Retro Video game

A diffirent view on Tetris

Tetris: Skyscraper **By Max Kersten**

Het concept

Ik heb de core mechanic van Tetris genomen en behoorlijke twist aan het hele spel gegeven. De core mechanic is het zelfde, je draait blokjes rond en plaatst ze op de juiste locatie.

Twist 1; 3D
Het spel zal zich in een 3D wereld afspelen. Het blokje kan dus om 3 assen draaien. en 3 assen bewegen.
Uitdaging: Om met veel toetsen te werken.

Twist 2: Physics
De blokjes zullen worden beinvloed door zwaartekracht en dus om kunnen vallen.
Uitdaging: Phisics is nog best moeilijk. Ook om collisions goed af te stellen.

Twist 3: TowerBuilder
Je moet een zo'n hoog mogelijke toren bouwen zonder blokjes er van af te gooien. Vallen te veel blokjes om, dan is het spel afgelopen. Als je een laag helemaal vult, zal deze laag veranderen in een verdieping van een gebouw. Hierdoor scoor je punten!
Uitdaging: Hoe bepaal ik welk blokje in welk stuk veranderd van het gebouw?

Plan van aanpak

Engine: Unity3D
Features: Physics, Monobehaviour

Code style:
Singleton en Listiner pattern

Verantwoording

Listener-Pattern:
Ik wil listeners gaan gebruiken om er voor te zorgen dat niet alles met alles in verbinding staat. Als dit wel het geval is dan kan het zijn dat je behoorlijk spaghetti-code ontwikkelt en je heel makkelijk de draad kwijt raakt tijdens de ontwikkelfase.

Singleton-Pattern:
Ik wil singletons maken om er voor te zorgen dat er ten alle tijden, van bepaalde classe/objecten, er maar 1 van kan bestaan. Denk aan de BlockQueue, daar heb ik maar 1 van nodig. Met meer dan 1 van deze classes zal alles stuk gaan. Ditzelfde geldt voor de Tower en PlayerController. Ook heb ik dit gedaan zodat ik geen directe pointers naar dit soort objecten hoef te maken. Ik call gewoon de instance van het object. En als deze niet bestaat, wordt deze aangemaakt en is er niks aan de hand.

Het Design

Classes The Basics

InitManager:
In de gamescene moet ten alle tijden een “EnitManager” bestaan.
Deze manager heeft alle pointers die de andere singletons gebruiken.
Als deze manager er niet is, kunnen de overige singletons niet hun werk doen en gaan ze errors gooien. Dit is niet het geval als een van de andere singletons niet aanwezig is. Dan wordt deze missende singleton netjes aangemaakt met de bijbehorende gameobjecten.

PlayerController:
De speler stuurt het PlayerController class aan dmv het keyboard.
WASD voor beweging en IJKL en UO voor rotaties. Deze class stuurt alles aan kwa bewegingen. Deze class vraagt ook aan het Tetrisblock object of deze kan bewegen zoals de speler wilt. Verder haalt deze class zelf nieuwe blokken op vanuit de “BlockQueue”. In deze class zitten ook een paar IEnumerator. Deze zorgen er voor dat de bewegingen pas worden uitgevoerd nadat de daadwerkelijke check voor de movement/rotation heeft plaatsgevonden.

BlockQueue:
De BlockQueue is verantwoordelijk voor het aanmaken van nieuwe tetrisblokken.
De queue heeft plek voor 3 blokken. Is er een plek leeg, dan wordt er een nieuw TetrisBlock geinstantieerd.

