DODGER

A mini project report submitted for pre final year of

Bachelor of Technology

in

Computer Science and Engineering

By

P.Vasavi (N140462)

P.PratyushaRamya (N140903)

V.PremKumar (N140577)

M.GopiChand (N140628)

Under the Supervision of

Mr. N.RamaKrishna

Department of Computer Science and Engineering

Rajiv Gandhi University of Knowledge Technologies – Nuzvid,

Krishna – 521202

November, 2018

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES

(A.P. Government Act 18 of 2008)

RGUKT-NUZVID, Krishna Dist - 521202

Tele Fax: 08656 – 235557/235150

CERTIFICATE OF COMPLETION

This is to certify that the work entitled, "**DODGER"** is the bonafied work of ***Vasavi , ID No: N140462, PratyushaRamya , ID No: N140903 , PremKumar , ID No: N140577 , GopiChand , ID No: N140628***carried out under my guidance and supervision for pre finalyear project of **Bachelor of Technology** in the department of Computer Science and Engineering under RGUKT IIIT Nuzvid. This work is done during the academic session August 2018– December 2018, under our guidance

---

Mr. N.RamaKrishna

Project Supervisor

Lecturer Dept. of CSE

RGUKT IIIT Nuzvid, Nuzvid

Mr. Upendar

Head of Department

Assistant Professor Dept. of CSE

RGUKT IIIT Nuzvid, Nuzvid

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES

(A.P. Government Act 18 of 2008)

RGUKT-NUZVID, Krishna Dist - 521202

Tele Fax : 08656 – 235557/235150

CERTIFICATE OF EXAMINATION

This is to certify that the work entitled, "**DODGER"** is the bonafied work of ***Vasavi , ID No: N140462, PratyushaRamya , ID No: N140903 , PremKumar , ID No: N140577 , GopiChand , ID No: N140628***and here by accord our approval of it as a study carried out and presentedin a manner required for its acceptance in pre final year of **Bachelor of Technology** for which it has been submitted. This approval does not necessarily endorse or accept every statement made, opinion expressed or conclusion drawn, as a recorded in this thesis. It only signifies the acceptance of this thesis for the purpose for which it has been submitted.

Mr. N.RamaKrishna

Project Supervisor

Lecturer Dept. of CSE

RGUKT IIIT Nuzvid, Nuzvid

Examiner

Project Examiner

Lecturer Dept. of CSE

RGUKT IIIT Nuzvid, Nuzvid

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES

(A.P. Government Act 18 of 2008)

RGUKT-NUZVID, Krishna Dist - 521202

Tele Fax : 08656 – 235557/235150

DECLARATION

We, ***Vasavi , ID No: N140462, PratyushaRamya , ID No: N140903 , PremKumar , ID No: N140577 , GopiChand , ID No: N140628***hereby declare thatthe project report entitle "**DODGER"**done by us under the guidance of **Mr. N.RamaKrishna M.Tech,** is submitted for pre final year of **Bachelor of Technology** in **Computer Science and Engineering** the academic session August2018-December 2018 at RGUKT – Nuzvid.

We also declare that this project is a result of our own effort and has not been copied or imitated from any source. Citations from any websites are mentioned in the references.

The results embodied in this project report have not been submitted to any other university or institute for the award of any degree or diploma.

Date: 14-11-2018

Place: Nuzvid

Vasavi [N140462]

PratyushaRamya [N140903]

PremKumar [N140577]

GopiChand [N140628]

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES

(A.P. Government Act 18 of 2008)

RGUKT-NUZVID, Krishna Dist - 521202

Tele Fax : 08656 – 235557/235150

ACKNOWLEDGEMENT

The success in this project would not have been possible but for the timely help and guidance rendered by many people. I wish to express my sincere thanks to all those who has assisted me in one way or the other for the completion of my project.

I thank to my project guide Mr **N.RamaKrishna** sir, Assistant Professor Department of Computer Science and Systems Engineering for guiding me all through the project works giving a right direction and shape to my learning by extending his expertise and experience inthe education. Really I'm indebted for his excellent and enlightened guidance.

It is my privilege to express deepest gratitude to Mr **R.Upendar Rao**, *Head of the Department*, for his valuable suggestions and constant motivation that greatly helped the project to successfully complete.

I thank **Prof. Venkata Dasu Veeranki** , *Director of rgukt*, for his support and to all the teaching, non-teaching staff of the Department of CSE who gave all possible help to bring project work to the present shape.

I thank all who contributed directly or indirectly in successfully carrying out this work.

P.Vasavi

P.PratyushaRamya

V.PremKumar

M.GopiChand

Table of contents:

List of figures

ABSTRACT

Now a day's games are very popular .Games are useful for mind relaxation and improves our logical thinking power. In internet so many computer based games are exist. Apart from physical games people are migrated to computer based games and mobile based games also. Especially Video games are easy to play but in case of development it was quite difficult. For developing a video game it requires player and environment for game play. Every one interested about to play the game but no one will think about what happened in the internal process of in it. Games have the environment like 2d or 3d animations which they have creating more interest to the players. Dodger game is a one of the video game. It has one player and bunch of objects .In this player used to avoid to touch the objects which was fallen from top of the screen. If player touch any object game will be over.

1. INTRODUCTION

In our project we developed the dodger game. It is a 2d animated video game. We developed this game using python language. This game is player controlled game. Player object called OFO (i.e player object name is UFO) will be moving on the surface which was controlled by the user(i.e. player) and it is avoid to touch the objects called asteroids(I.e. objects name is asteroids) which was fall from the top of the screen surface. Score is increased by frames per second. When the UFO is touches any asteroid then game will be over. By increasing the levels the rate of asteroids speed will be increases. It is very interesting and excited game .when speed is increases avoidance of touching asteroids is difficult. It makes to give an excitement to the user. Every level game player objects, backgrounds and asteroids war changes .If player wants break some time and then play for that purpose we add pose button .whenever player poses the game it will pose. When game is over at that time score will be the top score .Every time top score will updated with new high score. In this game we add background music. This game gives the good accuracy and interesting for the users. Users will more excite towards this game.

2. PROBLEM DEFINITION

2.1) Existed Game:

In this system game is existed like a moving object and a few objects are fallen from the top of the screen and player has to avoid touching the objects which was fallen from the screen top.

2.1.1) Drawbacks:

- GameOvers when player touches enemy Dimensions
- It doesn't have **backgroundimages**
- Speed and Count of the **fallingenemies** were **constant**
- Single enemy and player image
- Enemy falling stops after some time
- No **User** defined Size
- No **Pause** option
- No Score showing GameOver

2.2) Proposed Game:

By overcoming the limitations from the existing system we proposed the system game is increasing of each level there should be an eventual increasing of speed of objects fallen from top of the screen.In our project score will be increases by winning of every level. Player used to avoid the collision of the objects (i.e. asteroids). In this objects would be in different sizes like small objects to bigger objects.

2.2.1) Features:

- Pixel collision is added to the Game
- Added different background Images
- Top score
- Speed and Count of **asteroids** are increased by score
- Different **enemy** Images and **Player** images
- User Defined Size
- **Pause** option to pause the Game
- Showing Score After GameOver

2.3) Functional Requirement:

In Software Engineering and System Engineering, a **functional requirement** defines a function of a System or its component, where a function is described as a specification of behavior between outputs and inputs.

Requirements:

- The Game Developed using Python Language
- The Game will be controlled by Keyboard
- Game display Based on User Option
- The Game featured with music
- The Game has more than one level
- Player can move up,down,right,left using arrow keys
- Scoring is based on how many asteroids are crossed
- A sound will play as background
- A sound will play when player hits asteroid

2.4) Non-Functional Requirements:

A non-functional requirement (NFR) is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors.

Requirements:

- Frame Rate: No of Frame per second is 30
- User Clicks : Needs single Click to move player
- Response Time : Based on Frames per Second
- Required Resources :
  - Needs minimum of 64mb RAM
  - Needs minimum of 100mb Hard Disk Space
- Platform:
  - Any OS which can support python
  - Pygame package must be installed

2.5) Hardware Requirements:

- 512MB RAM
- 20GB Hard Disk Space
- Keyboard
- Nice Looking Display
- 32-bit / 64-bit BUS
- Minimum Intel i3 Processor

2.6) Software Requirements:

- Window / Linux / Mac
- Python installed OS
- Pygame Package

2.7) Packages Required:

- pygame: This is used to initialize the game. This can be used to set display size, Game initializing, diplay updation and more
- random: This contains methods to select random values, choices and etc.
- ctypes: A foreign function library for Python. **ctypes** is a foreign function library for Python. It provides C compatible data types, and allows calling functions in DLLs or shared libraries. It can be **used** to wrap these libraries in pure Python. Here we used to get system screen dimensions.
- sys: This **module** provides access to some variables **used** or maintained by the interpreter and to functions that interact strongly with the interpreter.

