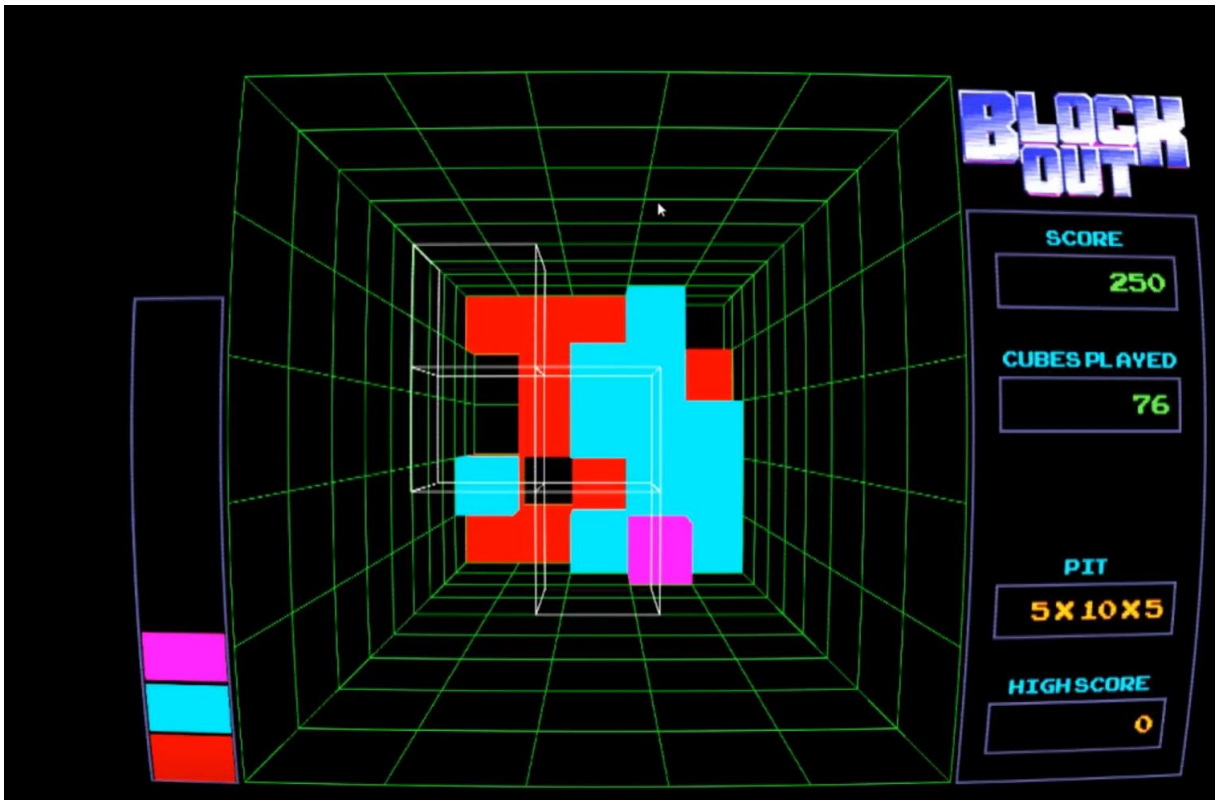
Technisch Document Tetris 3D(BlockOut)

Jelle Dekkers
GDEV2
3019367



CONTENTS

INLEIDING

Dit is het technisch document voor Tetris3D(ook bekend als Block Out) gemaakt door Jelle Dekkers in opdracht van KernModule Game Development Jaar 2.

Zie de bijlage voor grotere plaatjes en het Unity Project.

HET CONCEPT

Het spel is in feite een 3D Tetris gezien vanaf de 'top' perspectief. Groepen van blokken vallen na elkaar één voor één naar beneden en kunnen in alle richtingen geroteerd worden. Vallende blokken hebben een witte outline en zijn transparant. Blokken die niet meer in beweging zijn, oftewel vastzitten, zijn niet meer transparant en krijgen een solide kleur. Deze kleur is per laag bepaald en naast het speelscherm staat een interface element met alle lagen en kleuren boven elkaar. Op deze manier is goed vast te stellen op welke laag een blokje zit en is het probleem van diepte hiermee verholpen.

