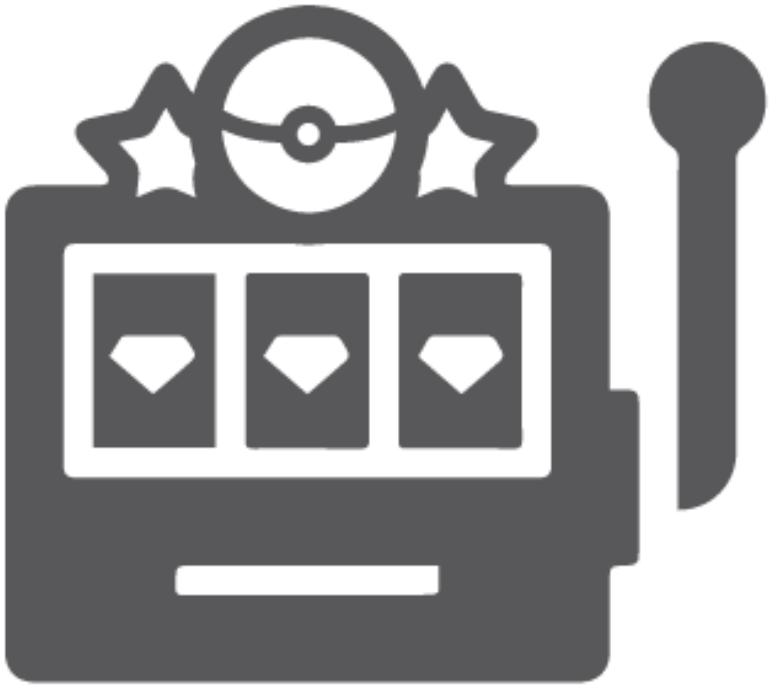
Pokémon Gamble Evolution



Naam: Sergi van Ravenswaay
Studentnummer: 3017869
Module: GDV1 – Technisch Document

