Oblig 3 – INF1100

Oppgaven er å fullføre fem funksjoner i et C program slik at programmet kan tegne, fargelegge, flytte, skalere og beregne «bounding box» til en gitt trekant definert i tre punkter.

Motivasjonen er at vi endelig kan se grafisk hva koden vi skriver gjør, istedenfor å bare jobbe med tekst.

Målet med oppgaven er å tegne en tekanne som er satt sammen av 1092 trekanter.

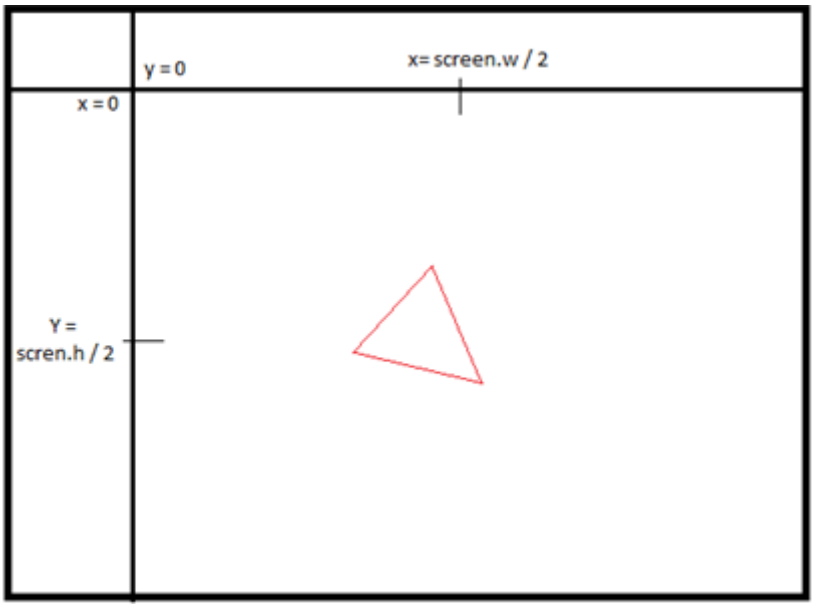
Teknisk Bakgrunn

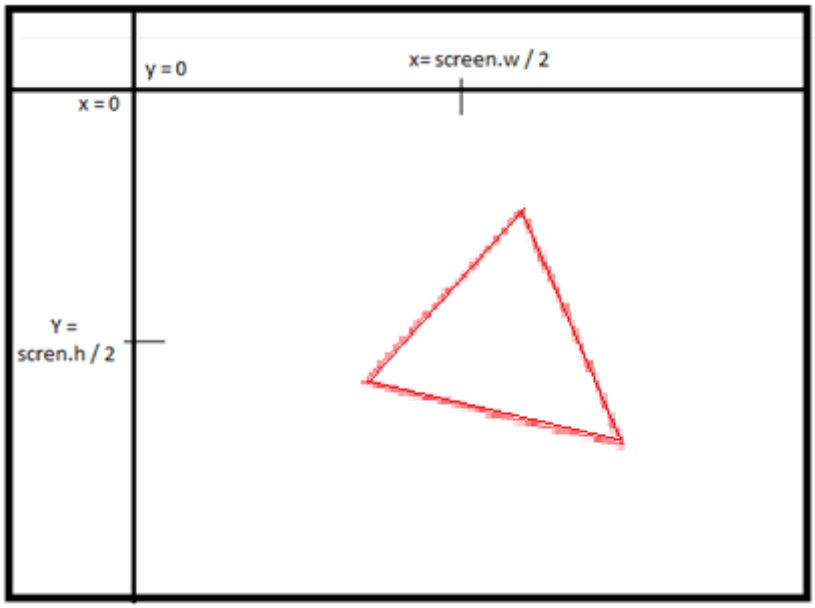
Nye kunnskaper som er nødvendig for denne oppgaven er

- Arrays, [];
  - En array er en variabeltype som består av en liste med variabler i rekkefølge.
  - anArray[1,2,3,4] => anArray[2] = 3
- Pointers, & * ->
  - Istedenfor å peke til variabelen peker man til minneadressen variabelen ligger på.
  - *triangle.scale er det samme som triangle->scale
- Structs, triangle_t eksempel {a1,a2,};
  - Et struct i C er en egen variabeltype som du definerer selv. Den kan funke som en blueprint eller en template ettersom du bestemmer hvilke verdier og variabler den skal ha.
  - Om du lager et struct eks: bil_ford minbil = {år, farge,};
    kan du lage mange variabler av typen bil_ford på samme måte som du lager nye variabler av typen int: int i = 3; bil_ford k = {år=2002, farge=rød};