Bij een volledig gevulde laag worden alle blokken in deze laag verwijderd en krijg de speler een aantal punten. Ook worden alle blokken boven deze laag, 1 laag naar beneden geschoven.

Bij game over wordt de score, indien van toepassing, opgeslagen in een highscore lijst. Deze lijst is dan in het menu te zien.

WAAROM

Ik heb voor dit concept gekozen omdat het mij een interessante uitdaging leek. Met name het maken van een grid waar alle blokken in opgeslagen worden en deze correct kunnen legen en verplaatsen leek mij interessant.

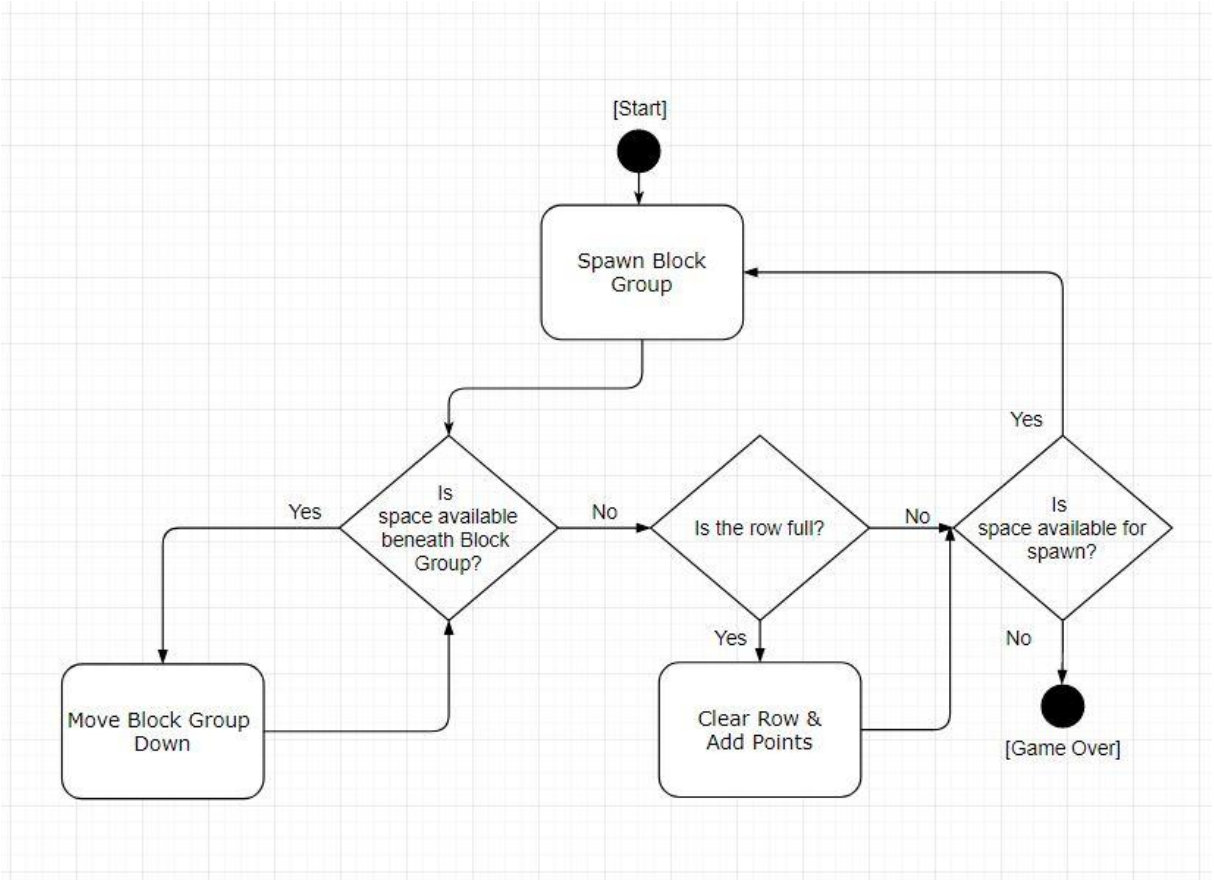
DESIGN PATTERNS

De gebruikte design patterns:

