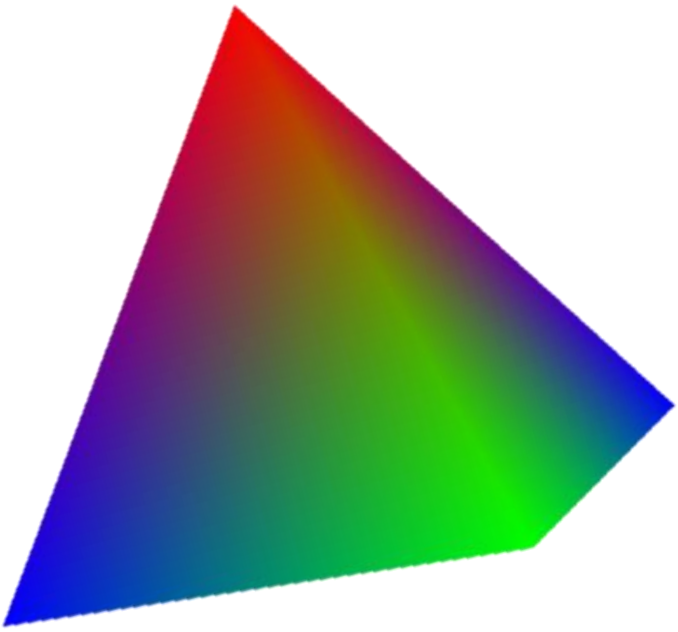


Rocket Mania

An OpenGL Game

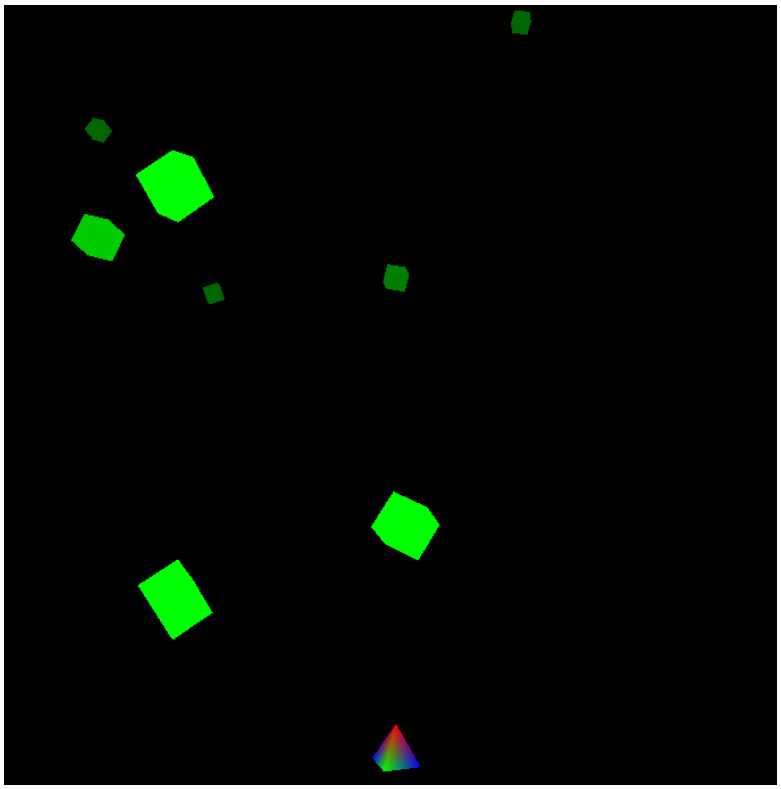
Oleh:

Royyan Abdullah Dzakiy

13515123

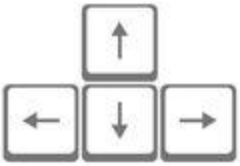
Deskripsi

Permainan yang memiliki latar cerita suatu perjalanan panjang di luar angkasa yang penuh tantangan. Pemain diharuskan mampu menghindarkan roket dari menabrak berbagai halang rintang berupa meteorit berbentuk kubus. Seiring dengan berjalannya permainan, score akan terus bertambah!



Cara Bermain

Gunakan tombol yang ada pada keyboard untuk menggerakkan Rocket.



