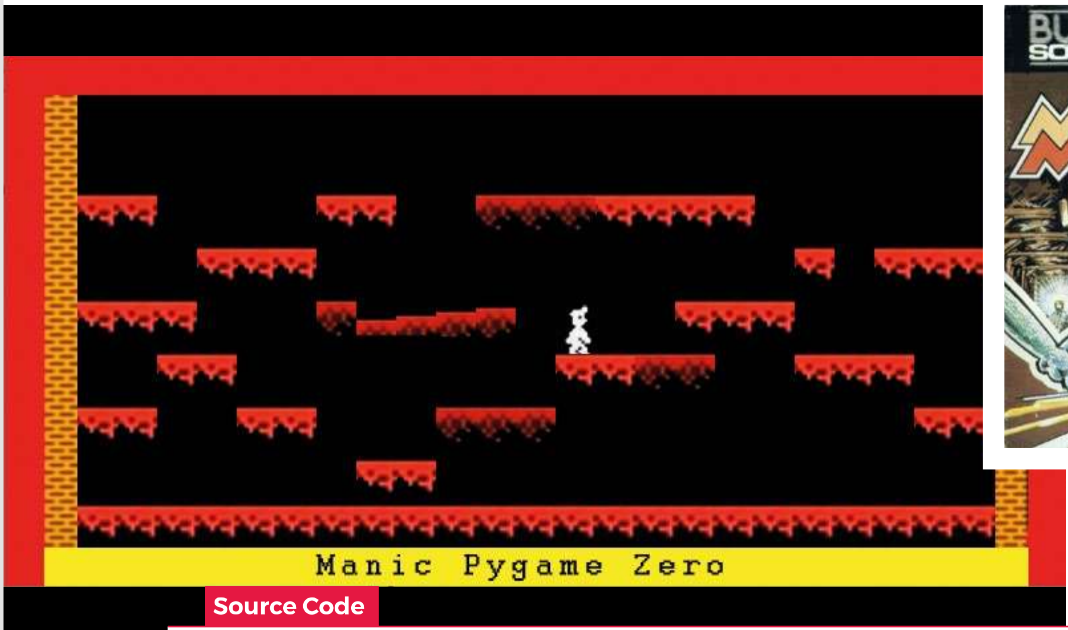
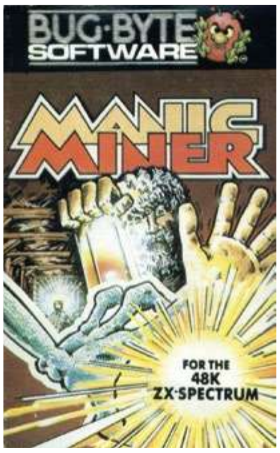




*Manic Miner's* cover art was wonderfully lo-fi.

Our homage to the classic *Manic Miner.*

Source Code

# Remake Manic Miner's collapsing platforms

AUTHOR
MARK VANSTONE

Traverse a crumbly cavern in our homage to a Spectrum classic

One of the most iconic games on the Sinclair ZX Spectrum featured a little man called Miner Willy, who spent his days walking and jumping from platform to platform collecting the items needed to unlock the door on each screen. *Manic Miner*'s underground world featured caverns, processing plants, killer telephones, and even a forest featuring little critters that looked suspiciously like Ewoks.

Written by programmer Matthew Smith and released by Bug-Byte in 1983, the game became one of the most successful titles on the Spectrum. Smith was only 16 when he wrote *Manic Miner* and even constructed his own hardware to speed up the development process, assembling the code on a TRS-80 and then downloading it to the Spectrum with his own hand-built interface. The success of *Manic Miner* was then closely followed by *Jet Set Willy*, featuring the same character, and although they were originally written for the Spectrum, the games very soon made it onto just about every home computer of the time.

Both *Manic Miner* and *Jet Set Willy* featured unstable platforms which crumbled in Willy's wake, and it's these we're going to try to recreate this month.

In this Pygame Zero example, we need three frames of animation for each of the two directions of movement. As we press the arrow keys we can move the Actor left and right, and in this case, we'll decide which frame to display based on a `count` variable, which is incremented each time our `update()` function runs. We can create platforms from a two-dimensional data list representing positions on the screen with 0 meaning a blank space, 1 being a solid platform, and 2 a collapsible platform. To set these up, we run through the list and make Actor objects for each platform segment.

For our `draw()` function, we can blit a background graphic, then Miner Willy, and then our platform blocks. During our `update()` function, apart from checking key presses, we also need to do some gravity calculations. This will mean that if Willy isn't standing on a platform or jumping, he'll start to fall towards the bottom of the screen. Instead of checking to see if Willy has collided with the whole platform, we only check to see if his feet are in contact with the top. This means he can jump up through the platforms but will then land on the top and stop. We set a variable to indicate that Willy's standing on the ground so that when the **SPACE** bar is pressed, we know if he can jump or not. While we're checking if Willy's on a platform, we also check to see if it's a collapsible one, and if so, we start a timer so that the platform moves downwards and eventually disappears. Once it's gone, Willy will fall through. The reason we have a delayed timer rather than just starting the platform heading straight down is so that Willy can run across many tiles before they collapse, but his way back will quickly disappear. The disappearing platforms are achieved by changing the image of the platform block as it moves downward.

