Visualisering og Simulering Eksamen

Innhold

[Introduksjon 3](#_Toc183974906)

[Metode 3](#_Toc183974907)

[1.1 3](#_Toc183974908)

[1.2 3](#_Toc183974909)

[1.3 4](#_Toc183974910)

[2.1 4](#_Toc183974911)

[2.2 5](#_Toc183974912)

[Resultater 5](#_Toc183974913)

[Diskusjon 5](#_Toc183974914)

# Introduksjon

For denne eksamen hadde vi totalt 11 oppgaver som skulle gjøres og de falt under 2 kategorier; konstruksjon og oppbygging av 3D terreng, og simulering. I denne eksamen så har jeg gjort alle oppgavene etter oppgave teksten med unntak av konstruksjon av 3D terreng ved bruk av bi kvadratisk tensorprodukt B-spline flater.

Flere av oppgavene i denne eksamen er tatt fra tidligere arbeid fra andre eksamener eller obligatoriske oppgaver, og en oppgave har jeg brukt et eksternt bibliotek som hjelpemiddel, og litt AI for en løsning til et stabilitets problem jeg hadde.

For noen av oppgaven så viste jeg hva jeg måtte gjøre fra før av, slik som å lage punktsky, eller å bruke barysentriske koordinater til å sjekke trekanter, men andre oppgaver så måtte jeg finne kreative løsninger, slik som Delaunay triangulering eller bruk av B-spline sporing.

# Metode

## 1.1

For denne oppgave så valgte jeg å laste ned punktdata fra en bydel i Trondheim for å bruke i mitt prosjekt, jeg brukte LAS filformat og konverterte dataen til å være z, y og z koordinater. Punktskyen var veldig stor så jeg reduserte størrelsen slik at den lastet inn raskere, og at den kunne bli sent opp til GitHub. I innleveringen så har jeg 2 punktsky filer, en som heter «Trondheim\_punkt\_sky\_comp3» og «Trondheim\_punkt\_sky\_comp6» den første var redusert 3 ganger, og den andre var redusert 6 ganger.

## 1.2

For å få datapunktene til å vises i rendervinduet så måtte noen modifikasjoner til min modell-kode gjøres, men det hadde blitt gjort for Obligatorisk oppgave 2, så brukte samme metode som var å legge punktene fra punktskyen in som vertecies til punktsky modellen, og ikke legge til indecies til modellen. Og når jeg kjører for å tegne modellen så sjekker jeg om indecies er tom, og hvis de er det så tegnes modellen som punkter istedenfor triangler.

## 1.3

For å gjøre trianguleringen for punktskyen så endte jeg opp med å bruke Delaunay triangulering, etter flere forsøk og diskusjoner med andre medelever så brukte jeg et bibliotek som allerede fantes på github som het delaunator-cpp laget av brukerne delfrr, flippmonke og mourner (Biloneko, Thompson, Agafonkin, 2018/2024).

Fra tidligere prosjekt så hadde jeg en måte å regne normaler på som jeg har brukt for alle prosjekt, denne måten var nødvendig for å regne ut normaler til bruk av Phong shader, shaderen har ikke gått gjennom noe endring siden 3D programmering vår 2024 (AslakDB, 2024).

## 2.1

For alle oppgavene under kategorien simulering så er simuleringen gjort på planet fra 1.3. Jeg hadde en ball klasse fra tidligere obligatoriske oppgaver fra spillmotor (AslakDB,2024,2) og brukte den til dette prosjekt.

Det som var viktigst i denne oppgaven var å bruke barysentriske koordinater til å finne ut hvilken trekant ballen jeg hadde var på. For denne oppgaven så kunne jeg ikke gjøre det slik som jeg hadde gjort før fordi hver ball gikk gjennom hver eneste trekant på planet for å sjekke om ballen var innenfor på en trekant eller ikke, og mengden trekanter på planet gjorde dette veldig ineffektivt.

Min initiale løsning var å lage et octree der en ball bare sjekket trekanter innenfor noden den var på, men bestemte meg imot det fordi jeg følte det var en overkomplisert måte å løse problemet på og ville finne en løsning som passet dette prosjektet bedre. Min løsning endte opp med å være å dele planet mitt in i et grid der en ball bare sjekket trianglene innenfor cellen den befant seg i. Jeg gjorde også slik at cellene ikke tok hensyn til y høyde, siden planet for dette prosjektet ikke hadde forskjellige etasjer.

Jeg følte at å gjøre planet grid basert istedenfor å dele planet opp i et octree var en bedre løsning fordi det var enklere og med den grid baserte løsningen så kunne man bestemme selv størrelsen på de individuelle cellene, som i mitt tilfelle var veldig liten, som gjorde at jeg kunne ha mange flere baller i prosjektet. Jeg brukte en del Ai til å hjelpe meg med å lage det grid baserte oppdelingen av planet, men endret det slik at det passet mitt prosjekt bedre.

Når jeg fikk fikset på barysentriske koordinatene fikk jeg først ballene til å sprette på bakken basert på normalen til de individuelle trekantene. Deretter så regnet jeg ut hvordan ballen burde reagere basert på retnings fart, masse og tyngdekraft slik at jeg fikk simulert rulling mere nøyaktig.

## 2.2

Interaksjon i prosjekter ev veldig simpelt. Jeg posisjonen på en av ballene og hvis man trykker på O knappen så bli ballens nye posisjon hvor enn kameraet sin posisjon er, og ballen får også fart i retningen som kameraet ser på, hvis man trykker på I så får du det samme, men farten er 0, trykker man på L så setter du posisjonen til en annen ball til å være på kameraet.

## 2.3

Friksjon er iverksatt i planet og ballens fysikk. Når vi lager planet så sjekker vi om punktet som blir lagret for punktsky er større en -25 i world space, hvis det er større så er friksjonen høyere og fargen på den vertecien blir satt til rød, og hvis den er mindre så er friksjonen laver og fargen blir satt til grønn. Når vi sjekker for barysentriske koordinater så sjekker man også trekanten sitt friksjonstall og bruker det tallet til å regne fysikken på ballen.

## 2.4

Kollisjon er den samme som ble brukt for spillmotor obligatorisk oppgave 2. Det er en funksjon som tar inn en vektor baller, så sjekker den alle ballene mot hverandre, men ikke seg selv, deretter sjekker den om distansen er mer en radiusen på ballene, så beregner den normalen og reflekterer hastigheten til ballen slik at de spretter av hverandre riktig.

## 2.5

B-spline sporingen blir gjort med at hver 20 frame i render så lagres possijonen til ballen som et punkt som kan bli brukt for å gjøre B-spilne funksjonen, funksjonen trenger minst 4 punkter før den kan begynne å regne ut B-spline sporingen. For å vise hvordan sporingen blir regnet ut så ligger den litt bak selve ballen

## 2.6

I prosjektet så har jeg 20 baller som blir sporet, for å få så mange baller så måtte jeg lage en vektor av punkter for hver ball modell, og får å ikke overbelaste programmet så sletter jeg de lagrede punktene når modellene sine vektorer bli større en 15 punkter

# Resultater

// resultater + bilder + hvordan jeg følte at dette var best

# Diskusjon

// hva kunne jeg ha gjort annerledes