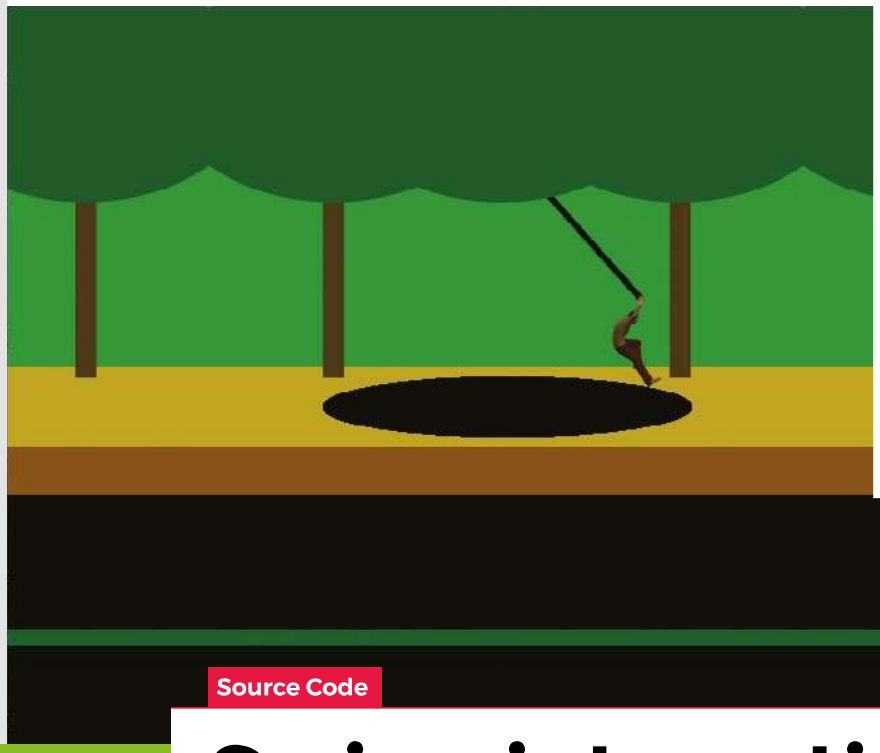
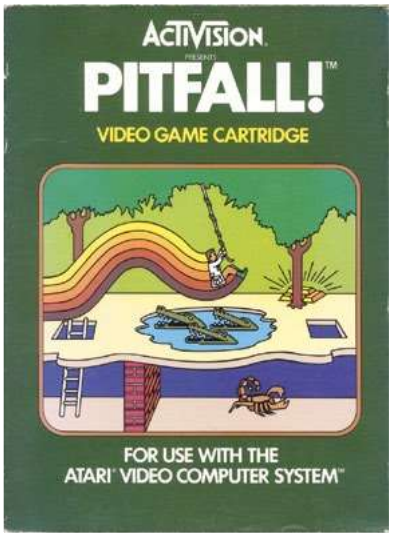




Designed by David Crane, *Pitfall!* was released for the Atari 2600 and published by Activision in 1982.

Our homage to the classic *Pitfall!* Atari game. Can you add some rolling logs and other hazards?

Source Code

Swing into action with an homage to Pitfall!

Grab onto ropes and swing across chasms in our Python rendition of an Atari 2600 classic

AUTHOR
MARK VANSTONE

Whether it was because of the design brilliance of the game itself or because *Raiders of the Lost Ark* had just hit the box office, Pitfall Harry became a popular character on the Atari 2600 in 1982. His hazardous attempts to collect treasure struck a chord with eighties gamers, and saw *Pitfall!*, released by Activision, sell over four million copies. A sequel, *Pitfall II: The Lost Caverns* quickly followed the next year, and the game was ported to several other systems, even making its way to smartphones and tablets in the 21st century.

The game itself is a quest to find 32 items of treasure within a 20-minute time limit. There are a variety of hazards for Pitfall Harry to navigate around and over, including rolling logs, animals, and holes in the ground. Some of these holes can be jumped over, but some are too wide and have a convenient rope swinging from a tree to aid our explorer in getting to the other side of the screen. Harry must jump towards the rope as it moves towards him and then hang on as it swings him over the pit, releasing his grip at the other end to land safely back on firm ground.

For this code sample, we'll concentrate on the rope swinging (and catching) mechanic. Using Pygame Zero, we can get our basic display set up quickly. In this case, we can split the background into three layers: the background, including the back of the pathway and the tree trunks, the treetops, and the front of the pathway. With these layers we can have a rope swinging with its pivot point behind the leaves of the trees, and, if Harry gets a jump wrong, it will look like he falls down the hole in the ground. The order in which we draw these to the screen is background, rope, treetops, Harry, and finally the front of the pathway.