3. DESIGN

3.1) PIXEL COLLISION :

Pixel Perfect Collision (PPC) is a precise type of 2D**collision** detection that uses bit-masks to determine whether two objects **collide or not.**

3.1.1) Description:

Bounding boxes are used by many games to detect if two things collide. Either a rectangle, a circle, a box or a sphere are used as a crude way to check if two things collide. However for many games that just isn't enough. Players can see that something didn't collide, so they are going to be crying foul if you just use bounding boxes.

Why rectangles aren't good enough: Here you can see a balloon, and a cave. The idea is you have to move the baloon through the cave without hitting the walls. Now if you used just bounding rectangle collisions, you will see how it would not work, and how the game would be no fun - because the rectangle(drawn in green around the balloon) would hit the sides when the balloon didn't really hit the sides.

Fig:3.1.1 bounding rectangle collisions

3.1.2)How is pixel perfect collision detection done? Masks:

Instead of using 8-32bits per pixel, pygames masks use only 1 bit per pixel. This makes it very quick to check for collisions. As you can compare 32 pixels with one integer compare. Masks use bounding box collision first - to speed things up.

Even though bounding boxes are a crude approximation for collisions, they are faster than using bitmasks. So pygame first does a check to see if the rectangles collide - then if the rectangles do collide, only then does it check to see if the pixels collide.

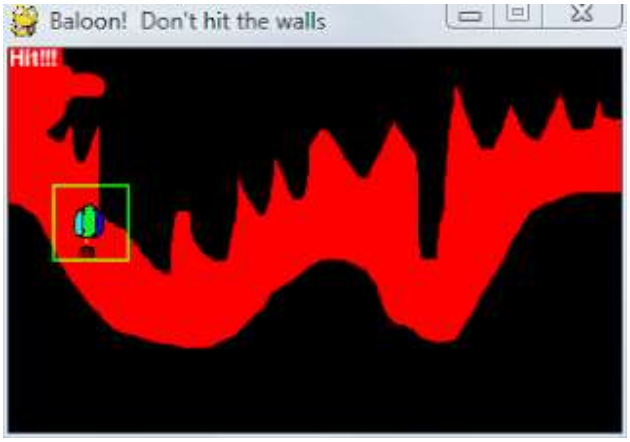


Fig:3.2.2 Pixel Collided image

3.2) How to use pixel perfect collision detection in pygame?

There are a couple of ways you can use pixel perfect collision detection with pygame.

- Creating masks from surfaces.
- Using the pygame.sprite classes.

You can create a mask from any surfaces with transparency. So you load up your images normally, and then create the masks for them.
Or you can use the pygame.sprite classes, which handle some of the complexity for you.

3.2.1) Mask.from_surface with Alpha transparency:

By default pygame uses either color keys, or per pixel alpha values to see which parts of an image are converted into the mask.

Color keyed images have either 100% transparent or fully visible pixels. Where as per pixel alpha images have 255 levels of transparency. By default pygame uses 50% transparent pixels as on, or ones that are to collide with.

It's a good idea to pre-calcuate the mask, so you do not need to generate it every frame.

3.2.2) Checking if one mask overlaps another mask:

It is fairly simple to see if one mask overlaps another mask.

Say we have two masks (a and b), and also a rect for where each of the masks is.

```
#We calculate the offset of the second mask relative to the first mask.
offset_x = a_rect[0] - b_rect[0]
offset_y = a_rect[1] - b_rect[1]
# See if the two masks at the offset are overlapping.
overlap = a.overlap(b, (offset_x, offset_y))
if overlap:
   print "the two masks overlap!"
```

3.2.3) Pixel perfect collision detection with pygame.sprite classes.

The pygame.sprite classes are a high level way to display your images. They provide things like collision detection, layers, groups and lots of other goodies. If you give your sprites a .mask attribute then they can use the built in collision detection functions that come with pygam'e.sprite.

3.3) Flow Charts:

3.3.1) To set Frame:

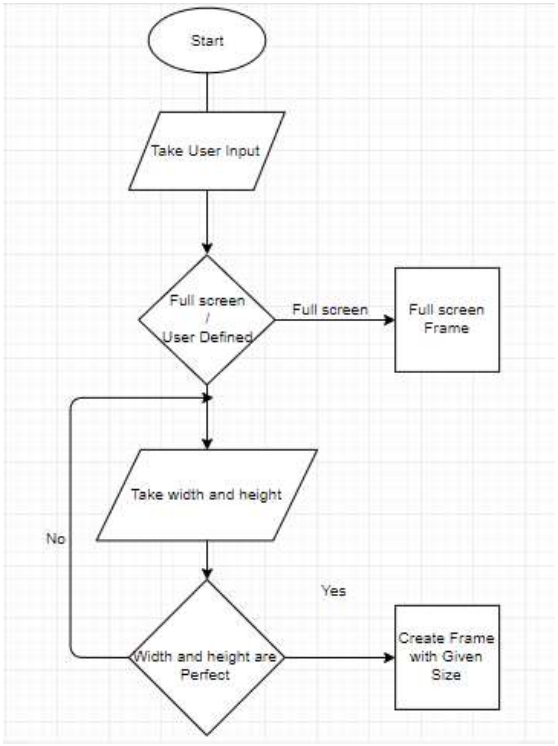


Fig:3.3.1 to set frame

3.3.2)Frame Loop:

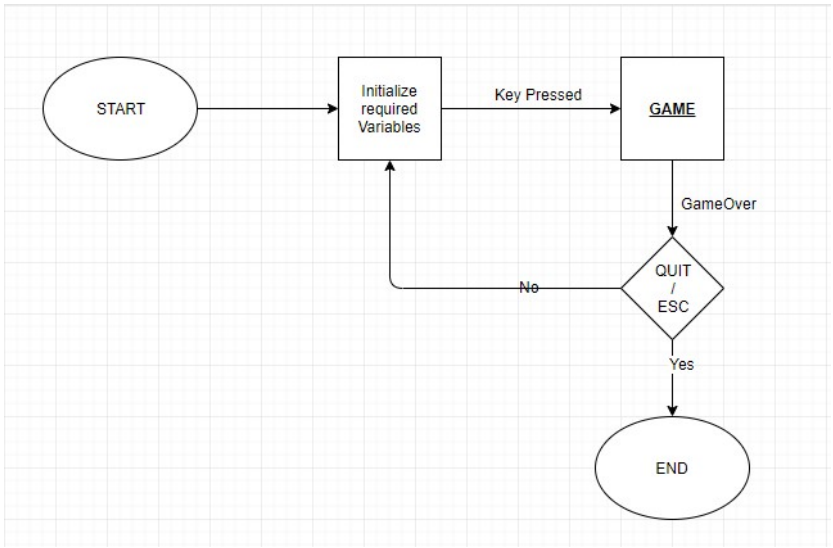


Fig:3.3.2 frame loop

Game:

