CAB202 Semester 1 2017
Assignment 1: The Diamonds of Doom

Due Date:	**Friday, 7 April 2017, 11:59 PM**
Submission:	AMS
Assessment Weight:	30% of your total score for the unit

Introduction

The Diamonds of Doom is a casual game that you will implement with the CAB202 ZDK character-based graphics library. In this game, the terminal window is partitioned into two regions, a status display that occupies the top three rows of the window, and a playfield that fills the remainder of the terminal. The player controls a spacecraft that moves horizontally at the lower edge of the playfield. Diamonds materialise at the top of the playfield, initially travelling at a constant speed in a generally downward direction, and bouncing off the boundaries of the playfield using the algorithms supplied in lectures and tutorials. The player can launch missiles, which travel vertically upwards at constant speed. If a missile impacts a diamond the player's score increases, and the diamond rematerializes at the top of the playfield. In more feature-complete implementations, diamonds split into smaller fragments when hit by the player's missile, only rematerializing when all diamonds (and splinters formed by smashing diamonds) have been cleared from the playfield. If a diamond (or splinter) collides with the player's spacecraft, then the player loses a life. The game is over when the number of remaining lives reaches zero.

Your task is to implement The Diamonds of Doom, using the ZDK library to target a generic UNIX-like system, and following the patterns and examples you have been shown in class. Detailed specifications of the required items of functionality appear in the following section. Additional resources, including video recorded during a lecture and a ZDK movie, accompany this specification. You are strongly advised to make use of these resources.

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 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.

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.

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 20 units.
    b. You may assume that the user will not resize the window while the game is running.
2. Border, Status Display, and Playfield: **[2 marks]**
    a. During normal play mode – that is, at any time when the Help Dialog or Game Over Dialog (see below) are not visible – a Border and Status Display are shown.
    b. The Border consists of a set of lines occupying the visible edges of the terminal window.
    c. The Status Display appears at the top of the terminal. 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. Lives, initially 10 (or 3, or something reasonable).
        ii. Score, initially 0.
        iii. Elapsed game time, measured in minutes and seconds, initially 00:00.
    d. The Playfield is the remainder of the terminal window. It is bounded by the Border on three sides and the Status Display on the fourth.
3. Help Dialog: **[2 marks]**
    a. A Help Dialog is displayed on the following occasions:
        i. Commencement of game, prior to the start of play.
        ii. If user presses 'h' at any time during play.
        iii. After the Game Over dialogue (see below), if user has selected “Play again”, before moving on to restart the game.
    b. The Help Dialog must show your name and student number, plus a list of keyboard commands required to play the game.
    c. The information should be displayed in the centre of the Dialog, and should be laid out in a tidy and readable manner.
    d. Help Dialog exits when user "Presses any key".
    e. While the Help Dialog is displayed, all other game dynamics are frozen, including the game time counter.

f. While the Help Dialog is displayed, all other visual elements including the Border and Status Display must be hidden.

g. The keyboard buffer should be purged before the help Dialog is displayed, and again before play resumes. That is, any pending keystrokes in the keyboard buffer must be removed and discarded. This ensures that:

  i. The user will be able to view the help Dialog before pressing a key to resume play.

  ii. Keystrokes entered while the help Dialog is displayed will have no effect other than to erase the help Dialog and restore the visual state of the game to play mode. Quitting directly from the Help Dialog should not work.

  iii. The following instruction can be used to purge the keyboard buffer:

```
while ( get_char() >= 0 ) {}
```

4. Game Over Dialog: **[1 Mark]**

  a. The Game Over dialog appears when user presses 'q', or 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 wold like to play again.

  c. Before the dialog is displayed, the keyboard buffer should be purged as noted above to ensure the user has a chance to interact properly with the dialog.

  d. After the dialog has been displayed, the program should wait for the user to enter 'y' or 'n', pausing until one or the other of these characters is pressed.

  e. If the user responds 'y':

    i. The dialog should close, and a new game should commence.

    ii. All objects in the game should be restored to their condition as the would be at the start of the first game. Thus:

      1. Any missiles disappear.

      2. Any splinters disappear.

      3. Spacecraft and Diamonds materialise according to the relevant rules.

  f. If the user responds 'n':

    i. The program should clean up the display and exit cleanly.

