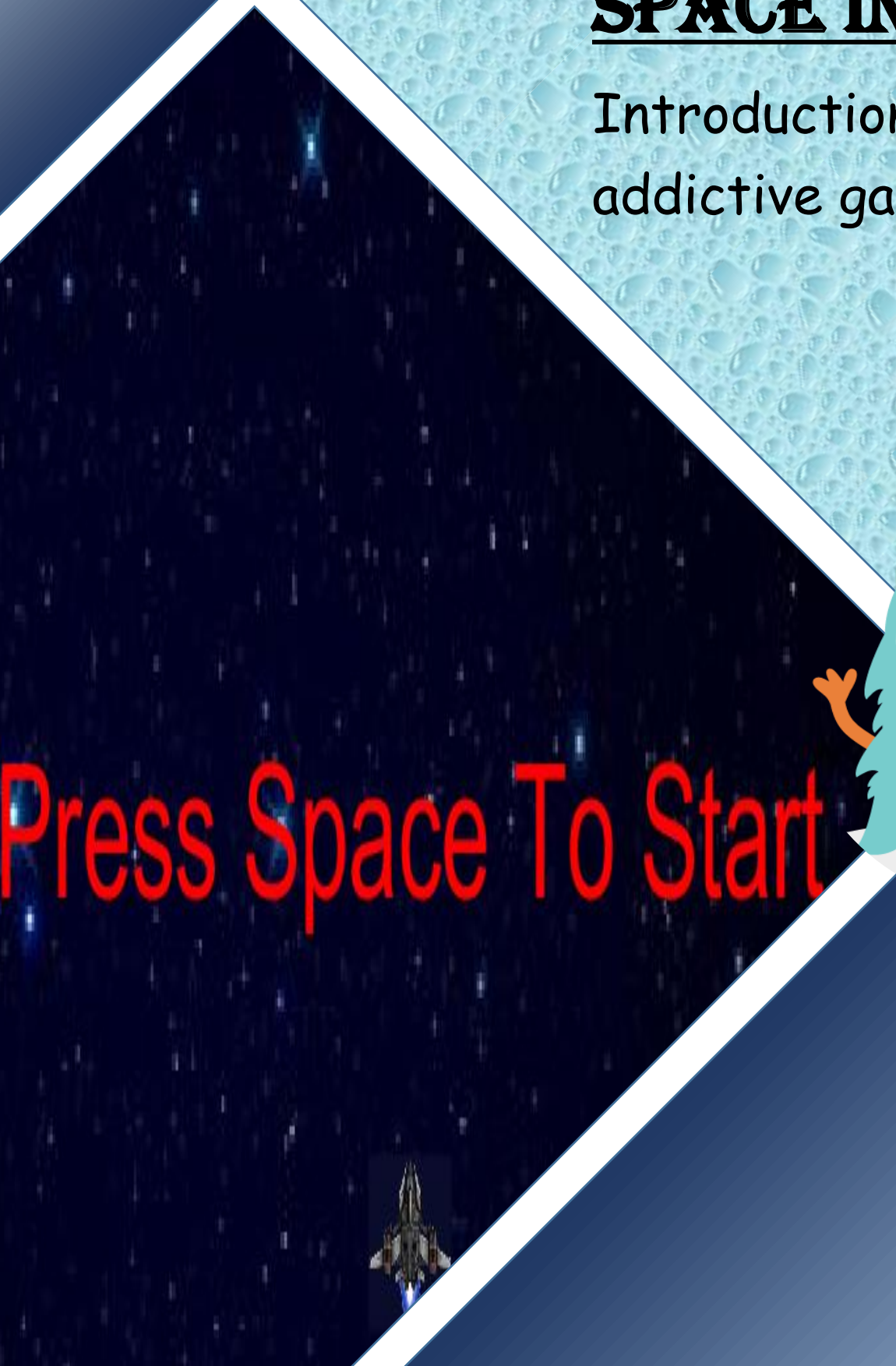
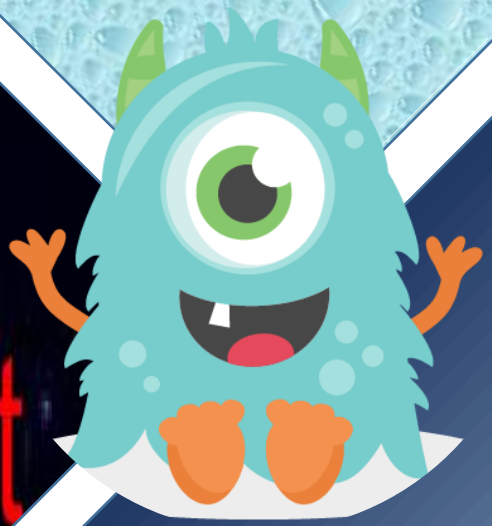
SPACE INVADERS:

Introduction to an addictive game





BANGLADESH UNIVERSITY OF ENGINEERING TECHNOLOGY

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

Course No: EEE-212

Course Title: Numerical Technique Laboratory.

