LINKÖPINGS UNIVERSITET

TDP003 Projekt: Egna datormiljön

Dokumentmall

Författare

Nils Bark, nilba048@student@liu.se
Hadi Ansari, hadan326@student.liu.se

INSTITUTIONEN FÖR DATAVETENSKAP

Höstterminen 2020
Version 1.0

2020-12-09

1 Revisionshistorik

Ver.	Revisionsbeskrivning	Datum
1.0	Kodgranskningsdokumentation skapad	2020-12-09

2 Mötet

Mötet hölls på morgonen den 9:e december kl 9. Egentligen skulle mötet ha hållts kl 8 men det skedde viss försening från vår del och på grund av detta kunde en medlem i den andra gruppen inte närvara. Båda grupper valde ändå att hålla mötet eftersom vi ansåg att det var tillräckligt viktigt att få informationen i god tid än att alla gruppmedlemmar var där. Mötet skedde via Zoom, där gruppen som gav feedback använde skärmdelning för att visa delarna av projektet man syftade på. Vi tog anteckningar under mötets gång för att komma ihåg den feedback vi fick. Mötet varade i ungefär 50 minuter.

3 Vår granskning

Vår granskning av den andra gruppens kod ledde inte till en stor mängd kommentarer. Detta på grund av att det visade sig vara förvånansvärt svårt att hitta fler än ett par delar som var värda att peka ut då koden var väl skriven, prydlig och välkommenterad. Vi inledde med att påpeka just det, och gav gruppen beröm för tydliga variabelnamn, bra inkapsling och god mängd förklarande kommentarer som gjorde det helt smärtfritt att läsa igenom koden.

Vi noterade även att de hade skapat filer och tomma klasser för de delar av projektet som inte var implementerade än. Till exempel hade de en fil med klassen `shooting_behaviour()` i trots att själva klassen var tom och spelet inte hade den funktionaliteten då vi testade det. Vi ansåg att detta var ett smart sätt att se hur filstrukturen i slutprodukten kunde se ut, och kunde även fungera som en checklista.

Den mest anmärkningsvärda kritiken vi hade var att det skedde mycket koduppprepning i deras draw-funktion, där de hade flera for-loopar för att rita ut olika objekt trots att de alla gjorde samma sak. Detta hade de precis innan mötet upptäckt på egen hand och åtgärdat, men vi bestämde oss ändå för att nämna det. Sedan hade vi ett par frågor om hur de hade strukturerat sitt UML-diagram. Vi undrade först varför både klassen `Game_Object` och `Moving_Object` hade kompositionsrelation direkt till `Level` när `Moving_Object` redan var en subklass av `Game_Object`. Sedan frågade vi varför de hade satt en relation mellan `Behavior` och `Enemy`, men inte mellan `Behavior` och `Player`, då vi fick intrycket att spelaren också kunde använda dem.

4 Andra gruppens feedback

Den första delen feedback vi fick var en fråga om varför alla klasser i vår kod var pure virtual. Efter en diskussion kom vi fram till att de hade rätt när de pekade ut specifika klasser där pure virtual var onödig. Sedan noterade de flera instanser av kodduplikation som bör åtgärdas och knöt detta till följande argument som var att upp till 50% av koden kunde tas bort och skrivas om för att få en dramatisk minskning i mängden kod som används för att få samma resultat. Ett exempel var att skapa en klass för våra plane-fiender och använda konstruktorn för att modifiera vilken specifik typ som skapas istället för att använda en seperat klass för varje fiendetyp. Vi ansåg att delar av detta exempel tillsammans med den överhängande kritiken om koduppprepning, och viss hårdkodning var korrekt.

Sedan pekades det ut att vi använde en oeffektiv lösning för kollisionshantering, där varje objekt kontrollerade om det kolliderade med alla andra enskilda objekt vid varje uppdatering. De rekommenderade även att vi inte använder sprites för att kolla kollision och gav ett exempel som använde sig av “squares” istället som skulle ha en inbyggd funktion för kollision. Vi fick viss kritik mot vår namngivning av variabler, då flera

av dem var kvar från testvariabler och aldrig hade uppdaterats till att vara mer förklarande. Dessutom var de inte namngivna i samma stil som våra nyare variabler (jämför till exempel clock1 mot shield_clock). En kommentar pekade på de klockor vi använde oss av i koden, och frågade ifall de faktiskt behövdes, men när vi förklarade deras funktion släppte de just den kritiken. Våra klasser använde mycket lite inkapsling, och de pekade ut att detta kunde leda till diverse problem när man ändrar på funktionaliteten i vissa klasser.

4.1 Åtgärder

De klasser där pure virtual pekades ut som onödigt har tagits bort eller skrivits om på ett bättre sätt. Vi har flyttat definitionen för flera medlemsfunktioner till basklassen och på så sätt minskat storleken på subklasserna. I vissa fall där en funktion i basklassen inte används av alla subklasser har vi flyttat ut funktionen till de specifika subklasserna som använder den istället. Till exempel har vi flyttat ut `want_shoot()` ur `entity` eftersom att bland annat power-ups aldrig kommer att vilja skjuta. Vi planerar att fortsätta med sådana lösningar och vi tror att vi kan minska kodstorleken rejält med hjälp av dem.

En ändring som gjordes efter mötet var att våra power-ups, som tidigare hade seperata klasser för varje typ, nu kunde skapas från konstruktorn i basklassen istället. Detta innebar att vi kunde radera de tre underklasserna som tidigare fanns. Vi kunde även då åtgärda den hårdkodning som fanns i de klasserna. Vi funderar även på liknande lösningar för att hantera bullets och fiender.

Den kritik vi fick mot vår kollisionshantering var helt rätt och vi inser att vår nuvarande lösning är oeffektiv. Vi tänker göra stora ändringar på vår kollisionshantering med hjälp av vägledning från assistenterna. Vi ska även se om exemplet vi fick i vår feedback kan vara relevant till spelet.

Namngivningen av variabler åtgärdade vi snabbt efter mötet. Alla variabler har nu mer förklarande namn och de håller alla samma stil. Till exempel heter `clock1` nu `shoot_clock`, medan `sht` togs bort helt och hållet.

Vi har åtgärdat alla inkapslingsproblem vi märkte i vår kod. Till exempel ändrade vi `Entity` från struct till klass, medan medlemsfunktionerna och variablerna nu är korrekt uppdelade i protected och public. Vi behöver nu två getters, men det är en mycket liten konsekvens av en förbättrad inkapsling. Vi kommer att tänka mer aktivt på hur vi hanterar vår inkapsling när vi jobbar vidare på projektet.