1. Acknowledgement

2. Index

|  |  |  |
| --- | --- | --- |
| **Serial no.** | **Title** | **Page no.** |
| 1. | Introduction |  |
| 2. | System requirements |  |
| 3. | Basic Algorithm |  |
| 3. | Modules |  |
| 4. | User Defined Functions |  |
| 5. | Variables |  |
| 7. | Source Code |  |
| 8. | Screen shots |  |
| 9. | Bibliography |  |

System Requirements

* System requirements:

1. 1 gigahertz (**GHz**) or faster 32-bit (**x86**) or 64-bit (**x64**) processor
2. Mouse
3. Keyboard
4. Colour Monitor
5. Speakers

* Software requirements:

1. Windows 7 or higher
2. Python 2.7.4 or higher

**BASIC ALGORITHM**

1. First off the screen and variables coming under the general framework are initialized.
2. Subsequently the first game loop is entered wherein the menu and instructions and rendered as per the users wish.
3. Once the user has made his selection the level is loaded and the program proceeds into the second game loop where the core physics engine the game is run.
4. A for loop for keyboard and mouse events is setup.
5. The bird as well as the sling are engaged and ready to be used.
6. After every iteration the draw method blits the required images on the screen
7. When the bird is launched the sling is disengaged and gravity begins acting on the bird
8. A few iterations after launch the program begins listening for collisions between the bird and other objects placed on the screen.
9. When a collision occurs the program identifies the two entities involved and carries out the required task be it deletion or updating trajectory and forces on the objects.
10. When the velocity of the bird falls below a minimum threshold in both dimensions it is deleted.
11. The sling is now reengaged with a new bird and the same process from step 6 through 10 are repeated.
12. Endgame is reached when either the number of remaining birds equals zero or when the number of pigs remaining equals zero.
13. At endgame the program breaks out the core loop.
14. Depending on which condition is met the win/loss menu is run and options to replay directly, return to menu and quit are displayed.
15. Depending on the users choice the program can rerun the desired location or terminate.

**Modules and Library Functions**

|  |  |  |
| --- | --- | --- |
| **Module** | **Library Functions** | **Usage** |
| pygame | .font | To generate fonts used |
| .render | To render the font |
| .draw | To draw a given shape |
| .blit | Adds object to pygame screen |
| .display.update | Refreshes pygame screen |
| .time.clock | Initializes the clock |
| .display.set\_mode | Initializes display |
| .event.get | Returns list of events |
| .mixer | Controls music |
| .image | Controls images |
| .transform | Modify images |
| .quit | Quits pygame |
| sys | .exit | Exits from the terminal window |
| thread | .start\_new\_thread | Starts new thread |
| slicer | .wooden\_block | Supplies blocks for the game |
| math | fabs | Returns absolute value |
| Sin | Returns sine of an angle |
| cos | Returns cosine of an angle |

**User Defined Functions**

|  |  |
| --- | --- |
| **Function name** | **Task** |
| menu | Renders and operates menu |
| instructions | Manages instructions page |
| lvl1 | Initializes level framework |
| victory | Handles win scenarios |
| lose | Handles loss scenarios |
| launchvel | Determines bird launch velocity |
| strap | Renders strap of sling |
| oob | Checks out of bound condition |
| draw | Renders dynamic graphics during gameplay |
| grav\_w\_detect | Controls gravity for blocks |
| grav\_p\_detect | Controls gravity for pigs |
| pigcol | Checks for collision between pig and bird |
| newcol\_w | Checks for collision between other movable ojects and bird |
| newcol\_r | Checks for collision between immovable objects and bird |
| poofanim | Renders poof animation |
| reddead | Controls removal of defunct birds |
| redreset | Resets birds after every play |
| substitutes | Controls birds not in play in current frame |
| slingrend | Renders sling |
| terminate | Removes defunct pigs |
| destroyer | Removes other defunct objects |

