おまけ

- この資料は授業とは関係ないので，参考程度に読んでください
- 解説はしません

Python

- 深層学習（ディープラーニング）が流行っており，深層学習のライブラリのほとんどがPythonで提供されているため，Pythonユーザが増加した
- 基本情報技術者試験のプログラミング言語は以前は「C, COBOL, Java, アセンブラ言語, 表計算ソフト」だったものが「C, Java, Python, アセンブラ言語，表計算ソフト」に変更された

OpenGL

- PythonでもOpenGLを使うことができる
- OpenGLのライブラリそのものはCで作られているが，それをPythonから呼び出せるようにしている
- Pythonはインタプリタ言語で，（オブジェクト指向言語で），動的型付け言語なので動作が遅いため，リアルタイムCGには向かないと考えるのが普通である
  - 通常の利用で気になるほどではない
  - OpenGLの中身そのものはCで作られているため，Cで実装した場合と動作速度は変わらない
- OpenGLプログラミングにPythonは適さない
  - OpenGLはC言語で利用することを前提として作られている
  - OpenGLはPythonを想定して作られたわけではないため，Pythonで実装したところでプログラミングの効率はほとんど上がらない

サンプルプログラム

- 私の作成したサンプルソースコードを載せるが，参考にしないように
  - 私はPython初心者である
  - このサンプルはかなり雑に作ってある

サンプルソースコード

```
from OpenGL.GL import *
from OpenGL.GLU import *
from OpenGL.GLUT import *
import sys
import time

STATE_GAME = 0
STATE_CLEAR = 1
STATE_END = 2
state = STATE_GAME
winw = 640
winh = 480
RADIUS = 0.5
enemy = [0.0, 0.0, RADIUS]
item = [3.0, 0.0, RADIUS]
player = [-3.0, 0.0, RADIUS]
enemy_v = [0.0, 0.1, 0.0]
delay = [50, 100, 500]
messagenum = 0
starttime = 0

def myGround():
    ground = [[0.6, 0.6, 0.6, 1.0], [0.3, 0.3, 0.3, 1.0]]

    glBegin(GL_QUADS)
    glNormal3d(0.0, 0.0, 1.0)
    for j in range(-5, 5):
        for i in range(-5, 5):
            glMaterialfv(GL_FRONT, GL_DIFFUSE, ground[abs(i + j) % 2])
            glVertex3d(i, j, 0.0)
            glVertex3d(i + 1, j, 0.0)
            glVertex3d(i + 1, j + 1, 0.0)
            glVertex3d(i, j + 1, 0.0)
    glEnd()

def myDisplayClear():
    message = 'Game Clear !'

    glClearColor(0.4, 0.6, 0.8, 1.0)
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)

    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()

    glColor3d(0.0, 0.2, 0.6)
    glRasterPos2d(0.0, 0.0)
    n = len(message)
    if messagenum < n:
        n = messagenum
    for i in range(0, n):
        glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, ord(message[i]))

    glutSwapBuffers()
```

