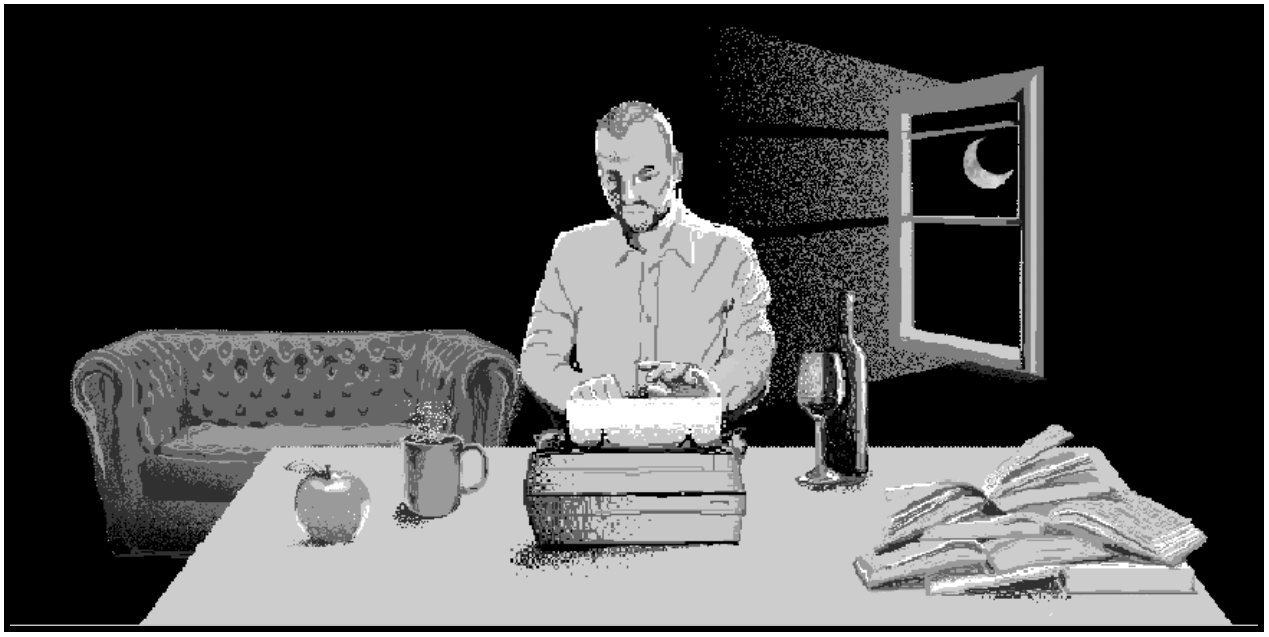
The Writer - simulation of a novelist

A game of needs and inspiration



Et C++ prosjekt av

Kristoffer Johansen

Hallbjørn Storruste

og Stine Marie Aas Grumheden

Innholdsfortegnelse

1.Readme first

1.2 Testet og kjørt på:

Lubuntu, OS X 10.10 og Windows 8.1.

1.3 For å starte:

I mappen “TheWriter” ligger pro’filen “TheWriter.pro”. Det er prosjektfilen. Åpne prosjektet “TheWriter” med QT Creator. Den har kall på alle filene som trengs for å bygge prosjektet. Build, og så kjør programmet. Det trengs ikke å gjøre noen endringer ved import, det holder med “vanlige” innstillinger out-of-the-box for å kjøre spillet.
OBS! På Windows må spillet kjøres fra QT Creator. Mens på Mac og Lubuntu kan det kjøres utenfra.
Under Lubuntu må du kanskje installere OpenGL bibliotek ved å kjøre denne kommandoen; “sudo apt-get install libgl1-mesa-dev”. Om du må installere dette vil det gi feilen “Cannot find -lGL” ved kompilering.

Pro-filen kan åpnes og vil vise alle klasser som trengs for å kompilere.

1.4 Easteregg:

Trykk ti ganger på forfatteren så vil bildet av forfatteren og skrivemaskinen bli byttet ut til noe annet.

2. Innledning

2.1 Om spillet

2.1.1 Målet

Du er en forfatter. Du har en slump penger og skal skrive bøker, som forhåpentligvis selger godt så du har råd til å fortsatt være forfatter. Samtidig som du skal skrive en bestselger må du betjene daglige behov og uventede hendelser. Samtidig må du holde deg inspirert for å skrive en bra bok! For å gi spillet en slutt har du bare 100 dager på deg å skrive en så bra bok som mulig. Man kan også skrive flere bøker.

2.1.2 Hvordan spille

Spillet fungerer som et pek og klikk spill. For å komme igang må du trykke på bokhaugen på Writer sin pult for å sette igang med å skrive en bok. Et pop-up vindu ber deg om å sette navn på boka. Nå er det bare å skrive i vei, men pass på at verdiene ikke blir for lave! Trykk på eplet for å spise og koppen med kaffe for å bli mer våken. Får å få inspirasjon kan du drikke litt vin - men nå blir du bakfull og litt trøtt. Underveis vil tilfeldige hendelser skje. Gjør valg og se hvordan disse påvirker spillet. I panelet nederst på skjermen kan du styre til og se hvordan du ligger ann.

Spillet blir autosavet hver time og ved avslutning. Du kan også loade spill.

