CAB202 Assignment 2: Alien Advance

Due Date: **26/05/2017 @ 11:59:59 PM**
Submission: to AMS
Marks: 40 (40% of your final mark)
The assignment will be marked in your registered tutorial during Week 13.

Introduction

CAB202 Assignment two requires you to design and implement a new game for the QUT Teensy device: Alien Advance. In this game, you must control the world's most advanced spacecraft in a deadly fight to save earth from the menacing alien forces. These ruthless alien attackers will stop at nothing to destroy humanity in their quest to cleanse the universe. Armed with their advanced alien technology, and backed up by their alien mothership, these foes present humanity's greatest challenge in their quest to solve the mysteries of the universe.

The following general constraints apply:

1. This is not a group assignment. While we encourage you to discuss and brain-storm with your associates, you must ensure that your submission is your own individual work.

Share ideas, not code.

2. 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. In particular, under the provisions of MOPP statement C/5.3.7, 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.

3. Abundant code samples, demonstrations, and exercises have been made available to support your effort toward this programming task. Therefore, written permission must be obtained from a Unit Coordinator if you want to use technology other than the teensy library to implement your game. Permission will only be granted if there are compelling special circumstances that make it impossible for you to use the teensy library. Without this permission, a game implemented with some other graphical framework will receive a score of 0.

Requirements

Your task is to implement the Alien Advance game. Throughout the semester, you have been provided with a number of examples of skeleton code for implementing games with the CAB202 teensy library. You have also been shown a demo version of the game (*see Blackboard -> Assessment*), which implements all of the requirements for the assignment.

Game Specification

The game consists of a single level, which contains several features as listed below. Your score will be determined by the completeness and quality of your implementation of these features. Part-marks will be awarded wherever possible if there are signs that a feature has been attempted, even if functionality is not perfect. An effort has been made to organise the feature list in a progression with technically simpler items appearing ahead of those that require more sophisticated programming. At the same time, the features are laid out in approximately the order they will appear in a working game. You are encouraged to survey the entire feature list and implement the features in the order that seems best for you.

The game consists of a single mode of play where the player must dodge enemy attacks and attempt to destroy the alien spacecraft with its projectile weapon systems. The following list details the specifications for all features required for this assignment:

1. Introduction Menu **[1.5 marks]**
    a. A screen is displayed when the teensy is powered on (the introduction menu). The menu shall display:
        i. Game Title
        ii. Your Name
        iii. Student Number
        iv. “Press a button to continue…” message
    b. The information should be displayed in the centre of the Dialog, and should be laid out in a tidy and readable manner.
    c. The introduction menu shall remain on the screen until the player presses either SW1 or SW2.
    d. Once the user has pressed a button, the screen shall display a countdown (3-2-1) in 300ms intervals.

2. Border, Status Display, and Playfield **[1 mark]**
    a. During normal play mode – that is, at any time when the Game Over Dialog (see below) is not visible – a Border and Status Display are shown

b. The Border consists of a set of lines occupying the visible edges of the LCD.
c. The Status Display appears at the top of the LCD. It consists of a row containing three text areas, underlined by a solid line extending the full width of the display. The text areas show:
    i. Player Lives, initially 10 (or 3, or something reasonable).
    ii. Player Score, initially 0.
    iii. Elapsed game time, measured in minutes and seconds, initially 00:00
d. The Playfield is the remainder of the LCD. It is bounded by the Border on three sides and the Status Display on the fourth.
e. The status bar values shall be updated to reflect the current state of the game (ie: it shall update when a player loses a life, etc).

3. Game Over Dialog: **[1 mark]**
    a. The Game Over dialog appears when the number of remaining lives reaches 0.
    b. The Game Over dialog informs the user that the game is finished and asks if they would like to play again.
    c. After the dialog has been displayed, the program should wait for the user to press SW1 or SW2. Once pressed, the game should return to the introduction screen and allow the player to play a new game.
    d. The game must be replayable after the user has lost the game, no game dynamics should be affected by returning to the Introduction screen.

