Assignment 3: Projektioner

Marie Elkjær Rødsgaard - dck495
Department of Computer Science

November 29, 2022

Contents

1 Introduktion

Dette er den tredje opgave i Grafik kurset og omhandler perspektiviske projektioner i et vertex program. I denne opgave skal man forstå hvordan man transformerer et vilkårligt perspektivisk view volumen til det kanoniske perspektiviske view volumen, og derfra videre til det kanoniske parallelle view volumen. Vi starter med at kigge på hvad en projektion er og hvilke typer der findes.

2 Teori

Projektioner er at transformere punkter i et koordinat system af n dimensioner til punkter i et koordinat system af mindre end n dimensioner. De projektioner vi vil kigge på her kalder vi for *planer geometric projections* da vi kigger på projektioner i et plan og ikke en buet overflade. *Planer geometric projections* eller bare projektioner har to forskellige typer som kan laves: perspektiver og parallelle. Forskellen på de to typer er relationen til *center of projection*. Hvis afstanden til *center of projection* er endelig er det en perspektivisk projektion, hvis afstanden derimod er uendelig er det en parralle projektion.

Parralle projektioner er kategoriseret i to typer som afhænger af relationen til den retning projektionen er og normalen til projektionens plan. I den første som er orthographic paralle projektioner er de retninger som nævnt før enten det samme eller eller omvendt af hinanden. For den anden paralle projektion oblique er dette ikke tilfældet. Vi kigger på de mest almindelig paralle projektioner som er orthographic projektioner. [Foley, 1990]

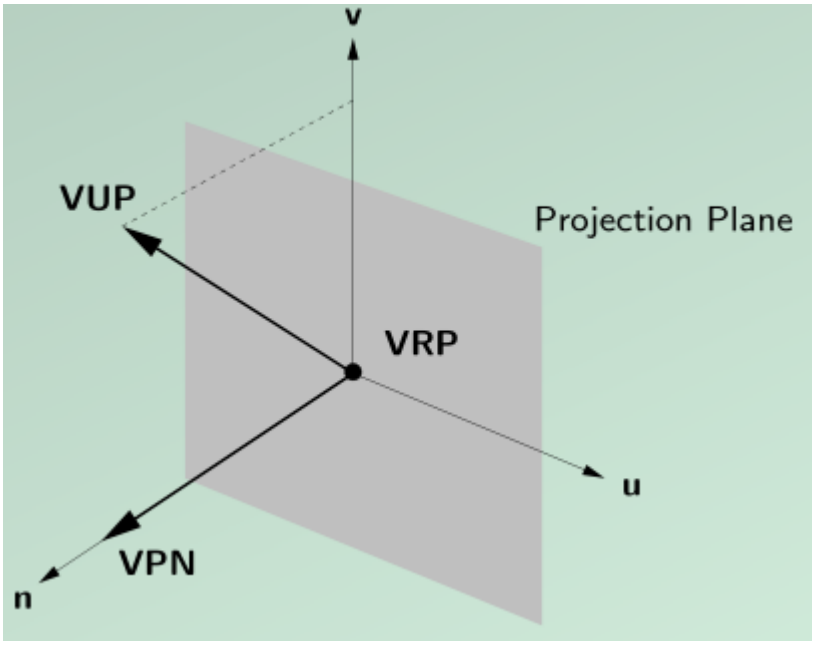

Figure 1: Øje-koordinatsystemet

Når man skal transformere projektioner gøres det i et projektions plan eller et øje-koordinat system. For at konstruere øje-koordinatsystemet skal man bruge to vektorer VPN *View Plane Normal* og VUP *View Up Vector*. Figuren ovenover viser hvordan koordinatsystemet ser ud. Punktet VRP *view reference point* er punktet vores system tager udgangspunkt i. Koordinatsystemet $(u, v, n)^T$ er definere af de tre varibler (u,v,n). n er en unit vektor i samme retning som VPN. Den kan skrives op som $n = \frac{VPN}{||VPN||_2}$. u er også en unit vektor som er vinkelret til VUP og VPN, den kan skrives op som $u = \frac{VUP \times VPN}{||VUP \times VPN||_2}$. v er en unit vektor i samme retning som VUP's projektion langs VPN på projektions planen. v findes så ved $v = \frac{VPN \times (VUP \times VPN)}{||VPN \times (VUP \times VPN)||_2} = n \times u$. Dette er sådan at VPN og VUP giver os vores øje-koordinatsystem.