2.1.3 Verdier

For å endre ulike verdier for å teste spillet mer:
Hastighet/speed: model/timeController.cpp. Metodene onSpeedPause,onSpeedNormal++ endre variabelen speed.
Antall ord per side: model/book.hpp. wordsPrPage. under private.
Antall sider per bok: model/book.cpp, pageGoal i konstruktørene på book.

2.1.4 Terminologier/uttrykk/klasseforklaringer

Writer: Hovedpersonen i spillet.
Book: Boken som Writer skriver på akkurat nå.
Needs: Behovene til Writer. Er behovene for lave går det saktere å skrive boken og kvaliteten blir dårligere. Inspiration avgjør kvalitet, hunger og sleep hastigheten.
Random event: En hendelse som tilfeldig dukker opp og krever avgjørelse og kan påvirke behovene eller skrivingen
Event: En hendelse man kan regne inn, som husleie, søvn, ferdigstilt bok.
Booster: Noe som påvirker hvordan skrivingen og behovene endrer seg.
EventBooster: Arver booster. Booster som avbryter skrivingen. Kan bli trigget av f.eks. drikke kaffe eller ved en randomevent. Kan kun bruke en om gangen.

EnvironmentBooster: (Ikke brukt men klar til bruk) Arver booster. Boostere som skal være tidløse og bygge seg opp over tid. F.eks. er tanken at rommet skal bli skitnere, været skal påvirke og den typen ting som ikke har en gitt tid men som påvirker.
SingleBooster: Arver booster. En booster som brukes umiddelbart og som ikke distraherer deg fra å skrive boken.
TimeController: Holder orden på tiden og sender ut tidssignaler, hvert sekund og ved "runde" tidspunkt som hver time, dag og uke. Hvert 1000ms så sender den et tick. Hadde dette tallet vært lavere kunne man oppnådd hyppigere oppdateringer av data og smoothere animasjoner. Men vi opplever at 1000ms er helt greit utgangspunkt (og enklere å regne med)
BoosterController: Holder for boosters og sørger for å oppdatere needs, og antall ord henhold til boosters.
StudyView: Den øvre del av skjermen som viser selve spillet og rommet og tar imot klikk som så sendes til boosters eller starter bok.
StatusView: Den nedre del av skjermen som viser status på needs, klokken, bok, og progresjon på boka. Den tar også imot klikkene som endre speed (som blir gjort i speedcontroller)
MainViewManager: Hjertet som mottar ticks fra TimeController og sender beskjed til de ulike klassene om at det er gått en tidsenhet, så gjør det dere må. BoosterController skal oppdatere needs og evt antall sider, StudyViewManager skal animere vær og evt Writer.

2.2 Utregningene

Hunger og sleep avgjør antall ord per minutt.

$$\frac{Hunger + Sleep}{200} * 10 = antall\ sider$$

Inspiration:

$$\frac{Inspirasjon}{100} * 10 = kvalitet$$

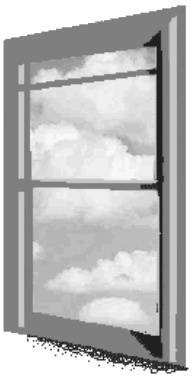
En bok har 250 ord per side og en bok inneholder 350 sider.

Når du har skrevet 350 sider har du skrevet ferdig en bok og får betalt.

$$Antall\ sider * kvalitet = \$\$\$$$

Dette er planen, men er ikke implementert så man får desverre ikke betalt for strevet.

2.3 Vi kan alltid snakke om været



Tidlig i prosjektet snakket vi om at vi ville la den virkelige verden få påvirke den virtuelle verden vi skulle skape. Ved å samle data og behandle denne skulle dette bli representert inne i spillet og påvirke det. Vi kom på at yr.no tilbyr gratis værdata i xml format og bestemte at dette var en god kilde til 'litt big data'

Xml-en vi henter ned er delt inn i tids-bolker på seks timer. 0.00-6.00 6.00 -12.00 og så videre. Vi bruker den samme inndelingen i vårt spill og for hver 6. time i spillet oppdateres været på skjermen

Slik som det er nå så påvirkes ikke spillet av den metrologiske dataen utover det at været blir vist i vinduet til forfatteren. En naturlig utvidelse ville vært å la forfatteren bli direkte påvirket av hvordan været var. Regnet det ute kunne hann bli inspirert til å skrive en skikkelig god krimroman.

2.4 Inspirasjonskilder

Vi har latt oss inspirere av Papers please, The Sims og idle spill som Cookie Clicker.

3. Ansvarsfordeling

Hallbjørn Storruste

- Modeller (Writer,Books)
- Events
- Popups

Abstract Factory

Stine Marie Aas Grumheden

XML-import/eksport
Filbehandling
View
Saving / Loading
Singleton

Kristoffer Johansen

Boostercontroller
TimeController
Knytte View og Modeller
Signals/slots
MVVM
Loading av spillet
Lasting av views

Vi har hatt et hovedfokus på ulike deler av programmet, men alle har vært med på å bidra på hverandres klasser. Dette fordi vi har bygd oss opp litt kompetanse på de ulike feltene og da kan lære av hverandre.

4. Programmet

4.1 Mappestruktur

Vi har forholdt oss til følgende mapper: model, images, utils, view og viewmodel

4.2 Forklaring på mappene

4.2.1 /

Klassene Main, initstart og startup. Disse "drifter" og starter programmet. Main er bare en starter. Initstart er et enkelt GUI som enten lager new game eller laster inn et spill. Startup er "ballongmannen", som blåser opp og holder alle trådene og slipper de når showet er over.

