Rapport, mappeinnlevering visualisering og simulering

Kandidat: 824

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 Unity, som bruker et koordinatsystem med Y som peker oppover. Rekkefølge for indeksering av triangulering i Unity gjøres også med klokken istedenfor mot klokken slik som vi gjorde i OpenGL.

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 X, Y og Z 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 vertecen 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 hvor på triangelet kula ligger, deretter tar jeg prikkprodukt av (barysentriske koordinater – kulas posisjon) \* normalvektoren til triangelet. Til slutt sjekker jeg om absolutt verdien til prikkproduktet er mindre eller lik en radiusen til kula, hvis den er mindre enn radiusen er 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.

Jeg lager en kvadratisk flate med triangulering, altså plassere vertexer, indekser for vertexene (og hvilke triangler som er nabo med hverandre (ikke implementert)). Størrelsen til planet bestemmes av punktskyens største X eller Z verdi. Brukeren kan bestemme oppløsning/antallet kvadrater som danner den sammensatte flaten, eksempel: Med oppløsning på 20 danner jeg en kvadratisk flate satt sammen av 20x20 kvadrater.

Høyden til hver vertex bestemmes ved at jeg lager et kvadratisk område rundt hver vertex i trianguleringen og ser så etter punkter som ligger innenfor kvadratet. Punktene som ligger innenfor kvadratet, blir lagt til i et array og jeg tar deretter gjennomsnittet av Y koordinaten (i spillmotoren Unity er Y koordinaten høyde koordinaten) og setter vertexen sitt Y koordinat lik gjennomsnittet av punktene funnet.

Simulerer kuler/regndråper som kan skli/rulle rundt på planet. Kulene spawner innenfor et gitt område. De kan kollidere med hverandre, men jeg mangler (Fjerne før innlevering)

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 i et kvadrat

Ballene ruller/sklir over

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

Hva som mangler og skal gjøres:

Tegne banen kulene tar

Kollisjon mellom kulene

Kulene ruller