Sten Duindam
KGDEV 2
Tools

BlockEditor & GameEditor

Inhoudsopgave

Inleiding

Tijdens blok twee van de kernmodule Game Development ben ik uitgedaagd om een tool te schrijven. Gelijk aan dit project liep het HybridSpace project en het leek mij leuk om hier een koppeling in te maken. Iets wat echt gemaakt zou worden dus. Na overleg met de designers van het project HybridSpace heb ik besloten twee tools te maken die het balancen en het creëren van content eenvoudig en toegankelijk maakte voor de designers.

Concept

In het spel Wreck it Richard!(HybridSpace) Wordt de speler belaagd door blokken met speciale attributen. Zo zijn er bijvoorbeeld explosieve, slijm en water blokken. De speler moet de blokken weten weg te slaan met een fysieke hamer tegen een interactieve muur waardoor de blokken geraakt worden in de virtuele omgeving.

Voor dit spel heb ik twee tools ontworpen en uitgewerkt.

Tool 1: BlockEditor

De belangrijkste game mechanic in het spel zijn de blokken. Om deze eenvoudig te maken en op te slaan heb ik een BlockEditor gemaakt.

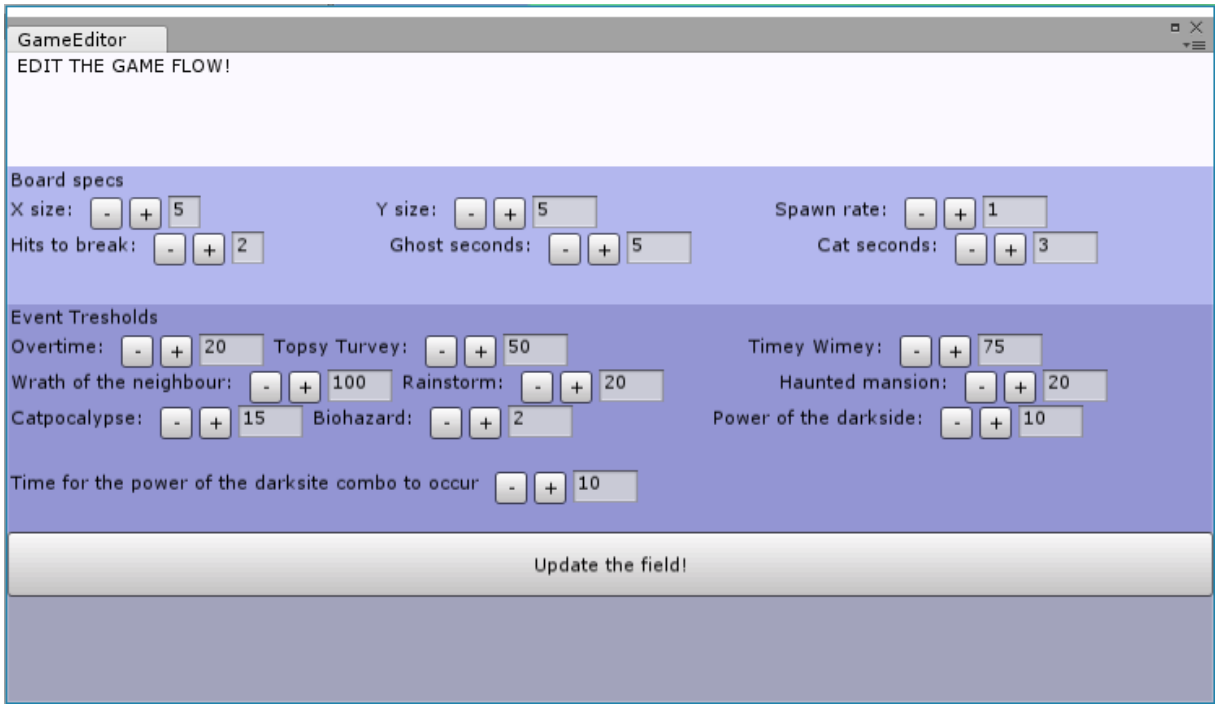
De BlockEditor, zoals hierboven te zien, laat de gebruiker attributen selecteren en aanpassen om vervolgens op te slaan en in het spel aan te maken. De tool slaat de data op in de volgende vorm:

Het spel leest vervolgens de XML uit en maakt de blokken aan.

