COMP2611: Computer Organization

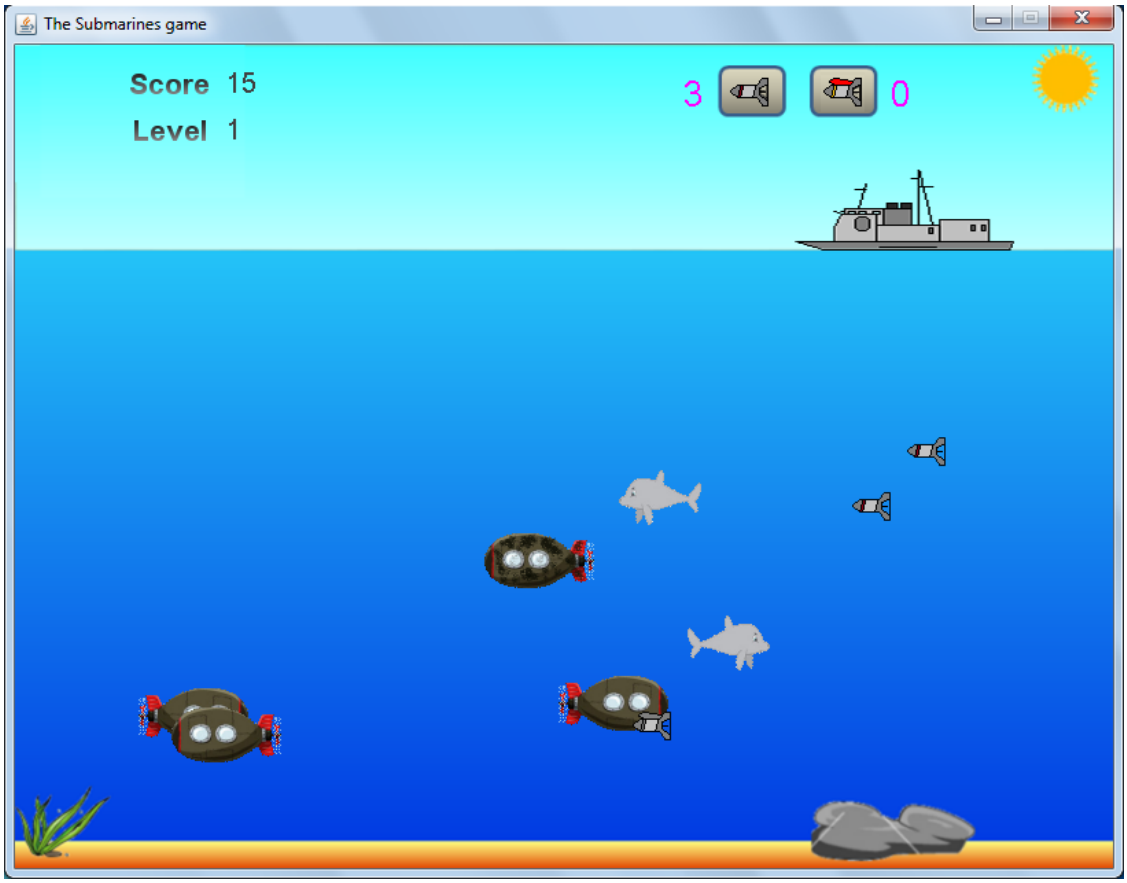
Spring 2012

Programming Project: The Submarines Game

Due Date: 21 May, 2012, 12:00pm

Introduction

In this project, you need to complete the game called *The Submarines*. A snapshot of the game is shown below. In the game, you control a ship to drop bombs to damage the submarines undersea in order to destroy all of them. There are also dolphins swimming undersea. Damaging a submarine will give you a positive score while damaging a dolphin will penalize your score.



Game objects

The game is played on the screen of 800-pixel width by 600-pixel height. The x-axis and y-axis go to the right and downward (from 0), respectively. Each game object: ship, submarine or dolphin is represented by a rectangular image. The location of an object is specified by the coordinate of the top-left corner pixel of its image. The size of an object is specified by its image size as follows:

Object	Image width (in pixels)	Image height (in pixels)
Ship	160	160
Dolphin	60	40
Submarine	80	40
Bomb	30	30

Note that the coordinates of the pixels of the game screen or an image are zero-based, e.g., the bottom-right pixel of the game screen is at the coordinate (799, 599).

Game levels

The game is divided into 10 levels. The score is initially zero and is for the whole game (not reset in each level). You input N (within a maximum limit of 5) for the number of dolphins in the first level. In each level, the number of submarines is 3 more of that for the dolphins. In the start of the first level, the location (top-left corner) of the ship is (x, y) = (320, 90). Thus, the ship is placed at the screen center on the surface of the sea (y level = 150). It is initially moving right. Its position remains unchanged for the next level. However, in the start of each level, the submarines and dolphins are randomly placed undersea. The y-coordinate of each object's location is randomly chosen between 250 and 500 according to the restriction: it must be different from those of the other objects. Also, the moving direction (left or right) of each of these objects is randomly chosen.