**Variables**

|  |  |
| --- | --- |
| **Data Type** | **Variable name** |
| String | - |
| Dictionary | - |
| List | Pigvel,piglist,exterminator,slingpos,slingcentre, redcentre, redpos,hard\_r,hard\_w,w\_guide,trash,colvel,exterminator |
| Integer/Floating Point | redsub,play,bird\_vel,slingengaged,slingpast,score,e,f,fricgage,skip,collidevar,setup,win,loss,cmd,button,x,y,redvelx,redvely,ct,ct1,ct2,appendct,grav |
| Class | blocks |
| Objects | display,col,tiny,minion,sling,red,endgame,fonts,f(file object) |
| Boolean | game\_exit,app\_exit,isreddead |
| Tuple | black, cyan,grey, green, blue, white |

**SOURCE CODE**

import pygame,math,slicer,thread

#pymunk

pygame.init()

app\_exit=False

game\_exit=False

black=(0,0,0)

white=(255,255,255)

grey=(150,150,150)

green=(0,255,100)

blue=(0,0,255)

cyan=(0,255,255)

display=pygame.display.set\_mode((1366,768),pygame.FULLSCREEN)

pygame.display.set\_caption("ANGRY BIRDS")

display.fill(white)

pygame.display.update()

img=slicer.blocks().wooden\_block()

tiny=pygame.image.load("tiny.png")

tiny=pygame.transform.scale(tiny,(80,8))

col=pygame.image.load("col.png")

col=pygame.transform.rotate(col,90)

col=pygame.transform.scale(col,(12,120))

minion=pygame.image.load("pigprime.gif")

minion=pygame.transform.scale(minion,(40,40))

pigvel=[]

piglist=[]

exterminator=[]

clock=pygame.time.Clock()

slingcentre=[] #150

sling=pygame.image.load("sling.PNG")

sling=pygame.transform.scale(sling,(50,50))

slingpos=[]

red=pygame.image.load("red2.GIF")

red=pygame.transform.scale(red,(30,30))

redcentre=[15,15]

redpos=[210,635]

redvelx,redvely=0,0

redsub=3

play=1

bird\_vel=0

slingengaged=2

slingpast=2

score=0

grav=.065

gravengaged=0

e=0.5

f=.95

fricgage=0

hard\_r=[]

hard\_w=[]

w\_guide=[]

trash=[]

colvel=[]

skip=0

collidevar=0

poofp=[pygame.image.load("poof1.png"),pygame.image.load("poof2.png"),pygame.image.load("poof3.png"),pygame.image.load("poof4.png")]

poofp=[pygame.transform.scale(poofp[0],(40,40)),pygame.transform.scale(poofp[1],(40,40)),pygame.transform.scale(poofp[2],(40,40)),pygame.transform.scale(poofp[3],(40,40))]

poofb=[pygame.image.load("poof1.png"),pygame.image.load("poof2.png"),pygame.image.load("poof3.png"),pygame.image.load("poof4.png")]

poofb=[pygame.transform.scale(poofp[0],(30,30)),pygame.transform.scale(poofp[1],(30,30)),pygame.transform.scale(poofp[2],(30,30)),pygame.transform.scale(poofp[3],(30,30))]

#pygame.mixer.music.load("bensound-buddy.mp3")

#pygame.mixer.music.play(-1)

#openning=pygame.image.load("openning.png")

#openning=pygame.transform.scale(openning,(1366,768))

endgame=pygame.image.load("background.png")

endgame=pygame.transform.scale(endgame,(1366,768))

win,loss=0,0

fonts=pygame.font.SysFont("Comic Sans",80)

fonts1=pygame.font.SysFont("Comic Sans",33)

setup=1

def menu():

global grey

cond=True

while cond:

imag=pygame.image.load("Menubackground.PNG")

imag=pygame.transform.scale(imag,(1366,768))

display.fill(white)

display.blit(imag,(0,0))

end=pygame.image.load("quit.PNG")

end=end.subsurface(140,160,320,130)

end=pygame.transform.scale(end,(200,60))

display.blit(end,(1146,20))

fonts=pygame.font.SysFont("Comic Sans",50)

label=fonts.render("MENU",True,cyan)

label2=fonts.render("LEVEL 1",True,cyan)

label3=fonts.render("LEVEL 2",True,cyan)

label4=fonts.render("INSTRUCTIONS",True,cyan)

pygame.draw.rect(display,(255,0,0),(480,90,400,80))

pygame.draw.rect(display,(255,0,0),(480,260,400,80))

pygame.draw.rect(display,grey,(480,430,400,80))

pygame.draw.rect(display,(255,0,0),(480,600,400,80))

