# Pythonsk 3D

Eksempler er i stor grad lånt fra https://github.com/vispy/vispy/blob/master/examples

#### **Motivasjon**

3D grafikk er en bra måte å vise frem data.

3D ser sinnsykt kult ut.

GPU'er kan brukes til utregning, og det er lurt å ha en forståelse av hvordan de virker.

#### **Kort om GPU**

En GPU har mange små prosessorkjerner som deler minne i flere hierarkier.

Disse prosessorene gjør gjerne utregning per kant i en overflate (vertex shader) eller per punkt i en overflate (fragment shader)

#### Kort om GPU 2

En GPU mottar store mengder data på en gang. Dette kalles en buffer.

Etter å ha sendt en buffer ønsker vi kun å oppdatere de delene som vi vet har endret seg.

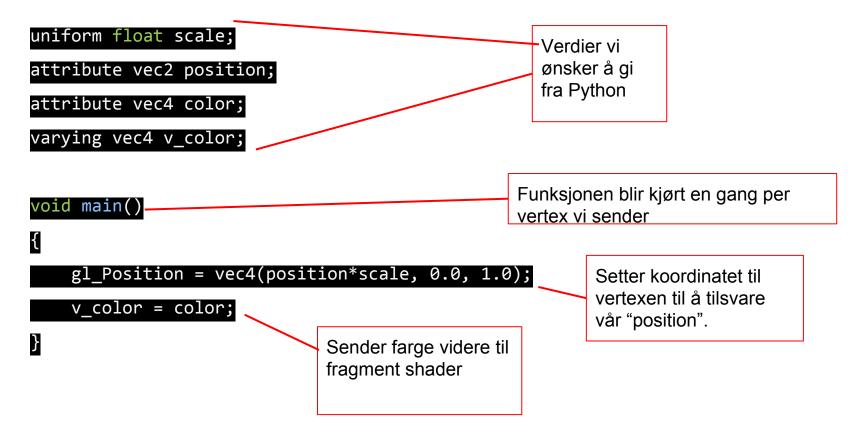
#### Kort om GPU 3

Vi bygger opp polygoner med å knytte vertexer (kanter) sammen.

Vi bestemmer selv hvor mange vertexer av gangen som skal settes sammen til et polygon.

Vi ønsker å ikke bruke mer avanserte

#### Vår første vertex shader



### Vår første fragment shader

varying vec4 v\_color;

Tar inn fargeverdien.

void main()

{

gl\_FragColor = v\_color;

}

Setter fragmentet til å ha samme fargeverdi som den vi sendte videre.

#### **Typer**

- Float, bool og int.
- Noen få skjermkort støtter double
- Vi har også structs fra C

#### Typer 2

- const compile time
- attribute Data som er forskjellig per vertex.
  Bare tilgjengelig for vertex.
- uniform Per primitiv (read only vertex og fragment)
- varying Skrives av vertex, leses av fragment.

### Hva er fargen mellom to punkter?

### **Hvor kommer Python og Vispy inn?**

Vispy binder GL, vindustegning og numpy sammen.

Vi skriver fortsatt shadere i GLSL.

Men vi kan generere formene/dataene våre i gode gamle python.

#### Vispy.app

App lager det vi ser på skjermen.

c = app.Canvas(keys='interactive')

### Vispy.gloo

Gloo lager bufferen og shaderene vi sender til GPU.

program = gloo.Program(vertex, fragment)

#### **Callbacks**

Vi henger en funksjon på en allerede eksisterende funksjon.

I vårt tilfelle ønsker vi kanskje at mer enn en ting blir gjordt når vi får en event.

Eksempelvis endre verden når vinduet blir skalert.

## Vi plotter en graf

### Vi lager en kube

#### Vi får kuben til å rotere

### Greie funksjoner å vite om

**Translate** 

Rotate

glModel

np.linspace

#### Vispy vs the world

matplotlib: Penere grafer for rapporter, men har lang kjøretid

PyQtGraph:

PyOpenGL:

Pygments: