Sop – simulation af roterende objekter

Indholdsfortegnelse

**Resume**

**Indledning**

**Metode**

**Inertimoment**

**Programmering**

**Sammenligning/test**

**Konklusion**

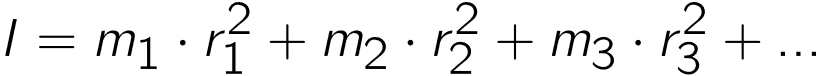
**Litteraturliste**

**Bilag**

okay, så what im going to do is to talk about the physics formulas, and then do some experiments to do the physics (or prove it). Also what is inertimoment og hvordan er det andreledes fra inerti? And then do some talking about programming and the programming way. And then after all the procrastination, do the talk about the simulation and how it works and then what you can do with it and then I would wish you could have animations in pdfs, because if you cannot, then that would be quite succc.

I fysik kender vi begrebet inertimoment som er et elements modstand mod at påbegynde en rotation. Altså hvis vi har et objekt som er stille på en akse og vi så begynder at rotere det objekt, så vil det kræve noget energi fra os. Men forskellige objekter kommer til at kræve forskellig mængder energi for at kunne komme op på samme rotationshastighed. Derfor bruger vi inertimoment til at beskrive hvor meget energi man skal bruge for at lave en vis rotationsændring.

Det er let nok at regne ud at inertimoment er afhængig af vægten af objektet siden at meget tunge objekter er sværere at rotere end lette objekter og udover det er distancen mellem massen og rotationsaksen også en del af inertimomentet. Da inerti er bestemt ud fra både masse og massens position er vi nødt til at dele emnet op i flere dele. Hver del har en masse og en distance til rotationsaksen og hver del skal summeres sammen så formlen vi skal bruge til at finde inertimoment, ser sådan ud



Dette er dog ikke en særlig god måde at udregne inertimoment da det kræver en masse udregninger hvis man har et meget stort objekt og man gerne vil have et præcist resultat. Derfor er der lavet tabeller med lettere måder at udregne inertimoment for typiske (common) former.

Nogle af dem ser sådan ud:

I = 1/12\*m\*r^2 --- stang med rotationsakse i enden

I = 1/3\*m\*r^2 --- stang med rotationsakse i midten

… skrive noget mere om hvordan man bruger dem …

Dem vil jeg teste senere i opgaven med min simulation.

I programmering laver vi et sæt instruktioner som kan læses og udføres af noget hardware, som fx en computer. Når man laver et program, bruger man programmeringssprog som er lettere for mennesker at forstå og skrive, men ikke alle programmeringssprog er lige hurtige til at udføre opgaver. Om programmeringssprog bruger man udtrykkene ”high-level” og ”low-level” til at beskrive hvor maskinnære de er og det hænger typisk også sammen med hvor hurtigt de kører. Grunden til at man ikke bare bruger low level sprog til alle sine programmer er at det er mere kompliceret at lave programmer med dem. I stedet bruger man high level programmeringssprog der er lettere at skrive som kompileres til maskin-kode som computeren kører.

I dette projekt har jeg valgt at bruge JavaScript, mest fordi det er det jeg har arbejdet i for tiden, men det har også de fordele at det er let at køre på andre computere da de ikke skal for installere noget til at køre det, men blot gå ind på nicklasbns.me/sop for at køre koden. … noget om positiver og negativer om js.

Comp thinking

Når man har et prolem/opgave i programmering er det meget godt at dele det op i dele som man skal lave. Fx da jeg besluttede at jeg skulle lave en simulation begyndte jeg først at overveje hvad jeg skulle lave den i, altså hvilket programmeringssprog jeg skulle bruge. En af de ting jeg tænkte var gdScript som bruges til at kode i programmet godot (udtales go-dou), fordi det har indbygget funktioner til at udregne kræfter, kollisioner og vægte så det ville gøre det ganske let at ”lave” en simulation ud fra det, men det valgte jeg at undgå fordi jeg hellere ville gå lidt dybere i opgaven og lave udregningerne selv. Java og python var også programmeringssprog som jeg overvejede, men valgte at bruge javascript, mest fordi det er det jeg har arbejdet mest i for tiden og det ville virke fint til dette formål.

Så før jeg begyndte at skrive mit program gik jeg i gang med at overveje de dele af programmet jeg skulle have lavet. …som i kan se på det diagram jeg har lavet… Overordnet set har vi hele min simulation. Den simulation består så af en brugerflade, en renderer, og en logik ting. Som i kan se er selve simuleringen ikke en særlig stor del af at lave sådan et program. Brugerfladen har jeg valgt at dele op i to dele, input og output, input er alle de knapper og tekstfelter og museklik. De er vigtige for programmet for at kunne fortælle hvad man gerne vil have programmet skal vise. Input felter sender kun data til logik delen af programmet. Output er det der bliver vist til os på skærmen, som fx selve 3d objekterne, teksten der forklarer hvad de forskellige inputs gør, og ikoner der viser status af programmet. Rendereren er den der ”tegner” alle de dele som simulatoren udregner. Til den har jeg valgt at bruge biblioteket three.js fordi det gør det dejlig let at lave visuelle 3d scener. …snakke om simualtionen…

Jeg har lavet et simulations environment som fokuserer på rotation, kraftmoment, og inertimoment. Siden det er en simulation jeg laver, ville jeg undgå at bruge nogle biblioteker eller programmer som har fysik simulationer indbygget da det lidt ville ødelægge opgaven. Det jeg har brugt af biblioteker, er et 3d visualiserings bibliotek som hedder three.js og udnytter browseres WebGL engine, og så bruger jeg et lille bibliotek til at lave fine popup menuer, men som ikke er vigtigt til funktionen af simulationen.

Simulationen består af flere forskellige dele af programmet, der er selve hjemmesiden der styrer hvordan programmet ser ud og hvordan den skal kører JavaScript koden, der er setup koden som kun kører en gang når den skal generere selve udseendet og menuerne der skal bruges, der er simuleringskoden der kører hele tiden og er den der for ting til at bevæge sige korrekt, og så er der eventkoden som bliver kørt hver gang man fx trykker på skærmen.

Skal have skrevet test/presentation af simulationen

Skal have skrevet test af precision af simulationen + eventuelle fejlkilder

Skal have skrevet gå i dybden med hvordan jeg lavede den