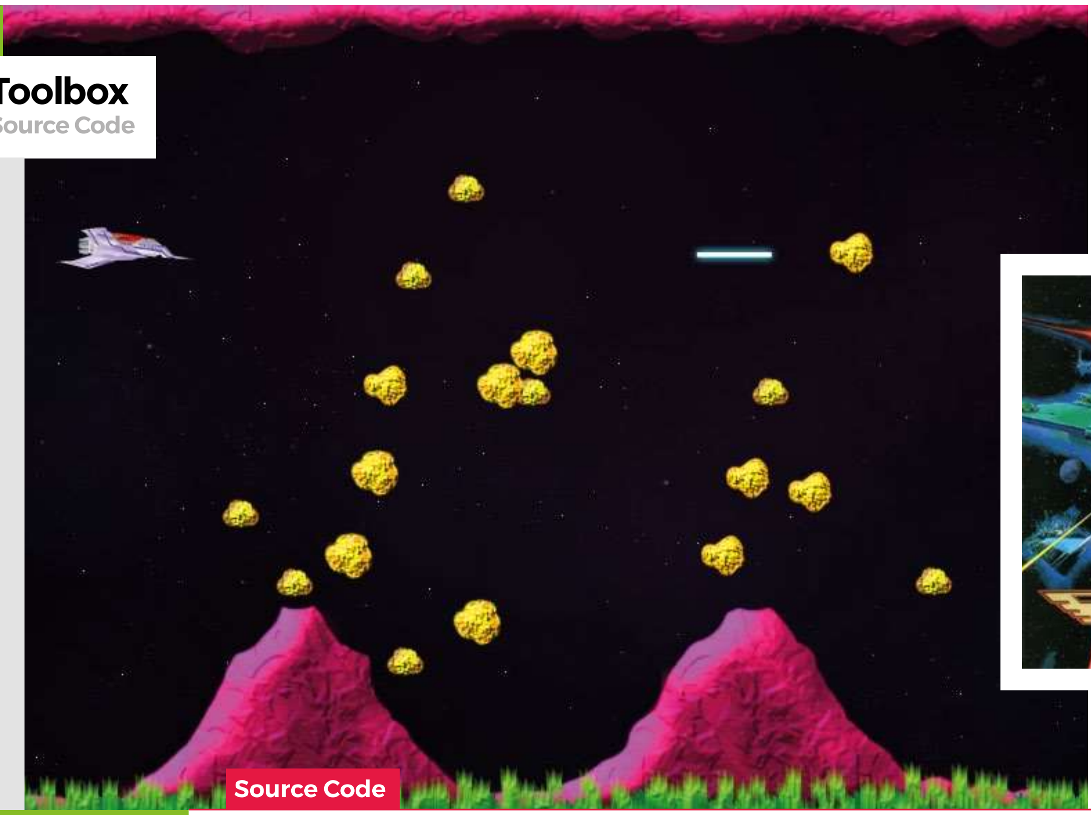


Dodge and shoot the rocks in our homage to the classic *Gradius*.

Recreate Gradius's rock-spewing volcanoes

AUTHOR
MARK VANSTONE

Code an homage to Konami's classic shoot-'em-up

Released by Konami in 1985, *Gradius* – also known as *Nemesis* outside Japan – brought a new breed of power-up system to arcades. One of the keys to its success was the way the player could customise their Vic Viper fighter craft by gathering capsules, which could then be 'spent' on weapons, speed-ups, and shields from a bar at the bottom of the screen.

A seminal side-scrolling shooter, *Gradius* was particularly striking thanks to the variety of its levels: a wide range of hazards were thrown at the player, including waves of aliens, natural phenomena, and boss ships with engine cores that had to be destroyed in order to progress. One of the first stage's biggest obstacles was a pair of volcanoes that spewed deadly rocks into the air: the rocks could be shot for extra points or just avoided to get through to the next section. In this month's Source Code, we're going to have a look at how to recreate the volcano-style flying rock obstacle from the game.

Our sample uses Pygame Zero and the `randint` function from the random module to provide the variations of trajectory that we need our rocks to have. We'll need an actor created for our spaceship and a list to hold our rock Actors. We can also make a bullet Actor so we can make the ship fire lasers and shoot the rocks. We build up the scene in layers in our `draw()` function with a star-speckled background, then our rocks, followed by the foreground of volcanoes, and finally the spaceship and bullets.

In the `update()` function, we need to handle moving the ship around with the cursor keys. We can use a `limit()` function to make sure it doesn't go off the screen, and the **SPACE** bar to trigger the bullet to be fired. After that, we need to update our rocks. At the start of the game our list of rocks will be empty, so we'll get a random number generated, and if the number is 1, we make a new rock and add it to the list. If we have more than 100 rocks in our list, some of them will have moved off the screen, so we may as well reuse them instead of making more new rocks. During each update cycle, we'll need to run through our list of rocks and update their position. When we make a rock, we give it a speed and direction, then when it's updated, we move the rock upwards by its speed and then reduce the speed by 0.2. This will make it fly into the air, slow down, and then fall to the ground.

