```
import pygame, random, math, copy, pygame_textinput

screenWidth=1200
screenHeight=675
gravity=0
gravityOn=True
Drag=[0]
airDragOn=True
elasticity=0



def almostEqual(x,y,epsilon=10**--5):
    return abs(x-y)<=epsilon
def distance(x1,y1,x2,y2):
    return ((x2-x1)**2 + (y2-y1)**2 )**.5


#Used formula from Peter Collingridge (http://
www.petercollingridge.co.uk/pygame-physics-simulation/mass)
def addVectors(angle1, length1, angle2, length2):
    x  = math.sin(angle1) * length1 + math.sin(angle2) * length2
    y  = math.cos(angle1) * length1 + math.cos(angle2) * length2

    angle = 0.5 * math.pi - math.atan2(y, x)
    length  = math.hypot(x, y)

    return angle, length


def setRadius(p1,x,y):
    mag= (x**2+y**2)**.5


    if mag==0:
        return p1.radius

    udx, udy = x/mag, y/mag

    i=0
    px=x
    py=y
    counter=0
    go=True

    while go ==True:

        if px<0 or px>p1.width or py<0 or py>p1.height:
            go=False
        elif p1.mask.get_at((px,py)) !=0:
```

```
            go =False

        px+=udx
        py+=udy

        counter+=1

    return mag + counter


#Objects
class entity(pygame.sprite.Sprite):


    screenHeight= screenHeight
    screenWidth= screenWidth
    ground= 5*screenHeight/6

    global gravityOn
    global airDragOn
    global gravity
    global drag
    global elasticity


    def __init__(self,x,y):
        pygame.sprite.Sprite.__init__(self)
        self.cx=x
        self.cy=y
        self.xSpeed= random.randint(1,5) * (-1)**random.randint(1,9)
        self.ySpeed= random.randint(-5,5)
        self.highlight=False
        self.angSpeed= 0
        self.speed= (self.xSpeed**2+self.ySpeed**2)**.5
        self.mass= random.randint(10,50)
        self.kEnergy= self.mass*.5*(self.speed)**2
        self.angle= math.atan2(self.ySpeed,self.xSpeed) + math.pi/2
        self.elasticity= elasticity
        self.drag=1
        self.myfont = pygame.font.SysFont("arial", 25)
        self.additionalVectors=[ (3*math.pi/2,2)]
        self.xSpeed=math.sin(self.angle) * self.speed
        self.ySpeed=math.cos(self.angle) * self.speed

        self.r = random.randint(0, 255)
        self.g = random.randint(0, 255)
        self.b = random.randint(0, 255)
        self.color= (self.r,self.g,self.b)
        self.highColor= (255-self.r,255-self.g, 255-self.b)
```

```
    def eq(self,other):
        return isinstance(other,type(self)) and self.color ==
other.color
    def addGravity(self,height):

        if (almostEqual(self.cy+ height, entity.ground,3) and
self.speed<1.9 and gravityOn==True):
            self.cy= entity.ground-height
            self.speed=0

    def groundRebound(self,top,down):
        if self.cy +down>= entity.ground:
            self.cy = 2*(entity.ground - down) - self.cy
            self.angle = math.pi - self.angle
            self.speed *= elasticity
        elif self.cy-top<=0:
            self.cy = 2*top - self.cy
            self.angle = math.pi - self.angle
            self.speed *= elasticity

    #Also from Peter CollingRidge.
    def checkWalls(self,leftSide,rightSide):
        if self.cx-leftSide<=0:
            self.cx = 2*leftSide - self.cx
            self.angle = - self.angle
            self.speed *= elasticity
        elif self.cx+rightSide>=circle.screenWidth:
            self.cx = 2*(circle.screenWidth - rightSide) - self.cx
            self.angle = - self.angle
            self.speed *= elasticity

    def changeSpeed(self):
        self.angle= math.atan2(self.ySpeed,self.xSpeed) + math.pi/2
        self.speed= (self.xSpeed**2+self.ySpeed**2)**.5


    def changePosition(self,width,height):

        if gravityOn==True and self.speed !=0:
            (self.angle, self.speed) = addVectors(self.angle,
self.speed, math.pi,gravity)
        if airDragOn==True:
            self.speed *= self.drag
        self.xSpeed=math.sin(self.angle) * self.speed
        self.ySpeed=math.cos(self.angle) * self.speed
        self.cx += self.xSpeed
        self.cy -= self.ySpeed

        dims=(self.cx-width/2, self.cy-height/2,width,height)
        self.rect= pygame.Rect(dims)
```

```
    def move(self,x,y):
        self.cx=x
        self.cy=y

    def update(self):
        self.checkWalls()
        self.groundRebound()
        self.changePosition()
        self.addGravity()




    def collided(self,other):
        p1=self
        p2=other
        dx = p1.cx - p2.cx
        dy = p1.cy - p2.cy



        ##Got Collision formula from Peter Collingridge (http://
www.petercollingridge.co.uk/pygame-physics-simulation/mass)
        dist = math.hypot(dx, dy)
        angle = math.atan2(dy, dx) + 0.5 * math.pi
        total_mass = p1.mass + p2.mass
        (p1.angle, p1.speed) = addVectors(p1.angle, p1.speed*(p1.mass-
p2.mass)/total_mass, angle, 2*p2.speed*p2.mass/total_mass)
        (p2.angle, p2.speed) = addVectors(p2.angle, p2.speed*(p2.mass-
p1.mass)/total_mass, angle+math.pi, 2*p1.speed*p1.mass/total_mass)
        p1.speed *= elasticity
        p2.speed *= elasticity

        if not isinstance(p1,circle) and p1.mask.overlap(p2.mask,
(int(-dx),int(-dy))) !=None:
            x,y =p1.mask.overlap(p2.mask,(int(-dx),int(-dy)))
            radius1= setRadius(p1,x,y)
        else:
            radius1= p1.radius

        if not isinstance(p2,circle) and p2.mask.overlap(p1.mask,
(int(dx),int(dy))) !=None:

            x,y =p2.mask.overlap(p1.mask,(int(dx),int(dy)))
            radius2= setRadius(p2,x,y)
        else:
            radius2= p2.radius
        overlap = 0.5*(radius1 + radius2 - dist+1)
```

```
        p1.cx += math.sin(angle)*overlap
        p1.cy -= math.cos(angle)*overlap
        p2.cx -= math.sin(angle)*overlap
        p2.cy += math.cos(angle)*overlap
class circle(entity):
    def chars(self):
        dims=(self.cx-self.radius, self.cy-self.radius,2*self.radius,
2*self.radius)
