計算機圖學單元介紹

一、英文主題：

Chapter 11 : scene graphs and real time

二、中文主題：

單元08 : 背景與程式

三、組別：第09組

四、組員：

B0829062 許亨宇；B0829061 蔡宇翔；B0829014 張玉歆；B0844108 陳昶至

B0829042 陳星潔；B0829040 許珈綺；B0829031 繆緹綸；B0829043 余若慈

五、功能簡述：

本單元內容為介紹如何用背景程式表現粒子移動的現象。

六、主要程式碼：

相關檔案：Ch_11_tm9_src1.cpp

```
/* Particles in a box */
#define MAX_NUM_PARTICLES 1000
#define INITIAL_NUM_PARTICLES 25
#define INITIAL_POINT_SIZE 5.0
#define INITIAL_SPEED 1.0

typedef int bool;
#define TRUE 1
#define FALSE 0

#include <stdlib.h>
#include <stdio.h>
#include <GL/glut.h>

void myDisplay();
void myIdle();
void myReshape(int, int);
void main_menu(int);
void collision(int);
float forces(int, int);
void myinit();
```

```
/* particle struct */
typedef struct particle {
  int color;        /*粒子的顏色*/
  float position[3];    /*粒子的三維座標(X, Y, Z)*/
  float velocity[3];    /*粒子的速度(dX, dY, dZ)*/
  float mass;        /*粒子的質量*/
} particle;

particle particles[MAX_NUM_PARTICLES]; /* 宣告粒子系統 */

/* 初始化粒子系統 */
int present_time;
int last_time;
int num_particles = INITIAL_NUM_PARTICLES; /* 初始化粒子數量(25) */
float point_size = INITIAL_POINT_SIZE; /* 初始化粒子大小(5.0) */
float speed = INITIAL_SPEED; /* 初始化粒子速度常數(1.0) */
bool gravity = FALSE; /* 初始化重力(關閉) */
bool elastic = FALSE; /* 初始化非完全彈性碰撞(關閉) */
bool repulsion = FALSE; /* 初始化排斥(關閉) */
float coef = 1.0; /* 初始化粒子彈性係數(1.0 = 完全彈性碰撞) */
float d2[MAX_NUM_PARTICLES][MAX_NUM_PARTICLES]; /* 粒子間碰撞速度向量 */

GLsizei wh = 500, ww = 500; /* 初始化視窗大小(500 x 500) */

/* 建立一組顏色索引(黑, 紅, 綠, 藍, 青, 紫, 黃, 白) */
GLfloat colors[8][3] = { {0.0, 0.0, 0.0}, {1.0, 0.0, 0.0}, {0.0, 1.0, 0.0},
  {0.0, 0.0, 1.0}, {0.0, 1.0, 1.0}, {1.0, 0.0, 1.0}, {1.0, 1.0, 0.0},
  {1.0, 1.0, 1.0} };

/* 建立視窗, 且當縮放或移動視窗時, 調整內容 */
void myReshape(int w, int h) {
  /* adjust clipping box */
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glOrtho(-2.0, 2.0, -2.0, 2.0, -4.0, 4.0);
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
  gluLookAt(1.5, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

  /* 根據視窗大小調整立方體大小, 使其縮放同時又能保持正方體外觀 */
  if (w < h) glViewport(0, 0, w, w);
  else glViewport(0, 0, h, h);
}

void myinit() {
  /* 構建粒子系統 */
  int i, j;
```

```
  for (i = 0; i < num_particles; i++) { /* 初始化粒子質量, 顏色, 座標, 速度 */
    particles[i].mass = 1.0;
    particles[i].color = i % 8;
    for (j = 0; j < 3; j++) {
      /* 偽隨機給予粒子座標和速度 */
      particles[i].position[j] = 2.0 * ((float)rand() / RAND_MAX) - 1.0;
      particles[i].velocity[j] = speed * 2.0 * ((float)rand() / RAND_MAX) - 1.0;
    }
  }
  glPointSize(point_size); /* 初始化粒子大小 */

  glClearColor(0.5, 0.5, 0.5, 1.0); /* 設置背景顏色為灰色 */
}

void myIdle() {
  int i, j, k;
  float dt;
  present_time = glutGet(GLUT_ELAPSED_TIME);
  dt = 0.001 * (present_time - last_time); /* 每幀單位時間 */
  for (i = 0; i < num_particles; i++) {
    for (j = 0; j < 3; j++) {
      particles[i].position[j] += dt * particles[i].velocity[j];
      particles[i].velocity[j] += dt * forces(i, j) / particles[i].mass;
    }
    collision(i);
  }
  if (repulsion)
    for (i = 0; i < num_particles; i++)
      for (k = 0; k < i; k++) {
        d2[i][k] = 0.0;
        for (j = 0; j < 3; j++)
          d2[i][k] += (particles[i].position[j] - particles[k].position[j]) *
          (particles[i].position[j] - particles[k].position[j]);