Name of Project: Designing the Space Invader game using Matlab.

Submitted to:

Shahed Ahmed
Lecturer

Shaimur Salehin Akash
Lecturer

Department of Electrical and Electronic Engineering,
BUET, Dhaka

Submitted by:

Md. Jarjis Mondal - 1806068

Md. Ishraq Anzum - 1906095

Section: B-1

Level 2 Term1

Date of Submission: 19/02/2022

Table of Contents

Topics **Page**

List of Figures

Figures **Page**

Introduction:

MATLAB (an abbreviation of "MATrix LABoratory") is a proprietary multi-paradigm programming language and numeric computing environment developed by MathWorks. MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages. In this project, we will use Matlab to implement our creativity. We will design the “Space Invader” game. For this, we had to follow some algorithms to fix the background, character, audio and some logics to make the game interesting. This game is going to be very classic and simple one. Using up arrow button, we will be able to shoot the monsters/ aliens. Thus it will increase the game score. If we fail to shoot any monster before it reaches the ground, we will lose the game. And the window will show the best and current scores.

Theory:

The whole program was encoded with only numerous functions and variables. Some of the commands and functions are explained briefly bellow:

1. warning():
   Our program might issue warnings that do not always adversely affect execution. To avoid confusion, we can hide warning messages during execution by changing their states from 'on' to 'off'.
2. audioread():
   ‘audioread’ provides a single, unified Matlab function for reading audio files in a range of different file formats, including wav, mp3, aac, flac, AIFF, etc. In most cases, access is actually provided by external binaries, but this is hidden within audioread.
3. audioplayer():
   We used an audioplayer object to play audio data. The object contains properties that enable additional flexibility during playback.
4. figure():
   This is used to create a new figure window using default property values. figure(n) finds a figure in which the Number property is equal to n , and

makes it the current figure.

5. Axes():
   axes is the low-level function for creating axes graphics objects. axes creates an axes graphics object in the current figure using default property values. MATLAB uses default values for any properties that you do not explicitly define as arguments.
6. Image():
   image(C) displays the data in array C as an image. Each element of C specifies the color for 1 pixel of the image. The resulting image is an m -by-n grid of pixels where m is the number of rows and n is the number of columns in C .
7. Text():
   The MATLAB function text() is defined to place description texts to data points on a plot. The inclusion of the text to single data point is carried out by adding text to one point that is specified with x and y as scalars. While text to multiple points is added by specifying x and y as vectors of equal length.
8. Imread():
   A = imread(filename, fmt) reads a greyscale or color image from the file specified by the string filename , where the string fmt specifies the format of the file. If the file is not in the current directory or in a directory in the MATLAB path, specify the full pathname of the location on your system.
9. Load():
   load(filename) loads data from filename . If filename is a MAT-file, then load(filename) loads variables in the MAT-File into the MATLAB® workspace. If filename is an ASCII file, then load(filename) creates a double-precision array containing data from the file.
10. Set():
    pv = set(h,'PropertyName') returns the possible values for the named property. If the possible values are strings, set returns each in a cell of the cell array, pv . For other properties, set returns an empty cell array. If you do not specify an output argument, MATLAB displays the information on the screen.
11. Play():

play(playerObj) plays the audio associated with audioplayer object playerObj from beginning to end. play(playerObj,start) plays audio from the sample indicated by start to the end. play(playerObj,[start,stop]) plays audio from the sample indicated by start to the sample indicated by stop .

12. Drawnow():
    drawnow updates figures and processes any pending callbacks. Use this command if you modify graphics objects and want to see the updates on the screen immediately. example. drawnow limitrate limits the number of updates to 20 frames per second.
13. Randperm():
    p = randperm(n) returns a row vector containing a random permutation of the integers from 1 to n without repeating elements. example. p = randperm(n , k) returns a row vector containing k unique integers selected randomly from 1 to n .