サンプルソースコード

```
def myDisplayEnd():
    message = "Game Over !"

    glClearColor(1.0, 0.0, 0.0, 1.0)
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)

    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()

    glColor3d(0.0, 0.0, 0.0)
    glRasterPos2d(0.0, 0.0)
    n = len(message)
    if messagenum < n:
        n = messagenum
    for i in range(0, n):
        glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, ord(message[i]))

    glutSwapBuffers()

def myDisplayGame():
    glClearColor(1.0, 0.6, 0.8, 1.0)
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
    glEnable(GL_DEPTH_TEST)
    glEnable(GL_LIGHTING)
    glEnable(GL_LIGHT0)

    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    gluPerspective(60.0, winw / winh, 0.1, 100.0)
    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()
    gluLookAt(0.0, -10.0, 10.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0)

    myGround()

    glPushMatrix()
    glTranslated(enemy[0], enemy[1], enemy[2])
    glMaterialfv(GL_FRONT, GL_DIFFUSE, [0.8, 0.2, 0.2, 1.0])
    glutSolidSphere(RADIUS, 10, 10)
    glPopMatrix()

    glPushMatrix()
    glTranslated(item[0], item[1], item[2])
    glMaterialfv(GL_FRONT, GL_DIFFUSE, [0.2, 0.2, 0.8, 1.0])
    glutSolidSphere(RADIUS, 10, 10)
    glPopMatrix()

    glPushMatrix()
    glTranslated(player[0], player[1], player[2])
    glMaterialfv(GL_FRONT, GL_DIFFUSE, [0.8, 0.8, 0.8, 1.0])
    glutSolidSphere(RADIUS, 10, 10)
    glPopMatrix()

    glDisable(GL_DEPTH_TEST)
    glDisable(GL_LIGHTING)
    glDisable(GL_LIGHT0)

    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()

    glColor3d(0.6, 0.1, 0.5)
    glRasterPos2d(-0.9, -0.9)
    difftime = time.time() - starttime
    message = 'elapsed time %d' % int(difftime)
    for ch in message:
        glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, ord(ch))

    glutSwapBuffers()
```

サンプルソースコード

```
def myDisplay():
    if state == STATE_GAME:
        myDisplayGame()
    if state == STATE_CLEAR:
        myDisplayClear()
    if state == STATE_END:
        myDisplayEnd()

def myTimer(value):
    global enemy, enemy_v, state, messagenum
    R2 = 2.0 * RADIUS * 2.0 * RADIUS

    if value == 1:
        if state == STATE_GAME:
            enemy[0] += enemy_v[0]
            enemy[1] += enemy_v[1]
            enemy[2] += enemy_v[2]
            if enemy[1] > 5.0:
                enemy[1] = 5.0
                enemy_v[1] = -enemy_v[1]
            if enemy[1] < -5.0:
                enemy[1] = -5.0
                enemy_v[1] = -enemy_v[1]

            dx = player[0] - enemy[0]
            dy = player[1] - enemy[1]
            dz = player[2] - enemy[2]
            if dx * dx + dy * dy + dz * dz < R2:
                state = STATE_END
                messagenum = 0

            dx = player[0] - item[0]
            dy = player[1] - item[1]
            dz = player[2] - item[2]
            if dx * dx + dy * dy + dz * dz < R2:
                state = STATE_CLEAR
                messagenum = 0

        elif state == STATE_CLEAR:
            messagenum += 1
        elif state == STATE_END:
            messagenum += 1

        glutTimerFunc(delay[state], myTimer, 1)
        glutPostRedisplay()

def mySpecial(key, x, y):
    global player
    WALKSPEED = 0.1
    if key == GLUT_KEY_LEFT:
        player[0] -= WALKSPEED
    if key == GLUT_KEY_UP:
        player[1] += WALKSPEED
    if key == GLUT_KEY_RIGHT:
        player[0] += WALKSPEED
    if key == GLUT_KEY_DOWN:
        player[1] -= WALKSPEED
    glutPostRedisplay()

def myKeyboard(key, x, y):
    key = key.decode('utf-8')
    if key == '¥x1b':
        sys.exit()

def myReshape(width, height):
    global winw, winh
    winw = width
    winh = height
    glViewport(0, 0, winw, winh)
```

サンプルソースコード

```
def myInit():
    global starttime
    starttime = time.time()

    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE)
    glutInitWindowSize(winw, winh)
    glutInitWindowPosition(0, 0)
    glutCreateWindow(b"cg2")

def main():
    glutInit(sys.argv)
    myInit()
    glutKeyboardFunc(myKeyboard)
    glutSpecialFunc(mySpecial)
    glutTimerFunc(delay[state], myTimer, 1)
    glutReshapeFunc(myReshape)
    glutDisplayFunc(myDisplay)
    glutMainLoop()

if __name__ == '__main__':
    main()
```