VisSim Oblig 3

Innledning

Dere kan bruke Qt-prosjektet VSIM101_H22_Rulleball_0 til obligatorisk oppgave 3.

For enkelhets skyld ligger dette som en zip-fil i Canvas. Klassene er kjent fra 3D-programmering. Egne matrise-og vektor-klasser benyttes. Det skal være kompilerbart. En trekant ate som består av to trekanter leses inn fra datafil, som på figuren nedenfor.

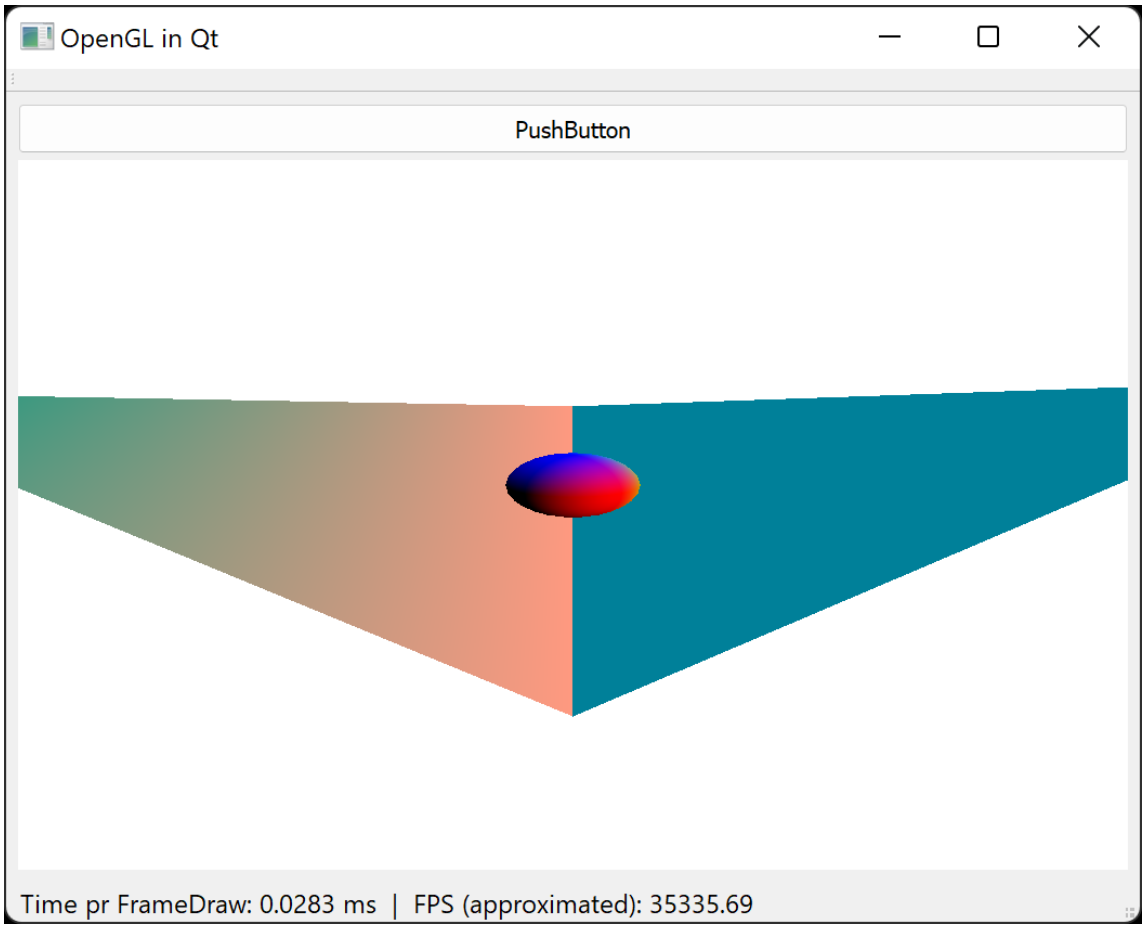
Kamerastyringen er primitiv (AWSDQZ).

- Hvis dere bruker dette prosjektet som utgangspunkt for oblig3, blir oppgaven i hovedsak å implementere move() som forklart i kapittel 8.

