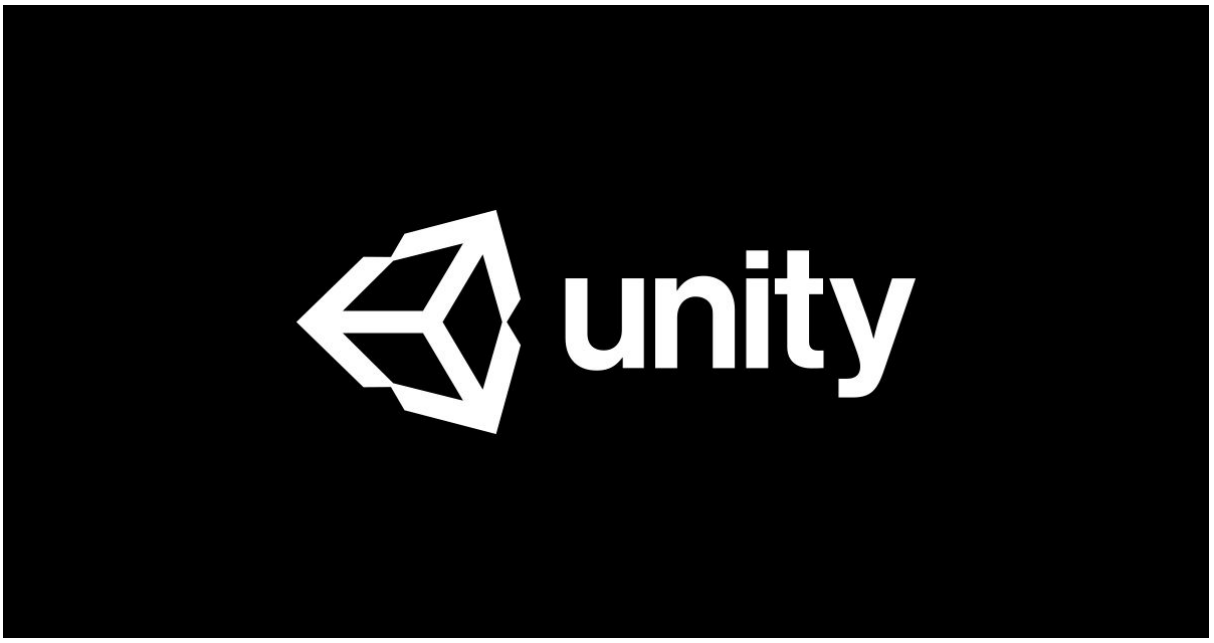


Unity VR GameDev Exam

Escape VR

Lavet af: Nijas Winum Hansen

Github: https://github.com/nijashansen/GameDevExam

Indholdsfortegnelse

Description

Overall description

Spillet hedder Escape VR, og er udviklet i Unity Game Engine. Spillet handler om at skulle slippe ud af et hus på en ø før tiden løber ud, spilleren har 3 minutter til at slippe ud ellers taber spilleren. Genren er Escape rooms, bare i VR. Spillet er udviklet via et framework som hedder XRToolkit(XRTK). Os som studerende kunne vælge imellem VRToolkit(VRTK) eller OculusVR (OVR), jeg har på en måde valgt at arbejde med OVR, og så alligevel ikke. Spillet er udviklet til oculus quest som er et standalone VR headset som ikke behøver ledninger og som er baseret på Android. Det Framework som jeg gik med hedder XRTK. XRTK er smart fordi det er unitys version af et VR framework som implementere OVR i sig. OVR bliver også fjernet ved senere opdateringer til XRTK. XRTK er lavet til unity som en samlet pakke, som gør det nemt for udviklere at komme i gang hurtigt. XRTK kan compile til både SteamVR(HTC VIVE, PCVR), og til Oculus Quest, man skal kun ændre i unitys build settings og sætte det op til den platform man ønske at udvikle til. Det er proppet med forskellige scripts som gør det nemt at implementere et VRRig, og Grabbable Objects. Jeg vil komme mere ind på XRTK og OVR senere i rapporten.

