SPACE INVADERS 3D

Tim Peeters 3029407 | Kernmodule Game Development 1

19-09-2019

Github Repo:

https://github.com/TimPeeters1/KernGameDev1_TimPeeters

Concept en Twist

Voor mijn retro game met een twist heb ik gekozen voor Space Invaders. Ik heb dit spel gekozen omdat het me een interessante game leek om een twist te geven. Daarnaast leek het me ook een reële game om na te maken, mede door het feit dat het mechanics heeft die ik al eerder voor andere games heb gemaakt (zoals schiet mechanics).

De twist in het spel is eigenlijk heel simpel. Zoals de titel al zegt is de twist dat de game in 3d is. Ik heb gekozen voor deze 'twist' omdat ik op internet eigenlijk geen 3d versie van Space Invaders kon vinden die een beetje in de buurt kwam bij het idee dat ik had.

Door het feit dat het spel nu een 3d ervaring is, komen er eigenlijk vanzelf meer twists 'tevoorschijn'. Zo moet je nu bijvoorbeeld als speler het 'geweer' ook moet richten met de muis, iets dat voorheen niet kon omdat je alleen naar links en naar rechts kon bewegen over het scherm.
Naar mijn mening is door de simpele verandering van 2 naar 3d de game nu veel interessanter.

Code patterns

Voor het spel het ik verschillende design patterns gebruikt.

Deze zijn het singleton pattern, het observer pattern en het object pool pattern.

Singleton Pattern

Singletons zijn in het algemeen in alle belangrijke classes van het spel verwerkt. Ik heb de singletons verwerkt in de classes zoals Player, GameManager en ObjectPoolManager. Dit heb ik gedaan zodat er altijd maar 1 instance van desbetreffende class bestaat in de scene.

Het voordeel van dit singleton pattern is voor mij dat ik zonder een GetComponent<> te callen altijd de instance van een class snel kan vinden en kan aanroepen vanuit een andere class zonder veel moeite te hoeven doen. Ik vind dit vooral handig voor o.a. mijn GameManager, omdat deze veel centraal regelt. Ik wil niet telkens in elke class een reference naar mijn GameManager maken, en de singleton lost dit goed op.

De singleton implementatie die op het moment aanwezig is in het spel is niet helemaal safe omdat dit een static class is. Ik wilde dit nog veranderen maar had er de tijd niet voor (Instance.destroy kan bv. nog worden aangeroepen).

Observer Pattern

Het observer pattern is in de game geïmplementeerd door middel van interfaces die gebruikt worden door onder andere de player en de enemies om elkaar te damagen.

Bullets zijn in de game de ‘subjects’. De player en de enemy’s *observen* de IDamagable interface die de bullets gebruiken en reageren hierop.

Ik vond het belangrijk om dit pattern te implementeren zodat ik een unified systeem heb om damage te doen aan veel objecten zonder veel references naar verschillende classes te hoeven maken. Zo gebruik in nu bijvoorbeeld ook de IDamagable class om niet alleen players en enemy’s te damagen, maar ook om bijvoorbeeld destructable walls te kunnen damagen.

Object Pool Pattern

Het object pool pattern is eigenlijk het ‘grootste’ pattern dat ik heb geïmplementeerd in het spel.

Aan het begin van het project realiseerde ik me dat ik met veel objecten in een scene ging werken. Ik moest bullets, enemy’s en particles gaan spawnen. Dit is vrij veel om allemaal te instantiaten, dus daarom ben ik me gaan focussen op performance.

Om te voorkomen dat er veel moet worden gespawned tijdens runtime heb ik gekozen voor een objectpool systeem, die alle nodige objecten spawned bij het begin van het spel, en deze objecten aan en uitzet wanneer ze nodig zijn.

Dit is beter voor performance omdat objecten nu alleen aan en uitgezet hoeven te worden, in plaats van gedestroyed en opnieuw geinstansiated te worden. Vooral voor de bullets die de enemy’s en de player gebruiken is dit heel handig, omdat er hier veel tegelijk van in de scene kunnen bevinden.