Up Arrow = Gerakkan Rocket ke atas

Right Arrow = Gerakkan Rocket ke kanan

Left Arrow = Gerakkan Rocket ke kiri

Down Arrow = Gerakkan Rocket ke bawah

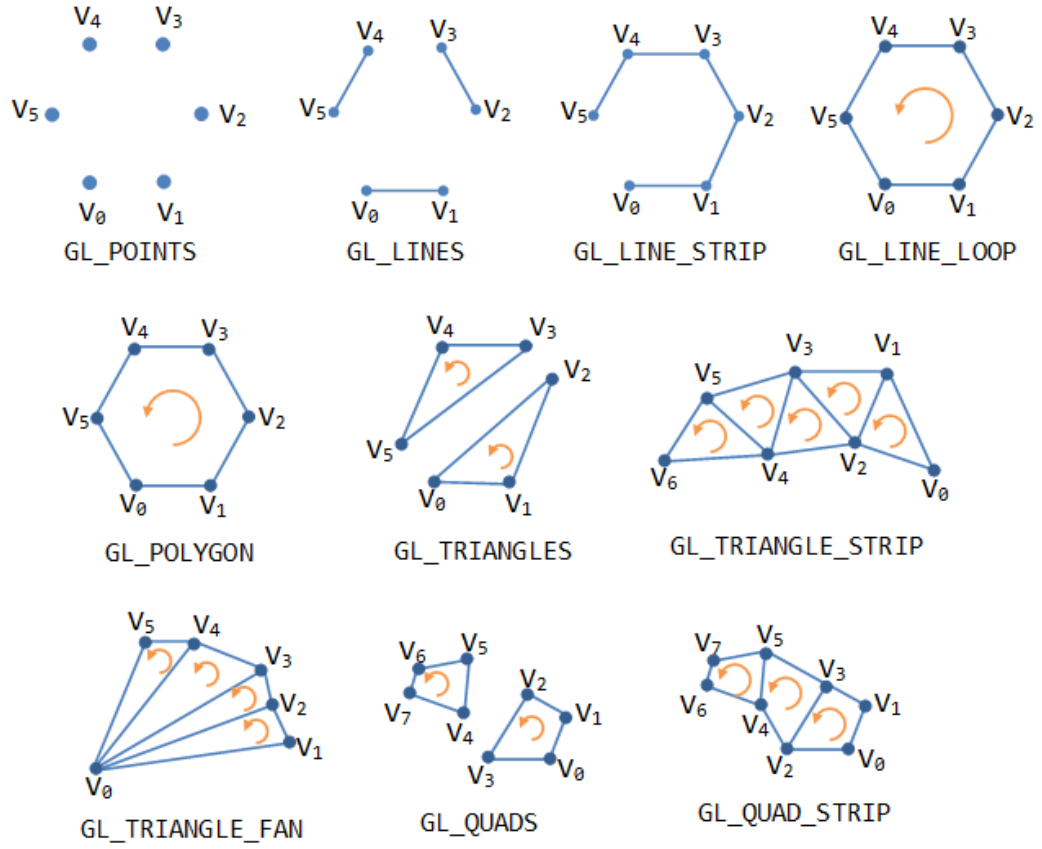
Langkah Pengerjaan

1. Set Environment
2. Implement
    - Tampilkan objek 3D: kubus, pyramid
    - Transformasi objek 3D: translasi, rotasi
    - Interaktif: mouse, keyboard
3. Gamify!

Teori

Geometric Primitives

Berupa primitif yang telah tersedia di opengl dan dapat digunakan untuk menggambarkan objek yang diinginkan.



OpenGL Primitives

Vertex Processing

Mekanisme menampilkan gambar atau dari model dan vertex pada opengl. Tampilan 3D dibuat seperti mengambil gambar dg kamera. Berikut transformasinya:

1. Posisikan object di dunia transformasi (Model Transformation)

2. Posisikan dan orientasikan Kamera (View Transformation)

3. Pilih lensa kamera (angle), focus length & zoom factor, field view (Projection Transformation)

Model Transform

$$\boldsymbol{V} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

1. Representasi Vertex: Menggunakan array yang menyimpan nilai posisi pada sumbu x, y, z.

2. Scaling: Mekanisme untuk melakukan scaling

$$\boldsymbol{S}(\boldsymbol{\alpha_x}, \boldsymbol{\alpha_y}, \boldsymbol{\alpha_z}) = \begin{bmatrix} \alpha_x & 0 & 0 \\ 0 & \alpha_y & 0 \\ 0 & 0 & \alpha_z \end{bmatrix} \qquad \boldsymbol{V'} = \boldsymbol{S}\,\boldsymbol{V} = \begin{bmatrix} \alpha_x & 0 & 0 \\ 0 & \alpha_y & 0 \\ 0 & 0 & \alpha_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \alpha_x x \\ \alpha_y y \\ \alpha_z z \end{bmatrix}$$

3. Rotation: Mekanisme untuk melakukan rotation

$$\boldsymbol{R_z}(\theta) = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, \boldsymbol{R_x}(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta & -sin\theta \\ 0 & sin\theta & cos\theta \end{bmatrix}, \boldsymbol{R_y}(\theta) = \begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix}$$

4. Translation: 4-component homogeneous coordinates

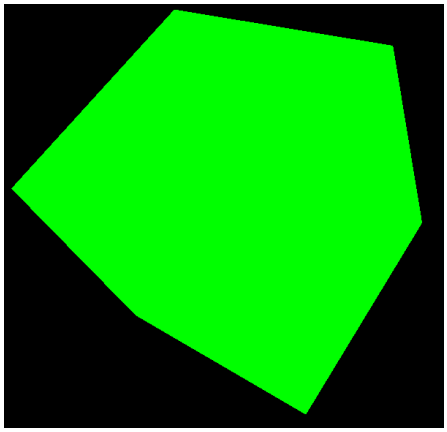
$$\boldsymbol{V'} = \boldsymbol{T}(d)\,\boldsymbol{V} = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x + d_x \\ y + d_y \\ z + d_z \\ 1 \end{bmatrix}$$

