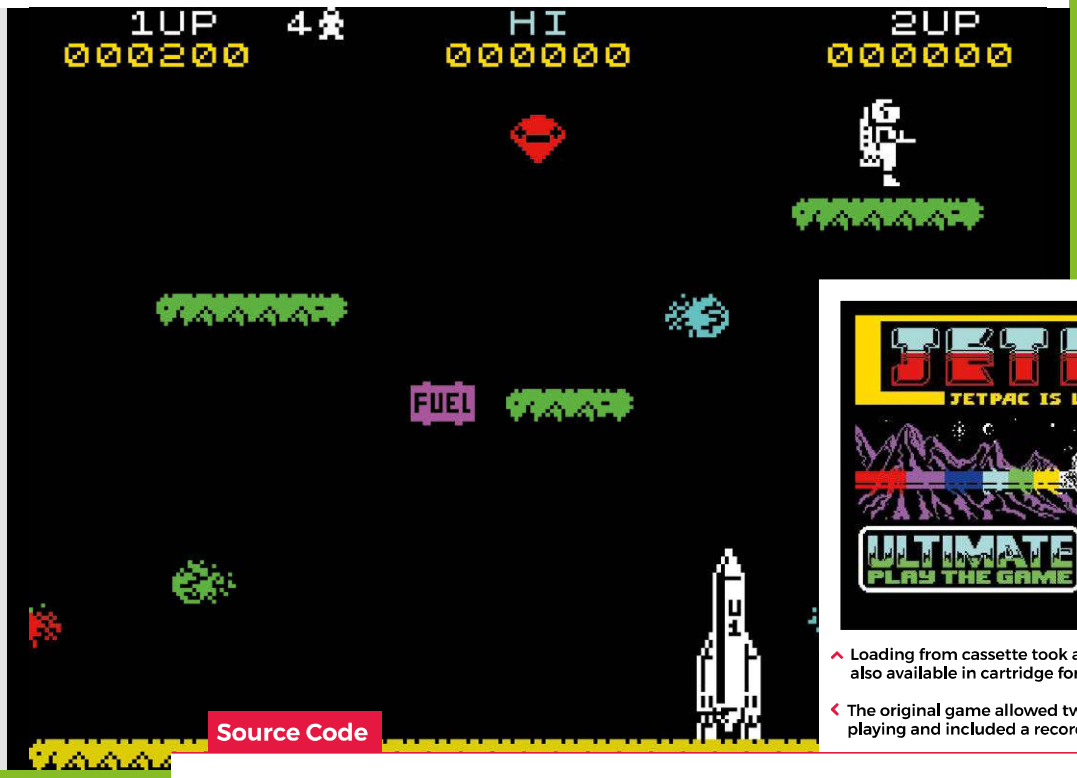

Loading from cassette took a while, but *Jetpac* was also available in cartridge format.

The original game allowed two players to take turns playing and included a record of the highest score.

Source Code

Code Jetpac's rocket building action

Pick up parts of a spaceship, fuel it up, and take off in Mark's rendition of a ZX Spectrum classic

AUTHOR
MARK VANSTONE

For ZX Spectrum owners, there was something special about waiting for a game to load, with the sound of zeros and ones screeching from the cassette tape player next to the computer. When the loading screen – an image of an astronaut and Ultimate Play the Game's logo – appeared, you knew the wait was going to be worthwhile. Created by brothers Chris and Tim Stamper in 1983, *Jetpac* was one of the first hits for their studio, Ultimate Play the Game. The game features the hapless astronaut Jetman, who must build and fuel a rocket from the parts dotted around the screen, all the while avoiding or shooting swarms of deadly aliens.

This month's code snippet will provide the mechanics of collecting the ship parts and fuel to get Jetman's spaceship to take off. We can use the in-built Pygame Zero Actor objects for all the screen elements and the Actor collision routines to deal with gravity and picking up items. To start, we need to initialise our Actors. We'll need our Jetman, the ground, some platforms, the three parts of the rocket, some fire for the rocket engines, and a fuel container. The way each Actor behaves will be determined by a set of lists. We have a list for objects with gravity,

> **"Assemble a rocket, fill it with fuel, and lift off"**

objects that are drawn each frame, a list of platforms, a list of collision objects, and the list of items that can be picked up.

Our `draw()` function is straightforward as it loops through the list of items in the draw list and then has a couple of conditional elements being drawn after. The `update()` function is where all the action happens: we check for keyboard input to move Jetman around, apply gravity to all the items on the gravity list, check for collisions with the platform list, pick up the next item if Jetman is touching it, apply any thrust to Jetman, and move any items that Jetman is holding to move with him. When that's all done, we can check if refuelling levels have reached the point where Jetman can enter the rocket and blast off.

