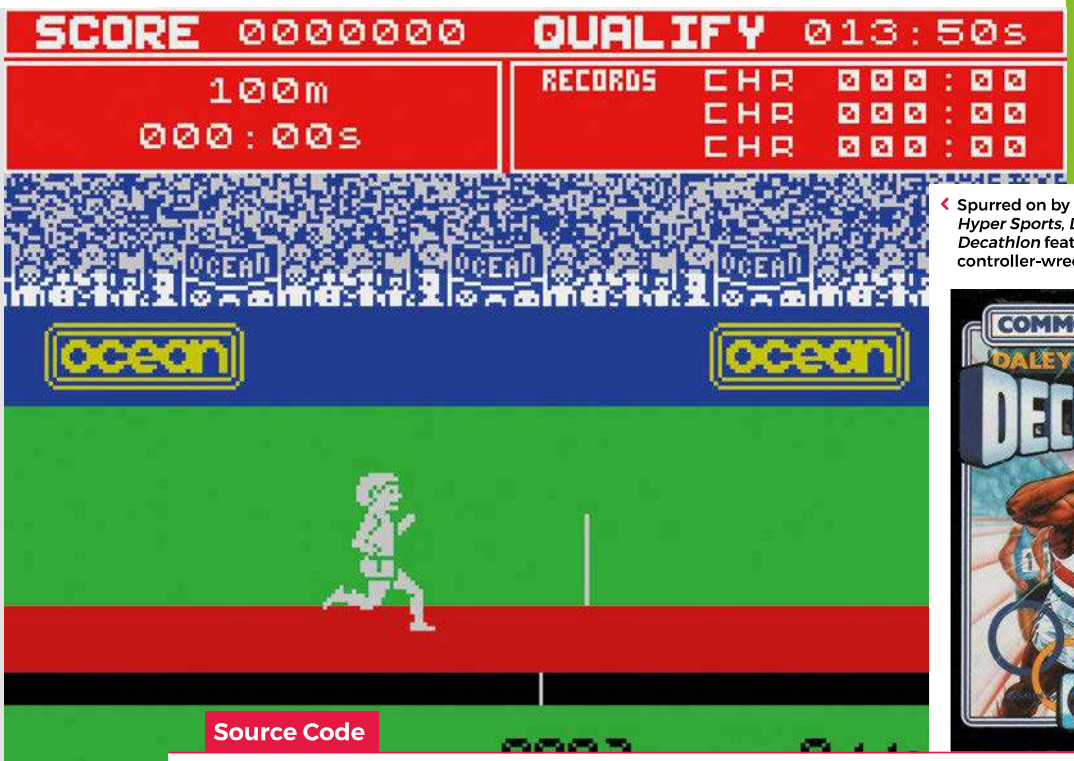
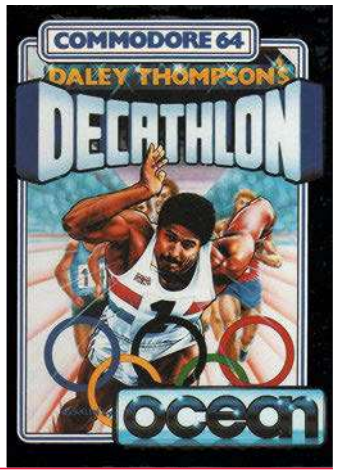


Spurred on by the success of Konami's *Hyper Sports*, *Daley Thompson's Decathlon* featured a wealth of controller-wrecking minigames.



Source Code

Make a keyboard-bashing sprint game

Learn how to code a sprinting minigame straight out of Daley Thompson's Decathlon

AUTHOR
RIK CROSS

Released in 1984, *Daley Thompson's Decathlon* was a memorable entry in what's sometimes called the 'joystick killer' genre: players competed in sporting events that largely consisted of frantically waggling the controller or battering the keyboard. I'll show you how to create a sprinting game mechanic in Python and Pygame. There are variables in the `Sprinter()` class to keep track of the runner's speed and distance, as well as global constant `ACCELERATION` and `DECELERATION` values to determine the player's changing rate of speed. These numbers are small, as they represent the number of metres per frame that the player accelerates and decelerates. The player increases the sprinter's speed by alternately pressing the left and right arrow keys. This input is handled by the sprinter's `isNextKeyPressed()` method, which returns `True` if the correct key (and only the correct key) is being pressed. A `lastKeyPressed` variable is used to ensure that keys are pressed alternately. The player also decelerates if no key is being pressed, and this rate of deceleration should be sufficiently smaller than the acceleration to allow the player to pick up enough speed.

