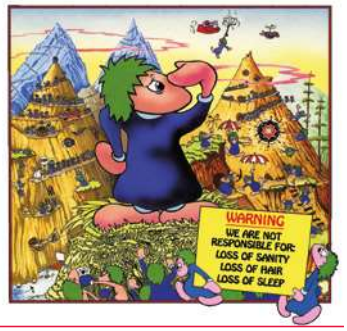
The original *Lemmings*, first released for the Amiga, quickly spread like a virus to just about every computer and console of the day.



Source Code

Path-following Lemmings

Learn how to create your own obedient lemmings that follow any path put in front of them

AUTHOR
RIK CROSS

Lemmings is a puzzle-platformer, created at DMA Design, and first became available for the Amiga in 1991. The aim is to guide a number of small lemming sprites to safety, navigating traps and difficult terrain along the way. Left to their own devices, the lemmings will simply follow the path in front of them, but additional 'special powers' given to lemmings allow them to (among other things) dig, climb, build, and block in order to create a path to freedom (or to the next level, anyway).

I'll show you a simple way (using Python and Pygame) in which lemmings can be made to follow the terrain in front of them. The first step is to store the level's terrain information, which I've achieved by using a two-dimensional list to store the colour of each pixel in the background 'level' image. In my example, I've used the 'Lemcraft' tileset by Matt Hackett (of Lost Decade Games) – taken from **opengameart.org** – and used the 'Tiled' software (**mapeditor.org**) to stitch the tiles together into a level.

The algorithm we then use can be summarised as follows: check the pixels immediately below a lemming. If the colour of those pixels isn't the same as the background colour, then the lemming is falling.

> **"Left to their own devices, the lemmings will follow the path in front of them"**

In this case, move the lemming down by one pixel on the y-axis. If the lemming isn't falling, then it's walking. In this case, we need to see whether there is a non-ground, background-coloured pixel in front of the lemming for it to move onto. If a pixel is found in front of the lemming (determined by its direction) that is low enough to get to (i.e. lower than its `climbheight`), then the lemming moves forward on the x-axis by one pixel, and upwards on the y-axis to the new ground level. However, if no suitable ground is found to move onto, then the lemming reverses its direction.

The above algorithm is stored as a `lemming`'s `update()` method, which is executed for each lemming, each frame of the game. The sample **level.png** file can be edited, or swapped for another image altogether. If using a different image, just remember to update the level's `BACKGROUND_COLOUR` in your code, stored as a (red, green, blue, alpha) tuple. You may also need to increase your lemming `climbheight` if you want them to be able to navigate a climb of more than four pixels.

There are other things you can do to make a full *Lemmings* clone. You could try replacing the yellow-rectangle lemmings in my example with pixel-art sprites with their own walk cycle animation (see my article in issue #14) or give your lemmings some of the special powers they'll need to get to safety, achieved by creating flags that determine how lemmings interact with the terrain around them.

Path-following critters in Python

Here's a code snippet that will send path-following creatures roaming around your screen. To get it running, you'll first need to install Pygame Zero – you can find full instructions at **wfmag.cc/pgzero**

Download the code from GitHub: **wfmag.cc/wfmag17**

```
from time import sleep
from PIL import Image

# screen size
HEIGHT=800
WIDTH=800

# level information
level_image = 'level'
BACKGROUND_COLOUR = (114,114,201,255)

# store the colour of each pixel in the 'level' image
img = Image.open('images/level.png')
pixels = [[img.getpixel((x, y)) for y in range(HEIGHT)] for x
in range(WIDTH)]

# a list to keep track of the lemmings
lemmings = []
max_lemmings = 10
start_position = (100,100)
# a timer and interval for creating new lemmings
timer = 0
interval = 10

# returns 'True' if the pixel specified is 'ground'
# (i.e. anything except BACKGROUND_COLOUR)
def groundatposition(pos):
    # ensure position contains integer values
    pos = (int(pos[0]),int(pos[1]))
    # get the colour from the 'pixels' list
    if pixels[pos[0]][pos[1]] != BACKGROUND_COLOUR:
        return True
    else:
        return False

class Lemming(Actor):
    def __init__(self, **kwargs):
super().__init__(image='lemming', pos=start_position,
anchor=('left','top'), **kwargs)
        self.direction = 1
        self.climbheight = 4
        self.width = 10
        self.height = 20
    # update a lemming's position in the level
    def update(self):
        # if there's no ground below a lemming (check both
corners), it is falling
        bottomleft = groundatposition((self.pos[0],self.
pos[1]+self.height))
        bottomright = groundatposition((self.pos[0]+(self.
width-1),  self.pos[1]+self.height))
        if not bottomleft and not bottomright:
            self.y += 1
        # if not falling, a lemming is walking
        else:
            height = 0
            found = False
            # find the height of the ground in front of a
            # lemming up to the maximum height a lemming
            # can climb
            while (found == False) and (height <= self.
climbheight):
                # the pixel 'in front' of a lemming
                # will depend on the direction it's
                # traveling
                if self.direction == 1:
                    positioninfront = (self.pos[0]+self.width,
self.pos[1]+(self.height-1)-height)
                else:
                    positioninfront = (self.pos[0]-1, self.
pos[1]+(self.height-1)-height)
                if not groundatposition(positioninfront):
                    self.x += self.direction
                    # rise up to new ground level
                    self.y -= height
                    found = True

                height += 1
            # turn the lemming around if the ground in front
            # is too high to climb
            if not found:
                    self.direction *= -1

def update():
    global timer
    # increment the timer and create a new
    # lemming if the interval has passed
    timer += 0.1
    if timer > interval and len(lemmings) < max_lemmings:
        timer = 0
        lemmings.append(Lemming())
    # update each lemming's
position in the level
    for i in lemmings:
        i.update()

def draw():
    screen.clear()
    # draw the level
    screen.blit(level_image,(0,0))
    # draw lemmings
    for i in lemmings:
        i.draw()
```

Sprites cling to the ground below them, navigating uneven terrain, and reversing direction when they hit an impassable obstacle.