Pens Model

Min Pens Model er lavet i samarbejde med en del game testing, det er en blanding af min egen mening og min kærestes mening. Pens modellen hjælper med at se hvordan ens spil er endt i forhold til andres mening og andres holdning. Den kan også hjælpe til at se hvad man kan gøre bedre til næste gang. i Pens modellen er der 3 hovedområder som skal opfyldes: kompetence, autonomi og tilknytning, Kompetencen i dette sammenhæng er at finde frem til hvordan man kommer ud af huset. Autonomi er den automatiske timer som tæller ned indtil spilleren enten vinder eller tiden løber ud som resultere i at spilleren taber. Tilknytning til characteren sker i start menuen, i form af en kort beskrivelse af hvad spillet går ud på. nedenunder ses tabellen for Escape VR:

Pens Model	**Fun / Enjoyment**	**Feel Immersed**	**Value Game**	**Will Buy more of developers games**	**Recommend game to others**
Kæresten	1.5 ⭐	1,5⭐	⭐	⭐	⭐
Udvikleren	⭐⭐	⭐	⭐	⭐	⭐⭐

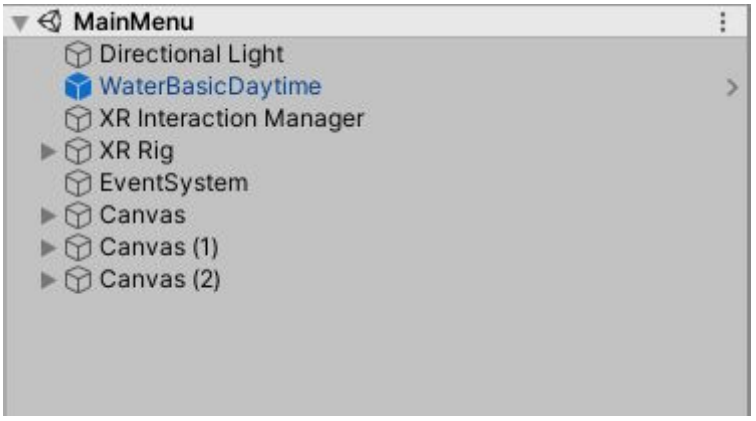
Kæresten syntes at spillet var godt taget i betragtning af at det er udviklet på godt og vel 10 dage, hun mente at der skulle være flere objekter til at interagere med og evt. noget som kommer imod spilleren ude fra øen eller vandet, hun mente også at storytelling skulle være bedre. Et videoklip eller noget til at fortælle historien om hvorfor man er i huset osv.

Overview Of Scenes

Spillet består af 4 scener, Firstlvl er hvor spilleren skal forsøge at komme ud af huset, de andre scener er en form for menu

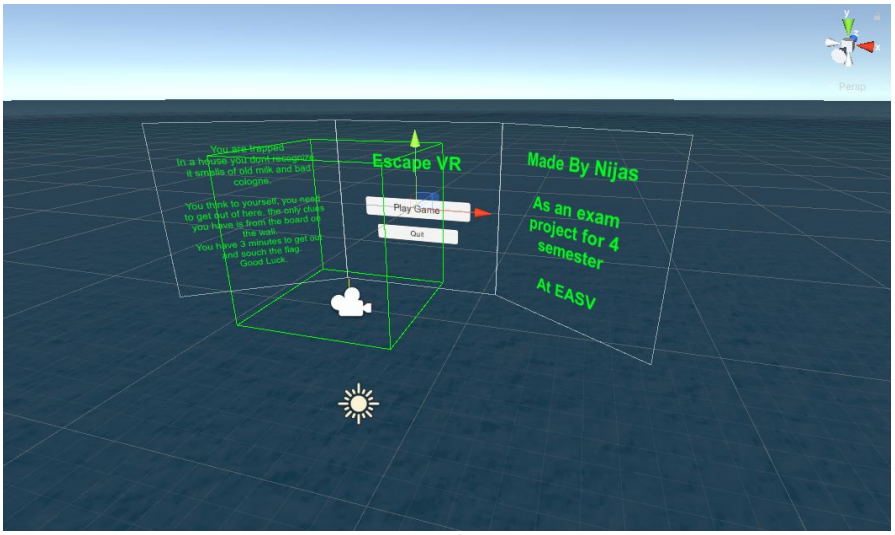
MainMenu

Vi kigger først på MainMenu Scenen fordi det er den første man bliver mødt med. hierarchy i denne scene består af flere UI elementer og en XR Rig med tilhørende XR Interaction Manager



XR Rig er der hvor Spilleren befinder sig, samt spillerens Camera (Headset) og spillerens hænder (Oculus Controllere). XR Interaction Manageren er den som sørger for at spilleren kan interagere med forskellige elementer. Dette objekt findes også i de andre Hierarchies som sørger for at se hvilke objekter kan samlet op og hvilke ikke kan.

