Computer Graphics: 3D Fractalen

Jakob Struye
Tim Leys

3D Fractalen (0.75 punten)

Voeg ondersteuning aan je grafische engine toe voor het genereren van 3D Fractalen op basis van platonische lichamen. Meerbepaald moeten er 3D Fractalen kunnen worden gegenereerd op basis van de volgende ruimtelijke figuren:

- Kubus (Cube)
- Dodecahedron
- Icosahedron
- Octahedron
- Tetrahedron

Merk op dat het genereren van deze 3D Fractalen onafhankelijk is van de manier waarop de uiteindelijke ruimtelijke figuur wordt weergegeven. Je engine moet dus in staat zijn om deze 3D Fractalen weer te geven volgens alle manieren die door je engine ondersteund worden. (Wireframe, Z-Buffering met lijnen, Z-Buffering met driehoeken, ...)

Invoerformaat

Het invoer formaat voor deze opdracht is een uitbreiding van het invoer formaat dat voor de vorige drie opdrachten werd gespecificeerd. `type` veld in de `General` sectie kan de waarden '*Wireframe*', '*ZBufferedWireframe*' en '*ZBuffering*' aannemen afhankelijk van de manier waarop de fractaal moet worden weergegeven.

Elke 3D Fractaal wordt als een nieuw type figuur beschouwd. Het `type` veld van deze figuren is afhankelijk van het platonische lichaam dat moet worden gebruikt om de fractaal te genereren. De mogelijke waarden zijn '*FractalCube*', '*FractalDodecahedron*', '*FractalIcosahedron*', '*FractalOctahedron*' en '*FractalTetrahedron*'. De `Figure*` sectie om een 3D Fractaal te beschrijven bevat, naast de standaard velden die in de vorige opgaves werden gespecificeerd, ook de onderstaande velden:

- **fractalScale** (double): De schaal factor die moet worden gebruikt bij het vervangen van de figuur door de kleinere figuren. Een schaal factor van 3, bijvoorbeeld, betekent dat bij elke stap de nieuwe figuren 3 keer kleiner moeten zijn dan het origineel.
- **nrIterations** (integer): Het aantal iteraties dat moet worden uitgevoerd bij het genereren van de 3D-Fractaal. Als dit veld gelijk is aan 0 moet de oorspronkelijke figuur worden weergegeven. Als dit veld gelijk is aan 1 moeten alle punten één keer worden vervangen, bij een waarde van 2 twee keer, etc.

Ter verduidelijking wordt hieronder een voorbeeld gegeven:

```
[General]
type = "Wireframe"
size = 1000
eye = (100, 50.0, 75.0);
backgroundcolor = (0.0, 0.0, 0.0)
color = (0.0, 1.0, 0.0)
nrFigures = 1

[Figure0]
type = "FractalCube"
scale = 1
rotateX = 0
rotateY = 0
rotateZ = 0
center = (0, 0, 0)
color = (1, 0, 0)
fractalScale = 3
nrIterations = 1
```

Hou er rekening mee dat er in één input bestand zowel 3D-Fractalen als gewone figuren kunnen voorkomen.

BuckyBall (0.125 punten)

Implementeer een functie om een Buckyball te genereren. Zorg er daarna voor dat je engine ook een 3D Fractaal op basis van deze ruimtelijke figuur kan genereren. De procedure om een Buckyball te generern op basis van een Icosahedron staat beschreven op pagina 55 van de cursus. Voor het genereren van de 3D Fractaal op basis van een Bucky ball kan dezelfde procedure worden gevolgd als voor de platonische lichamen.

Invoerformaat

Voor de BuckyBall is het `type` veld in de `Figure` sectie gelijk aan '*BuckyBall*'. Voor de rest worden alle velden overgenomen die voor de platonische lichamen gedefinieërd zijn. Een voorbeeld wordt hieronder gegeven:

```
[Figure0]
type = "BuckyBall"
scale = 1
rotateX = 0
rotateY = 0
rotateZ = 0
center = (0, 0, 0)
color = (1, 1, 1)
```

Voor de 'BuckyBall Fractaal' is het `type` veld in de `Figure` sectie gelijk aan '*FractalBuckyBall*'. De overige velden voor de '*FractalBucyBall*' zijn dezelfde als voor alle andere 3D Fractalen. Een voorbeeld wordt hieronder gegeven:

```
[Figure0]
type = "FractalBuckyBall"
scale = 1
```

```
rotateX = 0
rotateY = 0
rotateZ = 0
center = (0, 0, 0)
color = (1, 0, 0)
fractalScale = 3
nrIterations = 1
```