For at transformere fra perspektivisk view volumen til det kanoniske perspektiviske view volumen skal man følgene

1. Translate VRP til origo
2. Rotate øje-koordinatsystemet så det falder sammen med verdens koordinat systemet. Det vil sige vore u-akse $\rightarrow$ X -aksen, v-aksen $\rightarrow$ Y-aksen og n-aksen $\rightarrow$ Z-aksen.
3. Translate PRP til origo
4. Shear så centerlinjen af vores view volume bliver til z aksen.
5. Scale til det kanoniske perspektive view volume.

Så vi starter med at translatere vores VRP til origo. Det gør vi ved at bruge translations matrixen

$$T(-VRP) = \begin{pmatrix} 1 & 0 & 0 & -VRP_x \\ 0 & 1 & 0 & -VRP_y \\ 0 & 0 & 1 & -VRP_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Det næste vi gør er nu at rotere så vores (u,v,n) falder sammen med verdenskoordinatsystemet(x,y,z). For at gøre det skal vi bruge vores rotations matrice

$$R = \begin{pmatrix} r_{1x} & r_{2x} & r_{3x} & 0 \\ r_{1y} & r_{2y} & r_{3y} & 0 \\ r_{1z} & r_{2z} & r_{3z} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

vores rækker svare så til
$R_z^T = (r_{1z}, r_{2z}, r_{3z}) = VPN/VPN_2$
$R_x^T = (r_{1x}, r_{2x}, r_{3x}) = VUP \times Rz/VUPRz_2$
$R_y^T = (r_{1y}, r_{2y}, r_{3y}) = Rz \times Rx/Rz \times Rx_2$
Nu stemmer vores koordinatsystemer overens.

Nu skal vi translatere igen så vores *projection reference point* PRP som er toppen af pyraminden kommer til origo.

$$T(-PRP) = \begin{pmatrix} 1 & 0 & 0 & -PRP_u \\ 0 & 1 & 0 & -PRP_n \\ 0 & 0 & 1 & -PRP_v \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

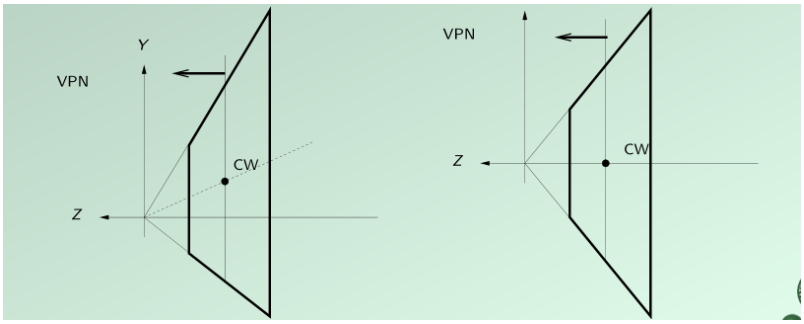

Figure 2: Figure før og efter Shear

Nu vil vores pyramide være lidt skæv som ses på figure 2 over. For at rette op på det bruger vi vores *center of window* CW. Så vil vi lave en Shear så vores CW kommer ned og ligge på vores z akse. Vi har at vores *direction of projection* $DOP = PRP - CW = (dop_u, dop_v, dop_n, 0)^T$. Shear matricen er

$$Sh_{per} = \begin{pmatrix} 1 & 0 & sh_x & 0 \\ 0 & 1 & sh_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Vi har her vores sh_x og sh_y som vi skal finde. Vi har at $DOP = Dh_{per}DOP = (0, 0, dop_n, 0)^T$ Den udregning gør vi på følgene måde.

$$\begin{pmatrix} 1 & 0 & sh_x & 0 \\ 0 & 1 & sh_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} dop_u \\ dop_v \\ dop_n \\ 0 \end{pmatrix} = \begin{pmatrix} dop_u + sh_x \\ dop_v + sh_y \\ dop_n \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ dop_n \\ 0 \end{pmatrix}$$