Ik heb gekozen voor een editor window omdat deze eenvoudig te vinden is, werkt terwijl de game zowel actief als inactief is en toe te voegen is aan de editor layout. Dit om zo weinig mogelijk verwarring bij de designer te creëren en WYSIWYG te gebruiken is. Hieronder een voorbeeld van een krat dat gemaakt kan worden. De blokken worden opgeslagen in een lijst en gebruikt door het spel.

Tool 2: Game Editor

De blokken bevatten verschillende eigenschappen. Iedere eigenschap heeft eigen waarden. Zo is er bijvoorbeeld een blok dat meerdere keren geraakt moet worden voordat hij kapot gaat. Dit soort waarden zijn allemaal aan te passen in de GameEditor. Naast blokken zijn er ook events in het spel. Een event komt voor wanneer de speler voldoet aan de voorwaarde. Deze voorwaarden, duraties en eigenschappen zijn ook aan te passen in de GameEditor. Met een simpele knop pas je vervolgens het spel aan.



Naast de game flow is ook het bord zelf aan te passen. O.a. de X en Y waarden. Dit alles is net als de BlockEditor zowel tijdens(on runtime) als buiten het spel aan te passen.

Design proces

Beide tools zijn ontworpen met de gedachtegang dat iedereen die een klein beetje kennis van Unity had er mee kon werken. In dit geval heb ik kort maar duidelijke teksten gebruikt, zo is er geen verwarring wat alles betekend of waar het voor staat. Daarnaast heb ik besloten met knoppen te werken om de variabele te beïnvloeden, dit zorgt er voor dat er geen vreemde input geleverd kan worden. Ook zijn alle variabele met een clamp begrenst. Mocht de waarde een bepaalde maximale waarde overstijgen wordt hij naar de maximale waarde terug gezet.