User Defined Variables:

We have used some variables to implement the algorithm. They are described below:

Monx = X axis co-ordinate of the monster

Mony = Y axis co-ordinate of the monster

Spaceinit = Position and size of spaceship

Monint = Position and size of monster

Bulinit = Position and size of bullet

new_start_game = Variable which indicates to start game

fire = Variable which indicates to fire a bullet

direction = Direction of spaceship to move

bul_stat = Firing status of the bullet

touch_bottom = Variable to check if the monster reached the bottom

score = Present score

runloop = Loop variable to keep the game running

mdir = Direction status of monster to move

bdir = Direction status of bullet to move

bullets = Total bullets fired

s_bul = Variable to store information about bullet

MainFigureSize = Size of the figure

MainAxesSize = Size of the axes

MainFigureinitPos = Variable to indicate the placement position of the figure in the screen

bgmplay = Background music

Iplay = Laser sound

explplay = Explosion sound

da = Background image

sp = Spaceship image

mon = Monster image

bul = Bullet image

User Defined Functions:

We had to use some user-defined functions for our code. They are described below:

MainFigureHd1 = The main figure function

MainAxesHd1 = Axes function in the main figure

MainCanvasHd1 = Background image

SpaceCanvasHd1 = Spaceship image

Mon1CanvasHd1 - Mon10CanvasHd1 = Monster image

GameOverHd1 = “Game over” text

PressKeyHd1 = “Press Space to start” text

ScoreInfoForeHd1 = “Score : 4” text

ScoreInfoHd1 = Current score and best score text

BulCanvasHd1 = Bullet image

stl_KeyPressFcn = Function to take input from the keyboard

stl_CloseReqFcn = Function to close the game

restart_game = Function to initialize the variable when the game restarts

Algorithm:

The step by step algorithm for the code is as follows:

1. Declare Monx, Mony, Spaceinit, Monint, Bulinit, new_start_game, fire, direction, bul_stat, touch_bottom, score, runloop, mdir, bdir, bullets, s_bul
2. Set MainFigureSize, MainAxesSize, MainFigureinitPos
3. Read the audio files in bgmplay, lplay, explplay
4. Declare the figure MainFigureHd1
5. Declare the axes MainAxesHd1
6. Delare the images MainCanvasHd1, SpaceCanvasHd1, Mon1CanvasHd1 - Mon10CanvasHd1
7. Declare the texts GameOverHd1, PressKeyHd1, ScoreInfoForeHd1, ScoreInfoHd1
8. Read the images in da, sp, mon, bul
9. Set the images in MainCanvasHd1, SpaceCanvasHd1, Mon1CanvasHd1 - Mon10CanvasHd1
10. Load avds.mat

```
11.bdir, bullets = 0
12.s_bul = 1
13.mdir = ones(1, 10)
14.runloop = true
15.While runloop is true
16.   If new_start_game == 1
17.     Reset the value of new_start_game to 0
18.     Set GameOverHd1, ScoreInfoHd1, PressKeyHd1 to off
19.     While runloop is true
20.       Play the background sound
21.       Change the position of the spaceship
22.       Reset the value of direction to 0
23.       If fire == 1
24.         bdir = 1
25.         bullets = bullets +1
26.        Take the position of the bullet from the position of the
    spaceship
27.         Declare the image BulCanvasHd1 (bullets)
28.         Set the images in BulCanvasHd1 (bullets)
29.         bul_stat = 1
30.         Reset the value of fire to 0
31.         Play the laser sound
32.       End
33.       If bul_stat == 1
34.         for i = s_bul : bullets
35.         Change the position of the bullet
36.         Try
37.           If (Bulinit{s_bul}(2, 2))<0
38.             s_bul = s_bul +1
39.           End
40.         Catch
41.         End
42.       End
```

43. For k = 1 : 10
44. Change the position of the monster in one direction on x axis if it’s odd number
45. Change the position of the monster in other direction on x axis if it’s even number
46. Put the value of touch_bottom(k) to 1 if the monster reaches the bottom
47. End
48. For i = s_bul : bullets
49. For j = 1 : 10
50. Check if the bullets hits the monster
51. If so then generate new position for the monster
52. Reset the position of the bullet
53. Score = Score +1
54. Play the collision sound
55. break
56. End
57. End
58. End
59. If sum(touch_bottom)
60. Set GameOverHd1, ScoreInfoHd1, PressKeyHd1 to visible on
61. Print the current score and high score
62. Update the high score if current score is higher
63. Call restart_game
64. Break
65. Else
66. Set the current positions in SpaceCanvasHd1, Mon1CanvasHd1 - Mon10CanvasHd1
67. Set the current score in ScoreInfoForeHd1
68. Draw
69. End
70. End
71. End
72. Draw