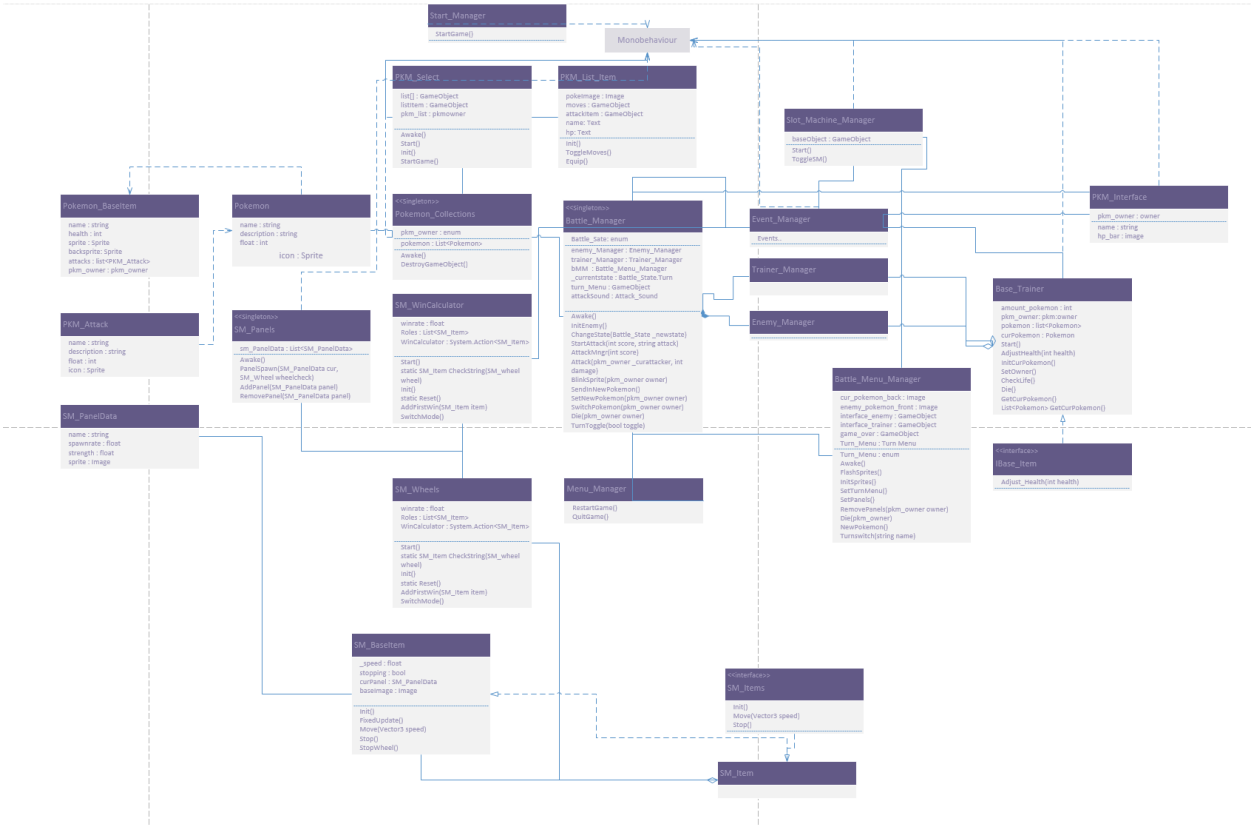
Inhoudsopgave

Naam: Sergi van Ravenswaay
Module: GDV1- Documentatie

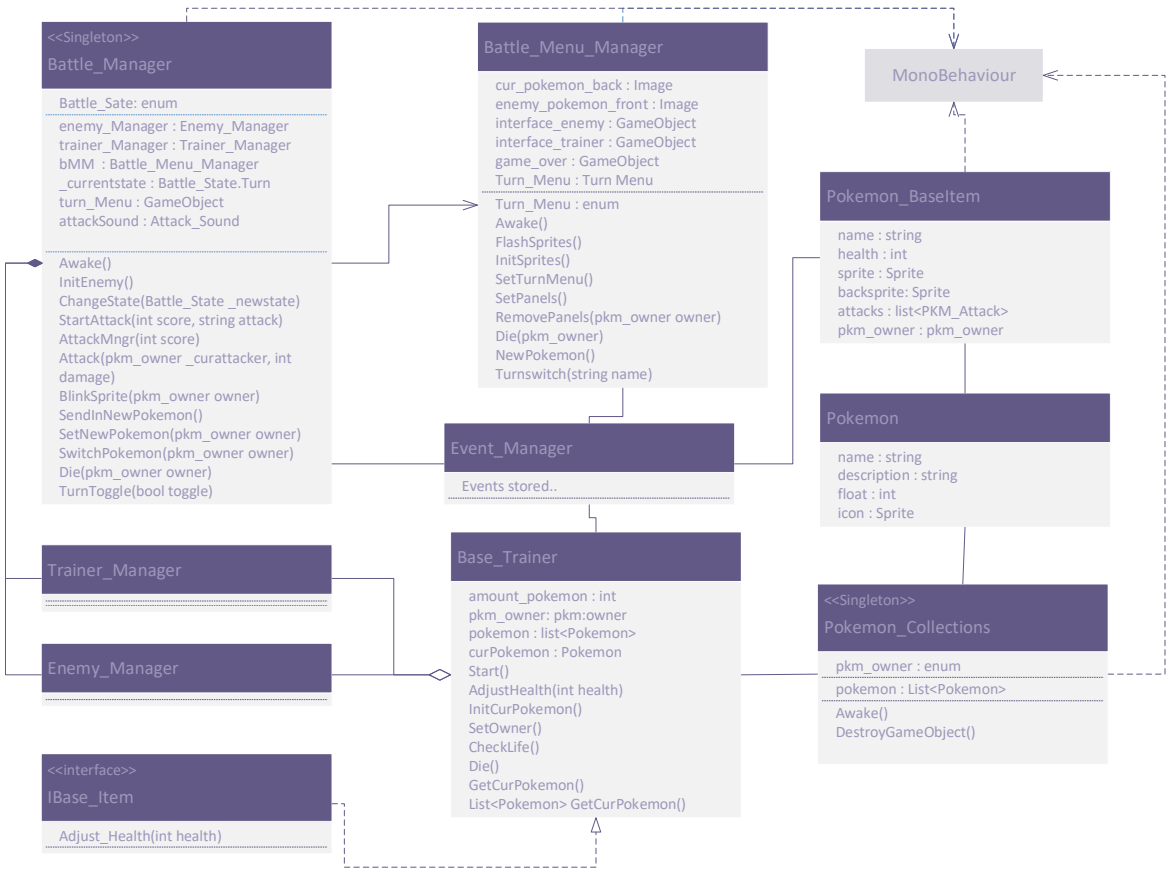
Concept:

Mijn game is gebaseerd op de Pokémon Gold Battle mechanic. In plaats van dat je aanvallen en Pokémon kan kiezen in dit spel word dit random bepaald met van een gokkast.

UML:



De UML die ik voor mijn game had gemaakt was in eerste instantie heel complex en groot. Ik heb daarom het belangrijkste element hiervan gepakt en een overzichtelijkere UML gemaakt.



Dit is de UML van de Pokémon Battles. De Pokémon Battles zijn attacks, switchen van Pokémon en de beurten worden geregeld door deze manager. Deze staat in contact met de Battle_Menu_Manager die alle UI componenten regelt.

De Battle_Manager is zo gemaakt dat hij alles achter de schermen regelt en dit doorgeeft aan of de Event_Manager of de Battle_Menu_Manager.

De Pokémon_Database_Collections is een singleton die accessable is voor iedereen (Singleton), maar deze word hier alleen door de Base_Trainer bezocht. De Pokémon is een empty script wat afgeleid is van Pokémon_BaseItem. Voor enige uitbreiding hoef ik alleen maar de Pokémon_BaseItem aan te passen.

In de Battle_Manager staan ook de Trainer_Manager en Enemy_Manager. Deze zijn beide afgeleid van de Base_Trainer. Op die manier kan ik gegevens via de Battle_Manager snel aanpassen en blijft data van de Enemy en de Player gescheiden.

In de Event_Manager staan alle events in opgeslagen. Deze kan je aanroepen door een static functie aan te roepen. Vandaar dat hij in mijn UML ook het meest is geconnect met de andere scripts.

Waarom?:

Pokémon zijn battle mechanic geeft je een enorm voordeel als je de sterkste Pokémon hebt. Nu heb ik ervoor gezorgd dat dit volledig random word bepaald. Doormiddel van een slot machine. Deze heeft gelukkig wel een hoge win kans en het is altijd prijs, maar geluk is nu een belangrijkere factor van deze game geworden.

Design Patterns:

Singleton – (Manager) Om het te beheren van de beurten en wat er in de game op dat moment voor actie gebeurt.
Factory Patern – Voor het instantiëren van Pokémon, Items en Aanvallen.
Observer – Als de speler op een van de knoppen klikt moet er een event plaatsvinden.
Fine State Machine – Voor het battle systeem en de verschillende staten waarin het spel zich moet bevinden.

Scripts:

Battle_Manager – Waar de fases van het battle system zich zullen bevinden
Menu_Manager – Waar de knoppen van het menu kunnen worden ingedrukt en er acties kunnen worden uitgevoerd.
Slot_Machine_Manager – Die de slotmachine elke beurt reset en voor beide partijen laat draaien en stoppen.
SM_Score_Counter – Die de slot machine zijn score telt op de 5 verschillende manieren.
Item_Manager – Managet de items die jij gedurende game hebt.
Item – Om de eigenschappen van de items in op te slaan.
Pokémon_Manager – Managed de Pokémons die jij gedurende game hebt
Pokémon – Managed de Pokémon, zijn aanvallen en health.
Score_Manager – Waar de score bij worden gehouden en de aantal gewonnen battles.

Waarom op deze manier:

Opzet:
Ik wou de verschillende scripts in deze game scheiden door het eerste woord van de script met het deel van de game te onderscheiden. Op die manier hou je overzicht van welke script voor welk is. Dit kon ik ook doen met een namespace, maar daarvoor moest ik eerst een script openen (en dat kost tijd. Developers zijn lui!).

