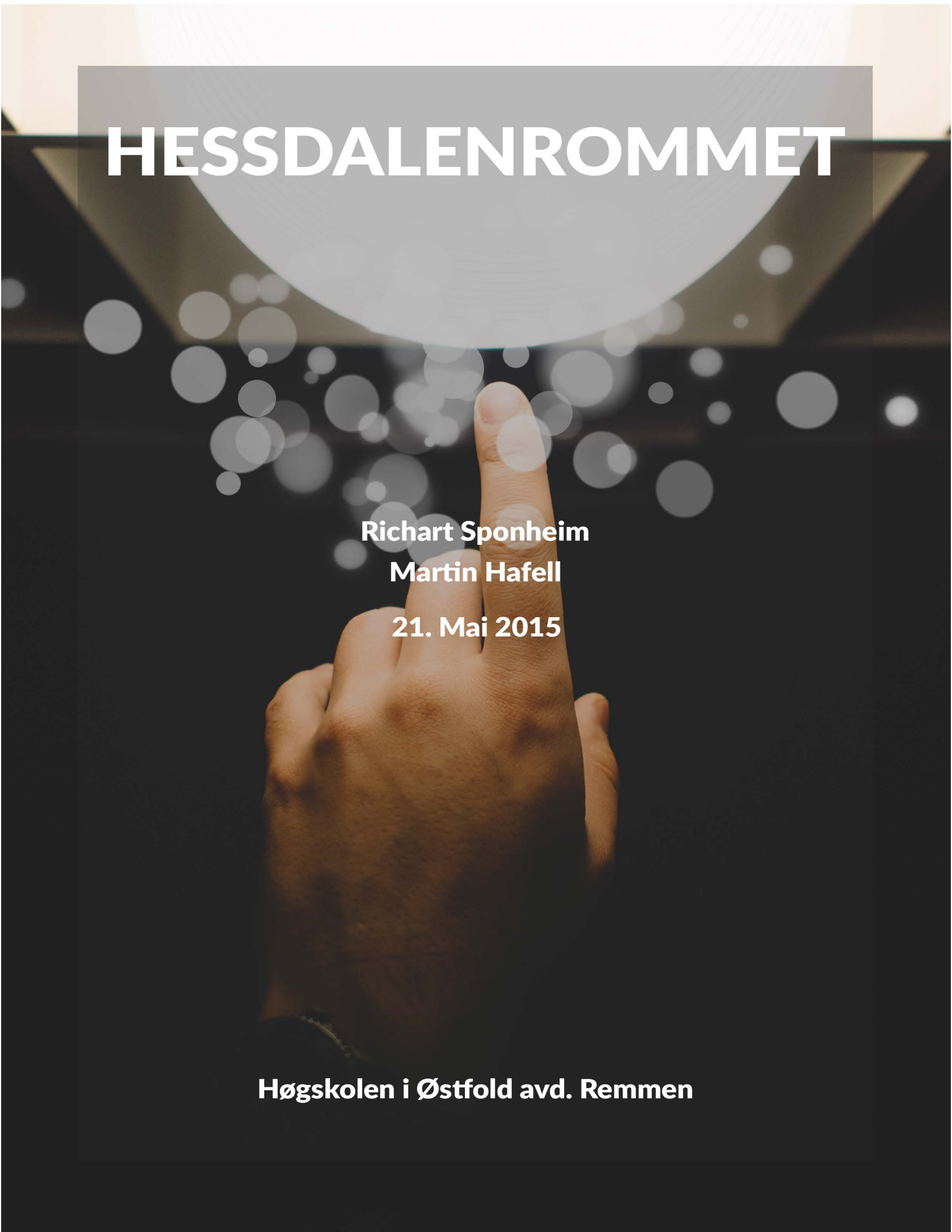
HESSDALENROMMET



Richart Sponheim
Martin Hafell

21. Mai 2015

Høgskolen i Østfold avd. Remmen

Høgskolen i Østfold

Avdeling for Informasjonsteknologi
Remmen
1757 Halden
Telefon: 69 21 50 00
www.hiof.no

Bacheloroppgave

Prosjektkategori: Bachelorprosjekt
Ompanf i studiepoeng: 20
Fagområde: Datateknologi

Oppgaven er fritt tilgjengelig for alle.

Rapporttittel: Hessdalenrommet	Dato: 21 Mai 2015 Antall sider: Antall Vedlegg:
Forfattere: Martin Hafell, Richart Sponheim	Veileder: Erling Strand
Avdeling/linje: Informasjonsteknologi	Prosjektnummer: BO15 - G30

Utført i sammarbeid med: Inspiria Science Center - Sarpsborg	Kontaktperson: Yngve Ellingsen

Ekstrakt:
Hessdalenrommet mangler interaktivitet. Formålet med prosjektet er å gjøre Hessdalenrommet mer interessant ved å skape interaktive og spennende aktiviteter som er knyttet til fenomenet. Resultatet av prosjektet vil gi økt interesse rundt Hessdalenfenomenet.

3 emneord:

Hessdalenfenomenet	Spill	Interaktivitet

Sammendrag

Hessdalenrommet ved Inspira Science Senter er laget for å informere om Hessdalenfenomenet, et lysfenomen som fortsatt er under forskning. Problemstillingen er at rommet i seg selv ikke er interaktivt nok til at folk interesserer seg for Hessdalenrommet. Vi har derfor fått i oppgave om å løse dette ved å gjøre rommet mer spennende og interaktivt.

Oppgaven var fra starten av ganske åpen og vi måtte selv finne ut hva som var den beste løsningen med hensyn til budsjett og kompetanse. Etter analyse av Hessdalenrommet og samarbeid med oppdragsgiver, kom vi fram til at vi skulle lage flere prototyper. Både fysiske og teknologiske.

Vi valgte å fokusere på tre av våre idéer. To av idéene baserte seg på programutvikling, hvor det ene programmet er et spill og det andre et presentasjonprogram. Spillet handler om å fange og sortere lys i Hessdalen. Presentasjonsprogrammet skal gi besøkende en oversiktelig platform for visning av bilder og video av fenomener i Hessdalen. Den tredje idéen presenterer gasser i form av plasma, der fokuset er å vise fram hvordan gasser reagerer ved høy spenning. Idéen ble basert på å kunne analysere selvlysende gass med emisjonsspekter.

Takk til

Vi vil gjerne takke Erling Strand for god veiledning underveis i prosjektet. Du sørget for at vi ikke mistet den røde tråden underveis, og kom med både gode idéer og råd, som vi ikke hadde klart oss foruten. Vi vil takke arbeidsgiver Yngve Ellingsen ved Inspiria for et godt samarbeid, en spennende oppgave som ga oss utfordringer og lærdom innenfor flere faglige områder. Vi vil også takke medarbeidere ved Inspiria som har bidratt med testing av prototyper.

Innhold

Figurliste:

Kapittel 1
Introduksjon

1.1 Prosjektgruppen

Prosjektgruppen består av to personer fra linjen Dataingeniør ved Høgskolen i Østfold. Begge er glad i løse oppgaver som behøver både teoretisk og praktisk utførelse. Følelsen av å skape et produkt som kan komme til nytte er noe som vi verdsetter høyt.

Richart Sponheim
Alder: 23 år.
Bosted: Sarpsborg.
Linje: Dataingeniør
Interesse: Front-end Design, IKT-innovasjon, trening og spill.
Annen kompetanse: Butikkmedarbeider/Selger

Martin Hafell
Alder: 23 år.
Bosted: Sarpsborg.
Linje: Dataingeniør.
Interesse: Programmering, datateknikk, musikk, dataspill og øl.
Annen kompetanse: Musikkutdannelse.

1.2 Oppdragsgiver

Inspiria science center er et populærvitenskapelig opplevelses- og læringssenter i Sarpsborg innen matematikk, naturvitenskap og teknologi hvor de besøkende lærer ved å eksperimentere selv. I et vitensenter kan barn og voksne utforske fenomener knyttet til natur, miljø, helse og teknologi gjennom egen aktivitet og i samarbeid med andre.

1.3 Oppdraget

Oppdraget vårt går ut på å gjøre Hessdalenrommet mer attraktivt for besøkende ved Inspiria science senter. Dette skal gjøres ved å implementere interaktive løsninger. Dette kan være alt fra fysiske spill eller utstillinger, til programmerte spill eller utstillinger som handler om lys og plasma, noe som er et sentralt tema i hypotesene. Det er meningen å få besøkende til å bli mer interessert og lære om Hessdalenfenomenet gjennom aktiviteter.

1.4 Formål, leveranser og metode

Hessdalenfenomenet er et av de store gjenværende mysteriene i fysikken. Inspiria jobber for å skape interesse og undring, og Hessdalenfenomenet vekker begge deler hos publikum. Siden vi i mange år har samarbeidet med Høgskolen om Hessdalen, er det viktig for oss å få vist fram hva som foregår der.

1.4.1 Formål

Formålet med prosjektet er å gjøre Hessdalenrommet mer interessant ved å skape interaktive og spennende aktiviteter som er knyttet til fenomenet. Resultatet av prosjektet vil gi økt interesse rundt Hessdalenfenomenet.

Hovedmål

Hovedmålet i prosjektet er å utvikle flere prototyper med hensyn på formålet, som skal hjelpe Inspiria med å skape nye produkter til Hessdalenrommet. Prototypene skal være en oppgradering av både eksisterende og nye utstillinger som er laget ved å kombinere fysikk, elektronikk og teknologi. Prosjektet skal dokumenteres i en hovedrapport som beskriver arbeidet og prototypene.

Delmål 1

Gjøre en analyse av Hessdalenrommets originale løsning. Gjennom analysen skal vi finne ut hva slags utstyr og teknologi som er i bruk, for å få en formening om hva som fungerer eller ikke fungerer i forhold til underholdningsverdi og lærdomspotensiale.

Delmål 2

Lage en detaljert plan over hva slags idéer vi har som kan forbedre Hessdalenrommet. Detaljplanen skal ta for seg hvordan rommet var da vi begynte og en forklaring av idéene som skal bli til prototyper.

Delmål 3

Utvikling av prototypene. Dette innebærer dokumentasjon og forklaring i detaljert form hvordan prototypene fungerer, og gjennomføring av prosjektet.

Delmål 4

Brukertester og fremtidig arbeid.

Delmål 5

Fullføring av prosjekt. Alt arbeid som er gjort skal dokumenteres i hovedrapporten.

1.4.2 Leveranser

Skolen har følgende innleveringskrav:

- **9. Januar -** Hjemmeside.
- **16. Januar -** Forprosjektrapport.
- **13. Mars -** Hoveddokument v1.
- **24. April -** Hoveddokument v2.
- **21. Mai -** Hoveddokument v3, endelig utgave.
- **1. Juni -** Opphenging av prosjektplakat.
- **3. Juni -** Presentasjon av prosjektet.

Våre leveranser til oppdragsgiver vil være en oppnåelse av hovedmålet, som innebærer dokumentasjon av prototyper, arbeidet og analyse. Det skal også leveres alt av kildekode, prosjektfiler og utstyr som tilhører prototypene.

1.4.3 Metode

For å få en oversikt over Hessdalenrommet og hva som burde gjøres, vil vi analysere rommet. Vi bruker informasjonen fra analysen til å lage nye idéer og et løsningsforslag til prosjektet. Videre skal idéene utforskes og drøftes med arbeidsgiver. Hvis en idé er gjennomførbar og godkjent av arbeidsgiver, skal den gå videre til utvikling. Til slutt skal prototyper gå til testing.

Arbeidsmetoden til gruppen vil hovedsaklig være gruppearbeid, der vi jobber med samme oppgave hele veien. Siden vi har noe forskjellig kompetanse, fordeles arbeidet innenfor oppgaven etter behov. Vi arbeider tett med veileder og oppdragsgiver slik at vi stadig kan få tilbakemeldinger.
I gruppen jobber vi på papir eller i Google Docs, noe som gjør det lett å samarbeide på dokumenter. For å sikre arbeidet bruker vi Google Drive som automatisk lagrer dokumenter og filer både på en skydisk over nett og på harddisk gjennom applikasjon. Det vil bli brukt kommersiellt utstyr til utføre oppgavene.

1.5 Rapportstruktur

Kapittel 2 handler om romanalysen. Vi vil analysere alle aspektene til rommet og forklare hvordan vi vil utvikle det. Vi tar også en vurdering av tidligere idéer gitt av Inspiria. I Kapittel 3 går vi gjennom planlegging og beskrivelser av prototyper. Kapittel 4 forklarer fremgangsmåten til utviklingen av prototypene. Kapittel 5 tar for seg egen testing og brukertesting.I Kapittel 6 diskuteres resultatene og hvordan det har gått i forhold til målene vi hadde satt. Kapittel 7 konkluderer rapporten.

Kapittel 2

Analyse

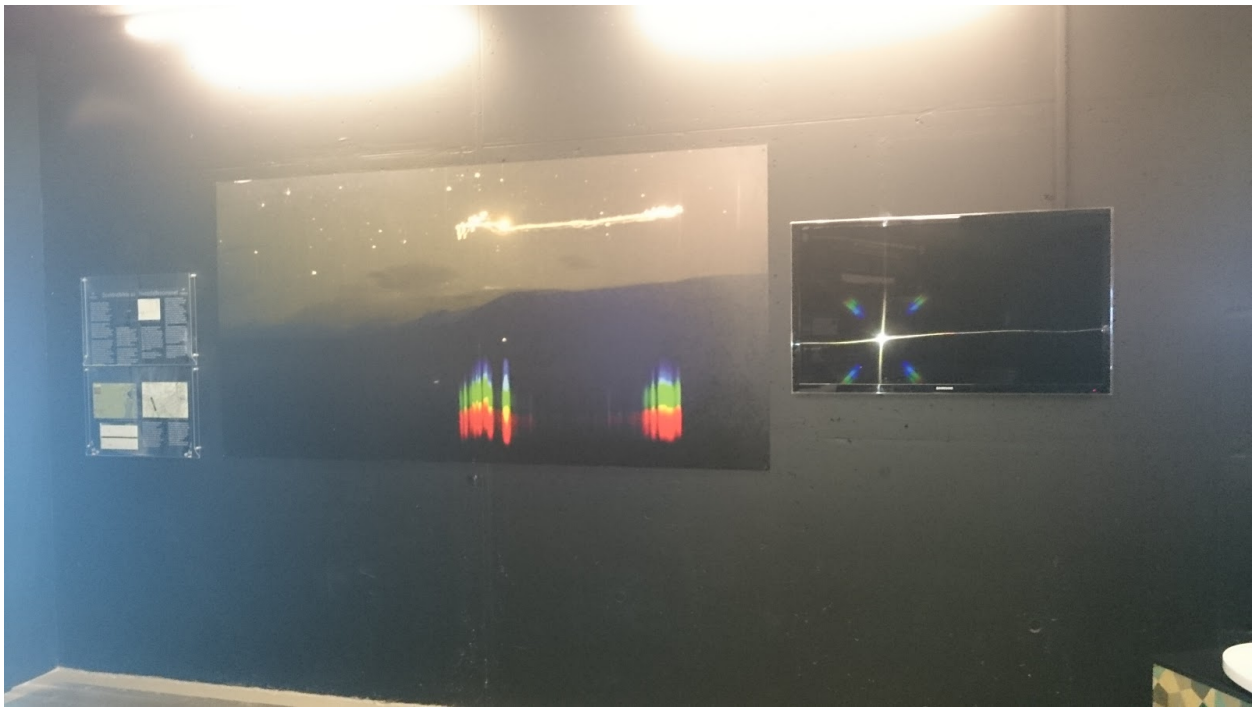
Hessdalenrommet består av fem vegger, fylt med forskjellig informasjon og forklaringer rundt hva som skjer i Hessdalen. Det er også flere skjermer, både interaktive og lineære, men samtidig ikke særlig spennende og funksjonelle. Vi skal ta for oss vegg for vegg, og forklare hvordan de fremstod under startfasen.

2.1 Hessdalenrommet før prosjektet:

“Vegg 1”

Vegg 1 består av følgende attraksjoner;

- En plakat med spektralbilde av et avbildet fenomen i Hessdalen.
- En TV-skjerm som viser en dokumentar om Hessdalenfenomenet.



Figur 2.1: Bilde av "Vegg 1" fra Hessdalenrommet

Analyse:

Plakaten gir en god introduksjon til rommet, med et spektrabilde av fenomenet i Hessdalen, men samtidig lite informativ for de som ikke vet hva lysfenomenet omhandler. TV-skjermen viser en dokumentar om Hessdalenfenomenet, som spilles om og om igjen. Dokumentaren er lang og utdatert, ikke særlig spennende og kan fort bli ignorert på grunn av for treg informasjon. Selv for de mest interesserte kan dette være lite underholdende.

