UAS IF3260 Grafika Komputer

Particle System

Oleh

Erick Wijaya 13515057

Kezia Suhendra 13515063

Catherine Almira 13515111

TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2018

1. Penjelasan *Particle System* yang Dibuat

a. Hujan

Partikel hujan dibuat dengan membuat tipe data bentukan yang diberi nama *particles* yang berisi beberapa variabel yang menandakan status dari partikel hujan tersebut seperti posisi x, y, z, warna RGB, kondisi *alive*, *velocity, gravity, decay,* dan masa hidup dari partikel. Lakukan inisialisasi untuk setiap variabel pada tipe bentukan, kemudian nilai-nilai tersebut di-*append* ke dalam array *vertex* yang ada. Partikel-partikel hujan digambar dengan menggunakan *shader*. Posisi dari partikel akan terus diubah berdasarkan *velocity* dan gravitasi untuk membuat hujan tampak bergerak pada layar. Kemudian setelah hujan memasuki posisi kedalaman tertentu, partikel hujan akan di *respawn* pada posisi awal dan proses ini akan dilakukan secara terus-menerus hingga program dihentikan. Berikut merupakan kode utama untuk menampilkan hujan.

```
void initParticles(int idx, bool starter) {
    par_sys[idx].alive = true;
    par_sys[idx].life = 1.0;
    par_sys[idx].fade = float(rand()%100)/1000.0f+0.003f;

    par_sys[idx].xpos = (float) (rand() % 21) - 10;
    par_sys[idx].ypos = 8.0f;
    par_sys[idx].zpos = (float) (rand() % 21) - 10;

    if (!starter) {
        inc_x_rain[idx] =  par_sys[idx].xpos - init_x_rain[idx];
        inc_z_rain[idx] = par_sys[idx].zpos - init_z_rain[idx];
        inc_y_rain[idx] = 0.0f;
    } else {
        init_x_rain[loop] = par_sys[idx].xpos;
        init_z_rain[loop] = par_sys[idx].zpos;
    }

    par_sys[idx].red = 0.5;
    par_sys[idx].green = 0.5;
    par_sys[idx].blue = 1.0;

    par_sys[idx].vel = velocity;
    par_sys[idx].gravity = -0.8;//-0.8;
}

void init(bool starter) {
    for (loop = 0; loop < MAX_RAIN_PARTICLES; loop++) {
        initParticles(loop, starter);
    }
}

void drawRain(bool starter) {
    float x, y, z;

    for (loop = 0; loop < MAX_RAIN_PARTICLES; loop++) {
        if (par_sys[loop].alive == true) {
             x = par_sys[loop].xpos;
             y = par_sys[loop].ypos;
             z = par_sys[loop].zpos + zoom;

             if (starter) {
```

```
                init_x_rain[loop] = x;
                init_z_rain[loop] = z;
             }

            vertices[idxRain++] = x;
            vertices[idxRain++] = y;
            vertices[idxRain++] = z;

            vertices[idxRain++] = 1.0f;
            vertices[idxRain++] = 1.0f;
            vertices[idxRain++] = 1.0f;

            vertices[idxRain++] = 1.0f;
            vertices[idxRain++] = 0.0f;

            vertices[idxRain++] = x;
            vertices[idxRain++] = y + 0.5f;
            vertices[idxRain++] = z;

            vertices[idxRain++] = 1.0f;
            vertices[idxRain++] = 1.0f;
            vertices[idxRain++] = 1.0f;

            vertices[idxRain++] = 0.0f;
            vertices[idxRain++] = 0.0f;

            // Update values
            //Move
            // Adjust slowdown for speed!
            par_sys[loop].ypos += par_sys[loop].vel / (slowdown*1000);
            inc_y_rain[loop] += par_sys[loop].vel / (slowdown*1000);
            par_sys[loop].vel += par_sys[loop].gravity;
            // Decay
            par_sys[loop].life -= par_sys[loop].fade;

            if (par_sys[loop].ypos <= accum) {
                par_sys[loop].life = -1.0;
            }
            //Revive
            if (par_sys[loop].life < 0.0) {
                initParticles(loop, starter);
            }
        }
    }
    idxRain = objectLastIndex;
}
```