4.2.2 model/

Her er klasser som holder på data og behandler data. De skal i utgangspunktet bare ta imot input og gjøre beregninger før det blir returnert til de klassene som spør. Vi har forsøkt å unngå bruk av QT-klasser i model.

4.2.3 images/

Bilder som brukes i prosjektet
Vi bruker en ressursfil resouces.qrc som er inndelt i ulike prefix. Vi difiransierer imellom /icons, /other, /stud og /wather. /stud inneholder bilder som kunn er tenkt til study view.

4.2.4 utils/

Her er klasser med konstanter, namespaces og verktøy som er nyttige men ikke nødvendigvis en del av programmets hovedstruktur som f.eks. wetherreport, filbehandler og xmlparser.

4.2.4.1 klassen WeatherReport

Vi trengte en klasse for å hente ned og håndtere xml data fra yr.no Qt hadde klassen QDomDocument som vi kunne bruke. QDomDocument representerer et XML dokument og er, konseptuelt, roten i xml-dokument-tre. Vi bruker QNetworkAccessManger for å sende en forespørsel til yr.no. Et objekt, QNetworkReply returneres med xml-en vi ønsker oss. Dataen blir lagret som report.xml. Det er klassen WeatherReport som tar seg av dette. Klassen leser så igjennom dokumentet som har blitt lastet ned og henter ut dataen vi ønsker oss. Den metrologiske dataen kan representere flere værtyper enn det vi har med i vårt program. Derfor bruker vi en swich case og ‘konverterer’ været til en type vær vi har i programmet. For eksempel så blir ‘kraftige sluddbyger og torden’ gjort om til skyet. Det er midlertidig lett å utvide programmet med flere typer vær.

4.2.5 viewmodel/

Evt. viewmodelmanager. Disse klassene kontrollerer views og knytter sammen views og modellene. De fanger opp signaler og sender de videre. Registrerer et view et klikk snappes det opp i manager som sørger for at oppgaven blir utført.

4.2.6 view/

Grafisk representasjon av innholdet fra modellene. Disse skal i utgangspunktet ikke gjøre beregninger, kun vise data. Det er gjort noen unntak, mest for å gjøre ting penere. Men de gjør ikke beregninger for modellene som skal lagres. De mottar manipulasjon fra bruker/spiller som så sender det til viewmodel.

4.2.6.1 Popupvinduene

PopUp fungerer som en base klasse for MessagePopUp, DialogPopUp, InputPopUp og LoadGamePopUp (se diagram). PopUp arver i sin tur QDialog. Obs: Vi har satt et flagg på PopUp, Qt::WA_DeleteOnClose, som gjør at de blir slettet når de blir lukket, selv om de er allokert med new. Hadde man gjort en delete på dem, hadde programmet kræsjet. Dette gjelder da selvfølgelig for alle klasser som arver PopUp. Se 6.3 Abstract factory for mer om hvordan pop-up's er brukt i spillet.

4.2.6.2 klassen StudyView:

Study view har sitt navn siden forfatteren sitter i sin 'study'. Tanker var at forfatteren skulle kunne befinne seg i flere rom eller utendørs og dette view-et kunne da erstattes med et annet.

Study view har ansvaret med å tegne opp alle de ulike objektene i rommet og ta seg av skallering og animering av disse. Study view består av flere induviduelle png tegninger som tegnes på skjermen. Vi har latt oss inspirere av pixelart som lar oss ha svært lavpløselige bilder som skalerer bra med skallerings algoritmen nearest neighbor. Dette er QPainter, som er klassen som tegner bildene, sin default instilling for skaleringen av bilder.

Qt har ingen widget der man kan ha et dynamisk, skalerbart, klikkbart bilde. Derfor laget vi klassen ScaleLabel som arvet QLabel klassen til Qt. Klassen oversktiver QLabel har metoden setImage() som tar imot en filsti for et bilde, og repositionImage() som tar imot skalleringsfaktor x og y posisjon. PaintEvent metoden bruker denne dataen til å sette bilde og størrelse på QLabel-subklassen. Noen av disse bildene skal være klikkbare, men ikke alle, så

klassen ClickLabel arver igjen ScaleLabel. Denne klassen emitter et signal ved en mousePressEvent.

Hver gang en PaintEvent blir kalt i StudyView, for eksempel ved window resize eller ved oppstart, blir nye verdier regnet ut. Denne matematikken kan se litt hårete ut, men fungerer som den skal. Alle ScaleLabels og subklasser av disse, som ligger i stidyview.ui, får en ny posisisjon og størrelse.