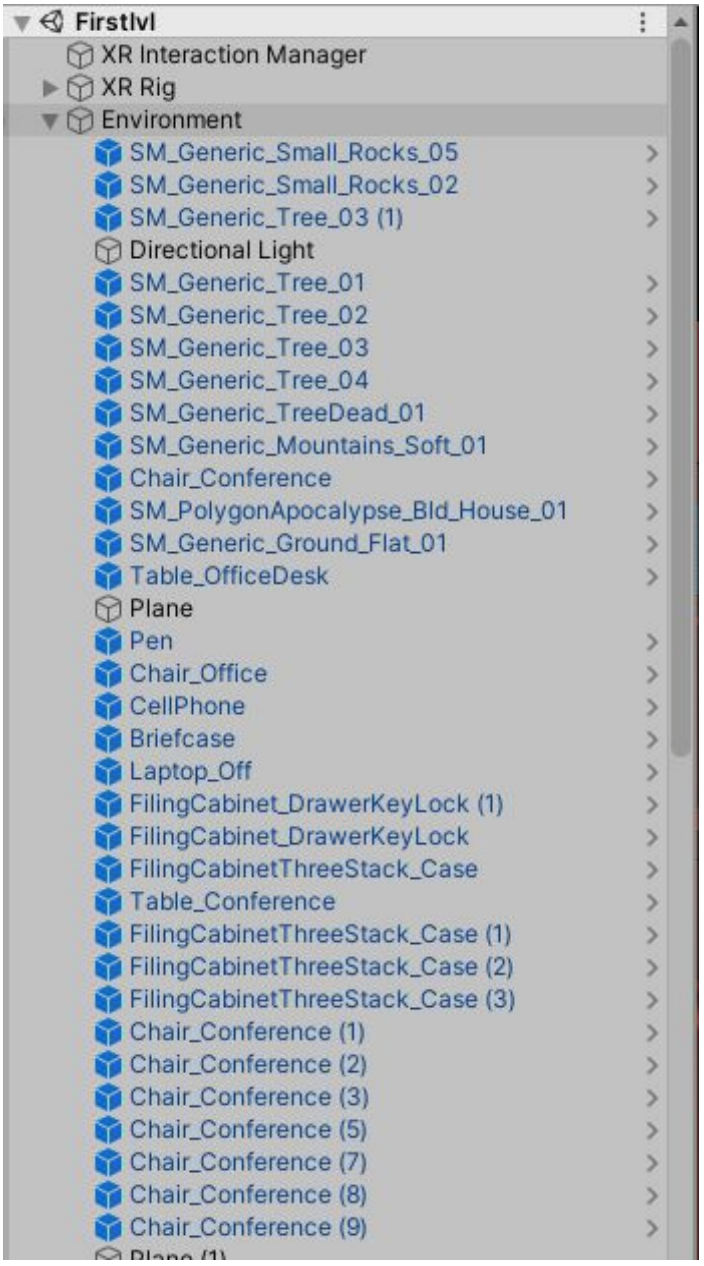
På hver canvas tilkobles et Script fra XRTK som gør at Kanvassen, kan integreres med, dette script hedder: XR Input Module, som fortæller interaction manageren at dette objekt kan tage imod XR Input i form af en trigger event eller andre former for input.



XR Rig er den grønne firkant hvor der befinder sig et kamera indeni.

Firstlvl

Den næste scene er Firstlvl, denne scene opholder en hel del Environment objekter som spilleren kan interagere med, samle op, låse op, og slå andre objekter med.



Igen ses XR Rig, og Interaction Manageren, i denne scene bruges de til at styre spilleren og til at se hvilke objekter som kan samlet op eller ej.

På hvert objekt som kan samles op tilføjes et script fra XRTK: XR Grab Interactable. Nogle objekter kan være lidt utilregnelige når man samler den op, for eksempel, en økse som befinder sig i en kiste, da jeg play testede samlede jeg den op og øksen kom ud af siden af hånden. Dette fixes ved hjælp at et script som fortæller XR Grab Interactable scriptet at den skal holdes ved et offset.