- Datafilen totrekanter.txt erstattes med deres egen datafil fra oblig1.

- Dere må erstatte medfølgende enkle shaderfiler med egne shaderfiler som har andre navn. Det trenger ikke være avansert, men det skal være deres eget.

- Husk å oppdatere navn til datafil og shaderfiler i renderwindow.cpp.



Oppgave 8.9.4 i forelesningsnotater. med følgende presiseringer.

1. Bestem koordinatene til to eller flere trekanter og angi passende høydeverdier. Velg enkle tall slik at det blir lett å regne manuelt. Lag manuelt en tekstfil med vertex data og naboinformasjon som i Matematikk3, og les inn data fra fil slik at denne oppgaven kan utvides til et større datasett.

   Presisering:
   Datasettet skal være nøyaktig som i oblig1. Dere skal altså bruke 4 trekanter, og har allerede gjort noen av de manuelle utregningene. Videre, dere **skal bruke glDrawElements**. Legg også til friksjon slik at rullingen fra oblig 1 simuleres.

2. Implementer en klasse med navn f.eks GravitasjonsBall/RollingBall som arver OktaederBall (fra 3D-programmering, slik at rendring er impementert) og har en peker til et TriangleSurface objekt.

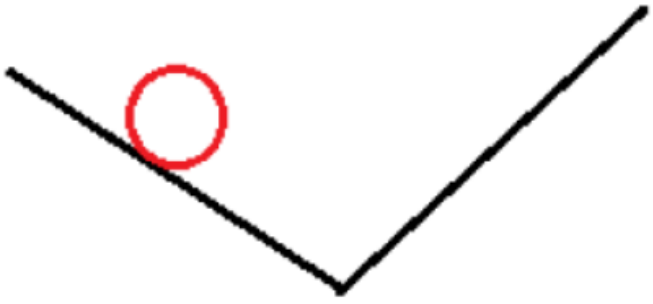
   Presisering:
   Dette er en anbefalt klassestruktur, men ikke noe absolutt krav.

3. Vi trenger en funksjon som gitt ballens koordinater (senter) finner hvilken trekant ballen beinner seg på (vi trenger vertexene) og normalvektoren. Implementer denne.

4. Implementer rulling fra et triangel over på det neste som på figur 8.9.

8.8 Rulling på triangler

TriangleSurface består av triangler med ulike normalvektorer. Når ballen ruller på et gitt triangel og kommer i berøring med et annet triangel med en annen normalvektor, kan det betraktes som en kollisjon. Hvis normalvektorene er ulike og det oppstår en kollisjon, skal opplagt ballens hastighetsvektor endre seg. Figurene 8.9 og 8.10 viser to ulike situasjoner. I tillegg kan det oppstå spesialtilfeller.



Figur 8.9: Overgang fra et triangel til et annet over en felles kant - 1.

På figur 8.9 er ballen på veg nedover et plan, og $\vec{v}$ vil opplagt endre retning når den skal rulle oppover det høyre planet. På figur 8.10 derimot er ballen på veg oppover planet og det inntreffer ingen kollisjon umiddelbart når den passerer toppen. Den vil falle fritt en liten stund og falle ned igjen - da får vi en kollisjon.

5. Implementer move() funksjonen med oppdatering av hastighet og posisjon (akselerasjon kan være konstant).
6. Tilordne ballen en passende initiell posisjon og hastighet. Simuler rulling over to eller flere trekanter.

   Presisering:
   Det er enklest å bruke $v_0=0$. Ballen skal altså rulle over 4 trekanter, og det skal være større friksjon på en av dem (se oblig 1).
7. Legg til en ekstra ball og implementer kollisjonsdetektering og kollisjonsrespons mellom ballene. (Dette er en utfordrende tilleggsoppgave som krever litt selvstudium) .

Dere kan arbeide og levere individuelt eller i grupper (2-3).
Bruk Git. Innleveringen for denne oppgaven skal bestå av:

- Lenke til fungerende prosjekt på Git.

- Rapport skrevet i Word eller Latex med utregning av ballens bane (bilde av håndskrevet utregning kan importeres) og kildereferanser.

- Kort videoforklaring (2-5 minutter).