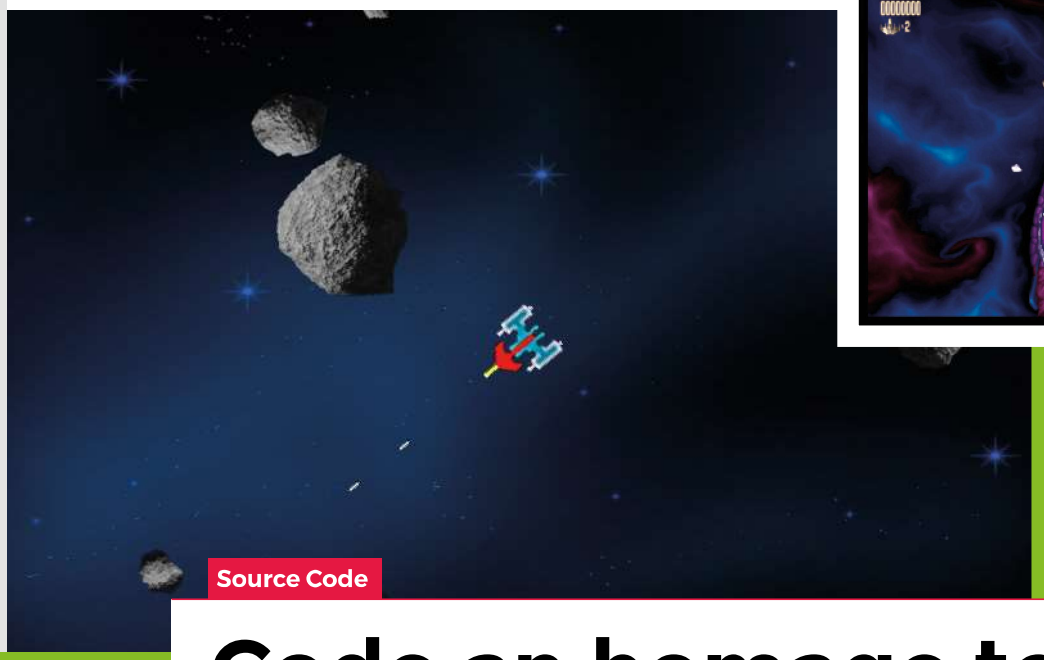

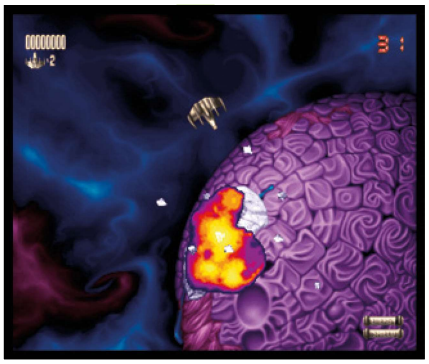

Super Stardust was originally released for Amiga and Amiga CD32.

Our Pygame Zero rock-breaking *Super Stardust* homage.

Source Code

Code an homage to Super Stardust

Vent your hatred of rocks in our Pygame Zero salute to the 16-bit hit

AUTHOR
MARK VANSTONE

The original rock-blasting arcade shooter was, of course, *Asteroids*, released by Atari in 1979. But since then, many clones; one notable example was *Super Stardust*, developed by Finnish developer Bloodhouse and released by Team17 for the Amiga platform in 1994 (for more on that studio's history, see page 28). The game featured considerably better graphics and sound than *Asteroids*, and got great reviews. One of the differences between the two is that *Asteroids* uses vector graphics to draw the screen objects whereas *Stardust* uses sprites.

If you have a look back at Wireframe issue four (**wfmag.cc/4**), you'll find a Source Code article describing how to make a spaceship move using thrust, *Asteroids* style. Rather than go over old ground, this example will cover how to code the asteroids that are broken into pieces by the ship's lasers. We'll use animated sprites for the asteroids, much the same as in the *Super Stardust* version of the game. We'll need three sizes of asteroids – the ones here are 128 × 128, 64 × 64, and 32 × 32 pixels. The twelve animation frames of each size of asteroids have been produced using Blender 3D, which you can download for free from **blender.org**.

We set up four asteroids as Actors, make them into a list, and place them in a square around the centre of the screen. We can set them at different angles so that when we move them in the `update()` function, they'll go in different directions. We also add a status flag that will tell us if we need to draw the asteroid or not. If we set a different start frame for the animation of each asteroid, then they won't look too much like they're all turning together. When we update each asteroid, we use a bit of trigonometry to move each one in the direction they're pointing. We also check to see if they've gone off the screen and, if so, make them appear on the opposite side of the screen. Each `update()` call, we increment the frame being displayed by the asteroid Actor to make it look like it's rotating as it moves.