I dette script bliver to metoder Overridet på XR Grab Interactable OnSelectEnter og OnSelectExit, her sættes offsettet til at vende den rigtige vej i forhold til spilleren.

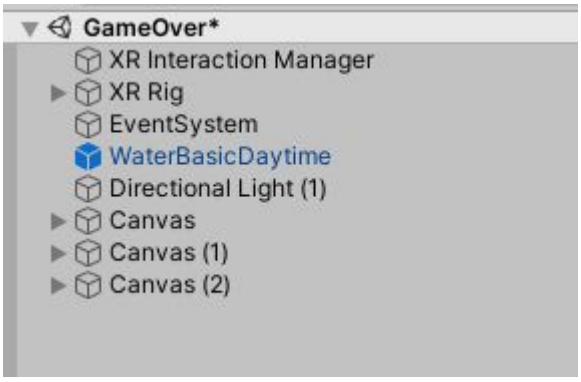
På XR Rig, sidder der en Canvas som ikke er aktiv, når spillet starter, hvis brugeren trykker på menu knappen, bliver dette canvas sat til aktiv og brugeren for en menu op foran sig, brugeren kan så vælge at vende tilbage til MainMenu Scenen eller afslutte applikationen helt.

For at komme ud af huset skal man have fat i en økse, som er det eneste som kan slå brædderne ned fra døråbningen, dette tjekkes i et script for sidder på hver enkelt planke, dette script tjekker om navnet på objektet er “Axe”, hvis det er det så bliver plankerne ødelagt, og spilleren kan komme ud, alt dette sker ved hjælp af triggers og colliders. Det er også med colliders og triggers der bliver set om spilleren har vundet eller ej.

I scenen finder man en timer som sidder over en opslagstavle, den timer tæller ned for brugeren, fra 3 minutter til nul, dette sker i et script som omregner sekunder til minutter og sekunder, hver gang billedet skifter, altså i Update() metoden. Hvis sekunderne rammer nul vil GameOver scenen blive vist.
Det to sidste scener minder meget om hinanden og om MainMenu Scenen.

GameOver

GameOver scenens hierarchy ser således ud:



det er en tro kopi af men i stedet for “velkommen til spillet” bliver man præsenteret for “you lost!”, man kan derefter trykke på “try again”, og spillet start for ny, eller trykke på “Quit” som afslutter applikationen.

GameWon

GameWon scenen er meget den samme dog med et par enkelte undtagelser.

Her er det teksten “You Won” samt nogle dummies som danser i baggrunden, her kan man returnere til MainMenu eller afslutte applikationen.

The Good Things

OpenTreasureBox Script

For at kunne slippe ud af huset skal man slå bræderne ned med en økse, men man skal først have fat i øksen før man kan komme ud, øksen ligger i en kiste som man skal finde en nølge til for at kunne åbne op. når man har fundet nøglen tager man den med over til kisten og låser så kisten op ved at stikke nøglen ind i nøglehullet. Når nøglen ramme nøglehullet, køres OpenTresureBox scriptet fordi kisten har en box collider som befinder sig direkte ved nøglehullet.

```
Unity Message | 0 references
void OnTriggerEnter(Collider other)
{
    if (other.gameObject.name == "key_silver" && count == 0)
    {
        TopPartOfBox = new Vector3(-45.0f, 0.0f, 0.0f);
        KeyAfterColidingPos = new Vector3(-5.1722f, -3.066f, -11.4463f);
        KeyAfterColidingRot = new Vector3(177f, 23.2f, -71.32f);

        topBox.transform.Rotate(TopPartOfBox);

        Destroy(other.GetComponent<XRGrabInteractable>());

        other.transform.position = KeyAfterColidingPos;
        other.transform.Rotate(KeyAfterColidingRot);

        count = 1;
    }
}
```

Når et game object kommer ind i kollideren køres der et check for om objecteds navn er “Key_Silver” hvis det er det ved vi at det er nøglen som er kollideret med kisten, dernæst sættes der nogle værdier i nogle Vector3 objekter som gør at kisten ser åben ud, og nøglens XR grabInteractable component bliver destrueret, fordi nøglen skal ikke bruges mere. dernæst bliver nøglens værdier sat til at det ser ud som om den sidder i kisten. den eneste værdi jeg ikke har forklaret er count, count tæller op hvis nøglen har kollideret med kisten før eller ej, dette kunne også gøres med en bool.