4. Spacecraft: **[4.5 marks]**
    a. The Spacecraft materialises at the beginning of play, and re-materialises each time a life is lost, unless the game is over.
    b. The Spacecraft image should be no less than 3 units high and 3 units wide. Preferably, it should resemble a space capsule or fighter craft of some kind.
    c. The Spacecraft should display some representation of their aim (a simple line will suffice) which is pointing in the last direction the player moved.
    d. The Spacecraft must materialise at a random location on the game screen that is not obstructed by an enemy spacecraft.
    e. The Spacecraft moves in all four directions (up/down/left/right) when the player uses the directional pad.
    f. The Spacecraft shall move smoothly when the player inputs an action.
    g. No part of the Spacecraft may be seen to overlap the border or leave the Playfield.

5. Aliens: **[5 marks]**
    a. At least one alien materialises at the beginning of the game.
        i. Rules for the scenario in which there are more than one alien are covered in the “Multiple Aliens” and “Mothership Fight” features. See below.
    b. The alien shall have a unique sprite (compared to the player) that resembles an alien spacecraft. It should be no less than 3 units high and 3 units wide.
    c. Alien(s) must materialise at random positions within the Playfield so long as they do not overlap with the player (ie they cannot spawn on top of the player).
    d. The alien shall attempt to kill the player sprite using a “dash-attack”:

i. At random intervals (between 2-4 seconds) the alien will attempt to ram the player's spacecraft by moving straight towards the location occupied by the player's spacecraft when the alien begins its attack. A Portfolio activity (Topic 3, Shooting Star) guided you through the implementation of similar functionality.
ii. The alien shall move in a straight line at a velocity of 4 pixels per second.
iii. The alien shall continue moving in a straight line until it either a) collides with a player or b) collides with the border.

e. When an alien collides with the Border, it must stop moving just before touching the border of the Playfield. Aliens must never visibly overlap the Border or leave the Playfield.

6. Missiles: **[4 marks]**
   a. The player should be able to launch at least one missile by pressing either SW1 or SW2 (you may choose which button).
      i. This section covers the situation where you choose to implement a single missile.
      ii. Rules for the scenario where there may be more than one missile in flight at the same time are covered under the "Multiple Missiles" feature, below.
   b. This section requires at least a partial implementation of the "Spacecraft" feature.
   c. If the launch key is pressed and a missile is already in flight, nothing should happen, and the missile should continue uninterrupted.
   d. If the launch key is pressed and no missile is in flight, a missile should be launched. When the missile is launched:
      i. It is made visible and placed immediately after the launch point on the Spacecraft.
      ii. It is set in motion at constant velocity, moving in the spacecraft's current heading, at a speed of approximately 5 to 10 pixels per second.
   e. After launch, the missile should fly until either:
      i. It reaches the border of the Playfield, at which time it disappears and becomes available for launch; or
      ii. It collides with an alien. See "Missile Collision" feature.
   f. The missile must be represented as a 2 by 2 pixel square on the LCD screen, and must move smoothly during its flight.

7. Bounding Box Collision Detection between Spacecraft and Alien. **[2 marks]**
   a. When the player has more than one remaining life and a collision is detected between Spacecraft and Alien:
      i. The Spacecraft rematerialises according to specification under "Spacecraft".
      ii. The alien ship is more durable than our spaceship, and will be able to survive the collision with your spacecraft. After the collision, it should continue along its trajectory until it collides with a wall.
      iii. Any missiles in flight become invisible, and can no longer hit Aliens.
      iv. The number of lives remaining decreases by 1, and the new value is displayed correctly in the Status Display.

b. When the player has only one remaining life and a collision is detected between Spacecraft and Alien:
    i. The Game Over Dialog must be displayed.
    ii. Further actions are fully determined by the user's interaction with the Game Over Dialog.
c. Bounding box collision detection consists of checking to see if the bounding boxes of two images overlap.

8. Missile Collision: **[0.5 marks]**
    a. When a missile intersects the bounding box of an Alien:
        i. The missile disappears and the Alien spacecraft hit disappears.
        ii. The score is incremented by 1.
        iii. In the basic scenario, the Alien spacecraft rematerialises at a random location not currently occupied by the player's spacecraft.
        iv. In the "Multiple Aliens" and "Mothership" scenarios, the behaviour is determined by the rules for that scenario.

9. Elapsed Time: **[0.5 marks]**
    a. The Elapsed Time field in the Status Display is updated in a timely manner once per second.
    b. Elapsed Time is the time spent in play. The time shown in the Status Display must be reset after a new game has been started.