If you look at the helper functions `checkCollisions()` and `checkTouching()`, you'll see that they use different methods of collision detection, the first being checking for a collision with a specified point so we can detect collisions with the top or bottom of an actor, and the touching collision is a rectangle or bounding box collision, so that if the bounding box of two Actors intersect, a collision is registered. The other helper function `applyGravity()` makes everything on the gravity list fall downward until the base of the Actor hits something on the collide list.

So that's about it: assemble a rocket, fill it with fuel, and lift off. The only thing that needs adding is a load of pesky aliens and a way to zap them with a laser gun.

Rocket building in Python

Here's Mark's *Jetpac* code snippet. To get it running on your system, you'll need to install Pygame Zero – you can find full instructions at **wfmag.cc/pgzero**.


```
import random
import time
t0 = time.clock()
jetman = Actor('jetman1',(400,500))
ground = Actor('ground',(400,550))
platform1 = Actor('platform1',(400,350))
platform2 = Actor('platform2',(200,200))
platform3 = Actor('platform3',(650,200))
rocket1 = Actor('rocket1',(520,500))
rocket2 = Actor('rocket2',(400,300))
rocket3 = Actor('rocket3',(200,150))
rocketFire = Actor('rocketfire',(521,0))
fuel = Actor('fuel',(50,-50))
gravityList = [jetman,rocket1,rocket2,rocket3,fuel]
drawList = [rocket1, rocket2, rocket3, ground,platform1,
platform2, platform3, fuel]
platformList = [ground,platform1,platform2,platform3]
collideList = [rocket1,rocket2,rocket3]
pickupList = [rocket2,rocket3,fuel,0,fuel,0,fuel,0,0]
gravity = 1.5
jetman.thrust = jetman.holding = jetman.item = gameState =
fuelLevel = timeElapsed = 0
jetman.dir = "l"
def draw():
    global timeElapsed
    screen.clear()
    for i in range(0, len(drawList)):
        drawList[i].draw()
    if gameState == 0:
        jetman.draw()
        timeElapsed = int(time.clock() - t0)
    else:
        rocketFire.draw()
        screen.draw.text("MISSION ACCOMPLISHED", center =
(400, 300), owidth=0.5, ocolor=(255,255,255), color=(0,0,255),
fontsize=80)
    screen.draw.text("TIME:"+str(timeElapsed), center= (400, 20),
owidth=0.5, ocolor=(255,255,255), color=(255,0,0), fontsize=40)
def update():
    global gameState, fuelLevel
    burn = ""
    if gameState == 0:
        if keyboard.up:
            jetman.thrust = limit(jetman.thrust+0.3,0,5)
            burn = "f"
        if keyboard.left:
            jetman.dir = "l"
            jetman.x -= 1
        if keyboard.right:
            jetman.dir = "r"
            jetman.x += 1
        applyGravity()
        coll =
checkCollisions(platformList,(jetman.x,jetman.y-32))
        if coll == False:
            jetman.y -= jetman.thrust
        if pickupList[jetman.item] != 0:
            if checkTouching(pickupList[jetman.item], jetman):
                jetman.holding = pickupList[jetman.item]
        jetman.thrust = limit(jetman.thrust-0.1,0,5)
        jetman.image = "jetman" + jetman.dir + burn
        if jetman.holding != 0 :
            jetman.holding.pos = jetman.pos
            if jetman.holding.x == rocket1.x and jetman.
holding.y < 440:
                jetman.holding = 0
                jetman.item += 1
        if fuel.x == rocket1.x and fuel.y+16 > rocket3.y-32 and
jetman.holding == 0:
            fuelLevel += 1
            if fuelLevel < 4:
                jetman.item += 1
                if fuelLevel < 3 :
                    fuel.pos = (random.randint(50, 750),-50)
                else:
                    fuel.pos = (0,650)
                gravityList[fuelLevel].image =
"rocket"+str(fuelLevel)+"f"
        if fuelLevel == 3 and jetman.x == rocket1.x and
jetman.y > rocket3.y:
            gameState = 1
    if gameState == 1:
        rocket1.y -= 1
        rocket2.y -= 1
        rocket3.y -= 1
        rocketFire.y = rocket1.y + 50
def limit(n, minn, maxn):
    return max(min(maxn, n), minn)
def checkCollisions(cList, point):
    for i in range(0, len(cList)):
        if cList[i].collidepoint(point):
            return True
    return False
def checkTouching(a1,a2):
    if a1.colliderect(a2): return True
    return False
def applyGravity():
    for i in range(0, len(gravityList)):
        if
checkCollisions(platformList,(gravityList[i].x,gravityList[i].
y+(gravityList[i].height/2))) == False and
checkCollisions(collideList,(gravityList[i].x,gravityList[i].
y+(gravityList[i].height/2))) == False:
            gravityList[i].y += gravity
```