Timer Script

Timer scriptet bruges til at omdanne det indtastede antal sekunder om til minutter og sekunder, og efterfølgende vise det på et gameobject. Scriptet lokale værdier er således:

```
public TextMesh TimerText;
public float timeRemaining = 10f;
public bool timerIsRunning = false;
```

TextMeshen er der hvor timeren skal vises i scenen, timeRemaining er hvor meget tid der er tilbage, pr standard er den sat til 10 sekunder. og timerIsRunning fortæller om timeren kører eller ej.

i start metoden bliver timerIsRunning sat til true, fordi spillet starter når spilleren loader scenen.

```
void Start()
{
    timerIsRunning = true;
}
```

i Update metoden bliver der lavet et check om timeren kører, hvis ikke dette tjek køres ville update metoden prøve at load GameOver scenen igen og igen.

```
// Update is called once per frame
Unity Message | 0 references
void Update()
{
    if (timerIsRunning)
    {
        if (timeRemaining > 0)
        {
            timeRemaining -= Time.deltaTime;
            DisplayTime(timeRemaining);
        }
        else
        {
            SceneManager.LoadScene("GameOver");
            timeRemaining = 0;
            timerIsRunning = false;
        }
    }

}
```

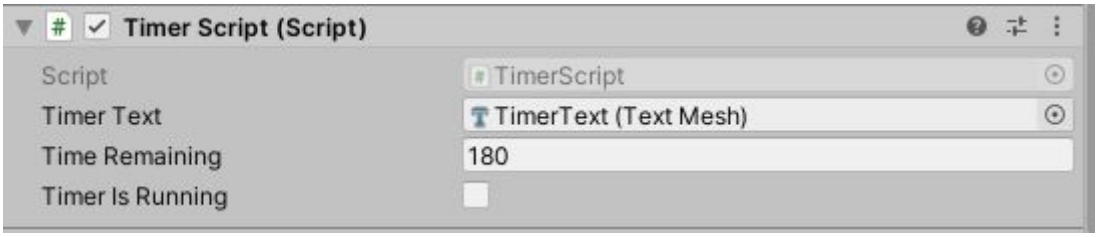
dernæst bliver der checket om timeRemaining er større end nul, hvis den er det bliver den - med Time.DeltaTime som er tiden på eksakt tid i unity, derefter bliver den vist på gameobjected, som jeg forklare længere nede.

hvis ikke dette if statement ikke er sandt længere bliver GameOver scene loaded, og time remaining, bliver sat til 0 og timerIsRunning sat til falsk.

Når tiden skal vises i gameobjected skal den formateres om til minutter og sekunder, dette gøres ved at tage det indtastede antal sekunder, og for at få minut antal divideres det med

```
void DisplayTime(float timeToDisplay)
{
    float minutes = Mathf.FloorToInt(timeToDisplay / 60);
    float seconds = Mathf.FloorToInt(timeToDisplay % 60);

    TimerText.text = string.Format("{0:00}:{1:00}", minutes, seconds);
}
```

60 for eksempel : 180 / 60 = 3. og for sekunder bruges der modulus i 60 for at den tæller til 60 og kun 60, således 180 % 60 = 60. jeg regner disse værdier om til ints for at få et exact tal og ikke et decimaltal som en float er, dernæst sættes disse værdier ind i en formateret string. stringen har plads til 2 objekter ved brug af 2 akkolade tegn, dernæst sættes værdierne ind som så tager plads i stringen på plads enten 0 som er minutter, eller 1 som er sekunder. dette bliver sat i textmeshen text property, som viser det på gameobjected.



Her bliver værdierne sat på objekted og objected ændre så tid, efter hvert sekund. selve timeren består af et plane og et empty gameObject som har en text mesh på.

Movement Script

Jeg er meget stolt af mit movement script. Dette script gør CharacterController Componentet mindre eller højere alt efter om spilleren bukker sig, kravler på alle fire, eller hopper.

