Project details about the code

```
playerImg=pygame.image.load("GreenHeart.png")
playerX=330
playerY=230

playerShield=pygame.image.load("shieldDown.png")
playerShieldX=307
playerShieldY=260

def player(z,x,y):
    screen.blit(playerImg,(playerX,playerY))
    screen.blit(z,(x,y))
```

The player function is the main subject, the reason why z x y is used only for the second screen.blit is for the shield

Since the way on how the game work depends on the shield hitting the projectile, the shield looks and coorditates of the x and y axises will change and be dependant.

```
ArrowRight=pygame.image.load("ArrowRight.png")
ArrowRightX=700
ArrowRightY=230
ArrowDown=pygame.image.load("ArrowDown.png")
ArrowDownX=330
ArrowDownY=500
ArrowLeft=pygame.image.load("ArrowLeft.png")
ArrowLeftX=-10
ArrowLeftY=230
ArrowUp=pygame.image.load("Arrowup.png")
ArrowUpX=330
ArrowUpY=-10
#arrow key command
def Arrows(z,x,y):
    screen.blit(z,(x,y))
```

4 projectiles up down left and right with different pictures and the coordinates and also to show the file

```
#song Battle Against a true hero by Toby Fox
mixer.music.load("BackgroundBATH.wav")
mixer.music.play(-1)
```

Adding the audio

```
DirectionOfArrow=["up","down","left","right"]

def chooseArrow():
    global ArrowCount
    global state
    DirectionsChoose=random.randint(0,3)
    print(DirectionsChoose)
    ArrowCount=DirectionOfArrow[DirectionsChoose]
    if ArrowCount == "up":
```

```
        Arrows(ArrowUp,ArrowUpX,ArrowUpY)
        state="active"
    elif ArrowCount =="down":
        Arrows(ArrowDown,ArrowDownX,ArrowDownY)
        state="active"
    elif ArrowCount =="left":
        Arrows(ArrowLeft,ArrowLeftX,ArrowLeftY)
        state="active"
    elif ArrowCount =="right":
        Arrows(ArrowRight,ArrowRightX,ArrowRightY)
        state="active"
    else:
        pass
```

List used to select the random and select

The if command will then choose which arrows to be shot

```
def Blocked(ArrowType):
    if ArrowType=="right":
        if 350>ArrowRightX < 360 and playerShieldX==360:
            return True
        else:
            return False
    elif ArrowType=="left":
        if 290<ArrowLeftX< 300 and playerShieldX==300:
            return True
        else:
            return False
    elif ArrowType=="up":
        if 190<ArrowUpY <200 and playerShieldY==209:
            return True
        else:
            return False
    elif ArrowType=="down":
        if 250<ArrowDownY<260 and playerShieldY==260:
            return True
        else:
            return False
    else:
        return False
```

making the block function that is used to reset the arrow in random and under 2 conditions of range so that you do not miss the arrow or have a higher success change to hit the arrow.

<u>The screen running</u>

```
ArrowRightX -= 0.5
    ArrowDownY -= 0.5
    ArrowLeftX += 0.5
    ArrowUpY += 0.5
```

Moving the arrow

```
if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_UP:
                playerShield=pygame.image.load("shield.png")
                playerShieldX=307
                playerShieldY=209
                player(playerShield,playerShieldX,playerShieldY)

            if event.key == pygame.K_LEFT:
                playerShield=pygame.image.load("shieldLeft.png")
                playerShieldX=300
                playerShieldY=210
                player(playerShield,playerShieldX,playerShieldY)

            if event.key == pygame.K_RIGHT:
                playerShield=pygame.image.load("shieldRight.png")
                playerShieldX=360
                playerShieldY=210
                player(playerShield,playerShieldX,playerShieldY)

            if event.key == pygame.K_DOWN:
                playerShield=pygame.image.load("shieldDown.png")
                playerShieldX=307
                playerShieldY=260
                player(playerShield,playerShieldX,playerShieldY)
```

controls of the game when the key up, down, left, right is either pressed, the movement of the shield will differ

calling the command and precautions of the game

```
    player(playerShield,playerShieldX,playerShieldY)
    hit=Blocked(ArrowCount)
    #continue the code
    while state=="empty":
        chooseArrow()
    else :
        if ArrowCount=="up":
            Arrows(ArrowUp,ArrowUpX,ArrowUpY)
        elif ArrowCount=="down":
            Arrows(ArrowDown,ArrowDownX,ArrowDownY)
        elif ArrowCount=="right":
            Arrows(ArrowRight,ArrowRightX,ArrowRightY)
        elif ArrowCount=="left":
            Arrows(ArrowLeft,ArrowLeftX,ArrowLeftY)
```

The state change is used here as a precaution for the continuous loop of chooseArrow() if we were to call that function outside.

The way how I overcome the problem is by putting a state so that it only is called when the arrow is gone.

The arrow count variable is set so that the precaution of whether up down left or right arrow will continue to move

```
    if hit==True:
        score +=1
        ArrowUpX=330
        ArrowUpY=-10
        ArrowLeftX=-10
        ArrowLeftY=230
        ArrowDownX=330
        ArrowDownY=500
        ArrowRightX=700
        ArrowRightY=230
        state="empty"

        print(score)
    else:
        pass
```

The hit that stored the command Blocked() is used to determine if the return value is true or false

True means that the arrow has successfully been hit and the commands below which resets all the arrow x and y coordinates in addition to changing the state to empty which will call the new arrow to be made