display.blit(label,[680-label.get\_width()/ 2,110])

display.blit(label2,[680-label2.get\_width()/ 2,280])

display.blit(label3,[680-label3.get\_width()/ 2,450])

display.blit(label4,[680-label4.get\_width()/ 2,620])

pygame.display.update()

button=0

for event in pygame.event.get():

if event.type==pygame.MOUSEBUTTONDOWN:

button=1

if event.type==pygame.MOUSEBUTTONUP:

pos=event.pos

if button==1:

if pos[0] in range(465,896) and pos[1] in range(240,351):

return 1

elif pos[0] in range(465,896) and pos[1] in range(410,531):

pass

#return 2

elif pos[0] in range(465,896) and pos[1] in range(580,701):

instructions()

elif pos[0] in range(1126,1376) and pos[1] in range(0,101):

return 0

button=0

def instructions():

global cyan

imag=pygame.image.load("Menubackground.PNG")

imag=pygame.transform.scale(imag,(1366,768))

display.fill(white)

display.blit(imag,(0,0))

font=pygame.font.SysFont("Copperplate Gothic Bold",50)

font2=pygame.font.SysFont("Copperplate Gothic Bold",80)

text("Welcome to Angry Birds",50,font2)

text("Use the birds to kill all the pigs",120,font)

text("Launch the bird:",170,font)

text("Click and hold the bird loaded in the sling ",220,font)

text("Use the MOUSE to aim",270,font)

text("Release the mousebutton to launch the bird",320,font)

text("Happy Slinging!!!",370,font)

back=pygame.image.load("back.PNG")

back=pygame.transform.scale(back,(100,50))

display.blit(back,(0,0))

pygame.display.update()

while True:

for event in pygame.event.get():

if event.type==pygame.MOUSEBUTTONDOWN:

pos=event.pos

if pos[0] in range(105) and pos[1] in range(55):

return

def text(msg,pos,fonts):

lbl=fonts.render(msg,True,(255,0,0))

display.blit(lbl,[683-lbl.get\_width()/ 2,pos])

def lvl1():

global hard\_r,hard\_w,w\_guide,colvel,pigvel,piglist,slingpos,slingcentre

r1=pygame.draw.rect(display,black,(700,600,700,200))

r2=pygame.draw.rect(display,black,(1200,350,200,250))

r3=pygame.draw.rect(display,black,(180,700,110,50))

r4=pygame.draw.rect(display,black,(0,750,290,50))

r5=pygame.draw.rect(display,black,(180,700,520,70))

i1=display.blit(col,(750,480))

i2=display.blit(col,(900,480))

col2=pygame.transform.scale(col,(12,170))

i3=display.blit(col2,(1050,430))

i4=display.blit(tiny,(716,472))

i5=display.blit(tiny,(866,472))

i6=display.blit(tiny,(1016,422))

colvel=[[0,0],[0,0],[0,0],[0,0],[0,0],[0,0]]

m1=display.blit(minion,(736,432))

m2=display.blit(minion,(886,432))

m3=display.blit(minion,(1036,382))

m4=display.blit(minion,(1220,310))

pigvel=[[0,0],[0,0],[0,0],[0,0]]

hard\_r=[r1,r2,r3,r4,r5]

hard\_w=[[i1,i2],[i3],[i4,i5,i6]]

w\_guide=[col,col2,tiny]

piglist=[m1,m2,m3,m4]

slingpos=[200,650]

slingcentre=[225,650]

def lvl2():

global hard\_r,hard\_w,w\_guide,colvel,pigvel,piglist,slingpos,slingcentre

r1=pygame.draw.rect(display,black,(0,370,500,400))

r2=pygame.draw.rect(display,black,(600,640,800,130))

r3=pygame.draw.rect(display,black,(1340,0,30,645))

r4=pygame.draw.rect(display,black,(800,200,350,70))#Done till here

i1=display.blit(col,(750,480))

i2=display.blit(col,(900,480))

col2=pygame.transform.scale(col,(12,170))

i3=display.blit(col2,(1050,430))

i4=display.blit(tiny,(716,472))

i5=display.blit(tiny,(866,472))

i6=display.blit(tiny,(1016,422))

colvel=[[0,0],[0,0],[0,0],[0,0],[0,0],[0,0]]