Performancewise heeft dit dus een best grote impact.

Code structuur

Hier volgt een kleine documentatie van wat de verschillende classes in mijn game doen. Ik zal niet relevante en niet gebruikte classes buiten beschouwing houden.

GameManager

De GameManger is eigenlijk het backbone van de game, het houdt bij hoeveel enemy’s er nog over zijn en stuurt naar de spawner als er nieuwe enemy’s moeten worden gespawned. Ook houdt het de scores bij die de player haalt en is het verantwoordelijk voor de GameOver als de speler het spel verliest.

ObjectPoolManager

De objectpool manager is de manager die verantwoordelijk voor het aanmaken en het spawnen van objecten uit de verschillende pools die het beheert. De objectpool manager instantiate via scriptable objects nieuwe pools en beheert deze. Andere

classes met een reference naar de manager kunnen functies aanroepen op de manager om zo objecten uit de pool te halen.

De objectpool manager gebruikt de interface IPoolObject die de functies OnObjectsSpawn en OnObjectDespawn heeft, die als het ware de OnEnable en OnDisable zijn van de objectpool. Door deze interface te gebruiken kunnen gepoolde objecten aan en uitgezet worden door andere classes.

Player

De player class is de class die alle user input handled en hiermee omgaat. In deze class beweegt de speler naar links en rechts d.m.v. respectievelijk A en D in te drukken. Ook kan de speler richten met de ‘turret’ door middel van de muis en kan hij/zij schieten door de linkermuisknop in te drukken.

In deze class wordt dus movement van de camera en het ‘turret’ object geregeld samen met het schieten van bullets. Daarbij bevat de player class ook nog health en UI variabelen die de levens bijhouden en updaten naar de UI van de game.

De player implementeerd ook de IDamagable interface die een Damage en Die functie bevat waardoor de bullets die de enemy’s gebruiken de player kunnen damagen.

PlayerAnimator

De PlayerAnimator class is een class die ervoor zorgt dat het 3d model van de ‘turret’ meebeweegt met de camera en de crosshair. Het gebruikt vectoren en quaternions om de huidige forward van de camera te berekenen en hierna zijn rotatie naar die forward aan te passen. Dit werkt zowel met horizontale als verticale rotatie.

Enemy

De enemy class is de hoofdclass voor de enemy’s/vijanden die de speler aanvallen tijdens het spel.

De enemy class bevat de functies die ervoor zorgen dat de enemy’s naar de speler bewegen en ook van links naar rechts bewegen. Daarnaast heeft het ook functionaliteit om naar de speler te schieten.

Ook de enemy bevat de IDamagable interface om damage te ontvangen op dezelfde manier als de speler. Wat echter anders is in vergelijking met de Player is dat de enemy ook een IPoolObject interface bevat, zodat de enemy’s ook gepooled kunnen worden.

Ik wilde nog meer functionaliteit toevoegen aan de enemy’s door de enemy class abstract te maken en verschillende soorten enemy’s te maken. Daarnaast wilde ik ook nog zorgen dat de enemy’s verder over het scherm bewegen als er minder aanwezig zijn in het spel. Ik had echter geen tijd meer om dit toe te voegen.

Spawner

De Spawner class bevat de functionaliteit om nieuwe rijen met enemy’s te spawnen over de lengte van de boxcollider die de spawner bevat.

De Spawner bevat functies om nieuwe rijen te spawnen met de functie Spawn() die de GameManager gebruikt. Daarnaast heeft het ook de mogelijkheid om meer rijen en verschillende poolable objects te spawnen. Het is mogelijk om meerder spawners in een level te hebben.

Bullet

De bullet class is de hoofdclass van alle varianten bullets die de game gebruikt. Alle classes die de bullet raakt met een IDamagable worden gedamaged.

In dit script wordt ook de bulletDamage meegegeven vanuit de player. Het is in principe een container van data die damage doet op alle IDamagable classes/objecten.