Object movements

When the ship, submarine or dolphin moves outside the left or right boundary of the game screen (completely or partially), its moving direction is changed to the opposite one. Its location's x-coordinate is changed to 0 or 800 minus its width for turning around from the left or right boundary, respectively. The ship, submarines and dolphins move horizontally at the constant speeds 4, 6 and 5, respectively, in pixels per a time unit (see the Implementation below). Here the moving direction is also specified using the sign of the speed such that moving right or left is specified by a positive or negative speed, respectively.

Bombs

A bomb is dropped from the bottom center of the ship. It falls down vertically at the constant speed 4 in pixels per a time unit (see the Implementation below). Each bomb is in either of the two status: Activated or Disabled. A bomb intersecting (including touching) a submarine or dolphin will only hit that object if the bomb is Activated; otherwise the bomb will just fall through the object. The intersection is checked based on the rectangles of those objects' images. After a bomb hits a dolphin or submarine or falls outside the game screen (completely or partially), it is removed from the game screen. A bomb can hit multiple objects if it intersects them at the same time.

There are two types of bombs: Simple and Remote. Pressing the key '1' will drop a Simple bomb. It is always Activated. Pressing the key '2' will drop a Remote bomb. It is initially Disabled. Pressing the key 'e' will change all the Remote bombs undersea to the Activated status. At the start of each game level, 1 Remote bomb and 5 Simple bombs are available. Dropping a bomb decreases the number of the available bombs of that type by 1. Each bomb removed from the game screen increases the number of the available bombs of that type by 1.

Damaging submarines or dolphins

At the start of each game level, each submarine and dolphin has a Hit point 10 and 20, respectively. For a submarine at the location (x, y), if an Activated bomb intersects the rectangle in 10-pixel width by 40-pixel height at the location (x + 35, y), then the middle part of the submarine is considered hit by the bomb. In this case, all the Hit point of the submarine will be deducted and added to the game score. If an Activated bomb intersects a submarine in any other cases, then the submarine's Hit point will only be deducted by 5 and the game score will be increased by 5. When the Hit point of a submarine becomes zero, it is destroyed, but its destroyed body (using another image) remains on the game screen for a short while (see the Implementation below).

An Activated bomb intersecting a dolphin (in any ways) will deduct all his/her Hit point, and the game score will also be deducted by the same amount. Like a submarine, when the Hit point of a dolphin becomes zero, he/she is destroyed, but his/her destroyed body (using another image) remains on the game screen for a short while.

Game winning

After all the submarines in a game level are destroyed, you will have won the

level. Then, the number of dolphins is increased by 3 for the next level, except that it is doubled if the next level is level 5 or 10. The game score is kept for the next level. Pressing the key 'q' during the game or winning the last game level terminates the game. The game screen will not be closed at the game termination. But it can be closed later by clicking the "Close" button on the top-right corner of its window.

Implementation

All the game objects (ship, submarines, etc.) and information (score, level no., number of available bombs) are initialized at the start of the first game level. Then, a new window of the game screen with the game objects and information will appear. The game then progress as an infinite loop of the following steps:

1. Get the current time (T1).
2. Remove any destroyed submarines and dolphins from the game screen.
3. Check for any input, which is stored using the Memory-mapped I/O scheme (see the section Hints for details). If an input is available, read it and perform the action for it as follows:

Input	Action
q	Terminate the game
1	Drop a Simple bomb (while one is available)
2	Drop a Remote bomb (while one is available)
e	Change the status of all the Remote bombs undersea to Activated

4. For each Activated bomb undersea, check for any hits between it and the submarines or dolphins. In each hit, update the Hit point of the submarine or dolphin involved and the game score. After all the hits of a bomb, remove it from the game screen and add it back to the available ones.
5. Update the image of each damaged or destroyed submarine according to its Hit point (5 or 0). Also, update the image of each destroyed dolphin.
6. Check whether the game level has been won. If the last game level has been won, display a message "You have won!" on the game screen and then terminate the game. For winning the other level, re-initialize the game

for the next level and then jump to Step 9.

7. Move the submarines, dolphins and ship by the number of pixels and the direction specified in their own speed.
8. Move the bombs undersea downward by the number of pixels specified in their own speed. Then remove the bombs below the game screen(completely or partially) and add them back to the available ones.
9. Redraw the game screen with the updated location and image of the game objects and game state information.
10. Get the current time (T2), and pause the program execution for (30 milliseconds - (T2 - T1)). Thus, the interval between two consecutive iterations of the game loop is about 30 milliseconds.

