Computer Graphics: Z-Buffering met driehoeken

Jakob Struye
Tim Leys

Z-Buffering met driehoeken (1.5 Punten)

1. Pas je engine aan zodat deze in staat is om alle gekende 3D Lichamen *getrianguleerd* te genereren. Dit betekent dat de gegenereerde 3D Lichamen enkel uit driehoeken mogen bestaan. Dit kun je doen door ofwel bij het genereren van de figuur ineens de gebruikte veelvlakken door driehoeken te vervangen, ofwel door de 3D Lichamen te *trianguleren* ná het genereren. Hou er echter rekening mee alle vorige opgaves nog correct moeten werken: daar mogen de figuren niét getrianguleerd worden. 3D Lijntekeningen en 3D L-Systemen moeten *niet* getrianguleerd kunnen worden gezien deze figuren uit lijnen en niet uit vlakken bestaan.
2. Breid je engine uit zodat deze Z-Buffering met driehoeken ondersteunt.

Invoerformaat

Het invoerformaat voor Z-Buffering met driehoeken is nagenoeg hetzelfde als het invoerformaat als voor Z-Buffering met lijnen. Het enige verschil is dat het `type`-veld uit de `General`-sectie de waarde '*ZBuffering*' heeft ipv. '*ZBufferedWireframe*'.

Tips

- Het implementeren van Z-Buffering met driehoeken kun je best in twee stappen doen. Zorg er eerst voor dat je engine driehoeken kan inkleuren. Eens dit in orde is kun je de nodige functionaliteit toevoegen om de inverse Z-waarde voor elke pixel te berekenen.
- Bij het tekenen van de objecten zonder belichting zal je enkel de contouren van de getekende objecten kunnen zien. Om je code toch te kunnen testen kan je zelf een andere kleur geven aan de verschillende vlakken van de figuren. Zorg er wel voor dat dit bij het insturen van de opgave niet meer gebeurt. Je kan dit desnoods optioneel maken door een parameter toe te voegen aan het configuratiebestand.
- Je kan voorbeeldbestanden vinden op Blackboard.

Exra: Backface culling

1. Voeg backface culling aan je engine toe.
2. Ga de impact van deze optimalisatie na, in runtime van je engine. Meet hiervoor de runtime van zowel je volledige engine als enkel het codeblok waarin backface culling toegevoegd werd, beide met en zonder backface culling.
   Heeft de optimalisatie impact op de runtime van het codeblok waarin het voorkomt? Was de relatieve runtime van dat blok groot genoeg om je engine merkbaar sneller te maken?

3. Momenteel kun je aannemen dat de achterkant van een vlak nooit zichtbaar is, gezien we enkel met gesloten figuren werken. Als je later clipping implementeert, vervalt deze aanname en mag je geen backface culling meer toepassen Hou er dus rekening mee dat je backface culling moet kunnen uitschakelen (evt. gewoon door de relevante code weg te commentariëren).