MULTIPLATFORM DEVELOPMENT

Naam: Sergi van Ravenswaay
Studentnummer: 3017869

Introductie en uitleg gameconcept

Tijdens Multiplatform development heb ik verschillende dingen uitgeprobeerd. Zo ben ik begonnen met het development voor de 3DS en Wii U. Ook ben ik uiteindelijk voor een netwerk game gegaan en heb ik deze geoptimaliseerd voor meerdere devices. Beide projecten waren heel interessant en heb ik veel van geleerd.

3DS en Wii U

Ik begon Multiplatform development met de research naar 3DS en Wii U development. Om via de officiële weg aan het development te beginnen heb ik mij aan gemeld voor een Nintendo Developer account. Met een Nintendo Development account kan je op je 3DS developen en deze gebruiken.

Na een tijdje wachten kreeg ik geen gehoor van Nintendo en was het voor mij tijd om door te gaan. Ik ben begonnen met het zoeken naar andere opties en ik vond een oplossing hiervoor, het gebruik van Homebrew.

Dit leek mij al super interessant, om zo'n scriptje werkend te krijgen op verschillende platforms. Helaas ondervond ik vele problemen met Homebrew. Zo kreeg ik het in eerste instantie niet geïnstalleerd op mijn devices en daarna kreeg ik vele Read/Write errors en permission errors. Zo heb ik dit project helaas moeten staken.

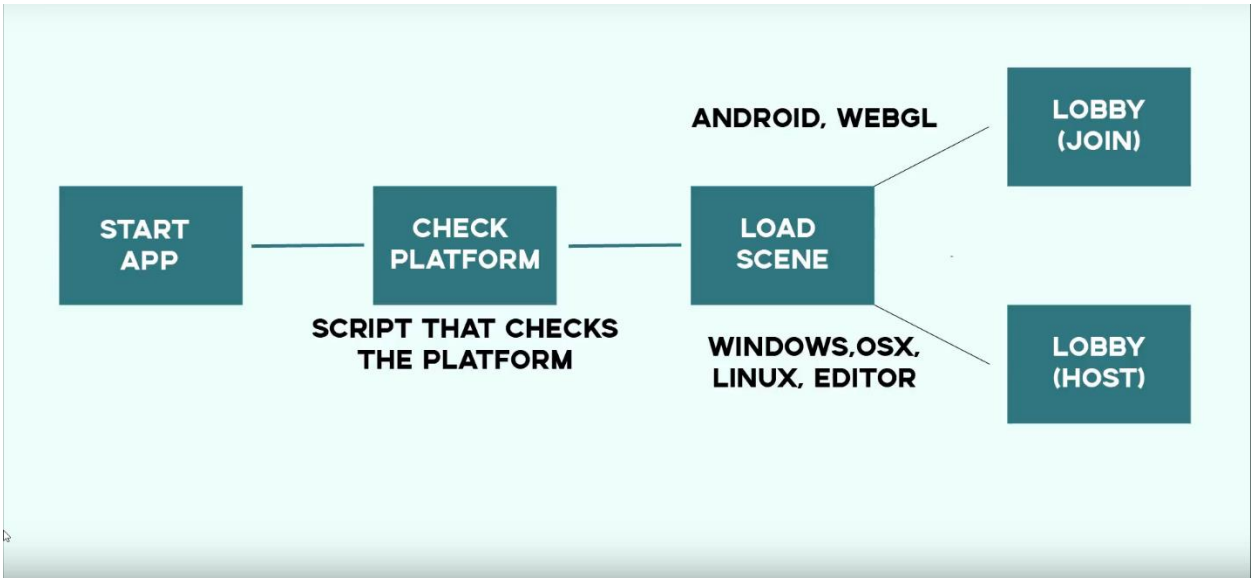
Homebrew
Homebrew is een software laag die je via je SD-Kaart op je 3DS of Wii u kan inladen. Deze kan customizaties of eventuele hacks voor games uitvoeren. Je kan ook via een launcher een C++ applicatie inladen.
Link:
http://smealum.github.io/3ds/

Multiplayer (Make Noise)

Omdat het einde van de periode naderde en ik dus een nieuw concept moest bedenken heb ik ervoor gekozen om een van mijn oude projecten uit de kast te trekken. Dit project was een multiplayer game die beide dezelfde gamedata hadden. Een van de applicaties kon gerund worden op de PC en Website of android applicaties konden zich hier bij aansluiten. Vervolgens kon de PC of Website applicatie als 2e scherm worden gebruikt voor de game. Via de android applicatie kon je je telefoon gebruiken als controller. Dit werkte allemaal vanuit hetzelfde project.