Vi starter med de variabler som bruges igennem scriptet

```
public float speed = 1.0f;
public float gravityMultiplier = 1.0f;
public List<XRController> controllers = null;

private CharacterController characterController = null;
private GameObject head = null;
```

speed er hastigheden på spilleren når han bevæger sig. GravityMultiplier er hvor meget tyngdekraft der er i verdenen. Liste med controllere er en liste som bruges til at hente værdier ud fra hvad brugeren trykker på. CharacterController og Head er spillerens collider.

jeg overrider en metode som hedder Awake() som kaldes når objektet bliver instantieret, og bliver kun kaldt en gang. Her hentes komponenter som var sat til null i lokal variablerne.

```
protected override void Awake()
{
    characterController = GetComponent<CharacterController>();
    head = GetComponent<XRRig>().cameraGameObject;
}
```

I Start() og Update() metoderne bliver der kaldt en metode ved navn PositionController, som sætter CharacterControllerens parametre. Først bliver characterControllerens højde sat ved hjælpe af et objekt som hedder head. Scriptet devidere controlleren med 2 for at sikre at hvis en person er høj vil han ikke være for høj til spillet og det samme gælder for hvis man er en lav person.

```
private void PositionController()
{
    float headHeight = Mathf.Clamp(head.transform.localPosition.y, 1, 2);
    characterController.height = headHeight;

    Vector3 newVector = Vector3.zero;
    newVector.y = characterController.height / 2;
    newVector.y += characterController.skinWidth;

    newVector.x = head.transform.localPosition.x;
    newVector.z = head.transform.localPosition.z;

    characterController.center = newVector;
}
```

Den næste metode checker om der er noget input fra controlleren som spilleren holder i sin hånd. Her checker vi om der er nogle controllere, samt om brugeren må trykke på knapperne eller analog pindene på controllerene, hvis man må bliver metoden Check for movement kaldt.

```
private void CheckForInput()
{
    foreach(XRController controller in controllers)
    {
        if (controller.enableInputActions)
        {
            CheckForMovement(controller.inputDevice);
        }
    }
}
```

Hvis spilleren rykker med analog pinden, sendes der en Vector2 ud, som så bliver først med ind i metoden StartMove.

```
1 reference
private void CheckForMovement(InputDevice device)
{
    if (device.TryGetFeatureValue(CommonUsages.primary2DAxis, out Vector2 position))
    {
        StartMove(position);
    }
}
```

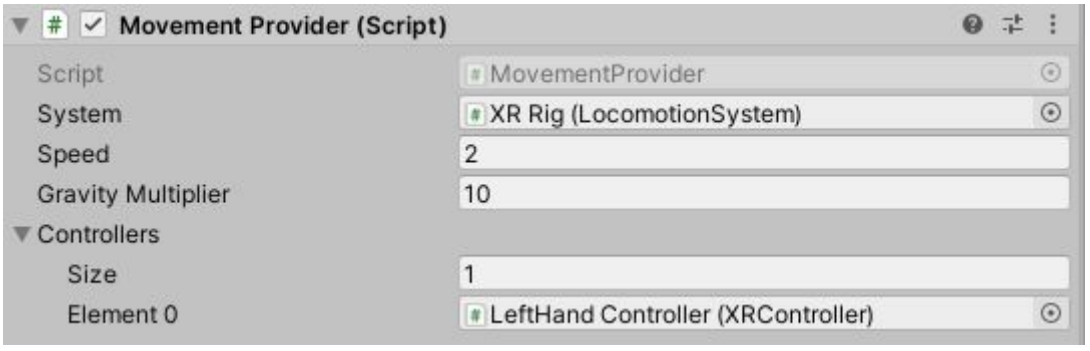
StartMove fungere på den måde at den tager den Vector2 parameter og laver en ny Vector3 som har Vector2'erens informationer i, dernæst laves der en anden Vector3 som ser hvilken vej spilleren vender. dernæst kobles disse to Vector3 sammen i den ene Vector3, som dernæst ryger ind i en ny Vector3, som til sidst så får CharacterControlleren til at bevæge sig.

```
private void StartMove(Vector2 position)
{
    Vector3 direction = new Vector3(position.x, 0, position.y);
    Vector3 headRotation = new Vector3(0, head.transform.eulerAngles.y, 0);

    direction = Quaternion.Euler(headRotation) * direction;

    Vector3 movement = direction * speed;
    characterController.Move(movement * Time.deltaTime);
}
```

Der er stadig en metode som mangler, det er ApplyGravity, som bliver kaldt hver frame for at sikre at spilleren befinder sig på jorden. der sættes værdier fra unitys physics engine, som bliver kaldt hvert frame på character controlleren.

```
private void ApplyGravity()
{
    Vector3 gravity = new Vector3(0, Physics.gravity.y * gravityMultiplier, 0);
    gravity.y *= Time.deltaTime;

    characterController.Move(gravity * Time.deltaTime);
}
```

For at sætte det op indeni unity sætter jeg scriptet på XRRig GameObjektet således:



Systemet er selve locomotion systemet som er et script der kommer fra XRTK, så sættes hastigheden på spilleren og spilleren tyngdekraft. Det næste som sættes er controlleren som skal styre Locomotion for spilleren, man fortæller at der er en controller som styrer om spilleren går frem eller tilbage eller fra side til side. I mit projekt er det den venstre controller som styre dette.