- Singleton. Ik heb 2 singletons gebruikts, bij GameManager en Level. Beide zijn maar 1 instantie nodig en moeten in de scene staan omdat ze gebruik maken van prefabs.
- Observer Pattern. Ik maak gebruik van meerdere events die, in combinatie met singleton, makkelijk aan te subscriben zijn vanaf andere classes. Bijvoorbeeld bij het legen van een laag, het plaatsen van een blokje en bij game over. Op deze manier heb ik bijvoorbeeld audio effecten vanaf hun eigen script uit kunnen voeren en hiermee voorkom ik spaghetticode.

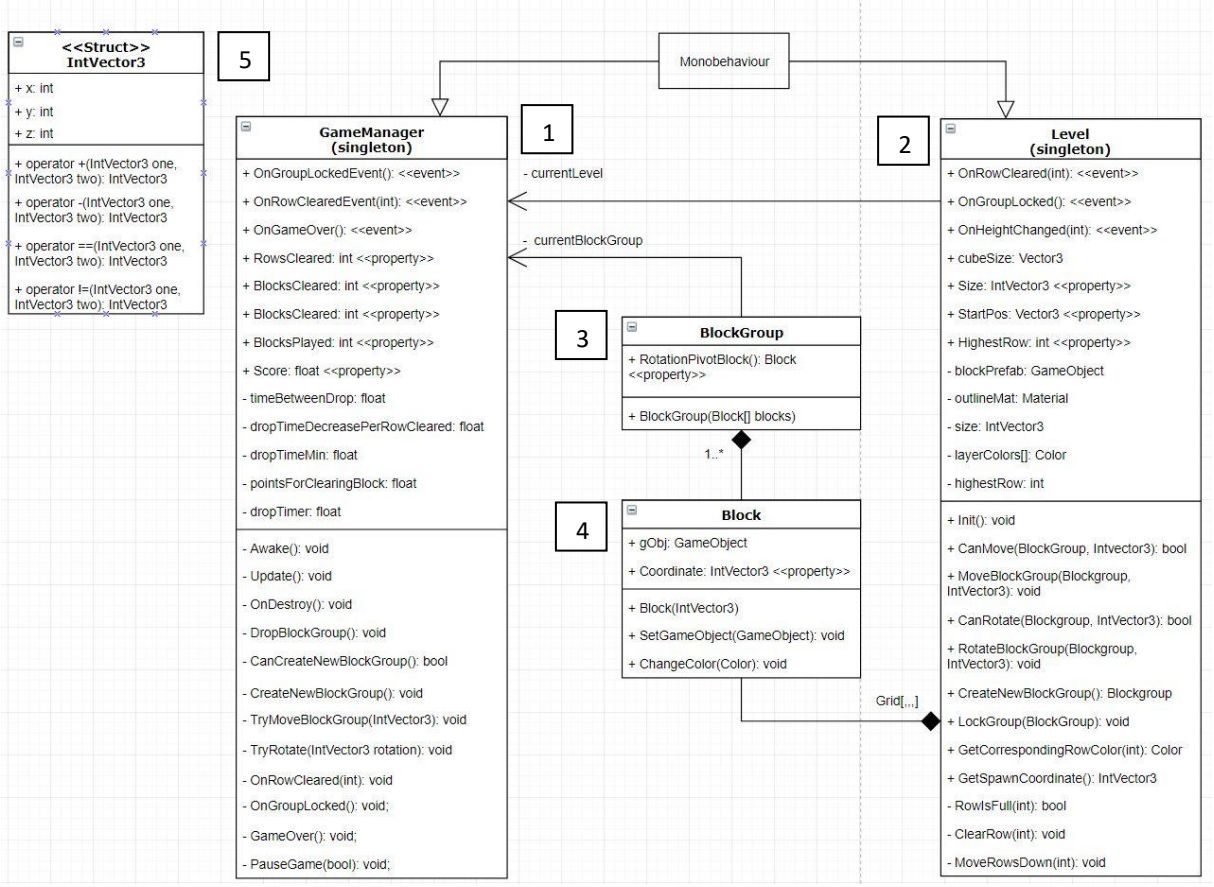
ACTIVITY DIAGRAM

Hieronder staat de activity diagram voor het verloop tijdens het spelen.