        self.rect= pygame.Rect(dims)
        self.image= pygame.Surface((2*self.radius,
2*self.radius),pygame.SRCALPHA)
        self.image= self.image.convert_alpha()
        pygame.draw.circle(self.image, (self.r, self.g, self.b),
                           (self.radius, self.radius), self.radius)
        self.mask = pygame.mask.from_surface(self.image)
        self.drag = (self.mass/(self.mass + .0001)) ** self.radius

    def __init__(self,x=10,y=10):
        super().__init__(x,y)
        self.radius=0
        self.inertia= self.mass*self.radius**2/2
        self.image= pygame.Surface((2*self.radius,
2*self.radius),pygame.SRCALPHA)
        self.image= self.image.convert_alpha()


        dims=(self.cx-self.radius, self.cy-self.radius,2*self.radius,
2*self.radius)
        self.rect= pygame.Rect(dims)

        pygame.draw.circle(self.image, (self.r, self.g, self.b),
                           (self.radius, self.radius), self.radius)

    def setSize(self,x,y):
        self.radius= min(int(distance(self.cx,self.cy,x,y)),150)
        self.chars()

    def move(self,x,y):
        super().move(x,y)
        dims=(self.cx-self.radius, self.cy-self.radius,2*self.radius,
2*self.radius)
        self.rect= pygame.Rect(dims)
    def addGravity(self):
        super().addGravity(self.radius)

    def changePosition(self):
        super().changePosition(2*self.radius,2*self.radius)
    def checkWalls(self):
        super().checkWalls(self.radius,self.radius)
```

```
    def groundRebound(self):
        super().groundRebound(self.radius,self.radius)



    def drawHighlight(self,screen):
        pygame.draw.circle(screen, (255,255,255),
                           (int(self.cx), int(self.cy)),
int(self.radius)+1,3)

    def drawSettingSizeDim(self,screen):
        if self.radius>4:
            radius= "Radius: %d m" %(self.radius)
            text = self.myfont.render(radius, 1, (255,255,255))
            screen.blit(text, (self.cx, self.cy))
class box(entity):

    def chars(self):

        #self.inertia= self.mass* (self.height**2 + self.width**2)/12

        dims=(self.cx-self.width/2, self.cy-self.height/2,
        self.width,self.height)
        self.radius=((self.width**2 + self.height**2)**.5)/2

        self.radius = (self.radius + self.width+ self.height)/3
        self.rect= pygame.Rect(dims)

        self.image=
pygame.Surface((self.width+2,self.height+2),pygame.SRCALPHA)
        self.image= self.image.convert_alpha()

        pygame.draw.rect(self.image, (self.r, self.g, self.b),
(0,0,self.width,self.height))
        self.mask = pygame.mask.from_surface(self.image)
    def __init__(self, x=10,y=10):
        super().__init__(x,y)
        self.width=0
        self.height=0

        self.chars()


    def setSize(self,x,y):

        self.height= min(int(abs(self.cy-y)),150)
        self.width= min(int(abs(self.cx-x)),150)
        self.chars()
        self.drag = (self.mass/(self.mass + .0001)) ** self.radius
    def addGravity(self):
```

```
        super().addGravity(self.height/2)


    def move(self,x,y):
        super().move(x,y)
        dims=(self.cx-self.width/2-1, self.cy-self.height/
2-1,self.width+1,self.height+1)
        self.rect= pygame.Rect(dims)


    def changePosition(self):
        super().changePosition(self.width,self.height)
    def checkWalls(self):
        super().checkWalls(self.width/2,self.width/2)
    def groundRebound(self):
        super().groundRebound(self.height/2,self.height/2)



    def drawHighlight(self,screen):
        pygame.draw.rect(screen, (255,255,255),
                         (self.cx-self.width/2, self.cy-self.height/
2,self.width,self.height),3)
    def drawSettingSizeDim(self,screen):
        width= "Width: %d m" %(self.width)
        text = self.myfont.render(width, 1, (255,255,255))
        screen.blit(text, (self.cx, self.cy))
        height= "Height: %d m" %(self.height)
        text = self.myfont.render(height, 1, (255,255,255))
        screen.blit(text, (self.cx, self.cy+20))
class triangle(entity):

    def chars(self):


        self.top= (self.radius, 0)
        self.left= (0, 2*self.radius)
        self.right= (2*self.radius,2*self.radius)

        dims=(self.cx-self.radius, self.cy-self.radius,
        2*self.radius,2*self.radius)
        self.points=[self.top,self.left,self.right]
        self.rect= pygame.Rect(dims)


        self.image= pygame.Surface((2*self.radius,
2*self.radius),pygame.SRCALPHA)
        self.image= self.image.convert_alpha()
```

```
        pygame.draw.polygon(self.image, (self.r, self.g, self.b),
self.points)

        self.mask = pygame.mask.from_surface(self.image)
        self.drag = (self.mass/(self.mass + .0001)) ** self.radius
    def __init__(self, x=0,y=0):
        super().__init__(x,y)

        self.side=0
        self.radius= self.side * (3**.5/2)
        self.width=int(2*self.radius)
        self.height=int(2*self.radius)

        self.r = random.randint(0, 255)
        self.g = random.randint(0, 255)
        self.b = random.randint(0, 255)

        self.chars()



    def move(self,x,y):
        super().move(x,y)
        dims=(self.cx-self.width/2, self.cy-self.height/
2,self.width,self.height)
        self.rect= pygame.Rect(dims)

    def setSize(self,x,y):
        self.side= min(int(distance(self.cx,self.cy,x,y)),150)
        self.radius= self.side * (3**.5/2)
        self.width=2*self.radius
        self.height=2*self.radius
        self.chars()
    def addGravity(self):
        super().addGravity(self.radius)
    def checkWalls(self):
        super().checkWalls(self.radius,self.radius)

    def changePosition(self):
        super().changePosition(2*self.radius, 2*self.radius)

    def groundRebound(self):
        super().groundRebound(self.radius,self.radius)
    def modify(self):
        pygame.draw.polygon(self.image, (0,0,0), self.points,2)