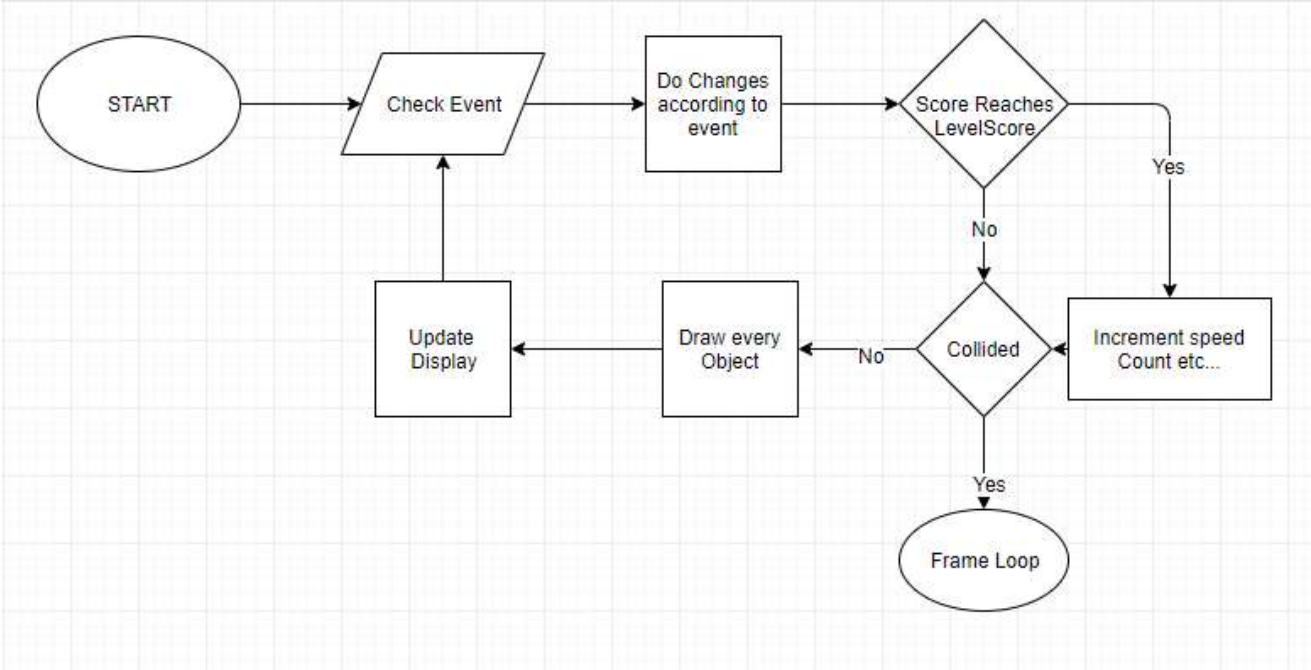
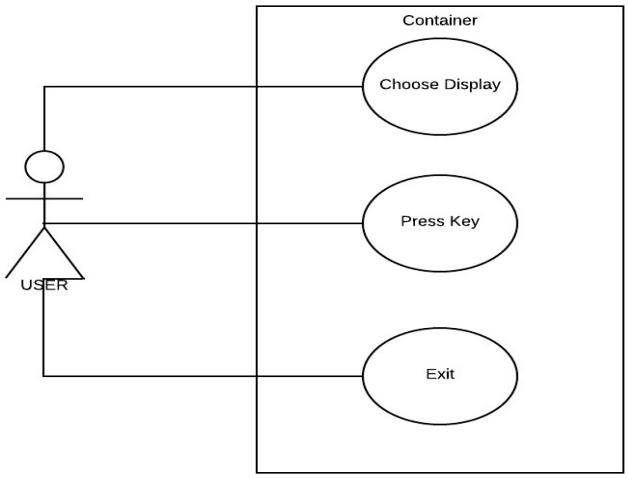


Fig:3.3.3 Game

UML Diagrams

3.5)DODGER Use Case Diagram:



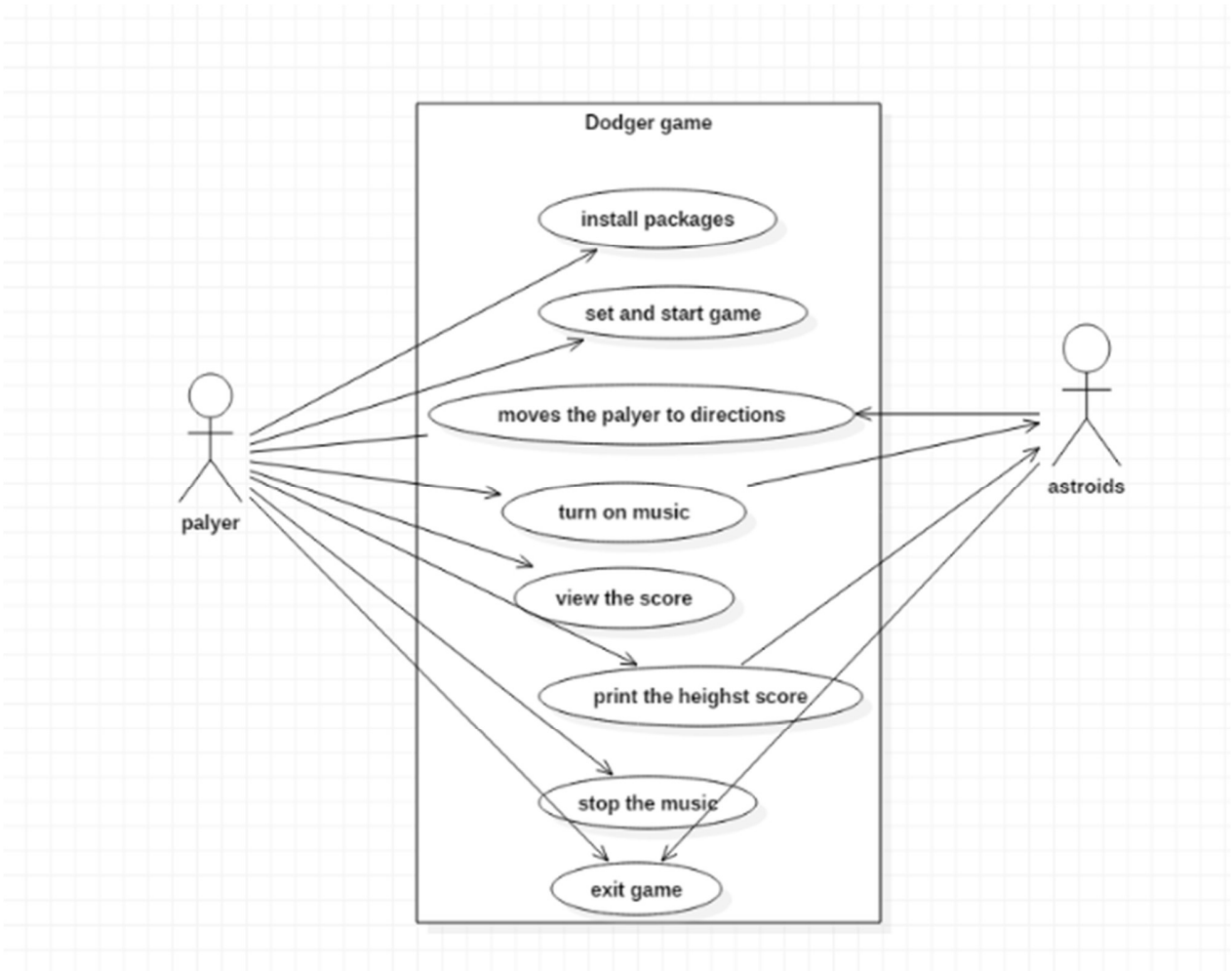


Fig:3.5 dodger use case diagram

3.6) Sequence Diagram:

