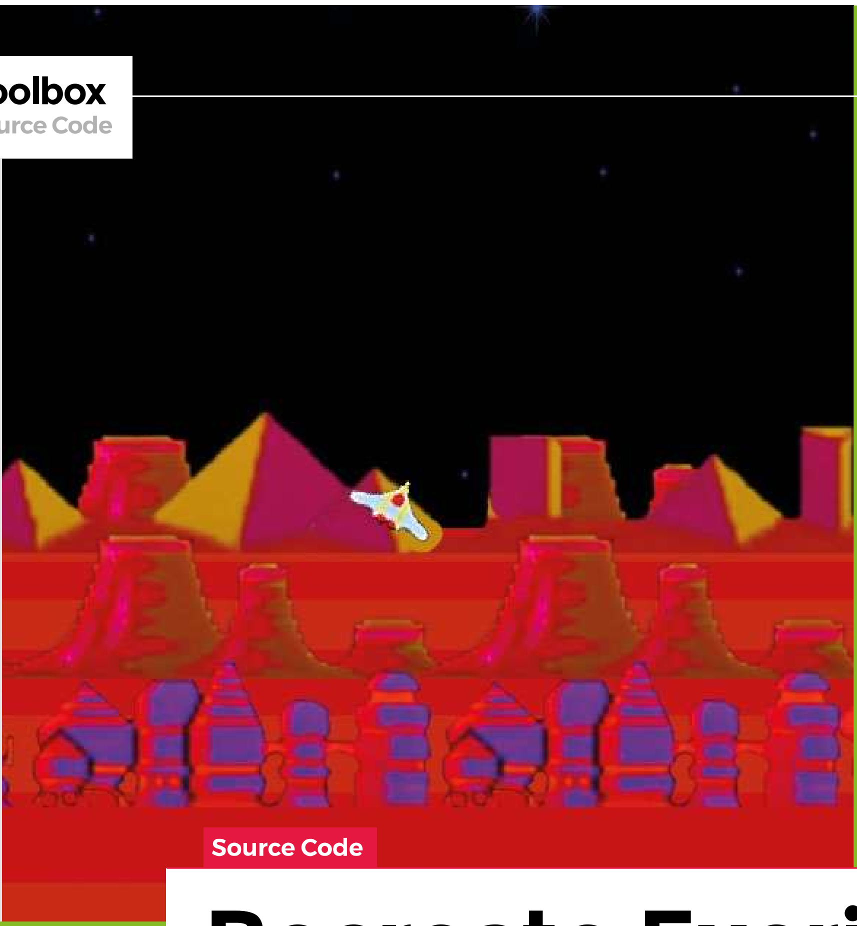


^ *Exerion*'s arcade flyer was careful to push the game's visual splendour.

< Our homage to *Exerion*. You can't tell from a static image, but the illusion of depth is amazing. Honest.

Source Code

Recreate Exerion's pseudo-3D landscape

AUTHOR
MARK VANSTONE

Swoop over mountains in our homage to Jaleco's shooter

Taking the shooting action of *Galaxian* from a few years earlier, Japanese developer Jaleco released *Exerion* in 1983. What helped *Exerion* stand out from other shoot-'em-ups of the period, though, was its pseudo-3D background, which used both a scrolling landscape and moving background elements to create the illusion of depth. This was quite an achievement considering the hardware of the day, and it's still an eye-catching effect even now.

To recreate *Exerion*'s scrolling in Pygame Zero, we need to break the effect down into three main elements. The first is the scrolling stripes that form the landscape's base layer. These are followed by the elements that roll over the landscape as it scrolls down the screen. Then thirdly, there's the player's movement, which affects both the other two elements. Let's start with the scrolling landscape, which is made of alternating coloured stripes. To give the sense of perspective, they start very thin on the horizon and, as they move down the screen, they grow in thickness. We can create this with a list that contains the heights of each stripe, increasing as we go through the list. Then in our `draw()` function, we run through the list, drawing the stripes downwards from the horizon using the heights in our list. Then we increase the height of each stripe. When the first stripe reaches a certain height, we take the last one off the end of the list and add it to the beginning, resetting its height to the smallest.

The next items to code are the landscape details. These are buildings and hills that we want to move with the stripes so that it looks as though the player's flying over them as they scroll by. We need to do this in two sections as some will be drawn behind the stripes as they're over the horizon, while others will be in front of the stripes. We'll give each landscape item an index which ties it to a stripe, but we'll give items that are beyond the horizon negative indexes, and those in front, positive. All the landscape items will start with a negative index to indicate that they all start beyond the horizon. So in the `draw()` function, we have an initial loop to draw all the items behind the horizon, and then while we're drawing the stripes, we also draw the items which have the same index as the stripes, so they appear in front. Once we have these two parts, we'll have a continuous carousel of stripes and landscape items.