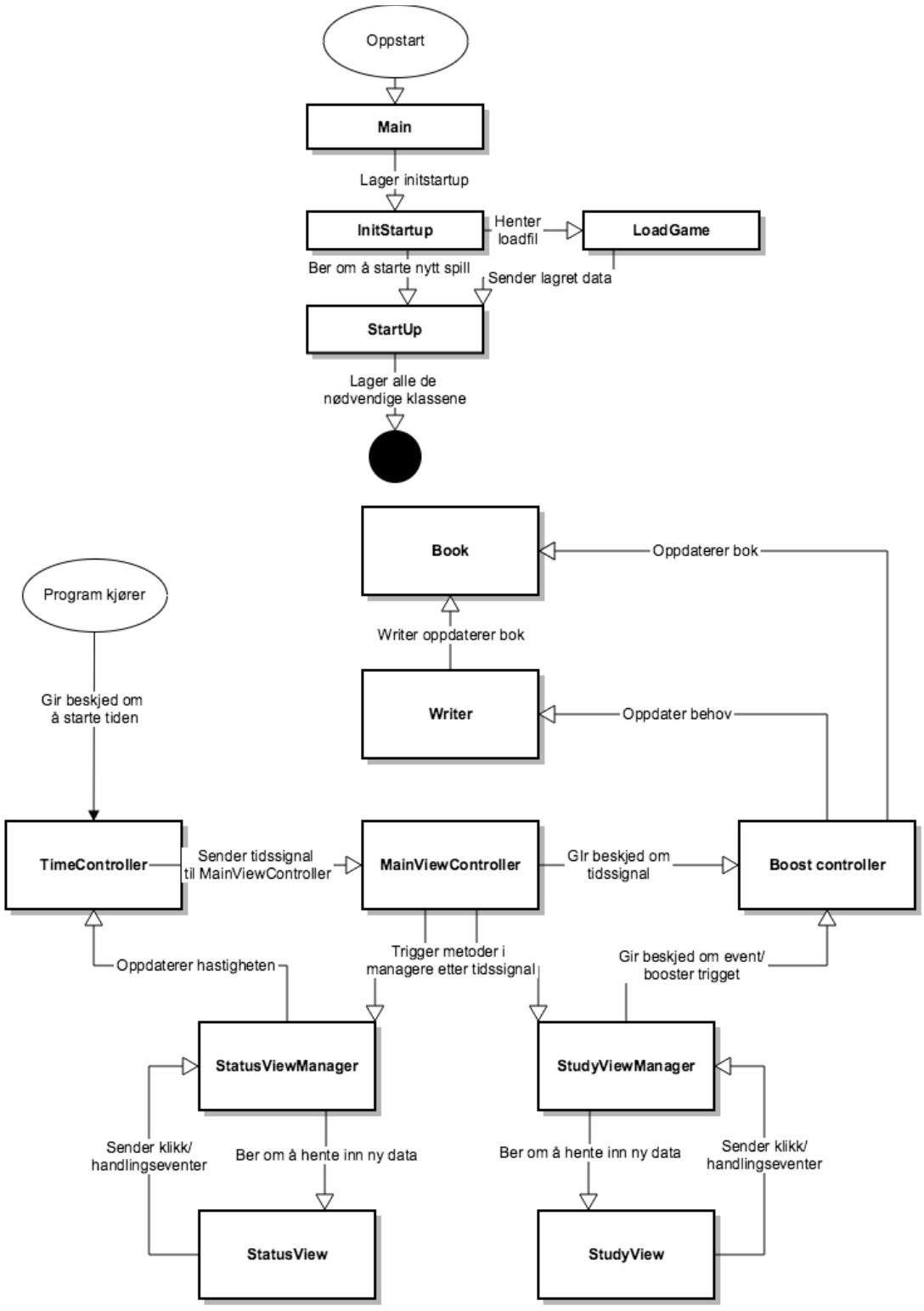
StudyView har også ansvar for enkle animasjoner. metodene AnimWrite og AnimView sørger for dette. Disse kalles for hvert tikk som sendes fra Time Controller klassen.