From this code, we can make rocks appear just behind both of the volcanoes, and they'll fly in a random direction upwards at a random speed. We can increase or decrease the number of rocks flying about by changing the random numbers that spawn them. We should be able to fly in and out of the rocks, but we could add some collision detection to check whether the rocks hit the ship – we may also want to destroy the ship if it's hit by a rock. In our sample, we have an alternative, 'shielded' state to indicate that a collision has occurred. We can also check for collisions with the bullets: if a collision's detected, we can make the rock and the bullet disappear by moving them off-screen, at which point they're ready to be reused.

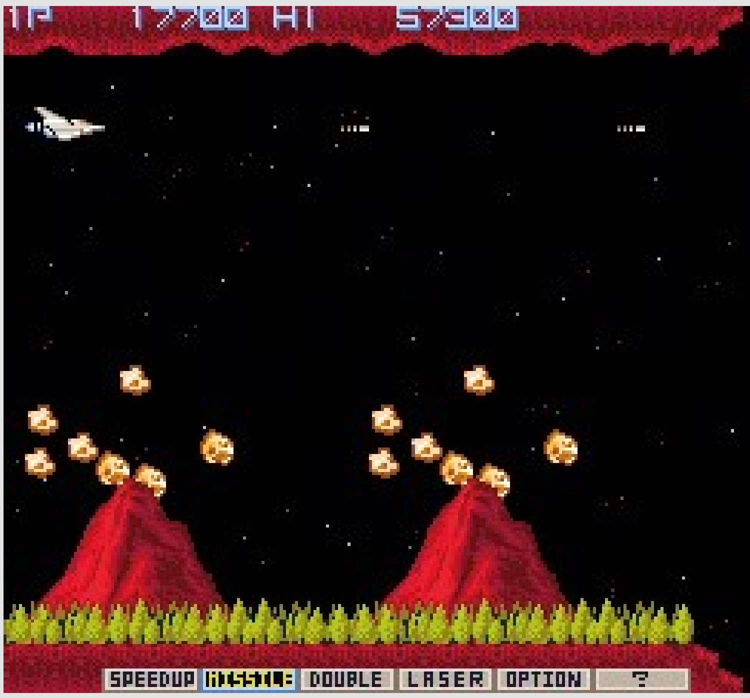
That's about it for this month's sample, but there are many more elements from the original game that you could add yourself: extra weapons, more enemies, or even an area boss.

Vulcan Venture

Here’s Mark’s volcanic code. To get it working on your system, you’ll need to install Pygame Zero – full instructions are available at **wfmag.cc/pgzero.**

Download the code from GitHub: **wfmag.cc/wfmag52**

```
# Gradius
import pgzrun
from random import randint

jet = Actor(‘jet’,(400,300))
bullet = Actor(‘bullet’, center=(850, 0))
rocks = []

def draw():
    screen.blit(“background”, (0, 0))
    drawRocks()
    screen.blit(“foreground”, (0, 0))
    bullet.draw()
    jet.draw()

def update():
    if keyboard.up: jet.y = limit(jet.y-5,50,550)
    if keyboard.down: jet.y = limit(jet.y+5,50,550)
    if keyboard.left: jet.x = limit(jet.x-5,10,790)
    if keyboard.right: jet.x = limit(jet.x+5,10,790)
    if keyboard.space :
        if bullet.x >= 850 : bullet.pos = (jet.x,jet.y+5)
    if bullet.x < 850: bullet.x += 20
    updateRocks()

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

def drawRocks():
    for r in range(0, len(rocks)):
        rocks[r].draw()

def makeRock(pos):
    r = len(rocks)
    if r < 100:
        rocks.append(Actor(‘rock’+str(randint(1,3)), center=pos))
    else:
        r = getOldRock()
        rocks[r].pos = pos
    rocks[r].speed = randint(6,12)
    rocks[r].dir = (randint(0,60)-30)/10

def updateRocks():
    if randint(0,10) == 1: makeRock((215,480))
    if randint(0,10) == 1: makeRock((540,480))
    shieldsUp = False
    for r in range(0, len(rocks)):
        if rocks[r].y < 800:
            rocks[r].y -= rocks[r].speed
            rocks[r].x += rocks[r].dir
            rocks[r].speed -= 0.2
            if jet.colliderect(rocks[r]):
                shieldsUp = True
            if bullet.colliderect(rocks[r]):
                rocks[r].y = 800
                bullet.x = 850
    if shieldsUp == True:
        jet.image = “jet2”
    else:
        jet.image = “jet”

def getOldRock():
    for r in range(0, len(rocks)):
        if rocks[r].y >= 800:
            return r

pgzrun.go()
```



The *Gradius* volcanoes spew rocks at the player just before the end-of-level boss ship arrives.