Συστήματα σωματιδίων (particle systems)

Σωματίδιο (particle) στα Γραφικά ονομάζουμε μια «οντότητα» που χαρακτηρίζεται από διάφορες παραμέτρους, κατ'ελάχιστον μια θέση στο χώρο, ένα χρόνο ζωής και ένα διάνυσμα ταχύτητας. Σε ένα σωματίδιο μπορούμε να αποδώσουμε φυσικές παραμέτρους όπως βαρύτητα, επιβράδυνση, μέγεθος, διαφάνεια. Επιπλέον μπορούμε να απεικονίσουμε μια υφή σε αυτό και να το φωτίσουμε με τα φώτα της σκηνής.

Αν θεωρήσουμε έναν πεπερασμένο αριθμό τέτοιων σωματιδίων και μια πηγή που τα εκπέμπει τότε έχουμε ένα σύστημα σωματιδίων (particle system). Ένα σύστημα σωματιδίων μπορεί να χρησιμοποιηθεί για να εξομοιωθούν διάφορες φυσικές διεργασίες όπως εκρήξεις, καπνός, φωτιά, νερό κλπ και η χρήση τους είναι ιδιαίτερα εμφανής στα παιχνίδια τελευταίας γενιάς.

Ένα σωματίδιο αναπαρίστανται συνήθως με ένα struct στην C, ενδεικτικά με τις εξής παραμέτρους:

```
typedef struct
{
   GLfloat position[3];
   GLfloat direction[3];
   GLfloat color[3];
   GLfloat life;
   GLfloat fade;
   GLfloat speed;
   GLfloat size;
}Particle;
```

Οι κυριότερες παράμετροι του σωματιδίου είναι η θέση του (position), κατεύθυνση και ταχύτητα (direction και speed) και η ζωή του (life). Κάθε σωματίδιο έχει πεπερασμένο χρόνο ζωής πέρα από τον οποίο σταματάμε να το απεικονίζουμε.

```
#define   MAX_PARTICLES   1000

Particle particles[MAX_PARTICLES];
```

Εφόσον αποφασίσουμε τον αριθμό των σωματιδίων στο σύστημα (1000 στην περίπτωση μας), ορίζουμε ένα πίνακα που θα αποθηκεύει τις πληροφορίες για κάθε ένα από αυτά. Την μνήμη την δεσμεύουμε στην αρχή του προγράμματος και όχι την ώρα που εκτελείται ο βρόγχος.

Βασική αρχή στις εφαρμογές γραφικών και κατ'επέκταση στα βιντεοπαιχνίδια είναι ότι ποτέ δεν δεσμεύουμε μνήμη, ούτε και την απελευθερώνουμε, την ώρα που τρέχει ο κύριος

βρόγχος του προγράμματος, εκεί δηλαδή που γίνεται η ανανέωση και η απεικόνιση των αντικειμένων. Η διαχείριση μνήμης είναι πολύ ακριβή, από πλευράς χρόνου, διεργασία και πρέπει να αποφεύγεται στα τμήματα του κώδικα που απαιτούμε την μέγιστη ταχύτητα.

Το σύστημα σωματιδίων αρχικοποιείται στην συνάρτηση init(). Η μόνη αρχικοποιηση που κάνουμε ουσιαστικά είναι να θέσουμε την ζωή κάθε σωματιδίου στο 0. Ο λόγος που το κάνουμε αυτό θα γίνει φανερός αργότερα:

```
   for (int i=0;i<MAX_PARTICLES;i++)
1  {
2      particles&#91;i&#93;.life=0.0f;
3  }
4  &#91;/sourcecode&#93;
5
6  <span style="font-family:Verdana;font-size:9pt;">Στην συνέχεια, στην συνάρτηση void renderScene(void),
7  ζωγραφίζουμε το σύστημα σωματιδίων σε μια καθορισμένη θέση. Χρησιμοποιούμε μια άσπρη υφή
8  για απεικόνιση κάθε σωματιδίου, στην ουσία θα μπορούσε να είναι μια οποιαδήποτε.
9  </span>
10
11
12glBindTexture(GL_TEXTURE_2D, textures[TEXTURE_PARTICLE]);
13for (int i=0;i<MAX_PARTICLES;i++)
14{
15    if (particles&#91;i&#93;.life > 0.0f)
16    {
17        float x=particles[i].position[0];
18        float y=particles[i].position[1];
19        float z=particles[i].position[2];
20        float size = particles[i].size;
21
22
23        glColor4f(particles[i].color[0],particles[i].color[1],particles[i].color[2],particles[i].life);
24        glBegin(GL_TRIANGLE_STRIP);
25        glTexCoord2d(1,1);
26        glVertex3f(x+size*0.5f,y+size*0.5f,z); // Top Right
27        glTexCoord2d(0,1);
28        glVertex3f(x-size*0.5f,y+size*0.5f,z); // Top Left
29        glTexCoord2d(1,0);
30        glVertex3f(x+size*0.5f,y-size*0.5f,z); // Bottom Right
31        glTexCoord2d(0,0);
32        glVertex3f(x-size*0.5f,y-size*0.5f,z); // Bottom Left
33        glEnd();
34
35        float speed = particles[i].speed;
36        particles[i].position[0] += speed*particles[i].direction[0];
37        particles[i].position[1] += speed*particles[i].direction[1];
38        particles[i].position[2] += speed*particles[i].direction[2];
39        particles[i].life -= particles[i].fade;
      }
```