73. End
74. Function stl_KeyPressFcn (hObject, eventdata)
75. Declare direction, fire, new_start_game
76. CurKey = get (hObject, 'CurrentKey')
77. Switch(CurKey)
78. Case rightarrow put direction = -1
79. Case leftarrow put direction = 1
80. Case space put fire = 1 and new_start_game = 1
81. End
82. End
83. Function stl_CloseReqFcn (hObject, eventdata)
84. Declare runloop
85. runloop = false
86. Close all
87. Clear all
88. Clc
89. Return
90. End
91. Function restart_game
92. Declare Spaceinit, Monx, Mony, Moninit, Bulinit, fire, direction, bul_stat, touch_bottom, score, new_start_game
93. Put an initial value for Spaceinit
94. Generate random number for Monx and Mony
95. Put the value of Monx and Mony in Moninit
96. fire, direction, bul_stat, score, new_start_game = 0
97. Touch_bottom = zeros(10, 1)
98. Draw
99. End

Main Code:

The main portion of the code is described below:

1. **Firing and movement of bullet:**

```
if fire==1
    bdir=1;bullets=bullets+1;
    Bulinit{bullets}(:,:)=Spaceinit(:,:)+[35 -35; 0 -46];
    BulCanvasHd1(bullets)=image(Bulinit{bullets}(1,:),Bulinit{bullets}(2,:),[], ...
        'Parent', MainAxesHd1, ...
        'Visible', 'on');
    set(BulCanvasHd1(bullets), 'CData', bul( :, :, :));
    bul_stat=1;
    fire=0;
    play(lplay,Fs);
end


if bul_stat==1

    for i=s_bul:bullets
        Bulinit{i}(2,:)=Bulinit{i}(2,:)-bdir*15;
        set(BulCanvasHd1(i), 'YData', Bulinit{i}(2,:));
    end
    try
        if (Bulinit{s_bul}(2,2))<0
            s_bul=s_bul+1;
        end
    catch
    end
```

2. **Function for taking keyboard input:**

```
function stl_KeyPressFcn(hObject, eventdata)
global direction fire new_start_game;
curKey = get(hObject, 'CurrentKey');
switch(curKey)
    case 'rightarrow'
        direction=-1;
    case 'space'
        fire=1;
        new_start_game=1;
    case 'leftarrow'
        direction=1;
end
end
```

3. Movement of monster:

```
for k=1:10
    if mod(k,2)
        if (Moninit{k}(1,1)<0)
            mdir(k)=-1;
            Moninit{k}(2,:)=Moninit{k}(2,:)+15;
        elseif (Moninit{k}(1,2)>800)
            mdir(k)=1;
            Moninit{k}(2,:)=Moninit{k}(2,:)+15;
        end
        Moninit{k}(1,:)=Moninit{k}(1,:)-mdir(k)*10;
    else
        if (Moninit{k}(1,1)<0)
            mdir(k)=1;
            Moninit{k}(2,:)=Moninit{k}(2,:)+15;
        elseif (Moninit{k}(1,2)>800)
            mdir(k)=-1;
            Moninit{k}(2,:)=Moninit{k}(2,:)+15;
        end
        Moninit{k}(1,:)=Moninit{k}(1,:)+mdir(k)*10;
    end
    Moninit{k}(2,:)=Moninit{k}(2,:)+1;
    touch_bottom(k)=Moninit{k}(2,2)>600;
end
```

4. Checking if bullet hits the monster:

```
for i=s_bul:bullets
    for j=1:10
        % check if bullet meets the monster
        if ((Moninit{j}(2,2)>Bulinit{i}(2,1))&&(Moninit{j}(2,1)<Bulinit{i}(2,2))&&(Moninit{j}(1,2)>Bulinit{i}(1,1))&&(Moninit{j}(1,1)<Bulinit{i}(1,2)))
            Moninit{j}=[randperm(800,1);randperm(100,1)]+[0 43; 0 43];
            score=score+1;
            Bulinit{i}(2,:)=[-63 0];
            play(explplay,Fs); % play collision sound
            break;
        end

    end
end
```