"Vegg 2"

Vegg 2 består av følgende attraksjoner;

- Geigerteller
- 3 TV-skjermer med direktesending fra kameraer i Hessdalen
- En plakat med en illustrert forklaring av bølgelengde til lys.

Figur 2.2: Bilde av "Vegg 2" fra Hessdalenrommet

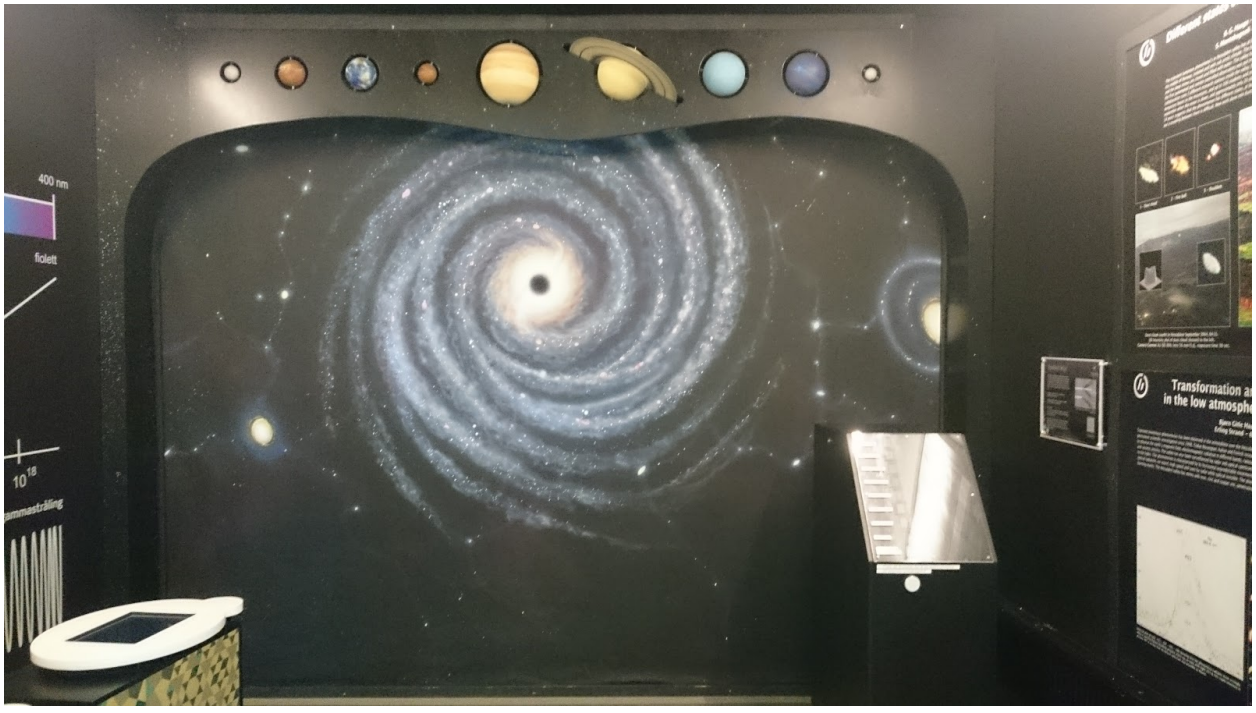
Analyse:

Geigertelleren skaper en mystisk atmosfære i rommet, men det var ikke selvforklarende hva den faktisk gjorde, som er å måle stråling. Førsteinntrykket til direktesendingen fra Hessdalenrommet gjør en nysjerrig, men nysgjerrigheten blir kortvarig da sjansen for å få et glimt av fenomenet er utrolig liten. I tillegg er Direktesendingen nokså ustabil, noe som gjør at man ved mange tilfeller kun ser en svart skjerm. Plakaten viser sammenheng mellom frekvens og bølgelengde. Det er også tatt ut hvilket frekvensområdet lys er synlig. Dette hjelper besøkende og forstå et emisjonsspekter. Men dette tar veldig mye plass, i forhold til at det kun er lett forståelig for spesielt interesserte.

“Vegg 3”

Vegg 3 består av følgende attraksjoner;

- En stor plakat av en galakse
- Modeller av planetene i vår galakse



Figur 2.3: Bilde av “Vegg 3” fra Hessdalenrommet

Analyse:

Veggen i seg selv er en galakseplakat som gir rommet en utenomjordisk atomsfære. Planetmodellene ser bra ut og skaper en helhet i konseptet deres. Denne veggen kunne blitt brukt på mer effektiv måte.

“Vegg 4”

Vegg 4 består av følgende attraksjoner;

- Plakat med detaljert informasjon om Hessdalenfenomenet på engelsk
- VLF Mottager
- Skjerm med graf tilhørende VLF mottager
- Plakat som omtaler hva VLF mottager er

Figur 2.4: Bilde av “Vegg 4” fra Hessdalenrommet

Analyse:

Plakaten består av detaljert informasjon om Hessdalenfenomenet, på engelsk. Den henger relativt høyt opp på veggen, noe som gjør det vanskelig å lese. Teksten på plakaten er også skrevet på en tung akademisk måte, som gjør den utfordrende å forstå.

VLF mottakeren er et godt bidrag til rommet, men vi føler den kunne blitt presentert på en bedre måte. Det er ikke lett å forstå sammenhengen mellom grafene og VLF mottakeren. Det er også lave utslag om man bare bruker armene i mottakeren. Det burde også vært bedre forklart hvilken sammenheng den har med Hessdalenfenomenet.

"Vegg 5"

- Informasjon om Hessdalenrommet

Figur 2.5: Bilde av plakat med informasjon og samarbeidspartnere

Analyse:
Denne plakaten gir kort, presis informasjon og inneholder også samarbeidspartnernes logoer.

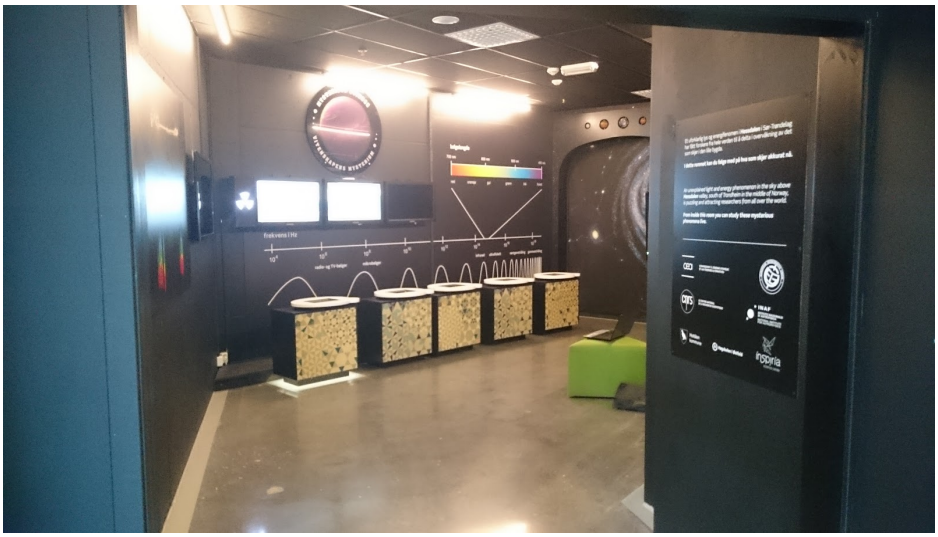
"Gulv"

Gulvet består av følgende attraksjoner;

- Touch skjermer:
  - Om overvåking geigerteller.
  - Om overvåking med spektralkamera.
  - Om overvåking himmelen.
  - Om overvåking temperatur i himmelen.

- Om overvåking lavfrekvens.

- Lysspekterboks.



Figur 2.6: Bilde av Lysspekterboks Figur 2.7: Bilde av inngangsparti til Hessdalenrommet

Analyse:

Touchskjermene består kun av informasjon om forskjellige temaer. Det er ingen form for interaktivitet, annet enn å velge hva du vil lese om. Dette er veldig tung informasjon.

Lysspekterboksen er en svart boks med en utydelig forklaring over hvordan den fungerer. Boksen i seg selv er en god idé, men den kunne blitt presentert på en mer forståelig måte.

Konklusjon

Gjennom analysen fant vi ut at det er mye informasjon på et akademisk nivå. Ting tar lang tid å forstå, selv for oss. Det er lite fokus på ren underholdning. Det er et lavt interaktivitetsnivå. Det er rotete oppsett i form av at informasjonen ikke blir levert på en systematisk måte. Det er mye god fakta, men lite som skaper undring. Rommet som helhet er mørkt, dystert og lite innbydende.

I kapittel 6 legger vi fram løsninger og forbedringer som kan bli gjort for å øke interaktiviteten med fokus på underholding og undring.

2.2 Ressurser knyttet til prosjektet

Vi hadde følgende ressurser knyttet til Hessdalenrommet:

- 5 touchskjermer koblet til hver sin dedikerte datamaskin.
- 1 stor TV med en dedikert datamaskin.
- 3 Skjermer koblet til en dedikert datamaskin.

Budsjett:

Det var ikke planlagt noe budsjett fra starten av prosjektet fra oppdragsgivers side. Via samarbeid med oppdragsgiver ble det avklart hvor mye penger vi fikk til kritisk materiale. Inspiria hadde også tilgang til ressurser som kunne brukes til utvikling som vi tok nytte av. Det ble også brukt eget materiale som var eid av gruppa til utvikling. Gruppa kjøpte også inn noe eget utstyr som Inspiria var villig til å dekke.

2.3 Inspirias liste med idéer og forbedringer til utstillingen

Momenter fra idemyldring i/om Hessdalen-rommet 12.2.2014.

1. Skilt ut mot utstillingsområdet mangler
2. Lys og neonskilt i inngangen, ikke “Hessdalen”. Bare spesielt interesserte har hørt om fenomenet.
3. Knytt utstillingene utenfor til rommet
4. Lag “grønn mann” i inngangen og grønne fotspor inn i rommet
5. Belysning i inngang + hologram av person som ønsker velkommen inn
6. Bruk hele veggen ved inngangen til illustrasjoner av UFOer, kloder osv.
7. Forheng i inngangen - skjermer rommet og gjør det skummelt
8. Hva er Hessdalen-fenomenet? “Enkel” forklaring savnes
9. Kopling som viser at Hessdalen har ekte UFO
10. En skjerm som viser observerte fenomener
11. Plakater må på norsk
12. Rommet er for mørkt
13. Må være inviterende
14. Musikk i rommet, X-files-type
15. Hessdalenlyd når fotocelle brytes
16. Når man går inn kommer lysglimt/blitzlys
17. Tåkemaskin starter når noen går inn i rommet
18. I forhold til planetene på vegegn => snorer å dra i som viser hvor tung en jordkilo er på den andre planeter
19. Flere interaktive saker
20. Opplæring sv verter i rommet, orienteringer på gitte tidspunkter i helgene. Praktiske hans-on elementer som illustrerer det som vises på veggene.
21. Enklere språk(eksempelvis er VLF-måleren full av liten tekst som må leses for å forstå skjermen ved siden av.)
22. Planetene i taket bør senkes i barnevennlig “ta på” høyde
23. Spektrallinje-boks er ikke i drift. Tekst-lappene for enkle
24. Lag saker og ting som koples mot fenomenet
25. Høyspenningskilde som får håret til å reise seg
26. Oppgaver i rommet som kårer dagens Hessdalenmenster. Quiz.
27. Har vi duppeditter som benyttes i Hessdalen som kan stilles ut?
28. Hologramperson som dukker opp og forklarer rommet
29. Bruk taket til noe
30. Bruk Geir Ø sine erfaringer fra “tilsvarende utstilling på gamle ØIH.

Kapittel 3
Design og planlegging av produkter

3.1 Vurdering av muligheter

Vi hadde mange idéer og forslag til hva vi kunne forandre eller forbedre i Hessdalenrommet, men måtte velge et fåtall av disse for å få ferdig noen prototyper i tide. Vurderingen gikk ut på å velge det som virket mest interaktivt og gjennomførbart. Dette ble gjort i samarbeid med oppdragsgiver. Her var idéene som ble tatt til vurdering:

Lysboks:
Vi ville vise fram gass omgjort til plasma ved høy spenning på en interaktiv måte, der brukeren må gjøre noe fysisk for å aktivere funksjonen. Dette er grunnet en sammenheng med emisjonsspekter.

Hologram:
Et hologram som utstilling for å vise fram lysfenomen på en mer spennende måte.

Spill:
Et spill der man skal fange lys i et miljø som simulerer Hessdalen.

Quiz:
En quiz som skal ha spørsmål med alternativer relatert til Hessdalenfenomenet og fysikken rundt det, der man kan bruke rommet som kilde til svar på oppgavene.

Presentasjonsprogram:
Et program som på en effektiv måte viser fram bilder og video med god kvalitet, tatt av Hessdalenfenomenet.

Samling og forenkling av informasjon:
Gjøre det enkelt å lære om Hessdalen ved å bearbeide informasjonen som allerede er presentert gjennom touchskjermene.

Magnet i VLF-måler:
Feste en magnet med tråd i VLF måleren, som kan svinges innenfor bruksområde for å få større utslag. Dette skal gjøre det tydeligere hvordan bruke VLF-måleren.

Gjøre data fra VLF måler mer interessant:
Dataen fra VLF måleren kan bli fremstilt på en annen måte, for eksempel et bedre grafisk design eller en ikke-digital metode.

Høyttaler til VLF måler:
Koble til en høyttaler til VLF måleren for å skape stemning og gjøre det mer morsomt å bruke VLF-måleren.

Direktesending med 12 timer forsinkelse:
Å forsinke direktesendingen fra Hessdalen med 12 timer vil gjøre at man får se Hessdalen om natten. Sannsynligheten for at kameraet får fanget opp lyset fra et fenomen.

"Quizkampen":
Sette opp en utstilling med quiz der to deltagere kan konkurrere mot hverandre.

3.2 Valg av løsning

Vi endte opp med å velge tre idéer som skulle utforskes og gjennomføres, grunnet estimert tid til prosjektene og basert på hva som virket mest verdifullt for Hessdalenrommet. Dette er også basert på analysen.

3.2.1 Lysboks

Idéen var å vise frem forskjellige gasstyper når de reagerer med høy spenning og omformes til plasma. Gassene er oppbevart i små ampuller som skal installeres i en utstillingsboks. For å gjøre dette interaktivt, skal disse gassampullene aktiveres ved hjelp av en ultralydsensor. For å få dette til å fungere bruker vi en mikrokontroller til styring.

Denne idéen ville vi utføre fordi det skaper undring rundt Hessdalenfenomenet, siden lysene fra gassampullene kan ligne på fenomener. Forskjellige lys har et eget karakteristisk emisjonsspekter. Med emisjonsspekter kan man fastslå hvilken gass som blir observert, noe som er relevant til forskningen i Hessdalen.

3.2.2 Presentasjonsprogram

Med denne idéen har vi et ønske om å lage en platform man kan bruke til å vise bilder og video av fenomenet. Tanken bak denne idéen var å kunne presentere både gamle og nye bilder og video relatert til fenomenet på en effektiv måte. Det skal være enkelt å oppdatere bildene og videoene.

3.2.3 Spill

Vi ville lage et underholdende spill for besøkende, med fokus på Hessdalenfenomenet. Spillet skulle være såpass utfordrende slik at spilleren ble engasjert, og at det kunne utviklet seg til en konkurranse mellom besøkende om å oppnå den beste poengsummen.

Spillet handler om å fange og sortere forskjellige lys som flyr rundt i et illustrert Hessdalen. Der har vi valgt å kalle lysene for "Flylys", "Bil-lys" og "Hessdalenfenomen". Dette er relatert til forskningen på fenomenet, der det er viktig å kunne skille lysene fra hverandre. Vi ønsker at spillet skal skape undring rundt forskningen i Hessdalen.

Grunnen til at vi valgte å lage et spill, er fordi det er veldig interaktivt, og en god selvbetjent underholdning som faller i smak for en stor del av befolkningen[1], spesielt unge mennesker.

3.3 Utfordringer ved løsningene

3.3.1 Lysboks

Da vi startet med prosjektet på lysboksen så vi for oss at det skulle være en utstilling med lys av gass som styres av fysiske bevegelser. Med dette systemet vil det være flere utfordringer som må løses for å kunne utvikle en prototype:

1. Få tak i gassampuller som kan brukes i en utstilling
2. Gjøre om gassen til plasma - med andre ord, få ampullene til å lyse
3. Finne komponenter som kan kobles til en mikrokontroller
4. Koble sammen komponenter til en mikrokontroller
5. Lage et kabinett til å fremstille lysene.
6. Forholde oss til et lavt budsjett.