Code, Opzet Uitleg

De code van het project ‘Make Noise’ was en is best rommelig. Meer en deel werd gedaan door 2 checks aan het begin. Deze laadde managers in voor het platform waar de applicatie op draaide. Zo voerde deze alleen scripts uit via events als het platform deze manager alleen actief had. Zo voorkwam je dat rare acties in deze scene werden uitgevoerd die daar niet uitgevoerd moesten worden.



Zoals je hierboven ziet word er bij het starten van de applicatie de app geladen. De applicatie laad dan een scene voor je in wat verschilt op welk platform je draait. Zo kan een PC de game hosten en kan een Mobiel of WebGL applicatie deze joinen. Zodra je deze ‘Host’ joint word je weer in de gamescene geplaatst. Alle (Android, WebGL en PC applicaties) komen weer uit in deze scene. Met de managers, het UNET netwerk systeem en de ingebouwde Platform herkening van Unity kan ik deze in die scene van elkaar onderscheiden.

```
Assembly-CSharp                Platform_Manager                Awake()
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4   using UnityEngine.SceneManagement;
5
6   public class Platform_Manager : MonoBehaviour {
7
8       void Awake() {
9           if (Application.platform == RuntimePlatform.Android || Application.platform == RuntimePlatform.WebGLPlayer)
10          {
11              SceneManager.LoadScene("Lobby_Mobile", LoadSceneMode.Single);
12              return;
13          }
14          if (Application.platform == RuntimePlatform.WindowsPlayer || Application.platform == RuntimePlatform.OSXPlayer ||
15              Application.platform == RuntimePlatform.LinuxPlayer || Application.platform == RuntimePlatform.WindowsEditor)
16          {
17              SceneManager.LoadScene("Lobby", LoadSceneMode.Single);
18              return;
19          }
20
21      }
22  }
23
```

```
    [ClientRpc]
    void RpcClientDisableLobby()
    {
        EventonToggle(false);
    }

    [Command]
    public void CmdLoadMiniGame()
    {
#if UNITY_STANDALONE_WIN
            int num = Random.Range(0, MiniGame_Container.instance.MiniGame.Length);
            MiniGameManager.instance.StartCorLM(num);
#endif
    }
}
```

Hierboven zie je een [Command] en [ClientRPC]. Een ClientRPC word aangeroepen op alle ‘Clients’ dus in dit geval de Android of WebGL applicatie en de Command word aangeroepen op de server en gebruikt op dat moment de data van de clients. Dit zorgde al voor een grote afscheiding van platforms en met wat IF statements erbij wat al in Unity stond werkte dit goed samen.

Ook de controls heb ik bij deze game afgescheiden. Mocht je op de PC of WebGL spelen, dan kan je met je muis swipen. Bij android kan je gewoon je touchscreen gebruiken!

```
    void Update()
    {
#if UNITY_STANDALONE || UNITY_WEBGL || UNITY_EDITOR
        SwipeMouse();
#endif
#if UNITY_ANDROID
        SwipeAndroid();
#endif

    }
```

```
void SwipeAndroid()
{
    if (Input.touchCount > 0)
    {
        Debug.Log("Touch");
        Touch touch = Input.GetTouch(0);
        if (touch.phase == TouchPhase.Began)
        {
            startTime = Time.time;
            startPos = touch.position;

        }
        else if (touch.phase == TouchPhase.Ended)
        {
            endTime = Time.time;
            endPos = touch.position;

            swipeDistance = (endPos - startPos).magnitude;
            swipeTime = endTime - startTime;
        }

        if (swipeTime < maxTime && swipeDistance > minSwipeDist)
            swipeFunction();
    }
}
void SwipeMouse()
{
    if (Input.GetMouseButtonDown(0))
    {
        firstPressPos = Input.mousePosition;
    }
    if (Input.GetMouseButtonUp(0))
    {
        currentSwipe = (Vector2)Input.mousePosition - firstPressPos;
        if(currentSwipe.normalized.x > 0)
        {
            removeStain();
        }
    }
}
```

Platform Specifieke issues en oplossingen

Bij de introductie heb ik al beschreven over hoe het 3DS/WII U project was afgelopen. Het project 'Make Noise' verliep zonder al te veel problemen. Wel had ik problemen met het feit dat hij een MovieTexture op Android niet kan gebruiken. Het is zelfs niet mogelijk om deze in je hierachy te hebben als jij de app probeert te bouwen. Deze heb ik dus telkens uit het project moeten halen tijdens het bouwen. Dit heb ik opgelost om een videoplayer van Unity te gebruiken. Deze werkte veel makkelijker en zorgde niet voor problemen.

Op het einde liep ik echter nog tegen problemen aan met het bouwen van de WEBGL versie van de Applicatie. Ik heb de Android en Windows applicatie wel zonder problemen kunnen bouwen!