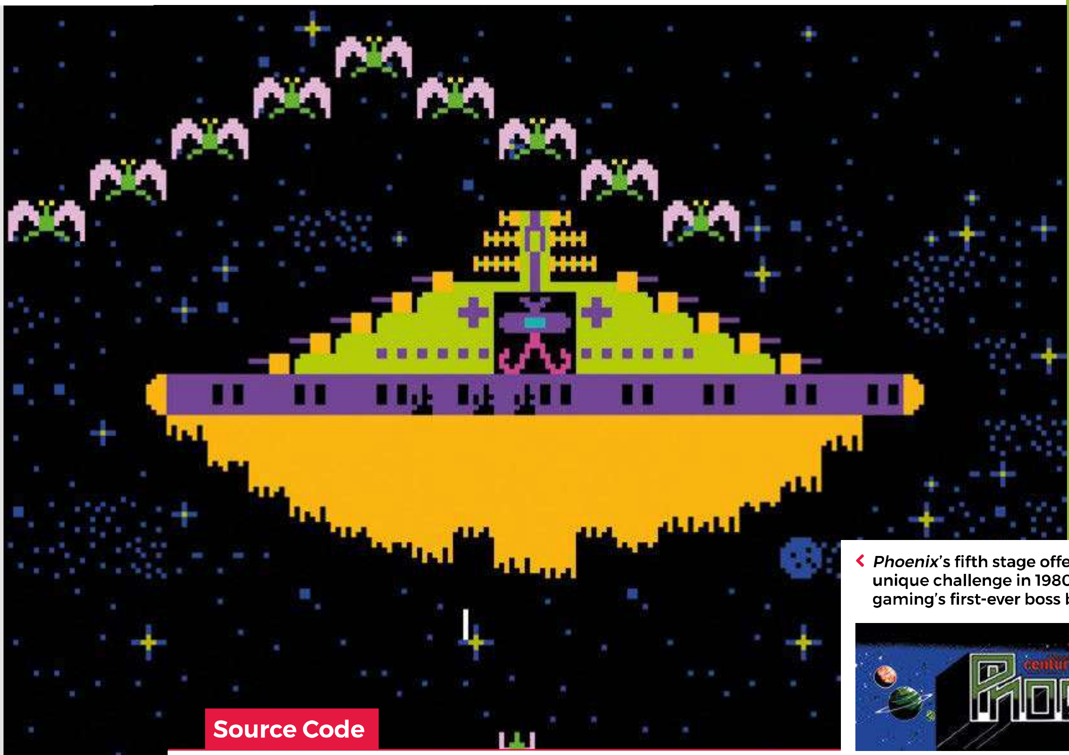


Source Code

Phoenix's fifth stage offered a unique challenge in 1980: one of gaming's first-ever boss battles.

Code a Phoenix-style mothership battle

It was one of gaming's first boss battles. Mark shows you how to recreate the mothership from 1980's Phoenix

AUTHOR
MARK VANSTONE

First released in 1980, *Phoenix* was something of an arcade pioneer. The game was the kind of post-*Space Invaders* fixed-screen shooter that was ubiquitous at the time: players moved their ship from side to side, shooting at a variety of alien birds of different sizes and attack patterns. The enemies moved swiftly, and the player's only defence was a temporary shield which could be activated when the birds swooped and strafed the lone defender. But besides all that, *Phoenix* had a few new ideas of its own: not only did it offer five distinct stages, but it also featured one of the earliest examples of a boss battle – its heavily armoured alien mothership, which required accurate shots to its shields before its weak spot could be exposed.

To recreate *Phoenix*'s boss, all we need is Pygame Zero. We can get a portrait style window with the **`WIDTH`** and **`HEIGHT`** variables and throw in some parallax stars (an improvement on the original's static backdrop) with some blitting in the **`draw()`** function. The parallax effect is created by having a static background of stars with a second (repeated) layer of stars moving down the screen.

"The mothership is made up of several Actor objects which move together"

The mothership itself is made up of several Actor objects which move together down the screen towards the player's spacecraft, which can be moved right and left using the mouse. There's the main body of the mothership, in the centre is the alien that we want to shoot, and then we have two sets of moving shields. In this example, rather than have all the graphics dimensions in multiples of eight (as we always did in the old days), we will make all our shield blocks 20 by 20 pixels, because computers simply don't need to work in multiples of eight any more. The first set of shields is the purple rotating bar around the middle of the ship. This is made up of 14 Actor blocks which shift one place to the right each time they move. Every other block has a couple of portal windows which makes the rotation obvious, and when a block moves off the right-hand side, it is placed on the far left of the bar.

