**ACKNOWLEDGEMENTS**

The experience of creating my own game was indeed a memorable one and it would have been incomplete if not for the guidance and support I received from the people with me in this journey.

I would like to thank our revered Principal, Shri Ramakrishnaih, for giving me this opportunity to explore new avenues through this game.

I would like to express my immense gratitude to my computer science teacher, Shri Venkateshwar Prusty, for always being there to guide and motivate me.

I would like to thank my parents for supporting me in this endeavor and always believing in me.

Finally, I would like to thank my classmates, with whom this I share the joy of this experience, for their valuable inputs and inspiration.

**CONTENTS**

1. Aim
2. Introduction
3. Theory
4. Program Design
5. Program Code
6. Snapshots
7. Future Enhancements
8. Bibliography

**AIM**

To develop the game ESCAPE using python and pygame.

**INTRODUCTION**

The last few decades have witnessed a gargantuan increase in the common man’s interest in digital entertainment. One major contributor to this revolutionary change is gaming. Gaming has become an engrossing and vivid source of entertainment for millions across the globe. The advent of online gaming, mobile gaming and console gaming have provided an impetus to fuel a captivating desire for gaming among the masses. Since its commercial birth in the 1950s as a technological oddity at a science fair, gaming has blossomed into [one of the most profitable entertainment industries in the world.](http://www.fastcompany.com/3021008/why-video-games-succeed-where-the-movie-and-music-industries-fail)

The mobile technology boom in recent years has revolutionized the industry and opened the doors to a new generation of gamers. Indeed, gaming has become so integrated with modern popular culture that now even grandmas know what Angry Birds is, and more than[42 percent of the world’s population are gamers and four out of five U.S. households have a console.](http://www.theesa.com/wp-content/uploads/2015/04/ESA-Essential-Facts-2015.pdf)

The enormity of the scope and impact that gaming can have is indeed overwhelming. This potential of gaming has inspired me to endeavor to program a game.

**THEORY**

The game, ESCAPE, is a based on a scenario where the player is an agent of an anti- terror organization. Input has been received of a notorious gang’s whereabouts and it is up to the player to go and retrieve the loot stolen by the gang. For this the player must cross drones and monsters to reach the loot after which he must battle the terrorists.

The game consists of three levels. The first level is inside the lobby of the gangs hideout. Here the player must destroy drones deployed to protect the safe room. The next level is inside the safe room which is actually a mummy’s tomb. Here the player must battle monsters before procuring the loot. But once the player obtains the loot, the terrorists are alerted and the player must fight the thugs before escaping.

He can do all this with only 100 sgrenades and 100 bullets. There is also a time limit to the game after which the game ends. Thus the player has to complete all the three levels within the time limit.

**PROGRAM DESIGN**

An extensive use of classes was incorporated in the design of this game. For ease of understanding, the various classes and their data members have been explained here:

**Class player:**

This class, as the name suggests contains all the variables and functions required for the control of the player. The data members are:

def manmove(self):

The manmove function moves the player based on input from the keyboard.

def healthdec(self,dam)

This function decreases the player’s health by a measure given by the variable dam. Thus the player’s health can be decreased by different measures based on the game play.

def hbar(self)

This function displays the health of the player on the screen during the game play.

def bomb(self)

This function is used to drop grenades on enemies and also counts the number of grenades used.

**Class drone:**

This class contains data members that control the movement health and attack of the drones that appear in the first level. It consists of the following member functions:

def move(self, dx, dy)

This function makes the drones move in a random fashion by incorporating the randint() function.

def armdec(self,de)

This function decreases the drones armour efficiency when hit by a grenade.

def armbar(self,pos)

This function displays the drones armour efficiency during the first level of the game.

**Class monster:**

This class has data members that control the monsters that protect the loot in the second level.

def \_\_init\_\_(self,x,y,d,t,s)

Its constructor function takes in values for its initial position, type of monster and initial direction of movement.

def monmove(self)

This function moves the monster and blits the particular image specified by the user. The user can initialize any of the three monsters as mentioned in the constructor function.

def stadec(self,low)

this function lowers the stamina of the monster by a measure given by low.

def stabar(self,po)

This function diplays the stamina of the monster using a coloured bar.

**Class terrorist:**

This class has functions that monitor the movement and health of the terrorists in the third level of the game.

def \_\_init\_\_(self,a,b,d)

This constructor function initializes the initial coordinates of the object.

def th(self,lo)

This function decreases the health of the terrorist when hit by a grenade.

def thbar(self,p)

This function displays the health of the terrorist using a coloured bar.

def tmove(self,a,b)

This function moves the monster.