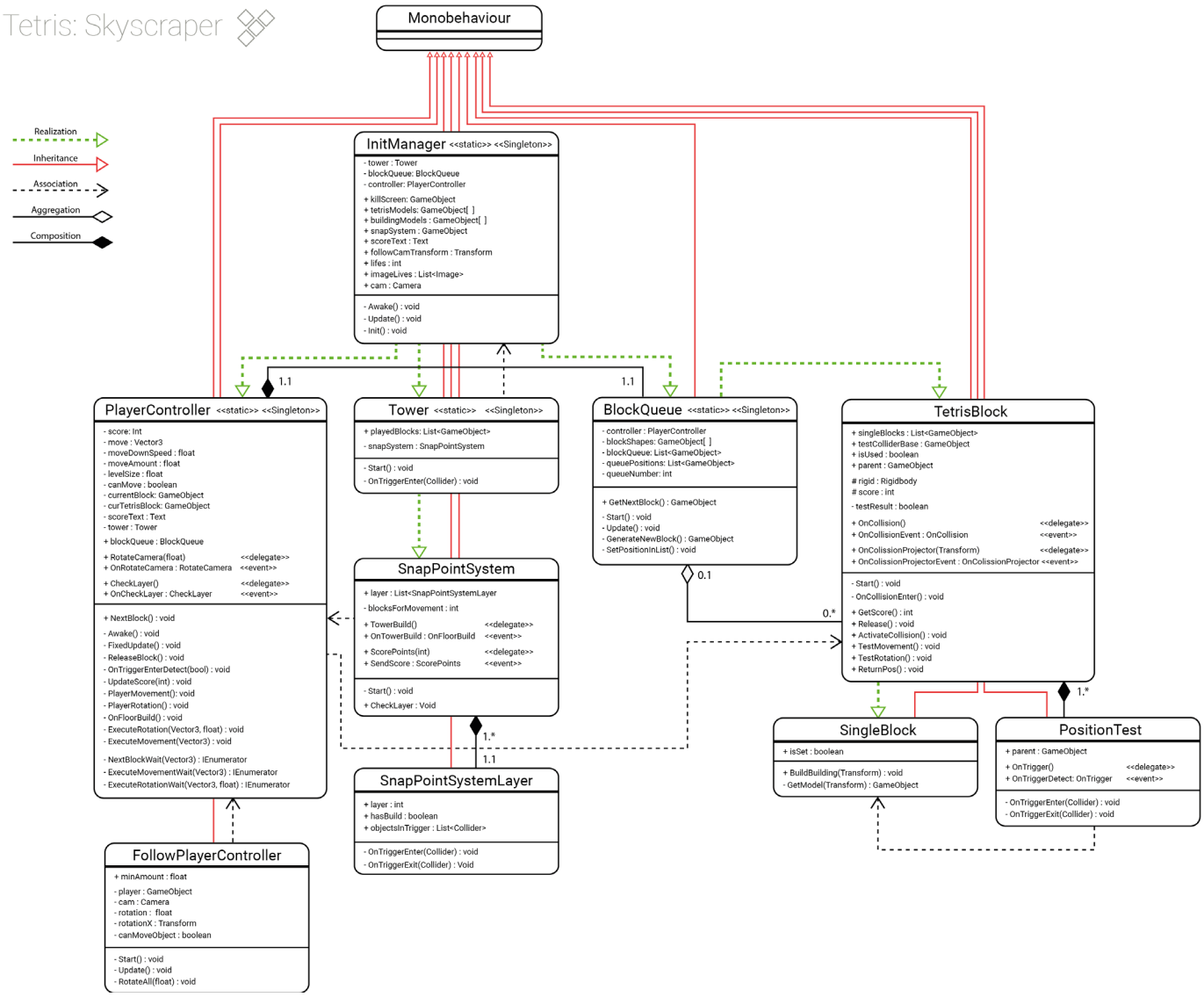
TetrisBlock:
Het Tetrisblock voert zelf een aantal functies uit. Hij detecteert of hij de vloer of een ander TetrisBlock raakt. Zo ja, gooit hij een event de lucht in. Deze wordt ontvangen door de PlayerController. Die dan dit blok loslaat en een nieuwe ophaalt.
Verder vraagt deze een positietest aan als de PlayerController hier om vraagt.
Deze werkt als volgt. Het TetrisBlock pakt het gameObject “testColliderBase” en verplaatst deze naar de te testen positie. Raakt een collider een ander object, gooit de collider zelf een event via de PositionClass.
Ook dit event wordt aangevangen door de PlayerController die dan niet de beweging zal uitvoeren.

Tower:
Deze class bouwt het snapSytem gameObject met bijbehorende classes.
Deze class detecteert of er een blok is omgevallen. Deze roept dan de emitManager instance aan en haalt er een leven af.

