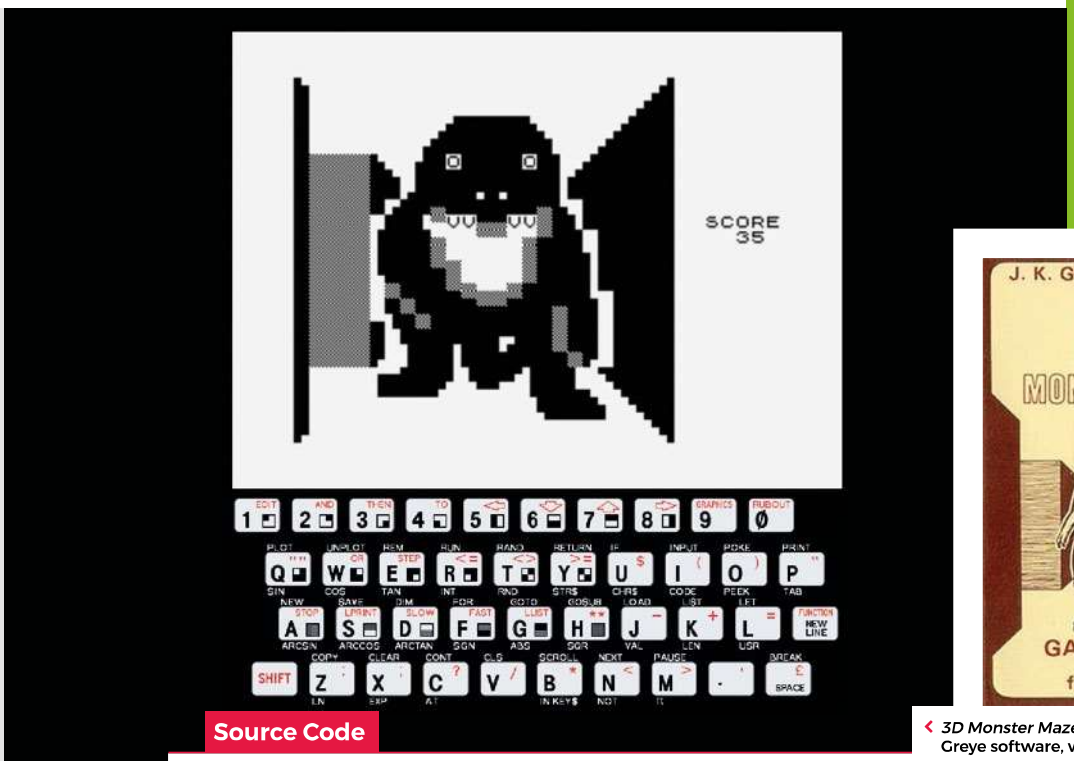
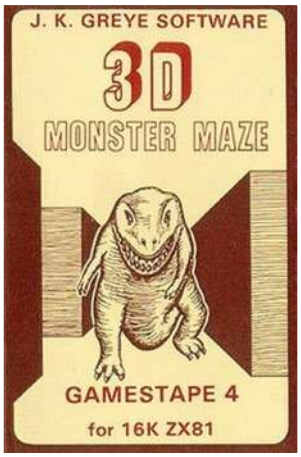




3D Monster Maze, released in 1982 by J.K. Greye software, written by Malcolm Evans.

Source Code

Recreate 3D Monster Maze's 8-bit labyrinth

AUTHOR
MARK VANSTONE

You too can recreate the techniques behind a pioneering 3D maze game in Python

While 3D games have become more and more realistic, some may forget that 3D games on home computers started in the mists of time on machines like the Sinclair ZX81. One such pioneering game took pride of place in my collection of tapes, took many minutes to load, and required the 16K RAM pack expansion. That game was *3D Monster Maze* – perhaps the most popular game released for the ZX81.

The game was released in 1982 by J.K. Greye Software, and written by Malcolm Evans. Although the graphics were incredibly low resolution by today's standards, it became an instant hit. The idea of the game was to navigate around a randomly generated maze in search of the exit. The problem was that a Tyrannosaurus rex also inhabited the maze, and would chase you down and have you for dinner if you didn't escape quickly enough. The maze itself was made of straight corridors on a 16×18 grid, which the player would move around from one block to the next. The shape of the blocks were displayed by using the low-resolution pixels included in the ZX81's character set, with 2×2 pixels per character on the screen.

"The maze itself was made of straight corridors on a 16×18 grid"

There's an interesting trick to recreating the original game's 3D corridor display which, although quite limited, works well for a simplistic rendering of a maze. To do this, we need to draw imaginary lines diagonally from corner to corner in a square viewport: these are our vanishing point perspective guides. Then each corridor block in our view is half the width and half the height of the block nearer to us. If we draw this out with lines showing the block positions, we get a view that looks like we're looking down a long corridor with branches leading off left and right. In our Pygame Zero version of the maze, we're going to use this wireframe as the basis for drawing our block elements. We'll create graphics for blocks that are near the player, one block away, two, three, and four blocks away. We'll need to view the blocks from the left-hand side, the right-hand side, and the centre.