3.3.2 Presentasjonsprogram, Spill

Før man begynner å lage et program er det en rekke utfordringer å ta hensyn til:

- Valg av programmeringsspråk - må støtte all funksjonalitet man ønsker.
- Layout - må være forståelig og ryddig.
- Stabilitet - minimalt med bugs og forsinkelser i programmet.
- Valg av utviklingsplatform - skal være enkelt og bruke.
- Programmet må være interessant/morsomt/utfordrende.
- Klare å lage et tilfredsstillende design.

3.4 Design

Design spiller en stor rolle for helheten av produktene siden utstillingen skal tiltrekke besøkende. Vi ville gjenspeile Hessdalen så godt vi kunne gjennom designet, der nøkkelordet for oss var rå natur. Samtidig tok vi inspirasjon i hvordan programdesign er i dag og vår kompetanse innenfor dette området.

3.4.1 Krav til design

Kravet vi satt oss for produktene, var at det skulle være godt nok til å brukes i utstillingen ved Hessdalenrommet. For å oppnå dette kravet, jobbet vi med disse retningslinjene:

- Det grafiske designet skal reflektere Hessdalen med tanke på geografiske kjennetegn og/eller fenomen.
- Det skal være enkelt og selvforklarende å bruke for alle.
- Viktig å vekke undring rundt fenomenet.
- Vi vil at produktet skal ha et nivå med en viss standard for en utstilling:
    - Utseende - se profesjonelt ut
    - Forutsigbart - optimalisere ytelse slik at den er kontinuerlig.
- Det skal være enkelt å installere for arbeidsgiver.

3.4.2 Produktdesign

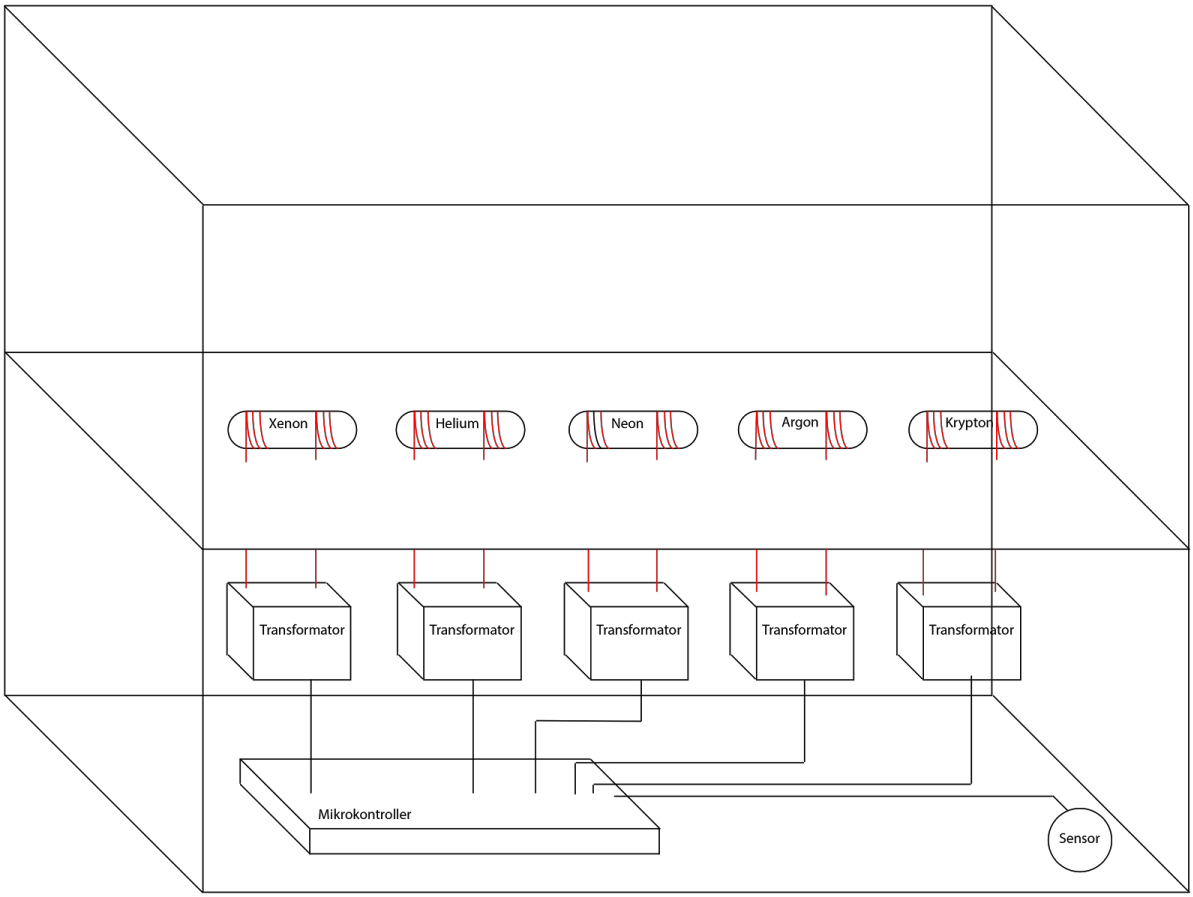
I dette kapittelet forklarer vi hvordan produktene skulle se ut og fungere ved slutten av prosjektet, forbeholdt kravene i kapittel 3.4.1, men med unntak av det grafiske designet.

3.4.2.1 Lysboks

Lysboksen skal være et fysisk objekt som skal kunne stilles ut i Hessdalenrommet. Det vil være mulighet for å feste boksen på veggen ved bruk av skruer. Boksen skal fremstille fem gassampuller med hver sin type gass. Gassen vil gi fra seg hvert sitt karakteristiske lys. Rundt hver gassampull er det to ledertråder. Ledertrådene blir satt på som en spole på hver sin side av gassampullen. Disse to ledertrådene skal ha stor nok avstand fra hverandre, slik at det ikke det oppnås en kobling direkte mellom ledertrådene.

Gassampullen skal fungere som en leder mellom de to ledertrådene ved tilstrekkelig spenning. Da vil gassampullen gi fra seg sitt karakteristiske lys. Ledertrådene er koblet opp til transformatorens utgang. Ved inngangen av transformatoren skal det være en styrbar spenningskilde. Denne spenningskilden skal være styrt av en mikrokontroller.

For å styre spenningen som går fra mikrokontrolleren til transformatoren skal det brukes en ultralydsensor. En ultralydsensor måler avstand. Bitverdien sensoren sender til mikrokontrolleren vil være høyere jo lengere unna et objekt befinner seg innenfor sensorens rekkevidde. Mikrokontrolleren vil bli programmert slik at bitverdier fra ultralydsensoren påvirker signalet til transformatoren. Denne funksjonen vil være programmert slik at ved bestemte bitverdier fra sensoren, vil et bestemt antall lys bli aktivert. Funksjonen vil være slik at jo nærmere man beveger hånden foran sensoren, jo flere lys vil bli aktivert.



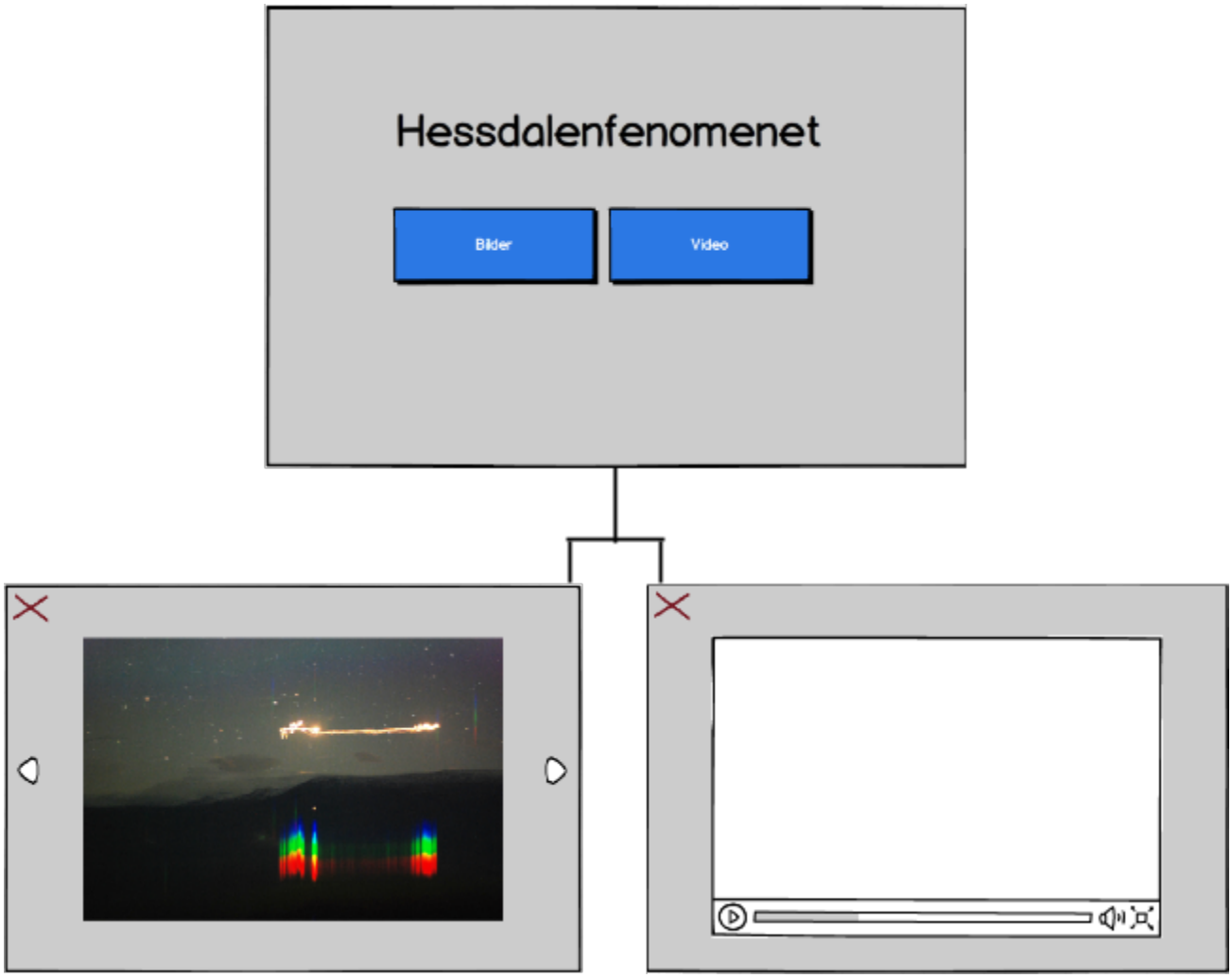
Figur 3.1: Skisse av Lysboks

3.4.2.2 Presentasjonsprogram

I menyen skal det være to knapper kalt for “Bilder” og “Video”. Knappene skal ta brukeren til et nytt vindu som viser et slideshow med bilder eller videoer. Det skal være en bakgrunnsanimasjon som går kontinuerlig.

Slideshowene skal ha en form for navigasjon ved hjelp av enkle frem og tilbake knapper. Dersom et av slideshowene er aktivert, skal programmet automatisk bla gjennom bilder og videoer. Programmet skal vises samtidig både på touchskjermen og tv-skjermen, slik at besøkende kan se bildene i større format. Arbeidsgiver skal lett kunne legge til nye bilder og video. Programmet skal være umulig for besøkende å lukke, slik at de ikke skal få tilgang til skrivebordet.

Skisser:



Figur 3.2: Skisse av Presentasjonsprogrammet

3.4.2.3 Spill

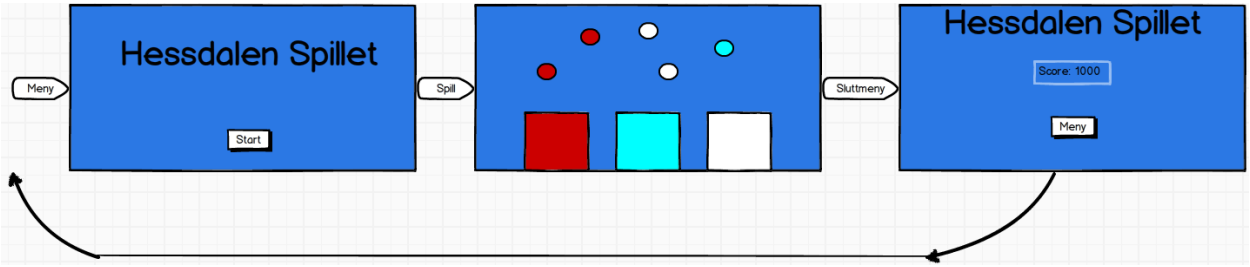
Hovedmenyen skal bestå av et bakgrunnsbilde med spilltittel og en startknapp som starter spillet. I spillvinduet skal det være et bakgrunnsbilde og fem lysobjekter med tilfeldig bevegelse. Lysobjektene vil ved berøring stoppe å bevege seg, og kan styres rundt på skjermen. Det skal også være tre bokser som brukes til å samle lysobjektene. Boksene og lysobjektene vil ha tre forskjellige farger; rød, blå og hvit.

Når man plasserer et lysobjekt med tilsvarende farge som boksen, vil lysobjektet bli fanget og omformet til en poengsum. Boksene vil gi forskjellige poengsummer i forhold til hvor vanskelig tilsvarende lysobjekt er å fange. Oppe i høyre hjørnet av skjermen vises den totale nåværende

poengsummen. Det er satt at spillet avsluttes etter 30 sekunder. Dette vises kontinuerlig med en nedteller oppe i det venstre hjørne av skjermen.

Når tiden har gått ut vil du komme til en avslutningsmeny som vil vise hvor stor poengsum du klarte å oppnå. I avslutnings- menyen skal det være et bakgrunnsbilde med spilltittel. Det vil også være en knapp som tar deg tilbake til hovedmenyen.

Skisser:



Figur 3.3: Skisse av Spill

Kapittel 4
Implementasjon, produksjon og gjennomføring av produkter.

4.1 Hardware

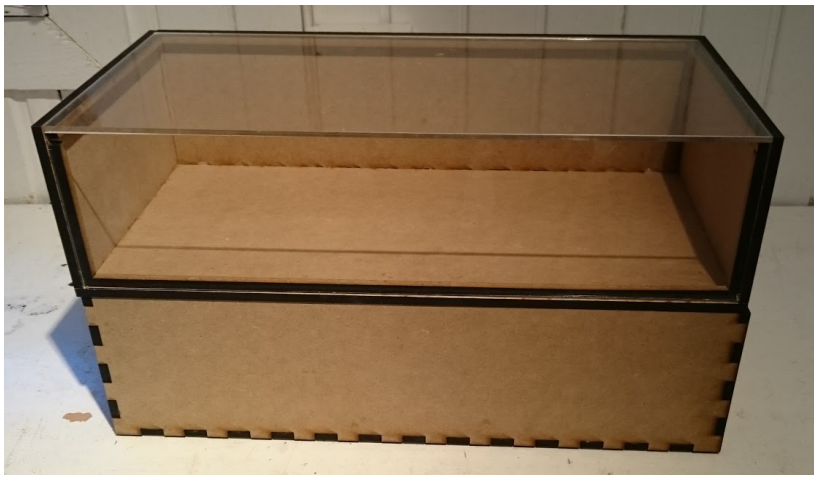
Til produksjonen av våre produkter benyttet vi en del hardware. For at fremtidig arbeid skal bli lettere, dokumenterte vi i dette underkapittelet hvilket hardware som ble brukt. I Lysboksens tilfelle går vi detaljert inn på hvordan den er koblet.

4.1.1 Lysboks

Vi skal her fortelle om gjennomføringen mot lysboksen. Dette produktet kom ikke til et protostype-stadiet, med forbeholdt løsningen i kapittel 3.4.2.1, som vi forteller om i kapittel 5.1.1 og 5.1.2. Det vi fikk produsert var en fungerende løsning i forhold til å få gassampullene til å lyse. Vi fikk også gjort klar en boks vi skulle bruke til prototypen.

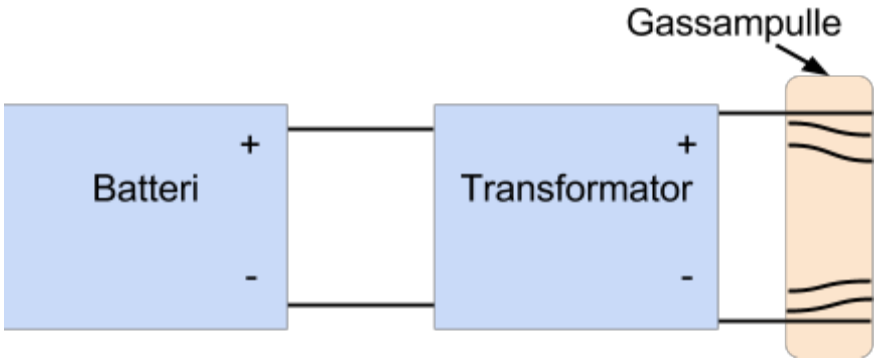
4.1.1.1 Produksjon av boks

Produksjonen av utstillingsboksen startet med å estimere hvor stor den skulle være. Vi kom fram til disse målene hovedsakelig forbeholdt at det skulle være plass til en god del komponenter. Utstillingsboksen er delt i to, der den ene delen skal bestå av komponenter, og den andre delen viser frem lysene. Boksene er like store, 40x20x10cm. Alle veggene er 6mm tykke og laget av MDF, med unntak av den øverste boksen, der to av veggene består av 3mm tykk akryl. Dette ble lagd med Inspirias ressurser, og kuttet av oppdragsgiver ved bruk av laserkutter.