    def drawHighlight(self,screen):
        #screen.blit(self.image2,(self.cx-self.radius,self.cy-
self.radius))
```

```
        top= (self.cx, self.cy-self.radius)
        left= (self.cx-self.radius, self.cy+self.radius)
        right= (self.cx+self.radius,self.cy+self.radius)
        dims=(top,left,right)
        pygame.draw.polygon(screen,(255,255,255),dims,3)
    def drawSettingSizeDim(self,screen):

        side= "Side Length: %d m" %(self.side)
        text = self.myfont.render(side, 1, (255,255,255))
        screen.blit(text, (self.cx, self.cy))
class polygon(entity):

    def __init__(self,x,y,numPoints):
        super().__init__(x,y)
        self.points=[]
        self.undoList=[]
        self.color= [self.r,self.g,self.b]
        self.numPoints=numPoints
        self.relativePoints=[]
        self.rect= pygame.Rect(x,y,5,5)
        self.image= pygame.Surface((5,5),pygame.SRCALPHA)
        self.image= self.image.convert_alpha()
        self.mask = pygame.mask.from_surface(self.image)


    def setDims(self):

        upMost=screenHeight
        downMost=0
        leftMost=screenWidth
        rightMost=0
        for i in self.points:
            if i[0]>rightMost: rightMost=i[0]
            if i[0]<leftMost:  leftMost=i[0]
            if i[1]>downMost:  downMost=i[1]
            if i[1]<upMost:    upMost=i[1]

        self.width= rightMost-leftMost
        self.height= downMost-upMost

        self.cx= leftMost + self.width/2
        self.cy= upMost + self.height/2
        self.rect= pygame.Rect(leftMost,upMost,self.width,self.height)

        self.leftMost=leftMost
        self.rightMost=rightMost
        self.upMost=upMost
        self.downMost=downMost
        self.radius=  (self.height**2 + self.width**2)**.5
```

```
        self.drag = (self.mass/(self.mass + .0001)) ** self.radius

        if self.width>0 and self.height>0:
            self.image=
pygame.Surface((self.width,self.height),pygame.SRCALPHA)
            self.image= self.image.convert_alpha()

    def setPoints(self,x,y):

        if len(self.points)<self.numPoints:
            self.points.append([x,y])
            self.setDims()
            self.relativePoints= copy.deepcopy(self.points)
            self.setRelativePoints()
            return True
        else:
            return False



    def move(self,x,y):
        super().move(x,y)
        self.rect= pygame.Rect(self.cx-self.width/2,self.cy-
self.height/2,self.width,self.height)

    def setRelativePoints(self):

        for i in range(len(self.points)):
            self.relativePoints[i][0]= int(self.points[i][0]-
(self.cx-self.width/2))
            self.relativePoints[i][1]= int(self.points[i][1]-
(self.cy-self.height/2))

        if len(self.points)>2:
            pygame.draw.polygon(self.image,self.color,
self.relativePoints,)
            self.mask = pygame.mask.from_surface(self.image)
        else:
            for point in self.relativePoints:
                pygame.draw.circle(self.image,(255,255,255), point,5)

    def addGravity(self):
        super().addGravity(self.height/2)
    def changePosition(self):
        super().changePosition(self.width,self.height)
    def checkWalls(self):
        #leftSide= self.cx-self.leftMost
        #rightSide= self.rightMost-self.cx
        super().checkWalls(self.width/2,self.width/2)
```

```
    def groundRebound(self):
        super().groundRebound(self.height/2,self.height/2)

    def modify(self):
        pygame.draw.polygon(self.image,(0,0,0), self.relativePoints,2)
    def drawHighlight(self,screen):
        pygame.draw.polygon(self.image,
(255,255,255),self.relativePoints,2)

#Screen Images
class Cloud(object):
    def Dims(self,leftBound,bound=None):
        if bound==None:
            bound= self.leftBound
        self.x=random.randint(2*leftBound,bound)
        self.y=random.randint(0, int(.2*self.screenHeight))
        self.speed=random.randint(20,30)/100

    def __init__(self,width,height):
        self.image= pygame.image.load("cloud1.png").convert_alpha()
        self.w,self.h =self.image.get_size()
        factor=random.randint(5,10)/100
        self.leftBound=int(-1*factor*self.w)
        self.screenWidth=width
        self.screenHeight=height
        self.Dims(self.leftBound,width)
        self.image=pygame.transform.scale(self.image,
            (int(self.w* factor), int(self.h*factor)))


    def update(self):
        if self.x >= self.screenWidth:
            self.Dims(self.leftBound)
        else:
            self.x+=self.speed
    def draw(self,screen):
        screen.blit(self.image,(self.x,self.y))
class Space(pygame.sprite.Sprite):


    def __init__(self,width,height):
        pygame.sprite.Sprite.__init__(self)
        self.image= pygame.image.load("space.jpg")
        self.w,self.h =self.image.get_size()
        self.width=width
        self.height=height

        self.Wfactor= self.width/self.w
        self.hFactor= self.height/self.h
```

```
        self.image=pygame.transform.scale(self.image,
            (int(self.w* self.Wfactor), int(self.h*self.hFactor)))

        self.image2= pygame.image.load("cartoon earth.png")
        self.wEarth,self.hEarth= self.image2.get_size()

        self.image2=pygame.transform.scale(self.image2,
            (int(self.wEarth* .5), int(self.hEarth* .5)))

    def draw(self,screen):
        screen.blit(self.image,(0,0))
        screen.blit(self.image2,(self.width/2-self.wEarth/
2,self.height/2- self.hEarth/2))
class Mars(pygame.sprite.Sprite):

    def __init__(self,width,height):
        pygame.sprite.Sprite.__init__(self)
        self.image= pygame.image.load("mars.jpeg")
        self.w,self.h =self.image.get_size()
        self.width=width
        self.height=height

        self.Wfactor= self.width/self.w
        self.hFactor= self.height/self.h

        self.image=pygame.transform.scale(self.image,