m1=display.blit(minion,(736,432))

m2=display.blit(minion,(886,432))

m3=display.blit(minion,(1036,382))

m4=display.blit(minion,(1220,310))

pigvel=[[0,0],[0,0],[0,0],[0,0]]

hard\_r=[r1,r2,r3,r4,r5]

hard\_w=[[i1,i2],[i3],[i4,i5,i6]]

w\_guide=[col,col2,tiny]

piglist=[m1,m2,m3,m4]

slingpos=[200,650]

slingcentre=[225,650]

def poofanim(x,y,c):

global poofb,poofp

if c=='p':

pygame.mixer.music.load("poof.mp3")

pygame.mixer.music.play()

for j in poofp:

display.blit(j,[x,y])

pygame.time.wait(130)

pygame.draw.rect(display,white,(x,y,40,40))

pygame.display.update()

elif c=='b':

pygame.mixer.music.load("birddead.mp3")

pygame.mixer.music.play()

for j in poofb:

display.blit(j,[x,y])

pygame.time.wait(130)

pygame.draw.rect(display,white,(x,y,30,30))

pygame.display.update()

def victory():

global endgame,green,app\_exit,fonts,grey,fonts1,setup

f=open("Highscore","a+")

f.seek(0)

if score>int(f.read()):

f.close()

f=open("Highscore","w+")

f.write(str(score))

f.close()

f=open("Highscore","a+")

f.seek(0)

label=fonts.render("LEVEL CLEARED",True,grey)

label1=fonts1.render("SCORE: "+str(score),True,grey)

label2=fonts1.render("HIGHSCORE: "+f.read(),True,grey)

label3=fonts1.render("REPLAY",True,grey)

label4=fonts1.render("MENU",True,grey)

label5=fonts1.render("QUIT",True,grey)

while True:

for event in pygame.event.get():#loop to handle events

if event.type==pygame.MOUSEBUTTONDOWN:

p=pygame.mouse.get\_pos()

if p[0]>40 and p[0]<280 and p[1]>540 and p[1]<660:

display.fill(white)

pygame.display.update()

return

elif p[0]>560 and p[0]<800 and p[1]>540 and p[1]<660:

setup=1

return

elif p[0]>1080 and p[0]<1320 and p[1]>540 and p[1]<660:

app\_exit=True

return

display.blit(endgame,(0,0))

pygame.draw.rect(display,green,(400,300,550,100))

pygame.draw.rect(display,green,(60,550,200,60))

pygame.draw.rect(display,green,(320,550,200,60))

pygame.draw.rect(display,green,(580,550,200,60))

pygame.draw.rect(display,green,(840,550,200,60))

pygame.draw.rect(display,green,(1100,550,200,60))

display.blit(label,[675-label.get\_width()/ 2,325])

display.blit(label3,[160-label3.get\_width()/2,565])

display.blit(label1,[420-label1.get\_width()/2,565])

display.blit(label4,[680-label4.get\_width()/ 2,565])

display.blit(label2,[940-label2.get\_width()/ 2,565])

display.blit(label5,[1200-label5.get\_width()/ 2,565])

pygame.display.update()

pygame.time.wait(25)

def lose():

global endgame,green,app\_exit,fonts,grey,setup

label=fonts.render("LEVEL FAILED",True,grey)

label2=fonts.render("REPLAY",True,grey)

label3=fonts.render("MENU",True,grey)

label4=fonts.render("QUIT",True,grey)

while True:

for event in pygame.event.get():#loop to handle events

if event.type==pygame.MOUSEBUTTONDOWN:

p=pygame.mouse.get\_pos()

if p[0]>115 and p[0]<435 and p[1]>540 and p[1]<660:

display.fill(white)

pygame.display.update()

return

elif p[0]>540 and p[0]<860 and p[1]>540 and p[1]<660:

setup=1

return

elif p[0]>965 and p[0]<1285 and p[1]>540 and p[1]<660:

app\_exit=True

return

display.blit(endgame,(0,0))

pygame.draw.rect(display,green,(400,300,550,100))

pygame.draw.rect(display,green,(125,550,300,100))

pygame.draw.rect(display,green,(550,550,300,100))

pygame.draw.rect(display,green,(975,550,300,100))

