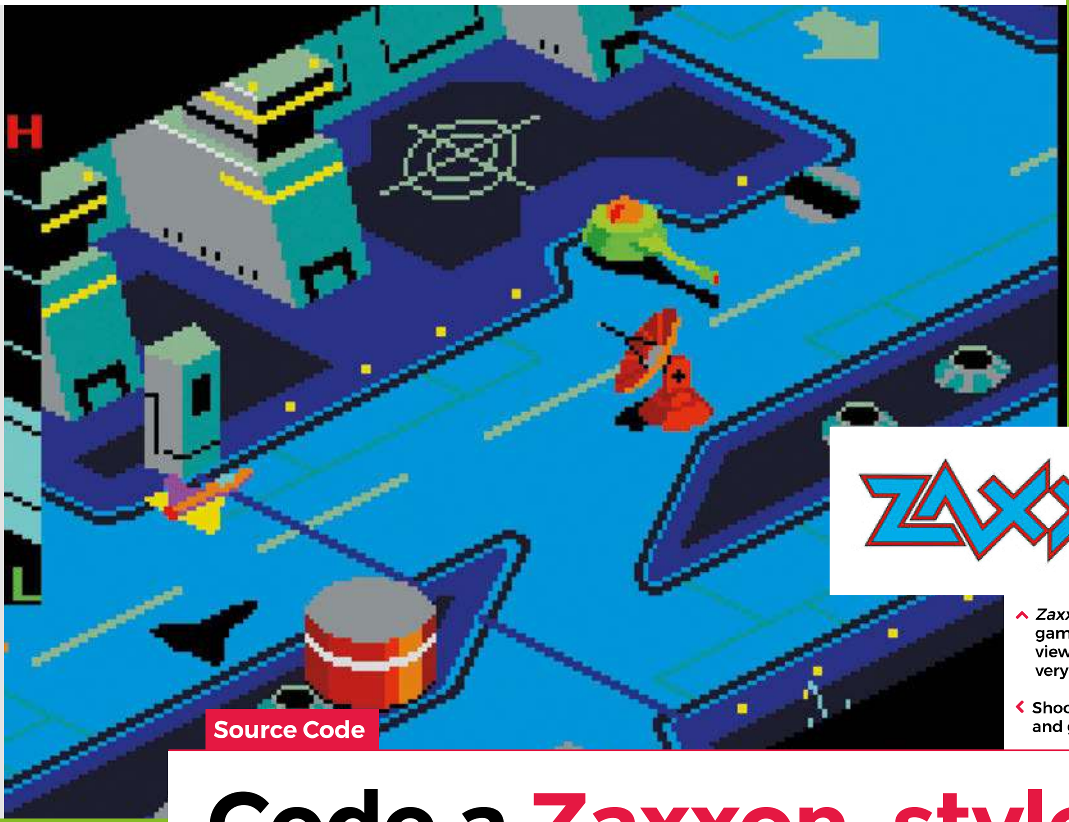


^ *Zaxxon* was the first arcade game to use an axonometric viewpoint, which made it look very different from its 2D rivals.

< Shoot fuel silos to score points and gain extra flying time.

Source Code

Code a Zaxxon-style axonometric level

AUTHOR
MARK VANSTONE

Fly through the space fortress in this 3D retro forced scrolling arcade sample

When *Zaxxon* was first released by Sega in 1982, it was hailed as a breakthrough thanks to its isometric, pseudo-3D graphics. This axonometric projection ensured that *Zaxxon* looked unlike any other shooter around in arcades.

Graphics aside, *Zaxxon* offered a subtly different twist on other shooting games of the time, like *Defender* and *Scramble*; the player flew over either open space or a huge fortress, where they had to avoid obstacles of varying heights. Players could tell how high they were flying with the aid of an altimeter, and also the shadow beneath their ship (shadows were another of *Zaxxon*'s innovations). The aim of the game was to get to the end of each level without running out of fuel or getting shot down; if the player did this, they'd encounter an area boss called Zaxxon. Points were awarded for destroying gun turrets and fuel silos, and extra lives could be gained as the player progressed through the levels.

For this code sample, we can borrow some of the techniques used in a previous Source Code article about *Ant Attack* (see Wireframe issue 15) since it also used an isometric display. Although the way the map display is built up is very similar, we'll use a JSON file to store the map data. If you've not come across JSON before, it's well worth learning about, as a number of web and mobile apps use it, and it can be read by Python very easily. All we need to do is load the JSON file, and Python automatically puts the data into a Python dictionary object for us to use.

In the sample, there's a short run of map data 40 squares long with blocks for the floor, some low walls, higher walls, and a handful of fuel silos. To add more block types, just add data to the `blocktypes` area of the JSON file. The codes used in the map data are the index numbers of the `blocktypes`, so the first `blocktypes` is `index 0`, the next `index 1`, and so on. Our `drawMap()` function takes care of rendering the data into visual form and blits blocks from the top right to the bottom left of the screen. When the `draw` loop gets to where the ship is, it draws first the shadow and then the ship a little higher up the screen, depending on the altitude of the ship. The equation to translate the ship's screen coordinates to a block position on the map is a bit simplistic, but in this case, it does the job well enough.

