XBOX party: Technisch document

Voor iedereen

Het idee achter dit project is dat we naast onze eigen minigame ook allemaal samen een groter spel bouwen genaamd “XBOX party”.

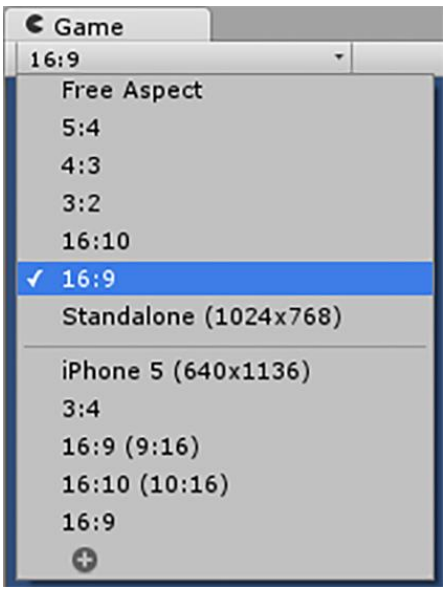
Om dit alles van een leien dakje te laten lopen hebben we hier enkele technische richtlijnen op een rijtje gezet waar we ons allemaal aan gaan houden. De belangrijkste is de volgende:

> *We starten allemaal vanaf de template die te vinden op de server.*

In deze template is al heel wat gedaan om alle minigames samen te laten werken. Als we allemaal van dezelfde template starten maken we zo meteen heel wat potentiële problemen onschadelijk.

Wanneer je hem gedownload maken jullie in de folder “Minigames” je eigen folder aan. Vanaf nu komen alle files die je maakt hierin.

Verder is het belangrijk om te weten dat we momenteel enkel 16:9 resoluties zullen ondersteunen (1920x1080, 1280x720, etc). Om dit altijd in de gaten te houden kan je in Unity i.p.v. “Free Aspect”, “16:9” in de dropdown selecteren.



Voor programmeurs

Input

Voor dit project gebruiken we de input manager van Unity **NIET**.

Dit om de volgende redenen:

- Geeft complicaties bij meerdere controllers
- Geen rumble functionaliteit

Daarom gebruiken we een eigen input manager die gebaseerd is op XInput (de standaard bibliotheek voor XBOX controllers).

Deze is in gebruik vrij identiek aan deze van Unity, al moet je wel via code de buttons & axis registreren. Dit doe je als volgt:

```
private void Start()
{
     InputManager.Instance.BindButton("BoardGame_Submit", …);
     InputManager.Instance.BindAxis("BoardGame_Horiz", …);
}

private void Update()
{
     bool isBtnTrue = InputManager.Instance.GetButton("BoardGame_Submit");
     bool isAxisTrue = InputManager.Instance.GetAxis("BoardGame_Horiz");
}
```

Vergeet niet "`using XBOXParty;`" bovenaan je document te typen. Anders vindt hij de klasse helaas niet.

Voeg ook altijd de **naam van je game als prefix** toe aan de keys hiervan. Anders gaan de knoppen van verschillende groepen alsnog door elkaar lopen!

Namespaces

Omdat we niet weten welke class, struct, enum & function namen jullie gaan gebruiken bestaat er een grote kans dat 2 of meerdere minigames met elkaar in conflict zullen treden. Dit b.v. omdat ze allebei een “GameManager” hebben.

Daarom gebruikt elke groep zijn eigen magische namespace! (zelf uit te kiezen)

In code hoef je alleen **bovenin al je files** het volgende te typen:

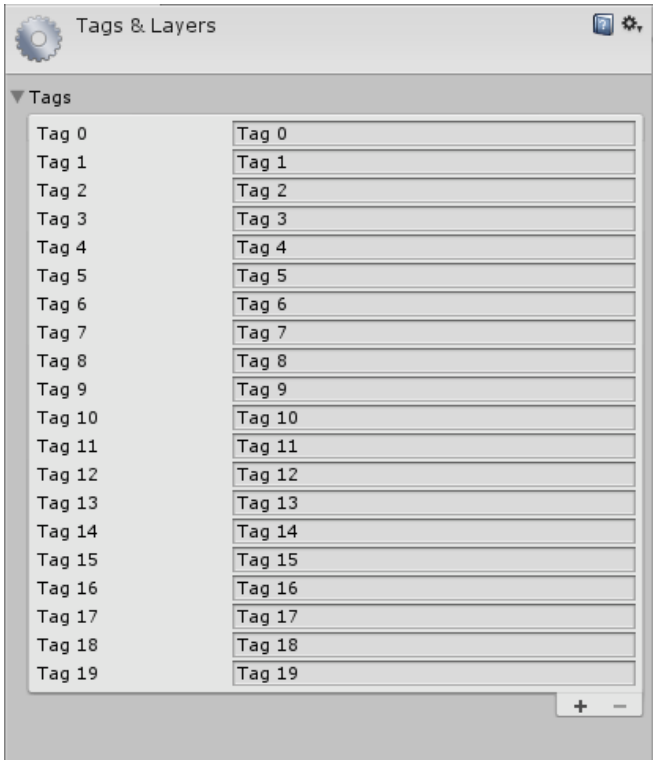
```
namespace MyNamespace
{
    public class MyClass
    {
        ...
    }
}
```

Wil je ooit code gebruiken uit de reeds bestaande code, vergeet dan niet de ”XBOXParty” namespace te includen. In die namespace zit namelijk alle overkoepelende code.

Tags & Layers

De tags & layers in dit project hebben allemaal reeds een (onoriginele) naam gegregen. Deze ben je dan ook verplicht aan te houden in eventuele code.

Als al jullie games op het einde van deze periode samenkomen is dit van essentieel belang. *Vorig jaar is het namelijk om deze reden helaas niet gelukt.*



Code conventions

Aangezien de meeste van jullie de enige programmeurs in de groep zullen zijn lijkt dit stuk waarschijnlijk een beetje overbodig.

Maar ook al voelt dit zo aan is het belangrijk doorheen het hele project een consistente code convention te volgen. (vooral voor variabelen, class & function namen) Dit bevorderd de leesbaarheid enorm, en stel dat iemand later met jou code aan de slag moet is dat meteen een heel stuk makkelijker. (+ werken in een mooie codebase is simpelweg ook leuker!)

Voor de groepen met 2 programmeurs is het uiteraard belangrijk dat jullie dit alles goed afspreken, succes!

TIP: Bij twijfel, volg de conventie van de reeds bestaande code.

Linken met het overkoepelende bordspel

Om jullie minigame te koppelen met het grotere bordspel heb je enkele functies nodig. (om een kleur op te vragen, het spel te beëindigen, etc...)

Al deze functies kunnen je vinden in de **GlobalGameplayManager**.

Dit is een object dat je van overal kan bereiken zonder dat je er zelf een reference naar hoeft te hebben. Hoe dit precies werkt zien we in de normale programmeerlessen, voor nu kan je een reference opvragen door de volgende code te schrijven. (Vergeet niet de XBOXParty namespace te includen bovenaan je document!)

```
private void GetGlobalGameManagerInstance()
{
    GlobalGameManager instance = GlobalGameManager.Instance;
}
```

Aantal spelers opvragen

Het aantal spelers opvragen is uiteraard belangrijk voor een heel aantal dingen. Zeker omdat ons spel 2 tot 4 spelers zal ondersteunen.

Je kan te weten komen hoeveel spelers er zijn door de volgende property van de GlobalGameManager aan te roepen.

Moest je niet meer weten wat properties zijn, check MSDN. (Spoiler: Ze zijn leuk.)

```
public int PlayerCount
```

Kleur opvragen

De kleur van een bepaalde speler opvragen doen we via de volgende functie.

```
public Color GetPlayerColor(int playerID)
```

De parameter is het playerID, tellende vanaf 0.

We gebruiken hier overkoepelende kleuren zodat elk minigame dan sowieso dezelfde kleuren gebruikt. Stel daarenboven dat we later rood toch liever willen inruilen voor magenta, dan kan dat zonder alle minigames aan te gaan passen!

Game resultaten doorgeven

Als het spel is afgelopen willen we terugkeren naar het bordspel en doorgeven wie gewonnen heeft. Dit doen we via de volgende functie:

```
public void SubmitGameResults(List<int> results)
```

Even ter demonstratie:

```
public void Submit()
{
    List<int> positions = new List<int>();

    int playerCount = GlobalGameManager.Instance.PlayerCount;
    for (int i = 0; i < playerCount; ++i)
    {
        positions.Add(i);
    }

    //This is the important function!
    GlobalGameManager.Instance.SubmitGameResults(positions);
}
```

De lijst bevat de posities van de spelers. Index 0 bevat de positie van speler 1, Index 1 van speler 2 etc...