Figur 4.1: Bilde av utstillingsboks

4.1.1.2 Transformator med batteri



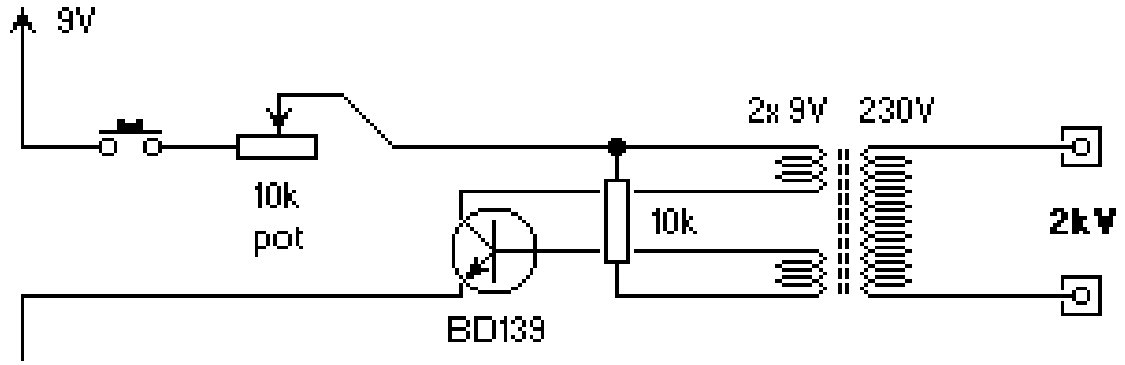
Figur 4.2: Skisse av transformator med batteri

Ved utforming av idéen vår, forutså vi en del utfordringer. Den første utfordringen er å kunne skape en krets med en gassampulle. Problemet er å finne komponenter som gir stor nok spenning til systemet. Det er viktig å ikke få for stor spenning, slik at det dannes en lysbue utenfor gassampullen, da kretsen ikke vil gå gjennom gassen.

Forhandleren angående gassampullene anbefalte en gitt spenning og hertz. Dette var gitt ved 2000 Volt med 35KHz. Å finne tilsvarende komponenter som kan fungere med et batteri og en mikro-kontroller var vår første utfordring. Vi fant fort en veiledning på hvordan vi kunne løse problemet[2]. Dette ble løst med en transformator.

En transformator brukes til å konvertere lav spenning til høy spenning. Vi bruker en transformator der inngangen er anbefalt mellom 5 og 9 Volt, der utgangen er estimert til 2000 Volt med 50-60 Hz. Vi fulgte veiledningen videre hvordan dette skulle kobles sammen med en transistor og en motstand på 10K.

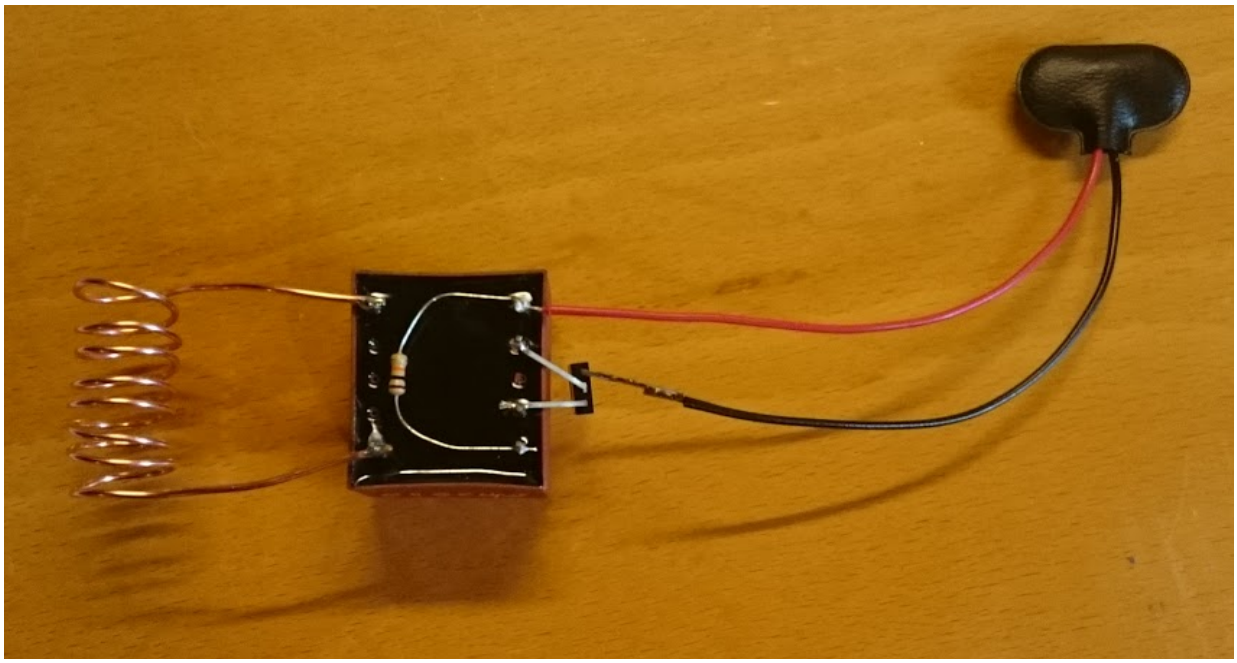
Kretsskjema:



Figur 4.3: Kretsskjema transformatorkobling

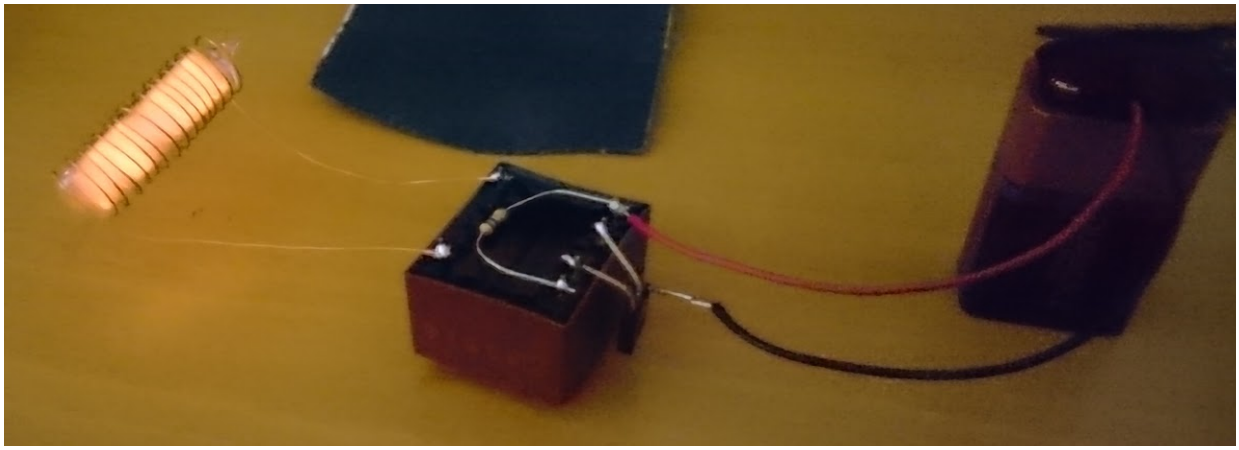
Ferdig kobling med gassampulle:

Dette er den første løsningen av systemet. Rød ledning er positiv(+) og svart ledning er negativ(-). I denne løsningen brukes det en kobbertråd som er 1,2mm i diameter.



Figur 4.4: Løsning 1; Transformator med batteri

Dette er den endelige løsningen på det første problemet. Forskjellen fra første løsning er diameteren på kobbertråden, som her er 0,5mm. Gassampullen i viklingen inneholder neongass. Det som skjer er at vi kobler til et 9V batteri. Dette starter prosessen i transformatoren. Vi ser at får nok spenning fra transformatoren slik at gassampullen begynner å lyse.



Figur 4.5: Løsning 2; Transformator med batteri

Her stopper produksjonen av lysboksen og prosessen går over til en testfase.

4.1.2 Spill og Presentasjonsprogram

Touchskjerm:

Programmene ble utviklet spesielt for Inspirias touchskjermer. Skjermen har en oppløsning på 1024x768 piksler(4:3 format), noe vi tar hensyn til videre i produksjonen. Skjermen vi har tilgjengelig er “1537L 15-inch Open-Frame Touchmonitor” fra Elo Touch Solutions[3].

Datamaskin:

Dette er datamaskinens spesifikasjoner som står vi tar i bruk under produksjonen for testing, og er en del av Hessdalenrommet i dag.

Merke: Dell Optiplex 990
Operativsystem: Windows 7 64Bit.
Prosessor : Intel Core i5-2500 3.30 GHz
Ram: 4 GB

4.2 Software

Vi har produsert to prototyper, et spill og presentasjonsprogram. Prototypene er laget etter retningslinjene gitt i kapittel 3.4. Her skal vi snakke om hvordan programmene er laget, og hvilke programmer som ble brukt til produksjon. Vi deler opp produksjonen i to deler; grafisk design og programmering.

4.2.1 Hessdalenspillet

Den første prototypen vi skal forklare produksjonen til, er Hessdalenspillet. Det er mye informasjon som kan være vanskelig å forstå uten en grunnleggende kompetanse for programmering. Produksjonen av det grafiske designet stiller også krav til god forståelse av programmene valgt til å utvikle denne delen av produktet.

4.2.1.1 Grafisk design

Før vi kan begynne å produsere det grafiske designet til spillet, må passende verktøy velges. Vi bruker Adobe sine designprogrammer, som har noe av de beste komersielle programmene i dag. Programmer som blir brukt er Illustrator og Photoshop.

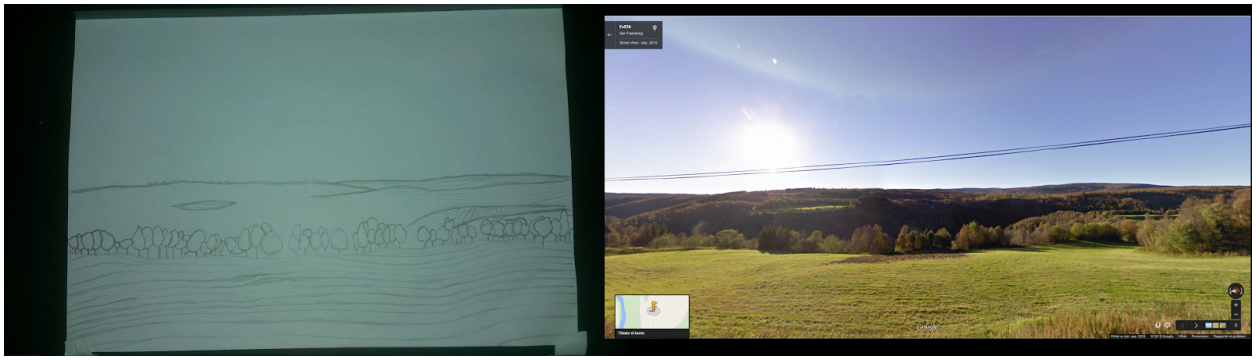
Illustrator er et program som er basert på vektorgrafikk. Uavhengig av hvilket størrelse- format du tegner vektorgrafikk i, vil den beholde den samme kvaliteten. Dette gjør den utmerket til å tegne logoer eller tegning av linjer. Photoshop er et godt verktøy for tegning, redigere farger og legge til effekter som for eksempel 3D eller lys.

Disse programmene kan lastes ned gratis med full tilgang i 30 dager[4].

Bakgrunnsbilde

Bakgrunnsbilde er kombinert med et bilde funnet fra Google Maps i Hessdalen, og et bilde fra kunsteren Sam Chivers som har gjort sitt eget kunstverk av Hessdalen og fenomenet[5].

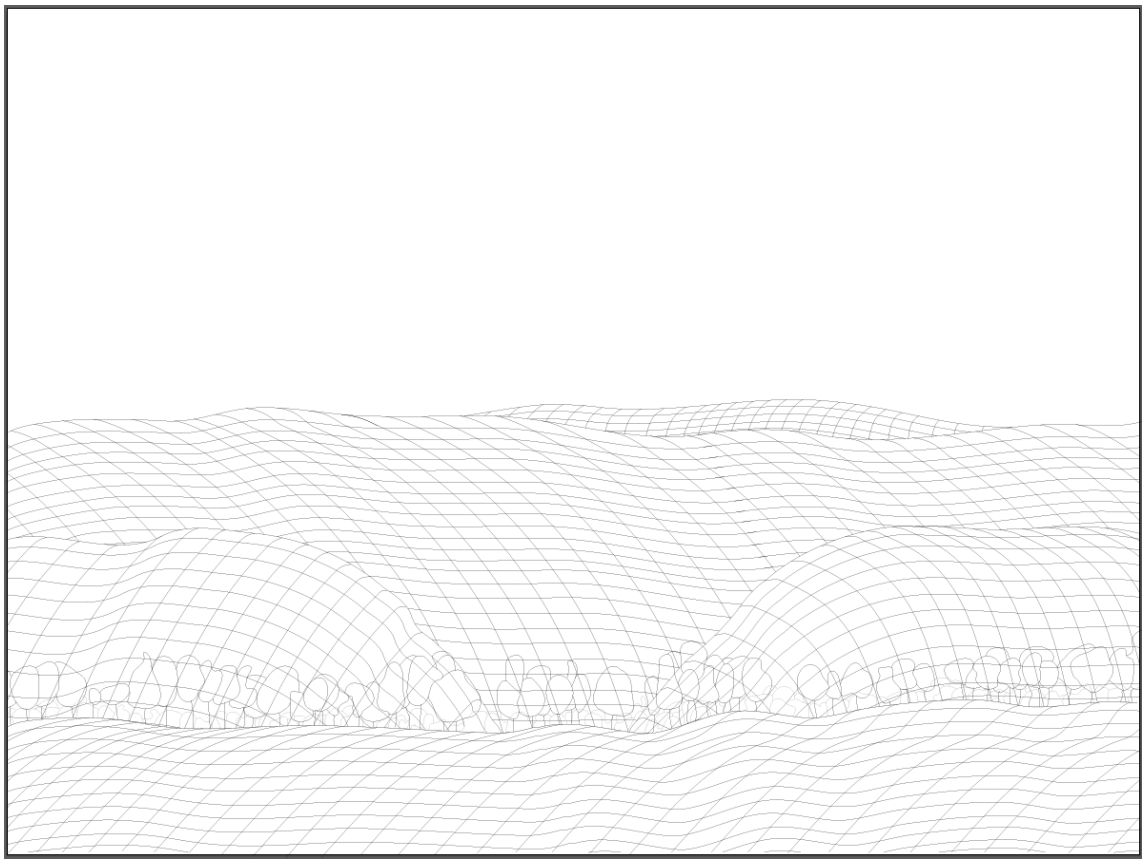
For å lage bakgrunnsbilde fant vi først et utsnitt av Hessdalen i Google Maps. For å få det mest mulig likt tar vi dataskjermen og legger flatt ned med et A4 ark over, og tegner en grov skisse.