Nu har vi en ligning så vi kan finde sh_x og sh_y

$sh_x = -\frac{dop_u}{dop_n}$

$sh_y = -\frac{dop_v}{dop_n}$

Nu ser vores Shear matrix ud på følgene måde

$$Sh_{per} = \begin{pmatrix} 1 & 0 & -\frac{dop_u}{dop_n} & 0 \\ 0 & 1 & -\frac{dop_v}{dop_n} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

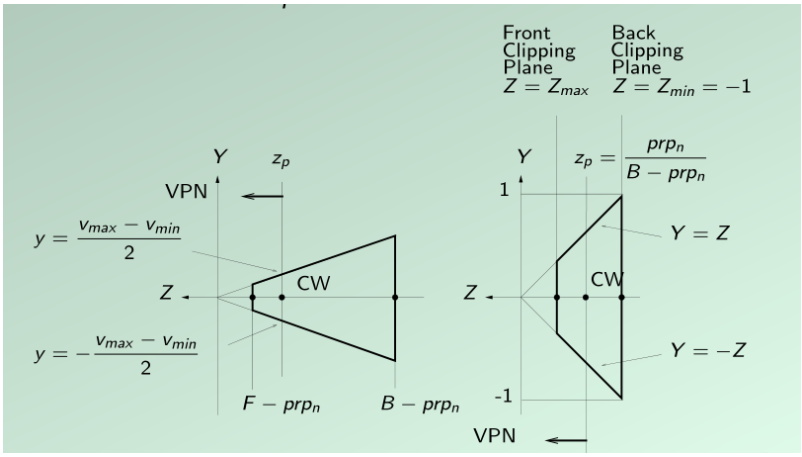

Figure 3: Figur før og efter skalering

Det sidste der skal gøres nu er at skalere til den kanoniske projektion så vi får den rigtige størrelse. Figur 3 ovenover viser den størrelse vi har nu og hvad vi ønsker. For at få den ønskede figur skal vi have sat vores front clipping plane og vores back clipping plane de rigtige steder hen, så pyramiden ender med en højde på 1 vi vil også have en vinkel på 45°. Det vi så skal gøre er at skalere længden ud til vores CW som er $-PRP_n$ og så har vi den halve længde af CW, de to længder skal være ens. Vi har nu vores nye VRP' efter VRP er blevet tranlateret og sheared $VRP' = Sh_{per}T(-PRP) =$

$$\begin{pmatrix} -PRP_u + \frac{dop_u}{dop_n} PRP_n \\ -PRP_v + \frac{dop_v}{dop_n} PRP_n \\ -PRP_n \\ 1 \end{pmatrix}$$

Først skal vi så skalere i x og y retningerne så den halve CW bliver lig med $VRP'_Z = -PRP_n$ og så får vi en 45° linje. Det gør vi således

$$s_x = \frac{-2VRP'_Z}{u_{max} - u_{min}} = \frac{2PRP_n}{u_{max} - u_{min}}$$

$$s_y = \frac{-2VRP'_Z}{v_{max} - v_{min}} = \frac{2PRP_n}{v_{max} - v_{min}}$$

Nu skal vi så skalere vores $VRP'_Z + B = B - PRP_n \to -1$ hvor B er back-clipping plane og den størrelse går over i -1. Så dividere vi med -1 så får vi $s = \frac{-1}{VRP'_Z+B} = \frac{-1}{B-PRP_n}$. Dette sørger for at vi får den rigtige højde på vores pyramide.
Nu har vi alle vores skaleringsfaktore til vores S_{per}.

$$s_x = \frac{-2PRP_n}{(u_{max} - u_{min}(B - PRP_n)}$$

$$s_y = \frac{-2PRP_n}{(v_{max} - v_{min}(B - PRP_n)}$$

$$s_z = \frac{-1}{B - PRP_n}$$

Det kan vi indsætte i vores endelige scale matrix

$$S_{per} = \begin{pmatrix} \frac{-2PRP_n}{(u_{max}-u_{min}(B-PRP_n)} & 0 & 0 & 0 \\ 0 & \frac{-2PRP_n}{(v_{max}-v_{min}(B-PRP_n)} & 0 & 0 \\ 0 & 0 & \frac{-1}{B-PRP_n} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Nu har vi så gjordt alle de trin der skal fortages for at komme frem til den kanoniske perspektiviske view volume ud fra et vilkårligt perspektivisktisk view volume.