display.blit(label,[675-label.get\_width()/ 2,325])

display.blit(label2,[275-label2.get\_width()/ 2,575])

display.blit(label3,[700-label3.get\_width()/ 2,575])

display.blit(label4,[1125-label4.get\_width()/ 2,575])

pygame.display.update()

pygame.time.wait(25)

def entry():

global openning

c=0

while c<1000:

display.blit(openning,(0,0))

c+=1

def slingrend(x,y):

display.blit(sling,(x,y))

def redclear(pos):

pygame.draw.rect(display,white,(pos[0],pos[1],30,30))

def redrend(x,y):

display.blit(red,(x,y))

def strapclear(pos):

global slingpos

pygame.draw.line(display,white,(slingpos[0],slingpos[1]),(pos[0]+redcentre[0],pos[1]+redcentre[1]),3)

pygame.draw.line(display,white,(slingpos[0]+50,slingpos[1]),(pos[0]+redcentre[0],pos[1]+redcentre[1]),3)

def strap(birdx,birdy):

global slingpos

pygame.draw.line(display,black,(slingpos[0],slingpos[1]),(birdx+redcentre[0],birdy+redcentre[1]),3)

pygame.draw.line(display,black,(slingpos[0]+50,slingpos[1]),(birdx+redcentre[0],birdy+redcentre[1]),3)

def launchvel(pos):

velx=(slingcentre[0]-pos[0]-redcentre[0])/7

vely=(slingcentre[1]-pos[1]-redcentre[1])/7

return velx,vely

def oob():

global redvelx,redvely,redpos

if redpos[0]>1370 or redpos[0]<-30 or redpos[1]>770 :

redvelx,redvely=0,0

def substitutes():

global redsub,red

x,y=140,720

if redsub>-1:

for i in range(3):

pygame.draw.rect(display,white,(x,y,30,30))

if i<redsub:

display.blit(red,(x,y))

x-=40

def reddead():

global redsub,slingengaged,deadcount,redvelx,redvely,redpos,gravengaged,collidevar,isreddead

if slingengaged==0 and math.fabs(redvelx)<0.05 and math.fabs(redvely)<=0.065:

thread.start\_new\_thread(poofanim,(redpos[0],redpos[1],'b'))

redsub-=1

isreddead=True

if redsub!=-1:

slingengaged=2

gravengaged=0

redvelx,redvely=0,0

collidevar=0

redclear(redpos)

redreset()

def redreset():

global redpos,slingpos

redpos=[slingpos[0]+10,slingpos[1]-15]

def draw():

global hard\_r,hard\_p,hard\_w,w\_guide,trash,piglist,exterminator,poofp,score

for i in hard\_r:

pygame.draw.rect(display,black,i)

for i in range(len(hard\_w)):

for j in hard\_w[i]:

display.blit(w\_guide[i],j)

for i in trash:

i=list(i)

pygame.draw.rect(display,white,i)

for i in piglist:

display.blit(minion,i)

for i in exterminator:

thread.start\_new\_thread(poofanim,(piglist[i][0],piglist[i][1],"p"))

score+=50

trash=[]

def grav\_w\_detect():

global hard\_w,white,grav,w\_guide

for i in range(len(hard\_w)):

for k in range(len(hard\_w[i])):

for j in range(hard\_w[i][k][0],hard\_w[i][k][0]+hard\_w[i][k][2]+1,5):

if display.get\_at((j,hard\_w[i][k][1]+hard\_w[i][k][3]+2))!=(255,255,255,255):

break

else:

pygame.draw.rect(display,white,hard\_w[i][k])

hard\_w[i][k][1]\*=1.005

display.blit(w\_guide[i],hard\_w[i][k])

def grav\_p\_detect():

global piglist,minion

for i in range(len(piglist)):

for j in range(piglist[i][0],piglist[i][0]+41):

if display.get\_at((j,piglist[i][1]+42))!=(255,255,255,255):

break

else:

pygame.draw.rect(display,white,piglist[i])

piglist[i][1]\*=1.005

display.blit(minion,piglist[i])

def pigcol():

global redpos,piglist,exterminator,redvelx,redvely

ct=0

for i in piglist:

if redpos[0]<=i[0]+i[2] and redpos[0]>=i[0] or redpos[0]+30>=i[0] and redpos[0]+30<=i[0]+i[2]:# end & start