5. Diagrammer

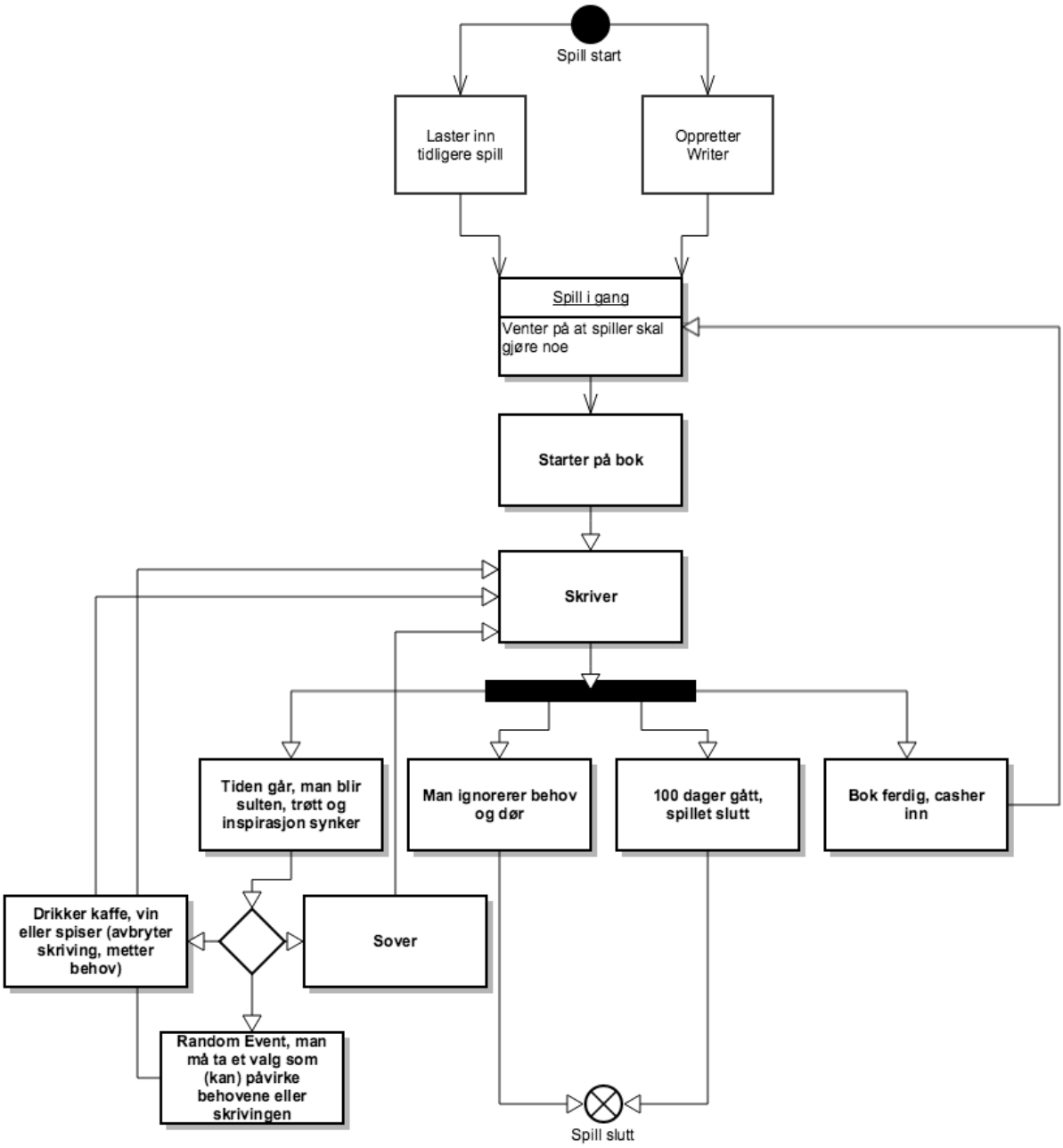
5.1 En klassereise -

hvor spillet starter og hvor veien går (i grove trekk)