b. Asap

Partikel asap yang dibuat pada tugas ini menggunakan metode *instancing*. Metode *instancing* adalah sebuah metode pembuatan partikel dimana terlebih dahulu dibuat sebuah bentuk yang akan menjadi dasar, yang dalam tugas kali ini berbentuk sebuah kotak. Kemudian dibuat banyak instance dari bentuk dasar tersebut sehingga dapat tercetak *particle system* dari banyak *instance* tersebut. Secara teknis, hal tersebut dapat diterapkan dengan menggunakan beberapa *buffer* untuk mendeskripsikan bentuk dasar dan mendeskripsikan setiap *instance* dari bentuk dasar tersebut. Untuk memperbaharui *particle system*, cukup

memperbaharui *buffer* pada pusat partikel pada setiap *frame*. Berikut ini adalah kode untuk membuat bentuk dasar beserta penggunaan *buffer*.

```
// Bentuk dasar
static const GLfloat g_vertex_buffer_data[] = {
 -0.5f, -0.5f, 0.0f,
 0.5f, -0.5f, 0.0f,
 -0.5f, 0.5f, 0.0f,
 0.5f, 0.5f, 0.0f,
};

// Atribut buffer pertama: vertices
glEnableVertexAttribArray(3);
glBindBuffer(GL_ARRAY_BUFFER, billboard_vertex_buffer);
glVertexAttribPointer(
  3, // attribute
  3, // size
  GL_FLOAT, // type
  GL_FALSE, // normalized
  0, // stride
  (void*)0 // array buffer offset
);

// Atribut buffer kedua: posisi pusat partikel
glEnableVertexAttribArray(4);
glBindBuffer(GL_ARRAY_BUFFER, particles_position_buffer);
glVertexAttribPointer(
  4, // attribute
  4, // size
  GL_FLOAT, // type
  GL_FALSE, // normalized
  0, // stride
  (void*)0 // array buffer offset
);

// Atribut buffer ketiga: warna partikel
glEnableVertexAttribArray(5);
glBindBuffer(GL_ARRAY_BUFFER, particles_color_buffer);
glVertexAttribPointer(
  5, // attribute
  4, // size
  GL_UNSIGNED_BYTE, // type
  GL_TRUE, // normalized
  0, // stride
  (void*)0 // array buffer offset
);
```

Partikel terbentuk dan hilang dalam waktu yang sangat cepat. Oleh karena itu, inisialisasi partikel dengan menggunakan perintah `new Particle()` setiap kali partikel akan dicetak dinilai kurang tepat. Untuk mengatasi hal tersebut, dibuat sebuah array yang berisi partikel. Pada array tersebut diberikan suatu penanda yang menandai partikel terakhir yang telah dicetak untuk kemudian mencetak partikel pada indeks berikutnya. Partikel memiliki atribut yang menyatakan apakah partikel tersebut masih aktif atau sudah hilang. Partikel yang akan hilang disimpan pada akhir *buffer*.