#top

if redpos[1]+30>=i[1] and redpos[1]+30<=i[1]+30:

if ct not in exterminator:

exterminator.append(ct)

#bottom

elif redpos[1]<=i[1]+i[3] and redpos[1]>=i[1]+i[3]-20:

if ct not in exterminator:

exterminator.append(ct)

if redpos[1]<=i[1]+i[3] and redpos[1]>=i[1] or redpos[1]+30>=i[1] and redpos[1]+30<=i[1]+i[3]:# start & end

#left

if redpos[0]+30>=i[0] and redpos[0]+30<=i[0]+20:

if ct not in exterminator:

exterminator.append(ct)

#right

elif redpos[0]<=i[0]+i[2] and redpos[0]>=i[0]+i[2]-20:

if ct not in exterminator:

exterminator.append(ct)

ct+=1

def terminate():

global exterminator,piglist

for i in exterminator:

del piglist[i]

exterminator=[]

def destroyer(destroy):

global hard\_w,trash,score

for i in destroy:

trash.append(hard\_w[i[0]][i[1]])

del hard\_w[i[0]][i[1]]

score+=25

def newcol\_w():

global hard\_w,redpos,redvelx,redvely

destroy=[]

ct1=0

for j in hard\_w:

ct2=0

for i in j:

appendct=0

if redpos[0]<=i[0]+i[2] and redpos[0]>=i[0] or redpos[0]+30>=i[0] and redpos[0]+30<=i[0]+i[2]:# end & start

#top

if redpos[1]+30>=i[1] and redpos[1]+30<=i[1]+30:

if redvely>1.7:

if appendct==0:

destroy.append([ct1,ct2])

appendct+=1

redvely/=1.9

else:

redvely=e\*(-redvely)

redvelx=f\*redvelx

#bottom

elif redpos[1]<=i[1]+i[3] and redpos[1]>=i[1]+i[3]-20:

if redvely<-1.7:

if appendct==0:

destroy.append([ct1,ct2])

appendct+=1

redvely/=1.9

else:

redvely=e\*(-redvely)

redvelx=f\*redvelx

if redpos[1]<=i[1]+i[3] and redpos[1]>=i[1] or redpos[1]+30>=i[1] and redpos[1]+30<=i[1]+i[3]:# start & end

#left

if redpos[0]+30>=i[0] and redpos[0]+30<=i[0]+20:

if redvelx>1.7:

if appendct==0:

destroy.append([ct1,ct2])

appendct+=1

redvelx/=1.9

else:

redvely=f\*(redvely)

redvelx=e\*(-redvelx)

#right

elif redpos[0]<=i[0]+i[2] and redpos[0]>=i[0]+i[2]-20:

if redvelx<-1.7:

if appendct==0:

destroy.append([ct1,ct2])

appendct+=1

redvelx/=1.9

else:

redvely=f\*(redvely)

redvelx=e\*(-redvelx)

ct2+=1

ct1+=1

if destroy!=[]:

destroyer(destroy)

def newcol\_r():

global hard\_r,redpos,redvelx,redvely,e,f

for i in hard\_r:

if redpos[0]<=i[0]+i[2] and redpos[0]>=i[0] or redpos[0]+30>=i[0] and redpos[0]+30<=i[0]+i[2]:# end & start

#top

if redpos[1]+30>=i[1] and redpos[1]+30<=i[1]+30:

if redvely>0:

if redvely>0.7:

pygame.mixer.music.load("ouch.mp3")

pygame.mixer.music.play()

redvely=e\*(-redvely)

redvelx=f\*redvelx

if math.fabs(redvely)<0.1:

redvely=0

#bottom

elif redpos[1]<=i[1]+i[3] and redpos[1]>=i[1]+i[3]-20:

if redvely<0:

pygame.mixer.music.load("ouch.mp3")

pygame.mixer.music.play()

redvely=e\*(-redvely)

redvelx=f\*redvelx

if redpos[1]<=i[1]+i[3] and redpos[1]>=i[1] or redpos[1]+30>=i[1] and redpos[1]+30<=i[1]+i[3]:# start & end

