Computer Graphics: 3D Lichamen

Ruben Mennes
Robin Verschoren

3D Lichamen (0.75 punten)

1. Implementeer een klasse voor het genereren van ruimtelijke figuren. Deze moet minstens in staat zijn om onderstaande 3D-lichamen te genereren:

   - Kubus (Cube)
   - Tetrahedron
   - Octahedron
   - Icosahedron
   - Dodecahedron
   - Cilinder (Cylinder)
   - Kegel (Cone)
   - Bol (Sphere)
   - Torus

   Elk van deze 3D Lichamen moet 'rond' de oorsprong (0, 0, 0) worden gegenereerd. Verder mogen de vlakken van de 3D figuren nog *niet* in driehoeken worden onderverdeeld. Voor de platonische lichamen en de bol moeten de punten geplaatst worden zoals ze in hoofdstuk 9 van de cursus worden gespecificeerd.

   De cilinder moet rond de Z-as worden gegenereerd met het centrum van het grondvlak in het punt (0, 0, 0). Verder moet het centrum van bovenvlak op de positieve helft van de Z-as liggen. Voor het genereren van de kegel geldt dat het grondvlak van de kegel in het XY-vlak moet liggen en dat de top van de kegel op de positieve helft van de Z-as moet liggen. De torus moet rond de Z-as worden gegenereerd en in het midden door het XY-vlak gaan. Voor meer informatie over het genereren van de kegel, de cilinder en de torus wordt verwezen naar de slides die het practicum begeleiden.

2. Voeg de nodige functionaliteit aan je grafische engine toe om *wireframe* tekeningen van bovenstaande 3D Lichamen te kunnen genereren. Net zoals met de vorige opgave moet een tekening meerdere figuren kunnen bevatten en elk van deze figuren moet geschaald, geroteerd en/of verplaatst kunnen worden vóór de projectie wordt uitgevoerd. Verder is het mogelijk dat er in één tekening zowel 3D Lijntekeningen als 3D figuren voorkomen.

Invoerformaat

De configuratiebestanden voor deze opgave bestaan uit twee delen. Het eerste deel beschrijft de `General` sectie. Het tweede deel bespreekt de beschrijving van de verschillende ruimtelijke figuren.

De `General` sectie

De `General` sectie van het configuratiebestand bevat dezelfde velden als de opgave van 3D lijntekeningen:

- **type** (string): Dit veld (dat in alle configuratiebestanden aanwezig is) bevat (voorlopig) nog altijd de waarde 'Wireframe'.
- **size** (integer): De grootte van de afbeelding in de richting die de meeste pixels vereist.
- **eye** (tuple van 3 doubles): Dit veld bevat een tuple van 3 reële waarden die de cartesische coördinaten van het eye aangeven.
- **backgroundcolor** (tuple van 3 doubles): De RGB waarden van achtergrondkleur. De waarden liggen tussen 0 en 1.
- **nrFigures** (integer): Het aantal figuren die in de .ini-file beschreven worden.

Een voorbeeld van *enkel* de `General` sectie wordt hieronder gegeven:

```
[General]
type = "Wireframe"
size = 750
eye = (150.0, 75.0, 75.0);
backgroundcolor = (0.0, 0.0, 0.0)
nrFigures = 2
```

Specificatie van 3D figuren

De verschillende ruimtelijke figuren die in de tekening voorkomen worden gespecificeerd in de secties `Figure0` t.e.m. `Figure<nrFigures-1>`. Elk van deze secties specificeert de configuratie van één van de figuren die moet worden getekend. Net zoals bij de opdracht 3D Lijntekeningen bevat elk van deze `Figure*` secties bevat *minstens* de volgende velden:

- **type** (string): Het type van de figuur. Dit veld specificeert wat voor soort figuur er moet worden getekend. De waarde van dit veld is afhankelijk van de figuur.
- **scale** (double): De factor waarmee de figuur moet geschaald worden. Deze transformatie moet eerst worden uitgevoerd.
- **rotateX** (double): De hoek (in graden) waarover de figuur moet worden gedraaid om de x-as. Deze transformatie moet worden uitgevoerd na de schaling.
- **rotateY** (double): De hoek (in graden) waarover de figuur moet worden gedraaid om de y-as. Deze transformatie moet worden uitgevoerd na de rotatie om de x-as.
- **rotateZ** (double): De hoek (in graden) waarover de figuur moet worden gedraaid om de z-as. Deze transformatie moet worden uitgevoerd na de rotatie om de y-as.
- **center** (tuple van 3 doubles): De cartesische coördinaten van het punt waarheen de figuur moet worden verplaatst. Deze transformatie moet als laatste worden uitgevoerd.
- **color** (tuple van 3 doubles): De RGB waarden van de lijnen van de figuur. Elk van deze waarden ligt tussen 0 en 1 (inclusief).