2. Tambahan Fitur/Operasi Lain

a. Car Movement

Mobil yang kelompok kami buat dapat bergerak maju, mundur, belok kiri, dan belok kanan. Gerakan mobil dikendalikan melalui keyboard dengan tombol *arrow*. Implementasi mobil maju dan mundur dibuat dengan menggeser seluruh titik mobil sesuai dengan arah sudut mobil, sedangkan implementasi mobil berbelok dibuat dengan memutar arah sudut mobil. Berikut adalah kode program untuk pergerakan mobil.

```
void DoMovement( ) {
    // Camera controls
    if(keys[GLFW_KEY_UP] )
    {
        if (CheckPositionCanMoveBackward(CAR_SPEED_FAST)) {
            GLfloat move = CheckPositionLowerSpeed(CAR_SPEED_FAST) ?
CAR_SPEED_SLOW : CAR_SPEED_FAST;
            middlePoint[0+2] -= move * cos(rad);
            middlePoint[0] -= move * sin(rad);
            for (int i=0; i<carLastIndex; i+=8) {
                vertices[i+2] -= move * cos(rad);
                vertices[i] -= move * sin(rad);
            }
            glBufferData( GL_ARRAY_BUFFER, sizeof( vertices ), vertices,
GL_STATIC_DRAW );
        }
    }

    if(keys[GLFW_KEY_DOWN] )
    {
        if (CheckPositionCanMoveForward(CAR_SPEED_FAST)) {
            GLfloat move = CheckPositionLowerSpeed(CAR_SPEED_FAST) ?
CAR_SPEED_SLOW : CAR_SPEED_FAST;
            middlePoint[0+2] += move * cos(rad);
            middlePoint[0] += move * sin(rad);
            for (int i=0; i<carLastIndex; i+=8) {
                vertices[i+2] += move * cos(rad);
                vertices[i] += move * sin(rad);
            }
            glBufferData( GL_ARRAY_BUFFER, sizeof( vertices ), vertices,
GL_STATIC_DRAW );
        }
    }

    if(keys[GLFW_KEY_LEFT] )
    {
        rad += 0.1f;
        for (int i=0; i<carLastIndex; i+=8) {
            GLfloat zi = vertices[i+2];
            GLfloat xi = vertices[i];
            vertices[i+2] = (zi-middlePoint[2])*cos(0.1f) -
(xi-middlePoint[0])*sin(0.1f) + middlePoint[2];
            vertices[i] = (zi-middlePoint[2])*sin(0.1f) +
(xi-middlePoint[0])*cos(0.1f) + middlePoint[0];
        }
        glBufferData( GL_ARRAY_BUFFER, sizeof( vertices ), vertices,
GL_STATIC_DRAW );
    }

    if(keys[GLFW_KEY_RIGHT] )
    {
```

```
        rad -= 0.1f;
        for (int i=0; i<carLastIndex; i+=8) {
            GLfloat zi = vertices[i+2];
            GLfloat xi = vertices[i];
            vertices[i+2] = (zi-middlePoint[2])*cos(-0.1f) -
(xi-middlePoint[0])*sin(-0.1f) + middlePoint[2];
            vertices[i] = (zi-middlePoint[2])*sin(-0.1f) +
(xi-middlePoint[0])*cos(-0.1f) + middlePoint[0];
        }
        glBufferData( GL_ARRAY_BUFFER, sizeof( vertices ), vertices,
GL_STATIC_DRAW );
    }

    // implementasi pergerakan kamera..
}
```

b. Car Speed Control

Mobil dapat bergerak dengan cepat pada jalanan, tetapi mobil akan bergerak lebih lambat apabila berada diatas tanah. Hal tersebut diimplementasikan dengan memeriksa posisi mobil; kecepatan mobil dikurangi bila mobil berada pada daerah tanah. Berikut adalah kode program untuk memeriksa posisi mobil.

```
bool CheckPositionLowerSpeed(GLfloat move) {
    return (middlePoint[0] + move < 22.0f)
        && (middlePoint[0] - move > 8.0f)
        && (middlePoint[2] + move < 10.0f)
        && (middlePoint[2] - move > -10.0f);
}
```

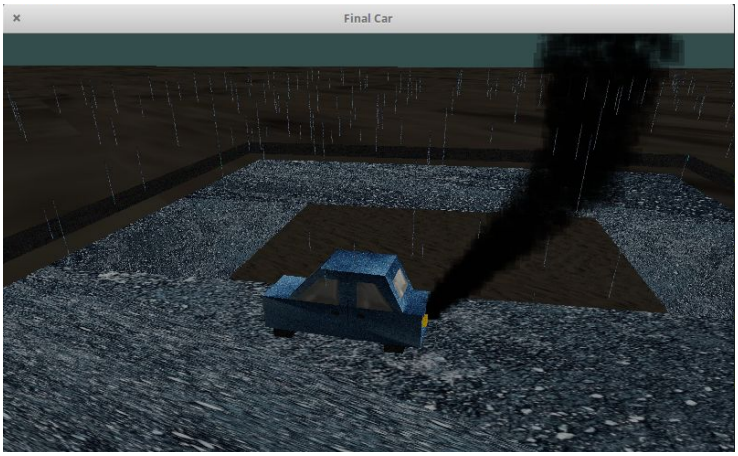
c. Car Collision

Selain mobil dapat bergerak, mobil juga dapat berhenti apabila menabrak pembatas. Pendeteksian tabrakan mobil diimplementasi dengan memeriksa posisi mobil dan posisi pembatas; mobil dibuat tidak dapat bergerak bila mobil akan menembus posisi pembatas. Berikut adalah kode program untuk melakukan pemeriksaan tabrakan mobil.

