Project ideeen

Balatro clone of andere card collection game.

# Leerdoelen herkansing cookieclicker

OOP

Inheritance

Interfaces

Meerlaagse OOP

Class diagrams

# Leerdoelen herkansen mvc

MVC fase 3

ORM

ERD

Login systeem

# Leerdoelen uitbreiding

Ajax requests

PHP en JS classes mirroren en mbv ajax en json serializen en deserializen

Abstract classes (can’t be initiated by itself, is meant to be initiated through exending a child class)

Polymorphisme (overriding + interfaces) (child class overwrites parent class for self)

Gebruik super keyword (call parent class constructor)

Factory en builder design pattern, misschien composition over inheritance concept.

SOLID

Dependency injection (composition over inheritance) (bv door method die item in frontend saved te injectecten, zodat je zowel naar de backend als lokaal kan opslaan door een andere implementatie te maken).

Visitor pattern gebruiken voor serialization.

Builder pattern gebruiken voor object creatie, en samen met strategy pattern voor deserialization.

# Project cardgame

Je gaat een digitaal kaartspel maken dat lijkt op spellen zoals Yu-gi-oh, Balatro, Pokemon, Magic the Gathering, of Hearthstone. In dit spel moet een spelen singleplayer (of, als uitbreiding, met 2 spelers) een spel kunnen spelen met kaarten. Deze kaarten moeten op het scherm te zien zijn voor de speler, ze moeten te selecteren zijn, eventueel met animatie en slepen.

De kaarten moeten een aantal eigenschappen kunnen hebben. Een aantal kaarten moeten special abilities hebben. Deze werken samen met de andere aspecten die je in je spel moet maken zoals levels, mana, hands, etc. Het is toegestaan een van de bovengenoemde spellen als directe bron te nemen voor de gameplay, zo lang je de code zelf schrijft. Auteursrechten zijn belangrijk, dus maak je eigen placeholder textures en namen. Afhankelijk van de complexiteit van je spel, werken je kaarten op verschillende manieren samen, zoals bv een kaart die zolang hij in je hand blijft, alle attacks met een water element 2x zo sterk maakt, eentje die een pokemon revived als hij gespeeld word, etc.

Het project heeft 3 beoordelingsmatrixen, op 3 niveaus. In de kolommen is per niveau aangegeven welke prioriteit de eis heeft, volgens MoSCoW, met w zijnde “won’t have”. Er is een niveau voor diegene die de basis van OOP moeten herkansen (het cookieclickerproject en mvc onvoldoende gehaald), een niveau voor diegene die MVC moeten herkansen maar cookieclicker gehaald hebben, en een verdieping voor diegenen die beide met een voldoende hebben afgerond.

Het is natuurlijk toegestaan om, als je klaar bent met het vorige niveau, items van het niveau erna te gaan doen.

Gemaakt werk van jezelf van een vorig project, zoals je eigen login systeem, mag je hergebruiken en ombouwen naar dit nieuwe spel. Dit mag alleen als jij de vorige keer het login systeem gemaakt hebt. Zet dit dan in een comment er bij, en zorg dat je het snapt.

Groepsgrootte van 2.

# Eisen

|  |  |  |  |
| --- | --- | --- | --- |
| Eis | Basis OOP | MVC | Ver-dieping |
| Userstories opstellen | M | M | M |
| ERD | W | M | M |
| SCRUM in sprints (van 2 weken per sprint), en dus ook een scheiding tussen backlog en todo | C | M | M |
| SCRUM burndown chart | W | C | M |
| SCRUM punten poker voor alle backlog items. | W | C | M |
| GIT, met vanaf het begin, met het hele groepje, regelmatig committen. | M | M | M |
| Definition of Fun | M | M | M |
| Definition of Done | M | M | M |
| Wireframes | M | M | C |
| Activity diagrams | W | C | M |
| Class diagrams | M | M | M |
| Use case diagram | C | M | M |
| Kaarten zijn gemaakt als class. | M | M | M |
| Er zijn ten minste 5 kaarten met elk verschillende functionaliteit (niet eigenschappen zoals 10 health vs 20 health, of een simpele statistische verhoging van een attack van 3 naar 4). | M | M | M |
| Er zijn 10 variaties op kaarten die verschillen puur door een property zoals aantal levens, attack, etc, te veranderen. (zelfde class, andere values in properties) | M | M | M |
| Er is een kaart wiens properties kunnen veranderen door het spel te spelen. | S | S | M |
| Voor het realiseren van de verschillen in functionaliteit wordt ten minste 1x gebruik gemaakt van een abstract class. Dit telt niet voor het punt overerving. | W | W | M |
| Voor het realiseren van de verschillen in functionaliteit wordt ten minste 1x gebruik gemaakt van overerving. | M | M | M |
| Alle kaarten implementeren ten minste 1 gedeeld interface. | M | M | M |
| De gameplay of progress kan worden opgeslagen in de browser. | M | W | W |
| De score, level of een simpele beschrijving van een spelers deck/inventory kan worden opgeslagen in de backend, en geladen/getoond worden. Niet te complex. | W | S | M |
| De volledige game state op elk moment kan worden opgeslagen in de backend. | W | C | S |
| De speler in de browser side (js) is geïmplementeerd als class (en het spel is dus niet opgeslagen in een aantal losse variabelen) | C | S | M |
| De backend is volgend MVC zoals in Fase 3 voorbeeld gemaakt. | W | M | M |
| Api points zijn ook in MVC Fase 3 gemaakt. | W | C | S |
| Er is een login systeem. | W | M | M |
| Het wachtwoord in het login systeem is gehashed en gesalt. | W | M | M |
| De ORM in de backend implementeert een interface die de CRUD operaties beschrijven. | W | S | M |
| MVC Fase 4 router is toegepast. | W | C | C |
| Data wordt opgehaald en gestuurd door middel van Ajax requests naar een API, voor de meeste onderdelen. | W | W | M |
| In PHP en JS bestaan er parralelle versies van dezelfde class met betrekking tot alle nodige informatie, op zijn minst de properties. Hiermee kan er in de backend en frontend met dezelfde data gewerkt worden in een OOP manier. Deze kunnen gesynchroniseerd worden, door b.v. JSON en Ajax. | W | W | M |
| Er wordt op 1 plek gebruik gemaakt van method overriding in inheritance om polymorphisme te bereiken. | W | W | M |
| Er wordt gebruik gemaakt van het super keyword. | W | W | M |
| De builder of factory design pattern wordt gebruikt voor het creëren en configureren van de kaarten. | W | W | S |
| Het concept “composition over inheritance” wordt gebruikt binnen het ontwerp van de kaarten. Als dit gedaan wordt, mogen eerdere inheritance en abstract classes eventueel weggehaald worden. Maak dan in git een tag aan voordat je deze weghaalt, zodat deze alsnog aangetoond kunnen worden. | W | W | S |
| Gebruik een visitor pattern om objecten te serializen. | W | W | S |
| Gebruik een strategy pattern om objecten te deserializen (tip, samen met een builder pattern). | W | W | S |
| Pas dependency injection toe in de frontend (js) voor het opslaan en ophalen van data, zodat je kan uitbreiden naar andere opslagmethoden zoals local in de browser. | W | W | C |
| Implementeer opslaan van data in de browser via dependency injection. | W | W | C |
| Neem de SOLID principes mee in je design, ligt toe hoe je ze hebt toegepast, of waar je er bewust van bent afgeweken en waarom. | W | W | M |
| Laat classes meerdere interfaces implementeren. | W | W | C |
| Acceptatietest | W | C | M |