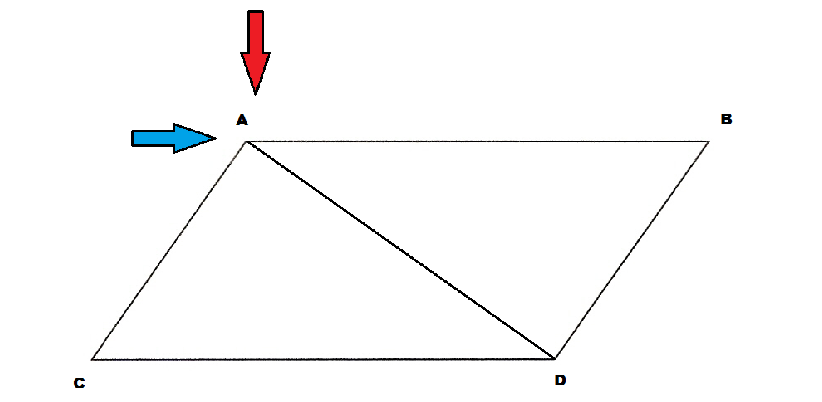
# Introduksjon.

<<Canonia>> er en simulator som simulerer en kanon og dens kuler i 2D.

## Fysikk og Ligninger.

Fysikken som ligger til grunnlag for spillet er fysikk for prosjektiler i to dimensjoner med luftmotstand.

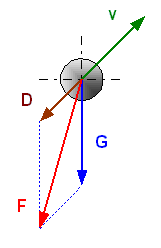
Newtons andre lov sier at ***summen av alle krefter*** er lik *masse* ***x*** *akselerasjon*. generelt kan dekomponeres til en kraft i y-retning og til en kraft i x-retning i 2 dimensjoner. Dette kan enkelt visualiseres ved et parallellogram.



Vis en kloss starter på posisjon ***A*** og en dytter klossen i den retningen den blå pilen viser vil pilen ende opp på posisjon ***B***.

Vis en kloss starter på posisjon ***A*** og en dytter klossen i den retningen den røde pilen viser vil pilen ende opp på posisjon ***C***.

Vis en kloss starter på posisjon A og en dytter klossen i begge retningene vil klossen ende opp på posisjon ***D***.



Her ser vi et bilde av alle kreftene som virker på en kanonball i fart. På grunn av ballens symmetri er luftmotstanden alltid i motsatt retningen av farten. Kraften til gravitasjonen som trekker ballen ned mot jorda er gitt ved navnet G. F er total summen av alle krefter, mens V er farten i en spesifikk retning.

Siden vi fra tidligere er klar over at summen av alle krefter i 2 dimensjoner kan dekomponeres i X- og Y- retning, blir formlene slik:

, hvor er summen av alle kreftene i x-retning og er luftmotstand i x-retning.

, hvor er summen av alle kreftene i y-retning, er luftmotstand i y-retning og , kraften til gravitasjonen.

D = , hvor D er luftmotstanden.

er lufttetthet, er farten, er luftmotstandskoeffisienten, og A er arealet til objektet.

Luftmotstand i x-retning og y-retning er gitt ved:

Tilslutt kan vi sette alt sammen og få (***akselerasjon i (x-retning) = kreftene i x retning)/m***, og akselerasjonen i y-retning er lik (***akselerasjon i (y-retning) = kreftene i y retning)/m-G).***

# Løsning ved kode.

Algoritmen som skal løse ligningene og finne den neste posisjonen, den neste farten og den neste akselerasjonen i et tidsbeløp heter Runga Krutta fjerde ordens metode. Runga Kutta fjerde ordens metode er en algoritme som approksimerer farten, posisjonen og akselerasjonen i en veldig liten tidsenhet fremover i tid. Dette gjør den om og om igjen, slik at vi får vite posisjon, fart og akselerasjon

Under ser vi en liten kodebit. Farten og hastigheten blir sendt inn som vx, og sx. Farten og hastigheten i y-retning blir sendt inn som vy og sy.

float v1x = vx;

float s1x = sx;

float v1y = vy;

float s1y = sy;

float a1x = FormelX(v1x, v1y); //Kalkulerer akselerasjon I x-retning

float a1y = FormelY(v1x, v1y); //Kalkulerer akselerasjon I y-retning

float v2x = (vx + (a1x \* dt \* 0.5f)); //Ny fart et halvt tidsteg fremover, x

float s2x = (sx + (v1x \* dt \* 0.5f)); //Ny posisjon et halvt tidssteg fremover, x

float v2y = (vy + (a1y \* dt \* 0.5f)); //Ny fart et halvt tidsteg fremover

float s2y = (sy + (v1y \* dt \* 0.5f)); //Ny posisjon et halvt tidssteg fremover

float a2x = FormelX(v2x, v2y);

float a2y = FormelY(v2x, v2y);