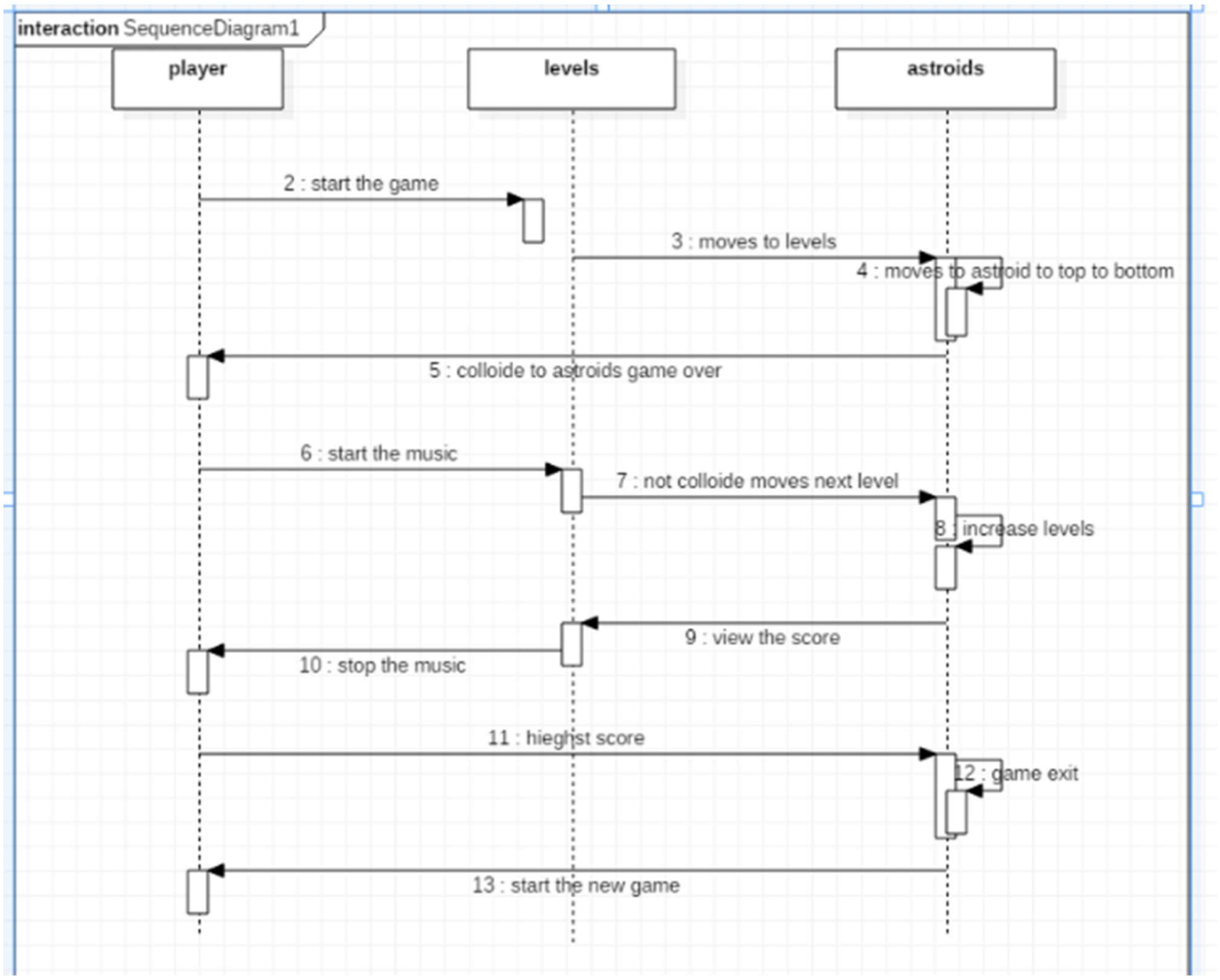


Fig:3.6 sequence diagram

4. IMPLEMENTATION

4.1) Modules:

1) *Display window size*
2) *Player*
3) *Asteroids*
4) *Levels*
5) *Background*
6) *Music*
7) *Score*

4.2) Modules description:

1. Display window size:

In this module user have a display window, in that he can choose display window size .Some players wants full screen they can choose full screen option. In case of player doesn't know the pixels of the screen if he given width and height is more than the display window size then it will generate reenter option that means it asks again for window size.

2. Player:

Player module gives the player has to move on the screen surface. By increasing the levels every time player object image replaced with another object image. Player moves on the surface up, down, left, right. The player has to avoid collide of asteroids. Player object images are UFO's which was controlled by the user (i.e. player). Player move rate is constant

3. Asteroids:

Asteroids are the modules which are fallen from top of the screen surface. In this module we have a different kind of asteroids are there .Every level asteroids images will be changed and rate of fallen down speed will be increases. They are in different type of sizes also. A big and very small asteroids are also been exist in this game. Size of the asteroids are taken randomly by using asteroid minimum size and maximum size values. We have a King steroid at certain score

4. Levels:

This game has the levels. By avoiding the collision of the asteroid which was done by the player is continuous process so at certain situation of the score of asteroid the level will be increases. Every level score and asteroid images and UFOs images will be updated.

Playing more levels it was quite interesting because speed will be increases then asteroids down rate will be increases collision will be occurring in any time.

5. Background:

A game must have backgrounds. Every level it changed to different type of backgrounds in it. The backgrounds are look like space based images. Especially it is a space based game so asteroids are fallen from the space and UFOs should avoid moving and surviving on the space.

6. Music:

While playing game sound is very important. Because attraction towards game has important thing is music. When starting a game with music according to designing of backgrounds and asteroids is give anxiety to the player who plays the game. Music was eventually increases when starting of the game .when pause button was on then music will stop. If we unpause the game music will start again. In this we have two music. One is for background and another is for GameOver sound.

7. Score:

Every player focuses on the score because earning of score as the main criteria of my project. How many asteroids are avoided by the player that is going to be added into the score. Every level score will be increases. When UFO got touched by any asteroid game will be over then score will updated with top score. Score will be updated by comparing the top score. If score is less than the top score it will not be updated. If it is higher then top score then it will be the top score. This process is done until game was closed by the user.

5.Code for the game

```
import pygame,random,sys,os
from pygame.locals import *
print("(1) FULLSCREEN\n(2) Manual Size")
option=int(input("Enter Option : "))
import ctypes
user=ctypes.windll.user32
Maxwidth=user.GetSystemMetrics(0)
Maxheight=user.GetSystemMetrics(1)
del ctypes
width,height=0,0
clock = pygame.time.Clock()
FPS=25
if(option==2):
    while(width<400 or height<400):
        print("(400<WIDTH<=%d and 400<HEIGHT<=%d"%(Maxwidth,Maxheight))
        width=int(input("Width : "))
        height=int(input("Height : "))
        if(width>Maxwidth or height>Maxheight):
            width=0
            height=0
                # ---------- Methods ---------- #
# Method for Terminate the Game
def terminate():
    pygame.quit()
    sys.exit()
# Method to Whether Player pressing any Key to Wait
```

```
def waitForPlayerToPressKey():
    while True:
        for event in pygame.event.get():
            if(event.type==QUIT):
                terminate()
            if(event.type==KEYDOWN):
                if(event.key==K_ESCAPE):
                    terminate()
                return
# Method To draw Text one Screen
def drawText(text, font, surface, x, y,center,color):
    textobj = font.render(text, 1, color)
    textrect = textobj.get_rect()
    if(center):
        textrect.center = (int(x), int(y))
    else:
        textrect.topleft= (int(x), int(y))
surface.blit(textobj, textrect)
# Method to pause Game
def pauseGame(Game):
    while True:

drawText("PAUSED",pygame.font.SysFont(None,int(width/18)),Game,width/2,height/2,True,(51,255,51))
        drawText("Press \"P\" to Continue",pygame.font.SysFont(None,int(width/40)),Game,width/2,height/2+height/15,True,(255,255,255))
        for event in pygame.event.get():
            if(event.type==QUIT): terminate()
```

```
            if(event.type==KEYUP):
                if(event.key==K_ESCAPE): terminate()
                if event.key==K_p:
                    return False
        pygame.display.update()
        clock.tick(1)
            # ---------------- DISPLAY ----------------------- #
pygame.init()
if(option==1):
    flags= pygame.FULLSCREEN | pygame.DOUBLEBUF
    Game=pygame.display.set_mode((0, 0),flags)
    width,height=Maxwidth,Maxheight
elif(option==2):
    Game=pygame.display.set_mode((width,height),pygame.DOUBLEBUF)
del option


            # ---------------- COLORS ----------------------- #
black=(0,0,0)
white=(255,255,255)
red=(255,0,0)
green=(0,255,0)
blue=(0,0,255)
            # ---------------- SIZES ----------------------- #
AstMinSize=int(width/60)
AstMaxSize=int(width/25)
PlayerMoveRate=int(width//100)
            # ---------------- PLAYER CREATING CLASS ----------------------- #
```

```
class Player(pygame.sprite.Sprite):
    def __init__(self, player,screen,begin,playerPos=None):
        super(Player, self).__init__()
        self.image = player
        self.dimension=(int(height/15),int(height/15))
        self.image = pygame.transform.scale(self.image,self.dimension)
        self.imrect=self.image.get_rect()
        if(begin):
            self.position=((width-self.imrect.width)/2,(height-self.imrect.height))
        else:
            self.position=playerPos
        screen.blit(self.image,self.position)
        self.rect = self.image.get_rect(center=self.position)
        self.mask = pygame.mask.from_surface(self.image)


            # ---------------- ENEMY CREATING CLASS ---------------------- #
class Enemy(pygame.sprite.Sprite):
    def __init__(self, image,screen,minim,maxim,king):
        super(Enemy, self).__init__()
        self.image = image
        if(king):
            self.enemySize=int(height/7)
        else:
            self.enemySize=random.randint(AstMinSize,AstMaxSize)
        self.image=pygame.transform.scale(self.image,(self.enemySize,self.enemySize))
        self.position=(random.randint(0, width-self.enemySize), 0 - self.enemySize)
        screen.blit(self.image,self.position)
```

```
        self.speed=random.randint(minim,maxim)
        if(king):
            self.speed=maxim-minim
        self.rect = self.image.get_rect(center=self.position)
        self.mask = pygame.mask.from_surface(self.image)
            # ---------- FRAME CAPTION ---------- #
pygame.display.set_caption("DODGER")
            # ---------- SOUNDS ---------- #
gameOver=pygame.mixer.Sound("gameover.wav")
pygame.mixer.music.load("bgm.mid")
            # ---------- IMAGES ---------- #

            # ---------- UFO ---------- #
ufo=[pygame.image.load("ufo/1.png")]
ufo.append(pygame.image.load("ufo/2.png"))
ufo.append(pygame.image.load("ufo/3.png"))
ufo.append(pygame.image.load("ufo/4.png"))
ufo.append(pygame.image.load("ufo/5.png"))
ufo.append(pygame.image.load("ufo/6.png"))
            # ---------- Asteroids ---------- #
aster=[pygame.image.load("asteroids/1.png")]
aster.append(pygame.image.load("asteroids/2.png"))
aster.append(pygame.image.load("asteroids/3.png"))
aster.append(pygame.image.load("asteroids/4.png"))
aster.append(pygame.image.load("asteroids/5.png"))
aster.append(pygame.image.load("asteroids/6.png"))
            # ---------- Background ---------- #
```

```
backgrounds=[pygame.image.load("bg/1.png")]
backgrounds.append(pygame.image.load("bg/2.jpg"))
backgrounds.append(pygame.image.load("bg/3.jpg"))
backgrounds.append(pygame.image.load("bg/4.png"))
backgrounds.append(pygame.image.load("bg/5.png"))
backgrounds.append(pygame.image.load("bg/6.png"))



    # ---------- STARTING THE GAME---------- #
if(width==height):
   startBg=pygame.transform.scale(pygame.image.load("sameBg.png"),(width,height))
else:
   startBg=pygame.transform.scale(pygame.image.load("diffBg.png"),(width,height))
Game.blit(startBg,(0,0))
drawText('DODGER', pygame.font.SysFont(None, int(width/15),True),
Game,width/2,height/2,True,(0,255,255))
drawText('Press a key to start', pygame.font.SysFont(None, int(width/20)), Game, width/2, (height / 2) +
height//10,True,white)
pygame.display.update()
waitForPlayerToPressKey()
del startBg


ScoreFontSize=0
if(width<Maxwidth/2):
   ScoreFontSize=int(width/20)
else:
   ScoreFontSize=int(width/50)
```

```
topScore=0
while True:
    bg=pygame.transform.scale(backgrounds[0],(width,height))
    bg2=pygame.transform.rotate(bg,180)
    bgx=0
    bgy=height
    AstMinSpeed=3
    AstMaxSpeed=10
    NewAstRate=12
    ConstAstRate=12
    player=Player(ufo[0],Game,True,None)
    enemies=pygame.sprite.Group(Enemy(aster[0],Game,AstMinSpeed,AstMaxSpeed,False))
    all_enemies=pygame.sprite.Group(player,enemies)
    pause=left=right=up=down=False
    score=0
    level=0
    pygame.mixer.music.play(-1,0.0)
    hit=0
    levelNum=0
    levelInc=False
    levelx=0
    while True:
        score+=1
        for event in pygame.event.get():
            if event.type == QUIT:
                terminate()
            if event.type == KEYDOWN:
```

```
            if event.key == K_LEFT or event.key == K_a:
                right = False
                left = True
            if event.key == K_RIGHT or event.key == K_d:
                left = False
                right = True
            if event.key == K_UP or event.key == K_w:
                down = False
                up = True
            if event.key == K_DOWN or event.key == K_s:
                up = False
                down = True
        if event.type == KEYUP:
            if event.key == K_ESCAPE:
                terminate()
            if event.key==K_p:
                pause=True
                pygame.mixer.music.pause()
                pause=pauseGame(Game)
                pygame.mixer.music.unpause()
            if event.key == K_LEFT or event.key == K_a:
                left = False
            if event.key == K_RIGHT or event.key == K_d:
                right = False
            if event.key == K_UP or event.key == K_w:
                up = False
            if event.key == K_DOWN or event.key == K_s:
```

