Et bilde som inneholder sort, mørke

Automatisk generert beskrivelse

Fakultet for Audiovisuelle Medier og Kreativ Teknologi

**827**

**Mappeoppgave**

**Rapport**

Visualisering og simulering

2VSIM101

**2023**

1. **Introduksjon**

Jeg skal skrive kode som plasserer punkter i en punktsky rundt om i en 3D-scene. En annen del med kode jeg skal skrive skal konstruere et plan via triangulering, som en eller flere kuler skal rulle over. Kulene skal kunne kollidere med hverandre. Siden valg av program eller spillmotor var valgfritt gikk jeg for spillmotoren Unity, som bruker et koordinatsystem med Y som peker oppover. Rekkefølge for indeksering av vertices og dermed triangel naboer må legges inn i rekkefølge som går med klokken, siden det er rekkefølgen Unity leser av indeksering for vertices.

1. **Metode**

Jeg begynner med å få tak i punktskydataen jeg skal bruke for å rendere punktskyen og definere høyden til den genererte flaten. Her laster jeg ned en LAS-fil og konverterer den til en txt-fil ved bruk av programmet *Laszip*, slik som er bedt om i oppgaveteksten. I txt-filen *Laszip* skriver ut dukker hvert punkt opp på sin egen linje med XYZ-koordinater. Jeg lager en klasse *Punktsky* som leser gjennom txt-filen og lagrer alle punktdataene i en Vector3 array *points*. Siden punktene sine koordinater er veldig lagt unna origo, lagrer jeg også de største og minste verdiene for X, Y, og Z i respektive float variabler *xMin*, *xMax*, osv. Ved å ta gjennomsnitt av *Min* og *Max* verdiene for XYZ får jeg sentrert alle punktene rundt origo i scenen. Siden jeg bruker Unity er det også nødvendig å benytte seg av enten *Direct GPU Instansing* eller *Indirect GPU Instansing* for at programmet skal ha en brukbar bildefrekvens. I mitt tilfelle valgte jeg å bruke *Indirect GPU Instancing* siden det gir noe høyere bildefrekvens. Disse punktene skal brukes til å bestemme høyden for flaten jeg konstruerer.

I en ny klasse skal jeg konstruere en kvadratisk flate satt sammen av mindre kvadrater, som igjen er satt sammen av triangler. For å kunne sette sammen flaten må jeg lagre noen forskjellige typer data. Dataene som lagres for dette er vertex-data, indeks-data for hver vertex, triangel nabodata og uv-koordinater hvis man skal ha teksturer på flaten. Størrelsen på planet i XZ-retning bestemmer jeg ved å benytte den større av *xMax* eller *zMax* til en ny variabel *size*. Ved å dividere *size* på en annen variabel *resolution* får jeg et nytt tall som gir meg skrittlengen mellom hver vertex. *Resolution* bestemmer også hvor mange kvadrater i XZ-retning, som skal bygge opp flaten, altså oppløsningen til flaten. Høyden til hver vertex bestemmes ved at jeg sender inn hver vertex sin posisjon på XZ-planet og definerer et lite kvadratisk område rundt vertecen. Jeg bruker så barysentriske koordinater for å sjekke om det eksisterer punkter innenfor det kvadratiske området, dersom det eksisterer punkter innenfor kvadratet lagrer jeg høyde (Y) verdien til punktet i et array og tar gjennomsnittet av de lagrede verdiene på slutten av funksjonen og setter verticen sin høyde til gjennomsnittet.

Kulene som skal rulle/skli over flaten lages i en egen klasse/C#-fil *RollingBall* som har en referanse til en triangelflate klasse. For hvert nytt bilde som vises kalkuleres akselerasjonen og farten til kulen, dersom den er i kontakt med en flate kalkuleres også ut normalen til triangelet kulen er i kontakt med og projiserer fart vektoren mot triangelets normalvektor. For å sjekke om kulen er i kontakt med en flate må vi først ha posisjonen til kula og bruke barysentriske koordinater for å finne ut av hvilket triangel kula kan være i kontakt med og hvor på det triangelet kula ligger. De barysentriske koordinatene gir oss kontaktpunktet mellom kula og flaten, dersom distansen mellom kula og kontaktpunktet er mindre enn radiusen til kula i kontakt med flaten.

For at kulene skal kunne kollidere med hverandre må de vite radiusen til de andre kulene. Jeg lagrer alle kulene i et array *balls* i en ny klasse *BallManager*, og bruker to for-løkker på arrayet til å sjekke om to kuler er nære nok til å kollidere. Dersom de er nærme nok gjennomfører jeg en elastisk kollisjon. *BallManager* klassen står også for å plassere nye kuler på tilfeldige posisjoner over flaten, og hver gang den plasserer en ny kule legges den til i *balls* arrayet. Her sjekker den om distansen mellom de to kulene er større eller mindre enn radiusen til begge kulene lagt sammen. Om distansen er mindre enn radiusen til de to kulene lagt sammen gjennomføres kollisjonsresponsen.

1. **Resultat**

Punktene plasseres rundt om scenens origo. Siden jeg bruker *Indirect GPU Instancing* opplever jeg også at bildefrekvensen langt høyere enn om det ikke hadde blitt tatt i bruk.

Flaten blir konstruert til et uniformt kvadrat satt sammen av mindre kvadrater (quads) som igjen er satt sammen av to triangler. Variabelen *resolution* som brukeren kan sette i motoren bestemmer antallet quads i XZ-retning. Størrelsen til flaten bestemmes av punktsky punktet som er lengst unna origo i XZ-retning, dette kan gi ganske store flater da jeg har opplevd at noen av punktsky dataene kan være ganske store. Høyden settes riktig. Siden jeg har regnet ut UV-koordinatene til flaten kan en også legge til teksturer på den.

Et bilde som inneholder tegnefilm, tegning, maling, kunst

Automatisk generert beskrivelseEt bilde som inneholder tegnefilm, Animasjon, Fiktiv karakter, skjermbilde

Automatisk generert beskrivelse

*Figur x: flaten sett fra oven med wireframe* *Figur x: Samme som høyre, men uten wireframe*

Kulene ruller/sklir over flaten. Om jeg søker gjennom alle trianglene for å finne hvilke triangel ballen kolliderer med resulterer lavere bildefrekvens når det er flere kuler eller antallet triangler i flaten er stort. Bildefrekvensen blir bedre når triangel søket utnytter regulariteten av flaten og søker i nabotrekanter dersom det første søket ikke er riktig.

Sjekke for kollisjon mellom kulene funker som forventet. Kollisjonsresponsen resulterer i at kulene dytter jeg fra hverandre slik at de aldri overlapper, men denne funksjonaliteten gjør også at kulene kan bli dyttet eller sklir gjennom flaten.

1. **Diskusjon**
2. **Referanser**

Anders Åsbø

Andreas

Dag Nylund

One Lone Coder (youtube kanalen hans heter noe annet)

Hva som mangler og skal gjøres:

Tegne banen kulene tar

Kollisjon mellom kulene

Kulene ruller