Mengerspons (0.125 punten)

Implementeer een functie om een Mengerspons te genereren op basis van een Kubus (Cube). Het aantal keer dat elke kubus door kleinere kubussen wordt vervangen moet instelbaar zijn en de Mengerspons moet (net zoals de oorspronkelijke kubus) rond de oorsprong gegenereerd worden. Een 'fractaal' van de Mengerspons moet *niet* worden ondersteund, gezien de Mengerspons op zich al een fractaal is. Een optionele extra optimalisatie is het weghalen van de ribben van de binnenin gelegen kleinere kubussen. Meer informatie over het genereren van de Mengerspons vind je hier: `http://en.wikipedia.org/wiki/Menger_sponge`

Invoerformaat

Het invoerformaat voor de Mengerspons is gebaseerd op het formaat voor 3D Fractalen. Het `type` veld van de figuur is voor de Mengerspons altijd gelijk aan '*MengerSponge*'. Daarnaast worden ook de `scale`, `rotateX-Z`, `center`, `color`, en `nrIterations` velden overgenomen. Het `fractalScale` veld komt *niet* voor! Een voorbeeld wordt hieronder gegeven:

```
[Figure0]
type = "MengerSponge"
nrIterations = 1
scale = 1
rotateX = 0
rotateY = 0
rotateZ = 0
center = (0, 0, 0)
color = (1, 1, 0)
```

View Frustum (0.5 punten)

1. Voeg de mogelijkheid toe om een kijkrichting en andere parameters van het view frustum in te stellen.
2. Clip (gedeeltelijk) driehoeken die niet volledig binnen het view frustum vallen, zodat het eyepoint zich ook "binnen" de figuren kan bevinden.

Invoerformaat

Het invoerformaat voor het view frustum bevindt zich volledig in de `General` sectie. De volgende velden zijn toegevoegd:

- **clipping** (boolean): Voer clipping uit als dit veld waar is. Enkel in dat geval zijn de verdere velden ook aanwezig.

- **viewDirection** (tuple van 3 doubles): De vector die de kijkrichting vanuit het eyepoint voorstelt.
- **dNear** (double): De afstand tussen het eyepoint en near plane.
- **dFar** (double): De afstand tussen het eyepoint en far plane.
- **hfov** (double): De horizontale kijkhoek vanuit het eyepoint, in graden.
- **aspectRatio** (double): De verhouding tussen de breedte en hoogte van de viewport (voorvlak van het frustum, op afstand `dNear` van het eyepoint).

Ter verduidelijking is hieronder een volledige `General` sectie gegeven.

```
[General]
size = 1024
backgroundcolor = (0, 0, 0)
type = "ZBuffering"
clipping = TRUE
eye = (0,0,0)
viewDirection = (1,0,0)
dNear = 1
dFar = 1000
hfov = 90
aspectRatio = 1.33
nrFigures = 1
```

Integratie

De nieuwe figuren worden in komende opdrachten niet meer uitgebreid. Hoewel ze nog kunnen voorkomen in voorbeeldafbeeldingen, zijn ze niet noodzakelijk voor het evalueren van latere opdrachten; een onvolledige of ontbrekende implementatie heeft geen verdere gevolgen. Hetzelfde geldt voor het view frustum. Voor diffuse en speculaire belichting in de volgende opdracht is wat extra code vereist om te werken met clipping. Hiervoor zullen aparte voorbeeldbestanden voorzien zijn. Een klein deel van het punt voor deze opdracht wordt pas toegekend als clipping met belichting correct werkt.

Tips

- Het is niet nodig om het genereren van een 3D-fractaal voor elk platonisch lichaam apart te implementeren. Voor het genereren van een 3D-fractaal worden er namelijk enkel schalingen en verschuivingen uitgevoerd op een aantal kopieën van de oorspronkelijke figuur. Als je engine in staat is om een gegenereerd lichaam te kopiëren (als de copy-constructor is geïmplementeerd) kan de fractaal dus berekend worden onafhankelijk van de oorspronkelijke figuur.
- Op BlackBoard zijn er een aantal voorbeeld configuratiebestanden te vinden samen met de verwachte output. Deze kun je gebruiken om je engine om de correctheid van je engine te verifiëren. Daarnaast kun je ook de voorbeelden die in de cursus worden gegeven implementeren.