Your tasks

You must complete the game using only the programming language MIPS. You are given a custom-made Mars program Mars_4_1_withSyscall100.jar and the skeleton program submarines.s for the game. Use this Mars program to run the game. The skeleton program contains some special syscall operations that are not taught in the course. You do not need to understand them or use them in your codes. You may read the program briefly or in details in order to understand how it implements the game. But you only need to complete the code of the functions mentioned below for certain game tasks. Once they have been completed, the program should have implemented the game completely. Read the comments of those functions in the program and the game description above for doing the tasks correctly.

Function	**Task**
Initgame	Initialization of a game level (e.g., generation of submarines)
removeDestroyedObjects	Step 2 of the game loop in the Implementation
processInput	Step 3 of the game loop in the Implementation
checkBombHits	Step 4 of the game loop in the Implementation
updateDamagedImages	Step 5 of the game loop in the Implementation
moveShipSubmarinesDolphins	Step 7 of the game loop in the Implementation

moveBombs	Step 8 of the game loop in the Implementation

To work with the game objects, you must use the arrays (of MIPS words) defined in the file: **ship**, **submarines**, **dolphins** and **bombs**. Every 5 words in **submarines**, **dolphins** and **bombs** stores five properties of one submarine, dolphin and bomb, respectively. Read the code comments beside the defining of the arrays in the program file for details. Both Simple and Remote bombs should be stored in the same array **bombs**. It is because only their properties "status" (Activated or Disabled) differ at the moment of dropping these different types of bombs.

Certain registers are used for storing the game progress information. You must use them in your codes. Read the code comments at about the top of submarines.s for details.

The game images are provided and are listed below. To set a game object to use a certain image, set the object's property "image index" to the corresponding index no. in your codes. If the image index is set to negative, the object will not be drawn on the screen.

Image index	**Image**	**Description**
0	background.png	Game screen background
1	shipR.png	Ship moving right
2	shipL.png	Ship moving left
3	subR.png	Submarine moving right
4	subL.png	Submarine moving left
5	subDamagedR.png	Damaged submarine moving right
6	subDamagedL.png	Damaged submarine moving left
7	subDestroyed.png	Destroyed body of submarine
8	dolphinR.png	Dolphin moving right
9	dolphinL.png	Dolphin moving left
10	dolphinDestroyed.png	Destroyed body of dolphin

11	simpleBomb.png	Simple bomb
12	remoteBombD.png	Remote bomb in Disabled status
13	remoteBombA.png	Remote bomb in Activated status

Bonus

You may get an additional 10% bonus marks (e.g., 10 marks if the full project mark is 100) for implementing a player-controllable ship. In each iteration of the game loop (mentioned in the section Implementation), instead of moving the ship by its current speed and direction, you move the ship left or right (by 4 pixels in the original speed) if the player has input a certain key. Use a different key for the left and right movements. These two keys must be different from those of the other game inputs defined earlier. If the player has not input such a movement key, the ship remains still in that iteration. If the ship moves outside a screen boundary, it should still be turned around as mentioned before.

Hints

The keyboard inputs for playing the game are stored using the Memory-mapped I/O scheme. The function **getInput** in the file submarines.s is provided for you to check and read any input stored using this scheme.

For any two rectangles, say A and B, they intersect (or touch) each other if and only if one of the following conditions holds:

- A's largest x-coordinate is smaller than B's smallest x-coordinate
- A's smallest x-coordinate is larger than B's largest x-coordinate
- A's largest y-coordinate is smaller than B's smallest y-coordinate
- A's smallest y-coordinate is larger than B's largest y-coordinate

The function **isIntersected** in the file submarines.s is provided for you to check intersections. The functions **randnum** and **randomSignChange** are provided for generating a random number and randomly changing the sign of a number, respectively.

Read the code comments of all these provided functions for how to use them, but using them is optional.

Submission

You should submit the file submarines.s with your completed codes for the project using the CASS (https://course.cse.ust.hk/cass). **No late submission is allowed.** The submitted file name must be exactly submarines.s. The CASS user manual is in this link http://cssystem.cse.ust.hk/UGuides/cass/index.html

At the top of the file, please write down your name, student ID, email address, lab section and the two ship movement keys (if you have completed the Bonus part) in the following format:

#Name:

#ID:

#Email:

#Lab Section:

#Bonus -- left-movement key: right-movement key:

Grading

Your project will be graded on the basis of the functionality listed in the game requirements. Therefore, you should make sure that your submitted codes can be executed properly in the provided Mars program.

Inline code comments are not graded. However, you may be asked to explain your codes in a face-to-face session (likely organized during this Spring 2012 examination period after the project due date).