The second set of shields are in three yellow rows (you may want to add more), the first with 14 blocks, the second with ten blocks, and the last with four. These shield blocks are fixed in place but share a behaviour with the purple bar shields, in that when they are hit by a bullet, they change to a damaged version.

Phoenix in Python

Here's Mark's code snippet, which recreates that pioneering boss battle in Python. To get it running on your system, you'll first need to install Pygame Zero – you can find full instructions at **wfmag.cc/pgzero**.

Download the code from GitHub: **wfmag.cc/wfmag26**

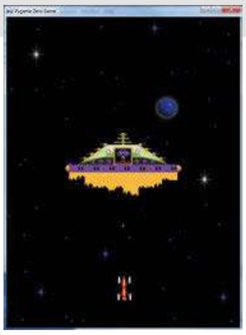
```
WIDTH = 600
HEIGHT = 800

mothership = Actor(‘mothership’, center=(300, 100))
bullet = Actor(‘bullet’, center=(0, -10))
alien = Actor(‘aliendude’, center=(300, 110))
ship = Actor(‘ship’, center=(300, 700))
barShield = []
lowerShield = []
backY = count = mothership.frame = gameover = 0
for b in range(0, 14):
    barShield.append(Actor(‘bar1’+str(b%2),
center=(310+((b-7)*20), 140)))
    lowerShield.append(Actor(‘shield1’,
center=(310+((b-7)*20), 160)))
    barShield[b].frame = lowerShield[b].frame = 1
for b in range(0, 10):
    lowerShield.append(Actor(‘shield1’,
center=(310+((b-5)*20), 180)))
    lowerShield[b + 14].frame = 1
for b in range(0, 4):
    lowerShield.append(Actor(‘shield1’,
center=(310+((b-2)*20), 200)))
    lowerShield[b + 24].frame = 1

def draw():
    screen.blit(“background”, (0, 0))
    screen.blit(“stars”, (0, backY))
    screen.blit(“stars”, (0, backY-800))
    mothership.draw()
    if gameover != 1 or (gameover == 1 and count%2 == 0):
alien.draw()
    for b in range(0, 28):
        if b < 14:
            if barShield[b].frame < 5:
                barShield[b].draw()
        if lowerShield[b].frame < 5:
            lowerShield[b].draw()
    bullet.draw()
    if gameover != 2 or (gameover == 2 and count%2 == 0):
ship.draw()
```

```

def update():
    global backY, count, gameover
    count += 1
    if gameover == False:
        backY += 0.2
        if backY > 800: backY = 0
        mothership.y += 0.1
        mothership.frame = int(count/10)%14
        alien.y = mothership.y + 10
        for b in range(0, 28):
            if b < 14:
                x = (((mothership.frame+b)-7)*20)
                if x >= 140: x -= 280
                barShield[b].y += 0.1
                barShield[b].x = (mothership.x+10)+ x
                if barShield[b].frame < 5 and barShield[b].
colliderect(bullet):
                    barShield[b].frame += 1
                    if barShield[b].frame < 5:
                        barShield[b].image =
“bar”+str(barShield[b].frame)
                    bullet.y = -10
            lowerShield[b].y += 0.1
            if lowerShield[b].frame < 5 and lowerShield[b].
colliderect(bullet):
                lowerShield[b].frame += 1
                if lowerShield[b].frame < 5:
                    lowerShield[b].image =
“shield”+str(lowerShield[b].frame)
                bullet.y = -10
        if alien.colliderect(bullet): gameover = 1
        if ship.colliderect(mothership): gameover = 2
        if bullet.y > -10: bullet.y -= 5

def on_mouse_down(pos):
    if bullet.y < 0: bullet.pos = (ship.x,700)

def on_mouse_move(pos):
  ^ship.x = pos[0]
```

There are four levels of damage before they are destroyed and the bullets can pass through. When enough shields have been destroyed for a bullet to reach the alien, the mothership is destroyed (in this version, the alien flashes).

Bullets can be fired by clicking the mouse button. Again, the original game had alien birds flying around the mothership and dive-bombing the player, making it harder to get a good shot in, but this is something you could try adding to the code yourself.

To really bring home that eighties *Phoenix* arcade experience, you could also add in some atmospheric shooting effects and, to round the whole thing off, have an 8-bit rendition of Beethoven's *Für Elise* playing in the background.


Like the original *Phoenix*, our mothership boss battle has multiple shields that need to be taken out to expose the alien at the core.