Now we need the player aircraft. We can move it around using the arrow keys, but we want to have the background graphics moving to give the impression of a 3D landscape: if the player moves upwards, we move the horizon down, and do the opposite if the player moves downwards. We then apply a parallax effect to the landscape items. The way we do this is by moving the items at the back a small amount in the opposite direction from the player's movement, and as we work down through the items, they move more and more. This enhances the impression of depth.

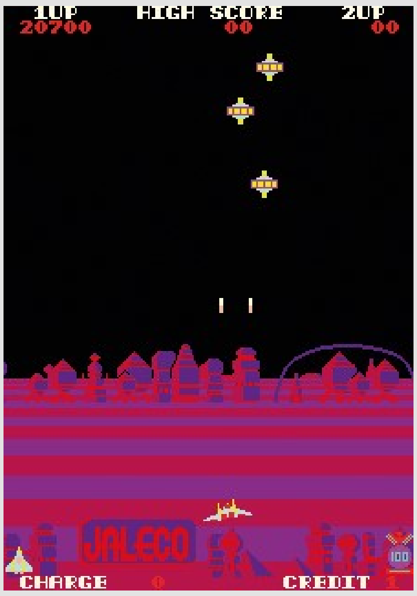
Once we've added a tilt to the aircraft as it turns, we have the makings of an *Exerion* clone. All that needs to be added are the aliens to shoot at – if you want to add these, then you could take the *Galaxian* routine from last month's Source Code.

An Exerion landscape in Python

Here's Mark's code, which will create a neat pseudo-3D effect in Python. To get it running on your system, you'll need ot install Pygame Zero – full instructions are available at **wfmag.cc/pgzero**.

Download the code from GitHub: **wfmag.cc/ wfmag51**

```
WIDTH = 600
HEIGHT = 800
ship = Actor(‘ship’, center=(300, 700))
count = 0
startcol = 0
stripes = []
for s in range(0, 20):
    stripes.append((s+1)*4)
landscape = []
landitems = [1,2,3,3,2,1,2,3,2,3,1]
landindexes = [-5,-8,-10,-13,-14,-20,-22,-26,-28,-30,-31]
for l in range(0, 10):
    landscape.append(Actor(‘landscape’+str(landitems[l]),
center=(300,1000)))
    landscape[l].index = landindexes[l]
    landscape[l].yoff = 0

def draw():
    drawLand()
    screen.draw.text(“EXERION ZERO”, center = (300, 60),
owidth=0.5, ocolor=(255,255,0), color=(255,0,0) , fontsize=80)
    ship.draw()

def update():
    global count,startcol
    count += 1
    if keyboard.left and ship.x > 100:
        if ship.angle < 30: ship.angle +=2
        ship.x -= ship.angle/6
    if keyboard.right and ship.x < 500:
        if ship.angle > -30: ship.angle -=2
        ship.x -= ship.angle/6
    if keyboard.up and ship.y > 400 : ship.y -= 4
    if keyboard.down and ship.y < 750 : ship.y += 4
    if not keyboard.left and not keyboard.right:
        if ship.angle > 0: ship.angle -= 2
        if ship.angle < 0: ship.angle += 2
    for s in range(0, 20):
        stripes[s] += 0.2
    if stripes[0] > 10:
        stripes.insert(0, stripes.pop())
        stripes[0] = 1
        if startcol == 0:
            startcol = 40
        else:
            startcol = 0
        updateLandscape()

def drawLand():
    sh = (800-ship.y)/2
    screen.blit(“background”, (0, sh/2))
    y = 300 + sh
    col = startcol
    for l in range(0, 10):
        if landscape[l].index < 0:
            landscape[l].x = parallax(y)
            landscape[l].y = y - landscape[l].yoff
-(landscape[l].index*23)-30
            landscape[l].yoff += 0.5
            landscape[l].draw()
    for s in range(0, 20):
        col = col + 40
        if col > 40: col = 0
        screen.draw.filled_rect(Rect((0, y), (600,
stripes[s])),(200,col,0))
        for l in range(0, 10):
            if landscape[l].index == s:
                landscape[l].y = y-stripes[s]-30
                landscape[l].x = parallax(y)
                landscape[l].draw()
        y += stripes[s]-1

def updateLandscape():
    for l in range(0, 10):
        landscape[l].index += 1
        landscape[l].yoff = 0
        if landscape[l].index > 20:
            landscape[l].index = -10

def parallax(y):
    sh = (800-ship.y)/2
    return ((300-ship.x) * ((y-sh)/500))+300
```



Exerion's pseudo-3D effect helped the game stand out from the crowd of other shooters packed into arcades at the time.