Lysreklamedokumentasjon

# Installasjon

Installer *git* og *Gradle* først, og en IDE som f.eks. IntelliJ IDEA eller Eclipse. Snakk med noen av programmererne for å få laget en SSH-nøkkel og å få lagt den til i GitHub. Åpne en kommandolinje, og gå til en mappe hvor du har lyst til at prosjektkoden skal dukke opp i en undermappe. Kjør følgende for å laste ned prosjektkoden:

git clone git@github.com:JanOveSaltvedt/darkness.git

Det vil nå dukke opp en mappe som heter darkness, som inneholder koden for generatoren og simulatoren.

# DMX

DMX er en hardwareprotokoll for å styre lysarmaturer og andre scenetenkniske innretninger. Et *DMX-univers* består av 512 *kanaler*, nummerert fra og med 1 til og med 512. Hver kanal har til enhver tid en *verdi* mellom 0 og 255 (inklusive), og kobles typisk opp mot lyspærer slik at en høyere verdi skaper kraftigere lys. Ved å koble tre kanaler til en RGB-lyspære kan man skape alle mulige farger og intensiteter. En *DMX-strøm* består av en sekvens av *frames*, hvor hver frame definerer verdiene til alle kanalene - så det er omtrent som en film (som jo består av en sekvens med stillbilder). Vi opererer med 24 frames per sekund. Merk at det ikke er nødvendig å bruke alle kanalene; ubrukte kanaler lar man som regel ha verdien 0 hele tiden.

# Patternfil

Selve plasseringen av lyspærene, og hvilke kanaler som er koblet til hvilke pærer, er beskrevet i en *patternfil*. Dette er en ren tekstfil hvor hver linje beskriver én lyspære. Formatet til hver linje er slik:

*id x y type rc gc bc perimeter*

Dette er betydningen av de forskjellige feltene:

*id* Pæren sin id, som man kan bruke fra sekvensfilene til å referere til denne pæren. Må være et ikkenegativt heltall.

*x y* To desimaltall som beskriver pærens plassering på skiltet. Hva som er origo kan vi justere i simulatorkoden, men *x*-aksen går fra venstre mot høyre, og *y*-aksen går ovenfra og ned.

*type* En bokstav som indikerer pæretypen. Inneværende versjon av simulatoren støtter bare R, som betyr RGB-pære.

*rc gc bc* Tre heltall mellom 1 og 512 som indikerer hvilke DMX-kanaler som styrer henholdsvis denne pærens røde, grønne, og blå komponent.

*perimeter* (Valgfritt) Angir formen til metallveggene som omgir pæren. Ikke støttet i nåværende simulator, som vil ignorere dem hvis de står der (bortsett fra at pæreplasseringen vil bli litt påvirket av dem). En liste med et partall antall desimaltall, minst tre par. Hvert par indikerer koordinatene til et hjørnepunkt i veggen, relativt til pæren sin plassering.

Her er et eksempel på en definisjon av en lyspære:

42 44.90 31.00 R 242 342 442

Dette er en RGB-pære med id 42, som befinner seg på (44.90, 31.00), og som styres av kanalene 242 (rød), 342 (grønn) og 442 (blå).

# Scriptklasser

Sekvenser defineres av *scriptklasser*, som arver fra ScriptBase, og de bør ligge i pakken darkness.generator.scripts.uka15 i generator-prosjektet. Se eksempelscriptet DemoScript.

Koden i et script må defineres i metoden public void run(). Først bør man definere hvilke enkeltpærer og pæregrupper man jobber med (lag gjerne en baseklasse som gjør dette), f.eks.:

BulbRGB topOfJ = bulb(6);

BulbGroup J = group(0, 1, 2, 3, 4, 5, 6);

Fra nå av vil topOfJ referere til pæren med id 6 fra patternfilen, og pæregruppen J refererer til alle pærene fra og med 0 til og med 6. Merk at samme lyspære godt kan være med i flere grupper.

Etter dette kan man begynne og definere en serie med *effekter*. En effekt er en kommando som medfører en endring i kanalverdier. Noen effekter har bare en umiddelbar virkning, mens andre varer over flere frames. I utgangspunktet starter man med en DMX-frame hvor alle kanalene er 0. Alle effekter blir anvendt på denne frame'n, helt til man sier next(), som indikerer at man er ferdig med denne frame'n og vil begynne på en ny. Den nye frame'n vil starte med å ha samme kanalverdier som sluttresultatet i den forrige, men disse kan endres av nye effekter og av effekter som er blitt startet i tidligere frames og allerede er i gang. Dersom man vil generere mange frames uten å legge til nye effekter, kan man bruke wait(*x*), som vil generere *x* frames.

De fleste effektene tar et BulbSet som parameter. Dette kan enten være en BulbRGB (en enkelt lyspære) eller en BulbGroup (en samling med flere lyspærer), og effekten vil påvirke alle de angitte pærene. Noen av effektene tar utgangspunkt i de aktive kanalverdiene på det tidspunktet effekten blir startet, og for pæregrupper vil dette være kanalverdiene til den første pæren.