Now, let's get our rope swinging. We can create an Actor and anchor it to the centre and top of its bounding box. If we rotate it by changing the angle property of the Actor, then it will rotate at the top of the Actor rather than the mid-point. We can make the rope swing between -45 degrees and 45 degrees by increments of 1, but if we do this, we get a rather robotic sort of movement. To fix this, we add an 'easing' value which we can calculate using a square root to make the rope slow down as it reaches the extremes of the swing.

Our Harry character will need to be able to run backwards and forwards, so we'll need a few frames of animation. There are several ways of coding this, but for now, we can take the x coordinate and

Swing when you're winning

Download the code from GitHub: **wfmag.cc/wfmag48**

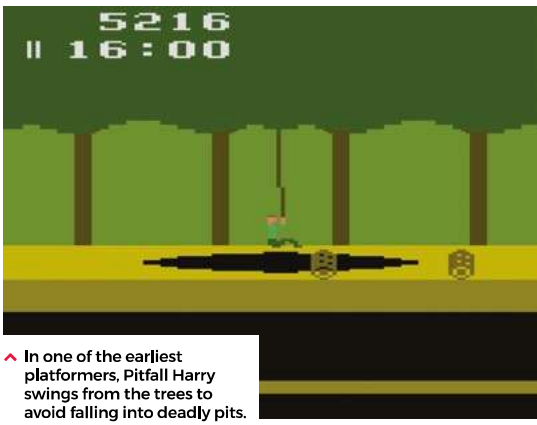
Here's Mark's code snippet, which gets a swinging rope and a jumping adventurer running in Python. To get it working on your system, you'll need to install Pygame Zero – full instructions are available at **wfmag.cc/pgzero**.

```
# Pitfall!
import math

rope = Actor('rope',midtop=(400,110), anchor=('center', 'top'))
harry = Actor('harry', (80,290))
harry.attached = False
harry.jump = 0
harry.onground = True
swing = -1

def draw():
    screen.blit("background", (0, 0))
    rope.draw()
    screen.blit("trees", (0, 0))
    harry.draw()
    screen.blit("platform", (0, 335))
    if harry.x > 550 and harry.y < 300: screen.draw.text("You
made it over!", center=(400, 560), owidth=0.5, ocolor=(0,0,255),
color=(255,255,255) , fontsize=40)

def update():
    global swing
    if rope.angle < -45:
        rope.angle = -45
        swing = 1
    if rope.angle > 45:
        rope.angle = 45
        swing = -1
    easing = (7-(math.sqrt(abs(rope.angle))))/3
    rope.angle += swing*easing
    oldx = harry.x
    harry.onground = False
    if (harry.y > 289 and harry.y < 293) or (harry.y > 468 and
harry.y < 471): harry.onground = True
    if harry.x > 260 and harry.x < 540 and harry.y > 290 and harry.y
< 470 : harry.onground = False
    if keyboard.right and (harry.onground == True or harry.jump > 0):
        harry.x += 2
        harry.image = "harry"+str(int((harry.x/20)%4))
    if keyboard.left and (harry.onground == True or harry.jump > 0):
        harry.x -= 2
        harry.image = "harry"+str(int((harry.x/20)%4))+"r"
    if harry.jump > 0:
        harry.y -= 2
        harry.jump -=1
        harry.image = "harry0"
    else:
        if harry.y < 290 and harry.jump == 0 and harry.attached ==
False:
            harry.y += 2
        elif harry.jump == 0 and harry.x > 255 and harry.x < 540 and
harry.y < 470:
            harry.y += 2

    if oldx == harry.x and harry.jump == 0 : harry.image = "harry"
    if harry.collidepoint (rope.left, rope.bottom) and rope.angle < 25:
        harry.attached = True
    if harry.attached == True:
        harry.image = "harryrope"
        harry.y = rope.bottom + 32
        harry.x = rope.x + (rope.angle * 2.7) - 12

def on_key_down(key):
    if key == keys.SPACE:
        if harry.y == 290 or harry.attached == True:
            harry.jump = 30
            harry.attached = False
        if harry.y > 450: harry.pos = (80,290)
```

work out which frame to display as the x value changes. If we have four frames of running animation, then we would use the `%4` operator and value on the x coordinate to give us animation frames of 0, 1, 2, and 3. We use these frames for running to the right, and if he's running to the left, we just mirror the images. We can check to see if Harry is on the ground or over the pit, and if he needs to be falling downward, we add to his y coordinate. If he's jumping (by pressing the **SPACE** bar), we reduce his y coordinate.

We now need to check if Harry has reached the rope, so after a collision, we check to see if he's connected with it, and if he has, we mark him as attached and then move him with the end of the rope until the player presses the **SPACE** bar and he can jump off at the other side. If he's swung far enough, he should land safely and not fall down the pit. If he falls, then the player can have another go by pressing the **SPACE** bar to reset Harry back to the start.

That should get Pitfall Harry over one particular obstacle, but the original game had several other challenges to tackle – we'll leave you to add those for yourselves.



In one of the earliest platformers, Pitfall Harry swings from the trees to avoid falling into deadly pits.