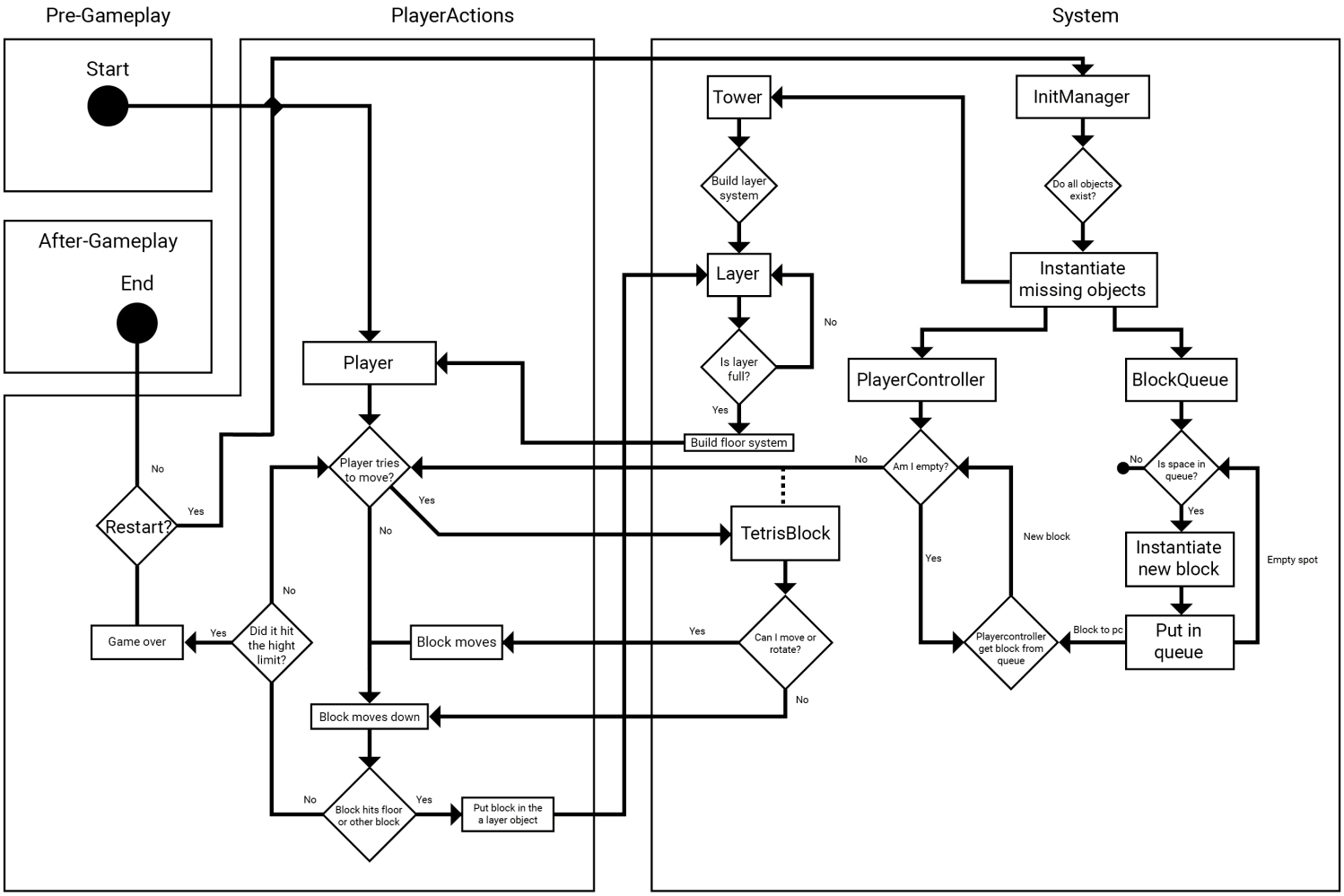
SnapPointSytem en SnapPointSystemLayer:
De naam van deze class komt niet overeen met de functie.
Het SnapPointSystem class loopt door alle SnapPointSystemLayer classes heen na een event van uit de PlayerController. Als er in een SnapPointSystemLayer class meer dan een bepaald aantal childs wordt gevonden zal deSnapPointSytem aan alle Transforms met een SingleBlock-class vragen om zichzelf om te bouwen. Als dit gebeurt wordt er een event gegooit en zal de PlayerController hier op reageren.

SingleBlock:
Deze class is verantwoordelijk voor het “Bouwen” van het gebouw.
Deze wordt aangesproken door het SnapPointSystem. Deze geeft een “Go” voor het instantiëren van het gebouw.

Class Diagram



Flow Chart


Pre-Gameplay
PlayerActions
System
Start
After-Gameplay
End
Restart?
No
Yes
Player
Player tries to move?
Yes
No
Did it hit the hight limit?
No
Yes
Game over
Block moves
Block moves down
Block hits floor or other block
No
Yes
Put block in the a layer object
Tower
Build layer system
Layer
Is layer full?
No
Yes
Build floor system
TetrisBlock
Can I move or rotate?
Yes
No
InitManager
Do all objects exist?
Instantiate missing objects
PlayerController
Am I empty?
No
Yes
New block
Playercontroller get block from queue
Block to pc
BlockQueue
Is space in queue?
No
Yes
Instantiate new block
Put in queue
Empty spot