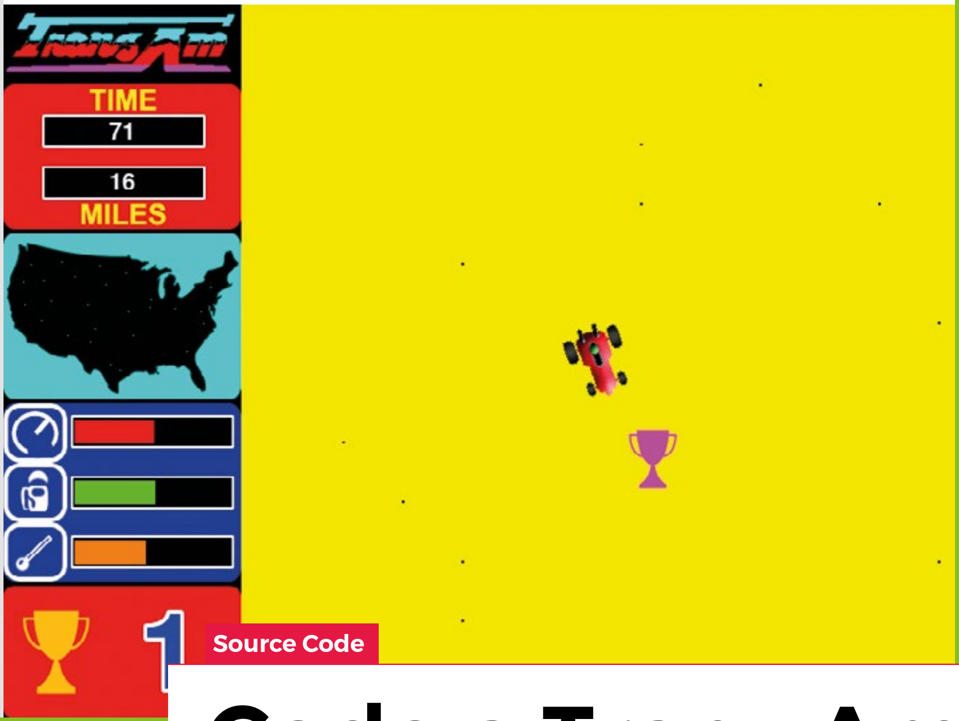
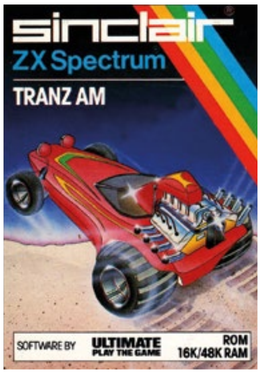


- Speed around the future USA trying not to run out of petrol.
- *Tranz Am* was available on cassette and also on cartridge format ROM.



Source Code

Code a Tranz Am-style top-down racer

AUTHOR
MARK VANSTONE

Race around a post-apocalyptic US in our homage to an Ultimate classic

Tranz Am zoomed onto the ZX Spectrum in 1983. Published by Ultimate Play The Game, it saw the player drive around a future version of the United States in search of eight cups. Along the way, the player needed to visit petrol pumps to avoid running out of fuel, and just to make things more difficult, several other cars were racing around, trying to crash into our motor. There were plenty of other obstacles to avoid, too, which made driving at full speed a bad idea.

On the left of the screen, a mini-map of the United States could be found, along with counters showing time elapsed and miles travelled. A speedo, plus fuel and temperature gauges, were also on display so that you could keep an eye on your car's status. Like most of Ultimate's games, *Tranz Am* was well-received in 1983, and was one of the studio's more arcade-like titles, before it began its cycle of isometric adventures with 1984's *Knight Lore*.

To recreate *Tranz Am* in Pygame Zero, we need a system that draws a mini-map, which then translates to a much larger map we can drive the car around on. In this case, the car stays in the centre of the play area and the map moves around it, depending on the speed and direction of the vehicle. For our example, we're going to have the main map 50 times the size of the mini-map, so one pixel on the mini-map will translate to 50 pixels on the main map. To do this, we'll make an image that will double as our collision and feature map, which will be the same size as the mini-map in the side panel. The main map features dots on the ground which show that it is moving if there are no other features in view, and these dots can be defined on our map image as random noise. We represent our boundary as white pixels, our petrol pumps as green pixels, and our cups as red pixels.

If we control the car's rotation using the left and right arrow keys, we'll need a bit of maths to calculate the movement of the map. We need to convert the angle of our car Actor to radians first, and then we can use sin and cos calculations to work out x and y increments based on the car speed (altered by the up and down arrow keys). As the map coordinates move, we redraw the map in a 50×50 pixel grid based on the pixels we read from the collision/features image (**noisemap.png**). We need to detect if the car is over the top of a fuel pump, which we can find if the centre pixel we're reading for our main map is green. If the car's over a petrol pump, we increase the fuel level; if it's not, we decrease the fuel level. If the car's over a cup, we change the colour of the pixel to black and add one to our cups tally. The last thing to check is whether the car's over a boundary; if it is, the car will turn around 180 degrees.

