**OBLIG 1**

**Andreas Holm**

**September 11, 2016**

# Introduksjon

«Turbo aerodynamic cow launcher» er et spill hvor man skal trene franskmenn til å beregne hvor stor hastighet ei ku må ha og hvilken vinkel den må kastes med for for å treffe usle engelske konger når man står i et 100m høyt tårn.

# Fysikken

Fysikken som ligger til grunn for spillet er hastigheten på kua og vinkelen den kastes med. Kua er dessverre særdeles aerodynamisk og den eneste akselerasjonen forekommer av tyngdekraften.

## Krefter på kua.

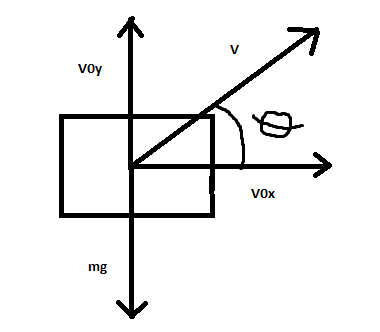


Figure 1: Ku i utskytningsøyeblikket. Farten i x retning er konstant. Farten i y retning vil avta da tyngdekraften fungerer i negativ retning.

Kreftene som virker på kua er tyngdekrafta og fart i X og Y retning.

## Variabler og input til spillet.

De tingene vi må vite om kua før vi kan simulere bevegelsen dens er.

1. Massen, m [m] = kg.
2. Vinkelen kua skytes ut med horisontalen, ø.
3. Initialfarten den skytes ut med v, [v] = m/s.

## Løsningsmetode

Måten vi simulerer bevegelsen på, er å bruke Newtons andre lov som gir akselerasjonen ut ifra kreftene. Og fra akselerasjonen, samt intialhastihet og initialposisjon, finner vi farta og posisjon for ethvert tidspunkt.

## 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: x0 = 0m, kjent startfart og vinkel som spilleren selv velger.

## Løsning for gitte startverdier

For å vise at spillet gir riktig fysikk har jeg forsøkt å løse problemet i octave, dessverre oppsto det problemer da jeg forsøkte å lage modellen som skulle demonstrere ruten kua tok. Jeg tar derfor bare med utdraget som viser at regnestykket går opp, men utelater å vise en figur inntil jeg får fikset problemet. Eventuelt bruker jeg matlab neste gang. Under følger utregningene ifra Octave fila.

%Finner Vy utifra V og sin theta.

%Antar at V er 50.

%Antar at Theta er 40.

vy =sind (40) \* 50;

vx =cosd (40) \* 50;

%Displacement i X rettning er -100m.

%Tyngeaksjelerasjonen er - 9.81 m/2^2

a = -9.81;

d = -100;

%Hva er Vf?

%Vf^2=Vi^2+2a\*d.

Vft = (vy)^2 +2\*(a)\*(d);

Vfy = sqrt(Vft);

Vfy \*= -1;

%Finner tiden.

t = (Vfy - vy)/a;

%Bruker tiden for å finne X pos

Vfx = (vx)\*(t);

# Implementasjon

Koden min benytter kua sin posisjon og vektorer for å beregne neste posisjon. Jeg har også lagt ved en klokke slik at man kan sjekke at avstand og tid henger sammen.

Spilleren starter spillet ved å velge initial hastighet og vinkel ved bruk av tastene a/d og w/s. Deretter har jeg laget en funksjon som omgjør vinkelen og hastigheten til en vektor jeg kaller «velocity».

void calculateVelocity()

{

float radAngle = angle \* Mathf.Deg2Rad;

vy0 = Mathf.Sin(radAngle) \* vel;

vx0 = Mathf.Cos(radAngle) \* vel;

iVel = new Vector3(vx0, vy0, 0);

velocity = (iVel);

}

Denne vektoren påvirkes av vektoren «acceleration» som er et produkt av summen av krefter.

void updateAcceleration()

{

acceleration = (netForce / mass);

}

void updateVelocity()

{

velocity += (acceleration) \* Time.fixedDeltaTime;

}

Posisjonen avgjøres av påvikningen ifra «velocity»

void updatePosition()

{

transform.position += (velocity \* Time.fixedDeltaTime);

verticalDisplacement = transform.position.y;

horizontalDisplacement = transform.position.x;

}

Når posisjonen på kua er -100m avsluttes spillet og du kan se avstand den har flydd og tiden det tok den å fly.

Dette spillet kan gjøres langt mer avansert med lite arbeid da den inneholder noe kode som ikke benyttes i den grad den burde, men den løser problemet den er satt til.

# 4.References

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