In de rest van deze sectie wordt het input-formaat voor de verschillende 3D Lichamen nader besproken.

Platonische lichamen
Voor de platonische lichamen worden enkel de velden gebruikt die hierboven beschreven worden. Het `type`-veld kan de waarden '*Cube*', '*Tetrahedron*', '*Octahedron*', '*Icosahedron*' en '*Dodecahedron*' aannemen. Een voorbeeld wordt hieronder gegeven:

```
[Figure0]
type = "Cube"
scale = 1
rotateX = 0
rotateY = 0
rotateZ = 0
center = (10, 0, 10)
color = (1, 0, 0)
```

Cilinder en Kegel
Voor een cilinder zal het `type`-veld gelijk zijn aan '*Cylinder*'. Voor een kegel zal het `type`-veld gelijk zijn aan '*Cone*'. In beide gevallen worden de volgende velden naast de standaard velden gespecificeerd:

- **n** (integer): Het aantal hoekpunten dat moet worden gebruikt om het cirkelvormige vlak te benaderen.
- **height** (double): De hoogte van cilinder of kegel.

De straal van het grondvlak wordt altijd verondersteld gelijk te zijn aan 1.

Een voorbeeld wordt hieronder gegeven:

```
[Figure1]
type = "Cone"
scale = 1
rotateX = 0
rotateY = 0
rotateZ = 0
center = (10, 0, 10)
color = (1, 0, 0)
n = 30
height = 10.0
```

Bol
Voor een bol zal het `type` veld gelijk zijn aan '*Sphere*'. Naast de standaard velden wordt ook nog het volgende veld gespecificeerd:

- **n** (integer): Het aantal iteraties dat moet worden uitgevoerd tijdens het genereren van de bol uit een icosahedron. Voor het geval $n = 0$ moet een normale icosahedron worden gegenereerd.

Een voorbeeld wordt hieronder gegeven:

```
[Figure1]
type = "Sphere"
scale = 1
rotateX = 0
rotateY = 0
rotateZ = 0
center = (10, 0, 10)
color = (1, 0, 0)
n = 5
```

Torus
Voor een torus zal het `type` veld gelijk zijn aan '*Torus*'. Naast de standaard velden wordt ook nog de volgende velden gespecifiëerd:

- **R** (double): De afstand van het middelpunt van de figuur tot aan het middelpunt van de buis van de torus.
- **r** (double): De straal van de buis van de torus.
- **n** (integer): Het aantal punten dat wordt gebruikt om de 'buis' van de torus in de horizontale richting te benaderen. (Zie slides)
- **m** (integer): Het aantal punten dat wordt gebruikt om de 'buis' van de torus in de verticale richting te benaderen. (Zie slides)

Een voorbeeld wordt hieronder gegeven:

```
[Figure1]
type = "Torus"
scale = 1
rotateX = 0
rotateY = 0
rotateZ = 0
center = (10, 0, 10)
color = (1, 0, 0)
R = 5.0
r = 1.0
n = 30
m = 30
```

3D L-Systemen (0.25 punten)

Implementeer een klasse voor het genereren van 3D L-Systemen. Deze moet in staat zijn om op basis van de specificatie van een 3D L-Systeem (Alphabet, Draw Functie, Replacement Rules, Initiator, Angle en aantal Iterations) uit een L3D-bestand in te lezen en vervolgens om te zetten naar een 3D *lijntekening*. Voor het inlezen van de L3D-bestanden kun je gebruik maken van de `LSystem3D`-klasse die samen met de 2D L-Parser op Blackboard te vinden is. Voor het omzetten van een `LSystem3D`-object naar een lijst met reële lijnen (deze keer in 3 dimensies) kun je gebruik maken van de algoritmes die in de cursus worden besproken. Bij het genereren van 3D L-Systemen moet je ervoor zorgen dat de 'Turtle' vertrekt in de oorsprong (0, 0, 0) en met initiële H-, L-, en U-vectoren respectievelijk gelijk aan de eenheidsvectoren op de X-as (1, 0, 0), Y-as (0, 1, 0) en Z-as (0, 0, 1).

Hou er ook rekening mee dat eens een 3D L-Systeem gegenereerd is, het net zoals alle andere soorten lijntekeningen een gewone ruimtelijke figuur is. Dit betekent dat 3D L-Systemen getransformeerd kunnen worden en dat een 3D L-Systemen samen met andere ruimtelijke figuren in een ini-bestand kunnen voorkomen.

