LINKÖPINGS UNIVERSITET

TDP005 Projekt: Objektorienterat system

TDP005 - Kodgranskningsprotokoll

Författare

Anton Sköld, antsk320@student.liu.se
William Utbult, wilut499@student.liu.se

INSTITUTIONEN FÖR DATAVETENSKAP

Höstterminen 2017
Version 1.0

2017-12-13

1 Revisionshistorik

Ver.	Revisionsbeskrivning	Datum
1.0	Första utkast	2017-12-13

Innehåll

2 Granskningen

Några dagar innan mötet så tog grupperna kontakt med varandra. Vi kom då överens om att lägga till opponentgruppen som reporter i Gitlab-projektet. Vi kom även överens om ett datum för mötet då vi skulle ha bytt feedback på andra gruppens kod med varandra.

Mötet skedde 2017-12-12.

Spelet vi granskade koden för heter ”Cave of Pascal”. Det är inte helt uppenbart vad spelet går ut på ännu, eftersom andra gruppen inte hade en fungerande kompilerad version. Spelaren är en ödla som heter Pascal, och vi tror att man ska utforska en nivå med en del fiender i, så som flugor och fladdermöss.

3 Vår respons

3.1 Fullständig uppräkning

I `Game.cpp-filen` var det väldigt mycket fullständig uppräkning”. De hade t.ex skrivit ner alla möjliga kollisionsfall och if-satser för varje objekt i spelet.

Det skulle vara med utökbart och läsbart om man lagrade spelobjekten i en lista, möjligtvis en vector<Character*> eller vector<Entity*> eftersom många spelobjekt ärver från de basklasserna.

3.2 Const-funktioner

Vissa funktioner inom klasserna kan markeras med const, t.ex get-funktioner.

3.3 Kompilering

Projektet verkar inte gå att kompilera helt ännu, det finns ingen ordentlig CMake/Make-fil. Det kan vara användbart att skapa en sådan tidigt i projektet så man vet att koden man skriver alltid fungerar.

Som läget är nu så kompilerades endast enskilda komponenter, men ingenting `"sammanbundet"`.

3.4 Includes

Det finns många includes inuti `.hpp-filerna`. Dessa bör förflyttas till .cpp-filerna, om möjligt.

Vissa saker inkluderas även i båda `.hpp` och `.cpp-filerna`, så som `SFML/Graphics.hpp` inuti `Sugar.cpp/hpp.`

3.5 Basklassfunktioner

Vissa funktioner i Character skulle möjligtvis kunna flyttas till Entity. Främst de funktionerna som har med formen på objektet att göra, så som `get_inner_radius()`, `collision_handling` etc. Det är ju i Entity som objekten får sin form med en RectangleShape.

3.6 Main-filen

Main-filen är väldit fin. Många funktioner är bortabstraherade så att det räcker med en enkel `engine.run_loop()` i huvudloopen.

3.7 Minneshanteringen

Minneshanteringen ser fin ut, de försäkrar sig om ordentlig hantering med en användbar destruktor i vardera klass.

3.8 Namngivning

De flesta namn var väl valda, men några så som `"g"` beskrev inte vad den betydde, och skulle kunna förvirras med den matematiska konstanten g, detta var inte heller fallet.

4 Respons på vårt projekt

4.1 Överflödig inkludering

Anledningen för flertalet includes är att vi hade problem med att få saker att fungera och tog många olika versioner av includes innan vi fick saker att fungera och har inte rensat upp sedan dess, detta bör göras för snyggare och mer lättförstårlig kod men har ingen större påverkan på programmet.

4.2 Datamedlemmar deklareras utanför konstruktorn

Alla fall där variabeln kan (och bör) initieras direkt är orsaken ovana av kodstilen. Detta är dock lätt åtgärdat.

4.3 4 klasser i mixin är förvirrande

Själv tycker vi strukturen med alla egenskapsklasser i samma fil är att föredra (i alla fall med detta antalet klasser) men kan hålla med om att mixin inte är ett självklart namn för alla. För nuvarande har vi inga planer på förändringar.

4.4 Radlängd

Radlängd är något som har ignorerats stundvis och bör åtgärdas.

4.5 Namngivning av variabler

> De flesta variabler har bra namn (man förstår vad de gör), men namnet på Spawner pekaren som skapas innan första while loopen är mindre bra och samma namn används även i for looparna vilket gör det lite förvirrande.

Namnet på spawn-pekaren utanför looparna är temporär och spawn under testning, men vi håller med om att s”är ett olämpligt namn framöver.

4.6 Utökade kommentarer

Vi själva har hållit med om att fler kommentarer är att föredra men har konstant fått feedback på att reducera antalet kommentarer på tidigare uppgifter om inte att ta bort alla, då koden är självbeskrivande. Vi har av denna anledning inte kommenterat lika mycket som vi hade velat.