Ook hier wilde ik nog een abstract class maken, maar dat is niet gelukt

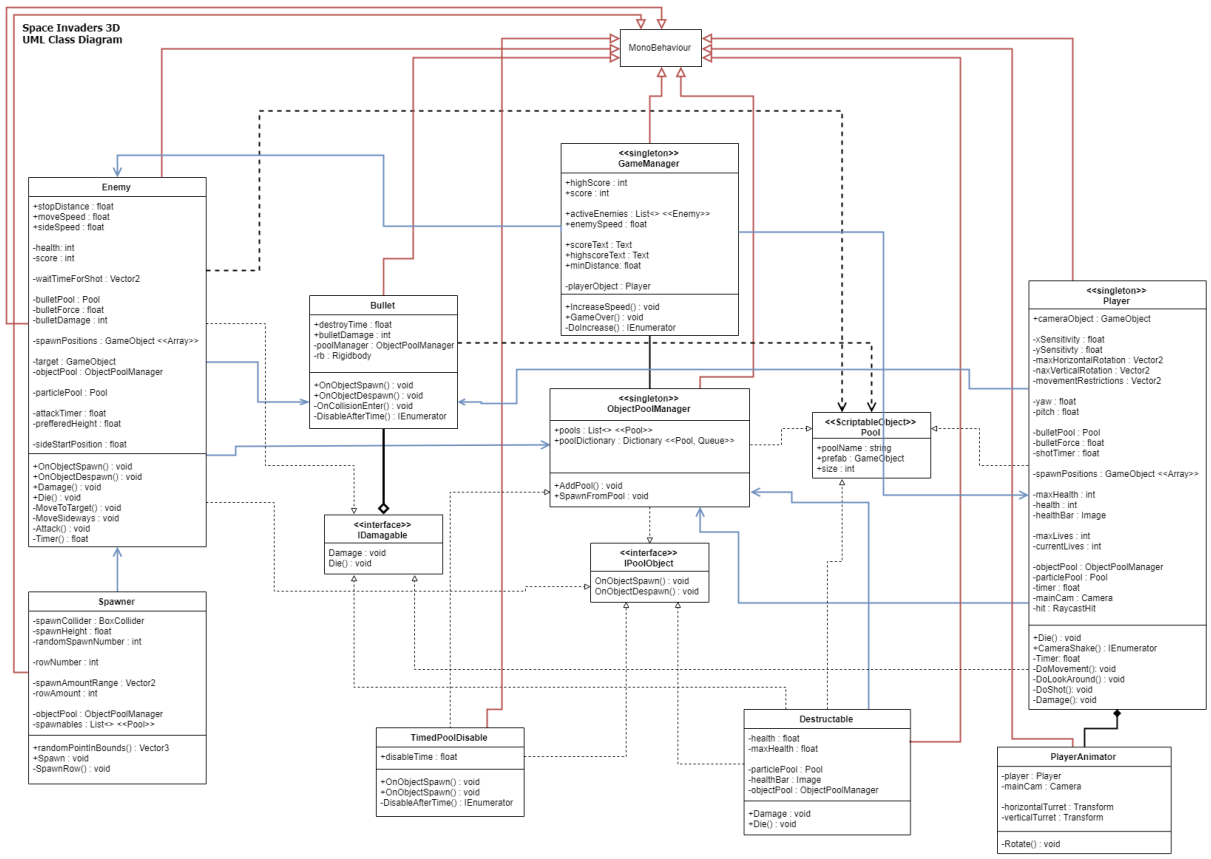
Destructable

De destructable class is simpelweg een class die gedamaged kan worden door de IDamagable interface, en hierdoor ook gedestroyed kan worden. Ik gebruik het in dit geval voor de destructable forceshields in het spel.

TimedPoolDisable

De TimedPoolDisable class zorgt ervoor dat alle objecten met deze class die gepooled worden (IPoolObject), na een bepaalde tijd weer worden uitgeschakeld. Ik gebruik dit op het moment voor gepoolde explosie particles, die na een paar seconden weer worden uitgezet als ze hebben afgespeeld.

UML Class Diagram



UML Activity Diagram