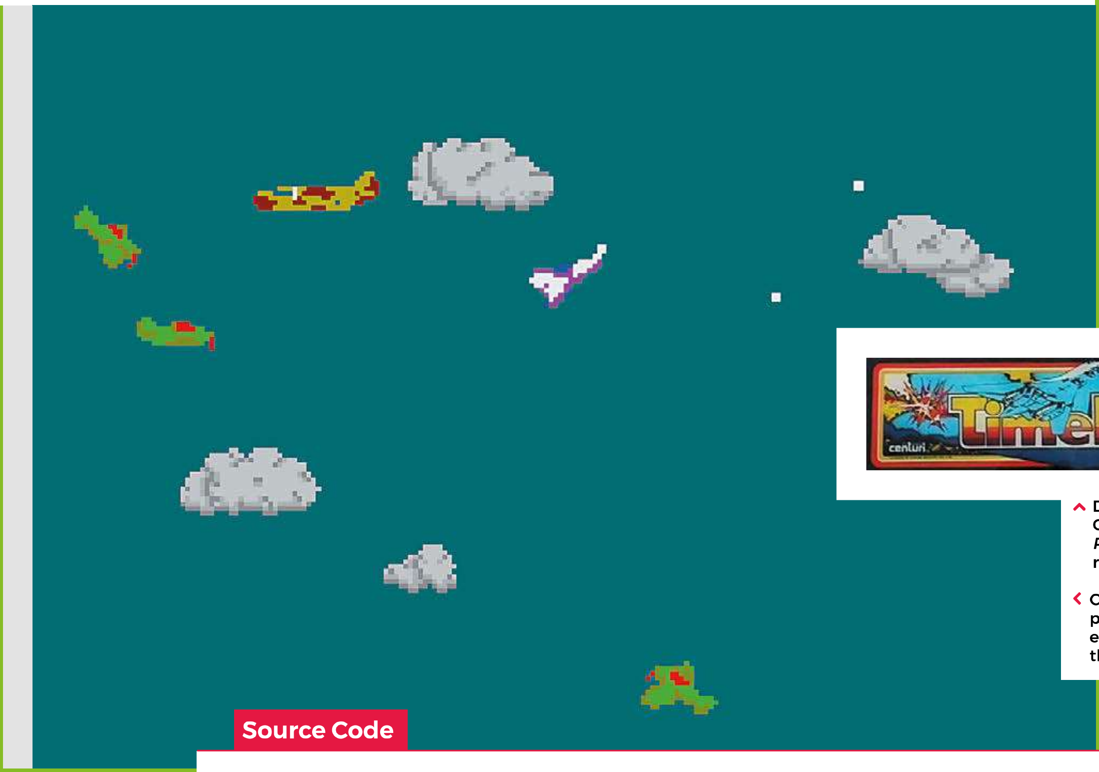

Designed by Yoshiki Okamoto, Konami's *Time Pilot* saw an arcade release in 1982.

One or two players could play *Time Pilot*. Shoot the enemy planes and rescue the parachutists.

Source Code

Recreate Time Pilot's free-scrolling action

AUTHOR
MARK VANSTONE

Fly through the clouds in our re-creation of Konami's shooter

Arguably one of Konami's most successful titles, *Time Pilot* burst into arcades in 1982. Yoshiki Okamoto worked on it secretly, and it proved so successful that a sequel soon followed. In the original, the player flew through five eras, from 1910, 1940, 1970, 1982, and then to the far future: 2001. Aircraft start as biplanes and progress to become UFOs, naturally, by the last level.

Players also rescue other pilots by picking them up as they parachute from their aircraft. The player's plane stays in the centre of the screen while other game objects move around it. The clouds that give the impression of movement have a parallax style to them, some moving faster than others, offering an illusion of depth.

To make our own version with Pygame Zero, we need eight frames of player aircraft images – one for each direction it can fly. After we create a player Actor object, we can get input from the cursor keys and change the direction the aircraft is pointing with a variable which will be set from zero to 7, zero being the up direction. Before we draw the player to the screen, we set the image of the Actor to the stem image name, plus whatever that `direction` variable is at the time. That will give us a rotating aircraft.

To provide a sense of movement, we add clouds. We can make a set of random clouds on the screen and move them in the opposite direction to the player aircraft. As we only have eight directions, we can use a lookup table to change the x and y coordinates rather than calculating movement values. When they go off the screen, we can make them reappear on the other side so that we end up with an 'infinite' playing area. Add a `level` variable to the clouds, and we can move them at different speeds on each `update()` call, producing the parallax effect. Then we need enemies. They will need the same eight frames to move in all directions. For this sample, we will just make one biplane, but more could be made and added.

