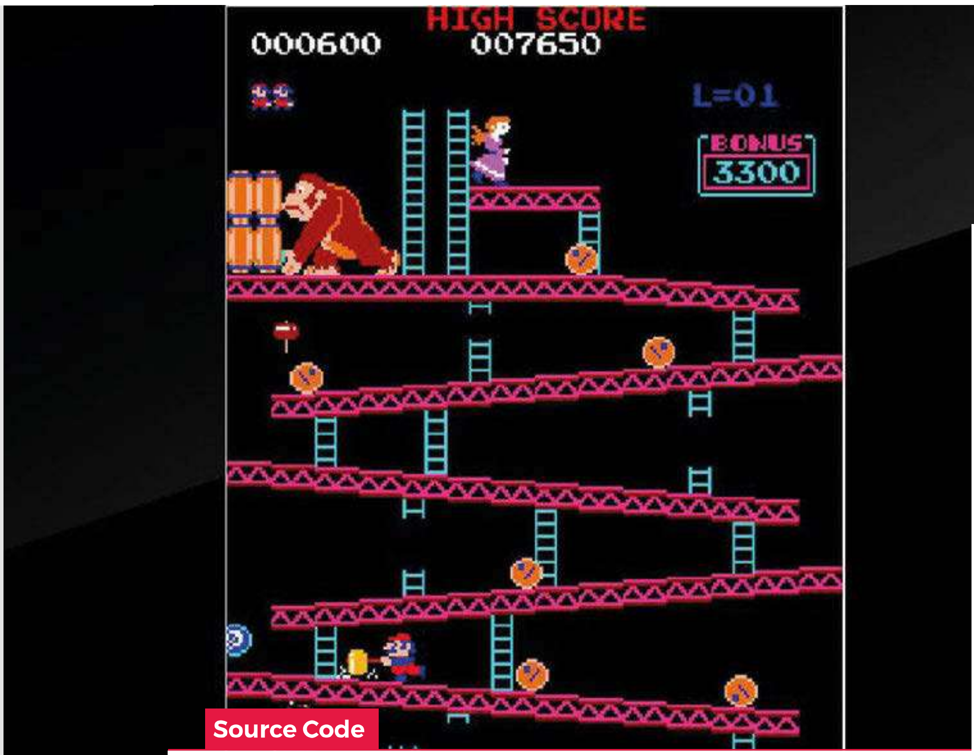


Released in 1981, *Donkey Kong* was one of the most important games in Nintendo's history.

It's fair to say Mario's changed quite a bit since this outing.

Source Code

Code your own Donkey Kong barrels

AUTHOR
MARK VANSTONE

Replicate the physics of barrel rolling – straight out of the classic Donkey Kong

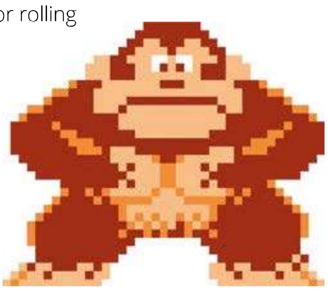
D*onkey Kong* first appeared in arcades in 1981, and starred not only the titular angry ape, but also a bouncing, climbing character called Jumpman – who later went on to star in Nintendo's little-known series of *Super Mario* games. *Donkey Kong* featured four screens per level, and the goal in each was to avoid obstacles and guide Mario (sorry, Jumpman) to the top of the screen to rescue the hapless Pauline. Partly because the game was so ferociously difficult from the beginning, *Donkey Kong*'s first screen is arguably the most recognisable today: Kong lobs an endless stream of barrels, which roll down a network of crooked girders and threaten to knock Jumpman flat.

Donkey Kong may have been a relentlessly tough game, but we can recreate one of its most recognisable elements with relative ease. We can get a bit of code running with Pygame Zero – and a couple of functions borrowed from Pygame – to make barrels react to the platforms they're on, roll down in the direction of a slope, and fall off the end onto the next platform. It's a very simple physics simulation using an invisible bitmap to test where the platforms are and which way they're sloping. We also have some ladders which the barrels randomly either roll past or sometimes use to descend to the next platform below.

"It's a very simple physics simulation using an invisible bitmap"

Once we've created a barrel as an Actor, the code does three tests for its platform position on each update: one to the bottom-left of the barrel, one bottom-centre, and one bottom-right. It samples three pixels and calculates how much red is in those pixels. That tells us how much platform is under the barrel in each position. If the platform is tilted right, the number will be higher on the left, and the barrel must move to the right. If tilted left, the number will be higher on the right, and the barrel must move left. If there is no red under the centre point, the barrel is in the air and must fall downward.

