Naam: Yassine El Abdellati

Studenten nummer: 2021807

Project Doodle Jump:

Dit is mijn report voor het project Doodle Jump (Advanced Programming). Er waren 7 key classen (Game, Stopwatch, World, Camera, Score, Random, BG Tile) ik heb ze daarom ook allemaal geïmplementeert. Ik heb ook alle design patterns geïmplementeert. Bij mijn Game klasse maak ik mijn window, statemanager en starting state, ik run ook de program daar. Mijn stateManager managed alle states, de eerste state is de menu state dit is de state waar je op play en exit kan drukken en je kan daar de highscores zien. Ik kan states adden en removen van de stack, ik kan weten of er nieuwe states geadd of geremoved moet worden via de functie stateChanges. De current state is dus altijd op de top van de stack, dit is handig voor bijvoorbeeld wanneer jij een level finished en dan naar de volgende level moet gaan. Mijn levelState is vrij dynamisch, je hebt een infinite wereld met verschillende platformen en powerups. De state wordt geïnitializeerd door de init functie, de functie inputController wordt daarna opgeroept and die kijkt naar de input van de user en je kan dan zo zien of de menuSelector naar links/rechts of boven/onder gaat, en bij enter wordt er een nieuwe state toegevoegd(hangt er vanaf op wat je enter gebruikt). bij de menuState en levelMenuState roept de update functie de draw functie op maar bij de levelState maakt die de world aan. Ik gebruik ook een window klasse die inherit van de windowV klasse zodat mijn world die window kan geburike. Mijn world heeft een update funtie die de player, BGtile, bonus, platform update, die checkt ook voor collisions. De camera gaat mee naar boven vanaf dat de player een bepaalde height heeft bereikt. Ik heb ook een score die geupdate wordt wanneer je een bonus gebruikt en wanneer je omhoog gaat. Ik maak een player, platform, bonus(jetpack, spring), bg tile, aan de hand van de factory. Ik heb een abstract factory die virtual is. Ik maak een shared pointer van een entityModel en ook van de entityView en ik add die aan de observer van de entityModel. Wanneer ik een platform aanmaak zet ik die in een vector voor mijn platforms die mijn world aanmaakt ik doe dit ook met mijn bonussen, omdat er potentieel meerdere zijn in een world en ik moet dan ook al hun collisions kunnen checken en ik kan dan ook alle observers updaten. Mijn entityModels hebben ook een subject die de observer als een shared pointer bijhoud. Die heeft ook een update functie die de observers update en dat zorgt ervoor dat de entities op de screen gedrawt worden. Ik heb ook een malus geimplement, hoe hoger je komt hoe trager je horizontaal en verticaal beweegt er is wel een limiet zodat je door kan spelen. Volgens de MVC pattern heeft elke entity een model en een view,mijn LogicLibrary directory bevat alleen maar logic en de GameRepresentation bevat sfml, de model bevat de logic van de entity en de view bevat hoe dat die eruit ziet. De entityView heeft een shared pointer van de entityModel zodat die weet waar dat die moet tekenen na elke observer update. De stopwatch is een singleton die het verschil gaat berekenen tussen alle frames, ik ga deze waarde gebruike wanneer mijn player beweegt zodat mijn player overal op elke monitor even snel beweegt.