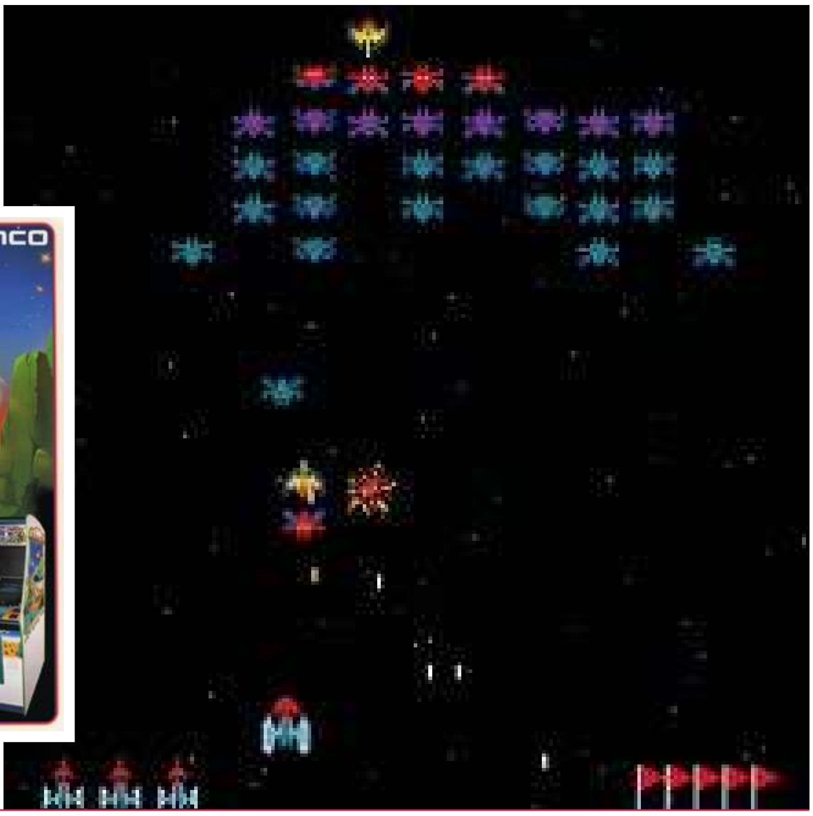


› Aliens swoop down towards the player, bombing as they go. Back in 1979, this was a big step forward from Taito's *Space Invaders*.

Source Code

Recreate Galaxian's iconic attack patterns

AUTHOR
MARK VANSTONE

Blast dive-bombing aliens in our salute to Namco's classic

Hot on the heels of the original *Space Invaders*, *Galaxian* emerged as a rival space shooter in 1979. Released by Namco, *Galaxian* brought new colour and unpredictable motion to the alien enemy, who would swoop down on the defending player. *Galaxian* was so popular in arcades that Namco released a sequel, *Galaga*, two years later – that game complicated the attack patterns even more. It's difficult to say how many ports and clones have been made of *Galaxian*, as there are several versions of similar games for almost every home platform.

The player's role in *Galaxian* is similar to *Space Invaders*, in that they pilot a ship and need to destroy a fleet of aliens. With *Galaxian*, however, the aliens have a habit of breaking formation and swooping down towards the player's ship, and dive-bombing it. The aim is to destroy all the enemy ships and move on to the next wave. The subsequent waves of enemies get more difficult as the player progresses. For this sample, we're going to look at that swooping mechanic, and make the bare nuts and bolts of a *Galaxian* game with Pygame Zero.

First, *Galaxian* has a portrait display, so we can set the play area's width and height to be 600 and 800 respectively. Next, we can create a scrolling backdrop of stars using a bitmap that we blit to the screen and move downwards every update. We need a second blit of the stars to fill in the space that the first one leaves as it scrolls down, and we could also have another static background image behind them, which will provide a sense of depth.

Next, we set up the player ship as an Actor, and we'll capture the left and right arrow keys in the `update()` function to move the ship left and right on the screen. We can also fire off a bullet with the **SPACE** bar, which will travel up the screen until it hits an alien or goes off the top of the screen. As in the original *Galaxian*, you can only shoot one bullet at a time, so we only need one Actor for this.