Hou in je achterhoofd dat het mogelijk voor bepaalde spelers om dezelfde positie te delen. (team battles, 1 winnaar 3 verliezers, etc...)

Daarnaast tellen we uiteraard 0 based. Dit betekend dat positie 0 het beste is!

Team opvragen

Voor team battles (2v2 en 1v3) kan je opvragen in welk team een bepaalde speler zit.

In 1v3 zal altijd de speler die eerst staat solo spelen, in 2v2 is dat de eerste en de laatste tegen de middelste 2 spelers.

Hoe dan ook. Deze functie werkt voor allebei:

```
public int GetPlayerTeamID(int playerID, MinigameMode
minigameMode)
```

De mogelijke minigamemodes zijn de volgende.

```
MODE_FFA
MODE_2V2
MODE_1V3
```

De return value is het team ID. Dit zal meestal 0 of 1 zijn. Bij Free For All heeft elke speler een eigen team.

Minigame toevoegen

Je kan je minigame in de gameplay loop testen door hem toe te voegen onder “Debug Minigame” in de MinigameManager. Deze kan je vinden in de “BoardScene” scene.

Als je daarna alles wilt testen, start je het spel vanaf de “SplashScene” scene.

Veel plezier!

Voor Artists

Samenwerking met programmeurs

Het eerste en misschien wel allerbelangrijkste punt.

Aangezien jullie over het algemeen maar 1 programmeur in het team zullen hebben zou het zonde zijn moest hij/zei tijd moeten spenderen aan het importeren van jullie art.

Importeer dus alles zelf even in Unity. Zorg er voor dat alles mooie namen heeft (zoals jullie hebben afgesproken) en exporteer het als een unitypackage. (of beter: push het naar een git server!)

Speler objecten

Aangezien ons spel 2 tot 4 spelers ondersteund, is het belangrijk dat de spelers duidelijk hun objecten op het scherm kunnen onderscheiden van elkaar.

Elke speler heeft hiervoor een kleur. Via code krijgt de programmeur deze mee en zal die gebruiken om de nodige objecten in te kleuren. Zorg dus dat de gewenste textures voorzien zijn op deze feature.

Let op: Dit wilt niet zeggen dat je niet op andere manieren onderscheid mag maken tussen de speler objecten! Hoe duidelijker/leuker, hoe beter!

Resoluties

Naast de resolutie van het hele spel (16:9) heeft ook het logo en het spelbord een specifieke resolutie. Deze zijn respectievelijk 1024x512 en 1920x1080.

Zorg er dus voor dat jullie versies hiervan dezelfde resolutie hebben!

Tip: Je kan het huidige logo openen in photoshop en in dat bestand werken. Dan komt dit automatisch goed.

Optimalisatie

Omdat XBOX party een heel groot spel wordt is het belangrijk dat we allemaal proberen om onze filesizes zo klein mogelijk te houden.

Er staat hier geen harde restrictie op (tricount, texture size etc.) omdat het niet de bedoeling is om jullie creatieve vrijheid te belemmeren.

Met andere woorden: Maak je model niet enorm hoekig en je textures niet enorm pixelig om hieraan te voldoen! Elke geval is uniek, en bij twijfel vraag het even aan een docent waar je kan optimaliseren!

Hou b.v. wel rekening met de speelhoek van het spel. Ga je de onderkant van een karakter nooit zien? Dan kan je daar wat polygons/texture space besparen. Zo kan je elementen die echt van belang zijn dubbel en dik in de verf zetten!

Texture sizes

Textures die niet voor UI gebruikt worden moeten altijd een “power of 2” formaat hebben. Dit wilt zeggen een resolutie van: 64x64, 128x128, 256x256, 512x512, 1024x1024 of 2048x2048. (Nooit groter!)

Oudere videokaarten weten namelijk niet zo goed hoe ze moeten omgaan met andere formaten. En aangezien we zo weinig mogelijk mensen willen uitsluiten van XBOX party (ook voor pc) moeten we hier rekening mee houden.

Unity zorgt er voor dat alles wat als “Sprite” geïmporteerd wordt automatisch aan dit criteria voldoet (texture atlasses). Dus daar hoeven jullie je geen zorgen om te maken.