Practical information

Beskrive alle featurene som er implementer og hvordan dette er gjort.   
Maks 8 sider  
Husk kilder!  
Leveres som pdf  
Leveres på canvas sammen med alle ressurser som trengs for å bygge og kjøre koden (zip)

Model: Model to World: Hvordan og hvor boksen skal vises i verden

View: World to View: Hvordan verden skal være I forhold til kameraet

Projection: View to Projection: Hvordan vise det 2-dimenesjonale bildet som 3D

# Skybox/Cubemap

Det første jeg implementerte i mitt prosjekt var Skybox(Også kalt cubemap). Dette er en enkel måte å få et inntrykk av et landskap eller verden rundt kameraet. Skyboxen vil bestå av 6 bilder som hvert for seg representerer en side inne i en kube. Disse 6 2D-bildene vil brettes rundt kameraet og på den måten vil det oppleves som en verden man kan se seg rundt i. <https://learnopengl.com/Advanced-OpenGL/Cubemaps>

## Implementere Skyboxen

Som nevnt over består Cubmap av 6 2D-bilder, altså 6 textures. For hvert bildet må derfor dette lastes inn og spesifiseres med glTexImage2D. I TextureLoading.

For Skyboxen har jeg valgt å opprette egne shadere (skybox.vert og skybox.frag).

I vertex-shaderen:

* Med layout (location 0) hentes det data med attributten 0 og legger dette i variabelen position.
* Returnerer en vec3 til neste steg i pipeline som inneholder texture-koordinater (sendes til fragment-shaderen)
* Det er deklarert to uniform variabler som brukes til å kommunisere med shadern utenifra. (Er read-only). Vi definerer ikke model-matrise da vi ikke ønsker å påvirke posisjonen til skyboxen.
  + Uniform mat4 view: Mottar informasjon om hvordan verden skal være I forhold til kameraet
  + Uniform mat4 projection: Mottar informasjon om hvordan vise det 2-dimenesjonale bildet som 3D på skjermen

Note that the interesting part of the vertex shader is that we set the incoming position vectors as the outcoming texture coordinates for the fragment shader. The fragment shader then takes these as input to sample a samplerCube:

I fragment-shaderen:

* Mottar TexCoords fra vertex-shaderen
* Sender ut en vec4 med info om color til neste steg i pipeline
* Henter inn data fra koden med info om skybox (SamplerCube)

# INSPIRASJON:

OPPSETT: Simple lighting eksempel!

CUBEMAP, OG CUBE : <https://www.youtube.com/watch?v=_EjsL3B9CRY>

LYS: Blanding av Simple Lighting og <https://learnopengl.com/Lighting/Lighting-maps>

CAMERA: <https://www.youtube.com/watch?v=ns9eVfHCYdg>

# UTFORDRINGER UNDERVEIS:

Mye i begynnelsen før jeg forstod hva kodelinjene gjorde.

**🡪 Deklarterte buffer på nytt. Hadde satt dette globalt:**

// Oppretter Vertex-array og buffer -object for KUBE

GLuint cubeVAO, cubeVBO;

// Oppretter Vertex-array og buffer -object for SKYBOX

GLuint skyboxVAO, skyboxVBO;

Men opprettet nye når de skulle benyttes for å tegne. Brukte ikke de som var satt tidligere og var knyttet til vertecisene.

🡪 Det samme gjaldt cubeTexture

Jeg initialiserte denne på nytt!

**🡪 Hadde satt feil viewmatrise for skybox**

Hadde: glm::mat4 viewSkybox = camera.GetViewMatrix( );

Skulle være glm::mat4 viewSkybox = glm::mat4( glm::mat3( camera.GetViewMatrix( ) ) );

**🡪 Få lys sammen med texture**

**HJELP:** <https://learnopengl.com/Lighting/Lighting-maps>

Måtte gange texture med ambient, diffuse og specular

**vec3** ambient = ambientStrength \* texture( cubeTexture, cubeTextureCoordinates ).rgb;

* Feil med normalverdier

Rekke følge på vertexer feil

Hjalp å kalkulere de for hver isteden for å sette de fast.

# Bibliotek

SOIL

LYS!

<http://www.chai3d.org/download/doc/html/chapter16-lighting.html>