Alle trinene kan nu skrives op for den tranformation der skal til.
$N_{per} = S_{per} \cdot Sh_{per} \cdot T(-PRP) \cdot R \cdot T(-VRP)$

Efter vores skalering har værdierne Z_{min}, Z_{max} og z_p ændret sig. De er nu
$Z_{min} = -1$
$Z_{max} = \frac{F-PRP_n}{B-PRP_n}$
$z_p = \frac{PRP_n}{B-PRP_n}$
De skal bruges når vi skal transformere til den kanoniske parralle view volume.

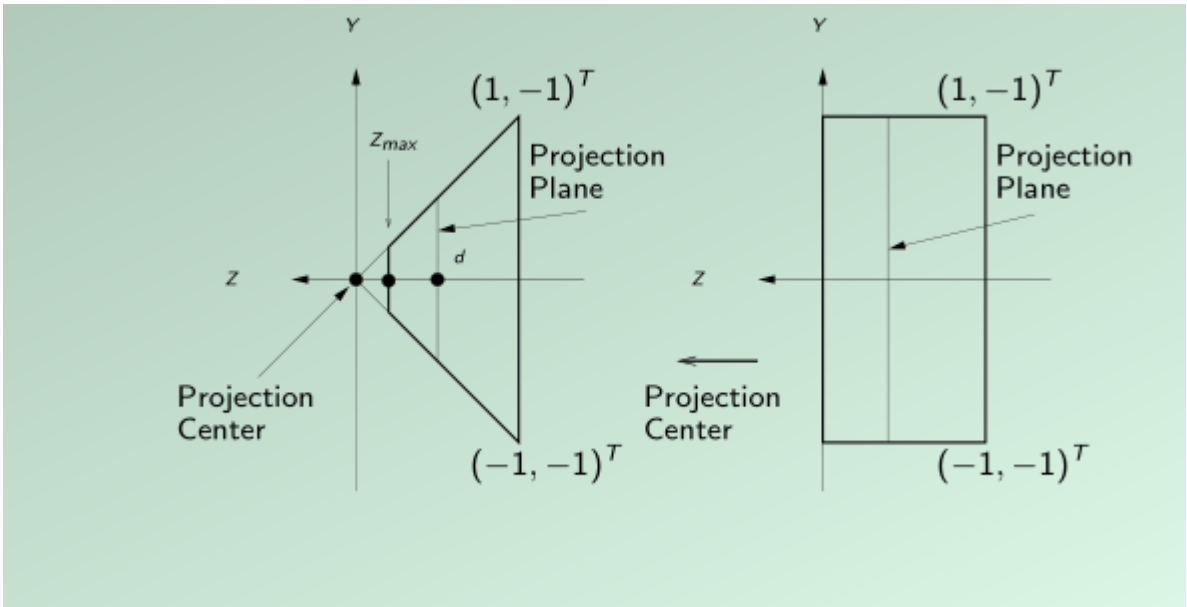

Figure 4: kanonisk perspektivistisk view volumen om til kanonisk parralle view volumen

På figure 4 ses de to view volumer, den kanoniske perspektiviske view volume vi fandt før og den kanoniske parralle view volume. For at transformere til den kanoniske parralle view volume tager vi fat i de to ydre punkter ved z_{max} og hiver dem op til kanterne så det ligner den parralle.

For at gøre dette bruger vi matricen

$$M_{perpar} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{1+Z_{max}} & \frac{-Z_{max}}{1+Z_{max}} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

Denne matrix kan tranformere det kanoniske perspektiviske view volume over til den kanoniske orthographic view volume.
Matrixen M_{perpar} kan vi vi få ved følgene trin. Hvis vi starter med at kigge på de to ligninger for perspektivisk projektion for $[x, y]^T$ som er lig

$$x = d\frac{X}{Z}$$

$$y = d\frac{Y}{Z}$$

De to ligninger vil vi have til at ligge i intervalet $[x,y]^T \in [-|d|,|d|]\times[-|d|,|d|]$ Vi vil så transformere dette så de to intervaller ligger i $[-1,1]\times[-1,1]$ ved at skalere med faktoren $\frac{1}{d}$ Så får vi

$$x = -(\frac{1}{d})d\frac{X}{Z} = -\frac{X}{Z}$$

$$y = -(\frac{1}{d})d\frac{Y}{Z} = -\frac{Y}{Z}$$

Nu er vores ligninger indenfor det ønskede interval, det svare også til vores back clipping plane fra tidligere.