            (int(self.w* self.Wfactor), int(self.h*self.hFactor)))


    def draw(self,screen):
        screen.blit(self.image,(0,0))
class Jupiter(pygame.sprite.Sprite):

    def __init__(self,width,height):
        pygame.sprite.Sprite.__init__(self)
        self.image= pygame.image.load("jupiter.jpg")
        self.w,self.h =self.image.get_size()
        self.width=width
        self.height=height

        self.Wfactor= self.width/self.w
        self.hFactor= self.height/self.h

        self.image=pygame.transform.scale(self.image,
            (int(self.w* self.Wfactor), int(self.h*self.hFactor)))


    def draw(self,screen):
        screen.blit(self.image,(0,0))
```

```
class MainScreen(pygame.sprite.Sprite):

    def __init__(self,width,height):
        pygame.sprite.Sprite.__init__(self)
        self.image= pygame.image.load("Main Screen.png")
        self.w,self.h =self.image.get_size()
        self.width=width
        self.height=height

        self.Wfactor= self.width/self.w
        self.hFactor= self.height/self.h

        self.image=pygame.transform.scale(self.image,
            (int(self.w* self.Wfactor), int(self.h*self.hFactor)))

    def draw(self,screen):
        screen.blit(self.image,(0,0))
class HelpScreen(pygame.sprite.Sprite):


    def __init__(self,width,height,screen):
        pygame.sprite.Sprite.__init__(self)
        self.image= pygame.image.load(screen)
        self.w,self.h =self.image.get_size()
        self.width=width
        self.height=height

        self.Wfactor= self.width/self.w
        self.hFactor= self.height/self.h

        self.image=pygame.transform.scale(self.image,
            (int(self.w* self.Wfactor), int(self.h*self.hFactor)))

    def draw(self,screen):
        screen.blit(self.image,(0,0))


#Buttons
class button(pygame.sprite.Sprite):
    def __init__(self,x,y,width,height,color):
        pygame.sprite.Sprite.__init__(self)
        self.x=x
        self.y=y
        self.width=width
        self.height=height
        self.color= color


        self.image=
pygame.Surface([self.width,self.height],pygame.SRCALPHA)
```

```
        self.image= self.image.convert_alpha()



    def isClicked(self,clickX,clickY):

        return (clickX>=self.x  and clickX<=self.x+self.width and
                clickY>=self.y and clickY<=self.y+self.height)

        def __repr__(self):

            return self.name
class CircleButton(button):

    def __init__(self,x,y,width,height,color):
        super().__init__(x,y,width,height,color)
        self.name= "drawCircle"
        self.radius= self.width//2
        self.image= pygame.Surface((2*self.radius,
2*self.radius),pygame.SRCALPHA)
        self.image= self.image.convert_alpha()
        dims=(self.x, self.y,self.width,self.height)
        self.rect= pygame.Rect(dims)

        pygame.draw.circle(self.image, color,
                           (self.radius, self.radius), self.radius,)
class SquareButton(button):
    def __init__(self,x,y,width,height,color):
        super().__init__(x,y,width,height,color)
        self.name= "drawSquare"
        self.dims=(self.x, self.y,self.width,self.height)
        self.rect= pygame.Rect(self.dims)


        self.image= pygame.Surface((self.width,self.height))
        self.image.fill(self.color)
        #self.image= self.image.convert_alpha()
        pygame.draw.rect(self.image, self.color, self.dims)
        pygame.draw.rect(self.image, (0,0,0), self.dims,2)
class TriangleButton(button):
    def __init__(self,x,y,width,height,color):
        super().__init__(x,y,width,height,color)
        self.name= "drawTriangle"
        self.radius= self.width/2
        self.cx, self.cy= self.x+ self.radius, self.y +self.radius
        self.dims=(self.x, self.y,self.width,self.height)

        self.top= (self.cx, self.cy-self.radius)
        self.left= (self.cx-self.radius, self.cy+self.radius)
```

```
        self.right= (self.cx+self.radius, self.cy+self.radius)


        self.top= (self.radius, 0)
        self.left= (0, 2*self.radius)
        self.right= (2*self.radius,2*self.radius)

        self.points=[self.top,self.left,self.right]


        self.rect= pygame.Rect(self.dims)
        pygame.draw.polygon(self.image, color,self.points )
class IconButton(button):

    def __init__(self,x,y,width,height,color,name,imageName):
        super().__init__(x,y,width,height,color)
        self.name= name
        self.radius= self.width//2
        self.image= pygame.image.load(imageName)
        self.image= self.image.convert_alpha()
        dims=(self.x, self.y,self.width,self.height)
        self.rect= pygame.Rect(dims)

        self.w,self.h =self.image.get_size()

        self.Wfactor= self.width/self.w
        self.hFactor= self.height/self.h

        self.image=pygame.transform.scale(self.image,
            (int(self.w* self.Wfactor), int(self.h*self.hFactor)))

    def draw(self,screen):
        screen.blit(self.image,(self.x,self.y))
class DeleteButton(IconButton):
    def __init__(self,x,y,width,height,color,name,imageName):
        super().__init__(x,y,width,height,color,name,imageName)
class Delete(pygame.sprite.Sprite):
    def __init__(self,x,y):
        pygame.sprite.Sprite.__init__(self)
        self.rect=pygame.Rect(0,0,3,3)
        self.image= pygame.Surface([3,3])
        self.image.fill((0,0,0))
        #pygame.draw.rect(self.image, (0,0,0), self,,2)
        self.mask = pygame.mask.from_surface(self.image)

    def setPoint(self,x,y):
        self.x, self.y= x,y
        self.rect=pygame.Rect(self.x,self.y,2,2)
class PolygonButton(IconButton):
```

```
    def __init__(self,x,y,width,height,color,name,imageName):
        super().__init__(x,y,width,height,color,name,imageName)
class MouseButton(IconButton):

    def __init__(self,x,y,width,height,color,name,imageName):
        super().__init__(x,y,width,height,color,name,imageName)