Het design waar ik voor heb gekozen is simpel. Er zijn verschillende managers die de game zijn waardes regelt. Sinds ik de Slot Machine en de Battle Mechanic van Pokémon in mijn game moest verwerken heb ik moeten kiezen voor deze opzet. Al deze managers in verbinding met elkaar vormen een solide en modulaire opzet die je makkelijk kan aanpassen.

Patterns:
De patterns die ik had willen gebruiken waren voornamelijk singletons en state machines. Met state machines kon ik snel van states in de game verwisselen en met singletons kon ik bij belangrijke info in de game zoals de Pokémon Database. Dit was bij een game zoals Pokémon cruciaal. Dit zorgde ervoor dat ik snel tussen states kon switchen. De Observer is om snel acties uit te voeren in de game, door het een event systeem op te zetten. Het factory pattern spreekt voor zich, om base classes te maken en daar op door te bouwen.

Hoe het ontwerp in de loop van het proces is verandert:

Het ontwerp is in de loop van het proces heel erg verandert. Door enige tegenslagen en tijdgebrek heb ik de items uit de game moeten schrappen. Voor de rest heb ik mij aan het concept gehouden en gezorgd dat deze goed geïmplementeerd waren. Ook was het aantal scripts veel meer geworden, dan gepland. Ik heb ook gebruikt gemaakt van elementen van een MVC. Dit was handig om te implementeren en snel objecten van de Pokémon aan te maken. Deze data kon je snel inladen en doormiddel van een Base Class snel acties mee aanroepen.

Ook door problemen met de singletons heb ik de restart functie en dat er Pokémon bij de trainer komt elke keer dat je het speelt niet kunnen implementeren. Door missende objecten en een naderende deadline.

Wat zou je nog willen toevoegen?:

Ik zou sowieso de Items die ik nog voor ogen had in de game willen toevoegen. Na wat testen zou ik ook wat meer van de speler zijn invloed willen toevoegen. Op die manier is het wat minder RNG en kan de speler ook echt winnen. Ook zou ik de AI wat slimmer willen maken. De Types die de Pokémon hebben zijn een interessant ding om in het concept te verwerken. Dit zorgt dat aanvallen van de ene Pokémon sterker zijn dan die van een ander. Ook lijken enige status effects die je aan de speler kan geven heel gaaf, waardoor misschien een van de wielen van de slot machine niet werken.

Wat ga je de volgende keer anders aanpakken?:

Wat ik bij elk concept heb is dat ik denk dat het er heel simpel uitziet, maar dat het toch complex lijkt. Bij dit concept begonnen we in week 4, waardoor er weinig tijd was om dit spel te optimaliseren. Dit had ik beter moeten in plannen en ik had een kleiner concept moeten kiezen. Naast dit wil ik ook zorgen dat ik mij meer aan de patterns en opzet hou. Dit had ik nu redelijk, maar er is nog altijd ruimte voor verbetering.

Ook zou ik de tegenstander wat beter willen visualiseren en programmeren. Door gebrek aan tijd en problemen met de singleton heb ik het niet voor elkaar gekregen om het spel te restarten. Deze functie heb ik er nog uit moeten halen en dit zorgde ook dat ik mij niet aan mijn Scene Manager kon houden.

Waar ik niet aan toegekomen ben:

De volgende dingen zijn waar ik niet aan toegekomen ben:

- Het maken van een restart-functie die een extra Pokémon toevoegde aan de enemy
- Het werken van random items die je kon gebruiken tijdens je battle
- Het werkend maken van de types om aanvallen nog wat effectiever te maken
- Het opruimen van mijn gehele code

Conclusie:

Het concept is naar mijn mening goed uitgewerkt. Ik heb deze vier weken volgens de patterns geprogrammeerd en een hele andere manier van denken gehad. Het anders denken en het netjes programmeren maakte het voor mij moeilijk in te schatten of ik het zou halen. Dit is gelukkig (met enige weglaten van elementen) prima gelukt.

Ik vond het een leuk project alleen had ik wel wat eerder willen beginnen met het bouwen van een game. Dit had misschien tot een nog leukere uitwerking kunnen komen!