Rapport – Avancerad C++

Namn här

1. Problem

Ett spel skall byggas med hjälp av Simple Fast Media Library 2.3.2 och C++. Spelets utformning kommer att inspireras av det välrenommerade spelet *Alien Invaders* av Kristian Brodal.

Spelets källkod kommer att följa en konsekvent kodstandard, innehålla exempel på arv och polmorfism, och vara fri från minnesläckor, den väntas också kompileras till en körbar version och inlämnas till examinatören utan överflödiga källfiler.

Ytterligare förhoppas att const-korrekthet kommer att bibehållas, och att programmet är välbyggt vad gäller minneshantering och effektivitet.

Gällande gameplay-design kommer många element ändras men i största mån tekniskt sett bibehålla de funktioner som motsvarande element representerar i *Alien Invaders*. Till exempel kommer spelfältet möjligen vara roterat, och input-schemat ändrat, men spelets mål och hinder kommer att vara desamma, om möjligtvis krångligare/dummare.

En av examinatören uttryckligen framhävt kriterie för spelets godkännande var att konsollfönstret är dolt under programmets exekvering, det är av denna anledning spelets grafik huvudsakligen kommer bestå av ett konsollfönster, detta väntas få examinatören att vidröra sin panna och sucka.

++SVÅRIGHEteR

**I denna del ska problemet beskrivas som du försöker lösa. Det är samma problem som presenteras i inlämningsuppgiften, men du måste förklara den med dina egna ord. Detta är även rätt del att ta upp de krav som ställs på lösningen av problemet och svårigheter som kan förväntas stötas på under projekttiden.**

1. Design

Spelet, har dubbats till *”Whitespace Invaders”*, kommer som nämnt utgöra sig för att vara ett konsollfönster.

SFML har en robust och renderbar text-klass med transform-kababiliteter, denna kommer att stå till grund för det mesta i grafisk väg i *whitespace invaders* istället för texturer*.* Den grundläggande basklassen kommer representera en sfml-text och all tillhörande funktionalitet som bedöms lämplig, kallad *”Line”.*

En styrande hanterare kommer att sköta interaktioner mellan de olika *Line*s som är aktiva på skärmen. Denna kommer även svara för uppdatering och rendering av dessa entiteter. (singleton)

En statisk klass kallad *”LineWriter”* kommer att ansvara för skapandet av de olika entiteter som kommer att agera i spelutrymmet.

En spelsession kommer att delas upp i *”Screens”*, med en cyklisk ordning som sköts av en separat hanterare.

Samtliga entiteter kommer att kunna skifta mellan att följa fönstrets usla tillämpning av *drag-and-drop*

**I denna del ska designval tas upp gällande lösningen på problemet. Det är viktigt att skilja på design och implementation i detta fall. Skillnaden på design och implementation är nivån av detalj som tas upp. Designdelen ska innehålla en mer övergripande diskussion gällande lösningen på problemet. Exempel på ämnen som hör hemma i design-delen är hur olika komponenter hänger ihop och kommunicerar med varandra. Här ska även ett komplett UML-diagram över lösningen på problemet bifogas som en figur.**

1. Analys

Det är uppenbart att implementationen lider av strukturella brister, spelets exekvering träder fram och tillbaka mellan sina hanterare och entiteter flera gånger per enskild entitet och uppdaterring, på grund av hur mycket av funktionaliteten som implementeras i en annan klass en den borde.

Grundläggande funktionalitet som att till exempel återställla spelet är inte centrala, utan har kopierats ut i diverse funktioner och således finns ingen särskild kontroll över dem.

Samma problem återfinns och återkommer i flera klasser, i synnerhet ObjectHandler är en otymplig klass med för röriga och för stora metoder.

\* Getter/Setter använts fel

\* Handler är för stor

\* const inkorrekthet och inkonsekvens

\* *Line* innehåller bara viss funktionalitet

\* FollowWindow är en jävligt kukig design

\* Många funktioner borde delas upp i mindre void-funktioner

\* Designen är idiotisk, borde vara horisontell, går ju för fan inte att spela ens. Vafan hände med ”moveleft” och ”moveright” sättet :(

\* Måste ju finnas high-score för fan (kör varannan gamover, varannan entername, och kötta sedan tiden tills newgame)

**I denna del ska lösningen på problemet diskuteras, om det lyckades uppfylla alla krav som ställdes på lösningen. Har du identifierat några problem med din lösning? Kan din lösning på problemet förbättras, på vilket sätt? Ett program är aldrig perfekt, en lösning är aldrig perfekt. Det är viktigt här att påpeka delar där koden brister och delar där koden är bra, för att påvisa god förståelse.**

1. Körexempel

**I denna del ska programmets exekvering beskrivas steg för steg, tillsammans med bifogade figurer som visar hur de olika faserna i exekveringen ser ut.**

Figurer

Figurer i texten ska infogas på följande sätt tillsammans med numrering och undertitel.

Exempel:



Figur 1 Applikationens Game Over skärm.