Invoerformaat

3D L-Systemen zijn, net zoals 3D lijntekeningen gewone 3D Figuren. Dit betekent dat de gebruikelijke `General` sectie, zoals beschreven in **??**, gehanteerd wordt. Voor elk 3D L-Systeem dat gerenderd moet worden wordt een `Figure*` sectie in het ini-bestand voorzien. Voor 3D L-Systemen is het `type` veld van de figuur gelijk aan '*3DLSystem*'. De `Figure*` sectie bevat naast de

gebruikelijke velden (`type`, `scale`, `rotateX`, `rotateY`, `rotateZ`, `center`, `color`) ook hetvolgende veld:

- **inputfile** (string): De naam van het L3D bestand waaruit de specificatie van het 3D L-Systeem kan worden ingelezen.

Een voorbeeld wordt hieronder gegeven:

```
[Figure0]
type = "3DLSystem"
inputfile = "Fig20_1.L3D"
scale = 1
rotateX = 0
rotateY = 0
rotateZ = 0
center = (0, 0, 0)
color = (1, 0, 0)
```

Tips

- Controleer dat tijdens het genereren van de 3D Lichamen de punten van een oppervlak in *tegenwijzerzin* worden opgegeven. Anders kun je onverwacht gedrag krijgen bij het implementeren van de volgende opdrachten.
- Ook voor deze opdracht zijn op blackboard zeer veel voorbeeld ini-configuratiebestanden te vinden, samen met de verwachte output. Je kunt deze gebruiken om te controleren of je engine naar behoren werkt.

Extra oefeningen

De volgende oefeningen zijn *niet* verplicht en de oplossingen worden *niet* gequoteerd.

- Door meerdere 3D Lichamen samen te stellen kunnen complexere ruimtelijke figuren worden opgebouwd. Stel, op basis van bovenstaande 3D Lichamen, onderstaande ruimtelijke figuren samen. Deze kun je later eventueel gebruiken om je grafische engine in het project van het vak Software Eengineering te integreren.
  - *Schip*: Een schip kan op vele manieren uit basisfiguren worden opgebouwd, afhankelijk van welk soort schip je wilt voorstellen. Een schip bestaat uit een romp met daar bovenop minstens een commando post en eventueel enkele kannonnen en/of raketinstallaties. In zijn meest eenvoudige vorm bestaat de romp van het schip uit een balk van de juiste afmetingen met een punt aan de voorkant. Deze balk kan, afhankelijk van het soort schip ofwel gebaseerd zijn op een rechthoek (vier punten) of een driehoek. Voor het aanmaken van de punt kun je je baseren op de tetrahedron of op een kegel (met slechts drie of vier punten). De commandopost kan eveneens door een balk worden voorgesteld, hoewel ingewikkeldere vormen eveneens mogelijk zijn.
  - *Kanon*: Een kanon kan zeer eenvoudig worden voorgesteld zuiver met bollen en cilinders. Het kanon zélf kan worden voorgesteld door één enkele bol en één enkele cilinder. Hierbij moet de cilinder dezelfde straal hebben als de bol en moeten beiden zó gepositioneerd worden dat de onderste helft van de bol door de cilinder verborgen wordt. Voor de loop kun je vervolgens een langere en smallere cilinder gebruiken.

- *Kogel*: Een kogel kan in zijn meest eenvoudige vorm door een bol worden voorgesteld. Meer ingewikkelde vormen zijn uiteraard ook mogelijk
- *Raket*: Een raket kan worden opgebouwd uit een cilinder met daarbovenop een kegel. voor de stuurvlakken kun je, indien gewenst, hele dunne balken gebruiken.

Op gelijkaardige wijze kun je ook tal van andere figuren opstellen.

- In de slides die dit practicum begeleiden werd uitgelegd hoe een torus gegenereerd kan worden op basis van zijn parameter vergelijkingen. Op gelijkaardige wijze kunnen ook andere 3D Lichamen worden gegenereerd. Implementeer, voor elk van de onderstaande ruimtelijke figuren, een functie die deze figuur kan generen op basis van zijn parameter vergelijkingen.
  - Möbiusband: $\begin{cases} x_{u,v} = (1 + \frac{1}{2}v\cos\frac{1}{2}u)\cos u \\ y_{u,v} = (1 + \frac{1}{2}v\cos\frac{1}{2}u)\sin u \\ z_{u,v} = \frac{1}{2}v\sin\frac{1}{2}u \end{cases}$
  - Navelvormige Torus: $\begin{cases} x_{u,v} = \sin u(7 + \cos(\frac{u}{3} - 2v) + 2\cos(\frac{u}{3} + v)) \\ y_{u,v} = \cos u(7 + \cos(\frac{u}{3} - 2v) + 2\cos(\frac{u}{3} + v)) \\ z_{u,v} = \sin(\frac{u}{3} - 2v) + 2\sin(\frac{u}{3} + v) \end{cases}$