Figur 4.6: Skisse og inspirasjon, bakgrunnsbilde Spill

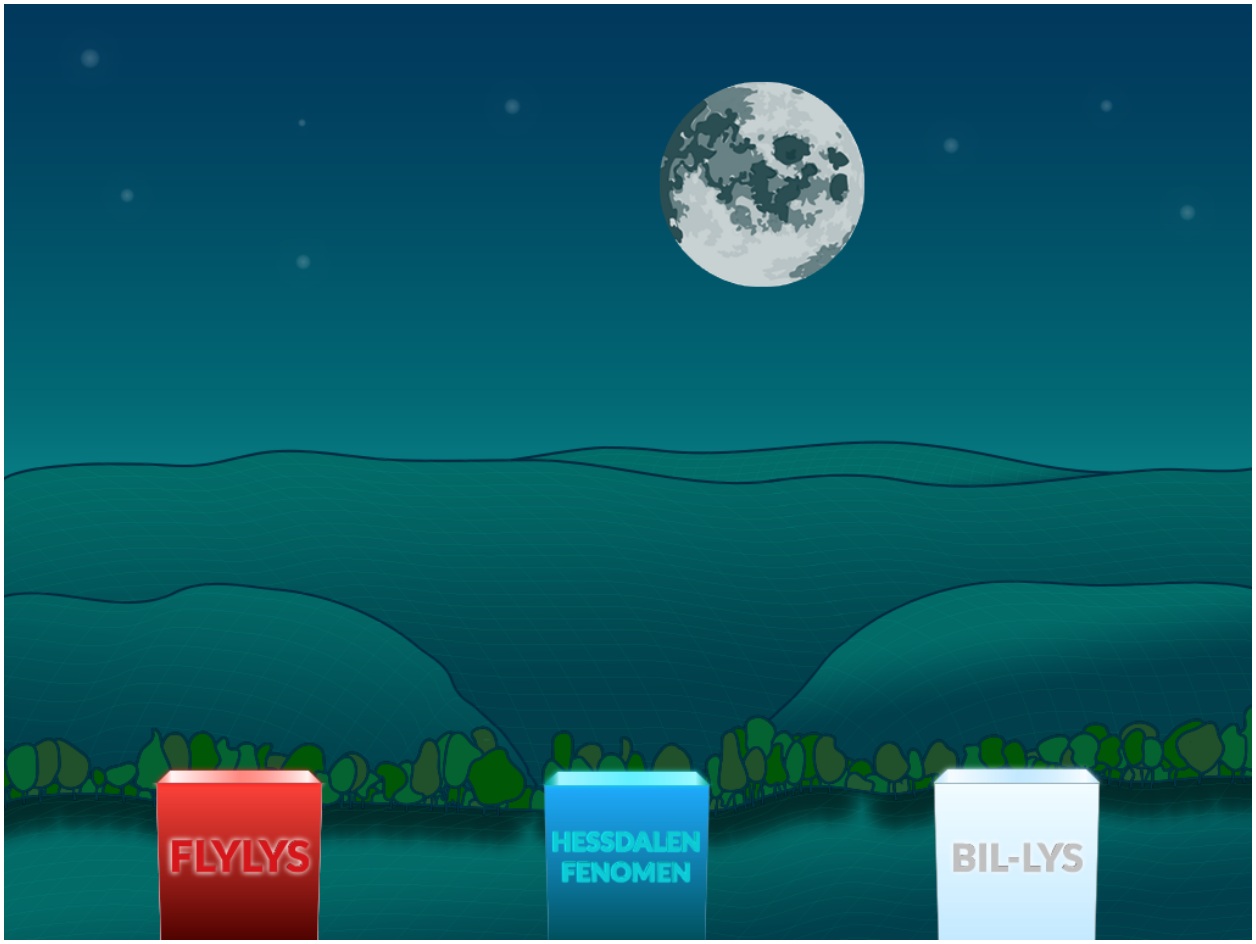
Deretter tar vi bilde av tegningen med mobiltelefonen og legger det inn i Adobe Illustrator. Vi tegner så over slik at vi får en skisse i vektorgrafikk. Dette fordi det er enkelt å jobbe med, og ser veldig bra ut uansett størrelse.

For å gjøre bildet mer levende, fyller vi inn skissen og legger på en 3D effekt for å få dybde i bildet. Samtidig bruker vi lignende metoder kunsteren Sam Chivers gjorde med sitt bilde.



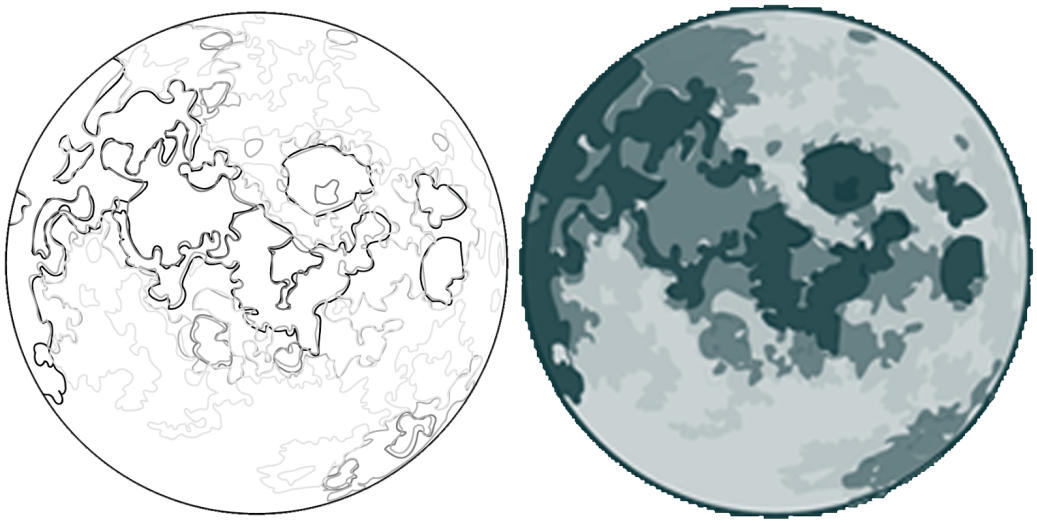
Figur 4.7: Vektorgrafikk, Spill

Vektorgrafikken legger vi inn i Adobe Photoshop. Der fyller vi inn resten av bildet med farger og legger på effekter. Målet er å få fram en inntrykk av at det er natt, men samtidig bruke lyse farger slik at man ser detaljer godt. Lyse omgivelser egner seg best i spill etter egen erfaring.



Figur 4.8: Bakgrunnsbilde, Spill

For å skape mer dybde i bilde legger vi til en himmel fylt med noen stjerner og en måne. Månen blir laget på samme måte som landskapet. Vi tegner månen ved å bruke en steg for steg metode og gjør designet til vårt eget.



Figur 4.9: Tegning av måne, Spill

Knapper

Knappene “Spill” og “Meny” lager vi i Illustrator. Vi lager to størrelser for hver knapp, slik at vi får en 3D effekt når de blir brukt. Dette blir gjort med funksjon i koden som forklares senere. For at funksjonen skal fungere legger vi til knappene i Photoshop, der den største legges til venstre i bilde og den minste rett til høyre for den.

3D effekten på knappene sin form blir generert med en effekt kalt Bevel and Emboss i Illustrator.

Figur 4.10: Knappdesign, Spill

Lys

I *PlayState*-delen finnes det fem lys som skal fanges i bokser. Det er to røde, to hvite og ett blått lys. Alle har forskjellige størrelser. Vi lager et lys i Photoshop, og endrer fargene i etterkant. Målet er å lage lysobjekter med en skinnende effekt.

Figur 4.11: Lysobjekter, Spill

Bokser

Det er tre bokser som hører til *PlayState*-delen av spillet, der lysene blir fanget. Disse tre boksene blir generert i Illustrator på samme måte som knappene. Vi lager en åpning i boksen. For å få effekten av at lysene går inn i boksen må vi skille åpningen og selve boksen fra hverandre. Dette blir gjort i Photoshop.

Figur 4.12: Bokser, Spill

Ovenfor ser vi hvordan boksen er delt. Åpningen i boksen ligger sammen med bakgrunnsbildet, mens boksene med tekst er sitt eget bilde som ligger ytterst/øverst slik at alle elementer i spillet kommer bak boksene.

Hver boks har også hver sin animasjon. Animasjonen er et bilde som "fader" inn og ut på under et halvt sekund. Det gjør at det ser ut som en blitz som kommer ut av boksen når du fanger et lys, med poengsum. Hver blitz har fargenyanser lik som boksen den kommer fra.

Den øverste delen av bilde er det som synes i spillet. Den delen som ikke synes blir ikke tatt vekk grunnet hvordan effekter fungerer på objekter i Photoshop. Selve blitzen er satt sammen av flere *brushes* som vi fant på nettet[6].

Figur 4.13: Animasjon, Spill

4.2.1.2 Programmering

Før vi kan begynne med produksjonen på programmeringsdelen må vi først velge programmerings-språk og utviklingsplatform. For mer detaljert beskrivelse av alle deler av koden - se de vedlagte filene. Åpne Hessdalenspillet-mappen og deretter åpne source-mappen. Her finner man alle klassene. Det er anbefalt og ha generell kodeforståelse i dette kapittelet.

Vi valgte HaxeFlixel som er et kryss-plattform programeringsspråk for spillprogramering[7]. Det har sine røtter i fra AS3 Flixel Framework, men istede for å bruke AS3 bruker det Haxe og Open-FL for å lett kunne kompilere til mange opperativsystemer. Vi bruker HaxeFlixel for å gi muligheten til å bruke spillet på mer en bare Windows touch-skjermer men også mobil og tablets. En full guide til installering finnes på HaxeFlixel sine hjemmesider[8].
For å kunne bruke Flixel-addons i vårt program må vi legge til <*haxelib name="flixel-addons"* /> i vår project.xml fil.

Sammen med HaxeFlixel bruker vi Nape[9]. Nape er en 2D fysikk motor som kan gi våre sprites en fysisk kropp. En fysikk-motor gjør at elementer i spillet kan reagere med hverandre, og sette parametere som for eksempel masse og tyngdekraft. I vårt tilfelle hjelper dette med å få lysene til å oppføre seg slik vi ønsker.
For å kunne bruke Nape i vart program må vi legge til <*haxelib name="nape"* /> i vår project.xml fil.

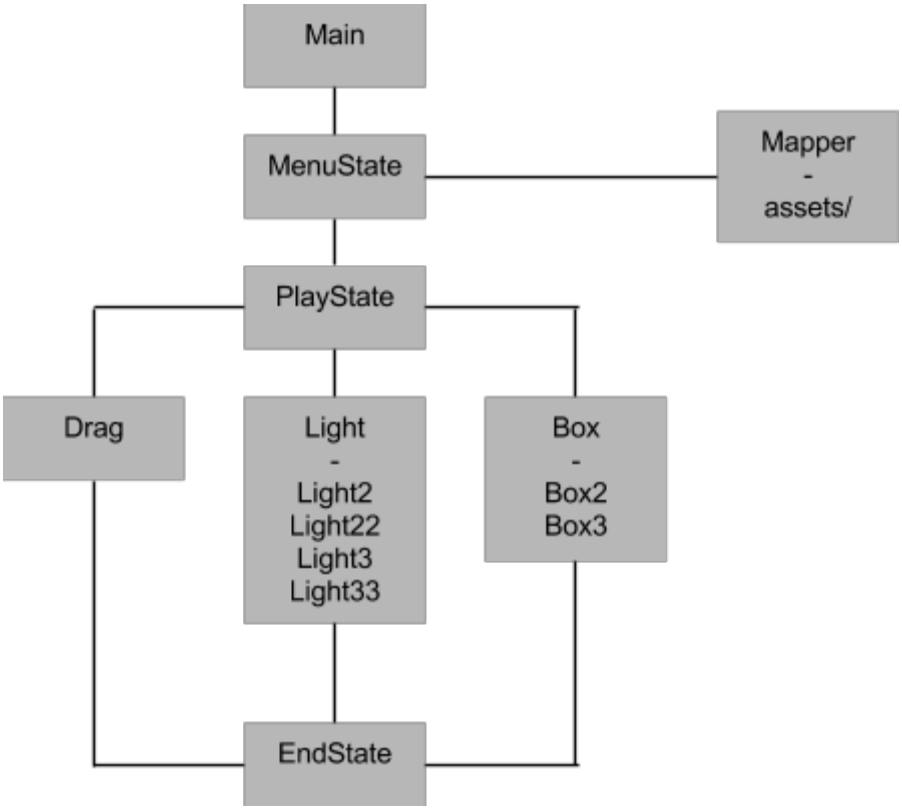
Utviklingsmiljøet vi bruker er FlashDevelop[10]. Det er hovedsakelig laget for flash-programmering, men er også den anbefalte platformen for programmering med HaxeFlixel. Dette er fordi HaxeFlixel og AS3 er veldig like i syntaks.

Et veldig nyttig verktøy vi bruker under produksjonen er Adobe Flash debugger[12]. En debugger hjelper til å finne feil i koden. Dette er den eneste debuggeren til FlashDevelop vi fant og ga oss en stor fordel.

For å få Adobe Flash debugger med FlashDevelop til å fungere, gjorde vi disse stegene:
Først laster man ned debuggeren. Gå til startmenyen - Default Programs - Associate a file type or protocol with a program. Gå ned til .swf - trykk Choose Program og velg Adobe Flash debugger-programmet som ble lastet ned tidligere.

Nå er alt klart til utvikling, kompilering og debugging av programmet. Videre i kapittelet forteller vi om programmets struktur og funksjonalitet. Vi går lag for lag å forklarer alle klassene og hva de gjør.

Klasseoversikt:



Figur 4.14: Klasseoversikt, Spill

MenuState: Startskjerm
PlayState: Det er her selve spillet blir laget.
Drag: Denne klassen gjør slik at vi kan dra lysa rundt på skjermen.
Light: Her former vi lysas utseende og adferd.
Box: Lager noen enkle bokser.
EndState: Sluttskjerm med poengsum.

Standard-funksjoner
Det er tre funksjoner som kommer til å bli referert til senere, så for å unngå repetisjon forklares alle funksjonene her.

create(): Håndterer skaping av sprites og objekter i spillrommet.

add(object): funksjonen for å legge til spesifike objekter.

update(): Oppdaterer kode som ligger i funksjonen kontinuerlig.

destroy(): Rydder opp objekter.

Main:
Alle program starter med en main klasse. Vi bruker en standard main klasse vi fant på HaxeFlixel sin demo-side[12].

Vi legger til et variabel for skjermstørrelse; 1024x768,
tar bort synligheten til musepekeren, setter *FlxSplash*-klassen sin *nextState*-objekt til *MenuState*.

MenuState:
MenuState er vår hovedmeny, den første skjermen man ser når spillet starter.
Her legger vi til en bakgrunn og en knapp.

#1 Bakgrunn
Bakgrunnen lagres ved bruk av et *FlxSprite* objekt. *FlxSprite* tar *SimpleGraphic* som variabel. Det betyr at vi kan hente filer rett fra vår *assets* mappe ved å bruke en *string* som inneholder en mappe sti, eksempel;
***"assets/Background_menuback.png"*. Objektet lages i *create()*.**

#2 Knapp

Knappen genereres ved hjelp av *FlxButton*-klassen. Funksjonaliteten *FlxButton* gir er mulighet til å sette størrelse på knappen og kalle på en funksjon.
I funksjonen knappen kaller på, blir klassen *FlxG* benyttet. *FlxG* gir oss muligheten til å *"fade"* ut *MenuState* ved bruk av *camera*-objektet sin *fade()*-funksjon, samt bytte *state* til *Playstate* via *switchState()*-funksjonen.

For å gi knappen en animasjon og et bilde brukes *FlxButton* sin *loadGraphic()*-funksjon.
loadGraphic() tar et *string*-variabel for å laste inn bilder fra en filsti. Hvis *loadGraphic()* sitt *Animated*-variabel er satt til *true* kan vi gi knappen en animasjon. Ved å sette to knapper ved siden av hverandre inne i det bildet vi laster inn, vil funksjonen velge venstre bilde hvis fingeren ikke er på knappen, og høyre bilde hvis fingeren er det. Se figur 4.10

PlayState:

Playstate er vårt spillvindu. Det er her man spiller spillet.

Funksjonalitet som må håndteres:

- Bakgrunn, forgrunn og mittgrunn
- Vegger
- Bokser
- Lys
- Mushåndtering
- Kollisjon mellom lys og boks
- Poeng telling
- Nedteller
- Animasjoner
- Lyd

#3 Bakgrunn, forgrunn og midtgrunn

Det er to bilder som må legges til. Bakgrunnsbilde og bilde til boksene.
For å passe på at bildene legger seg i riktig posisjon over hverandre, lager vi *FlxGroups*. En gruppe for forgrunn og en gruppe for bakgrunn. Vi legger da bildene inn i den passende gruppen i *create()*-funksjonen.
Animasjonene legges i mellom disse to lagene. Deretter lager vi enda en *FlxGroup* som legges til mellom bakgrunn og forgrunn.

#4 Vegger

Vegger legges til ved bruk av *createWalls()*-funksjonen. Vi må ha vegger slik at lysene ikke forsvinner utenfor spillrommet.

#5 Bokser

Vi må ha tre bokser. En for hver farge. Vi lager vår egen funksjon for generering av bokser. I funksjonen lager vi tre objekter - en for hver *Box*-klasse. Vi gir boksene en posisjon i rommet og deretter bruker *add(object)*-funksjonen som legger de til i rommet. Denne funksjon kalles på i *create()*.

#6 Lys

Vi skal ha fem lys; to røde, to hvite og en blå. Disse lysene må vi ha mulighet til å dra rundt på skjermen, dette bruker vi *Drag*-klassen til. For å legge til lysene lager vi én funksjon per lys. En slik funksjon må i denne rekkefølgen;

1. Lage et nytt *Light* objekt,
2. Bruke *add()* for å legge *Light*-objektet inn i rommet
3. Lage et nytt *Drag*-objekt slik at vi kan bruke *MouseEventManager*-en der
4. Bruke *add()* for å legge *Drag*-objektet inn i rommet
5. Legge *Light* objektet inn i *Drag* sin *registerPhysSprite()*-funksjon

Lys-funksjonene legges til to steder. I *create()* og i de forskjellige *collideLightBox*-funksjonene som blir forklart i #8.1.