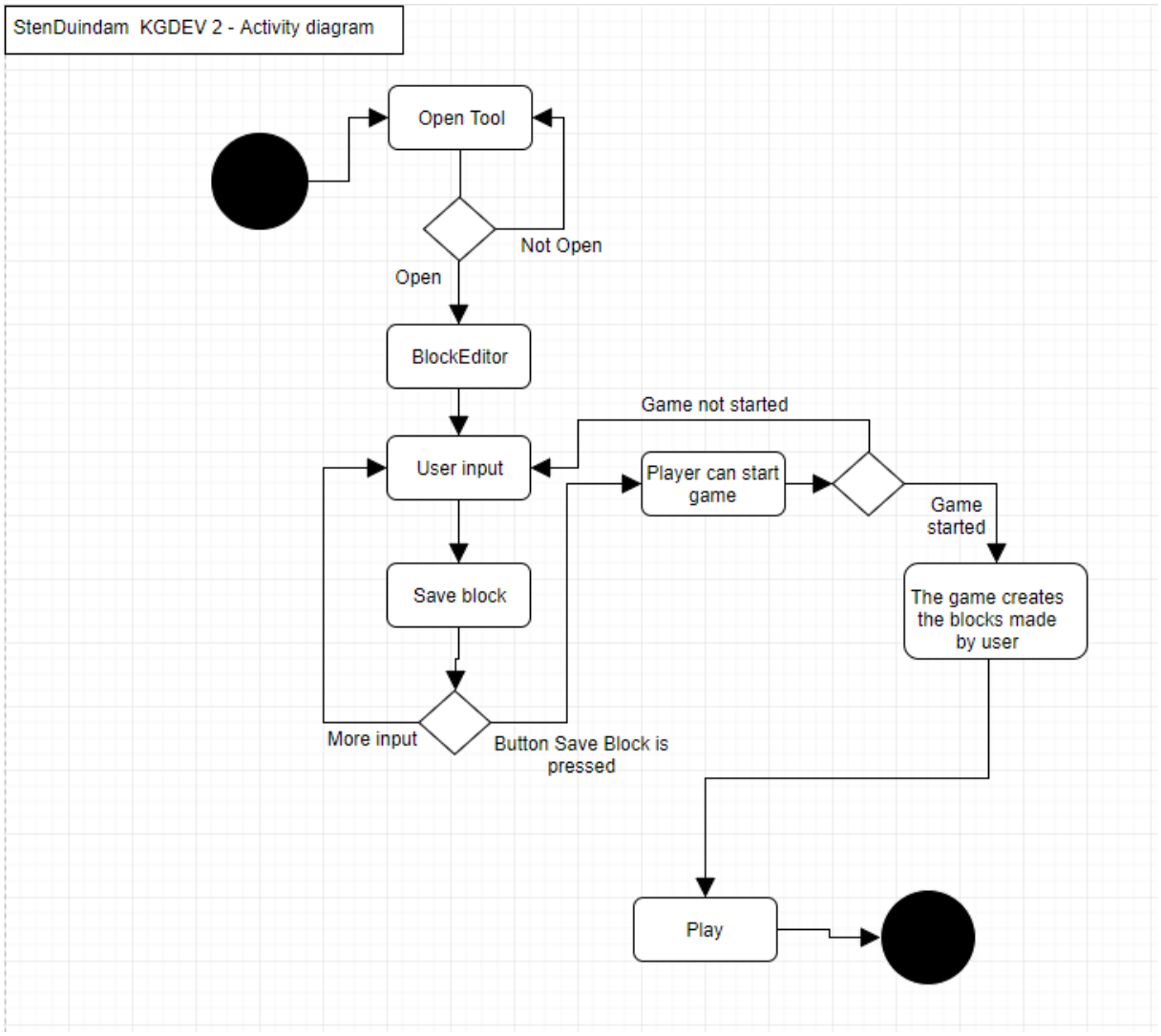
Een tweede keuze die ik heb gemaakt is om ieder 'blok ' een eigen kleur te geven en een titel. Hierdoor is het goed leesbaar, overzichtelijk en zijn er secties gevormd.

De derde bewuste keuze is zorgen dat bij de BlockEditor maar één veld een string is. Hierdoor kan de gebruiker met maar één keer zijn hand van zijn muis halen een geheel blok creëren. Na dat er één input is geweest is alles met de muis aan te passen.