class HomeButton(IconButton):
    def __init__(self,x,y,width,height,color,name,imageName):
        super().__init__(x,y,width,height,color,name,imageName)
class GravityButton(IconButton):
    def __init__(self,x,y,width,height,color,name,imageName):
        super().__init__(x,y,width,height,color,name,imageName)
class AirDragButton(IconButton):
    def __init__(self,x,y,width,height,color,name,imageName):
        super().__init__(x,y,width,height,color,name,imageName)
class OnImage(IconButton):
    def __init__(self,x,y,width,height,color,name,imageName):
        super().__init__(x,y,width,height,color,name,imageName)


class Mouse(pygame.sprite.Sprite):
    def __init__(self):
        pygame.sprite.Sprite.__init__(self)
        self.rect= pygame.Rect((0,0,1,1))
        self.image= pygame.Surface((1,1))
        #self.image= self.image.convert_alpha()

        pygame.draw.rect(self.image, (0,0,0), (0,0,1,1))

    def setDims(self,x,y):
        self.rect=pygame.Rect(x,y,1,1)

#Format taken from the pygame opitional lecture site 112
class PygameGame(object):


    global gravityOn
    global airDragOn
    global gravity
    global drag


    def __init__(self, width= screenWidth, height= screenHeight,
fps=100, title="Physics Engine"):
        self.width = width
        self.height = height
        self.fps = fps
        self.title = title
        pygame.init()
        self.myfont = pygame.font.SysFont("arial", 30)
```

```
        self.colors=dict()
        self.colors['lightblue']=(115,140,255)
        self.colors['black']=(0,0,0)
        self.colors['green']=(1,166,17)
        self.colors['grey']= (70,130,180)
        self.colors['moongrey']= (52,61,70)
        self.colors["yellow"]=(255,255,0)
        self.colors["orange"]=(255,165,0)
        self.colors["jupiterColor"]=(237,155,114)
        self.colors["clay"]=(152,45,25)
        self.clouds=[]
        self.dataChange= pygame_textinput.TextInput()
        self.dataChangeOn=False


        self.dataChangeinstructionText="Change Values of Object (i.e
'speed35'): "
        self.print =
self.myfont.render(self.dataChangeinstructionText, 1, (255,255,255))
        self.prevScreen=None
        self.drawDim=False
        self.mode= "draw"
        self.page=0
        self.isPaused= True
        self.entities=pygame.sprite.Group()
        self.space= Space(self.width,self.height)
        self.mainScreen= MainScreen(self.width,self.height)
        self.mars= Mars(self.width,self.height)
        self.jupiter= Jupiter(self.width,self.height)

        self.h1= HelpScreen(self.width,self.height,"h1.png")
        self.h2= HelpScreen(self.width,self.height,"h2.png")
        self.h3= HelpScreen(self.width,self.height,"h3.png")
        self.h4= HelpScreen(self.width,self.height,"h4.png")

        self.helpScreens= [self.h1,self.h2,self.h3,self.h4]
        self.modeButton=None
        self.place="main"
        self.drawingPolygon=False
        self.settingPoints=False
        self.deleteGroup=pygame.sprite.Group()
        self.textinput = pygame_textinput.TextInput()
        self.textinput2 = pygame_textinput.TextInput()
        self.buttons = pygame.sprite.Group()
        self.delete= Delete(0,0)
        self.instructionText="Number of Points/Sides: "
        self.label = self.myfont.render(self.instructionText, 1,
(255,255,255))
        self.deleteGroup.add(self.delete)
        self.currObject=None
```

```
        self.highObject=None

        self.drawInputs=False

    def analyzeText(self,text):

        for c in text:
            if c not in "0123456789":
                self.instructionText= "Invalid Input: "
                return False
        num= int(text)
        if num>10 or num<3:
            self.instructionText= "Invalid Input: "
            return False

        self.instructionText="Number of Points/Sides: "
        self.currObject.numPoints=num
        self.settingPoints=False

    def analyze3(self,data,num):

        if num != '':
            num= int(num)
        if data == "speed" or data== "Speed":
            num =min(num,50)
            self.highObject.speed= num
        elif data == "xSpeed" or data== "XSpeed" or data== "xspeed":
            num =min(num,50)
            self.highObject.xSpeed=num
            self.highObject.changeSpeed()
        elif data== "ySpeed":
            num =min(num,50)
            self.highObject.ySpeed=num
            self.highObject.changeSpeed()

        elif data== "radius":
            if num<6: num=6
            num =min(num,150)
            if isinstance(self.highObject, circle):

                self.highObject.radius=int(num)
                self.highObject.chars()
        elif data== "mass" or data=="Mass":
            if num<1: num=1
            num =min(num,500)
            self.highObject.mass=int(num)
            self.highObject.chars()
        elif data== ("width" or "Width") and
isinstance(self.highObject, box):
            if num<6: num=6
```

```
            num =min(num,150)
            self.highObject.width= int(num)
            self.highObject.chars()
        elif data== ("height" or "Height") and
isinstance(self.highObject, box):
            if num<6: num=6
            num =min(num,150)
            self.highObject.height= int(num)
            self.highObject.chars()
        self.dataChangeOn=False
        self.stop=True
    def analyzeText2(self,text):
        text= text.strip()
        data=''
        num=''
        s=''
        for c in text:
            if c != " ":
                s+=c

        for i, c in enumerate(s):
            if c.isalpha():
                data+=c
            elif c.isdigit():
                num+=c
                if i+1 < len(s) and not s[i+1].isdigit():
                    self.analyze3(data,num)
                    self.analyzeText2(s[i+1:])

        self.analyze3(data,num)

    def checkModifyEntities(self,x,y):
