**Expert review k1**

Siem Baltus

Ik heb de expert review gezamenlijk gehouden met de rest van mijn groepje. In de review hebben we het gehad over de verschillende aspecten van OOP in de game engine die we hebben gekregen. We begonnen de review met de vraag “Wat doe je voor je gaat beginnen met programmeren?”, Het antwoord hierop is dat je moet beginnen met het maken van een classdiagram. Als je de classdiagram al hebt gekregen hoef je deze niet opnieuw te maken maar dan gebruik je deze met het programmeren. Hierdoor weet je voordat je gaat programmeren hoe je je classes gaat benoemen en wat je erin gaat stoppen, ook weet je hoe je de classes met elkaar gaat verbinden. Daarna kregen we te horen dat we alles in onze code in het Engels moeten schrijven. Dit moet zo zodat iedereen de code kan begrijpen, dus mensen die geen Nederlands spreken het ook begrijpen. Als alles in de zelfde taal is geschreven wordt de code ook universeel en dus makkelijker te begrijpen.

Toen we daarna naar de code gingen kijken kregen we de vraag van wat een algoritme is. Hierover kregen we uitleg en in de uitleg werd verteld dat als er bij een if en else statement iets true of false kan zijn, dit is een boolean in mijn code. Als laatste hebben we uitleg gekregen over de verschillende pijlers van OOP zoals: abstraction, inheritance, encapsulation en polymorfisme.

**Starrt**

**Situatie:**

De situatie was dat wij als groep een expert review hadden ingeplant voor k1. In de expert review hebben we het vooral gehad over het toepassen van OOP aspecten binnen de game-engine.

**Taak:**

De taak was het bespreken van onze code, en om dus beoordeeld te worden op k1. Bij het maken van onze code hebben we gebruik gemaakt van de meegeleverde game-engine, en de uitleg video’s die gaan over het gebruik van de game-engine. Hierdoor hebben we veel geleerd over het werken met typescript, en ook het samenwerken in een grotere groep.

**Actie:**

In de review hebben we feedback en ook uitleg gekregen over de volgende onderwerpen:

* Dat we alles in onze code in het engels moeten gaan schrijven.
* Uitleg over wat algoritmes inhouden.
* Inheritence in OOP
* Encapsulation in OOP
* Abstraction in OOP
* Polymorphism in OOP

In de uitleg van de verschillende OOP principes hadden we het bij encapsulation over hoe een class eruit ziet. We moesten dus ook een class tekenen, hierdoor heb ik geleerd dat als je een class gaat aanmaken dat je begint met een classnaam bijvoordbeeld: beker. De class bestaat uit verschillende attributen en methodes, deze attributen kunnen bijvoorbeeld bij de class beker: de inhoud, type vloeistof. Voorbeelden van methodes zijn bijvoorbeeld: drinken of legen.

Verder hebben we het gehad over het onderverdelen van methodes in setters en getters. Een setter kan bijvoorbeeld een attribuut aanpassen of een waarde aan een attribuut geven. Een getter haalt informatie op bij attributen. Je kan een attribuut niet rechtstreeks wijzigen, dit kan wel doormiddel van een methode.

Bij de uitleg van encapsulation hebben we geleerd dat meestal de attributen private zijn en methodes meestal public. Als de attributen private zijn kunnen de methodes en parameters alleen binnen de class gelezen worden,subclassen met de zelfde attributen kunnen dus nog steeds niet bij de superclass. Als een class public is kan elke class bij de attributen of methodes van die class, deze classes kunnen dus ook input van buitenaf krijgen.

Daarna gingen we het hebben over inheritence. Hierin hebben we geleerd dat een class aspecten kan overnemen van andere classen ze “erven” het dus.

Over polymorphism hebben we het nog niet echt gehad omdat de tijd op was maar ik heb hier zelf nog wat onderzoek naar gedaan. Uit dit onderzoek heb ik begrepen dat classes een attribuut van een superclass overnemen maar dan zelf ook nog andere attributen en methodes hebben. Zo heb je bijvoorbeeld de class auto met als superclass voertuig, dan heb je de subclass bus de class bus heeft dus bijvoorbeeld ook wielen net als de auto maar de wielen van een auto zijn anders dan die van een bus.

**Resultaat:**

Door deze expert review snap ik de verschillende besproken principes van OOP een stuk beter. Hierdoor kan ik dus ook beter werken met OOP en typescript, en kan ik dus ook betere codes schrijven.

**Reflectie:**

Ik heb dus veel geleerd van en weer informatie omhoog gehaald door deze expertreview. Ik heb dus een goede uitleg gekregen over een aantal verschillende OOP principes die ik nog niet zo goed wist. En ik heb ook weer een hoop informatie die ik het vorige blok heb geleerd weer opgefrist. In blok 2 had ik namelijk veel geleerd over OOP en typescript. En dit blok een stuk minder nieuwe dingen geleerd, omdat we bijvoorbeeld al een classdiagram en een game-engine hebben gekregen dus hoefden we dit niet zelf te leren maken.

**Transfer:**

Wat ik allemaal heb geleerd van deze expertreview. Deze informatie kan ik dus in andere projecten ook weer gaan gebruiken. Ik wil voor het volgende blok dus ook meer gaan leren over OOP omdat ik dit nog wel een lastig onderwerp vindt dat wel erg belangrijk is met het programmeren.