Source Code

Galaxian's dive-bombing aliens

Namco's Galaxian took 2D shooting games to new heights thanks to its dive-bombing aliens

AUTHOR
RYAN LAMBIE

Released by Taito in 1978 and designed by Tomohiro Nishikado, *Space Invaders* earned millions of dollars at a point when the games industry was still in its infancy. In its wake, rival manufacturers scrambled to make shooting games of their own, almost all of them sticking rigidly to the *Space Invaders* template: the player moves left and right, shooting a phalanx of aliens slowly advancing from the top of the screen. Even the names hewed cheekily close to Taito's branding: late seventies arcades were packed with games like *Beam Invader*, *Space Attack*, and *Cosmic Monsters*.

Galaxian, on the other hand, represented a more concerted effort to expand the *Space Invaders* rulebook. Designed by Kazunori Sawano and Koichi Tashiro, 1979's *Galaxian* was Namco's aggressive response to Taito's invasion juggernaut: unlike the monochrome *Space Invaders*, *Galaxian* offered full-colour sprites, while the action took place against a shimmering, scrolling starfield that created a feeling of movement.

What really set *Galaxian* apart, though, was its enemy AI. Whereas *Space Invaders*' aliens simply sat passively at the top of the screen, waiting to be shot down, *Galaxian*'s insect-like invaders routinely peeled off from their main formation

"*Galaxian*'s invaders peeled off from their main formation to swoop down and attack the player's ship"

to swoop down and attack the player's ship. This dive-bombing motion not only made the game more hectic, as the player swerved to avoid both bullets and falling invaders, but it also created the illusion of a swarming intelligence.

BEHIND THE SCENES

In reality, the programming that drives the aliens' AI is quite simple. Aliens always peel off from the main formation in the same way, performing a half-turn to the left or right before beginning their descent. If we were to recreate the start of an alien's dive-bomb in Python, it would look something like the code on the right: we can see how the alien ship spends half a second performing a half-loop, and then begins to move downwards in a sinusoidal (side-to-side) motion.

Although there's more to *Galaxian*'s alien AI than this one movement – the aliens can also descend in groups of three, and rejoin their formation if they reach the bottom of the screen, for example – the invaders nevertheless follow this same wave-like pattern throughout the game. But with so many aliens diving from different positions, *Galaxian* creates the impression of an attacking alien force – and presents the player with a highly challenging opponent.

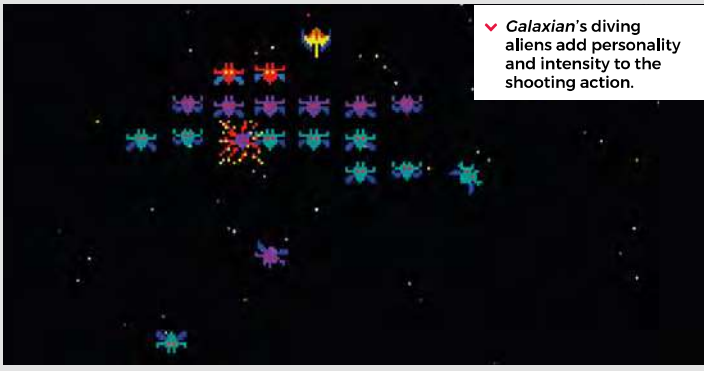
Galaxian was an early hit for Namco, and set the pace for its later arcade hits. *Pac-Man* and *Galaga*, released in 1980 and 1981 respectively, both featured colourful graphics and, more importantly, enemy AI filled with aggressive personality.

Download the code from GitHub: wfmag.cc/wfmag2

DIVE-BOMBING ALIENS in PYTHON

Here's a simplified look at how a dive-bombing alien routine would work in Python, courtesy of Daniel Pope. To get it running on your system, you'll first need to install Pygame Zero – you can find full instructions at **wfmag.cc/XVIIeD**

```
import math
WIDTH = 400
HEIGHT = 800
# How many wobbles the ship does while diving
DIVE_WOBBLE_SPEED = 2
# How far the ship wobbles while diving
DIVE_WOBBLE_AMOUNT = 100
def dive_path(t):
    """Get the ship's position at time t when diving.
    This is relative to the ship's original position (so, at
    t=0, dive_path(t)
    returns (0, 0)). Here we flip to the right before diving.
    """
    if t < 0.5:
        # During the first 0.5s, do a half-loop to the right
        return (
            50 * (1 - math.cos(2 * t * math.pi)),
            -50 * (math.sin(2 * t * math.pi))
        )
    # For the rest of the time, follow a sinusoidal path
    downwards
    t -= 0.5
    return (
        DIVE_WOBBLE_AMOUNT * math.cos(t * DIVE_WOBBLE_SPEED),
        t * 350,
    )
def make_individual_dive(start_pos, flip_x=False):
    """Return a function that will return a dive path from
    start_pos.
    If flip_x is True then the path will be flipped in the x
    direction.
    """
    dir = -1 if flip_x else 1
    sx, sy = start_pos
    def _dive_path(t):
        x, y = dive_path(t)
        return sx + x * dir, sy + y
    return _dive_path
def ship_controller_pan(ship, dt):
    """Update the ship when the ship is panning."""
    ship.x += ship.vx * dt
    if ship.right > WIDTH - 50:
        ship.vx = -ship.vx
        ship.right = WIDTH - 50
    elif ship.left < 50:
        ship.vx = -ship.vx
        ship.left = 50
def ship_controller_dive(ship, dt):
    """Update the ship when the ship is diving."""
    # Move the ship along the path
    ship.t += dt
    ship.pos = ship.dive_path(ship.t)
    # Look ahead along the path to see what direction the ship
    # is moving, and set the ship's rotation accordingly
    ship.angle = ship.angle_to(ship.dive_path(ship.t + EPSILON))
    # If we've reached the bottom of the screen, resume dive
    mode
    if ship.top > HEIGHT:
        ship.controller_function = ship_controller_pan
        ship.pos = ship.dive_path(0)
        ship.angle = 90
        clock.schedule(start_dive, 3)
EPSILON = 0.001
# Create an Actor using the 'ship' sprite
ship = Actor('ship', pos=(100, 100), angle=90)
ship.angle = 90  # Face upwards
ship.controller_function = ship_controller_pan
ship.vx = 100
def draw():
    """Just draw the actor on the screen."""
    screen.clear()
    ship.draw()
def update(dt):
    """Update the ship using its current controller function."""
    ship.controller_function(ship, dt)
def start_dive():
    """Make the ship start a dive."""
    # flip the dive if we're on the left of the screen
    flip_x = ship.x < WIDTH // 2
    ship.controller_function = ship_controller_dive
    ship.dive_path = make_individual_dive(ship.pos, flip_
x=flip_x)
    ship.t = 0
clock.schedule(start_dive, 3)
```



Galaxian's diving aliens add personality and intensity to the shooting action.