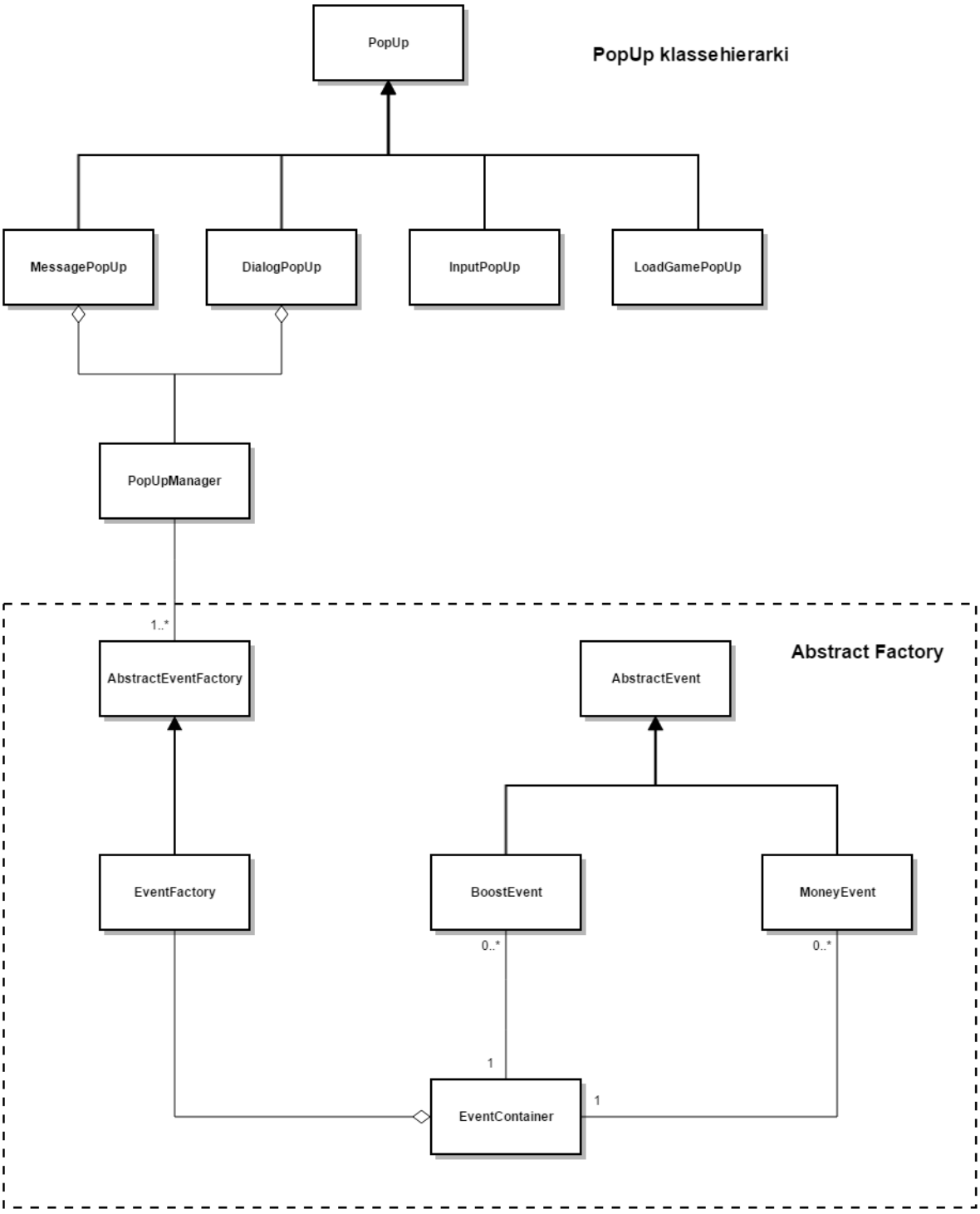
neste side


Oppstart
Main
Lager initstartup
InitStartup
Henter loadfil
LoadGame
Ber om å starte nytt spill
Sender lagret data
StartUp
Lager alle de nødvendige klassene
Book
Oppdaterer bok
Program kjører
Writer oppdaterer bok
Gir beskjed om å starte tiden
Writer
Oppdater behov
TimeController
Sender tidssignal til MainViewController
MainViewController
GIr beskjed om tidssignal
Boost controller
Oppdaterer hastigheten
Trigger metoder i managere etter tidssignal
Gir beskjed om event/ booster trigget
StatusViewManager
StudyViewManager
Sender klikk/ handlingseventer
Ber om å hente inn ny data
Ber om å hente inn ny data
Sender klikk/ handlingseventer
StatusView
StudyView

5.2 Flytdiagram



5.3 PopUp's og AbstractEventFactory



6. Patterns

6.1 ModelViewViewModel

Model, view, viewmodel handler om å separerere de ulike lagene. Se mer under mappestruktur. Det viktige her er at programmet sin logikk skulle fungert selv om man fjernet view-klassen.

MainViewController er “hjertet” på applikasjonen. Denne mottar “ticks”/signals fra TimeController hvert sekund. Så sender mainviewcontroller beskjed til alle klasser om at “nå har det gått et sekund”.

TimeController har som oppgave å holde rede på tiden. Når en time har gått, en dag, en uke, 100 dager. Dette forteller den til MainViewController som sørger for koblingene mellom de riktige klassene. 7 dager har gått, ok da skal husleieeventen kjøres fra popumanager. Som igjen sender et trekk til Writer.

Det er brukt pass-by-reference mellom alle klassene. Ved hjelp av signals til mainview får de andre klassene beskjed om hva de kan gjøre

6.2 Singleton

Klassene Filbehandler, XmlParser og WeatherReport og implementert med singletons. Vi følte dette var hensiktsmessig siden det ikke gir mening å ha flere enn en instanser av disse klassene. Singleton-klassene sine instanser er statiske og klasser har en lazy constructor som opretter en instanse i det den trengs. Siden instanser er deklarert som et statisk medlem inne i sin egen klasse, har vi latt destruktoren være privat. Hadde destruktoren vært public kunne instansen bli destruert ved en feiltakelse, for eksempel feil i koden. Klient koden kunne da ha kommet til å ha referanser til en instanse av singleton som ikke finnes. C++ sletter statiske medlemmer i det programmet avsluttes, etter at main har returner. Siden vi ikke implisitt kaller delete på instansene som har blitt allokert med new operatoren tror vi dette kan gi en feil positiv på minnelekasjer for eksempel i valgrind.

6.3 AbstractFactory

Vi har fulgt et abstract factory pattern når vi genererer random events. Vi har en baseklasse som heter AbstractEventFactory. Foreløbig finnes det bare en type konkret factory, EventFactory, men det er lagt til rette for å lage flere fabrikker. Factory'ene returnerer en EventContainer ved kall på create(). Det finnes bare en type EventContainer, derfor trengs det også bare en factory. Men EventContainer har virtual funksjoner og det er dermed lagt til rette for å arve denne klassen og så implementere andre konkrete fabrikker til å lage dem. Alle Quick-Time-Events(QTE) (de som pop'er tilfeldig opp på skjermen) lages i Eventfactory. Klassen har en constant: NR_OF_EVENTS som sier hvor mange forskjellige QTE som finnes. Hver QTE har sin egen metode, som trekkes tilfeldig ved kall på create(). QTE'en blir inkapsulert i en EventContainer, som inneholder all informasjon om de valgene og konsekvensene spilleren må ta.

Disse QTE'ene vises i PopUp-vinduer som dukker opp under spillet. PopupManager bruker en factory til å produsere disse og så koble de sammen med DialogPopUp- og MessagePopUp-vinduer. EventContaineren har AbstractEvent-pekere som den kjører execute() på, avhengig av hvilke valg spilleren gjør.

7. Refleksjon

7.1 Forklaring på noen designvalg

Formatering av ut-tekst ble gjort i view-ene. I utgangspunktet ville vi ikke at views skulle gjøre beregninger. Samtidig ville vi unngå bruk av QT-klasser såfremst det ikke var nødvendig for å utføre funksjonalitet.

View-klassene viser nå veldig mye tekst. Dette er for enklere debugge og vise fram dataen vi har lagret i modellene. I en ferdig versjon ville det vært en mer grafisk fremstilling av dataen fra modellene. Som f.eks. en stjerne som viser kvaliteten på boka, stor stjerne høy kvalitet e.l. Av samme grunn ser du de faktiske verdiene på needs. De kan gå til -10 og opp til 110. Dette for at det skal være litt å gå på. man skal kunne få en boost over normalen og en knekk under minimum. Men når den når -10 er du ferdig og når den 110 kan du ikke få mer. Det finnes et tak for needsverdiene.

