CG Mini-Project Report.
Topic - **PACMAN**

MEMBERS:

YASH SARANG
OM GAYDHANE
MANAV PAHILWANI
SURABHI TAMBE

Table of contents:

Abstract

The following report consists of a detailed description of our CG Mini Project. We have created a classic arcade game - ‘Pacman’ in which we have implemented Computer Graphics taught in our current Semester as well as basic gaming algorithms. To make it a bit user-friendly and less complex, we used a high-level coding language - Python over a low leveled language like C.

Introduction

We started by importing the turtle package in python which would help us in implementing CG in our code. Following by turtle, we imported the choice module from random and the vector and floor modules from freegames library.

The start of the code represents the initial state of the game/game objects. Then we have multiple user-defined functions

1. square( ) - which fills the tile with the x and y coordinates.
2. offset( ) - which returns the next index in which the pacman / ghosts will be moving.
3. valid( ) - which returns True if the next tile is valid for movement.
4. world() - used for Drawing the world, the tiles, and the score points.
5. move( ) - used For controlling the movement of all the pacman and ghosts.
6. change() - change the direction vector of pacman if its valid.

Then we use the setup( ) function from turtle to set up the GUI window for our game.

The write( ) function is used to display the score in which state['score'] is used to input the current score of the user.

And at last, the listen( ) function is used to read the inputs of the user and the code acts according to the input.

The game ends when the Pacman gets hit by a ghost, as soon as the Pacman hits a ghost the move( ) function stops, and hence, the game is over.

Implementation (Code Snippets)

Initialization

```
from random import choice
from turtle import *

from freegames import floor, vector


state = {'score': 0}

path = Turtle(visible=False)
writer = Turtle(visible=False)

aim = vector(5, 0)
pacman = vector(-40, -80)
ghosts = [
    [vector(-180, 160), vector(5, 0)],
    [vector(-180, -160), vector(0, 5)],
    [vector(100, 160), vector(0, -5)],
    [vector(100, -160), vector(-5, 0)],
]


tiles = [
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0,
    0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0,
    0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0,
    0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0,
    0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0,
    0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
    0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0,
    0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0,
    0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0,
    0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0,
    0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0,
    0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
    0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0,
    0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0,
    0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0,
    0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
    0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
]
```

World creating function

```
def square(x, y):
    path.up()
    path.goto(x, y)
    path.down()
    path.begin_fill()

    for count in range(4):
        path.forward(20)
        path.left(90)

    path.end_fill()


def offset(point):
    x = (floor(point.x, 20) + 200) / 20
    y = (180 - floor(point.y, 20)) / 20
    index = int(x + y * 20)
    return index

def valid(point):
    index = offset(point)

    if tiles[index] == 0:
        return False

    index = offset(point + 19)

    if tiles[index] == 0:
        return False

    return point.x % 20 == 0 or point.y % 20 == 0


def world():
    bgcolor('cyan')
    path.color('blue')

    for index in range(len(tiles)):
        tile = tiles[index]

        if tile > 0:
            x = (index % 20) * 20 - 200
            y = 180 - (index // 20) * 20
            square(x, y)

            if tile == 1:
                path.up()
                path.goto(x + 10, y + 10)
                path.dot(2, 'white')
```

Movement function

```
def move():
    writer.undo()
    writer.write(state['score'])

    clear()

    if valid(pacman + aim):
        pacman.move(aim)

    index = offset(pacman)

    if tiles[index] == 1:
        tiles[index] = 2
        state['score'] += 1
        x = (index % 20) * 20 - 200
        y = 180 - (index // 20) * 20
        square(x, y)

    up()
    goto(pacman.x + 10, pacman.y + 10)
    dot(20, 'yellow')

    for point, course in ghosts:
        if valid(point + course):
            point.move(course)
        else:
            options = [
                vector(5, 0),
                vector(-5, 0),
                vector(0, 5),
                vector(0, -5),
            ]
            plan = choice(options)
            course.x = plan.x
            course.y = plan.y

        up()
        goto(point.x + 10, point.y + 10)
        dot(20, 'red')

    update()

    for point, course in ghosts:
        if abs(pacman - point) < 20:
            return

    ontimer(move, 100)
```

Setting up the GUI window and Recieving user input

```
156  setup(600, 600, 500, 120)
157  hideturtle()
158  tracer(False)
159  writer.goto(160, 160)
160  writer.color('black')
161  writer.write(state['score'])
162  listen()
163  onkey(lambda: change(5, 0), 'Right')
164  onkey(lambda: change(-5, 0), 'Left')
165  onkey(lambda: change(0, 5), 'Up')
166  onkey(lambda: change(0, -5), 'Down')
167  world()
168  move()
169  done()
```