Chosen Advanced Topic

XRToolKit

Til at starte med fandtes der forskellige måder at koble sit vr udstyr til unity, OVR, SteamVR, og sikkert flere. Unity er kommet med et nyt framework som gør det nemt for udviklere at koble vr sammen udover de understøttede platforme, det vil sige at udviklere skal kun tænke på at gøre deres spil så godt som overhovedet muligt, uden at bekymre sig om det vil køre på andres platforme. XRTK gør det også nemt for udviklere ved at give dem en række af scripts som gør objekter Grabbable, samt scripts til at bevæge ens karakter. XR kommer understøttet med Snapturn og Teleportation locomotion. Snapturn bruges i den forstand at spilleren kan dreje 45 grader til højre uden at dreje hovedet. Teleportation locomotion, bruges til at flytte spilleren til forskellige punkter, så brugeren ikke er låst til et sted. Dette betyder at udvikleren kun skal tilføje det script som skal bruges og ikke skrive kode selv, dette gør udvikling til vr meget nemmere og tiden væsentlig kortere for udviklere at få deres spil ud på markedet. Ulempen ved XRTK er at det lige nu er i preview, det vil sige at frameworket er i sit udviklings stadie, og er ikke taget i betragtning som færdig, Dette er også grunden til at jeg som udvikler selv skulle lave min form for bevægelse, altså mit eget locomotion. XRTK kan ikke kun bruges til VR, men også Augmented Reality(AR), der efter navnet XR. De scripts som XRTK kommer med kan bruges til både VR, og AR. XR Toolkit havde release med unity version 2019.3, og er blevet taget godt imod af udviklere, som har lavet nogle gode spil ved brug af Frameworket.

XR Frameworket har en liste af forskellige ting til VR som indeholder:

- Cross platform XR controller input
- Basic object hover, select, og grab
- Haptisk feedback i form af vibrationer i controller
- Visual feedback, i form af linjer fra spillerens hænder
- Basic UI interaction med XR controllere
- VR Rig for både stationær grænse og rumstørrelse

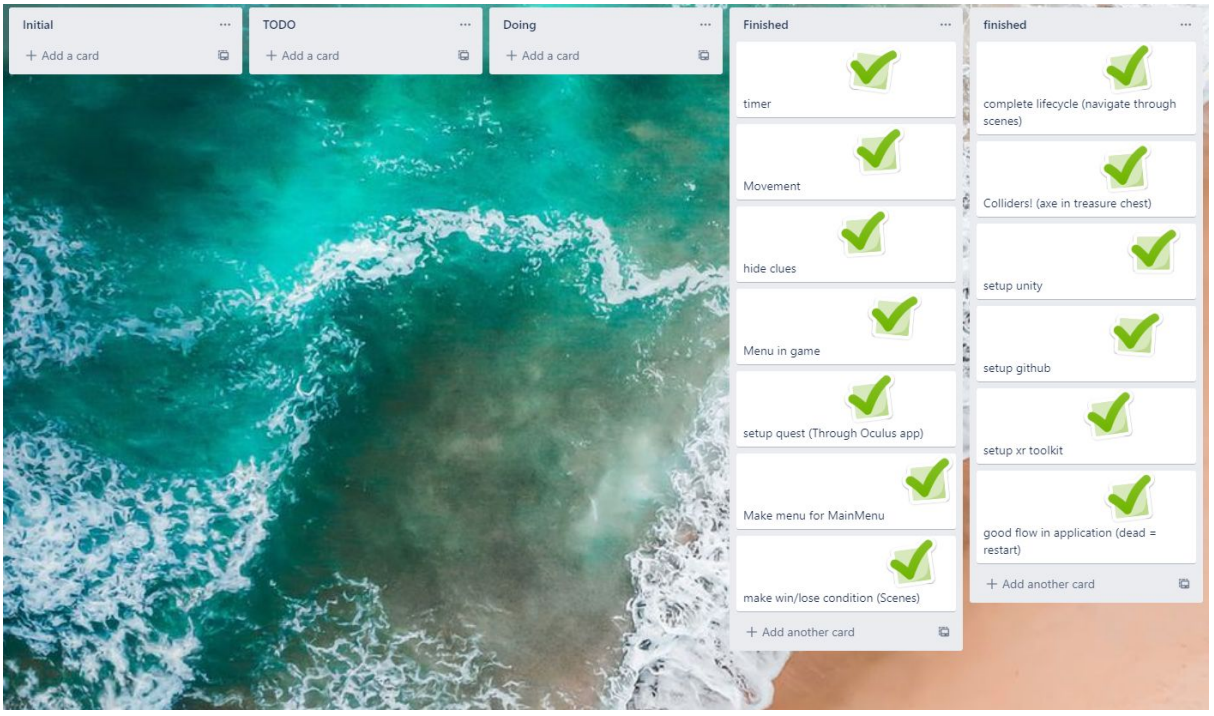
XR Holdet arbejder lige nu på at implementere Hand tracking fra headsets som Oculus Quest.