Vi ønsker nu at få vores z værdi. Vi starter med at lave en funktion f som kan transformere z koordinatet. $f(Z) = aZ + b$. Nu kan vi sætte nogle restriktioner på vores funktion.
$f(Z_{max}) = aZ_{max} + b = 0$
$f(-1) = a + b = -1$
Dette giver os ligningssystemet
$\begin{pmatrix} Z_{max} & 1 \\ -1 & 1 \end{pmatrix}\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} 0 \\ -1 \end{pmatrix}$
Når vi så løser ligningssystemet får vi a og b til følgene

$$a = \frac{\begin{vmatrix} 0 & 1 \\ -1 & 1 \end{vmatrix}}{\begin{vmatrix} Z_{max} & 1 \\ -1 & 1 \end{vmatrix}} = \frac{1}{Z_{max}+1}$$

$$b = \frac{\begin{vmatrix} Z_{max} & 0 \\ -1 & -1 \end{vmatrix}}{\begin{vmatrix} Z_{max} & 1 \\ -1 & 1 \end{vmatrix}} = \frac{-Z_{max}}{Z_{max}+1}$$

Det giver os så funktionen
$z = f(Z) = \frac{Z}{Z_{max}+1} - \frac{-Z_{max}}{Z_{max}+1}$
Vi kan nu bruge de ligninger vi har til at lave matricen M_{perpar}

$$x = -\frac{X}{Z}$$

$$y = -\frac{Y}{Z}$$

$$z = \frac{Z - Z_{max}}{1 + Z_{max}}$$

$$M_{perpar} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{1+Z_{max}} & \frac{-Z_{max}}{1+Z_{max}} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

3 Implementation

For at implementere transformationerne til de forskellige view volume fokusere vi på at finde ViewProjection og ViewOrientation. Alle trinene vi har kigget på i teorien herover kan vi sætte ind i de to formler.
Parralle projektioner

$$M_{mPar-total} = M_{wv} \cdot S_{par} \cdot T_{par} \cdot Sh_{par} \cdot R \cdot T(-VRP)$$

Hvor $S_{par} \cdot T_{par} \cdot Sh_{par}$ er vores ViewProjection og $R \cdot T(-VRP)$ er vores ViewOrientation. M_{wv} er en matrix som OpenGL gør automatisk så den kigger vi ikke på.

Perspektivistiske projektioner

$$M_{per-total} = M_{wv} \cdot M_{perpar} \cdot S_{par} \cdot Sh_{par} \cdot T_{par}(-PRP) \cdot R \cdot T(-VRP)$$

Hvor $M_{perpar} \cdot S_{par} \cdot Sh_{par} \cdot T_{par}(-PRP)$ er vores ViewProjection og $R \cdot T(-VRP)$ igen er vores ViewOrientation.

Vi Kigger nu på de to funktioner `Camera::ComputeViewOrientation` og `Camera::ComputeViewProjection` Som er dem vi bruger til at få vores `CurrentTransformationMatrix` eller dens inverse.
I `Camera::ComputeViewOrientation` starter vi med at lave en 4x4 matix $T(-VRP)$. Derefter konstruerer vi vores øje-koordinatsystem som bliver forklaret i teori afsnittet. Efter det skal vi så bruge rotation på (n,u,v), dette foregår også som nævnt i teorien. For så at få vores ViewOrientation matrix ganger vi bare vores Rotation matrix med vores translations matrix, vi finder også den inverse efter.

I `Camera::ComputeViewProjection` funktionen er der flere ting vi skal implementere. Vi starter med at igen at lave en 4x4 matrix, denne gang over den translaterede (-prp). Derefter definere vi vores to vektorer `CW` og `DOP`.

Så regner vi vores shx og shy ud og derefter indsætter vi det i en shearXY matrix. Så går vi videre til at udregne vores skalerings faktorer. Når de er fundet indsættes de i en matrix `S`. Nu mangler vi at definere Z_{max} og vores matrix M_{perpar}. Vi ved hvordan de ser ud, på baggrund af teoriafsnittet. Til sidst udregner vi så den inverse skalering matrix og den inverse M_{perpar} matrix. Nu har vi alt vi skal bruge til at få vores viewprojection matrix og den inverse viewprojection matrix.

