LINKÖPINGS UNIVERSITET

TDP005 Projekt: Objektorienterat system

Kodgranskningsdokument

Författare

Love Jansson, lovja643@student.liu.se
Charlie Simonsson, chasi127@student.liu.se

INSTITUTIONEN FÖR DATAVETENSKAP

Höstterminen 2019
Version 1.0

2019-12-12

1 Revisionshistorik

Ver.	Revisionsbeskrivning	Datum
1.0	Första utkast	191212

2 Metod för granskningen

Granskningen skedde 11-12-'19.
Grupperna bytte dator och läste igenom varandras kod samt klassdiagram. Därefter diskuterade vi vad vi hade för feedback till varandra. Vi lämnade dessutom kommentarer direkt i deras kod på vår feedback så de har den där den är relevant.

3 Granskningsprotokoll

Vi började med att läsa igenom koden i main.cc för att få en övergripande uppfattning om hur den fungerade. Sedan gick vi djupare ner i de funktioner som kallades för att se hur variabler passerades ner i koden och hur de användes. Efter det gick vi igenom klassstrukturer för att få en uppfattning om hur klassdiagramet överenstämde med inplementationen. Nedan följer de viktigaste saker vi hittade.

Mycket variabler var lagrade i main som senare skulle användas i skapandet av de olika objekten i programmet. Till exempel var texturer och positioner lagrade i main och i objektet. Dessutom gav vi rådet att flytta initiering av sf::Sprite till objektets konstruktor istället för att skapa sprite och textur i main funktionen för att sedan skapa objektet med dessa som variabler.

Vi gav även feedback på hur de kunde flytta sitt spelarobjekt så att det inte längre överlappade efter en kollision. Detta genom att använda Sprite klassens inbygga funktioner för att räkna ut överlappet och flytta tillbaka spelaren det avståndet.

Vidare såg vi att de hadde en pure virtual klass (StaticObject) som inte hade någon funktion utöver vad klassen ovan den(Object) i arvs hierarkin redan hade. Gruppen kunde dock motivera detta med argumentet att det var framtidssäkring.

Designen av klasserna såg genomtängt ut och även om gruppen var mitt i en omorganisation av sin kod så var det inte svårt att få en överblick av programmet.

3.1 Feedback till oss

Vi fick kommentar på våra pilar i UML diagrammet vilket vi ska tänka på när vi konstruerar vårt slutgiltiga klassdiagram. Vi bör ta rätt på vilka pilar som ska användas när det gäller arv från interface och andra föräldraklasser.

En annan kommentar vi fick av gruppen handlade om hur vi strukturerat arv mellan objekt. I nuläget skiljer sig nämligen designen från hur vårt klassdiagram ser ut. Tanken från början var att ha en objektklass varifrån statiska och dynamiska objekt ärver ifrån. De objekt som sedan finns i spelet skulle ärva från antingen statiska eller dynamiska objekt. Vi fick senare problem då vi via objekt-pekare försökte nå variabler etc. i de nedersta klasserna vilket medförde att vi tog bort uppdelningen på dynamiska och statiska objekt. Vi har dock återinfört uppdelningen på dynamiska och statiska objekt nu och lagrar istället pekare av två sorter. Gruppen som granskade vår kod gav tips om att vi skulle ha en övergripande objektklass så som vi hade tänkt från början eftersom de två klasserna delar egenskaper (en variabel och en funktion). Vi funderade på

om det var värt att göra en övergripande klass på grund av detta eller om det var “onödigt”. Men det kan vara något vi ändrar på eftersom det kanske gör strukturen mer överskådlig samtidigt som vi i framtiden kanske kommer på flera gemensamma aspekter mellan de olika objekttyperna.

Vi fick även råd om att städa upp koden och ange kommentarer på en del stycken som behöver förtydligas.