        self.mouse.setDims(x,y)
        self.highObject=None
        l=pygame.sprite.spritecollide(self.mouse,self.entities,False)
        for i in l:
            self.highObject=i

    def drawPause(self,screen):
        if self.place != "main" or self.place !="help":
            if self.isPaused==True:
                self.pauseText=self.myfont.render("Paused", 1,
(255,255,255))
            else:
                self.pauseText=self.myfont.render("Simulating...", 1,
(255,255,255))
            screen.blit(self.pauseText,(self.width-150,10))
    def drawDataList(self,screen):
        n=len(self.dataList)
        dims=(self.iX-10,self.iY-10,200,20 + n*30)
```

```
        pygame.draw.rect(screen,(0,0,128),dims )
        for i, data in enumerate(self.dataList):
            text = self.myfont.render(data, 1, (255,255,255))
            screen.blit(text, (self.iX, self.iY + i*30))
    def checkDrawInputs(self,x,y):

        speed="Speed: %d m/s " %(self.highObject.speed)
        xSpeed="xSpeed: %d m/s " %(self.highObject.xSpeed)
        ySpeed="ySpeed: %d m/s" %(self.highObject.ySpeed)
        mass="Mass: %d kg" %(self.highObject.mass)
        self.dataList=[speed,xSpeed,ySpeed,mass]

        if isinstance(self.highObject,circle):
            radius= "Radius: %d m" %(self.highObject.radius)
            self.dataList.append(radius)
        if isinstance(self.highObject,box):
            width= "Width: %d m" %(self.highObject.width)
            height= "Height: %d m" %(self.highObject.height)
            self.dataList.append(width)
            self.dataList.append(height)
        if isinstance(self.highObject,triangle):
            side= "Side Length: %d m" %(self.highObject.side)
            self.dataList.append(side)


        self.iX=x
        self.iY=y
        self.drawInputs= not self.drawInputs
        self.stop=False

    def drawChangeInstructions(self,screen):
        if self.dataChangeOn==True:
            if self.dataChange.update(pygame.event.get()):
                self.analyzeText2(self.dataChange.get_text())
            screen.blit(self.print, (10, 400))
            screen.blit(self.dataChange.get_surface(), (10, 420 ))

        #Draw Functions
    def highlightButton(self,screen):
        #pygame.draw.rect(screen,(192,192,192),(0,0,50,300), 4)
        pygame.draw.polygon(screen, (192,192,192),((0,0),(50,0),
(50,290),(100,290),(100,380),(0,380)),4)
        if self.modeButton !=None:
            button=self.modeButton
            pygame.draw.rect(screen,self.colors['grey'],

(button.x-1,button.y-1,button.width+1,button.height+1),2)

        if self.highObject != None:
            self.highObject.drawHighlight(screen)
```

```
    def drawPointInstructions(self,screen):
        if self.settingPoints==True:
            if self.textinput.update(pygame.event.get()):
                self.analyzeText(self.textinput.get_text())
            self.label = self.myfont.render(self.instructionText, 1,
(255,255,255))
            screen.blit(self.label, (10, 400))
            screen.blit(self.textinput.get_surface(), (10, 420 ))
    def drawGravityandDragState(self,screen):
        if gravityOn==True:
            screen.blit(self.onImage.image,(50,300))
        else:
            screen.blit(self.deleteButton.image,(50,300))
        if airDragOn==True:
            screen.blit(self.onImage.image,(50,340))
        else:
            screen.blit(self.deleteButton.image,(50,340))
    def drawInGameParts(self,screen):
        self.entities.draw(screen)
        self.highlightButton(screen)
        self.buttons.draw(screen)
        self.drawPointInstructions(screen)
        self.drawChangeInstructions(screen)
        self.drawGravityandDragState(screen)
        if self.drawInputs==True:
            self.drawDataList(screen)
            if self.stop==False:
                self.dataChangeOn=True
        self.drawPause(screen)


    #Screens
    def drawMain(self,screen):

        def drawBackground(self,screen):
            self.mainScreen.draw(screen)
        drawBackground(self,screen)

    def drawHelpScreen(self,screen):
        self.helpScreens[self.page].draw(screen)

    def drawEarth(self,screen):
        def drawSky(self,screen):
            dims=(0,0,self.width,self.height)
            color=self.colors['lightblue']
            pygame.draw.rect(screen,color,dims,0)
        def drawGrass(self,screen):
            #Draws green layer
            dims=(0,5*self.height/6,self.width,self.height/6 +1)
            color=self.colors['green']
```

```
        pygame.draw.rect(screen,color,dims,0)
        #Draws top black outline
        start=(0,5*self.height/6)
        end=(self.width,5*self.height/6)
        pygame.draw.line(screen,(0,0,0),start,end,1)

    def drawClouds(self,screen):
        for cloud in self.clouds:
            cloud.draw(screen)

    drawSky(self,screen)
    drawGrass(self,screen)
    drawClouds(self,screen)
def drawMoon(self,screen):

    def drawSpace(self,screen):
        self.space.draw(screen)
        # dims=(0,0,self.width,self.height)
        # color=self.colors['black']
        # pygame.draw.rect(screen,color,dims,0)
    def drawGround(self,screen):
        #Draws green layer
        dims=(0,5*self.height/6,self.width,self.height/6 +1)
        color=self.colors['moongrey']
        pygame.draw.rect(screen,color,dims,0)
        #Draws top black outline
        start=(0,5*self.height/6)
        end=(self.width,5*self.height/6)
        pygame.draw.line(screen,(0,0,0),start,end,2)
    drawSpace(self,screen)
    drawGround(self,screen)
def drawMars(self,screen):
    def drawMarsSky(self,screen):
        self.mars.draw(screen)

    def drawDirt(self,screen):
        #Draws green layer
        dims=(0,5*self.height/6,self.width,self.height/6 +1)
        color=self.colors['clay']
        pygame.draw.rect(screen,color,dims,0)