7.2 Mangler (ting vi ikke rakk å implementere)

Skalering på statusview og generell utforming. Her får man mye informasjon i form av tekst som kunne vært fremstilt visuelt. Det gjennstår forsatt en del arbeid på design og vi mangler grep for at de ulike designvalgene fungerere på like plattformer.
Environmentboosters finnes ikke.
Spillet stopper etter 100 dager - det er tenkt at man skal kunne spille mye lenger og at spillet skal øke i vanskelighetsgrad etterhvert som man spiller. Mange kortsiktige mål og nye utfordringer skal gi spilleren lyst til å spille mer
Man får ikke betalt ved fullført bok og man har ingen måte å tjene penger på.
Man kan ikke sove (på sofaen)
Boostercontroller burde nok vært splittet opp i flere klasser. Som f.eks. Needscontroller og bookcontroller.
Når en randomevent dukker opp så kan en ny randomevent av samme type dukke opp. Dette er ikke heldig men er der for at randomevent skal dukke opp, nettopp random. De skal avbryte det du gjør og tas hånd om på det øyeblikket. Det er ikke noe nødvendigvis noe galt i det, men det er kanskje en logisk brist når man i spillet blir kalt opp av sin mor mens man allerede snakker med henne på telefonen.
BookEvent - arver AbstractEvent. Denne skulle fjerne eller legge til x antall ord i boka. Skulle bli brukt i random-events som: “De siste sidene du skrev kommer bort i sterinlyset og brenner opp .” eller “Du finner igjen noen gamle notater og de passer rett inn i boka!” e.l.
Flere konkrete fabrikker, som arver AbstractEventFactory og subklasser av EventContainer som redefinerer executeFirst() og executeSecond().
Mange flere random-events.

7.3 Utvidelser

Her er en liste over elementer vi ser for oss som naturlige utvidelser og *relativt* lett lar seg implementere.
Environmentalboosters, f.eks. skittent rom som må tas hånd om

Oversikt over hvilke eventboosters som er i kø, som evt. kan sorteres i en eventview, og representert som et ikon
Mer feedback til brukeren om hva som skjer, f.eks. hvor mye bonus man får for de ulike boosterne, hvilke boostere som står i kø, hva koster boosterne
At events hentes fra en tekstfil, på den måten kan man lage mange events i en fil og lett utvide med flere hendelser.
Ha en innboks hvor man kan motta meldinger fra f.eks. forlaget, tilbud om små skrivejobber eller fanbrev
Flere typer boosters
En shop med boosters
La økonomi være et viktigere gameplay element, f.eks. at kaffe og vin koster noe. Det er lagt opp til det nå og brukes litt, men det kunne vært mer brukt og tjent mer penger.
Ha flere typer needs
Straffes for å ignorere lave needs. Med tvunget til søvn, og verstefall game over.
Flere menyvalg når man klikker på forfatter. Vi hadde det men det ble fjernet fordi vi ikke hadde klar funksjonalitet
Om du gjør et valg i en event så vil det få forskjellig utfall fra gang til gang, gjerne basert på needs og tidligere valg.
Statisikk over antall skrevne ord, tidligere bøker, hvor høy hastighet man har på skrivingen, her setter faktisk bare fantasien (og dataen man samler inn) grenser.
Masse (triviell) statistikk kunne vært lagret f.eks. antall kaffer drukket antall side.
Minigames for å påvirke skrivingen/needs. F.eks. fluedreping, stavekonkurranse, hangman.
Ulike typer bøker/skrivemateriale som varierer i lengde.
At været påvirker spillet og ikke bare vises
En fil som inneholder alle variablene så man lett kan endre konstanter som hastighet, antall sider per bok, hvor mange dager spillet skal gå osv.
Gjøre spillet mer gøy, at det skjer mer og er mer “spillete”

8. Kjente bugs

8.1 Minnelekkasjer.

Vi har gått gjennom det vi kan og gjort tiltak for å unngå minnelekkajser. Men vi tror singleton kan bli tolket som en minnelekkasje. I tillegg mistenker vi biblioteker fra QT til å skape situasjoner som vi mistenker Valgrind tolker som minnelekkasjer. Vi har også brukt flere verktøy for å prøve å nøste opp i lekkasjene, men uten hell bl.a. Deleaker. QT analytic viste ingen feil, men vi opplevde dette verktøyet som litt ustabilt

8.2 Værdata:

Været blir hentet inn. Dette skjer ved hver oppstart av programmet, og dataene blir lest inn i en vektor. Men første gang du startet spillet 'rekker' ikke spillet å lage en vektor av disse dataene og tilfeldige data blir generert. Dette vil dog fungere neste gang du starter spillet.

8.3 Skalering av vinduet:

Bilde kan skallere, men et problem med skallering på de induviduelle QLabel-ene gjør at noen objekter i studyview kan forsvinne ved skallering av vinuet.

8.4 Kjøring under Lubuntu under VirtualBox

vi får man tre feilmeldinger.

libGl error: pci id for fd 12: 80ee:beef, driver(null)

libGl error: core dri or dri2 extension not found

libGl error: failed to load driver: vboxvideo.