#7 Mushåndtering

Alt av mushåndtering får vi fra *Drag*-klassen ved bruk av *registerPhysSprite()*-funksjonen.

#8 Kollisjon mellom lys og boks

Det er fem lys vi ønsker å håndtere.

- To røde lys skal kollidere med boks1
- Det blå lyset skal kollidere med boks2
- To hvite lys skal kollidere med boks3

Dette håndteres på samme måte for alle lys og bokser så vi går gjennom metoden kun en gang. I *create()* må vi legge til en *listener*. Dette gjør vi ved hjelp av klassen *FlxNapeState*. *FlxNapeState* har et variabel *space* og dette variablet har en listener vi kan bruke. I *listener* sin *add()*-funksjon må vi

bruke en *InteractionListener*. Denne *InteractionListener*-en må starte en CbEvent og vente på en *InteractionType.COLLITION* mellom et lysobjekt og et boksobjekt.

For at *InteractionListener* skal kunne merke kollisjonen mellom lys og boks må lys- og boksklassene sine *body* 's bruke et *CbType* objekt.

#8.1 collideLightBox(callback:InteractionCallback)-funksjonen

Hvis kollisjon skjer vil *InteractionListener* kjøre en funksjon. Funksjonen tar et *callback:InteractionCallback*-objekt som argument. Vi må ha dette objektet for å kunne bruke funksjonen i en *InteractionListener*.
Først lager vi et *Light*-objekt. Dette objektet *cast*-er en *callback*. *callback*-en peker på *Light*-klassen sin *body.userData.sprite. userData* lagrer *Light* sin status i *sprite* variablet. Vi må passe på at *sprite* variablet er unikt per klasse.

Deretter dreper vi *Light*-objektet ved bruk av *kill()*-funksjonen. Når lyset blir drept må vi også fjerne *mouseJoint* og *destinationJoint* til *Drag*- og *Light*-klassene. Vi gjør dette ved å kalle på *dest()*-funksjonene til de to klassene. Mer om disse funksjonene i #14 og #15.

Poeng legges til når *collideLightBox()* kjører. Vi gjør dette ved å legge til et tall i variablet *point* ved å si *point += 500*. Når dette er gjort må vi oppdatere poengteksten på skjermen ved bruk av *updateText*-funksjonen vi har laget. Se #9.

En animasjon-funksjon skal kjøres når *collideLightBox()* kjører, vi forteller om funksjonen i #11.

Til slutt lager vi et nytt lys slik at vi alltid er fem lys på skjermen. Dette gjør vi ved bruk av funksjonene vi laget i #6.

#9 Poengtelling

Vi forteller allerede litt om Poengtelling i #8.1, men det vi også må gjøre er å sende *point* variablet sin verdi til skjermen. Vi gjør dette ved bruk av *FlxText*. *FlxText* gir oss muligheten til å lage tekstobjekter på skjermen. Vi gir objektet en posisjon, størrelse og en ny skrifttype. For å kunne benytte seg av andre skrifttyper må man laste ned *.ttf*-filen til skrifttypen man ønsker og deretter plassere filen i *assets*-mappen. Vi kan da legge filstien til skrifttypen i *FlxText*-objektet (se eksemplet i #1).

#10 Nedteller

Vi må legge til en nedteller slik at spillere har begrenset tid til å fang lysene. Vi lager først et variabel som er lik tiden vi ønsker. Deretter bruker vi *FlxG.elapsed* til å telle *timer*-variablet baklengs. Vi må deretter skjekke om *timer* er mindre en null i en *if-statement*. Hvis *timer*-variablet er mindre enn null "fader" *PlayState* over til *EndState*. Alt dette må gjøres i update().
Vi må også sende tekst til skjermen slik at spilleren vet hvor mye tid de har igjen. Dette gjør vi på akkurat samme måte som i #9. Det er bare en ting som er forskjellig å det er at vi kvitter oss med alle *float* desimalene. Dette gjør vi med *FlxMath* sin *roundDesimal*-funksjon.

#11 Animasjoner

Animasjoner må vi ha for å gi en god tilbakemelding når spilleren får poeng. Vi lager tre funksjoner som tar argumentet *x* og *y*. *x* og *y* er variabler som sier hvor i rommet animasjonen skal spilles. Vi trenger tre forskjellige funksjoner siden vi ønsker unike animasjoner per boks.
Funksjonene inneholder; Et nytt *FlxNapeSprite*-objekt, en *fadeIn*-funksjon og en lyd som spilles av ved bruk av *FlxG.sound.play()*-funksjonen.
I *FlxNapeSprite* objektet laster vi inn bildet som skal animeres. Vi får det til å animere ved bruken av *fadeIn*-funksjonen. *fadeIn*-funksjonen tar en *callback*-funksjon som argument som vil spilles av når *fade*In er ferdig. Det er slik vi får *fadeIn* og *fadeOut* til å gå sømløst.
Til slutt må vi legge animasjonen inn i rommet. Vi bruker en *FlxGroup* slik at animasjonen ligger i riktig lag, se #3.

#12 Lyd

Vi bruker fire lyder i programmet. En lyd for hver animasjons-funksjon[13] (se #11) og en sang som spilles av når spillet kjører[14]. De tre animasjonslydene laget vi selv, mens sangen fant vi gratis på nett.

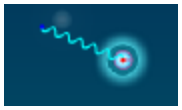
Light:

Vi har fem lys-klasser. Disse klassene er nesten identiske bortsett fra noe endringer på adferd.
Koden som har blitt brukt til å få adferd på lysa har vi funnet på HaxeFlixel sine demosider[15].

Vi endrer noen variabler for å få den adferden vi ønsker.

#13 Adferd

For å få adferd på lysene bruker vi *DistanceJoint*-klassen. Denne klassen fungerer som en "fjær" hvor den ene enden er plassert på en gitt plass i spillrommet og den andre enden er festet til lyset. Dette kan man lett se i debuggeren:



Figur 4.15: Lysobjekt adferd, Spill

DistanceJoint-klassen har en rekke funksjonaliteter som kan benyttes for å få adferd på lyset. En annen del av adferden er variablet *destinationTimer*. Dette variablet brukes som en teller. Denne telleren vil flytte ankerpungtet til lyset til en tilfeldig plass etter en tilfeldig tid ved bruk av *FlxRandom*-klassen. Vi passer på å sette rammene til den tilfeldige posisjoneringen slik at lysene aldri treffer boksene.

#14 dest()-funksjonen

Denne funksjonen er veldig viktig for kollisjonen mellom lys og boks. Funksjonen brukes slik at *destinationJoint*-objektet slipper *Light*-objektet når det treffer boksen. Uten denne funksjonen vil man alltid få en feilmelding. Funksjonen må ligge etter *update()* slik at vi alltid vet om *destinationJoint* er *null* eller ikke.

Box:

Her lager vi tre bokser. En boks med hver sin farge. Boksene bruker *FlxNapeSprite* for å få en fysikk-kropp. Bokser lager vi ved bruk av *makeGraphic()*- og *createRectangularBody()*-funksjonen. I *makeGraphic()* setter vi boksene til å være gjennomsiktig siden vi bruker et bilde som overlegg, men det er også en mulighet å gi boksene hvert sitt bilde ved bruk av *loadGraphic*-funksjonen.

Drag:

I *Drag* trenger vi fire funksjoner. Vi trenger først en *DistanceJoint*-funksjon som kan lage et anker mellom musa og lyset. Vi kaller dette objektet *mouseJoint*. For å vite om musa er trykket ned eller ikke, trenger vi en *MouseEventManager*-funksjon. Denne funksjonen tar et *FlxNapeSprite*-objekt som argument. Videre i *update()* trenger vi en *if-test* for å lage ankerpunktet mellom musa og et *sprite* for å skjekke om musa har sluppet.

#15 dest()-funksjonen

Denne funksjonen er veldig viktig for kollisjonen mellom lys og boks. Funksjonen gjør slik at variable *mouseJoint* slipper *Light*-objektet når det treffer boksen. Uten denne funksjonen vil man alltid få en feilmelding. Funksjonen må ligger under *update()* slik at vi alltid vet om *mouseJoint* er *null* eller ikke.

4.2.2 Presentasjonsprogram

Den andre prototypen vi skal forklare produksjonen til, er Presentasjonsprogrammet. Det er anbefalt å ha generell kodeforståelse og kjennskap til Visual Studio. Produksjonen av det grafiske designet stiller også krav til god forståelse av programmene valgt til å utvikle denne delen av produktet, slik som i Hessdalenspillet.

4.2.2.1 Grafisk design

Bakgrunn

Bakgrunnsbilde til presentasjonsprogrammet er basert på et bilde av en fjellvegg funnet på nettet[16]. Bildet redigerte vi slik at man får en følelse av dybde og undring, det kan på flere måter minne om en stjernehimmel. Animasjonen er basert på fenomenet i seg selv med mange små lys i tilfeldige bevegelser og farger i samspill med bakgrunnsbildet.

For å lage bakgrunnen følger vi en veledning for hvordan lage en god introduksjonsvideo i After Effects[17][18]. Vi bruker den samme teknikken, men med andre parametere som passer våre premisser. Vi skaper først et bakgrunnsbilde med Photoshop der vi bruker bildet av fjellveggen. Her får vi teksten til å gå mer i ett med bildet. Deretter implementerer bildet vi lagde til After Effects, der de siste endringer av bildet blir gjort.

Figur 4.16: Designoversikt bakgrunnsbilde, Presentasjonsprogram

Etter at bakgrunnsbilde er implementert legger vi til animasjon. Den består av mange små lysobjekter som flyter oppover langs sidene av bildet. Fargene på lysobjektet er balansert med bakgrunnsbildet, og endrer farge når de har beveget seg halvveis over bilde. Vi fulgte en egen veiledning for animasjonen, men endrer parameterene etter våre premisser. Forskjellen fra veiledningen er at vi bruker to animasjoner i stedet for én. Det legger også til en “blur”-effekt for å få mer dybde i selve bakgrunnen.

Figur 4.17: Design bakgrunnsanimasjon, Presentasjonsprogram

Knapper

Vi hadde fem forskjellige knapper som hadde hver sin oppgave i programmet. To av de er “Bilder” og “Video” som tar deg videre til en fremvisning av bilder eller video fra Hessdalen. De to knappene har lik bakgrunn. Både bakgrunnen og tekst/tegn i knappene lages med samme teknikk som teksten i bakgrunnsbildet. Deretter er det lagt på effekter. Vi krymper de respektive knappene og legger på en effekt bak knappen, for at det skal se ut som den går inn i bakgrunnsbilde.

Figur 4.18: Design menyknapper, Presentasjonsprogram

Vi lager to navigasjonspiler som peker til venstre og høyre. Disse brukes for å bla mellom bilder eller video når det vises.

Figur 4.19: Pekere, Presentasjonsprogram

Til slutt lager vi en knapp som avsluttet fremvisningen og navigerer deg tilbake til hovedmenyen.

Figur 4.20: Exitknapp, Presentasjonsprogram

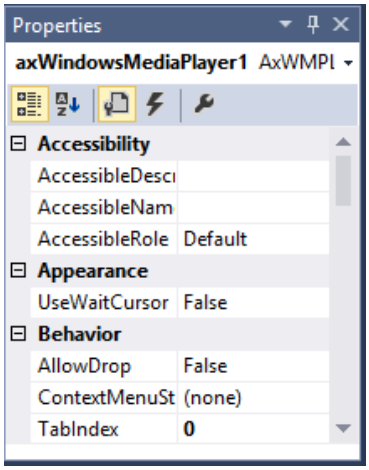
4.2.2.2 Programmering

Før vi kan begynne med produksjonen på programmeringsdelen må vi følge samme fremgangsmåte som i 4.2.1.2 - velge programmeringsspråk og utviklingsplatform. For mer detaljert beskrivelse av alle deler av koden - se de vedlagte filene. Åpne Presentasjonsprogram-mappen og deretter åpne TouchInterfaceBilderogVideo-mappen . Her finner man alle klassene. Det er anbefalt og ha generell kodeforståelse i dette kapittelet.

For vårt presentasjonsprogram valgte vi å bruke C# .NET framework 4.5. Dette er et språk vi har brukt mye tidligere. Ved å bruke C# kan vi benytte oss av Visual Studio sin “form generator” noe som gjør utviklingen enklere.

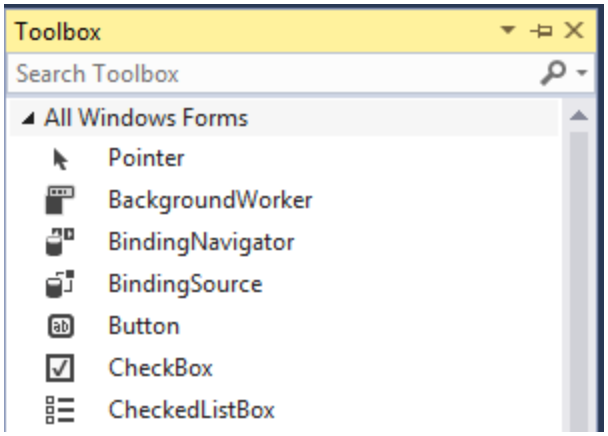
Vi bruker Visual Studio Express 2013 som utviklingsmiljø. Dette er et gratisprogram som kan lastes ned på Microsoft sine hjemmesider[19].

Mye koding kan unngås i Visual Studio fordi dette programmet genererer en rekke elementer selv.
I tillegg til dette kan man endre alt av elementenes verdier via *Properties*-vinduet.



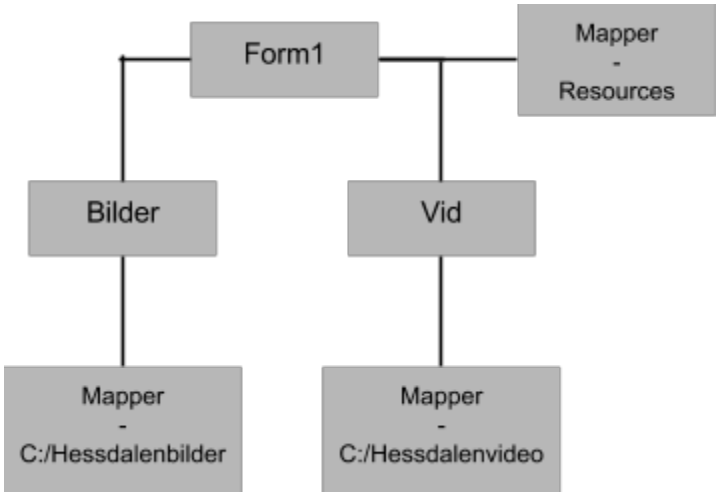
Figur 4.21: Properties, Presentasjonsprogram

Properties-vinduet kommer til å bli referert videre i kapittelet. En annen ting vi kommer til å referere til er Visual Studio sin *Toolbox*. *Toolbox* er verktøyet man bruker for å legge til elementer i *Windows Forms.*



Figur 4.22: Toolbox, Presentasjonsprogram

Klasseoversikt:



Figur 4.23: Klasseoversikt, Presentasjonsprogram

Form1: Meny-klassen

Bilder: Slideshow-klassen

Vid: Videofremvisnings-klassen

Mapper:

- Resources er hvor alle “layout” bilder og video befinner seg.
- Hessdalenbilder og Hessdalenvideo er mapper som må lages i C: disk.

Form1:

Denne klassen er vår meny klasse. Følgende funksjonalitet skal legges til:

- Fjerne standard WinForm rammer og sette full-skjerm
- Mediaspiller-bakgrunn
- “Timer” til mediaspilleren
- Knapper
- Knapp-animasjoner
- Event håndtering

#1 Fjerning av rammer og full-skjerm

For at brukere ikke skal kunne lukke programmet ved bruk av touchskjermen må vi fjerne rammen til WinForms. Ved å gjøre dette blir vi kvitt den standard *exit*-knappen. For å gjøre dette bruker vi *FormBorderStyle*-klassen sitt *none*-variabel.
For at programmet skal gå til full-skjerm brukes *FormWindowState*-klassen sitt *Maximized*-variabel.

#2 Windows Media Player(WMP)

WMP skal brukes til å vise vår bakgrunn. Dette er det beste verktøyet siden vi bruker en video som bakgrunn.

For å kunne bruke WMP må vi først legge den til i *Toolbox*. Dette gjøres slik:
Tools - Choose Toolbox items - COM components - merk av Windows Media Player - OK.
Nå kan vi søke opp WMP i *Toolbox*.
Etter å ha lagt til WMP i *Form1*-vinduet går vi til *Properties*. Der strekker vi spilleren slik at den dekker hele vinduet, skrur av *User interface* og laster inn bakgrunnsvideoen fra vår *Resources*-mappe.

