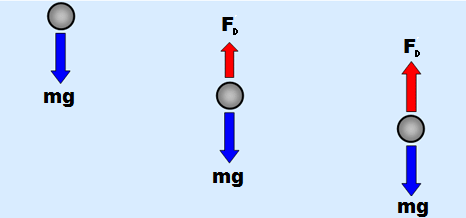
Snow Drop.

«***Snow drop***" er et spill som simulerer et fall av en pakke fra stor høyde med luftmotstand. I fallet kan en utløse en fallskjerm for å sørge for at pakken ikke blir ødelagt når den treffer bakken.



# Fysikken.

Fysikken som ligger til grunnlag i spillet er en differensiallikning for fall i en dimensjon med luftmotstand. I fall i en dimensjon vil det være 2 krefter som påvirker pakken: gravitasjon, og luftmotstand. Gravitasjonen vil være kraften som dytter pakken i negativ retning (nedover), mens luftmotstanden er kraften som dytter pakken i positiv retning (oppover). Luftmotstanden er en kraft som varierer med farten til objektet den påvirker.



# Variabler og Input.

1. Startfarten til pakken i Y-retning, gitt i .
2. Startposisjonen til pakken, gitt i  ***ovover bakken***.
3. Tettheten til mediumet objektet reiser gjennom, gitt i .
4. Flaten til objektet, gitt i .
5. Luftmotstands koeffisient, gitt som et tall .

# Formler og implementasjon.

Pakken kommer til å falle fra startposisjon , og startfart hver gang spillet kjøres. Siden fallet bare skjer i en dimensjon blir likningen for fallet, , hvor er luftmotstandskraften og dens variabler er .

* er tettheten til luften.
* er luftmotstands koeffisient.
* er flaten til objektet.
* er farten til objektet.
* er kraften til gravitasjonen.
* er akselerasjonen til objektet.

Videre kan vi løse likningen for , akselerasjonen for å finne differensiallikningen vi er ute etter.

Som gir differensiallikningen:

Siden denne oppgaven krever at jeg skal løse likningen med Runga Kutta fjerde ordens metode må likningen skrives om til to første ordens differensiallikninger.

Det gir differensiallikningene:

Disse likningene kan så plugges inn i Runga Kutta fjerde ordens metode, for å få tak i og .

Generell beskrivelse av Runga Kutta fjerde ordens metode:

Implementasjon av Runga Kutta fjerde ordens metode i kode:

private void RungaKutta4(float t\_n, float y\_n, float x\_n, float h) {

float k0 = h \*function1(t\_n, y\_n, x\_n);

float l0 = h \*function2(t\_n, y\_n, x\_n);

float k1 = h \*function1(t\_n + (h / 2), y\_n + (k0 / 2), x\_n + (l0 / 2));

float l1 = h \*function2(t\_n + (h / 2), y\_n + (k0 / 2), x\_n + (l0 / 2));

float k2 = h \*function1(t\_n + (h / 2), y\_n + (k1 / 2), x\_n + (l1 / 2));

float l2 = h \*function2(t\_n + (h/2), y\_n + (k1/2), x\_n + (l1/2));

float k3 = h \*function1(t\_n + h, y\_n + k2 , x\_n + l2);

float l3 = h \*function2(t\_n + h, y\_n + k2, x\_n + l2);

this.t\_n = t\_n + h;

this.y\_n = y\_n + ((c) \* ((k0 + (2 \* k1) + (2 \* k2) + k3)));

this.x\_n = x\_n + ((c) \* ((l0 + (2 \* l1) + (2 \* l2) + l3)));

}

private float function1(float t, float y, float x)

{

return (((0.5f \* p \* d \* a \* y \* y)/m) - g);

}

private float function2(float t, float y, float x)

{

return (y);

}