Utvikling i VR med LÖVR

Jørgen Aarmo Lund

DIPS AS

Agenda

Forberedelser

Rask innføring i Lua

Utvikling i LÖVR

Forberedelser

Forberedelser

- Kode tilgjengelig på *https://dev.azure.com/dips/Spike/_git/dHack-VR-Development*
- Last ned LÖVR:
  - *https://lovr.org/downloads* (Antakelig vil du ha Windows, ZIP, 64 Bit)
- For å teste apper på Quest:
  - Guide på *https://developer.oculus.com/documentation/native/android/mobile-device-setup/*
  - Krever utviklermodus på Questen
  - Trenger ADB fra Android SDK, Oculus ADB-driveren fra *https://developer.oculus.com/downloads/package/oculus-adb-drivers/*

Hva er LÖVR?

- ”A simple Lua framework for rapidly building VR experiences”
  - Merk: Foreløpig *kun* VR, AR ikke offisielt støttet
- Håndterer stereoskopisk tegning, sporing av håndkontrollere, fysikksimulering, lyd
- Støtter Vive, Index, Quest og alt med OpenXR-støtte på Mac, Linux, Windows og Android
- Kan også brukes uten VR-briller, simulerer briller dersom de ikke er koblet til
- Kun lage små prototyper veldig raskt, men også plass for å implementere kompleks grafikk og logikk selv

Rask innføring i Lua

Hva er Lua?

- ”Lua is a powerful, efficient, lightweight, embeddable scripting language.”
- Først lansert i 1993, siden brukt som skriptmotor for mange spill, design- og serververktøy (f.eks. Redis)
- Lite men utvidbart kjernespråk med dynamiske typer, automatisk minnestyring
- Referanse: *Programming in Lua*
  http://www.lua.org/pil/contents.html
- Jukseark:
  http://thomaslauer.com/download/luarefv51.pdf

Grunnleggende

```
-- Kommentarer starter med to linjer.
-- Vi kan definere variabler slik:
tall = 12
ord = 'Dette er en string.'
sant = true
-- Løkker og blokker merkes ikke med krøllparenteser,
-- men med THEN-END eller DO-END:
if tall == 12 then
    io.write('Hurra!')
end
sum = 0
for i = 1, 5 do
    sum = sum + i
end
```

Funksjoner og tabeller

```
-- Vi bruker function() for å definere nye funksjoner:
function fib(n)
  if n < 2 then return 1 end
  return fib(n - 2) + fib(n - 1)
end
io.write(fib(5))
-- Lua har ikke arrays eller lister,
-- men tabeller (tables):
tabell = {['navn'] = 'Lua', ['versjon'] = 5}
io.write(tabell['navn'])

-- Et "array" i Lua er en tabell der nøklene er tall:
tabell = {10, 20, 30}
io.write(tabell[2]) -- Merk: vi teller fra 1!
```

Eksempelprogram

```
function is_prime(x)
    if x > 2 and x % 2 == 0 then
        return false
    end
    for i = 3, (x - 1) do
        if x % i == 0 then
            return false
        end
    end
    return true
end
for n = 2, 10 do
    if is_prime(n) then
        io.write(n, " er et primtall\n")
    end
end
```

Moduler

- Vi kan bruke *`require`* for å laste inn plugins og bibliotek:

```
local lib = require('library')

lib.doStuff()
```

- (Alle VR-funksjoner ligger under biblioteket *`lovr`*, som lastes inn automatisk)

Utvikling i LÖVR

Utvikling i LÖVR

- Last ned LÖVR, pakk ut til egen mappe: *`https://lovr.org/downloads`*
- Et prosjekt i LÖVR består av en mappe som inneholder en *`main.lua`*-fil
  - Utover *`main.lua`* kan du legge hva som helst du trenger av kode, grafikk og lyd i mappen
- Dra mappen på *`lovr.exe`* eller *`lovrc.bat`* for å starte prosjektet

Hello World!

Opprett en mappe med navn *hello*, og opprett en *main.lua* med innholdet

```
function lovr.draw(pass)
  pass:text('Hello World', 0, 1.7, -5)
end
```

Lagre filen og dra *hello* til *lovr.exe* - du skal nå kunne klikke og dra for å se deg rundt, og bevege deg rundt med WASD-tastene.

Hello World!

```
function lovr.draw(pass)
  pass:text('Hello World', 0, 1.7, -5)
end
```

- Programmet over overstyrer *lovr.draw*, som kjøres hver gang brillene trenger nye bilder
  - Vi kan også overstyre *lovr.load* for å sette opp programmet, og *lovr.update* for generell logikk
- Deretter kjører vi *text* på *pass* for å legge til en tekstsnutt på koordinatene (0, 1.7, -5) i listen over ting som skal tegnes
- Når *lovr.draw* har kjørt ferdig, vil kommandoene i *pass* sendes til skjermkortet

Pause/oppgave

3D-modellering

- Lag en liten 3D-scene i Bloom3D: *https://bloom3d.com/*
- A og D for å panorere kameraet venstre og høyre, W og S for å zoome inn og ut
- Høyre museknapp roterer kameraet
- *Rectangle* lar deg tegne opp firkanter, *Extrude* trekker ut firkanter til bokser
- Klikk *Download* når du er ferdig med figuren

Vise 3D-modeller

- Legg .glb-filen i prosjektmappen
- Last den inn i *`lovr.load`*:

```
function lovr.load()
    model = lovr.graphics.newModel('dips.glb')
end
```

- Bruk *`pass.draw`* for å tegne den:

```
function lovr.draw(pass)
    pass:draw(model, 0, 1.7, -3, 1.0)
end
```

(Endre den siste verdien dersom modellen er altfor stor.)

Belysning

- Vi ser modellen, men alle flatene blir helt hvite – vi har ikke noen lysmodell
- Moderne skjermkort kjører *shader*-funksjoner for hver piksel for å anslå hvilken belysning modellen skal ha
- LÖVR kommer med noen få innebygde shadere: vi kan f.eks. kjøre `pass:setShader('normal')` før vi tegner modellen for å sette på en regnbueeffekt
- Vi kan bruke en enkel *diffus* shader-funksjon for å simulere enkel lyssetting fra et lys - se `lighting`-mappen i repoet
- (Alternativt, se Simple Lighting fra LÖVR-guiden: `https://lovr.org/docs/Simple_Lighting`)

Belysning, del 2

- Opprett en ny shader-funksjon i oppstarten med `lovr.graphics.newShader`
- Aktiver den med `pass:setShader` i `lovr.draw`
- Still inn parametre (farge, lysplassering) med `pass:send` i `lovr.draw`

Animasjon

- Vi kan bruke funksjonen *`lovr.headset.getTime()`* for å se hvor lenge applikasjonen har kjørt
- Vi kan også sende inn størrelse og rotasjon som ekstra parametre til *`pass:draw`*
- Prøv å legge inn tiden som ekstra parameter

Kontrollere

- Kan bruke `lovr.headset.getHands()` for å få en liste over håndkontrollere
- `lovr.headset.getPosition(hand)` forteller oss hvor kontrolleren er (i 3D-koordinater)
- `lovr.headset.isDown(hand, knapp)` forteller oss om en knapp er trykt inn

Godt jobbet!

References i