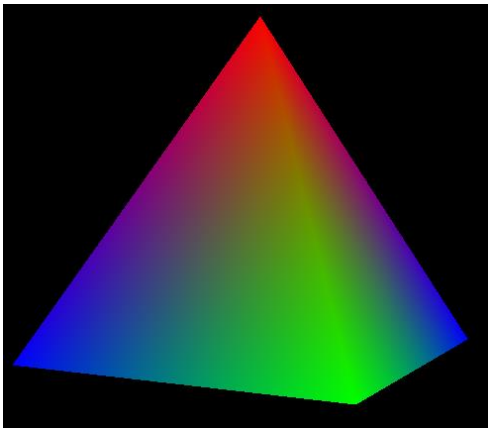
Pembuatan

Pembuatan game berdasarkan metode Model View Controller (M.V.C.) untuk memudahkan developer dikarenakan kompleksnya kode yang digunakan dan banyaknya library yang digunakan. Metode M.V.C membuat kode game semakin scalable.

Model

Terdapat 1 abstract class yaitu Entitiy, dengan penurunannya kepada 2 model utama berupa Cube dan Pyramid.



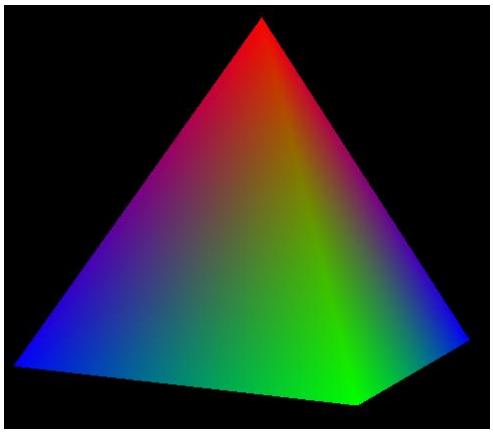


Adapun property dan method yang digunakan adalah

- Size = ukuran entitiy
- posX, posY, posZ, angle = value berupa posisi atau sudut
- speedX, speedY, speedZ, speedRot = kecepatan berubahnya value
- Move() = method untuk merubah pos dan angle sehingga entitiy tampak bergerak (pos & angle + speed)
- Draw() = mekanisme dasar untuk menggambarkan objek ke layar di opengl.

View

Tampilan dari objek yang ada pada permainan ini dibuat dengan metode draw(). Adapun metode ini menggunakan API opengl untuk menampilkan vertex dan memberikan properti padanya. Berikut contoh kode berupa cara menampilkan piramid:



```
glBegin(GL_TRIANGLES);           // Begin drawing the pyramid with 4
triangles
 // Front
glColor3f(size, 0.0f, 0.0f);     // Red
glVertex3f(0.0f, size, 0.0f);
glColor3f(0.0f, size, 0.0f);     // Green
glVertex3f(-size, -size, size);
glColor3f(0.0f, 0.0f, size);     // Blue
glVertex3f(size, -size, size);
// Right
glColor3f(size, 0.0f, 0.0f);     // Red
glVertex3f(0.0f, size, 0.0f);
glColor3f(0.0f, 0.0f, size);     // Blue
glVertex3f(size, -size, size);
glColor3f(0.0f, size, 0.0f);     // Green
glVertex3f(size, -size, -size);
// Back
glColor3f(size, 0.0f, 0.0f);     // Red
glVertex3f(0.0f, size, 0.0f);
glColor3f(0.0f, size, 0.0f);     // Green
glVertex3f(size, -size, -size);
glColor3f(0.0f, 0.0f, size);     // Blue
glVertex3f(-size, -size, -size);
// Left
glColor3f(size, 0.0f, 0.0f);       // Red
glVertex3f(0.0f, size, 0.0f);
glColor3f(0.0f, 0.0f, size);       // Blue
glVertex3f(-size, -size, -size);
glColor3f(0.0f, size, 0.0f);       // Green
glVertex3f(-size, -size, size);
glEnd();   // Done drawing the pyramid
```

Controller

Untuk memudahkan mengendalikan sekian banyak kubus, maka dibuat mekanisme kontroller berupa sebuah kode untuk mengiterasi seluruh kubus dan menjalankan fungsi bawaannya yaitu move(). Adapun pada saat awal inisiasi, seluruh kubus yang dibuat telah memiliki nilai random.

Selanjutnya adalah mekanisme collide, atau menabrak dan berinteraksi dengan objek lain. Hal ini diterapkan pada roket sehingga ketika roket menabrak asteroid (kubus), maka ia akan segera hilang. Rumusan yang digunakan adalah sebagai berikut.

(x_min <= x && x <= x_max && y_min <= y && y <= y_max && z_min <= z && z <= z_max)

Referensi

https://www.ntu.edu.sg/home/ehchua/programming/opengl/CG_BasicsTheory.html#zz-3.3

https://nccastaff.bournemouth.ac.uk/jmacey/RobTheBloke/www/opengl_programming.html

BGM:

- http://soundimage.org/sci-fi/