XR Toolkit holdet sigter efter at lave et framework som gør det nemt for udviklere at komme “Up’n’running” meget hurtigt, ved brug af komponenter og ingen kode.

XR var det framework jeg valgte at gå med fordi det er udført godt indtil videre og havde oculus integration indbygget, i sig, så jeg ikke skulle bøvle med OVR da jeg har hørt at det skulle være en smule træls. XR har hjulpet mig meget med at komme i gang med dette projekt og også hjulpet mig til at forstå spil bedre når jeg spiller dem.

Managed Time

jeg startede mit projekt i gamedev omkring d. 17 maj, jeg tænkte at med en deadline på d. 29. så ville jeg sagtens kunne nå det, jeg har arbejdet på at få unity til at build til android, hvilket var en udfordring for mig i starten, og var faktisk en udfordring så stor at jeg måtte lave et nyt projekt. så i realiteten har jeg startet d. 20 maj, men føler at jeg er kommet i mål med det som jeg ville og er tilfreds med min præstation. Mit projekt startede med at få XR op og køre og installere de nødvendige packages, så Unity var det eneste jeg skulle bekymre mig om. Dagene gik og jeg skulle til at kigge på rapporten. Jeg startede med at skrive, d. 27 maj, og afsluttede d. 29 om formiddagen med et tjek for kommaer, osv. Jeg har brugt trello til at holde styr på mit projekt, hvad jeg manglede, osv. jeg ville ikke bruge scrumwise eftersom jeg ikke er vil med deres platform, og syntes at trello er nemmere at hoppe til og finde rundt i. Da jeg startede mit projekt, gjorde jeg det klart for mig selv at jeg ikke ville bruge for megen tid på det, samtidig med fritid, og deltidsarbejde, så jeg besluttede mig for at lægge nogle arbejdstimer for projektet, jeg startede med at arbejde fra 9 til 15, hvilket virkede fint de først par dage. Så kom mit setback med unity's build settings, og jeg tilføjede en time om morgenen, så min arbejdstid, hed fra 8 til 15. hvilket også var fint.



Konklusion

Jeg har fået ny forståelse for spiludviklere, det er svært, og især hvis ens engine ikke fungere som den skal. Jeg har lært meget i denne process. Jeg har lært hvordan VR fungere i forhold til udvikling, samt lært et nyt framework, der er kraftigt og helt sikkert et jeg vil komme tilbage til hvis jeg skal udvikle et andet vr spil en dag. Ikke nok med at jeg har lært meget omkring udvikling men også meget omkring VR og hele det fællesskab som findes på internettet, samt rundt om på hele kloden. VR er et kraftigt værktøj som kan bruges til mange ting, ikke kun spil, men også hvis man skal designe et nyt hus, og man gerne vil vise kunden hvordan huset vil komme til at se ud, så bygger man det hurtigt i en spil engine, og viser det for kunden, dette giver hurtig feedback, som er meget vigtig for hurtige eksekveringer. Hele processen har hjulpet mig til at blive en bedre programmør og bedre til at fordele min tid, i forhold til en deadline.