5. Spacecraft: **[2 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 5 units wide. Preferably, it should look resemble a space capsule or fighter craft of some kind.

  c. The Spacecraft must materialise at the bottom of the Playfield, horizontally positioned in the centre.

  d. The Spacecraft moves horizontally *but not vertically* when the user presses designated keys of your choice.

  e. No part of the Spacecraft may be seen to overlap the border or leave the Playfield.

   f. The Spacecraft should move one unit left or right when a keystroke on the appropriate key is detected.

6. Diamonds: **[2 marks]**
   a. At least one Diamond materialises at the beginning of the game.
      i. Rules for the scenario in which there are more than one diamond are covered in the “Multiple Diamonds” and “Splinters” features, appearing below.
   b. A Diamond should be indicated by an image that resembles a diamond. It should consist of a 5 by 5 character image.
      i. *Note: a square is technically a Diamond lying on its side, so even a square will qualify at a basic achievement level. But for full marks in the “Pixel-level Collision Detection” feature, a proper Diamond is required.*
   c. Diamond(s) must materialise at the top of the Playfield, and must appear at randomly generated horizontal positions within the Playfield, even when there is only one Diamond. If a Diamond rematerializes, it should appear at a new randomly generated horizontal position.
   d. Each Diamond will initially move towards the bottom of the Dialog, at a speed of approximately 5 to 10 pixels per second.
      i. For full marks, each Diamond must move in a randomly generated initial direction. Part marks can be attained with vertical motion.
   e. Diamonds may not jump more than one unit between consecutive steps of the animation.
   f. When a Diamond collides with the Border, it must reflect in the manner demonstrated in supporting sample applications. Therefore, Diamonds must never visibly overlap the Border or leave the Playfield.

7. Missiles: **[2 marks]**
   a. The player should be able to launch at least one missile by pressing a designated key that you may choose.
      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 above the launch point on the Spacecraft.
      ii. It is set in motion at constant velocity, moving straight up at a speed of approximately 5 to 10 pixels per second.
   e. After launch, the missile should fly until either:
      i. It reaches the top of the playfield, at which time it disappears and becomes available for launch; or

   ii. It collides with a Diamond. See “Bounding Box Missile Hit” and “Interior Missile Hit” features.

8. Bounding Box Collision Detection between Spacecraft and Diamond. **[2 marks]**
   a. When the player has more than one remaining life and a collision is detected between Spacecraft and Diamond:
      i. All Diamonds that are required to rematerialize (“Multiple Diamonds” and “Splinters”, below) do so at the top of the Playfield, following all guidelines for materialisation listed under “Diamonds”. If you have a single Diamond, it is required to rematerialize.
      ii. The Spacecraft rematerializes according to specification under “Spacecraft”.
      iii. Any missiles in flight become invisible, and can no longer hit Diamonds.
      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 Diamond:
      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.
      i. Bounding box detection is sufficient to implement rudimentary collision detection and demonstrate the overall gameplay effects of collision.

9. Bounding Box Missile Hit: **[1 mark]**
   a. When a missile intersects the bounding box of a Diamond:
      i. The missile disappears.
      ii. The score is incremented by 1.
      iii. In the basic scenario, the Diamond rematerializes.
      iv. In the “Multiple Diamonds” and “Splinter” scenarios, the behaviour is determined by the rules for that scenario.

10. Elapsed Time: **[1 mark]**
   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 clock stops whenever the game is paused, that is, whenever a dialog (Help Dialog or Game Over Dialog) are visible. When the dialog closes, the clock resumes at the value it had at the time the dialog opened.

11. Multiple Missiles: **[2 marks]**
   a. This feature extends “Missile” by introducing the ability to display up to 100 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 100 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 100 missiles are showing on the screen, a missile should be launched. When the missile is launched:

   i. It is made visible and placed immediately above the launch point on the Spacecraft.

   ii. It is set in motion at constant velocity, moving straight up at a speed of approximately 5 to 10 pixels per second.

   iii. No other missiles in flight are affected in any way.

12. Multiple Missile Stations: **[2 marks]**

   a. This feature extends "Missile" by adding two more missile launch points to the Spacecraft.

   b. This feature requires at least a partial implementation of "Spacecraft" and "Missile" features.

   c. The mark for this section is added to any other marks for missile launch capability.

   d. Additional keys should be introduced (and documented in the help Dialog) to allow the player to choose which missile station to launch from.

   e. One keypress should launch one missile, in line with the "Missile" feature.