        d2[k][i] = d2[i][k];
      }
  last_time = present_time;
  glutPostRedisplay();
}

float forces(int i, int j) {
  int k;
  float force = 0.0;
  if (gravity && j == 1)
    force = -1.0; /* 簡易重力(向下1單位) */
  if (repulsion)
    for (k = 0; k < num_particles; k++) { /* 粒子間碰撞力計算 */
      if (k != i)
```

```
        force += 0.001 * (particles[i].position[j] - particles[k].position[j]) / (0.001 + d2[i][k]);
    }
  return(force);
}

void collision(int n) {
  /* 與立方體碰撞測試與碰撞反射 */
  int i;
  for (i = 0; i < 3; i++) {
    if (particles[n].position[i] >= 1.0) {
      particles[n].velocity[i] = -coef * particles[n].velocity[i];
      particles[n].position[i] = 1.0 - coef * (particles[n].position[i] - 1.0);
    }
    if (particles[n].position[i] <= -1.0) {
      particles[n].velocity[i] = -coef * particles[n].velocity[i];
      particles[n].position[i] = -1.0 - coef * (particles[n].position[i] + 1.0);
    }
  }
}

void main_menu(int index) {
  switch (index) {
    case(1): /* 粒子數量 x2 */
      num_particles = 2 * num_particles;
      myinit();
      break;
    case(2): /* 粒子數量 /2 */
      num_particles = num_particles / 2;
      myinit();
      break;
    case(3): /* 粒子速度 x2 */
      speed = 2.0 * speed;
      myinit();
      break;
    case(4): /* 粒子速度 /2 */
      speed = speed / 2.0;
      myinit();
      break;
    case(5): /* 粒子大小 x2 */
      point_size = 2.0 * point_size;
      myinit();
      break;
    case(6): /* 粒子大小 /2 */
      point_size = point_size / 2.0;
      if (point_size < 1.0)
        point_size = 1.0;
      myinit();
      break;
```

```
    case(7): /* 重力開啟/關閉 */
      gravity = !gravity;
      myinit();
      break;
    case(8): /* 粒子非彈性碰撞切換 */
      elastic = !elastic;
      if (elastic)
        coef = 0.9;
      else
        coef = 1.0;
      myinit();
      break;
    case(9): /* 粒子間碰撞切換 */
      repulsion = !repulsion;
      myinit();
      break;
    case(10):
      exit(0);
      break;
  }
}

void myDisplay() {
  int i;

  glClear(GL_COLOR_BUFFER_BIT);
  glBegin(GL_POINTS); /* render all particles */
  for (i = 0; i < num_particles; i++) {
    glColor3fv(colors[particles[i].color]);
    glVertex3fv(particles[i].position);
  }
  glEnd();
  glColor3f(0.0, 0.0, 0.0);
  glutWireCube(2.2); /* outline of box */
  glutSwapBuffers();
}


int main(int argc, char** argv)
{
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
  glutInitWindowSize(wh, ww);
  glutCreateWindow("particle system");
  glutDisplayFunc(myDisplay);
  myinit();

  glutCreateMenu(main_menu); /* 建立操作選單 */
  glutAddMenuEntry("more particles", 1);
```

```
  glutAddMenuEntry("fewer particles", 2);
  glutAddMenuEntry("faster", 3);
  glutAddMenuEntry("slower", 4);
  glutAddMenuEntry("larger particles", 5);
  glutAddMenuEntry("smaller particles", 6);
  glutAddMenuEntry("toggle gravity", 7);
  glutAddMenuEntry("toggle restitution", 8);
  glutAddMenuEntry("toggle repulsion", 9);
  glutAddMenuEntry("quit", 10);
  glutAttachMenu(GLUT_RIGHT_BUTTON); /* 右鍵觸發選單 */

  glutIdleFunc(myIdle);
  glutReshapeFunc(myReshape);
  glutMainLoop();
}
```

十一、參考資料：

(1) 可參考的公開網址

Interactive Computer Graphics: A Top-Down Approach with Shader-Based OpenGL (2-downloads) (inspirit.net.in)