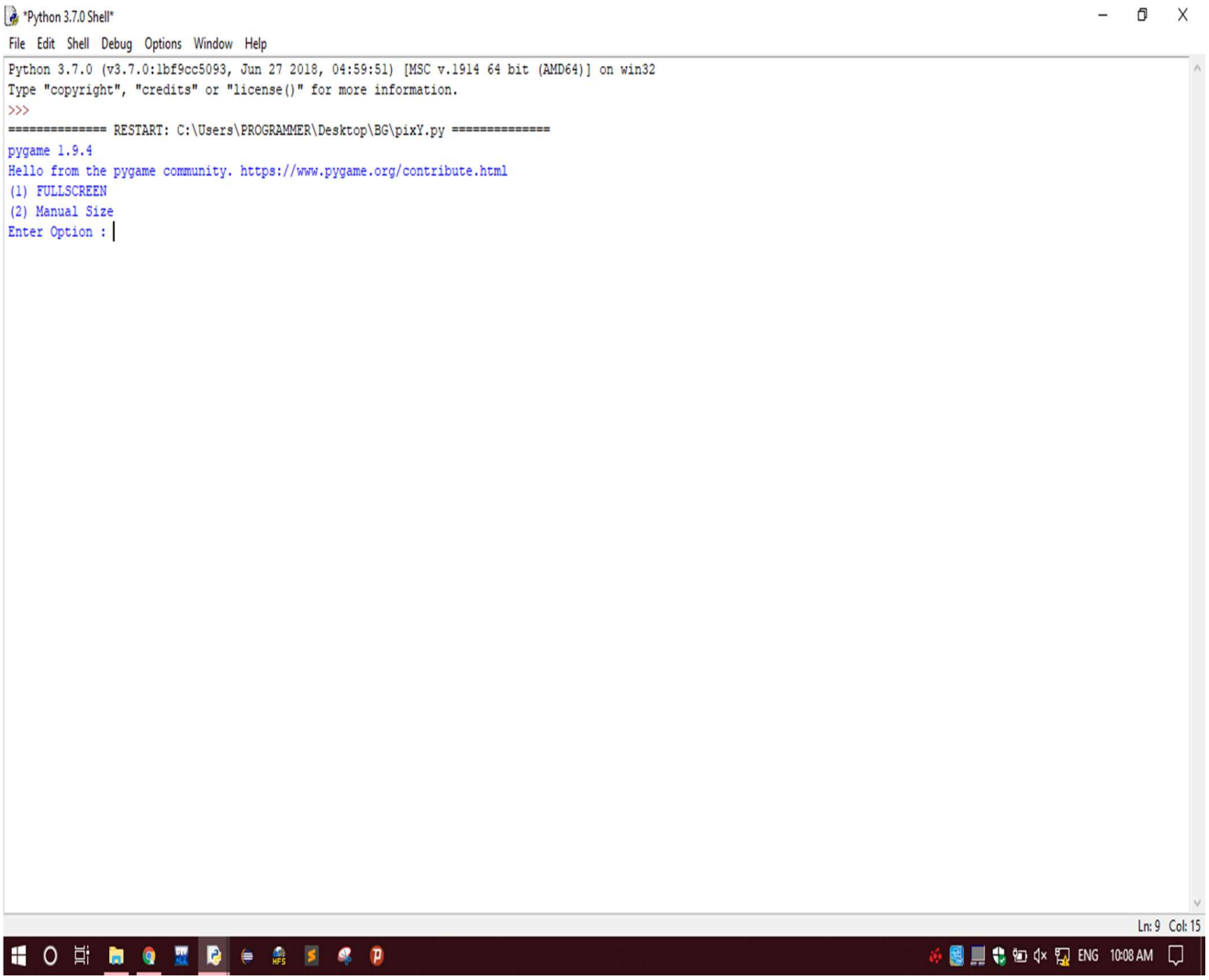
```
            down = False
    poss=player.image.get_rect()
    if left and (player.rect.centerx-(poss.width/2)-PlayerMoveRate)>= 0:
        player.rect.centerx-=PlayerMoveRate
    if right and (player.rect.centerx+(poss.width/2)+PlayerMoveRate)<= width:
        player.rect.centerx+=PlayerMoveRate
    if up and (player.rect.centery-(poss.height/2)-PlayerMoveRate)>= 0:
        player.rect.centery-=PlayerMoveRate
    if down and (player.rect.centery+(poss.height/2)+PlayerMoveRate)<= height:
        player.rect.centery+=PlayerMoveRate
    NewAstRate-=1
    if(score%100==0 and ConstAstRate>3):
        ConstAstRate-=1
    if(score%100==0):
        level+=1
        if(level==6):
            level=0
        player=Player(ufo[level],Game,False,(player.rect.centerx,player.rect.centery))
        AstMaxSpeed+=1

enemies.add(pygame.sprite.Group(Enemy(aster[level],Game,AstMinSpeed,AstMaxSpeed,True)))
        bg=pygame.transform.scale(backgrounds[level],(width,height))
        bg2=bg
        levelNum+=1
        levelInc=True
    if(NewAstRate==0):
        NewAstRate=ConstAstRate
```

```
enemies.add(pygame.sprite.Group(Enemy(aster[level],Game,AstMinSpeed,AstMaxSpeed,False)))
    all_enemies=pygame.sprite.Group(player,enemies)
    #spritecollide(sprite, group, dokill, collided = None)
    #The dokill argument is a bool.
    #If set to True, all Sprites that collide will be removed from the Group.
    if pygame.sprite.spritecollide(player, enemies, False, pygame.sprite.collide_mask):
        if(level%3==0 and hit<5):
            continue
        elif(score>topScore):
            topScore=score
        break
    for e in enemies:
        e.rect.centery+=(e.speed)
        position=e.image.get_rect()
        if((e.rect.centery-position.height/2)>height):
            enemies.remove(e)
    # Method to scroll background
    Game.blit(bg,(0,bgx))
    Game.blit(bg2,(0,bgy))
    if(levelx<=height and levelInc):
        levelx+=int(width/50)
        drawText(("Level
%s"%(str(levelNum))),pygame.font.SysFont("comicsansms",int(height/5)),Game,levelx,int(height/2),Tr
ue,white)
    if(levelx>=height):
        levelx=0
        levelInc=False
    bgx+=PlayerMoveRate
```