Mistenker dette å være VirtualBox-relatert. https://www.virtualbox.org/ticket/12941

8.5 Henting av data med QNetworkAccessManager:

Qsslsocket genererer en qWarning() i runtime. Disse funksjonene blir ikke kalt, men prøvd. Grunnen er at det skjer et kall til

QNetworkAccessManager::supportedSchemesImplementation() som returnerer skjema http og dersom det finnes støtte, https. Dette hadde ingen direkte innvirkning på programmet så vi har latt den være.

qt.network.ssl: QSslSocket: cannot resolve TLSv1_1_client_method
qt.network.ssl: QSslSocket: cannot resolve TLSv1_2_client_method
qt.network.ssl: QSslSocket: cannot resolve TLSv1_1_server_method
qt.network.ssl: QSslSocket: cannot resolve TLSv1_2_server_method
qt.network.ssl: QSslSocket: cannot resolve SSL_select_next_proto
qt.network.ssl: QSslSocket: cannot resolve SSL_CTX_set_next_proto_select_cb
qt.network.ssl: QSslSocket: cannot resolve SSL_get0_next_proto_negotiated

8.6 Tiden

Tiden fortsetter å gå mens man skal skrive inn navn på boka man skal lage. Dermed kan en randomevent dukke opp, mens man skriver inn navnet på boka. Tiden burde kanskje ha stoppet opp.

9. Oppsummering

9.1 Utfordringer

En av de store utfordringen i dette programmet har vært å strukturere og holde kontroll på informasjonsflyten. Få en oversikt over hva de ulike klassene skal ha som oppgave og hvordan de ulike oppgavene skal bli startet.

En annen utfordring var å jobbe på ulike plattformer. Vi måtte sørge for at programmet kompilerte og kjørte på Lubuntu, OS X 10.10 og Windows 8.1. Her fikk vi litt problemer med

bibilioteker som ikke fantes på alle platformer. Slik som boost som har bedre random funskjoner, men finne ikke til OSX 10.10

9.2 Prosjektbeskrivelse kontra ferdig resultat

9.2.1 Minimumsmål

Prosjektbeskrivelse: Et rom med en forfatter som skriver en bok. Han/hun har behov som vises som behovsmetere. Det skal vær klikkbare elementer i rommet, tiden skal gå, en bok skal skrives og fremgangen på den skal vises i form av et meter som fylles opp. Behovene skal påvirke skriveprosessens hastighet og kvalitet. Etter at boken er ferdig vil spilleren få vite hvordan det gikk.

Resultat: Alle minimumsmål er oppnådd!

9.2.2 Ønskemål

Prosjektbeskrivelse: Økonomi hvor man kan kjøpe boostere som påvirker behovsmetrene (f.eks. kaffe eller rødvin) Kortsiktige mål og achievements når man oppfyller visse krav. Spillet samler statistikk om ulike gameplay elementer og presentere den Spilleren skal kunne lagre spillet(autosave). Laste et lagret spill og kunne avbryte et spill og begynne på nytt. Spillet skal kunne pauses og endre fart.
Vi legger til litt backstory - hvorfor er denne boken så viktig, hva har skjedd tidligere? Et mysterie kan introduseres og gir mere stemning i spillet.

Resultat: Økonomi har en liten rolle i spillet. Boostere påvirker behovsmetrene Spillet autosaver og man kan loade et tidligre spill. For å avbryte spillet må man avslutte hele spillet og så starte på nytt (eller loade). Man kan endre fart og pause spillet.

Vi fikk ikke til kortsiktige mål, achievements, økonomi som et viktig gameplayelement, og ingen backstory (som er nevnt i spillet…)

9.2.3 Hårete mål

Prosjektrapporten: Et script samler real time informasjon om været fra for eksempel yr.no. Dette vil ha en effekt på spillet i form av været som kan sees ut av vinduet og innspirasjonen til forfatteren. Minigames som for eksempel består av å drepe fluer ved å klikke på de. Minigames kan foregå i forskjellige rom eller utendørs.
Økonomi som et enda viktigere gameplayelement. Forfatteren får lønn av forlaget etter å ha sendt inn et vist antall sider, eller ved å gjøre andre skrivejobber. Pengene brukes til å betale regninger og kjøpe varer. Penger kan også brukes til oppdateringer slik som en bedre skrive skrivemaskin. Man kan gå konkurs - game over!
At man ikke bare skriver en bok, men mange bøker og målet blir å skrive bedre og bedre bøker og på den måten gjør det til et endless game.
Animasjoner, musikk og lydeffekter.

Resultat: Vi fikk til et og et halvt hårete mål; import av ekte værdata som vises i vinduet. Det påvirker ikke forfatteren (men det er ikke mye som gjenstår for å få det til) Vi fikk også til enkle animasjoner.

10. Referanser:

QT dokumentasjonen
http://doc.qt.io/

Pluralsight: Introduction to Qt: A C++ Cross Platform Application Framework
http://www.pluralsight.com/courses/introduction-qt-cplusplus-framework
Denne ble brukt til å lære oss basics i QT.

For å finne feilen om bygging i QT
http://stackoverflow.com/questions/23098134/usr-bin-ld-cannot-find-lgl-when-building-qt5-app-in-qt-creator-on-linux-mint

Om feilen i VirtualBox

https://www.virtualbox.org/ticket/12941

Hentet informasjon om singletons og dens destruktor

https://sourcemaking.com/design_patterns/to_kill_a_singleton