5. Checking if monster reaches bottom:

```
if sum(touch_bottom)

    set(GameOverHd1, 'Visible', 'on');
    score_report = {sprintf('Score: %d', score), sprintf('Best: %d', Best)};
    set(ScoreInfoHd1, 'Visible', 'on', 'String', score_report);
    set(PressKeyHd1, 'Visible', 'on');
    if score>Best
        Best = score;
        save avds.mat Best -append
    end
    restart_game;

    break;
```

6. Function to restart the game:

```
function restart_game
load avds.mat
global Spaceinit Monx Mony Moninit fire direction bul_stat touch_bottom score new_start_game;
Spaceinit=[369 432; 520 583];
Monx=randperm(800,10);
Mony=randperm(100,10);
for i=1:10
    Moninit{i}=[Monx(i);Mony(i)]+[0 43; 0 43];
end
fire=0;
direction=0;
bul_stat=0;
touch_bottom=zeros(10,1);
score=0;
new_start_game=0;
drawnow();
end
```

Some Test Cases:

We have selected 10 monsters in the screen. So each time we will find only 10 monsters at a time. The following window will open when we run the program:

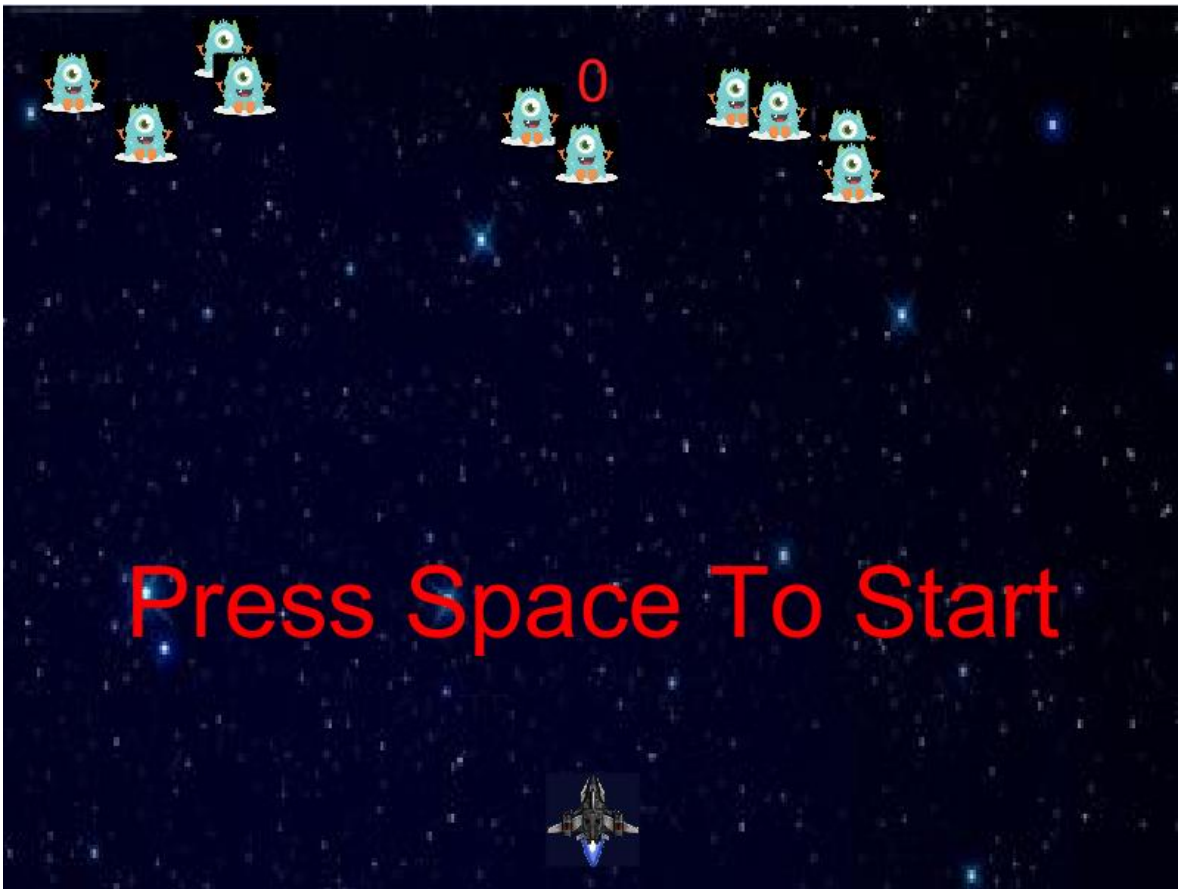


Fig: Game window at the beginning

By pressing ‘Spacebar’, the game will start. The monsters will move and come down. We can use left and right arrow buttons to control the spaceship. The background music will play continuously after starting the game. For each fire of the bullet, there will be specific sound.

