UiT
NORGES
ARKTISKE
UNIVERSITET

INF-1100 Oblig. 3

Fysikk Bachelor
21. Oktober 2016
Martin Soria Røvang

1. Introduksjon

I denne oppgaven skulle det skisseres opp trekanter til en tekanne og så fylle de med farger.

Her lærer man å bruke andre koder og programmer.

Når alt er gjort riktig så skal jeg få en slik figur.

Figur 1.

1.1. Krav

Her kreves det å kunne bruke makefiles og flere headerfiles.
Makefiles er en fil du kan kalle på som har en rekke operasjoner i seg, som for eksempel å kompilere alle filene som trengs å kompilere. Dette er veldig hjelpsomt når man har et større prosjekt og vil spare tid.

2. Teknisk bakgrunn

Her brukes SDL (Simple Direct media Layer) dette er et program som lar meg få grafikk ut av C koden.

Strukturer som er et sett med variabler som tilhører et navn, Typedef (Typedefinitions) som forenkler en funksjon ved å definere funksjonsuttrykket med noe du ønsker. Pekere som viser til en adresse for å lagre information utenfor funksjoner.

Bruk av array som er en 'hylle' som gjør det enklere å håndtere mye informasjon, her var hele teapot_data en stor array.

3. Oppbygging av programmet

For å klare oppgaven måtte jeg endre på disse funksjonene

- ScaleTriangle
- TranslateTriangle
- CalculateTriangleBoundingBox
- FillTriangle
- DrawTriangle

Måtte også legge til en loop i main for å skrive ut alle koordinatene til trekantene fra teapot_data arrayen og dermed DrawTriangle for å lage strek mellom punktene.

ScaleTriangle er en funksjon som skal forandre størrelsen på tekannen ved å legge inn en faktor på formen 1.0, her skal det kun endres på skjermkoordinatene ikke selve koordinatene til structen til hver av trekantene.

TranslateTriangle er en funksjon som skal bevege alle trekantene til den posisjonen du vil ha trekantene som her vil være i midten av pc'ens koordinatsystem.

CalculateTriangleBoundingBox er en funksjon som skal kalkulere firkanter rundt trekantene som gjør det enkelt å kunne «skanne trekanter» for å for eksempel fylle de med farger som jeg vil gjøre i denne oppgaven.

FillTriangle er en funksjon som skal fylle trekantene med deres respektive farge.

DrawTriangle er en funksjon som tegner streker mellom 3 punkter som skal bli en trekant

Dette er de funksjonene som ble brukt.

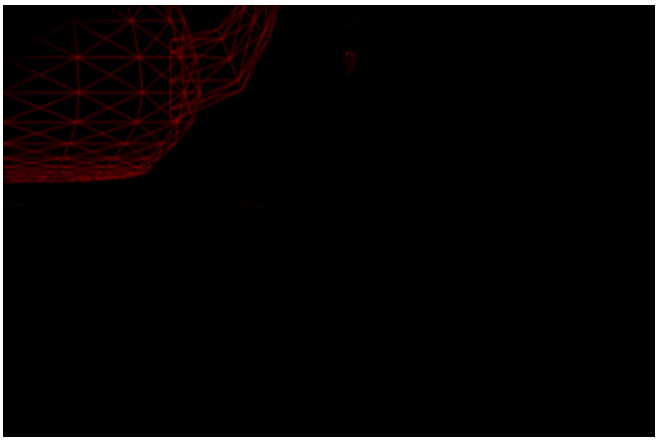
4. Diskusjon

Skrev først inn DrawLine for å få tegnet frem eksempel trekantene som kan forklares slik.

```
DrawLine(screen, triangle->sx1, triangle->sy1, triangle->sx2, triangle->sy2, TRIANGLE_PENCOLOR);
DrawLine(screen, triangle->sx2, triangle->sy2, triangle->sx3, triangle->sy3, TRIANGLE_PENCOLOR);
DrawLine(screen, triangle->sx1, triangle->sy1, triangle->sx3, triangle->sy3, TRIANGLE_PENCOLOR);
```

Her vil DrawLine finne alle x og y koordinatene for alle 3 punktene for hver trekant som ligger i main for så å tegne en strek mellom dem med DrawTriangle funksjonen. Derfor er det viktig å få inn alle trekantene fra teapot_data.h inn i main. Dette gjøres ved å legge inn en loop som skriver ut hele teapot_data arrayen med informasjonen til alle trekantene til tekannemodellen.

Hvis eksempeltrekantene er slettet så vil man få noe som dette.

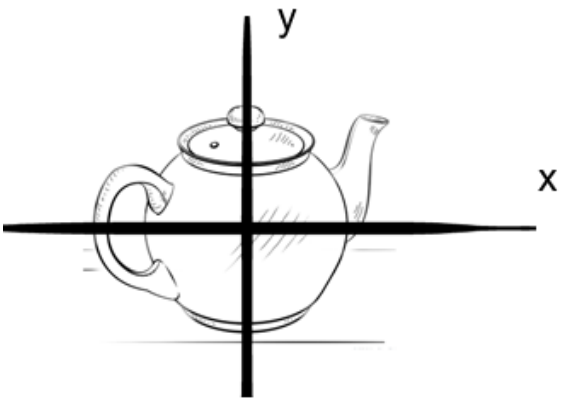


Figur 2.