To get the enemy plane to fly towards the player, we need a little maths. We use the `math.atan2()` function to work out the angle between the enemy and the player. We convert that to a direction which we set in the enemy Actor object, and set its image and movement according to that `direction` variable. We should now have the enemy swooping around the player, but we will also need some bullets. When we create bullets, we need to put them in a list so that we can update each one individually in our `update()`. When the player hits the fire button, we just need to make a new bullet Actor and append it to the bullets list. We give it a direction (the same as the player Actor) and send it on its way, updating its position in the same way as we have done with the other game objects.

The last thing is to detect bullet hits. We do a quick point collision check and if there's a match, we create an explosion Actor and respawn the enemy somewhere else. For this sample, we haven't got any housekeeping code to remove old bullet Actors, which ought to be done if you don't want the list to get really long, but that's about all you need: you have yourself a *Time Pilot* clone!

Download the code from GitHub: **wfmag.cc/wfmag41**

Shooting down planes in Python

Here's Mark's code for a *Time Pilot*-style free-scrolling shooter. To get it running on your system, you'll need to install Pygame Zero – full instructions can be found at **wfmag.cc/pgzero**.

```python
# Time Pilot

from random import randint
import math

gameState = 0
ship = Actor('ship0',(400,300))
biplane = Actor('biplane0',(200,0))
explosion = Actor('explosion1',(0,0))
ship.dir = ship.canfire = biplane.dir = explosion.frame = 0
clouds = []
dirs = [[0,1],[-0.7,0.7],[-1,0],[-0.7,-0.7],[0,-1],[0.7,-
0.7],[1,0],[0.7,0.7]]
for c in range(0, 20):
    clouds.append(Actor('cloud'+str(randint(1,3)),
center=(randint(0,1000)-100, randint(0,800)-100)))
    clouds[c].level = (c+5)/8
bullets = []

def draw():
    screen.fill((0,150,255))
    for b in range(len(bullets)):
        bullets[b].draw()
    ship.draw()
    biplane.draw()
    for c in range(0, 20):
        clouds[c].draw()
    if explosion.frame > 0 and explosion.frame < 10:
        explosion.draw()


def update():
    global gameState
    if gameState == 0:
        if keyboard.left:
            ship.dir -= 0.1
            if ship.dir < 0: ship.dir = 7.9
        if keyboard.right:
            ship.dir += 0.1
            if ship.dir > 7.9: ship.dir = 0
        if keyboard.space:
            if ship.canfire <= 0: fireBullet()
        ship.canfire -= 1
        ship.image = "ship"+str(int(ship.dir))
        myradians = math.atan2(ship.x-biplane.x, ship.y-
biplane.y)
        mydegrees = math.degrees(myradians)
        biplane.dir = (180-mydegrees)/45

        biplane.x += (dirs[int(ship.dir)][0]) -
((dirs[int(biplane.dir)][0])/2)
        biplane.y += (dirs[int(ship.dir)][1]) -
((dirs[int(biplane.dir)][1])/2)
        biplane.image = "biplane"+str(int(biplane.dir))
        if explosion.frame > 0:
            explosion.frame += 1
            explosion.x += (dirs[int(ship.dir)][0])
            explosion.y += (dirs[int(ship.dir)][1])
            if explosion.frame < 10 : explosion.image =
"explosion"+str(math.ceil(explosion.frame/3))
        for c in range(0, 20):
            clouds[c].x += dirs[int(ship.dir)][0]*clouds[c].
level
            if clouds[c].x > 900: clouds[c].x = -100
            if clouds[c].x < -100: clouds[c].x = 900
            clouds[c].y += dirs[int(ship.dir)][1]*clouds[c].
level
            if clouds[c].y > 700: clouds[c].y = -100
            if clouds[c].y < -100: clouds[c].y = 700
        for b in range(len(bullets)):
            bullets[b].x -= (dirs[bullets[b].dir][0]*5) +
(dirs[int(ship.dir)][0])
            bullets[b].y -= (dirs[bullets[b].dir][1]*5) +
(dirs[int(ship.dir)][1])
            if biplane.collidepoint(bullets[b].pos) == True:
                byplaneHit()

def limit(n, minn, maxn):
    return max(min(maxn, n), minn)

def fireBullet():
    bullets.append(Actor('bullet', center=(400, 300)))
    bullets[len(bullets)-1].dir = int(ship.dir)
    ship.canfire = 5

def byplaneHit():
    explosion.frame = 1
    explosion.pos = biplane.pos
    biplane.pos = (randint(0,800),0)
```

Our homage to Konami's arcade classic.