Once we've created our block graphics, we'll need to make some data to represent the layout of the maze. In this example, the maze is built from a 10×10 list of zeros and ones. We'll set a starting position for the player and the direction they're facing (0–3), then we're all set to render a view of the maze from our player's perspective.

The display is created from furthest away to nearest, so we look four blocks away from the player (in the direction they're looking) and draw a block if there's one indicated by

A simple 3D maze in Python

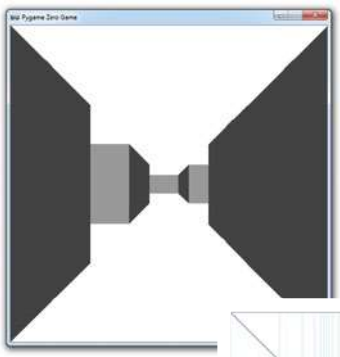
Here's Mark's code, which recreates *3D Monster Maze*'s network of corridors in Python. To get it running on your system, you'll need to install Pygame Zero – you can find instructions at **wfmag.cc/pgzero**

Download the code from GitHub: **wfmag.cc/wfmag18**

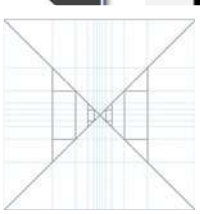
```
WIDTH = 600
HEIGHT = 600
maze = [[1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
        [1, 1, 0, 1, 0, 1, 0, 0, 0, 1],
        [1, 0, 0, 0, 0, 1, 0, 1, 0, 1],
        [1, 1, 0, 1, 0, 0, 0, 0, 0, 1],
        [1, 1, 0, 1, 0, 1, 0, 1, 0, 1],
        [1, 0, 0, 0, 0, 0, 0, 0, 0, 1],
        [1, 1, 0, 1, 0, 1, 1, 0, 1, 1],
        [1, 1, 0, 1, 0, 1, 1, 0, 1, 1],
        [1, 1, 0, 0, 0, 0, 0, 0, 0, 1],
        [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]]
playerX = 1
playerY = 4
playerDir = 2
dirX = [-1,0,1,0]
dirY = [0,1,0,-1]

def draw():
    screen.fill((255, 255, 255))
    screen.blit("back", (0, 0))
    drawMaze()

def update():
    pass

def on_key_down(key):
    global playerX, playerY, playerDir
    if key.name == "UP":
        newX = playerX + dirX[playerDir]
        newY = playerY + dirY[playerDir]
        if maze[newX][newY] == 0:
            playerX = newX
            playerY = newY
    if key.name == "DOWN":
        newX = playerX - dirX[playerDir]
        newY = playerY - dirY[playerDir]
        if maze[newX][newY] == 0:
            playerX = newX
            playerY = newY
    if key.name == "LEFT":
        playerDir -=1
        if playerDir < 0: playerDir = 3
    if key.name == "RIGHT":
        playerDir +=1
        if playerDir > 3: playerDir = 0

def drawMaze():
    dm = 1
    if(playerDir == 1 or playerDir == 3): dm=-1
    for l in range(4,-1,-1):
        x = playerX + (l*dirX[playerDir])
        y = playerY + (l*dirY[playerDir])
        if(x>=0 and x<10 and y>=0 and y<10):
            xl = x + (dirY[playerDir] * dm)
            yl = y + (dirX[playerDir] * dm)
            if(maze[xl][yl] == 1):
                screen.blit("left"+str(l), (0, 0))
            xr = x - (dirY[playerDir] * dm)
            yr = y - (dirX[playerDir] * dm)
            if(maze[xr][yr] == 1):
                screen.blit("right"+str(l), (0, 0))
            if(maze[x][y] == 1):
                screen.blit("mid"+str(l), (0, 0))
```

the maze data to the left; we do the same on the right, and finally in the middle. Then we move towards the player by a block and repeat the process (with larger graphics) until we get to the block the player is on.

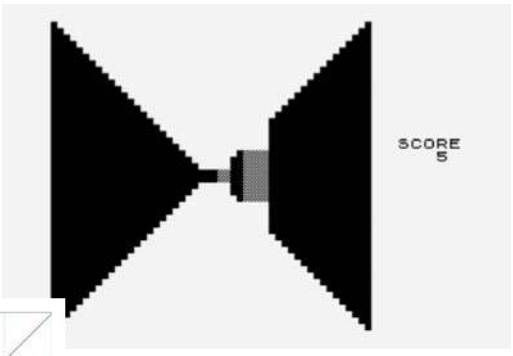
That's all there is to it. To move backwards and forwards, just change the position in the grid the player's standing on and redraw the display. To turn, change the direction the player's looking and redraw. This technique's obviously a little limited, and will only work with corridors viewed at 90-degree angles, but it launched a whole genre of games on home computers. It really was a big deal for many twelve-year-olds – as I was at the time – and laid the path for the vibrant, fast-moving 3D games we enjoy today.



Each visible block is drawn from the back forward to make the player's view of the corridors.



The maze display is made by drawing diagonal lines to a central vanishing point.



The original ZX81 game drew its maze from chunky 2×2 pixel blocks.