10. Multiple Missiles: **[1 mark]**
    a. This feature extends "Missile" by introducing the ability to display up to 5 missiles on the screen at the same time.
    b. This feature requires at least a partial implementation of "Spacecraft" and "Missile" features.
    c. The marks for this section are added to any other marks for missile launch capability.
    d. Each missile behaves in flight as described in the "Missile" feature above. The only difference should be the number of missiles and the rule for launching.
    e. If the launch key is pressed and 5 missiles are already showing on the screen, nothing should happen, and the missiles in flight should continue uninterrupted.
    f. If the launch key is pressed and fewer than 5 missiles are showing on the screen, a missile should be launched. When the missile is launched:
        i. It is set in motion at constant velocity, moving toward the location of the player at the time of missile launch (unless you implement the Advanced Aiming feature, covered below), at a speed of approximately 5 to 10 pixels per second.
        ii. No other missiles in flight are affected in any way.

11. Multiple Aliens: **[1 mark]**
    a. This feature extends "Aliens" by increasing the number of aliens that materialise on

commencement of game (or after the player loses a life). The total number of aliens is your choice, but it should be at least 5, and must not be so great that the game ceases to be playable.

b. The mark for this section is added to any other marks related to aliens.
c. Initial location, flight and collision properties of individual aliens are governed by the rules set out under the "Aliens" feature.
d. All of the Aliens materialise at the beginning of the game at random locations not occupied by the player's spacecraft.
e. Aliens do not rematerialise immediately after they are destroyed by a missile. Instead, they disappear and do not interact with the Spacecraft or missiles in any way until such a time as they all rematerialize together, under the following conditions:
    i. When the player has successfully managed to destroy all aliens by shooting them with missiles, or:
    ii. The Mothership alien has been destroyed (see "Mothership Fight" below)
f. All alien attacks must be independent from each other. That is, the frequency of attacks should be random for each alien, between 2 to 4 seconds per attack.

12. USB Serial Communication: **[6 marks]**
    a. You are required to implement a USB serial debugger which sends periodic messages to a computer along with the current system time.
        i. The USB Serial messages must report the current system time for your teensy game. This system time must be accurate to the millisecond (ie 0.001 second).
        ii. The USB Serial message will be printed using the following template:
            [DEBUG @ ###.###] {message}
        iii. The hashes in the USB serial message template (###.###) must be replaced by the current system time in seconds.
        iv. The {message} tag must be replaced by an appropriate message from the list below: (b), (c), (d), (e) and (f).
    b. The game must show some indication of an active serial communication before the Introduction screen is shown (on power-up of the teensy). To do this:
        i. Send a "Greeting" message from your teensy to the computer terminal. This message should indicate the communication channel is active and
        ii. display a message on the teensy LCD screen that it is connected via the USB.
    c. Every 0.5 seconds **during game play**, the teensy must broadcast information about the spacecraft. These values (listed below) must be easily read on the serial terminal.
        i. The Spacecraft's current (x,y) position on the LCD
        ii. The Spacecraft's current heading (direction of travel: only if not fulfilling the "Advanced Aiming" feature) and it's current aim (see "Advanced Aiming")
    d. The teensy must send a message to the serial console during the following events. The message must clearly indicate which event has occurred, and must be easily read from the serial terminal.

i. Player kills an alien
ii. Alien kills the player

e. The USB serial debugger must also allow the player to send action commands to the teensy. The user may implement this feature using whichever keyboard keys they desire, and should indicate to the player which keys are required at start up. The actions are listed below.
   i. Move (up, down, left, right)
   ii. Shoot
f. All game dynamics described before this feature must **not** be affected by the implementation of the USB serial communication feature. Any action commands sent to the teensy via the serial channel must have the exact same dynamics as described in their respective sections.