Grunnen til dette er fordi man ikke har gjort TranslateTriangle.

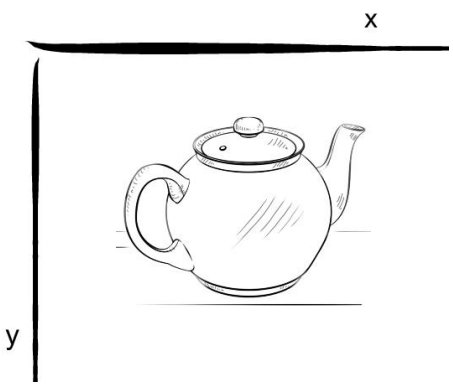
Koordinatene til trekantene fra teapot_data er fra kartesisk koordinatsystem og er dermed ikke kompatibel med pcen koordinatsystem.

I figur 3 ser du hvordan tekannen blir plassert



Figur 3

Her må man da få flyttet alt ned til pc koordinatsystemet som ser ut som i figur 4.



Figur 4.

Dermed må man endre TranslateTriangle slik.

Oppløsningen som blir satt kan du se i main, der står det at SetVideomode skal bli satt til (1024,768) pixler så kan først sette Triangle->tx til x som er 1024 og Triangle->ty til y som er 768, men siden vinduet kun blir åpnet i et halvt vindu så må du dele disse på 2 for halvparten ikke skal bli plassert utafor skjermen

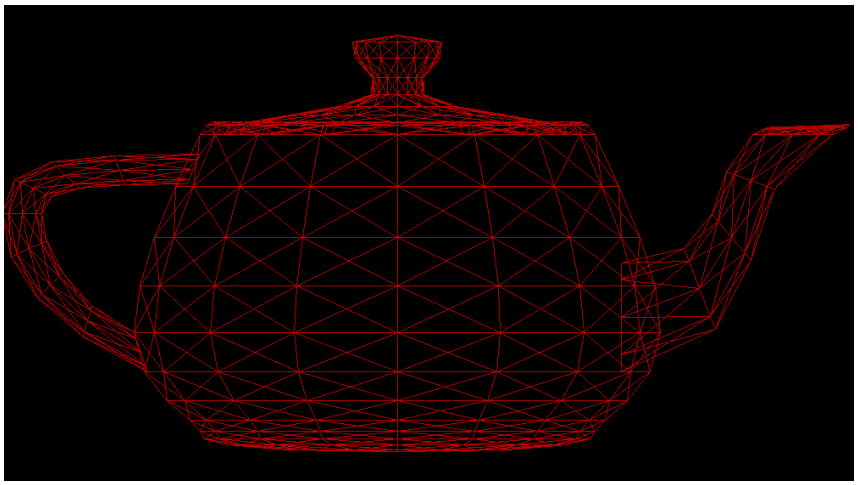
```
//Create a 1024x768x32 window
screen = SDL_SetVideoMode(1024, 768, 32, 0);
```

Figur 5.

Etter dette må de originale koordinatene endres slik at de kommer inn i pcens koordinatsystem disse heter 'sx' og 'sy' osv. altså screencoordinates.

Dette gjøres ved å plusse opp skjermkoordinatene med screenkoordinatene.

Når dette gjøres riktig vil man få resultatet som i figur 6.

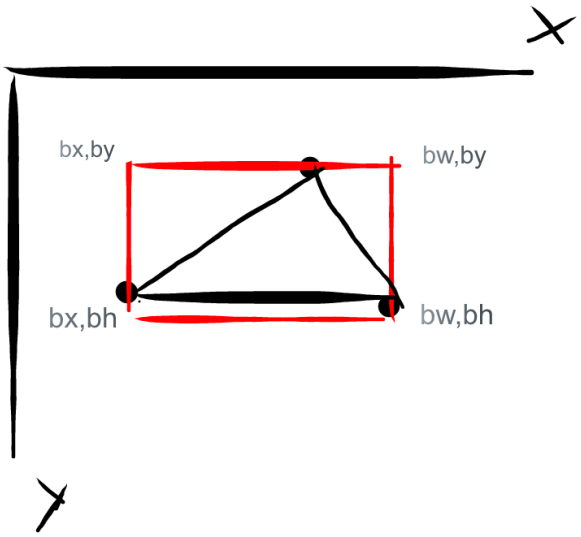


Figur 6.

Etter dette må man få laget bokser som skal gå rundt hver trekant.

Dette gjøres ved å finne den minste og største i x og y retning i hver trekant fordi da kan CalculateTriangleBoundingBox lage en firkant rundt trekanten for å fylle trekanten med farger senere.