**FUNCTIONS**

def Text(surf,text,size,x,y,color,font\_name)

This function is used to display text and numbers like the number of grenades and ammunition, and the game time on to the screen.

def collide(x,y,a,b,z)

This function is used to check if to images have collided and returns True if it is so.

def search(a,b,x,y)

This function is used by the drones. It enables them to ‘search’ for the player by comparing the coordinates of the player and the drone.

def levelchange()

def disi()

This function is used to display the initial introductory images at the start of the game.

**PROGRAM CODE**

""" Computer Science Project- Sridhar Sola, Class 12

---------------------------------------BREAKOUT---------------------------------------------

>>>>>>>>>>>>>>>>>>>>>>>>>>><<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< """

import pygame, sys

from pygame.locals import\*

import random

pygame.init()

FPS=40

fpsClock=pygame.time.Clock()

canvas=pygame.display.set\_mode((1350,700))

pygame.display.set\_caption("BREAKOUT")

"""----------------------------------------------------------------------------------------------------"""

#variables

fight=0

LEVEL=0

check=0

ch=0

x=0

y=0

d=0

t=0

s=0

i=0

le1=0

le2=0

le3=0

"""------------------------------------------------------------------------------------------------------"""

#COLOURS

White = (255,255,255)

Black = ( 0, 0, 0)

Dark = ( 10, 10, 10)

Grey = (100,100,100)

Red = (255, 0, 0)

RED = (255, 0, 0,5)

Green = ( 0,255, 0)

"""------------------------------------------------------------------------------------------------------"""

#IMAGES

man1=pygame.image.load('final1.png')

man2=pygame.image.load('final2.png')

man3=pygame.image.load('final3.png')

man4=pygame.image.load('final4.png')

fight1=pygame.image.load('fight1.png')

fight2=pygame.image.load('fight2.png')

fight3=pygame.image.load('fight3.png')

fight4=pygame.image.load('fight4.png')

droneImg=pygame.image.load('drone1.png')

laser1=pygame.image.load('laser1.png')

laser2=pygame.image.load('laser2.png')

laser3=pygame.image.load('laser3.png')

laser4=pygame.image.load('laser4.png')

escape=pygame.image.load('start.png')

gren=pygame.image.load('grenade.png')

blast=pygame.image.load('blast.png')

l1=pygame.image.load('l1.png')

l2=pygame.image.load('l2.png')

l3=pygame.image.load('l3.png')

I1=pygame.image.load('intro1.png')

I2=pygame.image.load('l1in.png')

I3=pygame.image.load('I3.png')

Ti=pygame.image.load('tombi.png')

lose=pygame.image.load('LOSE.png')

suc=pygame.image.load('suc.png')

mon1\_1=pygame.image.load('monster1\_1.png')

mon1\_2=pygame.image.load('monster1\_2.png')

mon1\_3=pygame.image.load('monster1\_3.png')

mon1\_4=pygame.image.load('monster1\_4.png')

mon2\_1=pygame.image.load('monster2\_1.png')

mon2\_2=pygame.image.load('monster2\_2.png')

mon2\_3=pygame.image.load('monster2\_3.png')

mon2\_4=pygame.image.load('monster2\_4.png')

mon3\_1=pygame.image.load('monster3\_1.png')

mon3\_2=pygame.image.load('monster3\_2.png')

mon3\_3=pygame.image.load('monster3\_3.png')

mon3\_4=pygame.image.load('monster3\_4.png')

police1\_1=pygame.image.load('police1\_1.png')

police1\_2=pygame.image.load('police1\_2.png')

"""-----------------------------------------------------------------------------------------------------"""

#CLASSES

class player:

def \_\_init\_\_(self):

self.manx=965

self.many=630

self.last=0

self.grenx=10000

self.greny=10000

self.grenade=1

self.bx=1000

self.by=700

self.health=200

self.bno=100

self.gno=100

def manmove(self):

dui=pygame.key.get\_pressed()

if dui[pygame.K\_LEFT]:

if self.manx>=0:

self.manx-=10

self.last=3

if dui[pygame.K\_RIGHT]:

if self.manx<=1200:

self.manx+=10

self.last=2

if dui[pygame.K\_UP]:

if self.many!=0:

self.many-=10

self.last=0

if dui[pygame.K\_DOWN]:

if self.many!=630:

self.many+=10

self.last=1

if self.last==0:

if fight==0:

canvas.blit(man1,(self.manx,self.many))

if fight==1:

canvas.blit(fight1,(self.manx,self.many))

if self.last==1:

if fight==0:

canvas.blit(man4,(self.manx,self.many))

if fight==1:

canvas.blit(fight4,(self.manx,self.many))

if self.last==2:

if fight==0:

canvas.blit(man2,(self.manx,self.many))

if fight==1:

canvas.blit(fight2,(self.manx,self.many))

if self.last==3:

if fight==0:

canvas.blit(man3,(self.manx,self.many))

if fight==1:

canvas.blit(fight3,(self.manx,self.many))

def healthdec(self,dam):

if self.health>0:

self.health=self.health-dam

def hbar(self):

pygame.draw.rect(canvas,Green,(1200,270,self.health/2,10))

def bomb(self):

if self.bno>=1:

du=pygame.key.get\_pressed()

if du[pygame.K\_SPACE] and self.grenade==1:

self.grenx=self.manx

self.greny=self.many

self.grenade=0

self.bno-=1

if self.grenade==0 :

canvas.blit(gren,(self.grenx,self.greny))

if du[pygame.K\_RETURN] and self.grenade==0:

self.bx=self.grenx-10

self.by=self.greny-10

canvas.blit(blast,(self.bx,self.by))

self.grenade=1

class drone:

def \_\_init\_\_(self):

self.dx=0

self.dy=0

self.img=droneImg

self.w=30

self.m=0

self.bx=0

self.by=0

self.arm=200

def armdec(self,de):

if self.arm>0:

self.arm=self.arm-de

def armbar(self,pos):

pygame.draw.rect(canvas,White,(1220,pos,self.arm/2,10))

def move(self, dx, dy):

if self.arm>=5:

if self.w==30:

self.m=random.randint(0,4)

self.w=0

self.w=self.w+1

canvas.blit(self.img,(self.dx,self.dy))

if self.m==0 and self.dy>10:

self.dy-=10

if self.m==1 and self.dx>10:

self.dx-=10

if self.m==2 and self.dy<650:

self.dy+=10

if self.m==3 and self.dx<1000 :

self.dx+=10

class monster:

def \_\_init\_\_(self,x,y,d,t,s):

self.mx=x

self.my=y

self.dir=d

self.t=t

self.s=s

self.sta=200

def stadec(self,low):

if self.sta>0:

self.sta=self.sta-low

def stabar(self,po):

pygame.draw.rect(canvas,Red,(1175,po,self.sta/2,10))

def monmove(self):

if self.sta>=15:

if self.dir==1:

if self.t==1:

canvas.blit(mon1\_1,(self.mx,self.my))

if self.t==2:

canvas.blit(mon2\_1,(self.mx,self.my))

if self.t==3:

canvas.blit(mon3\_1,(self.mx,self.my))

self.mx-=self.s

if self.mx<=10:

self.dir=2

if self.dir==2:

if self.t==1:

canvas.blit(mon1\_2,(self.mx,self.my))

if self.t==2:

canvas.blit(mon2\_2,(self.mx,self.my))

if self.t==3:

canvas.blit(mon3\_2,(self.mx,self.my))

self.my+=self.s

if self.my>=600:

self.dir=3

if self.dir==3:

if self.t==1:

canvas.blit(mon1\_3,(self.mx,self.my))

if self.t==2:

canvas.blit(mon2\_3,(self.mx,self.my))

if self.t==3:

canvas.blit(mon3\_3,(self.mx,self.my))

self.mx+=self.s

if self.mx>=1000:

self.dir=4

if self.dir==4:

if self.t==1:

canvas.blit(mon1\_4,(self.mx,self.my))

if self.t==2:

canvas.blit(mon2\_4,(self.mx,self.my))

if self.t==3:

canvas.blit(mon3\_4,(self.mx,self.my))

self.my-=self.s

if self.my<=40:

self.dir=1

class ter:

def \_\_init\_\_(self,a,b,d):

self.tx=a

self.ty=b

self.d=d

self.h=200

self.m=1

def th(self,lo):

if self.h>=10:

self.h-=lo

def thbar(self,p):

pygame.draw.rect(canvas,Red,(1225,p,self.h/2,10))

def tmove(self,a,b):

if self.h>=10:

if self.d==1:

if a<self.tx and b<40:

canvas.blit(police1\_2,(self.tx,self.ty))

self.tx-=5

if a>=self.tx and self.tx!=1000 and b<40:

canvas.blit(police1\_1,(self.tx,self.ty))

self.tx+=5

else:

if self.m==1:

canvas.blit(police1\_2,(self.tx,self.ty))

