Source Code

Code an homage to the classic Skate or Die!

AUTHOR
MARK VANSTONE

Master the half-pipe with our retro skating minigame challenge

Skate or Die! came out in 1987 for the ZX Spectrum, NES, and other computers. Players were treated to five different skateboarding events including half-pipe stunts and downhill races. The aim was to score points by completing the challenges faster or with better sequences of stunts.

For our Pygame Zero take on *Skate or Die!*, we'll focus on the half-pipe minigame. The original had quite complicated controls to perform tricks and steer the skateboard, but for this example, we'll just focus on the skateboard speed and making turns at the apex of the half-pipe. We will use the left and right arrows to increase speed in each direction and the up arrow to do a turn.

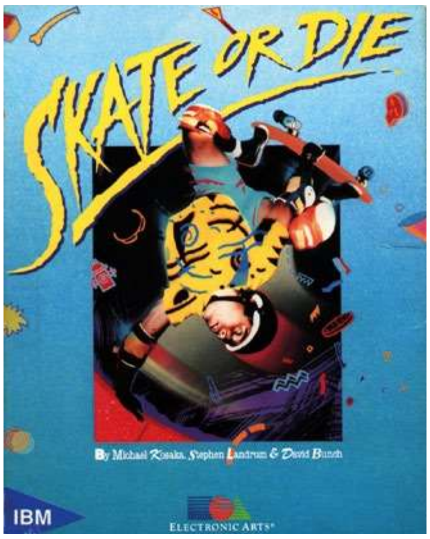
The first thing to do is draw our background. For this example, we've used the background from the C64 version, but you could make up your own version. We also need two skater images: one facing left and one facing right. We'll start the skater at the top of the half-pipe on the right-hand side of the screen. Once we've defined our skater as an Actor, we can add properties like direction and speed. We'll control the speed value with the arrow keys. The left arrow will reduce the speed value, and the right arrow will increase the speed value. A minus value will mean the skater moves left and a positive value means the skater moves right.

Now to make our skater move with the contour of the half-pipe. We need a way to work out what angle the ground is and where the skater is, and to do this we can make a gradient map. We use black pixels to indicate 90 degrees, white pixels to represent flat ground, and shades of grey for everything in between. So when our skater is over white pixels on the gradient map, they'll move horizontally, and as the pixels tend towards black, the movement will be more vertical. We don't have to draw this gradient map to the screen, but we access it by loading it with the Pygame image module and then using the **`image.get_at()`** function to read the pixel colour directly under our skater Actor.

As the gradient map is being read, we should see our skater moving down the half-pipe, across the flat floor, and then up the other side. We next need to change the angle of the skater to match the angle of the half-pipe, and we can get that from the gradient map, too. White pixels set the skater to 0 degrees; if they're on the right-hand side of the screen, black pixels mean 90 degrees, while on the left side, -90 degrees. Now we have an angle for our skater, we need to apply some gravitational effect to the speed so that as they're heading downwards, they accelerate, and when going up, they slow down. This will mean that the skater needs to be going fast enough to reach the apex of the half-pipe in order to do a turn in the air, but if they go too fast, they'll end up on the flat ground on the other side.

To get a turn in the air, we'll detect the up arrow key press. If the skater angle is greater than 75 degrees or less than -75, we trigger a turn. We set a countdown variable called **`switch`**, which we'll check in the **`update()`** function and move the skater vertically up for 30 frames, switch their direction, and then move them down again to complete the turn. We can count a score of how many times our skater completes a turn without ending up back on the flat ground at the top of the half-pipe. If the skater fails too many times, they'll end up falling off the bottom of the half-pipe. We can use the **SPACE** bar to reset them back to the starting point.

And that's the mechanics of our half-pipe game. The original *Skate or Die!*'s minigames each used slightly different controls and backgrounds, but we'll leave you to adapt our existing project to those challenges.