Sånn at det blir slik.

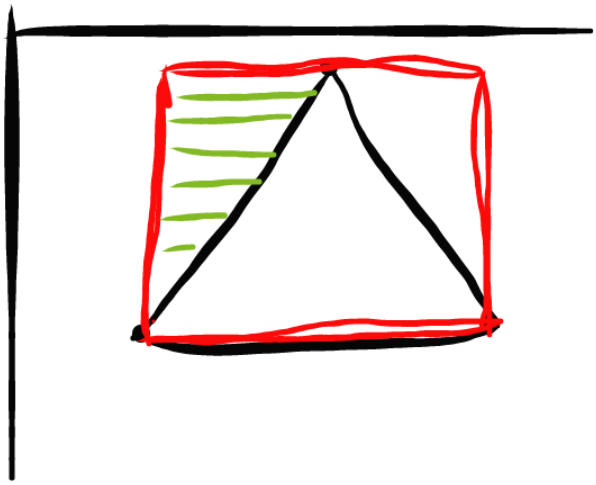


Figur 7.

Her må da bx og by være de minste koordinatene og bw(width), bh(height) de største.

Dette danner da boksen.

FillTriangle er funksjonen som skal fylle trekantene med farge og dette kan gjøres ved å lage en kode som «skanner» etter trekanten ved å så fylle med farge slik som figur 8.



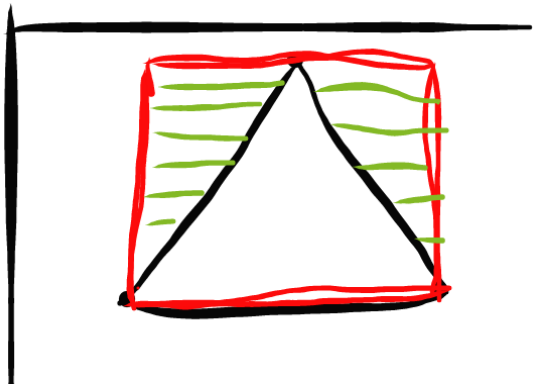
Figur 8.

Her er grønnfargen pixelene som har blitt «skannet» ikke at den fargelegger grønt.

Her «skanner» jeg fra venstre til høyre altså funksjonen leser av hver piksel bortover x aksen fra en bestem y akse for å se om pikselen er TRIANGLE_PENCOLOR som er fargen som ble brukt for å tegne trekantene. Dette gjøres selvfølgelig i en loop fordi det er svært mange piksler som må leses.

Her kommer Getpixel funksjonen inn. Med denne så kan man lese av og se om pikselen er en TRIANGLE_PENCOLOR. Ved å ha break; i loopen hvis dette er sant så kan man lese av fra neste side.

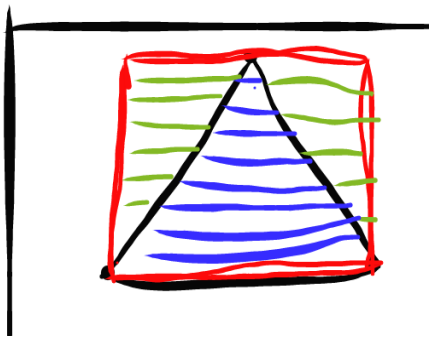
Så må en ny loop ta over og lese fra venstre for å finne den andre siden av trekanten.



Figur 9.

Den første loopen skal dermed plusse minste koordinatene opp helt til den treffer kanten av trekanten. Og så skal den andre trekke fra de største koordinatene til den treffer kanten.

Da vet man hvor trekanten ligger! Hvis man da bruker Drawline til å tegne farge fra og med pixlene som ble oppdaget av Getpixel så vil man fylle trekanten med farge. Her brukes også Fillcolor i stedet for TRIANGLE_PENCOLOR fordi alle structene til hver trekant har sin respektive farge.



Figur 10.

Dette er da selvfølgelig ikke i skala, fordi det vil være 1092 slike trekanter.

Da skal resultatet bli slik;

Man skal kommentere vekk screen update som tillater koordinatene til width og height og implementere boundingbox update som er kommentert ut.

Screenupdate med koordinater krever mye mer av pcen så ved å kun oppdatere bounding boksene så vil programmet kjøre raskere.

Da må det legges inn differanse i boundingbox funksjonen i stedet for koordinater, dette gjøres ved å trekke fra minstekoordinatet fra det største i bredden (bw) og høyden (bh).

Dette må dermed kompenseres i fillcolor funksjonen og dermed plusse opp bredden og høyden i fillcolor løkka.

5. Konklusjon

I denne oppgaven måtte jeg være kreativ. Først måtte jeg studere all koden for å få en forståelse av hvordan koden var bygget og hva jeg kunne endre på for å gjøre oppgaven. Det var litt «kaotisk» i starten fordi det var mye informasjon og fordøye, men gikk greit etter hvert som jeg skjønte oppbygginga av koden og hvordan funksjonene fungerte.