self.tx-=5

if self.tx==10:

self.m=2

if self.m==2:

canvas.blit(police1\_1,(self.tx,self.ty))

self.tx+=5

if self.tx==1000:

self.m=1

if self.d==2:

if b<self.ty and a<40:

canvas.blit(police1\_2,(self.tx,self.ty))

self.ty-=5

if b>=self.ty and self.tx!=1000 and a<40:

canvas.blit(police1\_1,(self.tx,self.ty))

self.ty+=5

else:

if self.m==1:

canvas.blit(police1\_2,(self.tx,self.ty))

self.ty-=5

if self.ty==10:

self.m=2

if self.m==2:

canvas.blit(police1\_2,(self.tx,self.ty))

self.ty+=5

if self.ty==690:

self.m=1

class bullets(pygame.sprite.Sprite):

def \_\_init\_\_(self):

pygame.sprite.Sprite.\_\_init\_\_(self)

self.k = 0

def update(self):

print('passed test for sprite...')

class bullet(pygame.sprite.Sprite):

def \_\_init\_\_(self):

pygame.sprite.Sprite.\_\_init\_\_(self)

self.k = 0

def update(self):

print(' test for sprite...')

ALL = pygame.sprite.Group()

Bull= bullets()

a = bullet()

ALL.add(a)

ALL.add(Bull)

#FUNCTIONS

def search(a,b,x,y):

if a<x and b<y:

return 1

if a>x and b<y:

return 2

if a<x and b>y:

return 3

if a>x and b>y:

return 4

def collide(x,y,a,b,z):

if x<=a+z and x>=a-z:

if y<=b+z and y>=b-z:

return True

def shoot(ch,x,y):

if ch==1:

canvas.blit(laser4,(x,y))

y+=3

if ch==2:

canvas.blit(laser1,(x,y))

y-=3

if ch==3:

canvas.blit(laser4,(x,y))

x+=3

if ch==4:

canvas.blit(laser3,(x,y))

x-=3

def levelchange():

global LEVEL

ke=pygame.key.get\_pressed()

if ke[pygame.K\_a]:

LEVEL=1

if ke[pygame.K\_b]:

LEVEL=2

if ke[pygame.K\_c]:

LEVEL=3

def disi():

if i==1:

canvas.blit(escape,(0,0))

if i==0:

canvas.blit(I1,(0,0))

def Text(surf,text,size,x,y,color,font\_name):

font=pygame.font.Font(font\_name,size)

text\_surface=font.render(text,True,color)

text\_rect=text\_surface.get\_rect()

text\_rect.midtop=(x,y)

surf.blit(text\_surface,text\_rect)

return(text\_rect)

"""----------------------------------------------------------------------------------------------------"""

#OBJECTS

p=player()

D1=drone()

D2=drone()

m1=monster(100,210,1,1,5)

m2=monster(600,550,3,2,10)

m3=monster(200,200,2,3,15)

t1=ter(20,20,1)

t3=ter(1000,200,1)

t2=ter(1000,100,2)

#text=pygame.font.Font

"""------------------------------------------------------------------------------------------------------"""

#MAIN GAME LOOP

while True:

canvas.fill(Black)

for event in pygame.event.get():

if event.type==QUIT:

exitgame()

if LEVEL==0:

disi()

d=pygame.key.get\_pressed()

if d[pygame.K\_RETURN] and i==0:

i=1

if d[pygame.K\_RETURN] and i==1:

i=2

if d[pygame.K\_RETURN] and i==2:

i=3

if LEVEL==1:

if le1==0:

canvas.blit(I2,(0,0))

pygame.display.update()

pygame.time.delay(5000)

le1=1

canvas.blit(l1,(0,0))

D1.move(40,40)

D2.move(100,100)

D1.armbar(490)

D2.armbar(530)

if collide(p.manx,p.many,D1.dx,D1.dy,20)==True:

p.healthdec(5)

if collide(p.manx,p.many,D2.dx,D2.dy,20)==True:

p.healthdec(5)

if collide(p.grenx,p.greny,D1.dx,D1.dy,50)==True:

canvas.blit(blast,(D1.dx,D1.dy))

D1.armdec(10)

p.grenx=10000

p.grenade=1

if collide(p.grenx,p.greny,D2.dx,D2.dy,50)==True:

canvas.blit(blast,(D2.dx,D2.dy))

D2.armdec(10)

p.grenx=10000

p.grenade=1

if D1.arm<20 and D2.arm<20:

LEVEL=2

check=search(D1.bx,D1.by,p.manx,p.many)

bomb=str(p.bno)

gun=str(p.gno)

Text(canvas,bomb,30,1250,350,White,"mine.ttf")

Text(canvas,gun,30,1250,300,White,"mine.ttf")

if LEVEL==2:

if le2==0:

canvas.blit(Ti,(0,0))

pygame.display.update()

pygame.time.delay(5000)

le2=1

canvas.blit(l2,(0,0))

m1.monmove()

m2.monmove()

m3.monmove()

m1.stabar(410)

m2.stabar(460)

m3.stabar(515)

if collide(p.manx,p.many,m1.mx,m1.my,20)==True:

p.healthdec(10)

if collide(p.manx,p.many,m2.mx,m2.my,20)==True:

p.healthdec(10)

if collide(p.manx,p.many,m3.mx,m3.my,20)==True:

p.healthdec(10)

if collide(p.grenx,p.greny,m1.mx,m1.my,40)==True:

m1.stadec(15)

canvas.blit(blast,(m1.mx,m1.my))

p.grenx=10000

p.grenade=1

if collide(p.grenx,p.greny,m2.mx,m2.my,40)==True:

m2.stadec(15)

canvas.blit(blast,(m2.mx,m2.my))

p.grenx=10000

p.grenade=1

if collide(p.grenx,p.greny,m3.mx,m3.my,40)==True:

m3.stadec(15)

canvas.blit(blast,(m3.mx,m3.my))

p.grenx=10000

p.grenade=1

bomb=str(p.bno)

gun=str(p.gno)

Text(canvas,bomb,30,1250,310,White,"mine.ttf")

Text(canvas,gun,30,1250,285,White,"mine.ttf")

if m1.sta<15 and m2.sta<15 and m3.sta<15:

LEVEL=3

if LEVEL==3:

if le3==9:

canvas.blit(I3,(0,0))

pygame.display.update()

pygame.time.delay(5000)

le3=1

canvas.blit(l3,(0,0))

bomb=str(p.bno)

gun=str(p.gno)

Text(canvas,bomb,30,1240,310,White,"mine.ttf")

Text(canvas,gun,30,1240,280,White,"mine.ttf")

t1.tmove(p.manx,p.many)

t2.tmove(p.manx,p.many)

t3.tmove(p.manx,p.many)

t1.thbar(400)

t2.thbar(440)

t3.thbar(480)

if collide(p.manx,p.many,t1.tx,t1.ty,20)==True:

p.healthdec(10)

if collide(p.manx,p.many,t2.tx,t2.ty,20)==True:

p.healthdec(10)

if collide(p.manx,p.many,t3.tx,t3.ty,20)==True:

p.healthdec(10)

if collide(p.grenx,p.greny,t1.tx,t1.ty,40)==True:

t1.th(15)

canvas.blit(blast,(t1.tx,t1.ty))

p.grenx=10000

p.grenade=1

if collide(p.grenx,p.greny,t2.tx,t2.ty,40)==True:

t2.th(15)

canvas.blit(blast,(t2.tx,t2.ty))

p.grenx=10000

p.grenade=1

if collide(p.grenx,p.greny,t3.tx,t3.ty,40)==True:

t3.th(15)

canvas.blit(blast,(t3.tx,t3.ty))

p.grenx=10000

p.grenade=1

if t1.h<10 and t2.h<10 and t3.h<10 and t<300:

LEVEL=4

print(p.manx)

print(p.many)

if LEVEL!=0:

p.manmove()

p.hbar()

p.bomb()

bomb=str(p.bno)

gun=str(p.gno)

time=pygame.time.get\_ticks()

t=int(time/1000)

timi=str(t)

Text(canvas,timi,30,1225,50,White,"mine.ttf")

levelchange()

time=pygame.time.get\_ticks()

t=int(time/1000)

if LEVEL==4:

canvas.blit(suc,(0,0))

if t>=300:

LEVEL=10

canvas.blit(lose,(0,0))

pygame.display.update()

fpsClock.tick(F

**FUTURE ENHANCEMENTS**

* Different modes of the game, such as, easy, medium and hard can be made where the time limit for the game varies.
* The use of the mouse can be incorporated to reduce the use of the keyboard to make the game easier to play.
* More attractive backgrounds and icons can be designed to be used in the game.
* Sound effects can also be used to increase the game’s overall appeal.

**BIBLIOGRAPHY**

* Making Games with python and pygame by Alwert Sweigart.
* Python and pygame documentations
* [www.pygame.org](http://www.pygame.org)
* www.github.com