Space Invaders

Structuur

Observer en subject

Observer en Subject staan op het hoogste niveau in mijn project, alle views en models erven hiervan over. Deze 2 klassen werken in een observer pattern. subject heeft een lijst met observers, subject heeft ook een functie notifyobservers en wanneer deze aangeroepen wordt worden alle observer geupdated. Dit is handig voor wanneer bv de positie van een entitymodel wordt aangepast deze dan notifyobservers oproept, waardoor de updatefunctie van entityview dan wordt aangeroepen die dan de drawable op de juiste positie plaats volgens de correspondente locatie op het scherm.Deze correspondente locatie wordt berekent met de transformklasse wat een singleton is.

Modelabstract,viewabstract en controllerabstract

Modelabstract erft van subject en viewabstract erft van observer Controllerabstract is de interface hiertussen.Viewabstract doet alles omtrent het grafische deel van het object. Het heeft een sf::drawable shape wat met de drawfunctie op het scherm(wat ook een variable pointer is van ViewAbstract) wordt getekend. controllerabstract heeft een pure virtual functie tick die elke keer wordt aangeroepen bij de volgende tick van de game. bv de playershipcontroller moet elke tick input lezen en dan bv wanneer er rechts is gedrukt het model van het playership naar rechts verplaatsen(en als het model naar rechts wordt verplaatst roept het notifyobservers wat dan de shape van view verplaatst). In modelabstract staat niets het is gewoon een klasse dat gemaakt is om de structuur op orde te houden.
Door deze hele structuur te gebruiken wordt het logische deel(model) gesplitst van het visuele(view).

ObjectManager

Objectmanager is een singleton klasse die wordt gebruikt om alles omtrent objecten te beheren. Het heeft functies zoals create playership dat dan een playership aanmaakt met bijbehorende controller model en view. Objectmanager heeft een vector van objecten(Een object is een struct met als variabele controllerabstract viewabstract en modelabstract). Deze vector gaat game gebruiken for tijdens elke tick over alle objecten te loopen en controller[i].tick() te doen en bij elke view view[i].draw te doen.

Game

De gameklasse maakt alles klaar voor het spel. Het creëert de speler en de shields nodig voor het spel te spelen. met de constructor moet er een jsonfile meegegeven worden. Deze jsonfile heeft een array met alle locaties waar de json level files staan. In een json level file staat een array van aliens met bijhorende parameters die dan ingelezen kunnen worden met loadlevel. In de gameklasse staat ook een parser die dit allemaal inleest en handled.

Entity

Deze klasse heeft ook weer een model view en controller en erft van de abstracte klasses. Hier heb ik een belangrijke beslissing moeten maken. In het begin van mijn project had update een parameter positie. maar later had ik een andere update nodig met parameter healthpoints. hierdoor kon ik geen proper modulair design creëren. Om dit op te lossen heb ik een pointer bijgehouden in Entityview naar een Entitymodel.
Wanneer update() dan wordt aangeroepen kan je dan makkelijk de nodige waarden van het model gebruiken. anders moest je bij elke notifyobservers nakijken van welk type de observer is en dan te casten naar dat type, vervolgens de bijhorende update(parameter) aanroepen, als je dit doet moet je ook enorm veel update functies in observer gaan aanmaken waarvan je view er misschien 1 nodig heeft. Dit leek mij enorm onhandig. In EntityModel houd ik position en size bij. Wanneer setposition wordt aangeroepen roept het natuurlijk ook notifyobservers aan.

Collidable

Dit is enkel een controller dat erft van entitycontroller. deze kan nakijken of het model van deze controller collided met het model van een andere controller. Het heeft ook een pure virtualfunctie oncollision wat dan later gedeclareerd wordt. Bv bullet collided met alien dan gaat bullet oncollision(aliencontroller) oproepen wat dan zegt wat er moet gebeuren.

Alive

Dit is een klasse voor levende objecten zoals een player of een alien. Hier is enkel een model van geïmplementeerd en erft van collidable. Deze bevat healthpoints firecooldown speed en functies moveleft, moveright.

Enemy

Deze klasse inherit van Alive.Deze klasse zal de movement voor de enemies regelen(zodat ze van links naar rechts gaan en soms naar beneden).

AlienShip, PlayerShip, Bullet en Shield

Dit zijn de laagste niveau klassen en zijn ook de objecten die door het spel zullen gaan. Playership zal kunnen schieten op alienships met bullets. De bullets kunnen

ook tegen een shield vliegen etc. Door het modulair design van mijn project kan je ook makkelijk nieuwe objecten toevoegen. stel je wilt een ander soort enemy toevoegen dan inherit je gewoon van enemy stel je de klasse op en maak je een bijhorende create functie aan in objectmanager. Ik heb er ook voor gekozen elk object van het laagste niveau een vaste sprite te geven zodat dit altijd hetzelfde is voor dat object en het niet als parameter moet meegeven worden. een playership zal bij het creëren altijd de sprite player.png inladen, een alienship dan alien.png enzoverder. Dit wordt gedaan met een generateshape functie die wordt aangeroepen in de constructor.