The following are examples of what size independent behaviour should occur:

- The score panel should be dynamically aligned (i.e. not statically). Anything that is right aligned, should always align on the right edge, centre aligned always in the centre, etc. You would **not** receive any marks if everything is simply left aligned (i.e. a static placement).
- The limits in which the ships can travel (both human and alien) should always be the edge of the screen no matter what the size. When ships go off the right edge they must **immediately** reappear on the left edge (i.e. not travelling in off screen space).
- The limits in which the bullets can travel (both human and alien) should always be the top of the screen and top edge of the score panel. The human must be immediately able to fire another bullet when it goes off screen (i.e. no travelling off the top of the screen).
- The parabolic limits in level 5 must all relate to the size of the screen (e.g. your horizontal displacement must be a percent of the screen rather than absolute value).

Marking

The assignment will be out of 30 marks and will be worth 30% of your total grade in this subject. The CRA explains the breakdown of marks. 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**.
- 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 Sunday September 6th 2015 at 11:59pm. Submission will be online through the AMS used in the tutorials. Submission will be in the form of a ***.zip** file containing all files used for your solution. Your submission should include a **README.txt** file indicating which options have been implemented. A template, and explicit instructions for how you should submit, will be provided closer to the due date.

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.