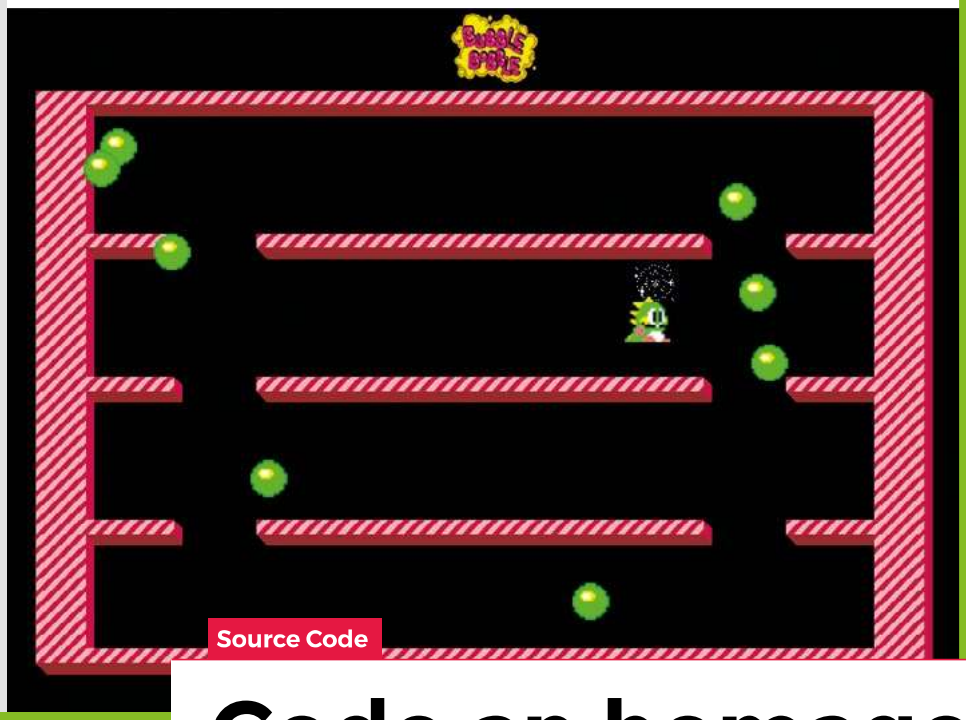

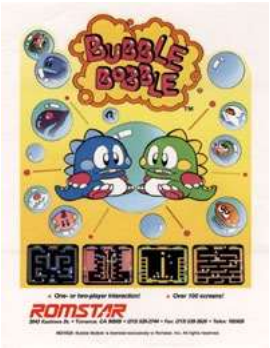

Bubble Bobble brought us a whopping 100 levels to explore.

Our homage to the classic *Bubble Bobble*. No monsters, but the bubble floating and bursting mechanics are spot on.

Code an homage to Bubble Bobble

Blow bubbles and burst them in our tribute to an arcade classic

AUTHOR
MARK VANSTONE

Released in 1986, *Bubble Bobble* looked quite unlike any other game available in arcades at the time. Designed by Taito's Fukio Mitsuji, it was a cooperative game in which two players took control of a pair of dragons, Bub and Bob, and blew bubbles to trap enemies, which in turn could then be popped. This nobbled the enemy inside and released bonus items which could be collected for extra points. Unlike the various shooting, driving, and fighting games at the time, *Bubble Bobble* was a cartoon-like game with a broad appeal; Mitsuji said in later interviews that he wanted the game to be enjoyed by couples. Maybe this is partly why *Bubble Bobble* was a big hit for Taito, and remains one of its most widely ported and fondly remembered games of the 1980s.

For our Pygame Zero sample this month, we'll concentrate on the game's bubble mechanics and see how we can fire them, push them around, and pop them as they float around the level. To start with, we'll need a set of platforms for our little dragon to run around on. We'll keep it simple for this example and have a border of walls and three levels of platform that we can define with a simple nested loop. Each of the platform blocks are Actors which we put into a list before we start the main program. Our `draw()` function is quite simple, drawing our background, then the platforms, our bubbles, and finally Bub, our little dragon.

To move Bub around, we can detect the arrow keys in the `update()` function. We cycle through three frames to make him walk along the platforms and then apply some gravity to him so that if he walks off the end, he'll fall down to the next level. To get Bub jumping, we use the `on_key_down()` function and test to see if he's on the ground, and if so, set the Bub Actor property 'jump' to 60, which will set off upward motion for 60 frames – at which point Bub will start falling back down as our gravity check takes over again.

Now for the bubbles. We detect the **SPACE** bar being pressed, create a new bubble, and add it to a list. To start with, it will fly out horizontally in the direction Bub is facing. We can check how far it has travelled with the status property of the new bubble. After the status property reaches zero, we stop the bubble moving horizontally and let it rise upwards using the `driftx` and `drifty` properties. We can change these properties at random to give the bubble a bit more of an uncertain direction. We can also reverse the properties to make the bubble bounce off the side and top walls. We check all the bubbles each frame to make sure they aren't colliding with any other bubbles, too.