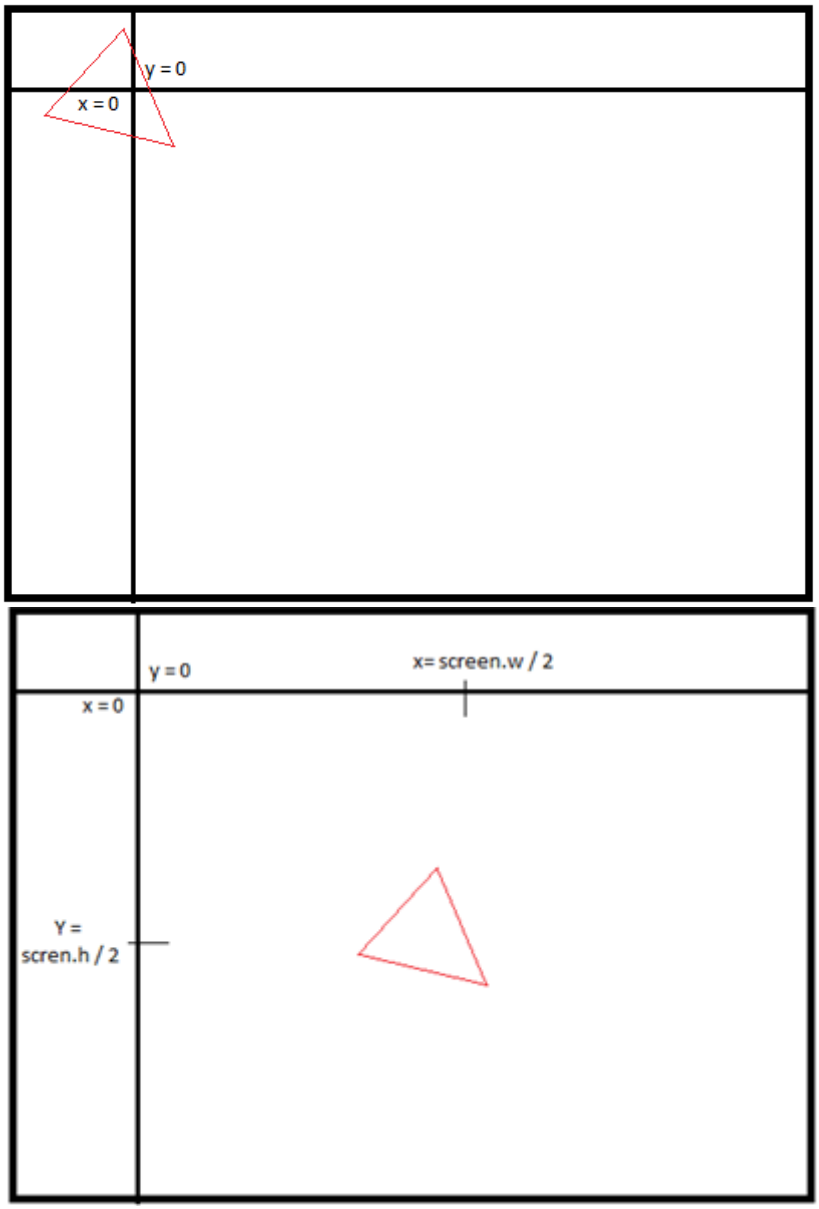
Design og implementering

ScaleTriangle er ganske enkel. Alle koordinatene til trekanten beholder forholdstallene seg imellom dersom du ganger de med samme konstant. Derfor, når jeg ganger alle koordinatene med ‘scale’, blir trekanten ‘scale’ ganger større. Bildene under illustrerer hvordan ScaleTriangle fungerer.





TranslateTriangle er også nokså enkel. For å tegne en trekant midt på skjermen er det bare å plusse midten av skjermen sin x og y verdi på trekantens x og y verdier. Gitt at trekanten er definert rundt origo. I min kode er det to løsninger på denne funksjonen. En som bruker trekantens tx og ty for å flytte den, og en løsning som setter tekannen midt på skjermen. Bildene under illustrer grovt hvordan TranslateTriangle fungerer.