13. Multiple Diamonds: **[2 marks]**

   a. This feature extends "Diamonds" by increasing the number of diamonds that materialise on commencement of game (or after the player loses a life). The total number of diamonds is your choice, but it should be at least 10, 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 diamonds.

   c. Initial location, flight, and collision properties of individual diamonds are governed by the rules set out under the "Diamonds" feature.

   d. All diamonds materialise at the same time:

      i. At the start of the game,

      ii. When the player has lost a life,

      iii. When the player has successfully managed to destroy all diamonds by shooting them with missiles.

   e. Diamonds do not rematerialize 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.

14. Splinters: **[4 Marks]**

   a. This feature extends collision detection between missiles and diamonds.

   b. The mark for this feature is added to any other marks for diamond-related functionality.

   c. When a missile hits a diamond, instead of rematerializing or disappearing, the following behaviour is required.

      i. If the diamond has dimensions 5×5:

        1. The diamond disappears, and is replaced by two smaller diamonds, each of size 3×3.

    ii. If the diamond has dimensions 3×3:

        1. The diamond disappears, and is replaced by two tiny diamonds, each of size 1×1.

    iii. If the size of the diamond is 1×1:

        1. The diamond disappears.

d. In the two cases where a diamond is replaced by a pair of splinters:

    i. Both smaller diamonds materialise at the centre of the original diamond.

    ii. Both smaller diamonds fly at the same speed as the original diamond.

    iii. One of the smaller diamonds flies on a path that differs from that of the original by +45°.

    iv. The other smaller diamond flies on a path that differs from that of the original by –45°.

e. As for "Multiple Diamonds", the diamonds rematerialize only when all diamonds have been destroyed.

    i. The original 5×5 diamonds rematerialize.

    ii. Splinters do not reappear until a 5×5 diamond is destroyed.

15. Pixel Level collision detection: **[4 Marks]**

    a. This feature extends collision detection, so that objects are deemed to collide only when a non-blank character from each occupies the same location on the Dialog.

    b. The mark for this feature is added to any other marks related to collision detection.

    c. Under this feature the bounding boxes of two objects may overlap, allowing a greater range of near miss behaviours.

Remarks

- You may choose to implement any of the functionality described in this specification sheet in any order, as long as dependencies listed under individual features are satisfied.
- No extra marks will be given for implementing functionality other than 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 all mobile objects must be smooth. For the purpose of this assignment, smooth means that no object is ever observed to move more than one Dialog unit in x or y direction between successive frames.

Marking

The assignment is worth 30% of your total grade in this subject, and it is marked out of 30. The breakdown of marks is outlined in the Game Specification section. The following points should be noted about marking:

1. If your code does not compile when submitted to AMS, the mark awarded will be 0.

2. If the program has been implemented via a framework other than the ZDK without prior written permission from a Unit Coordinator, the mark awarded will be 0.
3. If the program fails with segmentation faults or other fatal errors, marks will be awarded for the features that were observed prior to the crash. No effort will be made to work around the crash, nor will we debug your code to make it compile or run.
4. It is your responsibility to implement each feature sufficiently well that it is readily detected through normal operation of the game. Any feature that is not apparent through normal play will be deemed to be unimplemented.
5. We require tutors to adhere to a strict time limit of 3 minutes play time when marking each submission. Any feature that cannot be assessed in that time will be deemed to be unimplemented. Therefore you must avoid defects in game dynamics such as extremely fast motion, extremely slow motion, inconsistent control settings, premature program termination, or other properties that render the game unfit for use.

   It is better to implement some of the features extremely well than to try to do everything and deliver a substandard product.
6. Penalties will be applied if the code exhibits general defects or undesirable behaviour not otherwise covered in this document. This includes but is not limited to such things as: errors in object motion, such as objects jumping more than one character position per frame; objects moving outside their permitted area; failure to clear the screen appropriately between updates; collisions involving invisible or non-existent objects; disturbing or flashing display; failure to purge the keyboard buffer at appropriate times. Where a penalty is necessary, the mark deducted will be proportional to the impact of the defect on any affected features.

Submission

Submission will be online through AMS via the same process as used for your portfolio items. The link to the submission page will be published on the unit Assessment page in Blackboard 14 days prior to the due date. When submitting, be careful to follow all instructions on the submission page. It is your responsibility to make sure that AMS is able to compile your assignment. We recommend that you submit your work in progress at least once a day in the fortnight prior to the submission deadline. The latest submission received before the deadline will be marked.