Requirement analysis (s/w and h/w)

HARDWARE-

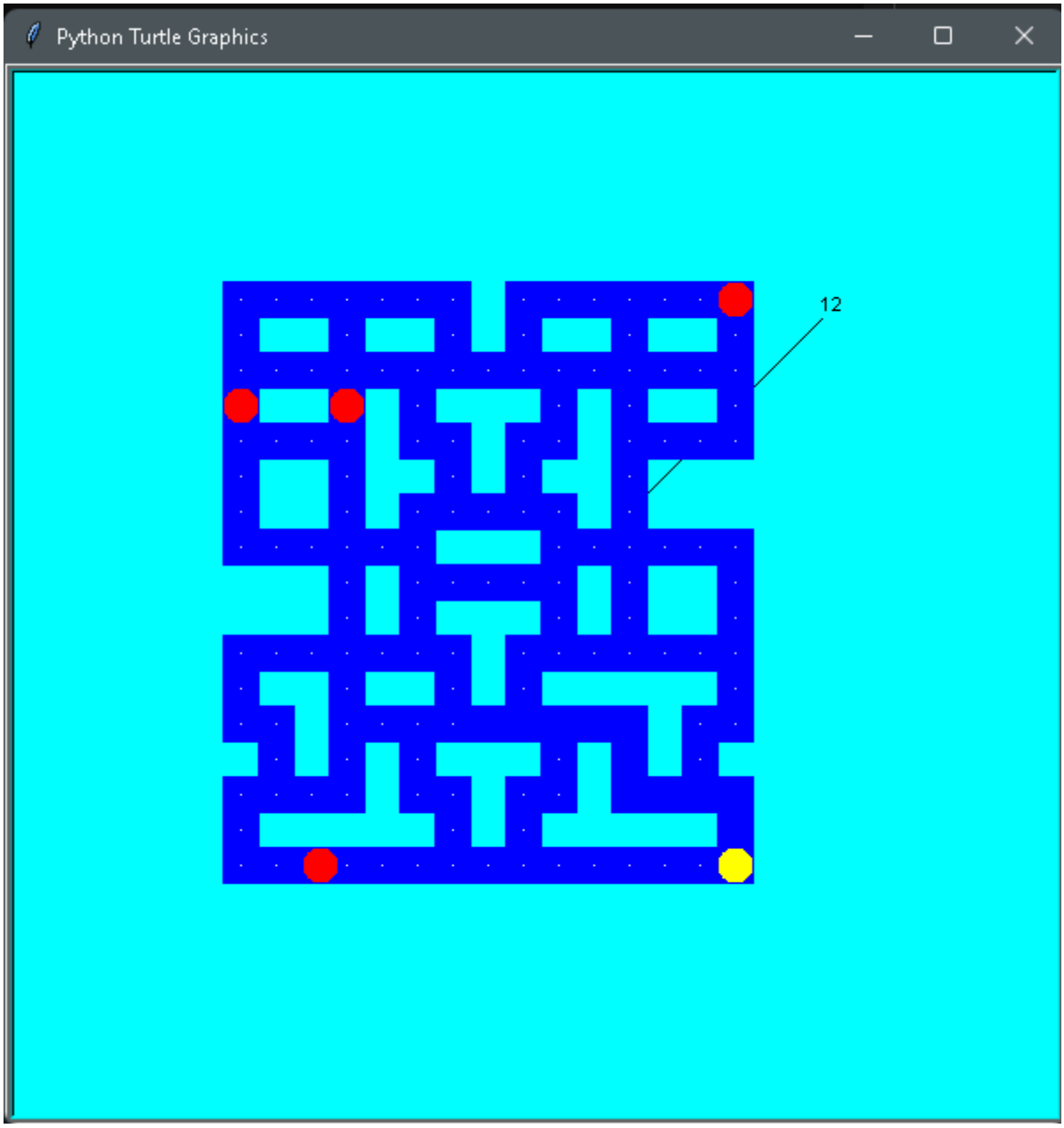
Computing Device (PC or Laptops)

SOFTWARE-

Python 2.5 or newer

(imported with pygames and free games packages)

Snapshots



Conclusion

As we have implemented CG in our PACMAN game, we can use CG in various day-to-day programs from movie making, video game development, scientific modeling, designing for catalogs, and other commercial art.