## Effekter

Subklasser av ScriptBase har tilgang til metodene setRGB() og setHSB(), som begge tar et BulbSet som parameter og deretter komponentene til en RGB- eller HSB- farge. De(n) angitte pæren(e) vil da få den angitte fargen i den aktive frame'n, og vil holde på denne fargen til en annen farge eller effekt blir aktivert. Merk at selv om man angir en BulbGroup, er det ikke slik at pærene "henger sammen", så man kan godt starte en effekt senere som bare påvirker noen av pærene i den gruppen. Disse kommandoene er strengt tatt ikke effekter, siden de har en umiddelbar virkning og ikke har noen varighet.

En effekt er en subklasse av EffectBase, som igjen er en subklasse av ScriptBase. En effekt startes ved å instansiere klassen og sende objektet som parameter til effect(). Dette kan forenkles ved å legge til en eller flere hjelpemetoder per effect i ScriptBase. En effektklasse må override run(), og koden i denne må være en løkke som i hver iterasjon setter de ønskede pærefargene og deretter kaller next() (som ikke vil lage noen ny frame som sådan; det sier bare "denne effekten er ferdig med det som skal gjøres i denne frame'n; neste iterasjon beskriver neste frame"). Hittil har vi to effekter: RGBFade og HSBFade, som fader en pære eller en pæregruppe fra én farge til en annen i løpet av et visst antall frames, ved linær interpolering av enten RGB-komponentene eller HSB-komponentene.

## Subsekvenser

Man kan kombinere flere sekvenser - typisk hvis man vil gjenta en bestemt ting flere ganger. Man lager da en subklasse av ScriptBase som gjør denne tingen (f.eks. å få en av bokstavene til å blinke i ett sekund), og man kan når som helst lage en instans av subsekvensklassen og kalle merge() med instansen som parameter; dette vil starte avspilling av subsekvensen i parallell med hovedsekvensen.

Subsekvenser kan også starte hverandre rekursivt (dvs. at subsekvensene kan ha egne subsekvenser).

Hvis flere subsekvenser kjemper med hverandre eller med hovedsekvensen om samme pære, er det den "ytterste" sekvensen som vinner (typisk hovedsekvensen).

Kun de pærene som aktivt settes av subsekvensen vil bli påvirket (så det er ikke slik at alle de andre pærene blir svarte, bortsett fra de pærene som subsekvensen setter til svart).

## Testing av script

Åpne simulator/build.gradle og endre linjen som begynner med args. La oss si at du vil bruke patternfilen simulator/patterns/skilt-013.txt, og at du vil spille av scriptklassen MySequence. Skriv i så fall dette (merk at alt er på samme linje, og at det ikke er noe mellomrom eller linjeskift etter bindestrekene, og at filnavnene er relative til simulator-mappen):

args = ["--pattern", "./patterns/skilt-013.txt", "--script", "MySequence"]

Hvis scriptklassen ligger i en annen pakke enn darkness.generator.scripts.uka15, må pakkenavnet også oppgis, f.eks. "mypackage.MySequence".

Alternativt, hvis du har en PGM-fil som du vil spille av (f.eks. simulator/sequences/uka13/uka13.pgm), kan du bruke disse argumentene:

args = ["--pattern", "./patterns/skilt-013.txt", "--sequence", "./sequences/uka13/uka13.pgm"]

Åpne en kommandolinje og gå til rotmappen for darkness, og kjør følgende:  
gradle simulator:run

# PGM-filer

Vi bruker det enkle bildeformatet PGM (Portable Graymap Format) til å lagre genererte sekvenser - siden hver frame kan representeres som én rad i et bilde, kan en hel sekvens representeres som et helt bilde. Dermed blir det også lett å skrive C-programmet som konverterer sekvensene til DMX som sendes til selve skiltet. Dessuten er PGM er et tekstbasert format, noe som gjør det enkelt å debugge, og man kan til og med håndskrive filene. Formatet er slik:

* Første linje inneholder den magiske teksten P2, som indikerer at dette er en gråskala PGM-fil
* Andre linje inneholder to heltall: høyden og bredden på bildet. Hos oss vil bredden alltid være 512 (antall kanaler), og høyden vil være antallet frames.
* Tredje linje inneholder verdien som indikerer hvitt. Alle pikselverdiene må være mellom 0 og dette tallet, inklusive (0 indikerer svart). Normalt vil dette tallet være 255. Vi har innført et lite hack: hvis tallet er 256, betyr det at piksler med verdi 256 er gjennomsiktige - dette benyttes bare til "overlays", som er animasjoner som kan spilles av "oppå" andre animasjoner (usikkert om vi kommer til å ta dette i bruk).
* De påfølgende linjene (like mange som høyden på bildet) inneholder 512 heltall hver, mellom 0 og tallet på tredje linje.