We need to draw bars on the speedo and fuel and temperature gauges to show their levels and keep a count of time and number of miles travelled. We also need to display the number of cups collected in the left-hand panel. When the temperature goes too high, the speed is reduced until the temperature returns to a suitable level. With all that in the program, you have quite a playable *Tranz Am* clone. The only things you need to add are a few more obstacles and those pesky computer-controlled cars that try to crash into you. As always, we'll leave those for you to add in.

Tranz Am in Python

Download the code from GitHub: **wfmag.cc/wfmag55**

Here's Mark's code for a *Tranz Am*-esque racer. To get it working on your machine, you'll first need to install Pygame Zero. You can find full instructions at **wfmag.cc/pgzero**.

```
import pgzrun
from pygame import image, Color
import math

car = Actor('car', center=(500, 300))
car.angle = 180
car.speed = 0
car.fuel = 130
car.temp = 60
mapx = 100*50
mapy = 70*50
cups = 0
miles = 0
count = gameStatus = 0
noisemap = image.load('images/noisemap.png')

def draw():
    drawMainMap()
    car.draw()
    screen.blit("sidepanel",(0,0))
    drawMiniMap()
    screen.draw.filled_rect(Rect((60, 400), (car.fuel,
20)),(0,255,0))
    screen.draw.filled_rect(Rect((60, 349), (car.speed*26,
20)),(255,0,0))
    screen.draw.filled_rect(Rect((60, 450), (car.temp,
20)),(255,128,0))
    screen.draw.text(str(cups), center = (140, 548), owidth=0.5,
ocolor=(255,255,255), color=(0,0,255) , fontsize=130)
    screen.draw.text(str(int(count/50)), center = (100, 108),
color=(255,255,255) , fontsize=30)
    screen.draw.text(str(int(miles)), center = (100, 150),
color=(255,255,255) , fontsize=30)
    if gameStatus == 1 : screen.draw.text("YOU GOT ALL THE
CUPS!", center = (400, 300), owidth=0.5, ocolor=(255,255,255),
color=(0,0,255) , fontsize=80)

def update():
    global mapx,mapy,miles, count,gameStatus
    if gameStatus == 0 :
        checkInput()
        if car.fuel > 0:
            mapx += -car.speed * math.sin(math.radians(car.angle))
            mapy += -car.speed * math.cos(math.radians(car.angle))
            car.fuel -= car.speed/100
            car.temp = limit(car.temp+((car.speed-3)/100),60,130)
            if car.temp > 120:
                car.speed -= 0.1
            miles += car.speed/500
        flagCount = 0
        count += 1

def checkInput():
    if keyboard.left: car.angle = (car.angle + 5)%360
    if keyboard.right: car.angle = (car.angle - 5)%360
    if keyboard.up: car.speed = limit(car.speed + 0.1, 0, 5)
    if keyboard.down: car.speed = limit(car.speed - 0.1, 0, 5)

def drawMainMap():
    global cups, gameStatus
    screen.draw.filled_rect(Rect((200, 0), (600, 600)),(255,255,0))
    xoff = mapx%50
    yoff = mapy%50
    for x in range(-1,13):
        for y in range(-1,13):
            pixel = noisemap.get_at((int((mapx/50)+x),
int(mapy/50)+y))
            if pixel == Color('white'):
                screen.blit("boundary",(200+(x*50)-xoff,(y*50)-
yoff))
                if x == 6 and y == 6: car.angle = (car.angle +
180)%360
            elif pixel == Color('green'):
                if x == 6 and y == 6: car.fuel = limit(car.
fuel+1,0,130)
                screen.blit("fuel",(200+(x*50)-xoff,(y*50)-yoff))
            elif pixel == Color('red'):
                if x == 6 and y == 6:
                    noisemap.set_at((int((mapx/50)+x),
int(mapy/50)+y), Color('black'))
                    cups += 1
                    if cups == 8: gameStatus = 1
                else: screen.blit("cup",(200+(x*50)-xoff,(y*50)-
yoff))
            elif pixel.b > 200: screen.blit("dot",(200+(x*50)-
xoff,(y*50)-yoff))

def drawMiniMap():
    carRect = Rect((5+(mapx/50),198+(mapy/50)),(4,4))
    if count%10 > 5: screen.draw.filled_rect(carRect,(0,0,0))
    else: screen.draw.filled_rect(carRect,(100,100,100))

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

pgzrun.go()
```