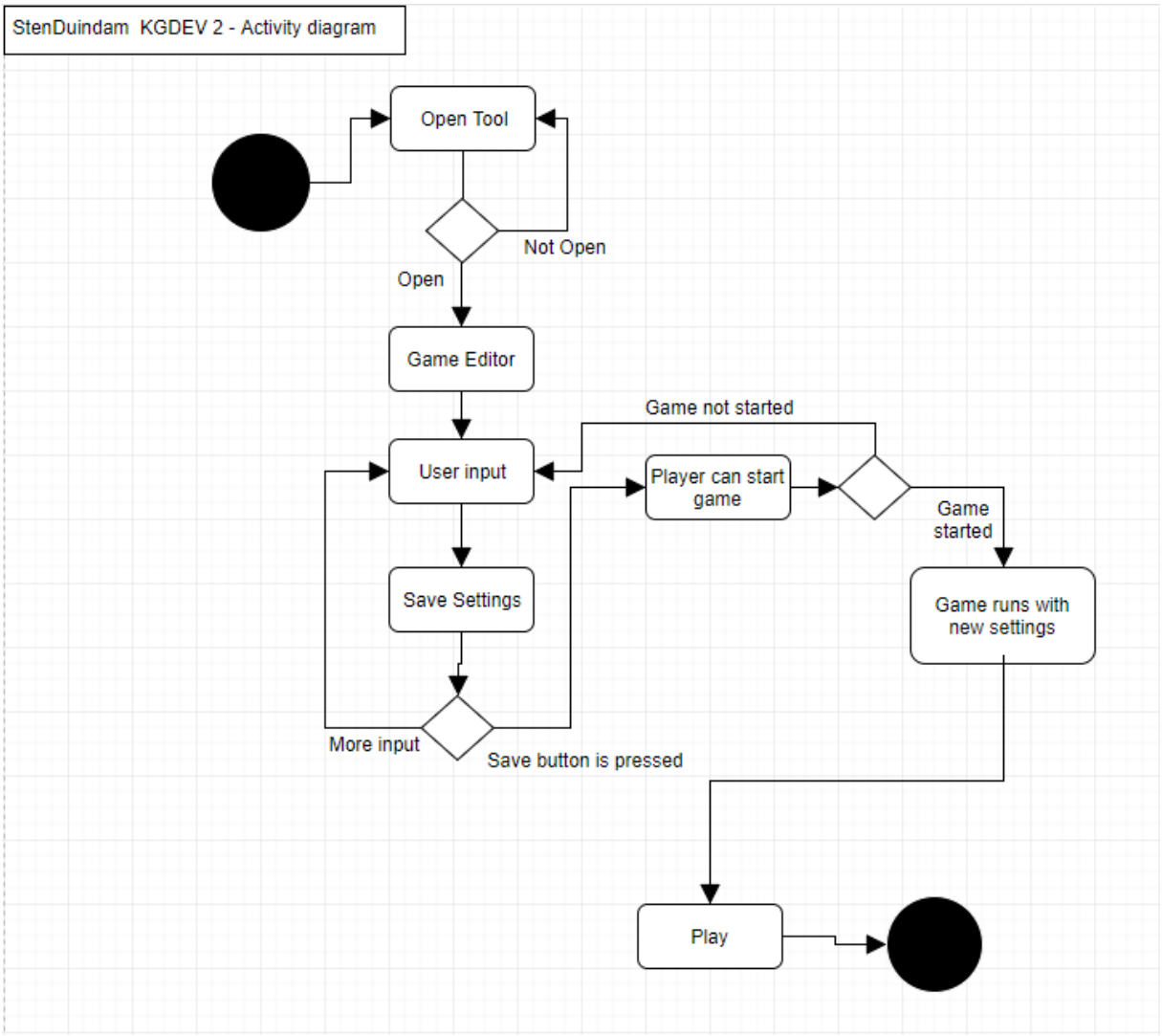
User Flow

Zoals deels beschreven bij het design proces doelde ik op een eenvoudig navigeerbare tool. De gebruiker opent de window, werkt van boven naar beneden om zo het gehele proces te doorlopen. Met de muis is alles aan te passen behalve de naam van het blok bij de BlockEditor. Vervolgens drukt de speler op 'Save' of 'Update' en alle aanpassingen zijn doorgevoerd in het spel. De gebruiker kan nu het spel spelen. Bevalt iets tijdens het spel niet? Kan het op runtime aangepast worden om direct effect te zien.