13. Advanced Aiming System: **[4 marks]**
    a. This feature extends "Spaceship", and "Missiles" features by requiring you to implement 360 degree aiming for missiles fired from your spaceship.
    b. The aiming system must be implemented via the right potentiometer. You must implement the aiming system such that:
       i. It is independent of the movement of the spacecraft. Pressing the movement keys no longer changes the aim of your missiles.
       ii. Potentiometer values (which will vary between 0 and 1023) should be scaled to produce angular values between 0 and 720 degrees.
       iii. Some visual indication of the current aim of the spacecraft must be visible on the screen at all times (and must not overlap the border or leave the Playfield). This representation is up to you, but the simplest visual indication would be a straight line rotating around the spacecraft (see demonstration for further clarification)
    c. Missiles fired by the spacecraft must now fire in the direction indicated by the aiming system, and must continue to follow the specifications from the "Missiles" and "Multiple Missiles" sections

14. Mothership Fight: **[8 marks]**
    a. This feature requires you to implement a Mothership battle, which features a larger, tougher alien for you to battle.
       i. The Mothership must be represented by a sprite that is at least 8 by 8 pixels and has a distinctive image.
    b. The Mothership attempts to destroy the player using attacks already described in previous sections:
       i. The Mothership uses the dash attack described in the "Aliens" section, but it moves slower, at a rate of 2 pixels per second.
       ii. The Mothership has a single missile that it fires towards the player at random intervals of between 2 to 4 seconds. The Mothership's missile must follow the same dynamics as the player's missiles (as described in "Missiles").
    c. The Mothership can sustain multiple hits of the player's missiles before it explodes.
       i. You must always have a visible, clear indication of how many more missiles

are required to kill the mothership (health pool). At no point should this health pool indicator leave the game screen and must not obscure the status bar.

ii. The Mothership should initially be able to withstand at least 10 missile hits before it is destroyed (or something reasonable, it must remain beatable to demonstrate full functionality)

iii. Once the Mothership's health pool has been depleted, it should be hidden from the screen (in the same way described for other alien spacecraft), and the player should receive 10 points.

iv. If the Mothership has been removed, all aliens should materialize in the same way specified in "Aliens" and "Multiple Aliens" features.

d. If the player's spacecraft is destroyed by the Mothership, the player's spacecraft should rematerialize as described in the "Spacecraft" feature, making sure the player's spacecraft does not rematerialize on top of the Mothership.

i. The Mothership does not dematerialize until it has been destroyed, and thus will remain on the play field even after the player's destruction.

e. This feature also extends the "USB Serial Communication" feature by requiring you to also implement an event message for Mothership events.

i. You must include in your USB serial communication messages for:

1. Player Destroyed Mothership
2. Mothership Destroyed Player

ii. The mark for this section is added to any other marks related to USB serial communication.

iii. You must implement this feature following the same template specified in the "USB Serial Communications" section.

Remarks

- 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***.

- At all times, movement of the missile(s), player, and alien(s) must be smooth. That is, the movement of the sprites shall not move more than one pixel per update of the screen (ie They do not jump around the screen when moving).

Marking

The assignment will be out of 40 marks and will be worth 40% of your total grade in this subject. The following should be noted about marking:

- **If your code does not compile, you will get 0 marks for the entire assignment.**
- If the game crashes or locks up during testing, you will receive marks for what has been tested up to that point. 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**.
- The assignment will be marked during your scheduled tutorial in week 13. To receive a mark for the assignment, you are required to:
    - attend the tutorial in person,
    - bring your student ID card,
    - demonstrate your program to a tutor,
    - explain details of your implementation, and
    - hand in your Teensy.

If you fail to do any of these things you will receive a mark of 0 for the assignment.

Submission

Your assignment is due on Friday May 26th, 2017 at 11:59pm. Submission will be online through the AMS used in the tutorials. You must submit through the submission page which will be published at least two weeks before the submission deadline.

When submitting to the AMS, it is your responsibility to make sure that your assignment compiles correctly. In general, AMS will compile your code and return any errors that occur. There are circumstances when very poor code causes the compiler on the server to terminate abnormally, or to hang. *You are responsible for ensuring that this does not happen* by submitting only valid source code which you have already compiled using the same settings as those used by AMS. As mentioned above, if you do not submit a compiled assignment, you will receive 0 marks. You will have 50 submission attempts for the assignment, so there are no excuses for not resolving any compilation issues.

AMS operates in unattended mode outside regular Monday to Friday office hours. Please note that issues such as software failure, hardware failure, and network congestion problems DO NOT constitute exceptional circumstances for the purpose of Special Consideration. It is in your best interest to complete this assignment early to eliminate the possibility of a last-minute disaster. To ensure success you should aim to submit a substantially complete version of your assignment by the beginning of Week 12.