LINKÖPINGS UNIVERSITET

TDP005 Projekt: Objektorienterat system

Kodgranskningsprotokoll

Författare

Denis I. Blazevic, denbl369@student.liu.se
Frans Bergström, frabe808@student.liu.se

INSTITUTIONEN FÖR DATAVETENSKAP

Höstterminen 2016
Version 1.0

2016-12-09

Denis I. Blazevic
Frans Bergström

1 Revisionshistorik

Ver.	Revisionsbeskrivning	Datum
1.0	Första version	161209

2 Möte

3 Granskat projekt

Beställargruppens kod har givit oss ett gott intryck. De visar goda kunskaper i C++ med hjälp av sitt Behavior upplägg de har med sina spelare och fiender. Men beställargruppen har poängterat att de inte ska använda det systemet då de inte fick det att funka så som de ville. Därför ska de byta ut detta till ett nytt system. Men om de fick Behavior upplägget att fungera så skulle det vara ett väldigt bra sätt att futureproofsitt spel.

Koden ser väldigt strukturerad ut och man vet i vilka klasser man hittar det man söker efter.

Nedan kommer ett par saker som vi har hittat i beställargruppens kod som kan förbättras. Dock är den kod vi kollat på inte den slutgiltiga versionen, och kan därför ha ändrats tills nu.

3.1 Variabler

Variabler i C++11 ska initieras på följande sätt:
int unsigned score **{0};**
Beställargruppen har däremot gjort det på följande sätt, och bör därför ändra sina initialiseringar från:
int unsigned score **= {0}**;

3.2 Inkonsekvent initialisering

Beställargruppen använder sig utav inkonsekvent initialiseringar. Till exempel skriver de:
sf::RenderWindow* window
sf::RenderWindow& _window
men i övrig kod dyker det upp annan stil också, såsom:
Player * player = new Player(window)
För att ovan kod ska bli detsamma som koden innan så ska det skrivas på följande sätt:
Player* player = new Player(window)

3.3 Inkonsekvent variabelnamn

I koden har vi märkt att det finns inkonsekvent variabelnamn. I de flesta fall används variabelnamn av typen '**__varName**' när variabeln är protected eller private. Men i resterande fall så används inte samma typ av variabelnamn.

3.4 Minnesläckor

Det är möjligt att det finns minnesläckor i koden. Detta pågrund av att många 'new' objekt skapas men de tas aldrig bort. Såg inte om de tas hand om i dekonstruktur.

Kan lösas genom att rensa alla objekt man inte behöver. Eller genom att använda **std::unique_ptr<>**.

4 Vårt projekt

Följande text är den konstruktiva feedbacken vi har fått från beställargruppen:

Angående andra gruppens kod så ger första anblicken ett väldigt positivt intryck. Koden ser väl strukturerad ut och varje klass har ett specifikt syfte. Även om detta till ett relativt litet projekt kan verka lite overkill, så anser vi detta positivt eftersom det ger goda möjligheter till expansioner och vidareutveckling framöver, om det skulle önskas. De utnyttjar många av funktionerna som C++ erbjuder, till exempel templates och #defines. Detta visar en bred kunskap inom språket.

Angående andra gruppens kod så ger första anblicken ett väldigt positivt intryck. Koden ser väl strukturerad ut och varje klass har ett specifikt syfte. Även om detta till ett relativt litet projekt kan verka lite overkill, så anser vi detta positivt eftersom det ger goda möjligheter till expansioner och vidareutveckling framöver, om det skulle önskas. De utnyttjar många av funktionerna som C++ erbjuder, till exempel templates och #defines. Detta visar en bred kunskap inom språket.

Det finns några få saker som skulle kunna förbättras.

På ett par ställen användes funktionsnamn som inleds med liten bokstav, men på de allra flesta så används stor bokstav.

Det finns en del “magic numbers” i koden, till exempel positionen spelaren startar på samt hur mycket spelaren flyttas per tidsenhet. Dessa https://www.ida.liu.se/ TDP004/current/ borde kanske vara definierade konstanter istället.

Delar av koden följer bra kommentering, andra delar saknar helt.

Tabb avstånd är ej konsekvent, se t.ex. “Object.h”.

Möjligen byta ut World initieringen, vad som är väggar och vad som är golv, så den läses in från en fil med ettor och nollor, istället för att ha det i koden.

4.1 Funktionsnamn

I vår kod så har vi funktionsnamn som inte följer projektets kodstil. Vi har funktioner som inleds med små bokstäver men även med stora bokstäver. Detta ska och kan fixas snabbt med hjälp av att använda refactor systemet i CLion.

4.2 Magic numbers

På ett par ställen i vår kod har vi så kallade magic numbers”(hårdkodade konstanter som aldrig ändras). Dessa konstanter ska bytas ut mot konstanter som komponenterna i vårt projekt bestämmer över. De magic numbers vi har just nu används bara i testsyfte.

4.3 Kommentering av kod

Kommentarer till koden ska läggas till, och definitivt finnas när klasserna är färdiga.

4.4 Tabb avstånd

Tabb avstånd ska fixas, detta görs genom CLions reformat funktion.

4.5 World initering

Detta ska vi ha i åtanke när refactor av World klassen sker.