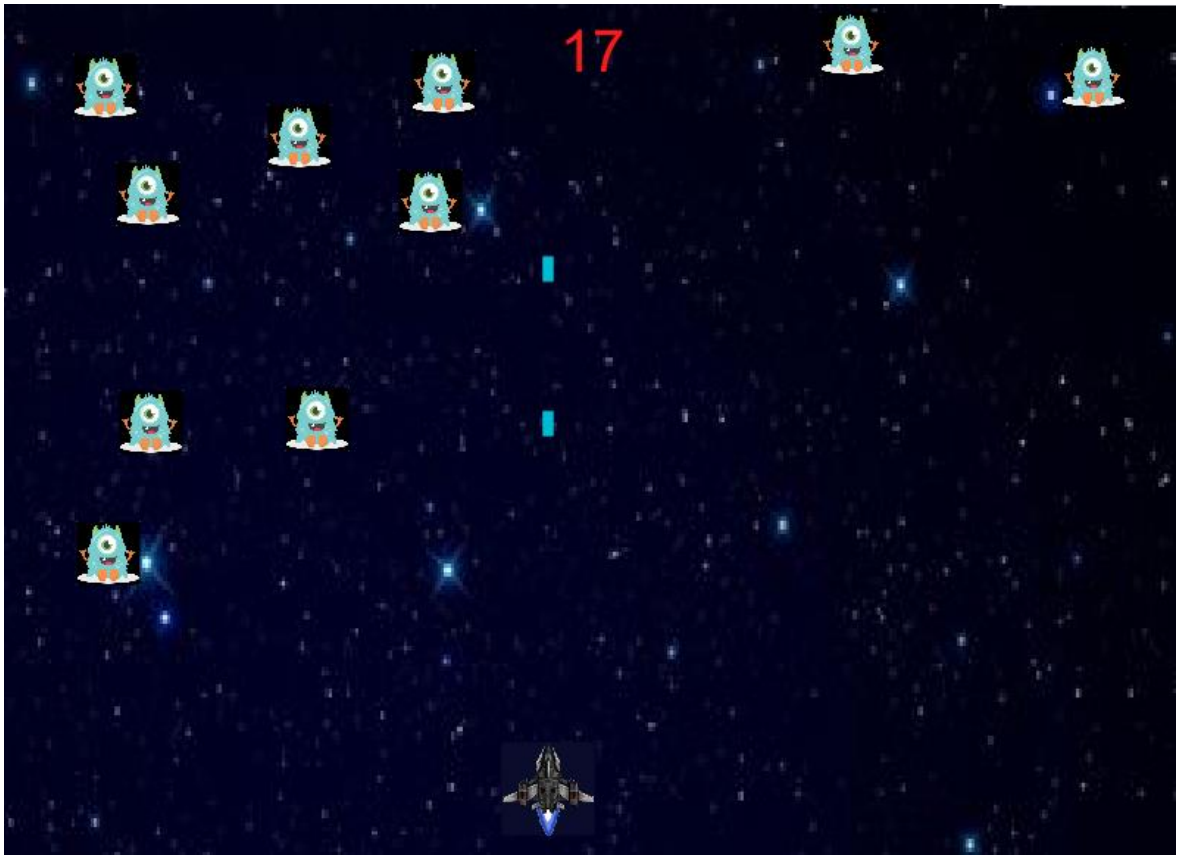


Fig: Game window at the middle of the game

Every time a bullet hits the monster, the monster will disappear with a specific explosive sound and a new monster will generate at the random place. And it will increase the score at the same time.