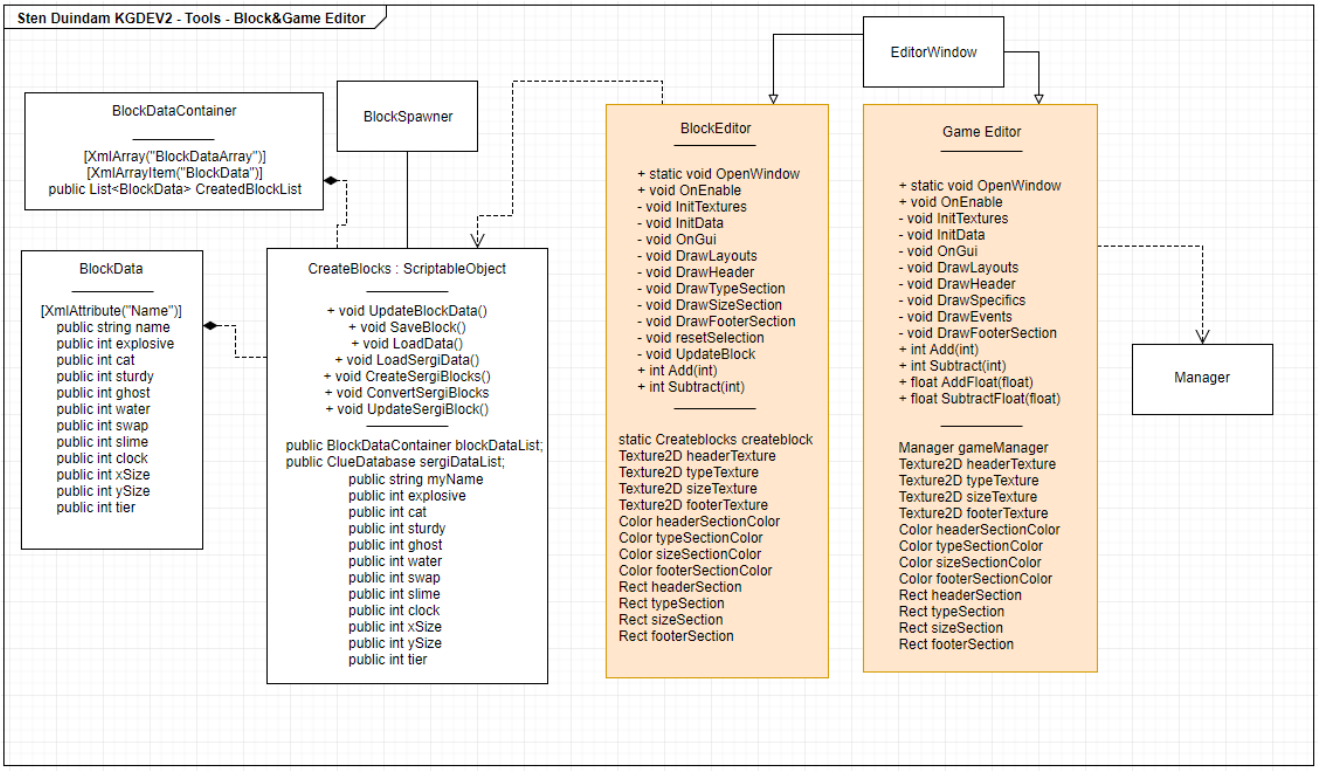
BlockEditor activity diagram



GameEditor activity diagram



UML


Sten Duindam KGDEV2 - Tools - Block&Game Editor
EditorWindow
BlockDataContainer
[XmlArray("BlockDataArray")]
[XmlArrayItem("BlockData")]
public List<BlockData> CreatedBlockList
BlockSpawner
BlockEditor
+ static void OpenWindow
+ void OnEnable
- void InitTextures
- void InitData
- void OnGui
- void DrawLayouts
- void DrawHeader
- void DrawTypeSection
- void DrawSizeSection
- void DrawFooterSection
- void resetSelection
- void UpdateBlock
+ int Add(int)
+ int Subtract(int)
static Createblocks createblock
Texture2D headerTexture
Texture2D typeTexture
Texture2D sizeTexture
Texture2D footerTexture
Color headerSectionColor
Color typeSectionColor
Color sizeSectionColor
Color footerSectionColor
Rect headerSection
Rect typeSection
Rect sizeSection
Rect footerSection
Game Editor
+ static void OpenWindow
+ void OnEnable
- void InitTextures
- void InitData
- void OnGui
- void DrawLayouts
- void DrawHeader
- void DrawSpecifics
- void DrawEvents
- void DrawFooterSection
+ int Add(int)
+ int Subtract(int)
+ float AddFloat(float)
+ float SubtractFloat(float)
Manager gameManager
Texture2D headerTexture
Texture2D typeTexture
Texture2D sizeTexture
Texture2D footerTexture
Color headerSectionColor
Color typeSectionColor
Color sizeSectionColor
Color footerSectionColor
Rect headerSection
Rect typeSection
Rect sizeSection
Rect footerSection
Manager
BlockData
[XmlAttribute("Name")]
public string name
public int explosive
public int cat
public int sturdy
public int ghost
public int water
public int swap
public int slime
public int clock
public int xSize
public int ySize
public int tier
CreateBlocks : ScriptableObject
+ void UpdateBlockData()
+ void SaveBlock()
+ void LoadData()
+ void LoadSergiData()
+ void CreateSergiBlocks()
+ void ConvertSergiBlocks
+ void UpdateSergiBlock()
public BlockDataContainer blockDataList;
public ClueDatabase sergiDataLis;
public string myName
public int explosive
public int cat
public int sturdy
public int ghost
public int water
public int swap
public int slime
public int clock
public int xSize
public int ySize
public int tier

