Obligmal

Eirik Ovrum

September 6, 2016

1 Introduksjon

"The avengers of Ultron: The sliding game" er et spill hvor man skal prøve å skli en boks opp et skråplan så langt som mulig uten å falle over enden.

2 Fysikken

Fysikken som ligger til grunn for spillet er friksjon og tyngdekraft på en kloss på et skråplan.

2.1 Krefter på klossen

I figur 2.5 ser vi systemet satt opp, med kreftene som virker på klossen. Her peker friksjonskrafta feil vei, siden klossen vår skal skli oppover på skråplanet.

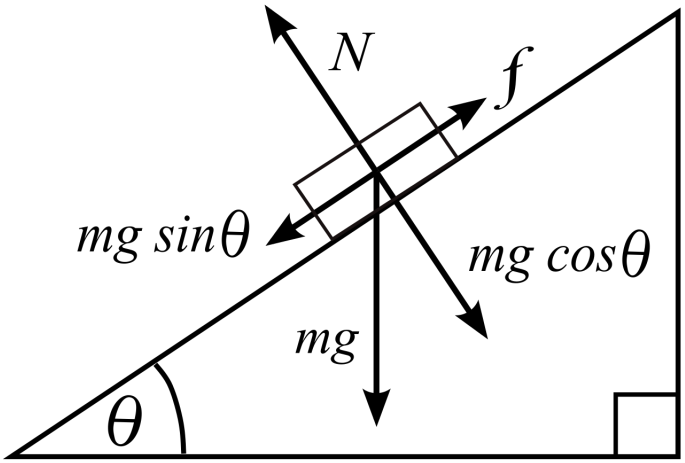

Figure 1: Kloss på skråplan. Her er er f friksjonskrafta når klossen sklir nedover skråplanet. I spillet derimot så sklir klossen oppover skråplanet, og $\vec{f}$ peker da motsatt vei, nedover skråplanet.

Kreftene som virker på klossen er tyngdekrafta ($\vec{G}$), normalkrafta fra underlaget ($\vec{N}$) og friksjonskrafta fra underlaget ($\vec{f}$).

2.2 Variabler og input til spillet

Det tingene vi må vite om klossen før vi kan simulere bevegelsen dens er

1. Massen, m, $[m] = \text{kg}$
2. Vinkelen skråplanet danner med horisontalen, θ
3. Startfarten oppover skråplanet, v_0, $[v_0] = \text{m/s}$
4. Den dynamiske friksjonskoeffisienten mellom klossen og underlaget, μ_k
5. Lengden av skråplanet, L, $[L] = \text{m}$.

2.3 Løsningsmetode

Måten vi simulerer bevegelsen på, er å bruke Newtons andre lov som gir akselerasjonen ut i fra kreftene. Og fra akselerasjonen, samt startfart og startposisjon, finner vi farta og posisjonen for ethvert tidspunkt.

Først velger vi oss et kartesisk koordinatsystem, med positiv x-retning oppover skråplanet, og med positiv y-retning normalt oppover fra skråplanet. Så dekomponerere vi kreftene i de to retningene og setter opp Newtons andre lov for hver av retningene.

Først for y-retning,

$$\Sigma F_y = N - G_y = N - G\cos\theta = 0 \Rightarrow N = G\cos\theta \tag{1}$$

som gir oss størrelsen på normalkrafta, siden akselerasjonen i y-retning er null. Som igjen gir oss friksjonskrafta,

$$f = \mu_k N = \mu_k mg\cos\theta. \tag{2}$$

Newtons andre lov i x-retning gir oss så akselerasjonen i x-retning.

$$\begin{aligned}\Sigma F_x = -f - G\sin\theta = -mg(\mu_k\cos\theta + \sin\theta) = ma_x \\ \Rightarrow a_x = -g(\mu_k\cos\theta + \sin\theta).\end{aligned} \tag{3}$$

Løsningen av dette problemet kan finnes i enhver fysikkbok for ingeniørstudenter eller i boka vi bruker i dette kurset, [1].

2.4 Bevegelsesligningene

Vi har her bevegelse med konstant akselerasjon, og trenger dermed ikke å bruke en numerisk løser for å integrere opp posisjonen og farten fra akselerasjonen. Vi har et tilfelle med kjent startposisjon: $x_0 = 0$ m, kjent startfart v_0 i x-retning. Samtidig så kjenner vi akselerasjonen, og vi finner da posisjonen som en funksjon av tiden:

$$x = x_0 + v_0 t + \frac{1}{2}a_x t^2. \tag{4}$$

Trenger vi farta, så er den

$$v = v_0 + a_x t. \tag{5}$$

2.5 Løsning for gitte startverdier

For å vise at spillet gir riktig fysikk så har jeg også løst problemet i octave, og tar med en figur som viser den riktige oppførselen til klossen. Under følger et utdrag fra octave fila, hele fila ligger i repositoriet under navnet *sb1.m*.

```
% I x' retning bruker vi N2.
a_x2 = 1/m*( G2(1) - f ); % akselerasjonen i x' retning, m/s^2

t = linspace(t_0, dt*t_N, t_N);
x2= zeros(t_N, 1)'; % En kolonnevektor med t_N nuller, x'
                    % posisjonen.
x= zeros(t_N, 1)'; % x posisjonen
y= zeros(t_N, 1)'; % y posisjonen


x2(1) = 0;
for i=2:t_N
    x2(i) = v_0 * t(i) + 1/2 * a_x2 * t(i)^2;
end

plot(t,x2)
```

Den figuren som kalles ved koden over er denne

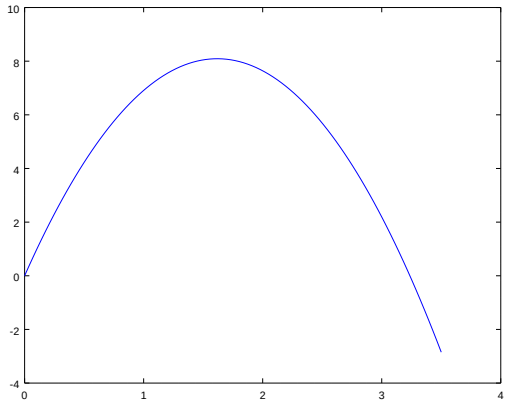

Figure 2: Her vises posisjonen i x-retning for klossen som en funksjon av tiden. Den vertikale aksen i figuren er altså hvor langt oppover skråplanet klossen har kommet. Som vi ser, så stopper den på rundt 8 m, mens skråplanet bare er 1 m langt, og vi taper dermed spillet med de startverdiene vi har brukt her.

3 Implementasjon

Hvordan har fysikken blitt implementert? Gjerne med pseudokode eller faktisk kode. Forklar hvordan dere har fått de ligningene som gir korrekt løsning inn i unity.

References

[1] Grant Palmer. *Physics for Game Programmers*. Apress, 2005.