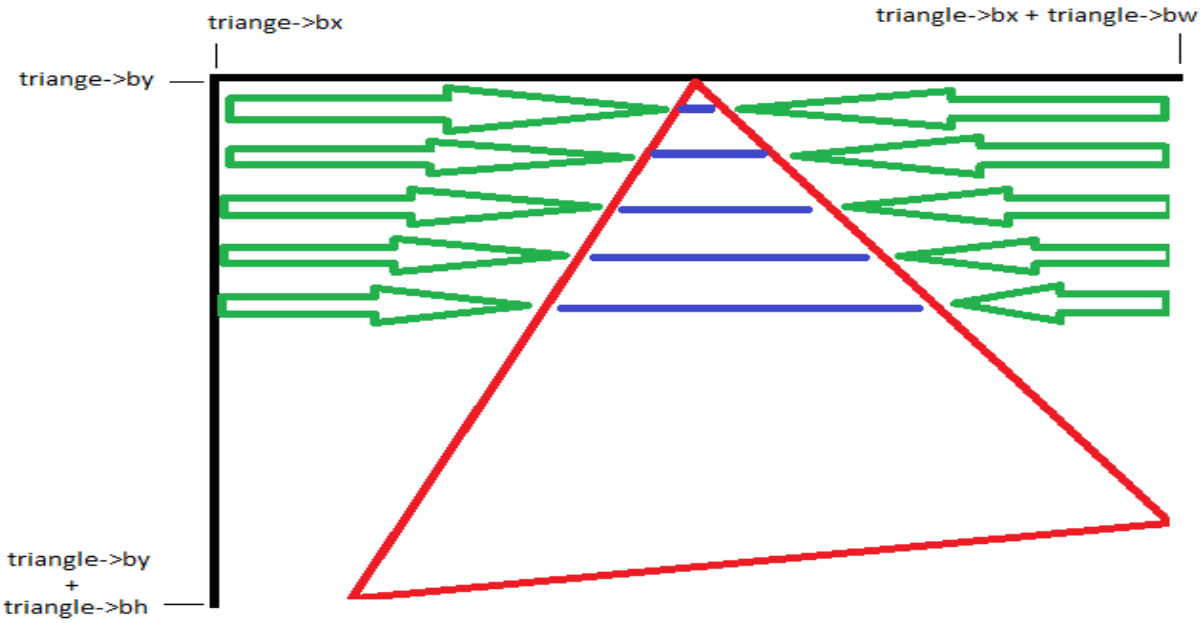
CalculateTriangleBoundingBox er nok den enkleste. Jeg sjekker hver av x og y verdiene for å se hvilke verdier som er minst. Her er det lurt å bruke if og else slik at programmet ikke kjører unødvendig kode. Det øverste vesntre hjørnet til bounding boxen, i tillegg til høyden og bredden, lagrer jeg i variabler slik at jeg kan bruke de senere.

FillTriangle er den mest utfordrende funksjonen å skrive. Den første fremgangsmåten gikk ut på å sjekke alle x verdier fra venstre mot høyre til jeg finner en side av trekanten, for så å fargelegge hver pixel til den møter andre siden av trekanten. Dette gikk ikke ettersom øverste punkt i trekanten bare består av én pixel og derfor fikk jeg en infinite loop.

Den framgangsmåten som virket var å starte øverst i bounding boxen og sjekke x verdier fra venstre mot høyre til jeg finner venstresiden, etterpå sjekket jeg x verdier fra høyre mot venstre for å finne

høyresiden. Deretter brukte jeg DrawLine mellom punktene for å fargelegge trekanten på den y verdien. En loop kjører denne koden for hver y verdi og så lenge y verdien er mindre eller lik høyden til bounding boxen.

Hvordan FillTriangle og CalculateTriangleBoundingBox fungerer er grovt illustrert med bildet nedenfor.



Konklusjon

Det største problemet jeg hadde var infiniteloopen som jeg fikk med den første metoden for å fargelegge trekanten. Løsningen var å bruke en mer generell metode for å finne høyre og venstresiden av trekanten. Jeg føler at koden som regner ut bounding box burde vært kortere men har for øyeblikket ikke sett hvordan den skulle vært kortere.