Data uitwisseling

Voor mijn data uitwisseling heb ik een XML ontvangen van Sergi. Zijn XML bestaat uit de volgende structuur:

ClueData.xml - Kladblok

Bestand Bewerken Opmaak Beeld Help

```
<?xml version="1.0" encoding="Windows-1252"?>
<ClueDatabase xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <clues>
    <Quest_Clues>
      <ID>333</ID>
      <clue>There's shoe marks on the wall next to the fence. A cheap brand of sneakers.</clue>
      <found>0</found>
      <isKeyClue>false</isKeyClue>
    </Quest_Clues>
```

Om dit te gebruiken en uiteindelijk om te zetten naar bruikbare data heb ik een conversie function geschreven. Met drie functies zet ik de data van Sergi om naar bruikbare data op een blok. In de tool heb ik twee buttons toegevoegd om zijn data in te laden en aan de lijst van blokken toe te voegen. Op deze manier worden de list van mijn eigen tool en de list die gemaakt wordt van Sergi zijn data samengevoegd tot één geheel.

Met als resultaat:

Wat ik dus heb moeten doen is de class structuur van Sergi namaken. Deze tegenover mijn eigen structuur aan zetten en de waarden naar elkaar overschrijven. Hieronder de functies:

```
public void CreateSergiBlocks() {
    LoadData();
    LoadSergiData();
    Debug.Log("Sergi's listcount: " + sergiDataList.Clues.Count);
    Debug.Log("De Blockdata count: " + blockDataList.CreatedBlockList.Count);

    for (int i = 0; i < sergiDataList.Clues.Count; i++) {
        UpdateSergiBlock(sergiDataList.Clues[i]);
        }
    Debug.Log("Sergi + Blockdata lists merged: " + blockDataList.CreatedBlockList.Count);
    }

//Convert sergi's data to usable block data, create an instance and save it to the blocklist
public void ConvertSergiBlocks() {
    LoadSergiData();
    BlockData thisBlock = new BlockData();
    thisBlock.name = myName;
    thisBlock.explosive = explosive;
    thisBlock.cat = cat;
    thisBlock.sturdy = sturdy;
    thisBlock.ghost = ghost;
    thisBlock.water = water;
    thisBlock.swap = swap;
    thisBlock.slime = slime;
    thisBlock.clock = clock;
    thisBlock.xSize = 1;
    thisBlock.ySize = 1;
    thisBlock.tier = 1;
    blockDataList.CreatedBlockList.Add(thisBlock);
    }

//Update data to then convert
public void UpdateSergiBlock(Quest_Clues newBlock) {
    myName = newBlock.clue;
    explosive = newBlock.id;
    cat = newBlock.found;
    ConvertSergiBlocks();
    }
```

En de data classes:

```
//XML VAN SERGI
[System.Serializable]
public class Quest_Clues {
        public string clue;
    public int id;
    public int found;
    public bool isKeyClue;
    }

[System.Serializable]
public class ClueDatabase {
    [XmlArray("clues")]
    [XmlArrayItem("Quest_Clues")]
    public List<Quest_Clues> Clues = new List<Quest_Clues>();
    }
```

Verder volgt het de pipeline van de normale structuur van het spel. De code is ook te vinden op mijn github in een unitypackage.

Slotwoord

Wat ik graag nog had toegevoegd was een overzicht van bestaande blokken in de editor. Daarnaast misschien een visuele feedback van het gemaakte blok. Wel ben ik tevreden met de uitkomst omdat het direct toegepast en daadwerkelijk gebruikt kon worden. Het maakt het balancen van mijn spel eenvoudiger en dat was mijn doelstelling.