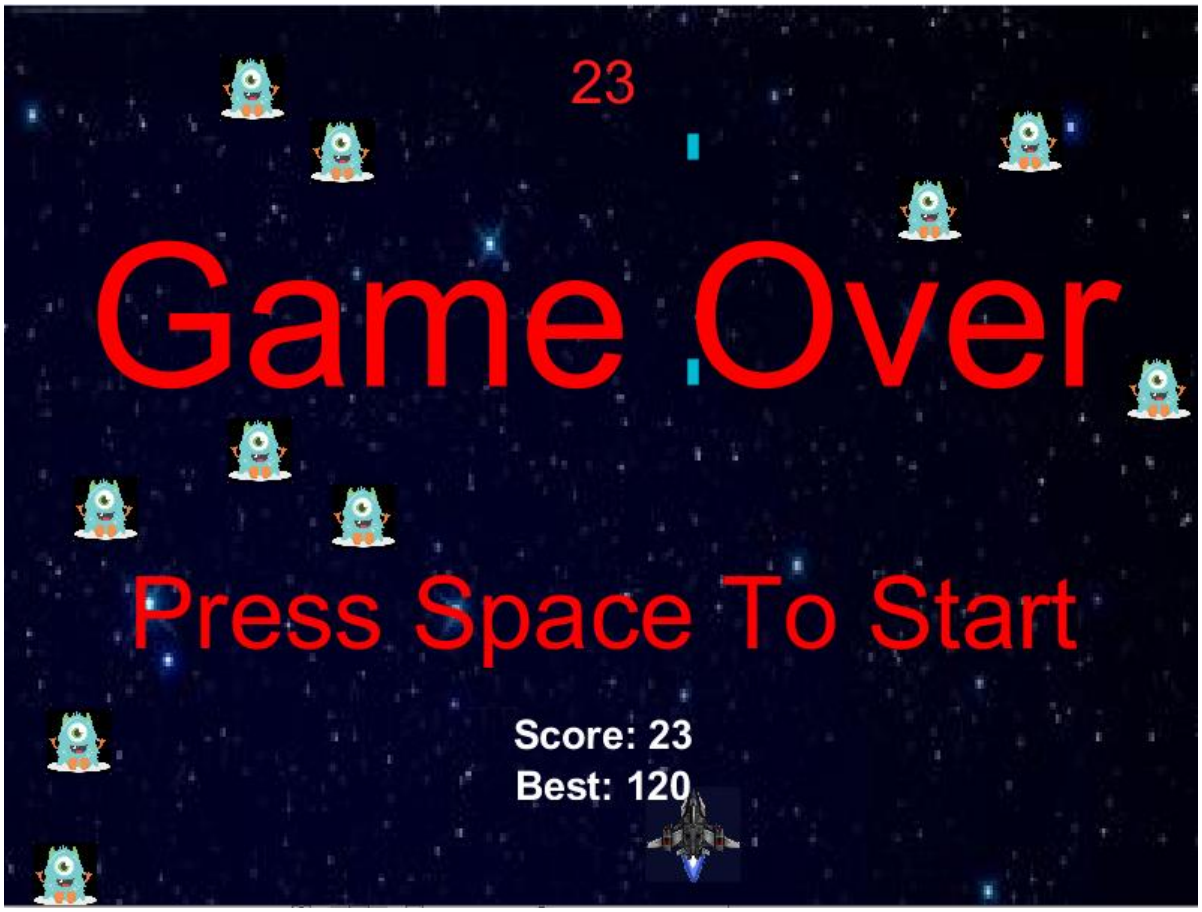


Fig: Game window at the end of the game.

The game ends when any of the monsters touches the bottom part of the screen. The above window will open when the game ends. The window will display 'Game Over' and show the best and the current scores. The game restarts by pressing spacebar. For the new high score, the score will be preserved in advs.mat file.

Application:

Kids around the globe are fond of this type of games. Though this game is already invented, we successfully build the algorithm of the game using Matlab only. However, the knowledge earned by working with this game will be helpful for our further ability to create something new. Games are one of the most popular way of entertainment. Day by day, it's market value is increasing as well. Recently a cricket game was designed by some Bangladeshi programmers, which was the

first cricket game by any Bangladeshi. And we still take pride on that. With our project, we could gain knowledge about the logics and algorithms behind the games.

Conclusion:

Matlab is an interesting tool. We used advs.mat file to store the workspace variables. We have generated some functions and variables to make the game. This project will help us to understand the code and will massively influence us to make a new game again. The game might be improved by using better background image and music. However, by using all the functions and variables properly, we developed the program successfully.