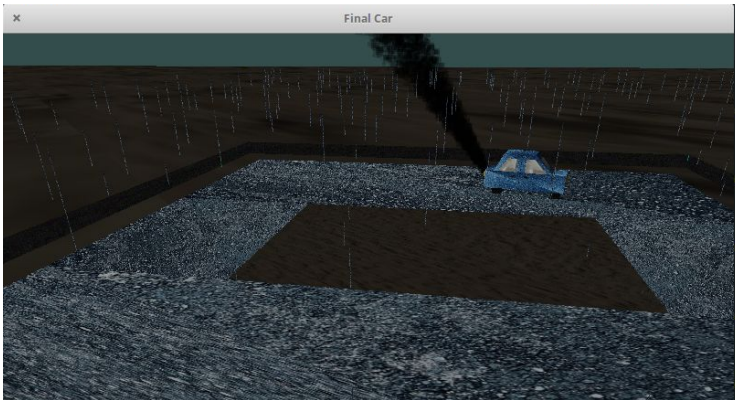
```
// Collision Detection Mobil Maju
bool CheckPositionCanMoveForward(GLfloat move) {
    return (middlePoint[0] + move * sin(rad) < 38.0f)
        && (middlePoint[2] + move * cos(rad) < 20.0f)
        && (middlePoint[0] + move * sin(rad) > -8.0f)
        && (middlePoint[2] + move * cos(rad) > -20.0f);
}

// Collision Detection Mobil Mundur
bool CheckPositionCanMoveBackward(GLfloat move) {
    return (middlePoint[0] - move * sin(rad) < 38.0f)
        && (middlePoint[2] - move * cos(rad) < 20.0f)
        && (middlePoint[0] - move * sin(rad) > -8.0f)
        && (middlePoint[2] - move * cos(rad) > -20.0f);
}
```

3. Screenshot Penggunaan

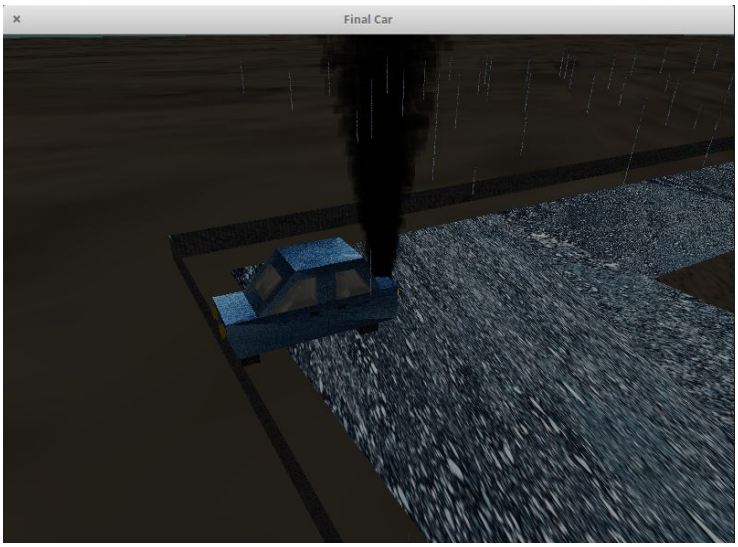
a. Smoke and Rain



b. Car Movement



c. Car Collision



4. Pembagian Tugas

No.	NIM	Nama	Tugas
1.	13515057	Erick Wijaya	Fitur Tambahan, Rain
2.	13515063	Kezia Suhendra	Rain Particle
3.	13515111	Catherine Almira	Smoke Particle