Source Code

Code your own landscape engine

Recreate The Lords of Midnight's pioneering graphics technique

AUTHOR
MARK VANSTONE

Due to the limitations of 1980s computers, the amount of graphics a typical adventure game could actually display was limited – at least, until *The Lords of Midnight* came along. In it, players were treated to views of trees, mountains, and buildings from any viewpoint on what at the time felt like a huge map. The term used for this technique was 'landscaping', which provided a pseudo-3D view for the player as they travelled around the Land of Midnight.

The Lords of Midnight was written by Mike Singleton in 1984, first for the ZX Spectrum. The adventure sees a group of characters set off to complete two tasks: destroy the Ice Crown or battle the evil Doomdark by storming his citadel in the north. The game was a success and was quickly followed by a sequel, *Doomdark's Revenge*.

The technique to create the views around the landscape uses 'billboarding' to translate a 2D overhead view of a map into a pseudo-3D view of eight points on the compass. Billboarding refers to the idea that each object in the scene always faces towards the camera and is a 2D image but is positioned and scaled to appear 3D.

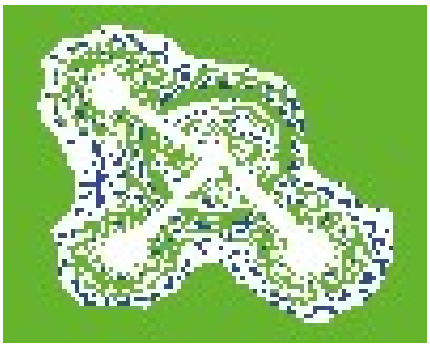
Our example of the landscaping technique Mike Singleton pioneered in *The Lords of Midnight*.

To create this effect, we start at the back of the scene on the horizon, drawing the objects that appear the furthest away, and then move towards the camera position drawing the objects that are closer and incrementally larger.

To write this in Pygame Zero, we'll need a little help from the Pygame library so that we can read our map pixels from an image and scale the images we're using for the billboarding. With a map image of 100×100 pixels, we'll start our player position at the X and Y coordinates 50,50. We can define our compass direction in a list of tuples, including the X direction, the Y direction, and a string describing the direction we're pointing in. After loading our image map and the images we'll use as billboards (in this case, we will have a tree, a mountain, and a tower), we can write a `draw()` function. This will draw a background of an empty landscape, then call a function to draw the landscape, and finally print a message telling the player which direction they're facing. Because this is a fantasy adventure game, we'll use a fancy font for our messages.

Unusually, we don't need to do anything with the `update()` function in our code example, but we do need to set up some key presses for rotate left, right, and move forward. The left and right keys rotate the `playerdir` variable by 45 degrees, and the up key moves the player forward by one pixel on our map.

Now comes the interesting part. We need to look along the map in the direction the player's facing and start reading nine pixels away from the player. We need to read a row of pixels that will cover the horizon and then move toward the player position, reading fewer pixels each row in a sort of wedge shape, so if our map was all trees (green pixels), we would want to see rows of trees extending to the horizon whichever way we look. The way we create this illusion is to start by scaling the trees to be very small, and as we read towards the player position, the trees are scaled larger, creating a perspective effect.



The map, from which our routine takes its landmarks – you could try adding extra elements of your own to create a full game.

The routine to read the map and translate that into trees and mountains is a fairly straightforward embedded loop, reading rows and columns of the map image, but only if we're facing north. If we're facing any other direction, we need to change everything to read in a different direction. On the face of it, this might sound complicated, but have no fear: if we rotate the map to point in the direction the player's facing, we can use exactly the same reading routine. We do need to keep track of where the player is on the map and use that as our starting point once we have rotated our map, however.

So that's the concept of how to translate a 2D map into a 3D view using 'landscaping'. Now all that's required is to write an adventure for our players to embark on. We'll leave that bit to you as it'll require a heck of a lot more work!

Loads of Midnight

Here's Mark's code, which will draw verdant landscapes wherever you look. To get it working on your system, you'll first need to install Pygame Zero. Full instructions can be found at **wfmag.cc/pgzero**.

Download the code from GitHub: **wfmag.cc/wfmag57**

```
# The Lords of Midnight
import pgzrun
from pygame import transform, image, Color

playerx = 50
playery = 50
playerdir = 0
myDirs = [(0,1, "North"),(-1,1, "North East"),(-1,0, "East"),(-1,-1,
"South East"),(0,-1, "South"),(1,-1, "South West"),(1,0, "West"),(1,1,
"North West")]
landscape = image.load('images/landscape.png')
tree = image.load('images/tree1.png')
mountain = image.load('images/mountain1.png')
tower = image.load('images/tower1.png')

def draw():
    screen.blit("background", (0, 0))
    drawLandscape()
    d = int(playerdir/45)
    if d > 7: d -= 8
    screen.draw.text("You are facing "+myDirs[d][2], center = (400,
50), owidth=0.5, ocolor=(255,255,255), color=(0,0,0) , fontsize=60,
fontname="blackchancery" )

def update():
    pass

def on_key_down(key):
    global playerdir
    if key.name == "RIGHT":
        playerdir += 45
        if playerdir > 360: playerdir -= 360
    if key.name == "LEFT":
        playerdir -= 45
        if playerdir < 0: playerdir += 360
    if key.name == "UP":
        movePlayer()

def drawLandscape():
    global gameStatus
    rotatedLand = rotatedLandscape()
    playerpos = getPlayerPos(rotatedLand)
    x = playerpos[0]
    y = playerpos[1]
    for r in range(9,0,-1):
        for c in range(-5*int(r/2),5*int(r/2),1):
            pixel = rotatedLand.get_at((x-c,y-r))
            s = r*10
            d = (10-r)*20
            if pixel == Color('blue'):
                i = transform.scale(mountain, ((10-r)*50, (10-r)*40))
                screen.blit(i,(200+s-(d*c),180+(r*5)))
            if pixel == Color('green'):
                i = transform.scale(tree, ((10-r)*20, (10-r)*20))
                screen.blit(i,(290+s-(d*c),310-(r*8)))
            if pixel == Color('red'):
                i = transform.scale(tower, ((10-r)*20, (10-r)*20))
                screen.blit(i,(290+s-(d*c),310-(r*8)))

def rotatedLandscape():
    land = landscape.copy()
    land.set_at((playerx,playery),Color("black"))
    land.set_at((playerx+1,playery),Color("black"))
    rotated_image = transform.rotate(land, playerdir)
    return rotated_image

def getPlayerPos(i):
    s = i.get_size()
    for x in range(s[0]):
        for y in range(s[1]):
            if i.get_at((x,y)) == Color("black"): return (x,y)

def movePlayer():
    global playerx, playery
    d = int(playerdir/45)
    if d > 7: d -= 8
    if playerx - myDirs[d][0] > 25 and playerx - myDirs[d][0] < 75 and
playery - myDirs[d][1] > 25 and playery - myDirs[d][1] <75:
        playerx -= myDirs[d][0]
        playery -= myDirs[d][1]

pgzrun.go()
```

The Lords of Midnight was first released for the ZX Spectrum in 1984. Its wealth of views caused quite a sensation at the time.