Ένα σωματίδιο απεικονίζεται στην οθόνη μόνο αν η ζωή του είναι μεγαλύτερη του μηδενός (particles[i].life > 0.0f). Για να απεικονίσουμε το κάθε σωματίδιο χρησιμοποιούμε ένα Triangle Strip. Ο λόγος που επιλέγουμε αυτό αντί ενός Triangle List (λίστα τριγώνων, GL_TRIANGLES) είναι διότι μας επιτρέπει να ζωγραφίσουμε ένα ορθογώνιο με 2 κορυφές λιγότερο (4, αντί τις 6 που απαιτεί ένα Triangle List). Σε περιπτώσεις που απεικονίζουμε μεγάλο αριθμό πολυγώνων, όπως στην δική μας, ο μετασχηματισμός 2 κορυφών λιγότερο ανά σωματίδιο είναι μεγάλο κέρδος. Παρόμοιο αποτέλεσμα θα είχαμε αν επιλέγαμε και GL_QUADS σαν πολύγωνο.

Πριν απεικονίσουμε το κάθε σωματίδιο θέτουμε και το χρώμα του με την glColor. Άξιο προσοχής είναι στη περίπτωση αυτή το ότι χρησιμοποιούμε το εναπομείναντα χρόνο ζωής του κάθε σωματιδίου ως διαφάνεια του. Καθώς η ζωή (particles[i].life) κάθε σωματιδίου πέφτει στο μηδέν, τόσο πιο διάφανο αυτό θα γίνεται μέχρι να εξαφανιστεί.

Τέλος αλλάζουμε την θέση και την εναπομένουσα ζωή του κάθε σωματιδίου ανάλογα με την κατεύθυνση και την ταχύτητα του (particles[i].direction και speed) και το ρυθμό εξασθένισης (particles[i].fade).

Όταν η ζωή ενός σωματιδίου μηδενιστεί, τότε σταματάμε να το απεικονίζουμε στην οθόνη. Επαναχρησιμοποιούμε όμως την μνήμη που του αντιστοιχεί για να δημιουργήσουμε ένα νέο, σε καινούργια θέση και με καινούργιες παραμέτρους.

```
1    else
2    {
3        particles[i].life=1.0f;
4        particles[i].fade=0.001f+float(rand()%100)/3000.0f;
5        particles[i].position[0]=0.0f;
6        particles[i].position[1]=0.0f;
7        particles[i].position[2]=0.0f;
8        particles[i].direction[0]=float((rand()%100)-50.0f);
9        particles[i].direction[1]=float((rand()%100));
10       particles[i].direction[2]=float((rand()%100)-50.0f);
11       particles[i].speed=float((rand()%100)/100000.0f);
12
13       particles[i].size=float((rand()%50)/50.0f)*5.0f;
14       particles[i].color[0]=(rand()%256)/255.0f;
15       particles[i].color[1]=(rand()%256)/255.0f;
16       particles[i].color[2]=(rand()%256)/255.0f;
17   }
```

Οι περισσότερες παράμετροι προσδιορίζονται τυχαία. Θεωρούμε ότι το σύστημα σωματιδίων βρίσκεται στην αρχή των αξόνων οπότε κάθε σωματίδιο ξεκινά από την θέση (0,0,0). Η θέση της πηγής ενός συστήματος σωματιδίων μπορεί να μετακινείται φυσικά. Αν το σύστημα περιέγραφε καπνό που έβγαινε από τους προωθητήρες ενός πυραύλου, η θέση του συστήματος θα μετακινούνταν ανάλογα με την θέση του πυραύλου.

Η κατεύθυνση, η ταχύτητα, το μέγεθος και το χρώμα του κάθε σωματιδίου προσδιορίζονται τυχαία. Επειδή όταν αρχικοποιήσαμε το κάθε σωματίδιο στην init() θέσαμε το χρόνο ζωής του στο 0, στην renderScene() πριν απεικονίσει το κάθε σωματίδιο θα το αρχικοποιήσει (το if (particles[i].life > 0.0f) θα είναι ψευδές αρχικά). Το λόγος που το κάναμε αυτό είναι για να αποφύγουμε επανάληψη του κώδικα αρχικοποίησης. Διαφορετικά θα έπρεπε να το συμπεριλάβουμε και στην init(). Αλλιώς, όπως θα έπρεπε ορθότερα, θα μπορούσαμε και να

τον μετακινήσουμε σε μια ξεχωριστή συνάρτηση!

Έτσι κατασκευάσαμε ένα πολύ απλοϊκό σύστημα σωματιδίων. Δοκιμάστε να αλλάξετε την υφή του κάθε σωματιδίου (το παράδειγμα που ακολουθεί το κείμενο αυτό περιλαμβάνει και μια υφή φωτιάς), και τις παραμέτρους του για διαπιστώσετε πως πραγματικά λειτουργεί ένα σύστημα σωματιδίων.

Παρατήρηση: Το σύστημα σωματιδίων που φτιάξαμε είναι απλοϊκό. Στην πραγματικότητα κάθε σωματίδιο απεικονίζεται με ένα sprite το οποίο είναι πάντα παράλληλο με την οθόνη (και δεν περιστρέφεται ότι περιστρέψουμε την κάμερα όπως συμβαίνει στο παράδειγμα). Μπορείτε σαν άσκηση να κάνετε το κάθε σωματίδιο παράλληλο με την οθόνη περιστρέφοντας το κατάλληλα.