There are just three frames of animation for the barrel rolling (you could add more for a smoother look): for rolling right, we increase the frame number stored with the barrel Actor; for rolling to the left, we decrease the frame number; and if the barrel's going down



Rolling barrels in Python

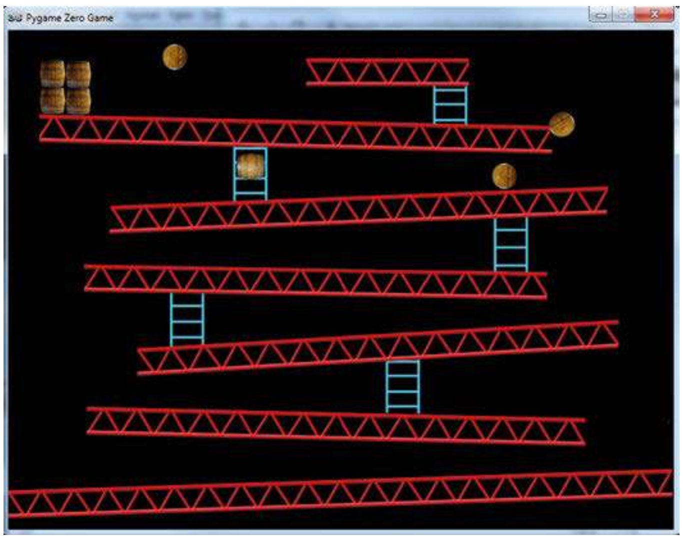
Here's Mark's code snippet, which recreates *Donkey Kong*'s rolling barrels in Python. To get it running on your system, you'll first need to install Pygame Zero – you can find full instructions at **wfmag.cc/pgzero**

Download the code from GitHub: **wfmag.cc/ wfmag24**

```
# Donkey Kong Barrels
from random import randint
from pygame import image, Color
import math

barrels = []
platformMap = image.load(‘images/map.png’)
spacer = 0

def draw():
    screen.blit(“background”, (0, 0))
    for b in range(len(barrels)):
        if onScreen(barrels[b].x, barrels[b].y):
            barrels[b].draw()

def update():
    global spacer
    if randint(0,100) == 1 and spacer < 0:
        makeBarrel()
        spacer = 100
    spacer -= 1
    for b in range(len(barrels)):
        x = int(barrels[b].x)
        y = int(barrels[b].y)
        if onScreen(x,y):
            testcol1 = testPlatform(x-16,y+16,0)
            testcol2 = testPlatform(x,y+16,0)
            testcol3 = testPlatform(x+16,y+16,0)
            move = 0
            if testcol1 > testcol3: move = 1
            if testcol3 > testcol1: move = -1
            barrels[b].x += move
            if move != 0: barrels[b].frame += move * 0.1
            else: barrels[b].frame += 0.1
            if barrels[b].frame >= 4: barrels[b].frame = 1
            if barrels[b].frame < 1: barrels[b].frame = 3.9
            testladder = platformMap.get_at((x,y+32))
            if testladder[2] == 255:
                if randint(0,150) == 1:
                    barrels[b].y += 20
            if testcol2 == 0: barrels[b].y += 1
            frame = str(math.floor(barrels[b].frame))
            if testPlatform(x,y+16,2) > 0:
                barrels[b].image = “bfrfront” + frame
            else:
                barrels[b].image = “bfrside” + frame

def onScreen(x,y):
    return x in range(16,784) and y in range(16,584)

def makeBarrel():
    barrels.append(Actor(‘bfrfront1’, center=(200, 30)))
    barrels[len(barrels)-1].frame = 1

def testPlatform(x,y,col):
    c = 0
    for z in range(3):
        rgb = platformMap.get_at((x,y+z))
        c += rgb[col]
    return c
```

a ladder, we use the front-facing images for the animation. The movement down a ladder is triggered by another test for the blue component of a pixel below the barrel. The code then chooses randomly whether to send the barrel down the ladder.

The whole routine will keep producing more barrels and moving them down the platforms until they reach the bottom. Again, this is a very simple physics system, but it demonstrates how those rolling barrels can be recreated in just a few lines of code. All we need now is a jumping player character (which could use the same invisible map to navigate up the screen) and a big ape to sit at the top throwing barrels, then you'll have the makings of your own fully featured *Donkey Kong* tribute.



Our *Donkey Kong* tribute up and running in Pygame Zero. The barrels roll down the platforms and sometimes the ladders.