As we've seen, there were several other elements to each *Manic Miner* screen, such as roaming bears that definitely weren't from *Star Wars*, and those dastardly killer telephones. We'll leave you to add those...

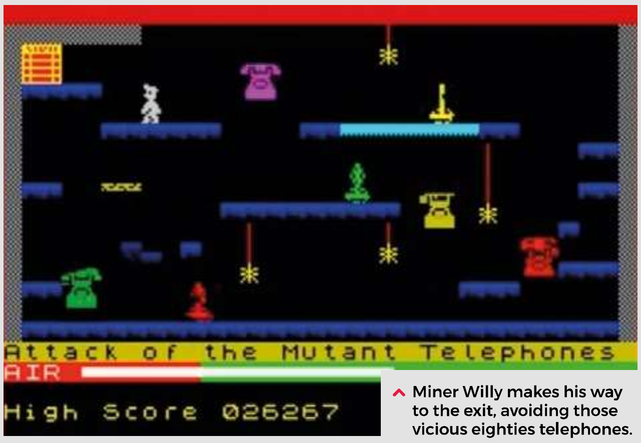
# Crumbly platforms in Python

Here's Mark's code for a *Manic Miner* screen, complete with collapsing platforms. To get it working on your system, you'll need to install Pygame Zero – full instructions are available at **wfmag.cc/pgzero**.

Download the code from GitHub: **wfmag.cc/wfmag49**

```
# Manic Miner

HEIGHT = 400
willy = Actor('willyr0',(400,300))
willy.direction = "r"
willy.jump = 0
willy.onground = False
count = 0
platforms = [[1,1,0,0,0,0,1,1,0,0,2,2,2,1,1,1,1,0,0,0,0,0,0],
            [0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,1],
            [1,1,1,0,0,0,2,2,2,2,2,0,0,0,0,1,1,1,0,0,0,0,0],
            [0,0,1,1,0,0,0,0,0,0,0,0,1,1,2,2,0,0,1,1,1,0,0],
            [1,1,0,0,1,1,0,0,0,2,2,2,0,0,0,0,0,0,0,0,0,1,1],
            [0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0]]
platformActors = []
for r in range(len(platforms)):
    for c in range(len(platforms[r])):
        if(platforms[r][c] != 0 ): platformActors.
append(Actor('platform'+str(platforms[r][c])+"1",(70+(c*30),1
20+(r*40))))
        platformActors[len(platformActors)-1].status = 0

def draw():
    screen.blit("background", (0, 0))
    willy.draw()
    drawPlatforms()

def update():
    global count
    willy.image = "willy"+ willy.direction + "0"
    if keyboard.left:
        moveWilly(-1,0)
        willy.direction = "l"
        willy.image = "willyl"+ str(int(count/8)%3)
        pass
    if keyboard.right:
        moveWilly(1,0)
        willy.direction = "r"
        willy.image = "willyr"+ str(int(count/8)%3)
        pass
    checkGravity()
    count += 1

def on_key_down(key):
    if key.name == "SPACE":
        if willy.onground == True:
            willy.jump = 40

def drawPlatforms():
    for p in range(len(platformActors)):
        if platformActors[p].status != -1:
            platformActors[p].draw()
```

```
def moveWilly(x,y):
    if willy.x+x < 730 and willy.x+x > 70:
        willy.x += x

def checkGravity():
    if willy.jump > 0:
        willy.y -=2
        willy.jump -=1
    if willy.y < 320:
        willy.onground = False
        for p in range(len(platformActors)):
            frame = int(platformActors[p].image[-1])+1
            if platformActors[p].status > 0 :
                platformActors[p].status -= 1
                if platformActors[p].status == 0 :
                    platformActors[p].y += 1
                    if frame > 8 :
                        platformActors[p].status = -1
                    else:
                        platformActors[p].image =
"platform2"+str(frame)
                        platformActors[p].status = 30
            if((willy.x > platformActors[p].x-20 and willy.x <
platformActors[p].x+20) and willy.y+20 == platformActors[p].y-
14+(frame-1) and platformActors[p].status != -1):
                willy.onground = True
                if platformActors[p].image[8] == "2":
                    if platformActors[p].status == 0 :
platformActors[p].status = 30
        if willy.onground == False:
            willy.y += 1
    else:
        willy.onground = True
```



Miner Willy makes his way to the exit, avoiding those vicious eighties telephones.