Now we have our moving, spinning asteroids, we need a ship to shoot them. We'll plop the ship in the middle of the screen and use the arrow keys to rotate it. If you want to add thrust to get the ship to move around the screen, have a look back at the issue four Source Code article mentioned above. We'll use the **SPACE** bar to fire a laser Actor which we'll put in a list. If we assign the opposite rotation of the ship to the bullet, then we can move it in the direction it was fired until it hits an asteroid or goes off the screen. In this sample, there's a cleaning mechanism in the bullet update which filters out all the bullets that are no longer required, dropping them from the list.

To make asteroids break up, we detect when a bullet hits an asteroid actor, and when that happens, it's removed from the list to be drawn by setting its status to 1. Then we create two more smaller asteroids and set their angle to be at right angles to where the bullet came from. This makes them fly off in opposite directions, then in turn, if they're shot, we do the same but create the smallest size of asteroid. If the small asteroids are shot, they have their status set to 1 and no more asteroids are created. When all the asteroids have a status of 1, the level's been cleared. You might want to add some collision detection between the ship and the asteroids, perhaps a scoring system, and moving the ship around the screen, but we'll leave those for you to add.

Gettin' Ziggy Wit It

Here's Mark's code for a *Super Stardust*-style shooter 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.**


```
# Pygame Stardust
import pgzrun
import math

ship = Actor('ship', center=(400, 300))
count = gameover = 0
asteroids = []
bullets = []
for a in range(0, 4):
    asteroids.append(Actor('ast1_'+str((a+1)*3), center=(100+(a*200),1
00+((a%2)*400))))
    asteroids[a].angle = (80*a) + 20
    asteroids[a].status = 0

def draw():
    screen.blit("background", (0, 0))
    for b in range(0, len(bullets)):
        bullets[b].draw()
    drawAsteroids()
    if gameover != 1 or (gameover == 1 and count%2 == 0): ship.draw()
    if gameover == 1 : screen.draw.text("YOU CLEARED ALL THE
ASTEROIDS", center = (400, 300), owidth=0.5, ocolor=(255,255,0),
color=(255,0,0) , fontsize=50)

def update():
    global count
    count += 1
    if gameover == 0:
        if keyboard.left : ship.angle += 2
        if keyboard.right : ship.angle -= 2
        updateBullets()
        updateAsteroids()

def on_key_down(key):
    if gameover == 0:
        if key.name == "SPACE": makeBullet()

def drawAsteroids():
    for a in range(0, len(asteroids)):
        if asteroids[a].status == 0: asteroids[a].draw()

def updateAsteroids():
    global gameover
    asteroidsLeft = False
    for a in range(0, len(asteroids)):
        if asteroids[a].status == 0 : asteroidsLeft = True
        i = int(asteroids[a].image[5:])
        if count%5 == 0: i += 1
        if i > 12: i = 1
        imagebase = asteroids[a].image[0:5]
        angle = asteroids[a].angle
        asteroids[a].x += math.sin(math.radians(angle))
        asteroids[a].y += math.cos(math.radians(angle))
        if asteroids[a].x > 850 : asteroids[a].x -= 850
        if asteroids[a].x < -50 : asteroids[a].x += 850
        if asteroids[a].y > 650 : asteroids[a].y -= 650
        if asteroids[a].y < -50 : asteroids[a].y += 650
        asteroids[a].image = imagebase + str(i)
        asteroids[a].angle = angle
    if asteroidsLeft == False : gameover = 1

def updateBullets():
    global bullets
    bulletsTemp = []
    tb = 0
    for b in range(0, len(bullets)):
        if isOnScreen(bullets[b]) and not hitAsteroid(bullets[b]) :
            bulletsTemp.append(Actor('bullet'))
            bulletsTemp[tb].x = bullets[b].x + 5 * math.sin(math.
radians(bullets[b].angle))
            bulletsTemp[tb].y = bullets[b].y + 5 * math.cos(math.
radians(bullets[b].angle))
            bulletsTemp[tb].angle = bullets[b].angle
            tb += 1
    bullets = bulletsTemp

def hitAsteroid(b):
    for a in range(0, len(asteroids)):
        if asteroids[a].collidepoint(b.pos) and asteroids[a].status ==
0:
            breakAsteroid(a,b.angle)
            return True
    return False

def breakAsteroid(a,angle):
    acount = len(asteroids)
    anum = int(asteroids[a].image[3])
    if anum < 3:
        anum += 1
        asteroids.append(Actor('ast'+str(anum)+'_1',
center=(asteroids[a].pos)))
        asteroids[acount].angle = (angle + 90) % 360
        asteroids[acount].status = 0
        acount += 1
        asteroids.append(Actor('ast'+str(anum)+'_6',
center=(asteroids[a].pos)))
        asteroids[acount].angle = (angle - 90) % 360
        asteroids[acount].status = 0
    asteroids[a].status = 1

def makeBullet():
    a = len(bullets)
    bullets.append(Actor('bullet', center=(400,300)))
    bullets[a].angle = (ship.angle + 180) % 360

def isOnScreen(b):
    if b.x > 0 and b.x < 800 and b.y >0 and b.y < 600 :
        return True
    else :
        return False

pgzrun.go()
```