We'll also want to check for collision between Bub and any bubbles. If he runs into a bubble, we want him to push the bubble along, but if he jumps and hits the bubble with his spikes, the bubble will pop. There's also a timer property on the bubbles that means after a while the bubble will burst by itself. So that's the basics of the bubbles – you may want to add some monsters and goodies for Bub to collect, or perhaps make the platforms more complicated. As always, we'll leave that for you to work out.

Blowing bubbles in Python

Here's Mark's code, which re-creates *Bubble Bobble*'s blowing-and-bursting mechanics in Python. To get it working on your machine, you'll first need to install Pygame Zero. You can find full instructions at **wfmag.cc/pgzero**.


```
# Bubble Bobble
import pgzrun
import random
bub = Actor('bubr0',(400,300))
bub.direction = "r"
count = bub.jump = 0
bub.onground = False
platformActors = []
bubbles = []
for r in range(40):
    for c in range(20):
        if c == 0 or c == 19 or r == 0 or r== 39 or ((r == 10 or r == 20
or r == 30) and c != 3  and c != 4 and c != 15  and c != 16):
            platformActors.
append(Actor('platform11',(50+(c*37),80+(r*12))))

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

def update():
    global count
    if count%20 == 0: bub.image = "bub"+ bub.direction + "0"
    if keyboard.left:
        moveBub(-1,0)
        bub.direction = "l"
        bub.image = "bubl"+ str(int(count/8)%3)
    if keyboard.right:
        moveBub(1,0)
        bub.direction = "r"
        bub.image = "bubr"+ str(int(count/8)%3)
    checkGravity()
    updateBubbles()
    count += 1

def on_key_down(key):
    if key.name == "UP":
        if bub.onground == True: bub.jump = 60
    if key.name == "SPACE": fireBubble()

def drawPlatforms():
    for p in range(len(platformActors)): platformActors[p].draw()

def moveBub(x,y):
    if bub.x+x < 720 and bub.x+x > 80:
        bub.x += x*2
        for b in range(len(bubbles)):
            if bubbles[b].collidepoint((bub.x,bub.y)): bubbles[b].x += x*2

def checkGravity():
    if bub.jump > 0:
        if bub.y > 85: bub.y -= 3 + (bub.jump/30)
        bub.jump -=1
        if bub.jump <= 0: bub.y = (bub.y // 2) *2
        for b in range(len(bubbles)):
            if bubbles[b].collidepoint((bub.x,bub.y)) and bubbles[b].
status == 0:
                bubbles[b].countdown = 10
                bub.jump = 0
                bub.y = (bub.y // 2) *2
    bub.onground = False
    for p in range(len(platformActors)):
        if((bub.x > platformActors[p].x-20 and bub.x < platformActors[p].
x+20) and bub.y+18 == platformActors[p].y-14): bub.onground = True
    if bub.onground == False: bub.y += 2

def fireBubble():
    bub.image = "bub"+ bub.direction+ "3"
    bubbles.append(Actor('bubble4',(bub.x,bub.y)))
    bubbles[len(bubbles)-1].status = 20
    bubbles[len(bubbles)-1].direction = bub.direction
    bubbles[len(bubbles)-1].driftx = 0
    bubbles[len(bubbles)-1].drifty = -0.5
    bubbles[len(bubbles)-1].countdown = 1000

def drawBubbles():
    for b in range(len(bubbles)): bubbles[b].draw()

def updateBubbles():
    for b in range(len(bubbles)):
        if bubbles[b].status > 0:
            if bubbles[b].direction == "l":
                bubbles[b].pos = checkCollision(bubbles[b],-bubbles[b].
status,0)
            else:
                bubbles[b].pos = checkCollision(bubbles[b],bubbles[b].
status,0)
            bubbles[b].status -= 1
        else:
            bubbles[b].pos = checkCollision(bubbles[b],bubbles[b].
driftx,bubbles[b].drifty)
            if random.randint(0, 500) == 1: bubbles[b].driftx = (random.
randint(0, 4)-2)/10
        if bubbles[b].countdown > 10: bubbles[b].image =
"bubble"+str(int(bubbles[b].status/5))
        else: bubbles[b].image = "bubble-1"
        bubbles[b].countdown -= 1
    for b in range(len(bubbles)):
        if bubbles[b].countdown < 0:
            bubbles.remove(bubbles[b])
            break;

def checkCollision(o,xinc,yinc):
    if o.x+xinc > 720 or o.x +xinc < 80 or o.y+yinc < 90 or o.y+yinc >
550:
        o.drifty = o.drifty *-1
        o.driftx = o.driftx *-1
        return o.pos
    for b in range(len(bubbles)):
        if bubbles[b].collidepoint((o.x+xinc,o.y+yinc)) and o !=
bubbles[b]:
            bubbles[b].x += xinc
            bubbles[b].driftx = (random.randint(0, 4)-2)/10
    return o.x+xinc,o.y+yinc

pgzrun.go()
```