Design document
Kernmodule GDV2
Marlon Sijnesael

Mijn tool:
De tool die ik heb gemaakt voor dit vak is een audio laad- en plaats tool, genaamd “AudioHelper”. Deze tool moet het makkelijker maken om voor niet developers en sound designers audio in unity te plaatsen en te testen. Daarnaast is het ook een manier om overzichtelijker om te gaan met audio files in unity.

Waarom:
Door mijn muzikale achtergrond en interesses is audio vaak een groot onderdeel van mijn projecten, of ik er zelf verantwoordelijk ben of niet. Ik merkte echter al snel dat audio in unity in no time een onoverzichtelijke berg audiosources, clips en settings wordt en dat zorgt ervoor dat het onnodig veel werk wordt om het geluid in projecten zo goed mogelijk te krijgen. Dit geldt zeker als de persoon die verantwoordelijk is, geen developer is, maar bijvoorbeeld een sounddesigner.

wat:
Met deze tool kan je in enkele stappen vanuit een XML file inladen welke clips geladen moeten worden en wat de bijbehorende settings (volume, positie, etc.) zijn.
De tool werkt als volgt:

1. Kopieer de XML sjabloon, die te vinden is in het mapje template in de map “Template” in “Resources”, in en sla hem op als: “audio**[nummer]**.xml” “.
2. Plaats de XML files genaamd “audio**[Nummer]**.xml” in de map “Resources” in de assets folder.
3. Ga naar het kopje “Window” en klik op de knop “**Audio Editor”.**
4. Klik nu op “Create DataFile”.
5. Ga nu naar de map Resources en klik op het mapje “Prefabs”.
6. Sleep de AudioPlacer prefab in de hierarchy en druk op start.
7. Unity zou nu zelf de ingeladen audio clips in de scene moeten plaatsen (deze worden echter wel weer verwijderd bij het stopzetten).

Hoe:
Op het moment dat er op de knop “Create audio data” geklikt wordt, maakt unity een scriptable object die de data van de xml-class deserialiseert en opslaat in de variabelen van de class “AudioPlanClass.cs”.

Daarna zoekt het audioplacer script er zelf voor dat er voor alle audio plans een gameobject met een audio source, de ingeladen clip en de gekozen settings geïnstantieerd wordt.

Ontwerp keuzes:

Serialisatie:
De reden dat ik voor XML heb gekozen, is omdat XML er voor zorgt dat de template makkelijk leesbaar en aanpasbaar is voor de (sound)designers. Ook speelt het mee dat ik al ervaring had met XML en dat ik dus kon focussen op de andere functionaliteit van de tool.

UI:
De reden dat ik de UI zo simpel heb gehouden is dat eveneens terug te leiden naar het gebruiksgemak. Op deze manier is de tool bijna letterlijk met één druk op de knop te gebruiken door de gene die hem moet gebruiken.

Data opslag:
Om de ingelezen data uit de XML op te slaan, instantieer ik een AudioPlanClass met variabelen, die ik vervolgens weg schrijf naar een scriptable object genaamd “audioPlan[**nummer**]”

output van anderen:

Als mensen andere xml files in de tool gebruiken, zullen deze gefilterd worden en in de console zal hier een melding over komen.