CODE STRUCTURE

Hieronder staat een groffe uitleg van de belangrijkste code structuur met extra uitleg.



1. **GameManager** bevat alle logica die belangrijk zijn voor het de gameplay, zoals de score en de player input. GameManager heeft een currentBlockGroup variabele van type BlockGroup waarin de huidig bestuurbare blokken in staan. Ook bevat GameManager een private variabele voor currentLevel van type Level. GameManager maakt gebruikt van de singleton pattern. Er is er maar 1 van maar heeft wel een instance nodig in de scene.
2. **Level** bevat een driedimensionaal Grid variabele van type Block voor het bijhouden van alle blokken en bevat alle code om te checken of het bewegen of roteren van blokken mogelijk is. Ook staan hier de benodigde events voor het clearen van een row, locken van een groep en bij het veranderen van de hoogte.
3. **BlockGroup**. Een handige class voor het bijhouden van alle blokken die bestuurbaar zijn in de GameManager.
4. **Blocks** is de class die ik gebruik voor alle blokken. Blocks inheriten niet van Monobehavour maar hebben in plaats daarvan een referentie naar een GameObject: het blokje in de scene. Ook hebben blokken een coordinate variabele die bijhoudt op welke coördinaat in het grid ze zitten.
5. **IntVector3**. Alle coördinaten in het grid zijn van type IntVector3. Dit is een struct bestaande uit een x, y en z van type int. Op deze manier kan makkelijk coördinaten worden berekend zonder enige rounding errors. Bijvoorbeeld: (0, 0, 1) + IntVector3.up = (0, 1, 1). Dit kan dan weer makkelijk in het grid van Level worden ingevuld.

PROCES

Er zijn niet veel dingen die erg anders gegaan zijn dan ik van te voren had gepland. Ik heb veel na moeten denken over hoe ik bepaalde dingen zoals de GameManager en Level wou opzetten. Ik wou het zo 'retro' mogelijk houden. Wat ik hiermee bedoel is dat ik zo min mogelijk Unity functionaliteit wou gebruiken en zo veel mogelijk met pure scripts wou werken, zodat de logica ook op andere engines zou werken. Dit was soms nog best lastig omdat je soms gewoon toegang tot prefabs of andere referenties nodig hebt. Bij nader inzien had ik bijvoorbeeld de Block class iets handiger kunnen maken door deze bijvoorbeeld wel van MonoBehaviour over te laten erven.

VERANDERINGEN VOOR TOEKOMST

Ik ben best blij met het eindproduct en er zijn niet veel dingen die ik zou veranderen. Ik had nog wel graag een 'animatie' gehad bij het verplaatsen en roteren, i.p.v. dat hij ze de blokken direct verplaatst. Een online highscore zou ook leuk zijn maar misschien iets te groot voor dit project.

Ik denk dat ik voor de volgende keer het niet veel anders aan zou pakken.

BIJLAGE

Github:
https://github.com/JelleDekkers/Tetris3D

Unity Project:
https://github.com/JelleDekkers/Tetris3D/tree/master/Tetris3D%20Unity

UML:
https://github.com/JelleDekkers/Tetris3D/blob/master/Opdrachten/UML_Final.JPG

Activity Diagram:
https://github.com/JelleDekkers/Tetris3D/blob/master/Opdrachten/Tetris%20Activity%20Diagram_Final.JPG

Filmpje met uitleg:
https://www.youtube.com/watch?v=-T-tnnQZZ2s&t=56s