        #Draws top black outline
        start=(0,5*self.height/6)
        end=(self.width,5*self.height/6)
        pygame.draw.line(screen,(0,0,0),start,end,2)

    drawMarsSky(self,screen)
    drawDirt(self,screen)
def drawJupiter(self,screen):
    def drawJupiterSky(self,screen):
        self.jupiter.draw(screen)
```

```
    def drawGround(self,screen):
        #Draws green layer
        dims=(0,5*self.height/6,self.width,self.height/6 +1)
        color=self.colors['jupiterColor']
        pygame.draw.rect(screen,color,dims,0)
        #Draws top black outline
        start=(0,5*self.height/6)
        end=(self.width,5*self.height/6)
        pygame.draw.line(screen,(0,0,0),start,end,2)

    drawJupiterSky(self,screen)
    drawGround(self,screen)


def redrawAll(self,screen):
    if self.place== "main":
        self.drawMain(screen)
    elif self.place=="earth":
        self.drawEarth(screen)
        self.drawInGameParts(screen)
    elif self.place=="moon":
        self.drawMoon(screen)
        self.drawInGameParts(screen)
    elif self.place=='mars':
        self.drawMars(screen)
        self.drawInGameParts(screen)
    elif self.place=='jupiter':
        self.drawJupiter(screen)
        self.drawInGameParts(screen)
    elif self.place=='help':
        self.drawHelpScreen(screen)



    self.deleteGroup.draw(screen)

def setForces(self):
    global gravity
    global elasticity

    if self.place== "earth":
        gravity=1
        elasticity=.8
    elif self.place=="mars":
        gravity=.3
        elasticity=.9
    elif self.place== 'moon':
        gravity=.1
        elasticity=.95
```

```
        elif self.place== 'jupiter':
            gravity=2.5
            elasticity=.7

    def isKeyPressed(self, key):
        ''' return whether a specific key is being held '''
        return self._keys.get(key, False)
    def keyPressed(self, event):

        self.drawDim=True
        global gravity
        global drag
        global elasticity
        if event == pygame.K_p and self.place != "main" and
self.place != "help":
            self.isPaused= not self.isPaused
            self.drawDim=False

        elif event == pygame.K_e:
            self.place="earth"

        elif event == pygame.K_n:
            self.place="mars"

        elif event == pygame.K_m:
            self.place="moon"

        elif event == pygame.K_j:
            self.place="jupiter"

        elif event == pygame.K_h:
            if self.place=="help":
                if self.prevScreen != None:
                    self.place= self.prevScreen
            else:
                self.isPaused=True
                self.prevScreen=self.place
                self.place= "help"
        elif event == pygame.K_b:
            self.isPaused=True
            self.place= "main"

        elif ((event ==pygame.K_RIGHT or event==pygame.K_DOWN) and
self.place=="help"):
            if self.page==3:
                self.page=0
            else:
                self.page+=1
```

```
        elif ((event ==pygame.K_LEFT or event==pygame.K_UP) and
self.place=="help"):
            if self.page==0:
                self.page=3
            else:
                self.page-=1

    def mousePressed(self,x,y,currObject):

        if self.checkButtonPressed(x,y,self.buttons)== True:
            self.highObject=None
            return None
        else:

            if self.mode == "drawCircle":
                self.drawingPolygon=False
                self.modeButton=self.circleButton
                return circle(x,y)
            elif self.mode== "drawSquare":
                self.drawingPolygon=False
                self.modeButton= self.squareButton
                return box(x,y)
            elif self.mode== "drawTriangle":
                self.drawingPolygon=False
                self.modeButton= self.triangleButton
                return triangle(x,y)
            elif self.mode== "delete":
                self.drawingPolygon=False
                self.modeButton= self.deleteButton
                self.checkDelete(x,y)
            elif self.mode=="drawPolygon":
                if self.drawingPolygon==False:
                    self.drawingPolygon=True
                    self.settingPoints=True
                    return polygon(x,y,3)
                else:
                    return currObject

            elif self.mode=="mouse":
                self.checkModifyEntities(x,y)
    def checkButtonPressed(self,clickX,clickY,buttons):
        global gravityOn
        global airDragOn
        counter=0
        for button in buttons:
            if button.isClicked(clickX,clickY):
                self.mode=button.name
                self.modeButton=button
                if isinstance(button,GravityButton):
                    gravityOn= not gravityOn
```

```
                elif isinstance(button,AirDragButton):
                    airDragOn= not airDragOn
                elif isinstance(button,HomeButton):
                    self.place="main"
                    self.modeButton=None
                return True
        return False

    def init(self):

        #Create Buttons
        self.homeButton=
HomeButton(10,20,30,30,self.colors["grey"],"home", "home.png")

self.circleButton=CircleButton(10,60,30,30,self.colors["yellow"])
        self.squareButton=
SquareButton(10,100,30,30,self.colors["green"])
        self.triangleButton=
TriangleButton(10,140,30,30,self.colors["orange"])

self.polygonButton=PolygonButton(10,180,30,30,self.colors["grey"],"dra
wPolygon","Hexagon.png")

self.mouseButton=MouseButton(10,220,30,30,self.colors["grey"],"mouse",
"mouse.png")
        self.deleteButton=
DeleteButton(10,260,30,30,self.colors["grey"],"delete","x.png")

        self.gravityButton=
GravityButton(10,300,30,30,self.colors["grey"],"gravity", "apple.png")
        self.airDragButton=
AirDragButton(10,340,30,30,self.colors["grey"], "airDrag",