The aliens are arranged in rows and move left and right across the screen together. We'll stick to just one type of alien for this sample, but draw two rows of them. You could add extra types and any number of rows. When we create the alien Actors, we can also add a status flag, and we need to determine which side of the row they're on as when they break formation, the two sides fly in opposite directions. In this case, there'll be four aliens on the left of each row and four on the right. Once they're set up in a list, we can iterate through the list on each update and move them backwards and forwards. While we're moving our aliens, we can also check to see if they've collided with a bullet or the player ship. If the collision is with a bullet, the alien cycles through a few frames of an explosion using the status flag, and then, when their status reaches five, they're no longer drawn. If the collision is with the player, then the player dies and the game's over. We can also check a random number to see if the alien will start a bombing run; if so, we set the status to one, which will start calls to the `flyAlien()` function. This function checks which side the alien's on and starts changing the alien's angle, depending on the side. It also alters the x and y coordinates, depending on the angle. We've written this section in longhand for clarity, but this could be collapsed down a bit with the use of some multiplier variables for the x coordinates and the angles.

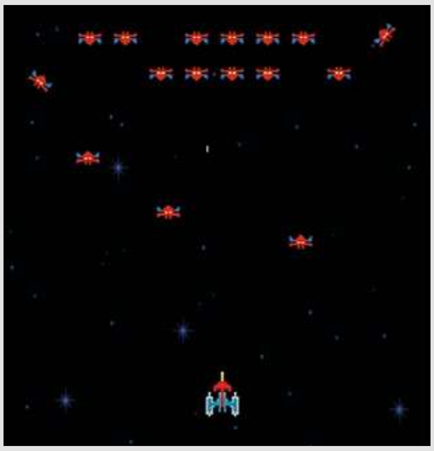
There we have it: the basics of *Galaxian*. Can you flesh it out into a full game?

Download the code from GitHub: **wfmag.cc/wfmag50**

Massive attack

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

```
# Galaxian
from random import randint
WIDTH = 600
HEIGHT = 800

bullet = Actor(‘bullet’, center=(0, -10))
ship = Actor(‘ship’, center=(300, 700))
backY = count = gameover = 0
aliens = []
for a in range(0, 8):
    aliens.append(Actor(‘alien0’, center=(200+(a*50),200)))
    aliens[a].status = 0
    aliens[a].side = int(a/4)

for a in range(0, 8):
    aliens.append(Actor(‘alien0’, center=(200+(a*50),250)))
    aliens[a+8].status = 0
    aliens[a+8].side = int(a/4)

def draw():
    screen.blit(“background”, (0, 0))
    screen.blit(“stars”, (0, backY))
    screen.blit(“stars”, (0, backY-800))
    bullet.draw()
    drawAliens()
    if gameover != 1 or (gameover == 1 and count%2 == 0): ship.
draw()

def update():
    global backY, count
    count += 1
    if gameover == 0:
        backY += 0.2
        if backY > 800: backY = 0
        if bullet.y > -10: bullet.y -= 5
        if keyboard.left and ship.x > 50 : ship.x -= 4
        if keyboard.right and ship.x < 550 : ship.x += 4
        if keyboard.space :
            if bullet.y < 0: bullet.pos = (ship.x,700)
        updateAliens()

def drawAliens():
    for a in range(0, 16):
        if aliens[a].status < 5 : aliens[a].draw();

def updateAliens():
    global gameover
    for a in range(0, 16):
        aliens[a].image = “alien0”
        if count%30 < 15 : aliens[a].image = “alien1”
        if count%750 < 375:
            aliens[a].x -=0.4
        else:
            aliens[a].x +=0.4
        if aliens[a].collidepoint(bullet.pos) and aliens[a].
status < 2:
            aliens[a].status = 2
            bullet.y = -10
        if aliens[a].colliderect(ship) : gameover = 1
        if randint(0,1000) == 1 and aliens[a].status == 0 :
aliens[a].status = 1
        if aliens[a].status == 1 : flyAlien(a)
        if aliens[a].status > 1 and aliens[a].status < 5:
            aliens[a].image = “alien” + str(aliens[a].status)
            aliens[a].status += 1

def flyAlien(a):
    if aliens[a].side == 0:
        if aliens[a].angle < 180 :
            aliens[a].angle += 2
            aliens[a].x -= 1
            if aliens[a].angle < 90: aliens[a].y -= 1
        if aliens[a].angle >= 90 :
            aliens[a].y += 2
        if aliens[a].angle >= 180 :
            aliens[a].angle = 180
            aliens[a].x += 1
    else:
        if aliens[a].angle > -180 :
            aliens[a].angle -= 2
            aliens[a].x += 1
            if aliens[a].angle > -90: aliens[a].y -= 1
        if aliens[a].angle <= -90 :
            aliens[a].y += 2
        if aliens[a].angle <= -180 :
            aliens[a].angle = -180
            aliens[a].x -= 1
```



Our homage to the classic *Galaxian*, with angry aliens that love to break formation.