#3 WMP Timer

Når vi først kjørte programmet fant vi ut at videoen "blinker" hver gang den spilles på nytt. For å fikse dette bruker vi en *timer*. Timeren teller til videoens slutt, det vil si at timeren må være like lang som bakgrunnsvideoen. Når *timeren* er ferdig forteller vi WMP at den skal hoppe tilbake til et halvt sekund inn i videoen. På denne måten får vi en sømløs overgang hver gang bakgrunns- -videoen ender.

#4 Knapper

Knapper legges til via *Toolbox*. I *Properties* legger vi til bilder fra *Resource*-mappa, setter størrelse og for å få knappene adaptive i forhold til skjermstørrelse, setter vi ankerpunktet på knappene til *bottom*. Det knappene gjør når de blir trykket på er å åpne sitt respektive *Form* og deretter gjemme begge menyknappene i *Form1* slik at de ikke er synlig når de andre vinduene er åpne.

#5 Knapp-animasjoner

Alle knappene vi bruker har de samme metodene for knapp-animasjon. Vi lager først flere *MouseEventHandlers* for hver knapp. Vi trenger tre av disse per knapp. En for *MouseUp, MouseDown* og *MouseMove. MouseUp og MouseDown* resetter knapp-bildet til det orginale bildet, mens *MouseDown* gir knappen et nytt bilde som gir brukeren en respons. Sammen med

MouseEventHandeler-ene trenger vi metoder, siden *MouseEventHandler* tar metoder som argument. Det metodene gjør, er å bytte bildet. Disse bildene blir hentet fra *Resource* mappen.

#6 Event håndtering

Eventer er noe vi bruker i *Bilder* og *Vid* -klassene slik at vi får tilgang til *Form1* elementer utenfor *Form1*-klassen. *Form1* lytter etter event-aktivering i *Bilder* eller *Vid. Eventene* i *Bilder* eller *Vid* aktiveres når den respektive *Exit*-knappen bli trykket på. *Form1* vil da kjøre *event*-metoden.
Det *event*-metodene gjør er å vise bilder- og videoknappene.

Bilder:

Bilder-*formet* er hvor vi laster inn bilder i et slideshow. I dette *formet* må vi legge til følgende funksjonalitet:

- Gjennomsiktig bakgrunn
- Bildefremviser
- Knapper
- Knapp-animasjoner (se #5)
- Fade-in

#7 Gjennomsiktig bakgrunn

Vi ønsker en gjennomsiktig bakgrunn slik at vi kan se bakgrunnsvideoen til *Form1* når vi kjører *Bilder-formet*. Det er ingen standard innstilling for dette i C# .NET så dette må kodes på en litt spesiell måte. Vi må sette *Bilder* sin *TransparencyKey* til en farge som skjeldent blir brukt, i dette tilfelle bruker vi turkis. Hvis vi gjør dette i tillegg til å sette *Bilder* sin *BackColor* til turkis ender vi opp med en gjennomsiktig bakgrunn.

#8 Bilderfremviser

For å vise bilder bruker vi en *pictureBox*. Dette er etstandard verktøy for å vise bilder i *WindowsForms*.

#9 Innlasting av filer

Før vi laster inn bilder i *pictureBox*-en må vi finne ut hvor vi skal laste inn bilder fra, og på hvilken måte. For å gjøre lett å legge til nye bilder i fremtiden lager vi en mappe vi kaller “Hessdalenbilder”.

"Hessdalenbilder"-mappen legges direkte i *C:* -disk. På denne måten kan man bruke dette programmet på alle Windows maskiner siden Windows sitt standard disk-navn er C. Vi ender da opp med denne filstien "C:\Hessdalenbilder\".

Nå som vi har en standard mappe som fungerer på alle maskiner må legger vi til en løsning som henter filnavnene til filene i mappen. Vi bruker en *string-array* til dette. I denne arrayen må det gjøres flere ting; laste inn kun filnavnene på filene i mappa - ikke hele filstien og vi må sjekke hvilken filtype filene har. Gjør det slik:

```
string[] bilder = Directory.EnumerateFiles(@"C:\Hessdalenbilder\", "*.*", SearchOption.AllDirectories)
        .Where(s => s.EndsWith(".png") || s.EndsWith(".jpg") || s.EndsWith(".jpeg") || s.EndsWith(".gif"))
        .Select(path => Path.GetFileName(path))
        .ToArray();
```

Som vi ser er det kun mulig å laste inn: PNG, JPG, JPEG og GIF filer.

Vi må gjøre alt dette siden vi skal bruke frem og tilbake knapper i *Bilder* formet.

Til slutt laster vi inn det første bildet i *Bilder_load*-metoden. Vi bruker *pictureBox* sitt *Image* objekt til å gjøre dette. Eksempel:

```
pictureBox1.Image = Image.FromFile(@"C:\Hessdalenbilder\" + bilder[i]);
```

Vi legger *bilder-arrayen* etter vår standard filsti slik at det ikke har noe å si hvilket bilde som ligger først i mappen. Variablet *i* = 0 i dette tilfellet.

#10 Knapper

Nå som vi har en god måte å laste inn og lese bilder på kan vi legge til fram- og tilbakeknapper. Det disse knappene gjør er å bruke arrayen vi lagde i #8.1 til å gå til neste eller forrige bilde. Det fram-knappen gjør er å legge 1 til variablet *i*. Dette gjør at vi kan gå til neste bilde i arrayen. For at bildefremviseren ikke skal stoppe når vi kommer til det siste bildet i arrayen trenger vi en *if-test*. *if-testen* skjekker om arrayen sin lengde er lik variablet *i*. Hvis dette er tilfellet, vil *if-testen* sette variablet *i = 0*. Det samme gjøres i tilbake-knappen, bare motsatt.

Exit knappen sin funksjon er å lukke *Bilder*-formet og deretter starte eventen vi snakket om i #6.

#11 Fade-in

Vi ønsker at *Bilder*-formet skal "fade" inn når det åpnes. Dette gjøres ved å sette *Bilder* sin *opacity* veldig lavt(0.1) i *Bilder-load*-metoden. Deretter brukes en timer til å gradvis sette *opacity* nermere 1.

Video:

Video-formet er hvor vi viser fram videoene. Dette *formet* bruker mye av den samme funksjonaliteten til *Bilder-formet*. For å slippe repetisjon kommer vi til å referere en del dit.

Funksjonalitet som må legges til:

- Gjennomsiktig bakgrunn (se #7)
- Windows Media Player
- Knapper
- Knapp-animasjoner (se #5)
- Hide-button timer
- Vis knappene når skjermen trykkes

#12 Windows Media Player(WMP)

Det første vi gjør er å legge til en mediaspiller. Denne har samme *Properties* som mediaspilleren i *Form1(#2).* Etter vi har lagt til spilleren må vi legge til en metode å laste inn filer på. Vi gjør dette på nesten samme måte som i #8.1. Forskjellene er at vi kaller mappen "Hessdalenvideo", ser etter andre filtyper i arrayen og bruker WMP sitt *URL*-variabel til å laste inn filer istedet.

Arrayen:

```
string[] VidPath = Directory.EnumerateFiles(@"C:\Hessdalenvideo\", "*.*", SearchOption.AllDirectories)
        .Where(s => s.EndsWith(".avi") || s.EndsWith(".mpeg") || s.EndsWith(".wmv") || s.EndsWith(".mp4"))
        .Select(path => Path.GetFileName(path))
        .ToArray();
```

Vi ser over at det kun kan lastes inn: AVI, MPEG, WMV og MP4 filer.

Innlastingsmetoden:

```
axWindowsMediaPlayer1.URL = @"c:\Hessdalenvideo\" + VidPath[i];
```

#13 Knapper

Knappene i *Vid* gjør akkurat det samme som *Bilder* knappene(#9). Den eneste forskjellen er at knappene bruker den nye arrayen og innlastningsmetoden.

#14 Hide-button timer

Vi bruker en *timer* til å gjemme frem-, tilbake- og exit-knappene. Vi vil at besøkende skal få den beste video-opplevelsen. *Timeren* teller til ti før den gjemmer knappene.

#15 Vis knappene når skjermen trykkes

I samsvar med *timeren* over(#13), lager vi en metode som viser knappene igjen når brukeren trykker på skjermen. Dette gjør vi via WMP sin *MouseDownEvent*-metode. Denne gir oss muligheten til å kjøre kode når WMP blir trykket på. I tillegg til å vise knappene må vi passe på å restarte timeren i #13.

Kapittel 5
Testing og Evaluering

5.1 Testing av gruppa

Det ble gjort en god del testing gjennom alle tre prosjektene før vi kom fram til et produkt eller en konklusjon. Vi gjorde mye testing internt i gruppen for å finne nye eller bedre løsninger på problemene som dukket opp underveis. Det ble også gjort testing med andre personer utenfor prosjektet, for å høre hva folk syntes om produktet.

5.1.1 Lysboks med batteri

For oss var en av de store utfordringene å få gassampullene til å lyse. Det ble gjort flere tester med forskjellige teknikker. Vi prøvde først med en step-up som ga opptil 400KV. Før vi testet den på gassampullene, testet vi den mot seg selv ved å lage en lysbue. Det viste seg fort at lysbuen var veldig stor, noe som gjorde at vi konkluderte med at det ikke kunne brukes til gassampullene. Dette ble også relativt farlig i forhold til å ha i en utstilling med tanke på at andre kunne skade seg. Step-up-en ble også varm og derfor egnet seg ikke.

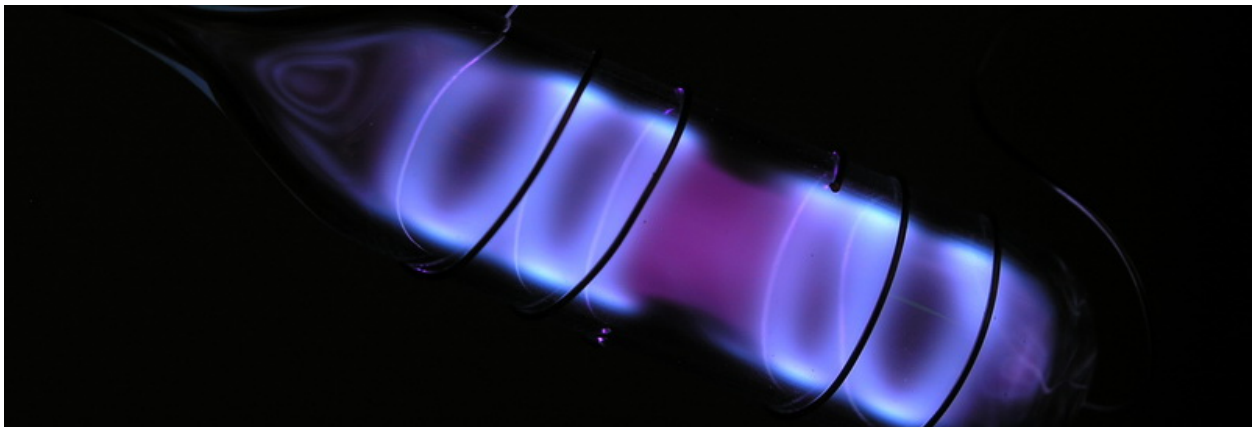
Vi forsket videre og fant en transformator med anbefalt styrke på spenningen - 2000 Volt og med 50-60Hz. Først fikk vi den ikke til å fungere, men det viste seg at testene ikke var godt nok gjennomtenkt. Vi kjørte nye tester der vi ga systemet mer tid i mørkere omgivelser. Dette medførte at vi fikk fire av fem ampuller til å fungere slik vi håpet, der Krypton ikke fungerte. Eneste ulempen var at det tok veldig lang tid før ampullene begynte å lyse. Allerede da kunne vi ta konklusjonen om at prosjektet for lysboksen ikke var mulig å gjennomføre med tanke på løsningen vi hadde sett for oss, og tiden som var igjen.

5.1.1.1 Oversikt over gassene vi har testet

Alle gassampullene inneholder en edelgass(gruppe 18) uten farge, lukt eller smak. De har hver sin unike farge i form av plasma[20].

Xenon:

Xenon presentert i gassampulle ved høy spenning:



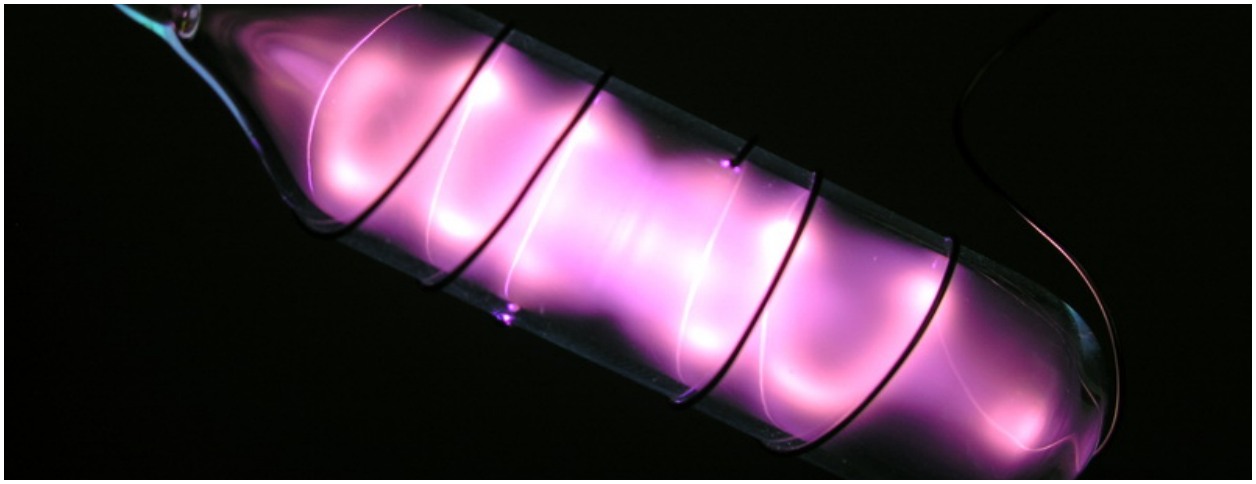
Figur 5.1: Xenon i gassampulle ved høy spenning [22]

Emisjonslinjer for xenon:

Figur 5.2: Xenon, emisjonslinjer [21]

Helium:

Helium presentert i gassampulle ved høy spenning:



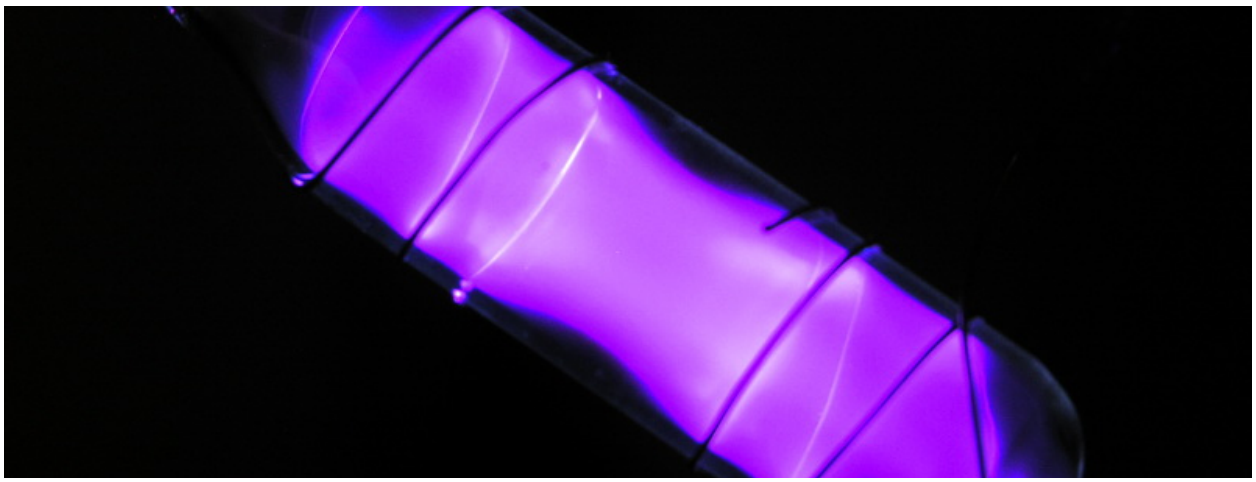
Figur 5.3: Helium i gassampulle ved høy spenning [22]

Emmisjonslinjer for helium:

Figur 5.4: Helium, emisjonslinjer[21]

Argon:

Argon presentert i gassampulle ved høy spenning:



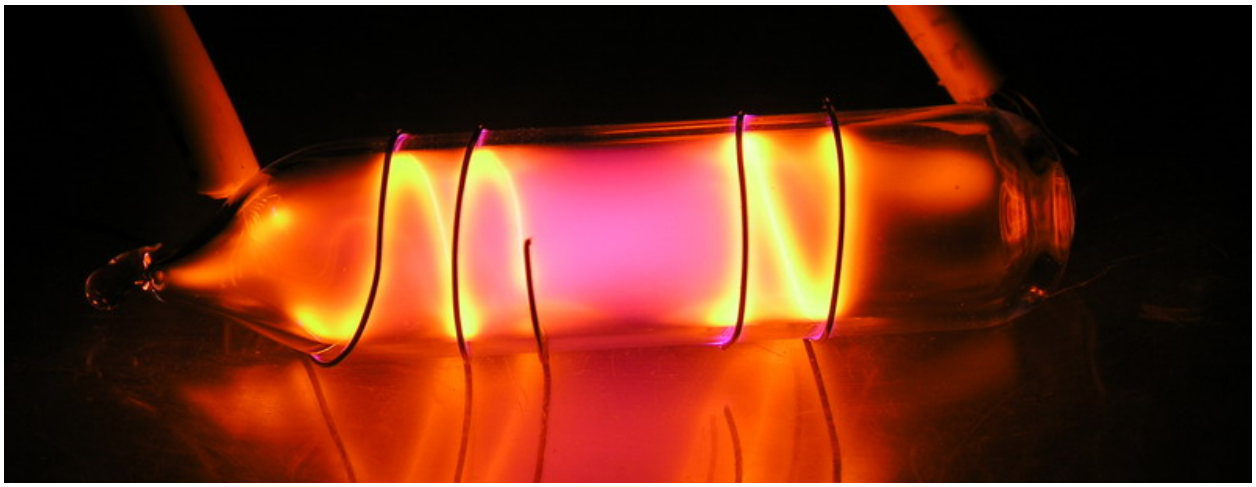
Figur 5.5: Argon i gassampulle ved høy spenning [22]

Emmisjonslinjer for argon:

Figur 5.6: Argon, emisjonslinjer [21]

Neon:

Neon presentert i gassampulle ved høy spenning:



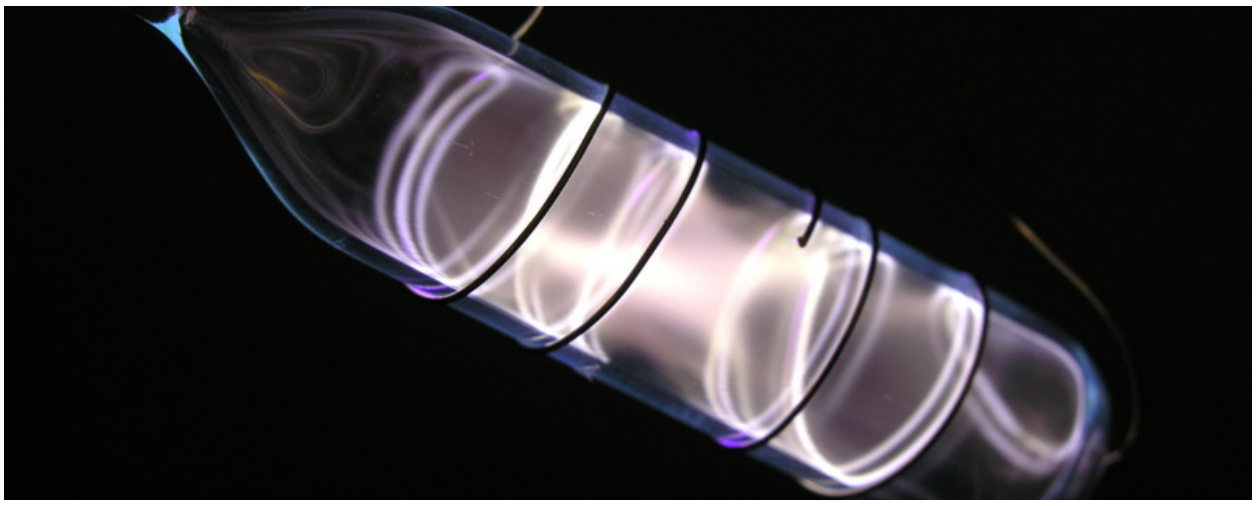
Figur 5.7: Neon i gassampulle ved høy spenning [22]

Emmisjonslinjer for neon:

Figur 5.8: Neon, emissjonslinjer [21]

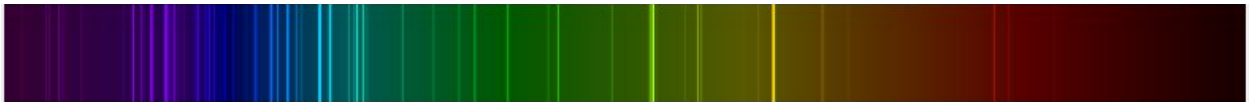
Krypton

Krypton presentert i gassampulle ved høy spenning:



Figur 5.9: Krypton i gassampulle ved høy spenning [22]

Emmisjonslinjer for krypton:



Figur 5.10: Krypton, emissjonslinjer [21]

5.1.2 Presentasjonsprogram

Under produksjonen av presentasjonsprogrammet fungerte de fleste funksjonene ganske raskt. Det var en god del testing for å optimalisere selve designet og få flyt i programmet. Vi hadde problemer med å få bakgrunnsanimasjonen til å kjøre flytende. Mye av testene ble gjort i forhold til design, og det var der den meste av tiden gikk bort.

5.1.3 Spill

Spillproduksjonen var det som krevde mest testing og forskning, siden dette var et helt nytt språk for oss begge. Det som var det største gjennom produksjonen var å få lysene til å forsvinne når de traff boksen. Problemet var at når man holdt et lys, fikk man en feilmelding når det skulle bli drept. Dette var fordi vi benyttet to ankerpunkt - et for mushåndtering og et for lysadferd. Vi stod helt fast i lang tid, og benyttet en “hack” for unngå feilmeldingen, slik at vi kunne arbeide videre. Samtidig la vi ut problemet på internett, for å få hjelp av andre. Etter en tid fikk et forslag om hvordan vi kunne fikse dette problemet. Vi tok forslaget til utvikling, og til slutt løste problemet. Dette var kritisk i forhold til å kunne levere en prototype.

Det er en kjent bug i spillet. Denne bug-en skjer ved å trykke hurtig på et av lysene flere ganger. Dette fører til at lysene setter seg fast. Vi føler ikke at dette er kritisk, siden dette er unaturlig å gjøre og vanskelig å få til.

5.2 Brukertesting av Spill

Vi gjorde en brukertest på Hessdalenspillet med 5 personer fra Inspiria. Vi presenterte spillet som en del av utstillingen gjennom en av touchskjermene deres. Det ble en uformell brukertest der vi fikk direkte tilbakemeldinger under testen. Vi kom fram til at det måtte bli tydeligere hva som skulle gjøres og hvor mange poeng hvert enkelt lys gir. De syntes også at spillet var morsomt, og skulle gjerne hatt en topp-poengsum funksjon der man legger ved sine initialer.

Figur 5.11: Utbedring bakgrunnsbilde, Spill

Kapittel 6
Diskusjon

6.1 Prosessen

I starten av prosjektet fikk vi tidlig en idé vi følte at tilfredsstilte kravene til oppgaven. Før vi fikk grundig utforsket denne idéen, skrev vi forprosjektsrapporten. Dette var en stor feil fra vår side. Først og fremst var ikke idéen gjennomførbar med tanke på budsjettet. Vi konkluderte da at vi måtte endre idéen.

En annen stor feil fra vår side, var at vi ikke tok en analyse av rommet tidlig nok. Dette er noe av det første vi skulle gjort. Etter analysen fikk vi mange gode idéer som var gjennomførbare. Ved gjennomgang av analysen med oppdragsgiver fikk vi et klarsignal at vi kunne bruke datamaskinene med touchskjerm til prototyping. Dette gjorde at vi endret hovedmålet i oppgaven. Det nye hovedmålet satt fart på produksjonen siden vi ikke var begrenset av materialer eller penger.

Løsningen vi valgte, innebærte å produsere flere prototyper. Dette ble en utfordring, siden det ble et delt fokus. Det gjorde at vi måtte ha en god struktur på arbeidet for å gjennomføre alle prototypene. Lysboksen ble mindre prioritert av flere grunner. En av de var at vi følte de andre idéene var mer nyttig for Inspiria. Det var også enklere for oss å produsere kvalitetsprodukter gjennom programmering.

Spillet var en stor utfordring. Vi hadde lite kjennskap til spillprogrammering, og måtte sette oss inn i dette, noe som tok lang tid. Språket vi valgte var også ukjent for oss, og det mindre hjelp på nettet enn vi hadde håpet på. Denne utfordringen viste seg å være veldig verdifull, siden det tvang oss til å lære mye på egen hånd.

Kommunikasjonen med både veileder og oppdragsgiver har vært meget god. Veileder har gitt oss klare beskjeder om hva som må gjøres for at vi skulle klare å levere en bacheloroppgave. Vi hadde møter jevnlig, ca. annenhver uke, der vi gjennomgikk prosessen og arbeidet som hadde blitt gjort. Veileder holdt oss på rett spor, og kom med gode innspill hele veien.

Vi har hatt jevnlig kommunikasjon med oppdragsgiver, både via mail og gjennom møter. Hver gang vi har spurt om hjelp eller informasjon, har vi fått svar med en gang, noe som gjorde at vi fikk god flyt i prosessen. Oppdragsgiver sin kompetanse har vært nyttig, og det har kommet gode innspill som vi har hatt stor nytte av.

Gruppen har hatt samme prosjektleder hele veien. Tanken bak dette var at det var mest effektivt, siden vi kun er to personer i gruppen. Dette gjorde at kommunikasjon med oppdragsgiver og veileder gikk uten problemer. Gruppen sin arbeidsstruktur var allerede godt strukturert før prosjektet begynte, fordi vi har jobbet som gruppe de tre siste årene. Vi lærte mye om prosjektarbeid av veileder, som vi vil få stor nytte av i arbeidslivet.

6.2 Forslag til forbedringer av Hessdalenrommet

Først og fremst mener vi det er viktig å sette større fokus på selve fenomenet. Det er for mye fokus på forskning. Vi ville gjerne sett flere bilder og videoer av fenomenet, som er en av grunnene til at vi lagde presentasjonsprogrammet vårt. Det burde også være flere bilder utenfor touchskjermene. Vi syns det er en god idé å bytte galakseplakaten med ett eller flere store bilder av fenomenet.

VLF-måleren kunne vært mye bedre presentert. Det er vanskelig å forstå hvordan den brukes, og forslag til forberdinger kan man også finne i kapittel 3.1.

Informasjonen på touchskjermene bør fjernes, og heller brukes til enten spill eller quiz.

Inngangen til Hessdalenrommet bør bli mer innbydende. Inngangen tilsier ikke at det er utstilling der. Den kan markeres med et stort skilt som lyser, eller sette fotspor i en retning som er naturlig å gå gjennom rommet.

Selv om vi ikke fikk fullført vår Lysboks, syns vi det er et interessant konsept. Vi føler at vi fikk bidratt til en god start til et større prosjekt med fokus på å vise fram plasma gjennom gassampuller.

6.3 Fremtidig arbeid

Selv om ikke alle prosjektene ble ferdig, er vi selv veldig fornøyde med prototypene og forskningen vi fikk gjennomført. Med dette som grunnlag vil vi snakke om hva som kan gjøres videre med de to prototypene for at disse skal bli fullstendige produkter.

6.3.1 Bilder og video program

Siden det er mange skjermer som står tomme, eller som ikke blir aktivt brukt, kunne vi splittet programmet til to deler. Da kunne vi hatt bilder på en touchskjerm, og video på en annen. Videre kunne de bli koblet de opp til hver sin TV-skjerm. Dette gjør at skjermer ikke står tomme, og bidrar til mer interaktivitet.

Det anbefales å gjøre tester på stabilitet over lengre tid, slik at det kan optimaliseres til å bli stabilt nok. Først da ville vi følt oss komfortable å bruke programmet som en fast utstilling.

6.3.2 Spill

En av funksjonene som vi satt fast på ved slutten av prosjektet var å implementere en “hi-score”-fremviser, der man kunne vise frem den beste poengsummen med initialer til spilleren i avslutningsmenyen i spillet. Dette er også noe som ble nevnt i brukertesten i kapittel 5.2

Vi hadde en idé om forskjellige bakgrunner til playState-delen av spillet, der man kunne få opplevd Hessdalen på vinterstid, dagtid og lignende. Det er veldig vanlig i spill å kunne bytte kart, og vi føler at det hadde vært en kul implementasjon vi gjerne skulle ha testet.

Ikke alle er like konkurranseinnstilte når det kommer til spill - noen vil bare prøve spillet og føle at de har mestret noe. Derfor ville vi implementert flere vanskelighetsgrader slik at spillet passer for alle.

Disse funksjonalitetene er basert på alle de timene, dagene og årene vi har spilt selv og hva som finnes av funksjoner i spill i dag. Men på en annen side, så kan dette bli for komplisert i en utstilling. Disse funksjonalitetene vil nok være bedre på en datamaskin, mobil eller tablet. En utstilling skal være enkel og forstå, og det skal ikke ta for lang tid å gjennomføre spillet.

Kapittel 7

Konklusjon

Prosjektets hovedmål var å utvikle flere prototyper ved å kombinere fysikk, elektronikk og teknologi- som skal hjelpe Inspiria med å forbedre Hessdalenrommet. Selv om hovedmålet ikke ble oppnådd, har vi oppnådd målet med hensyn til oppgaven, der oppgaven var å lage spennende aktiviteter knyttet til lys, skape undring rundt Hessdalenfenomenet og øke interaktiviteten i Hessdalenrommet. Vi oppnådde målet ved å lage et spill som omhandler Hessdalenfenomenet. Spillet gir økt interaktivitet til rommet og undring rundt selve fenomenet. Gjennom en brukertest viste det seg at spillet også er underholdende for flere. Presentasjonsprogrammet bidrar med å skape undring rundt Hessdalenfenomenet, siden besøkende nå kan få se flere bilder og videoer av selve fenomenet. Siden besøkende selv kan bestemme hva de vil se på av bilder og video, øker interaktiviten i rommet. Rapporten inneholder en analyse av Hessdalenrommet og diskusjon om hvordan forbedre rommet. Vår forskning på bruk av gassampuller ved høy spenning er grundig dokumentert. Det er også detaljerte beskrivelser av prototypenes design, funksjon og oppbygning.
Vi kan konkludere med at rommet fortsatt har forbedringspotensiale, men at vi har bidratt med forskning og prototyper som øker interaktivteten og undring rundt Hessdalenfenomenet.

Bibliografi:

[1]: Bruk av TV/PC-spill en gjennomsnittsdag.
http://www.medienorge.uib.no/statistikk/medium/ikt/334
[2]: Simple 2000 Volts Transformer - Electrostatic Sticky Notes
http://www.doityourselfgadgets.com/2011/12/simple-2000-volts-transformer.html
[3]: Elo Touch skjerm
http://www.elotouch.com/Products/LCDs/1537L/default.asp
[4]: Adobe software nedlasting
http://www.adobe.com/no/downloads.html
[5]: Hessdalen kunst
http://www.samchivers.com/New-Scientist-Hessdalen-Lights
[6]: Photoshop brushes
http://www.brusheezy.com/brushes/50106-22-rays-of-light-brushes
[7]: HaxeFlixel intro
http://haxeflixel.com/documentation/the-power-of-haxeflixel/
[8]: HaxeFlixel install
http://haxeflixel.com/documentation/part-i-setup/
[9]: Nape
http://napephys.com/
[10]: FlashDevelop
http://www.flashdevelop.org/
[11]: Adobe Flash debugger
https://www.adobe.com/support/flashplayer/downloads.html
[12]: HaxeFlixel demos
http://haxeflixel.com/demos/
[13]: 8 bit sound maker
http://www.bfxr.net/
[14]: Spill sang
http://ericskiff.com/music/
[15]: Haxe demo
https://github.com/HaxeFlixel/flixel-demos/blob/master/Flixel%20Features/FlxNape/source/states/Blob.hx
[16]: Fjellvegg bilde
http://www.textureking.com/dsc_4272/

[17]: Aftereffects tutorial
https://www.youtube.com/watch?v=5rqpdoDv-pI
[18]: Aftereffects tutorial
https://www.youtube.com/watch?v=dq2op6yqvFk
[19]: Visual studio download
https://www.visualstudio.com/en-us/products/visual-studio-express-vs.aspx
[20]: Grunnstoffenes periodesystem
https://snl.no/grunnstoffenes_periodesystem
[21]: Emisjonslinjer
http://astro.u-strasbg.fr/~koppen/discharge/
[22]: Bilder av gassampuller
http://www.chemiestun.de/pse.php