For the animation, I used a free sprite called 'The Boy' from **gameart2d.com**, and made use of a single idle image and 15 run cycle images. The sprinter starts in the idle state, but switches to the run cycle whenever its speed is greater than 0. This is achieved by using `index()` to find the name of the current sprinter image in the `runFrames` list, and setting the current image to the next image in the list (and wrapping back to the first image once the end of the list is reached). We also need the sprinter to move through images in the run cycle at a speed proportional to the sprinter's speed. This is achieved by keeping track of the number of frames the current image has been displayed for (in a variable called `timeOnCurrentFrame`).

To give the illusion of movement, I've added objects that move past the player: there's a finish line and three markers to regularly show the distance travelled. These objects are calculated using the sprinter's `x` position on the screen along with the distance travelled. However, this means that each object is at most only 100 pixels away from the player and therefore seems to move slowly. This can be fixed by using a `SCALE` factor, which is the relationship between metres travelled by the sprinter and pixels on the screen. This means that objects are initially drawn way off to the right of the screen but then travel to the left and move past the sprinter more quickly.

Finally, `startTime` and `finishTime` variables are used to calculate the race time. Both values are initially set to the current time at the start of the race, with `finishTime` being updated as long as the distance travelled is less than 100. Using the time module, the race time can simply be calculated by `finishTime - startTime`.

Sprinting in Python

Here's Rik's code snippet, which creates a sprinting game 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/wfmag23**

```
from time import time

WIDTH  = 800
HEIGHT = 300

ACCELERATION = 0.005
DECELERATION = 0.0008
# number of pixels representing 1m
SCALE = 75

# display an image at a specific distance along the track
def displayAt(img, pos, y):
    screen.blit(img,(sprinter.x + (pos * SCALE) - (sprinter.
distance * SCALE), y))

class Sprinter(Actor):
    def __init__(self, **kwargs):
        super().__init__(image='idle', pos=(200,220), **kwargs)
        self.startTime = time()
        self.finishTime = time()
        self.runFrames = ['run' + str(i) for i in range(1,16)]
        self.timeOnCurrentFrame = 0
        self.speed = 0
        self.lastPressed = None
        self.keyPressed = False
        self.distance = 0

    def nextFrame(self):
        # start the running animation if currently idle
        if self.image == 'idle':
            self.image = self.runFrames[0]
        else:
            nextImageIndex = (self.runFrames.index(self.image)
+ 1) % len(self.runFrames)
            self.image = self.runFrames[nextImageIndex]

    def isNextKeyPressed(self):
        if keyboard.left and self.lastPressed is not 'left' and
not keyboard.right:
            self.lastPressed = 'left'
            return True
        if keyboard.right and self.lastPressed is not 'right'
and not keyboard.left:
            self.lastPressed = 'right'
            return True
        return False

    def update(self):
        if self.isNextKeyPressed() and self.distance < 100:
            self.speed = min(self.speed + ACCELERATION, 0.15)
        # decelerate if no key pressed
        else:
            self.speed = max(0, self.speed-DECELERATION)
        # use the sprinter's speed to update the distance
        self.distance += self.speed
        # animate the sprinter in relation to its speed
        self.timeOnCurrentFrame += 1
        if self.speed > 0 and self.timeOnCurrentFrame > 10 -
(self.speed * 75):
            self.timeOnCurrentFrame = 0
            self.nextFrame()
        # set to idle animation if not moving
        if self.speed <= 0:
            self.image = 'idle'

sprinter = Sprinter()

def update():
    # move and animate the sprinter
    sprinter.update()
    # add to the finish time if race is still in progress
    if sprinter.distance < 100:
        sprinter.finishTime = time()

def draw():
    screen.clear()
    # draw the track
    screen.blit('track', (0,0))
    # draw distance markers and finish line
    displayAt('25m', 25, 200)
    displayAt('50m', 50, 200)
    displayAt('75m', 75, 200)
    displayAt('finishline', 100, 230)
    # draw the sprinter
    sprinter.draw()
    # draw the current distance and time
    screen.draw.text('Distance (m): ' + str(int(min(100,
sprinter.distance))), (20, 20), fontsize=32, color="white")
    screen.draw.text('Time (s): ' + str(round(sprinter.
finishTime - sprinter.startTime, 2)), (250, 20), fontsize=32,
color="white")
```

Press the left and right arrow keys alternately to increase the sprinter's speed. Objects move across the screen from right to left to give the illusion of sprinter movement.