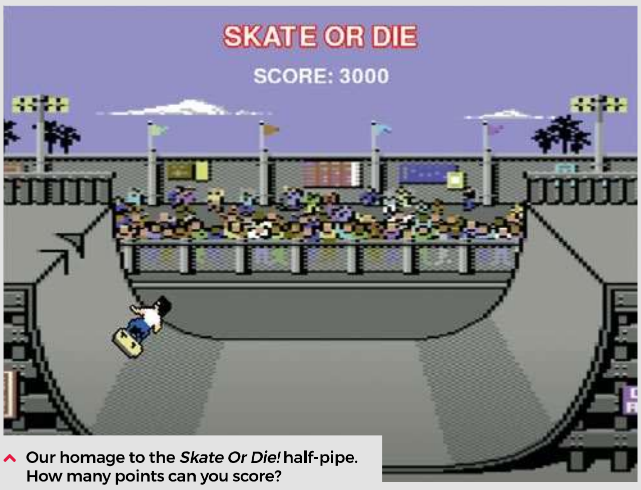
^ *Skate Or Die!* featured some fabulous music by the legendary Rob Hubbard. If you hold the magazine closely, you can just about hear it.

Heaven is a half-pipe

Here’s Mark’s code for a radical skateboarding minigame. To get it working on your system, you’ll first need to install Pygame Zero. Full instructions can be found at **wfmag.cc/pgzero**.

Download the code from GitHub: **wfmag.cc/wfmag58**

```
# Skate or Die
import pgzrun
from pygame import image

skater = Actor(‘skaterl’,center=(700,230), anchor=(‘center’,’bottom’))
skater.direction = “l”
skater.speed = 0
skater.switch = 0
halfpipe = image.load(‘images/halfpipe.png’)
score = 0

def draw():
    screen.blit(“background”, (0, 0))
    skater.draw()
    screen.draw.text(“SKATE OR DIE”, center = (400,
40),color=(255,255,255) , owidth=0.5, ocolor=(255,0,0), fontsize=50)
    screen.draw.text(“SCORE: “+str(score), center = (400,
90),color=(255,255,255) , fontsize=38)

def update():
    if skater.y < 600:
        if keyboard.left and skater.angle  > -20 and skater.speed <= 0:
            skater.speed = limit(skater.speed - 0.2,-13,0)
            skater.y -= 0.2
        if keyboard.right and skater.angle < 20 and skater.speed >= 0:
            skater.speed = limit(skater.speed + 0.2,0,13)
            skater.y -= 0.2
        pixel = halfpipe.get_at((int(skater.x),int(skater.y)))
        if skater.switch > 0:
            skater.switch -= 1
            angle = skater.angle
            if skater.switch == 30:
                if skater.direction == “l”:
                    skater.direction = “r”
                    skater.speed = 1
                    angle = -90
                else:
                    skater.direction = “l”
                    skater.speed = -1
                    angle = 90
                skater.image = “skater”+skater.direction
            skater.angle = angle
            if skater.switch > 30:
                if skater.direction == “l”:
                    skater.x += 0.6
                else:
                    skater.x -= 0.6
                skater.y -= 4
            else:
                skater.y += 3
        else:
            skater.x = limit(skater.x+skater.speed,20,780)
            if skater.x <= 20 or skater.x >=780 and skater.speed > 0:
                skater.speed = 0
                if skater.x <= 20:
                    skater.direction = “r”
                else:
                    skater.direction = “l”
                skater.image = “skater”+skater.direction
            if skater.x > 400:
                offset = 255-pixel.b
            else:
                offset = pixel.b-255
            skater.angle = (offset)/3
            yinc = (offset*(-skater.speed)/100)
            skater.y += yinc
            skater.speed -= (skater.angle/100)
            skater.speed = skater.speed/1.005
    else:
        skater.image = “fallen”+skater.direction

def on_key_down(key):
    global score
    if key.name == “UP”:
        if (skater.angle > 75 and skater.speed > 0) or (skater.angle <
-75 and skater.speed < 0):
            skater.speed = 0
            skater.switch = 60
            score += 1000
    if key.name == “SPACE” and skater.y > 600:
        skater.direction = “l”
        skater.speed = 0
        skater.pos = (720,230)
        skater.image = “skaterl”
        skater.angle = 0
        skater.switch = 15
        score = 0

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

pgzrun.go()
```



Our homage to the *Skate Or Die!* half-pipe. How many points can you score?