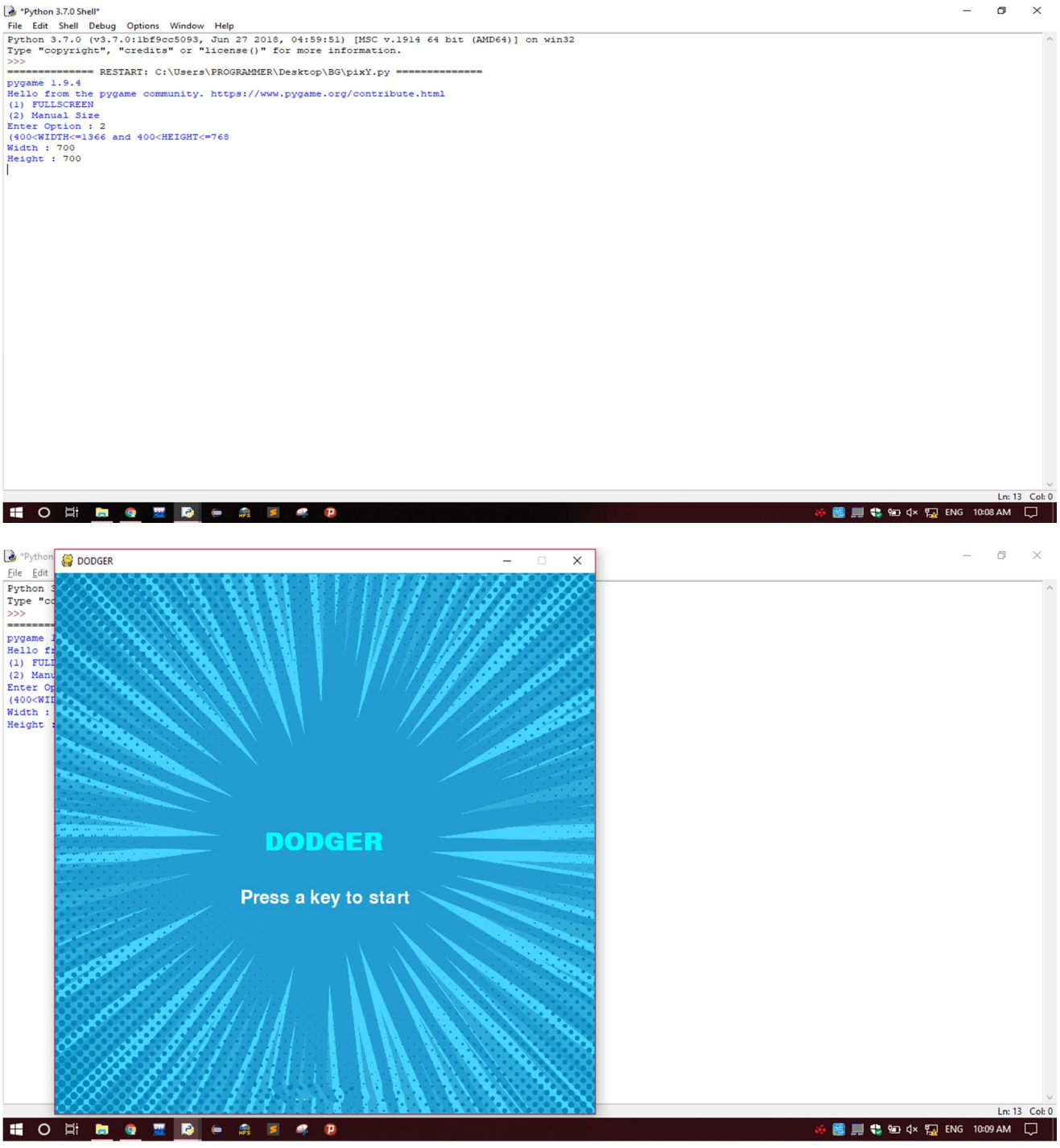
```
        bgy+=PlayerMoveRate
        if(bgx>=height):
            bgx=-1*(height-bgy)
        if(bgy>=height):
            bgy=-1*(height-bgx)
        all_enemies.draw(Game)
        drawText('Score :
%s'%(int(score)),pygame.font.SysFont(None,ScoreFontSize),Game,10,10,False,white)
        drawText('Top Score:
%s'%(int(topScore)),pygame.font.SysFont(None,ScoreFontSize),Game,10,30,False,white)
        pygame.display.flip()
        clock.tick(FPS)
    pygame.mixer.music.stop()
    gameOver.play()
    drawText("%s"%(str(score)),pygame.font.SysFont(None,int(width/10)),Game,(width / 2), (height /
2)-height/10,True,(255,0,43))
    drawText('GAME OVER', pygame.font.SysFont(None, int(width/20)), Game, (width / 2), (height /
2),True,(85,0,255))
    drawText('Press a key to play again', pygame.font.SysFont(None, int(width/20)), Game, (width / 2),
(height / 2)+height/10,True,white)
    pygame.display.update()
    waitForPlayerToPressKey()
    # Stopping GameOver Sound
    gameOver.stop()
```

6. Results and discussions

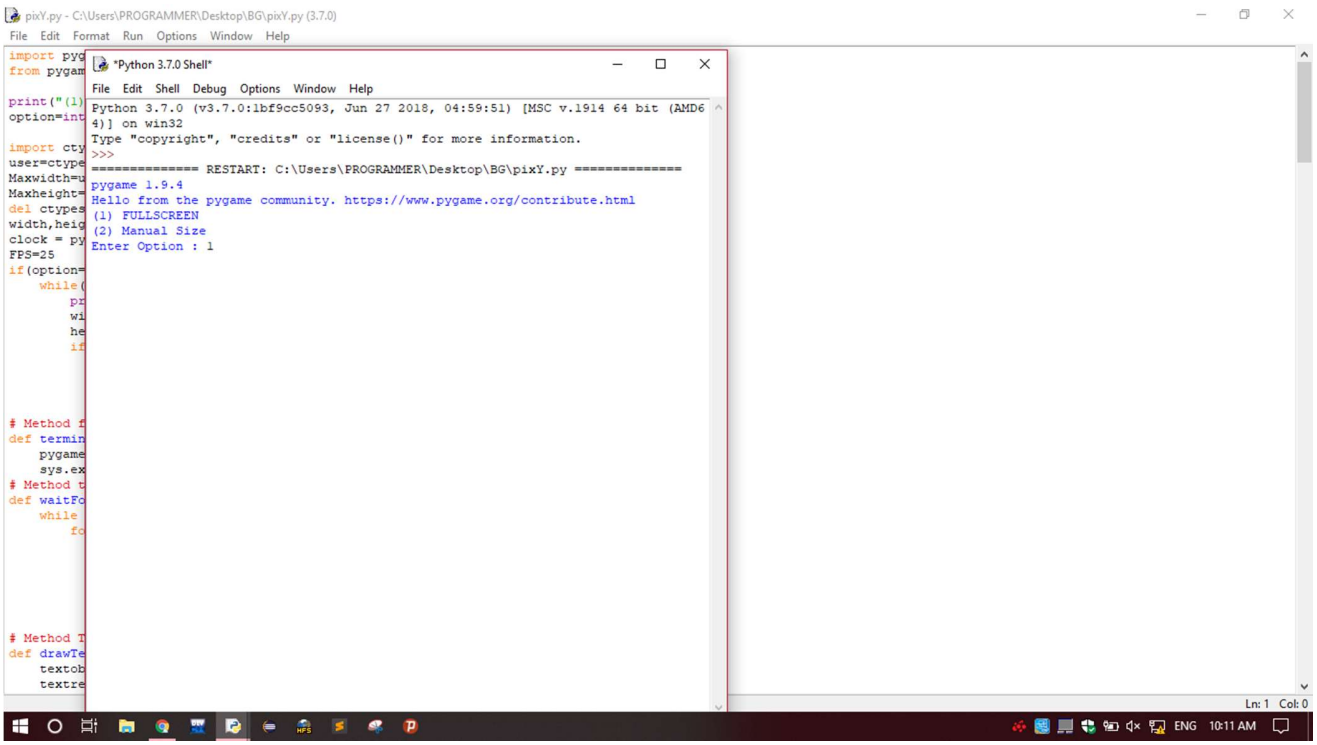
6.1) screen shots and description:



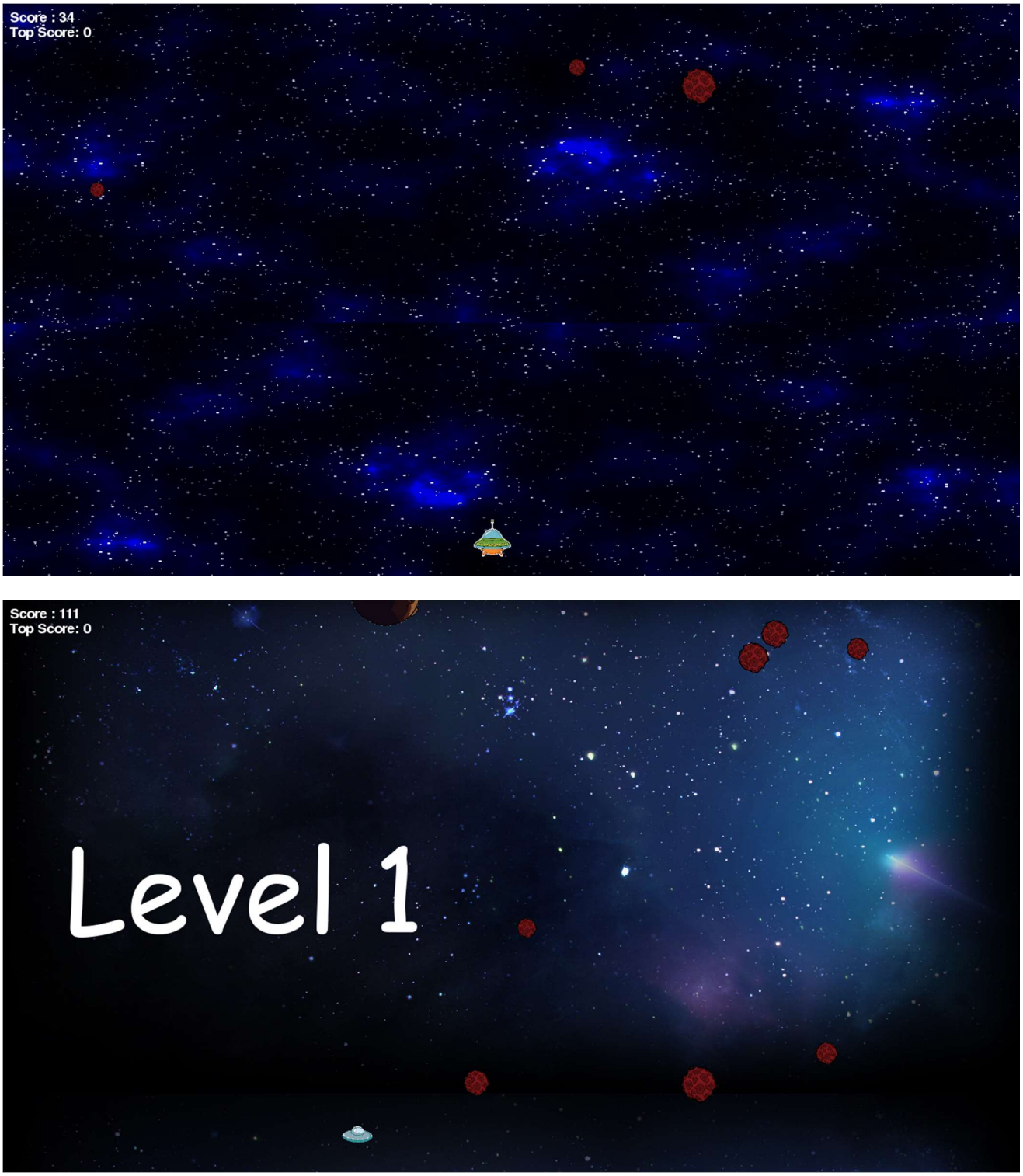
Screen shot 1:select window size



Screen shot 2:select option no 2



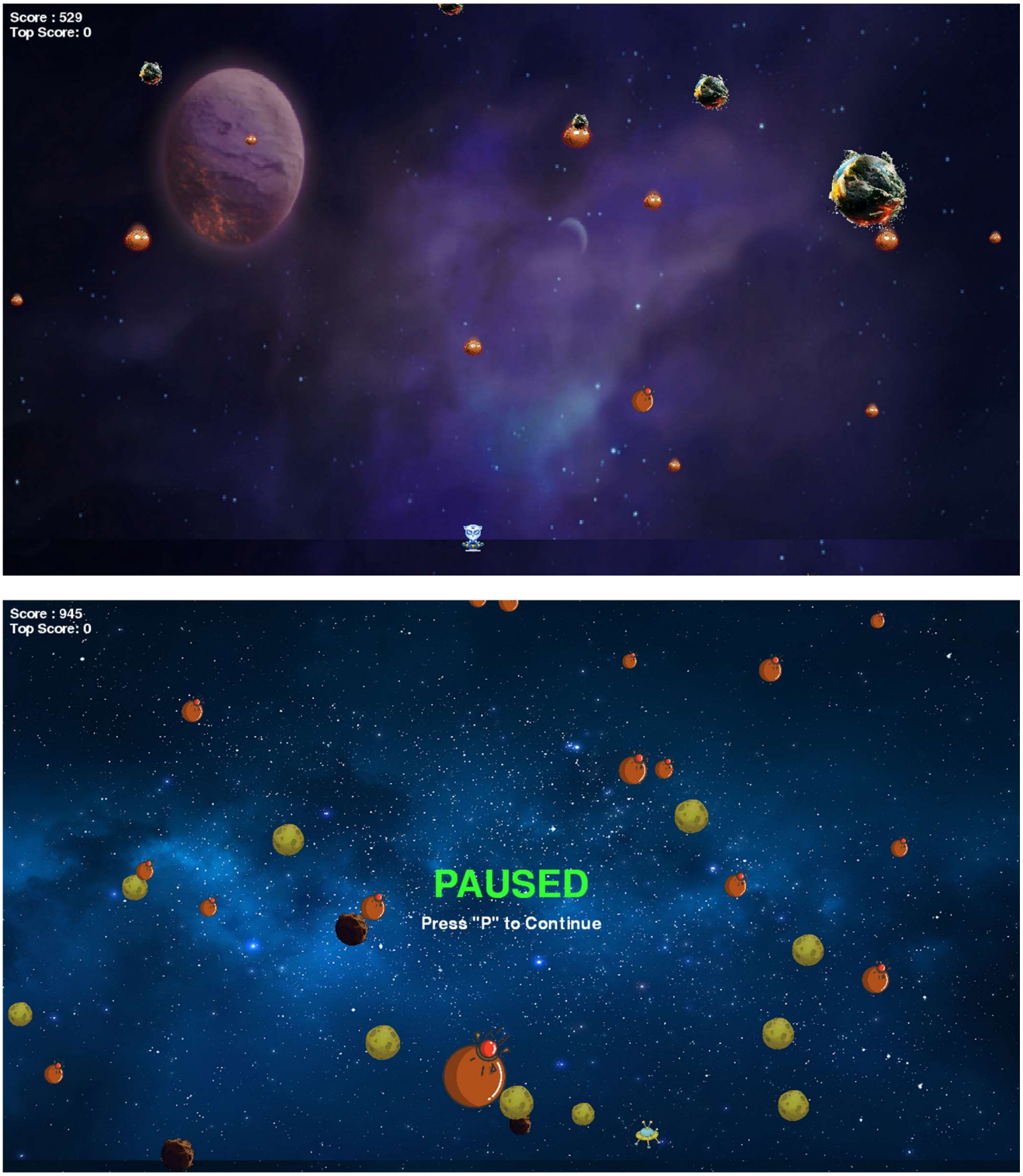
Screen shot 3:select option no 1



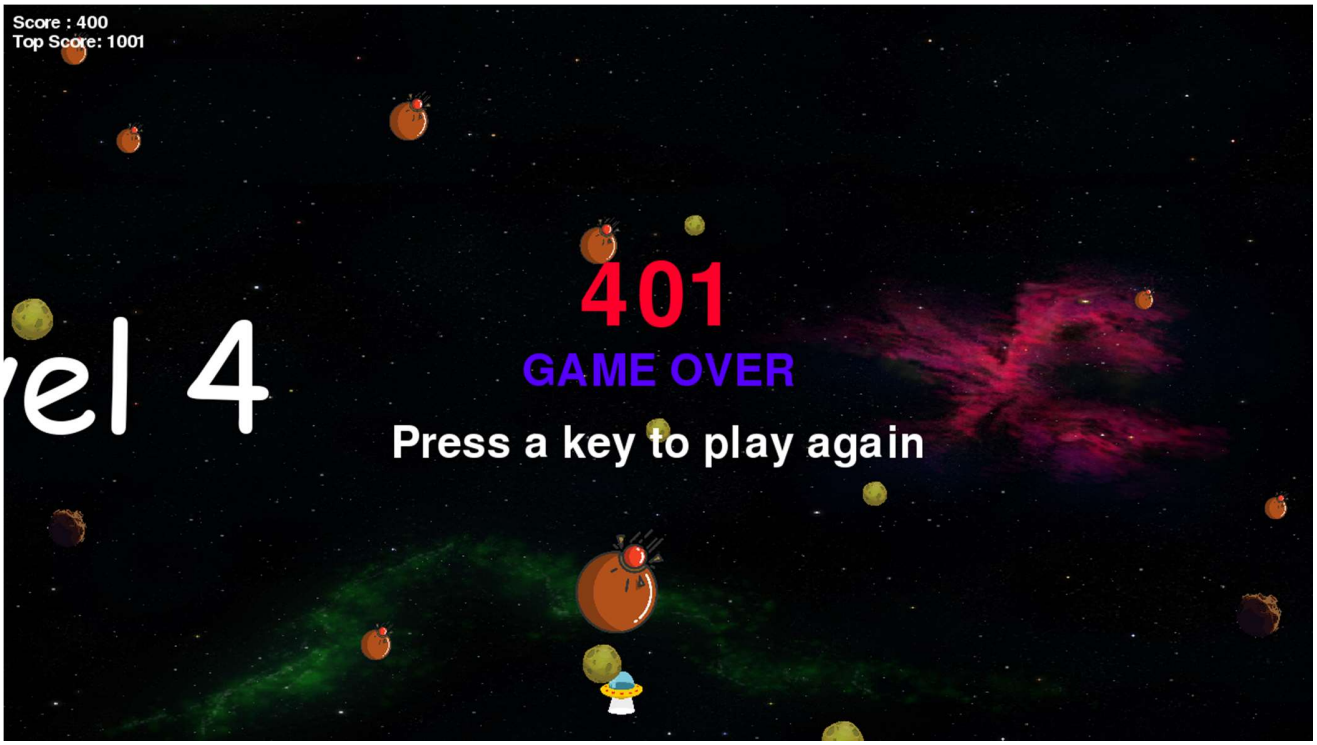
Screen shot 4:Level 1 of the game

Screen shot 5:level 2 and level 3 of the game

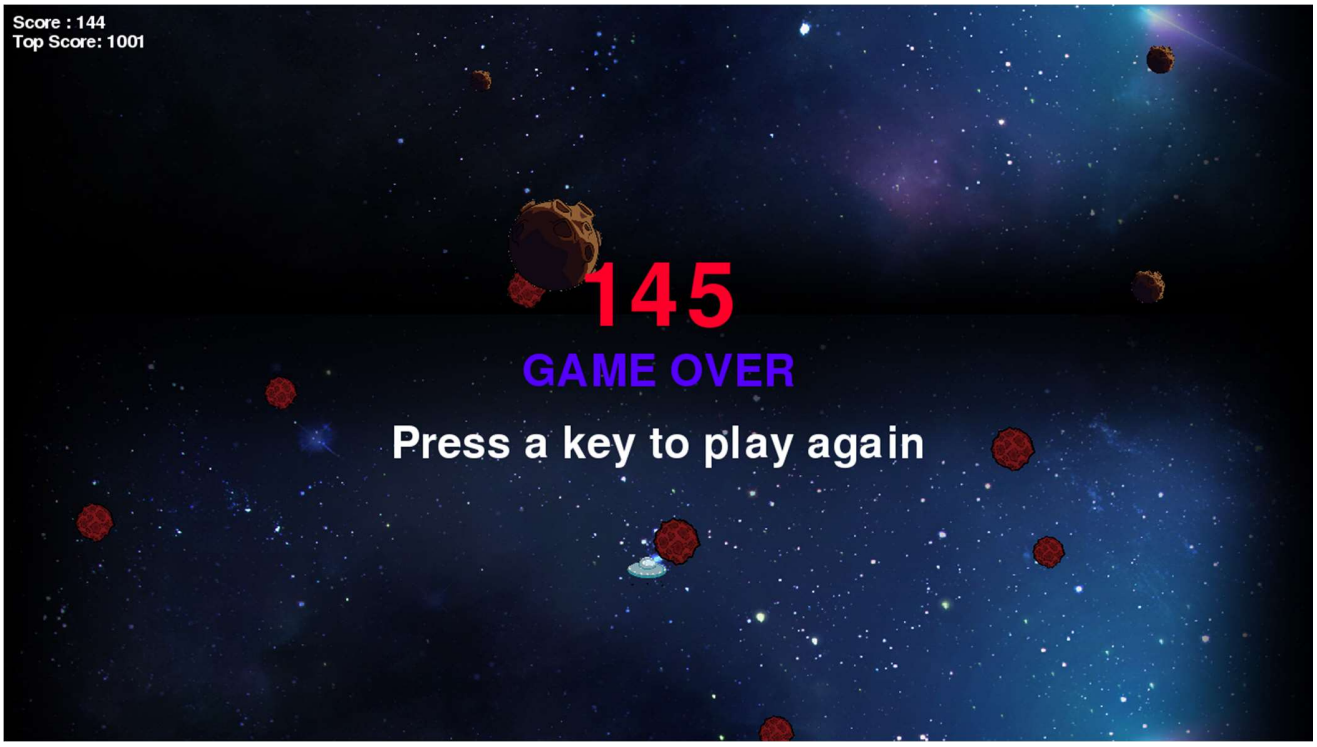
Screen shot 6: level 4 of the game



Screen shot 7:Pause button activation



Screen shot 8:game over



Screen shot 9: press a key to play again

7. CONCLUSION

A software project means a lot of experience. In this section we summarize the experience gained by project team during development of "**DODGER**"

7.1 The Obstacles

1. Working with the games completely a new experience for us.
2. We adopt these thing by video tutorals, internet and learning materials given by the tool themselves. It is matter of time, patience and hard work.
3. It is very sensible work and it demands much time because Games try to connect game environment with the real world.

7.2 The Achivements

1. Now we know much more aboutbasic pc games. How it works? The properties objects and others
2. Co-Operation between group members
3. Develop communication skills
4. Growing creatice thinking and imagination capability

8. Future Enhancement

- Improve graphical representation
- Introduce new game Features
- Introduce new environment and scenes

9. APPENDIX

Appendix A: References

8.1 General References

1. Pixel Collision - http://renesd.blogspot.com/2017/03/pixel-perfect-collision-detection-in.html
2. Sprites - https://www.pygame.org/docs/ref/sprite.html
3. Pixel Collision - https://www.youtube.com/watch?v=Idu8XfwKUao
4. Pause Game - https://pythonprogramming.net/pause-game-pygame/

8.2 Special Thanks To

- pygame - http://pygame.org
- YouTube - https://www.youtube.com/
- Stackoverflow – https://www.stackoverflow.com
- Pythonprogramming - https://pythonprogramming.net
- Slideshare - https://www.slideshare.net