Vi bruger så de matrixer til at udregne vores `CurrentTransformationMatrix` og `InvCurrentTransformationMatrix()`.

4 Test

For at teste om implementationen virker som forventet ser vi om vi får tegnet de rigtige billeder, som også ses i opgave teksten. Vi kan se på figuren nedeunder at programmet virker som ønsket.

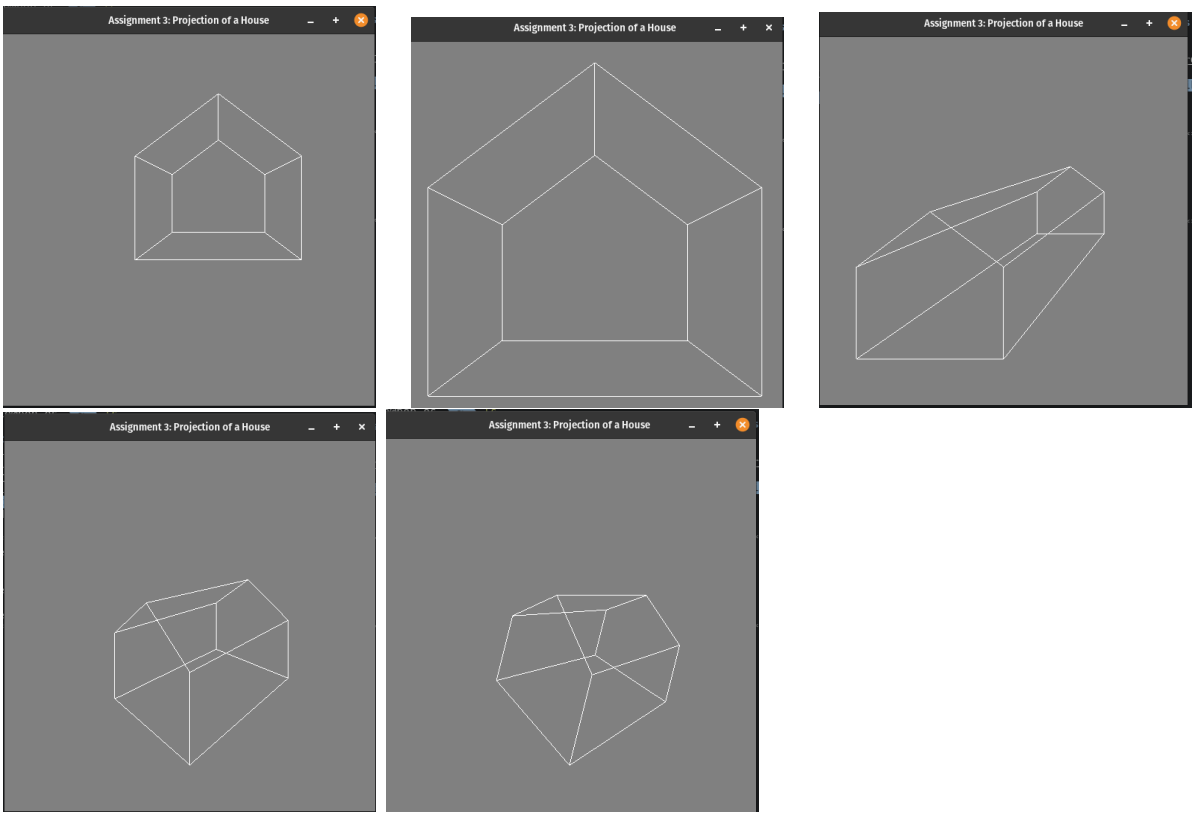

Figure 5: Ønskede eksempler af hus

5 Konklusion

Vi har nu gennemgået hvordan man teoretisk ville implementere transformation af et vilkårligt perspektivisk view volumen til det kanoniske perspektiviske view volumen, og derfra videre til det kanoniske parallelle view volumen. Vi har kigget på hvordan dette ville blive implementeret og derefter testet om programmet virker som ønsket. Vi har fået vores ønskede figur ud af programmet som giver os en basis for at programmet er blevet implementeret korrekt.

References

[Foley, 1990] Foley, J. (1990). *Computer graphics: principles and practice.* The Systems Programming Series. Addison-Wesley.