LINKÖPINGS UNIVERSITET

TDP005 Projekt: Objektorienterat system

Kodgranskningsprotokoll

Författare

Niklas Åsberg, nikas214@student.liu.se

INSTITUTIONEN FÖR DATAVETENSKAP

Höstterminen 2023
Version 1.0

2023-12-05

1 Revisionshistorik

Ver.	Revisionsbeskrivning	Datum
0.1	Utkast	2023-12-03
1.0	Dokument efter mötet	2023-12-05

Innehåll

2 Introduction

I detta dokument så granskar jag projektet gjort av Lucas och Theo. Vi började med att bjuda in varandra till respektive GitLab-repo och sen hade vi ett möte där vi gick igenom varandras projekt. Mötet skedde tisdagen den 5:e december 2023 klockan 13:30 i salen LIS4. Mötet gick väldigt bra och kritiken som lades fram var mycket konstruktiv.

3 Lukas och Theos projekt

Projektet är ett tvådimensionellt arkadspel som visas ut ett fågelperspektiv där man spelar som maskoten för IP-programmet (IP-spöket) med målet att döda så många fiender som möjligt innan IP-spöket dör.

3.1 Övergripande kommentar

Det förekommer mycket användning av iteratorer, dessa kan vara bra för prestanda men kan bli aningen svårlästa, framförallt om de inte är namngivna väl.

Smartpekare förkommer nästan överallt, dessa är mycket användbara för att säkerställa att minnesläckor hålls nere. Jag tycker personligen att de är aningen svårlästa jämfört med vanliga pekare.

Ganska hög coupling förkommer mellan klasserna, framförallt mellan klasserna World och Player. I ett objektorienterat projekt vill man ha så låg coupling som möjligt.

3.2 Klasser

Klass	Beskrivning	Beroenden
Character	Representerar en karaktär	Förälder till Player och Enemy
Enemy	Fiende som vill skada spelaren	Barn till Character
GameObject	Ett object till spelet	Förälder till TextureObject
TextureObject	Ett object som kan ha en texture	Förälder till Character
Player	Spelaren	Barn till Character
State	Grundklassen som håller koll på spelets 'state'	Innehåller en instans av World
TextureManager	Hanterar texturer för karaktärer	Kallas av TextureObject
World	Representerar världen	Innehåller alla gameobject
Weapon	Vapnet som spelaren använder	Ägs av Player

3.2.1 State

I klassen State så finns en metod som heter render, den metoden tar in en referens till spelfönstret (sf::Window). Referensen är namngiven 'to' vilken är ett ganska otydligt namn då den inte förklarar vad det är för något. Ett bättre namn hade kunnat vara 'window' eller 'gameWindow'.

3.2.2 TextureManager

TextureManager-klassen känns onödig att ha. Det är bra att samla texture-objekten på en och samma ställe men det hade nog blivit lättare om de fanns i en vector eller enum i till exempel World-klassen.

3.2.3 Player

Hanteringen av kollisioner sker uppdelat mellan Player-klassen och World-klassen. Det hade blivit lättare och följt den objektorienterade filosofin om det endast hanterades i en av dessa klasser för att få så kallat 'separation of concerns'.

3.2.4 World

Sättet att hålla koll på hur länge spelaren överlevt känns onödigt komplicerad, delar sker i World-klassen och andra delar sker i Player-klassen.

Ett sätt att lösa detta kan vara att ha en klocka som ränkar från det att Player skapas tills det att Player är förstörd.

Denna klass delar även mycket ansvar med Player-klassen.

3.2.5 TextureObject

TextureObject ärver ifrån klassen GameObject och ärvs bara av klassen Character. Man skulle kunna göra sig av med denna klass utan några problem.

4 Mitt projekt

Theo och Lucas gav mig en hel del bra konstruktiv kritik.

4.1 Wall

Varje gång en instans av Wall-klassen skapas så skapas en separat instans av sf::texture, för att minska användandet av resurser så bör det bara finnas en instans som Wall-instanserna pekar mot.

4.2 Player

När spelaren rör sig så använder den sig av directions, dessa bör normaliseras så att spelaren inte rör sig snabbare på diagonalen är vad som är tänkt.

4.3 Map

I Map-klassen ligger alla GameObject-instanser i en vector. Den bör delas upp i två, en för GameObjects och en för Characters.

4.4 Generellt

I konstruktorerna sker mycket tilldelning av attribut, dessa för göras i en initieringslista istället.