#left

if redpos[0]+30>=i[0] and redpos[0]+30<=i[0]+20:

if redvelx>0:

pygame.mixer.music.load("ouch.mp3")

pygame.mixer.music.play()

redvely=f\*(redvely)

redvelx=e\*(-redvelx)

#right

elif redpos[0]<=i[0]+i[2] and redpos[0]>=i[0]+i[2]-20:

if redvelx<0:

pygame.mixer.music.load("ouch.mp3")

pygame.mixer.music.play()

redvely=f\*(redvely)

redvelx=e\*(-redvelx)

while app\_exit==False:

slingengaged=2

slingpast=2

grav=.065

gravengaged=0

fricgage=0

hard\_r=[]

hard\_w=[]

w\_guide=[]

trash=[]

colvel=[]

skip=0

collidevar=0

redcentre=[15,15]

redpos=[210,635]

redvelx,redvely=0,0

if play!=1:

redsub=4

bird\_vel=0

pigvel=[]

piglist=[]

exterminator=[]

score=0

if setup==1:

cmd=menu()

display.fill(white)

pygame.display.update()

if cmd==0:

break

elif cmd==1:

lvl1()

elif cmd==2:

lvl2()

setup=0

while game\_exit==False:

for event in pygame.event.get():#loop to handle events

if event.type==pygame.QUIT:

game\_exit=True

app\_exit=True

if event.type==pygame.KEYDOWN:

game\_exit=True

app\_exit=True

if event.type==pygame.MOUSEBUTTONDOWN and slingengaged!=0:

if pygame.mouse.get\_pos()[0] in range(int(redpos[0]),int(redpos[0]+30)):

if pygame.mouse.get\_pos()[1] in range(int(redpos[1]),int(redpos[1]+30)):

slingengaged=1

if event.type==pygame.MOUSEBUTTONUP:

if slingengaged==1:

redvelx,redvely=launchvel(redpos)

gravengaged=1

strapclear(redpos)

slingengaged=0

if slingengaged==1:

redclear(redpos)

strapclear(redpos)

if ((slingcentre[0]-pygame.mouse.get\_pos()[0])\*\*2+(slingcentre[1]-pygame.mouse.get\_pos()[1])\*\*2)\*\*0.5<=70:

redpos=[pygame.mouse.get\_pos()[0]-redcentre[0]+redvelx,pygame.mouse.get\_pos()[1]-redcentre[1]+redvely]

strap(redpos[0],redpos[1])

redclear(redpos)

slingrend(slingpos[0],slingpos[1])

draw()

terminate()

if collidevar>=3 and skip==0:

newcol\_r()

pigcol()

newcol\_w()

grav\_w\_detect()

grav\_p\_detect()

if slingengaged==0:

collidevar+=1

if slingengaged==0 or slingengaged==2:

if gravengaged==1 and fricgage==0:

redpos=[redpos[0]+redvelx,redpos[1]+redvely]

redvely+=grav

else:

redpos=[redpos[0]+redvelx,redpos[1]+redvely]

strap(slingpos[0]+10,slingpos[1]-10)

redrend(redpos[0],redpos[1])

try:

if skip==int((redvelx/redvely)\*redvely/2)+1:

skip=0

except ZeroDivisionError:

pass

if skip!=0:

skip+=1

slingpast=slingengaged

oob()

substitutes()

reddead()

if len(piglist)==0 or redsub==-1:

if len(piglist)==0:

win=1

if isreddead==True:

break

else:

loss=1

if isreddead==True:

break

pygame.display.update()

clock.tick(60)

isreddead=False

pygame.time.wait(1000)

if win==1:

pygame.mixer.music.load("lvlwin.mp3")

pygame.mixer.music.play()

win=0

score=score+(redsub+1)\*100

victory()

elif loss==1:

pygame.mixer.music.load("lvllose.mp3")

pygame.mixer.music.play(2)

loss=0

lose()

play+=1

pygame.quit()

quit()

SLICER MODULE:

import pygame

class blocks(object):

def wooden\_block(self):

img=pygame.image.load("wood.png")

img=pygame.transform.scale(img,(232,208))

img=img.subsurface(0,0,42,42)

return img

**SCREENSHOTS**

# C:\Users\Tintin\Documents\Python Programs\Angry birds\ScreenShot.pngC:\Users\Tintin\Documents\Python Programs\Angry birds\Screenshot2.png