"airDrag.png")
        self.onImage= OnImage(50,260,30,30,self.colors["grey"], "On",
"check.png")
        self.mouse=Mouse()

        self.buttons.add(self.circleButton)
        self.buttons.add(self.squareButton)
        self.buttons.add(self.triangleButton)
        self.buttons.add(self.deleteButton)
        self.buttons.add(self.polygonButton)
        self.buttons.add(self.mouseButton)
        self.buttons.add(self.gravityButton)
        self.buttons.add(self.airDragButton)
        self.buttons.add(self.homeButton)

        #Create clouds
        for i in range(random.randint(5,9)):
            self.clouds.append(Cloud(self.width,self.height))
```

```
    def checkDelete(self,x,y):
        self.delete.setPoint(x,y)
        l=pygame.sprite.spritecollide(self.delete,self.entities,
pygame.sprite.collide_mask)
        for i in l:
            self.entities.remove(i)

    def checkDrawDim(self,screen):
        if (self.currObject != None and not
isinstance(self.currObject,polygon) and self.drawDim==True):
            self.currObject.drawSettingSizeDim(screen)

    def selectSurface(self,x,y):

        if x>=35 and x<=235 and y>=415 and y<=652:
            self.place= "earth"
        elif x>=329 and x<=529 and y>=415 and y<=652:
            self.place= "moon"
        elif x>=610 and x<=810 and y>=415 and y<=652:
            self.place= "mars"
        elif x>=916 and x<=1116 and y>=415 and y<=652:
            self.place= "jupiter"
    def run(self):

        clock = pygame.time.Clock()
        screen = pygame.display.set_mode((self.width, self.height))
        # set the title of the window
        pygame.display.set_caption(self.title)


        # stores all the keys currently being held dow
        def checkcollisions(group):
            # smallGroup=group.copy()
            # for entity in group:

            #     smallGroup.remove(entity)
            #
l=pygame.sprite.spritecollide(entity,smallGroup,False,
pygame.sprite.collide_mask)

            #     for otherEntity in l:
            #         entity.collided(otherEntity)
            #         smallGroup.remove(otherEntity)
            # smallGroup.empty()
            for entity in group:
                if (isinstance(entity,circle) or
isinstance(entity,triangle)) and entity.radius<5:
                    self.entities.remove(entity)
                    continue
```

```
                l=pygame.sprite.spritecollide(entity,group,False,
pygame.sprite.collide_mask)

                for otherEntity in l:
                    if not entity.eq(otherEntity):
                        entity.collided(otherEntity)



        def timerFired(self,time):
            if self.place != "main" and self.place !="help":
                self.setForces()
                if self.isPaused==False:
                    self.entities.update()
                    checkcollisions(self.entities)
                if self.drawDim==True:
                    self.checkDrawDim(screen)

                if self.place=="earth":
                    for cloud in self.clouds:
                        cloud.update()



        # call game-specific initialization
        self.init()


        playing = True
        while playing:
            time = clock.tick(self.fps)

            self.redrawAll(screen)
            timerFired(self,time)

            #Handle Events
            for event in pygame.event.get():


                if (event.type == pygame.MOUSEBUTTONDOWN and
event.button==1):
                    if self.place=="main":
                        self.selectSurface(event.pos[0],event.pos[1])
                    elif self.isPaused==True:
                        self.drawInputs=False
                        x=event.pos[0]
                        y=event.pos[1]
                        self.currObject=
self.mousePressed(x,y,self.currObject)
```

```
                        if self.currObject !=None:
                            if self.currObject not in self.entities:
                                self.entities.add(self.currObject)
                            else:
                                self.currObject.setPoints(x,y)

                #elif event.type == pygame.MOUSEBUTTONUP and
event.button == 1:
                # elif (event.type == pygame.MOUSEMOTION and
                #       event.buttons == (0, 0, 0)):
                #     self.mouseMotion(*(event.pos))

                elif (event.type == pygame.MOUSEBUTTONDOWN and
event.button==3 and
                self.isPaused==True and self.place != "main" and
self.highObject !=None):
                    self.checkDrawInputs(x,y)




                elif (event.type == pygame.MOUSEMOTION and
event.buttons[0] == 1 and self.isPaused==True and self.place !=
"main"):
                    if self.currObject !=None:
                        x=event.pos[0]
                        y=event.pos[1]
                        if not isinstance(self.currObject, polygon):

                            self.currObject.setSize(x,y)
                            self.DrawDim=True

#self.currObject.drawSettingSizeDim(screen)

                    elif self.highObject !=None:

self.highObject.move(event.pos[0],event.pos[1])


                elif event.type == pygame.KEYDOWN:
                    #self._keys[event.key] = True
                    self.keyPressed(event.key)#, event.mod)
                #elif event.type == pygame.KEYUP:
                    #self._keys[event.key] = False
                    #self.keyReleased(event.key, event.mod)


                if event.type == pygame.MOUSEBUTTONDOWN and
event.button==1 and self.currObject !=None and not
isinstance(self.currObject, polygon):
```

```
                    pass

                if event.type == pygame.QUIT:
                    playing = False



            pygame.display.flip()

        pygame.quit()
PygameGame().run()
```