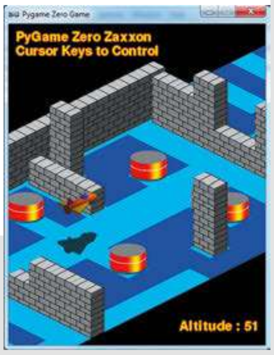
Cursor keys guide the movement of the spaceship, which is limited by the width of the map and a height of 85 pixels. There's some extra code to display the ship if it isn't on the map – for example, at the start, before it reaches the map area. To make the code snippet into a true *Zaxxon* clone, you'll have to add some laser fire and explosions, a fuel gauge, and a scoring system, but this code sample should provide the basis you'll need to get started.

A Zaxxon scrolling map in Python

Here's Mark's code snippet, which creates an isometric scrolling map in Python. To get it running on your system, you'll need to install Pygame Zero – you can find full instructions at **wfmag.cc/pgzero**.

Download the code from GitHub: **wfmag.cc/ wfmag33**

```
import json

WIDTH = 400
HEIGHT = 500

gameState = count = shipHeight = 0

with open(‘mapdata.json’) as json_file:
    mapData = json.load(json_file)
    mapBlocks = mapData[‘blocks’]
    mapBlockTypes = mapData[‘blocktypes’]
    mapWidth = mapData[‘width’]
    mapLength = mapData[‘length’]

mapPosX = 200 + (mapLength*32)
mapPosY = 150 - (mapLength*16)
shipPos = [50,300]

def draw():
    screen.fill((0,0,0))
    drawMap()
    screen.draw.text(“PyGame Zero Zaxxon \nCursor Keys
to Control”, (10, 10), owidth=0.5, ocolor=(255,0,0),
color=(255,255,0) , fontsize=30)
    screen.draw.text(“Altitude : “+ str(shipHeight),
topright=(390, 460), owidth=0.5, ocolor=(255,0,0),
color=(255,255,0) , fontsize=30)

def drawMap():
    global gameState
    shipBlock = getShipXY()
    shipFrame = “0”
    if keyboard.left: shipFrame = “-1”
    if keyboard.right: shipFrame = “1”
    for x in range(0, mapWidth):
        for y in range(0, mapLength):
            bx = (x*32) - (y*32) + mapPosX
            by = (y*16) + (x*16) + mapPosY
            if -64 <= bx < WIDTH + 32 and -64 <= by < HEIGHT
+ 64:
                if mapBlocks[x][y] > 0:
                    if shipBlock == [x,y]:
                        if
mapBlockTypes[mapBlocks[shipBlock[0]][shipBlock[1]]][‘height’]
> shipHeight+32 : gameState = 1
                    screen.blit(mapBlockTypes[mapBlocks[x][y]]
[‘image’], (bx, by-mapBlockTypes[mapBlocks[x][y]][‘height’]))
                if shipBlock == [x-1,y-1]:
                    if(gameState == 0 or count%4 == 0):
                        screen.
blit(“shadow”+shipFrame,(shipPos[0],shipPos[1]+10))
                        screen.blit(“ship”+shipFrame,(shipPos[
0],shipPos[1]-shipHeight))

    if shipBlock[1] >= mapLength-1 or shipBlock[1] < 0 or
shipBlock[0] == mapWidth-1:
        screen.
blit(“shadow”+shipFrame,(shipPos[0],shipPos[1]+10))
        screen.blit(“ship”+shipFrame,(shipPos[0],shipPos[1]-
shipHeight))

def update():
    global count, gameState, mapPosX, mapPosY, shipHeight
    if gameState == 0:
        mapPosX -=1
        mapPosY +=0.5
        shipBlock = getShipXY()
        if keyboard.left:
            if shipBlock[0] > 0:
                shipPos[0] -=1
                shipPos[1] -=0.5
        if keyboard.right:
            if shipBlock[0] < mapWidth-1:
                shipPos[0] +=1
                shipPos[1] +=0.5
        if keyboard.up: shipHeight = max(min(85,
shipHeight+1), 0)
        if keyboard.down: shipHeight = max(min(85,
shipHeight-1), 0)
    count += 1

def getShipXY():
    x = ((shipPos[0]+82)/32)
    y = mapLength -
((shipPos[1]/16) + (mapPosY/16) +
((mapWidth/2)-x))-2
    return [int(x),int(y)]
```



PyGame Zero Zaxxon
Cursor Keys to Control

Altitude : 51

> Our *Zaxxon* homage running in Pygame Zero: fly the spaceship through the fortress walls and obstacles with your cursor keys.

HIGHS AND LOWS

In the original game, phase one obstacles were mostly on ground level, which meant that the player had the option to avoid these dangers by flying at a high altitude, but to score points and earn extra fuel (which would be necessary later in the level), the player has to keep lower to the ground. If the player decided to stay high for too long, a heat-seeking missile came in to destroy the player's ship. Later levels included attacking enemy spaceships, which brought a new level of difficulty as the targets were moving and varied their altitude.