2019-
2020

3D Game Programming

THE HEIST

LENNERT VAN RYSSELBERGHE

Project

Algemeen

Het spel is first person parkour game waarin de speler het verhaal volgt van een dief die een object genaamd “The Thing” probeert te stelen uit een bedrijf. Om dit te bereiken beschikt de speler over een paar handige parkour technieken die hem zeker van pas zal komen.

Contributie

Omdat ik alleen werkte heb ik alles alleen gedaan.

Features

- Verhaal (via dialogue)
- First Person
- Wallrunning
- Crouching
- Sliding
- Climbing
- Shuffeling
- Item pickups
- Handgemaakte map
- Animaties
- Start scherm
- Eind scherm

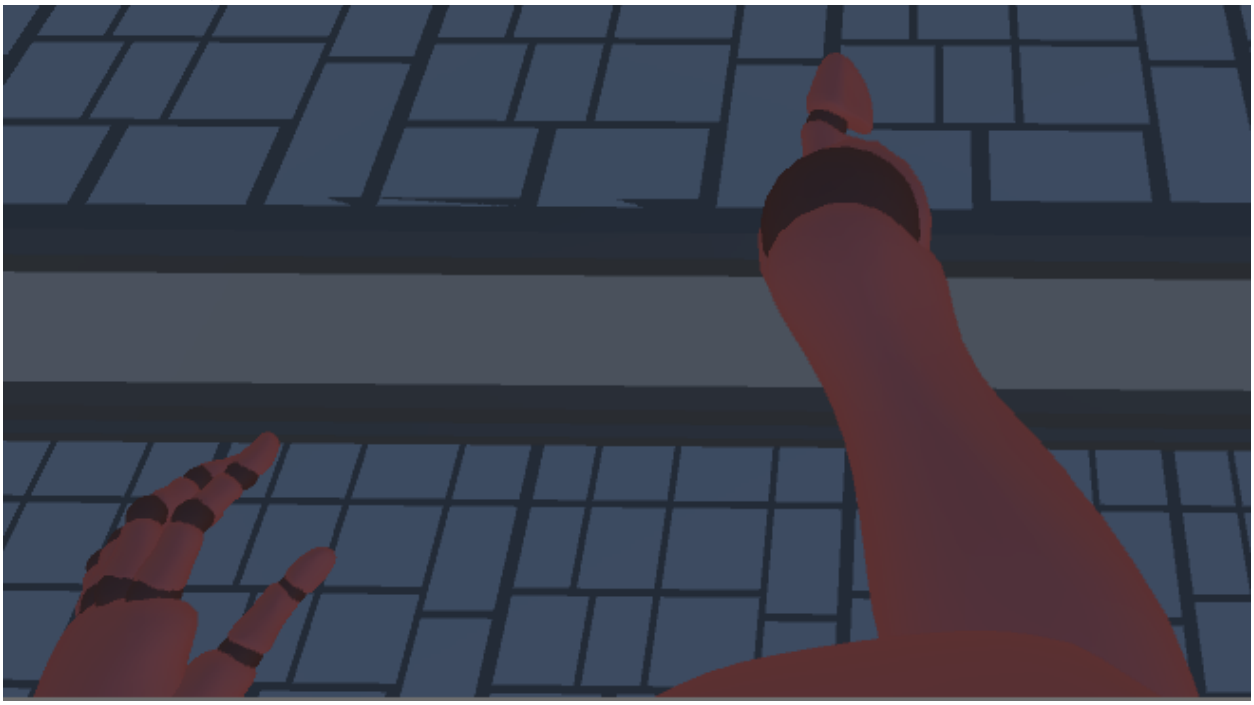
Screenshots

Main menu

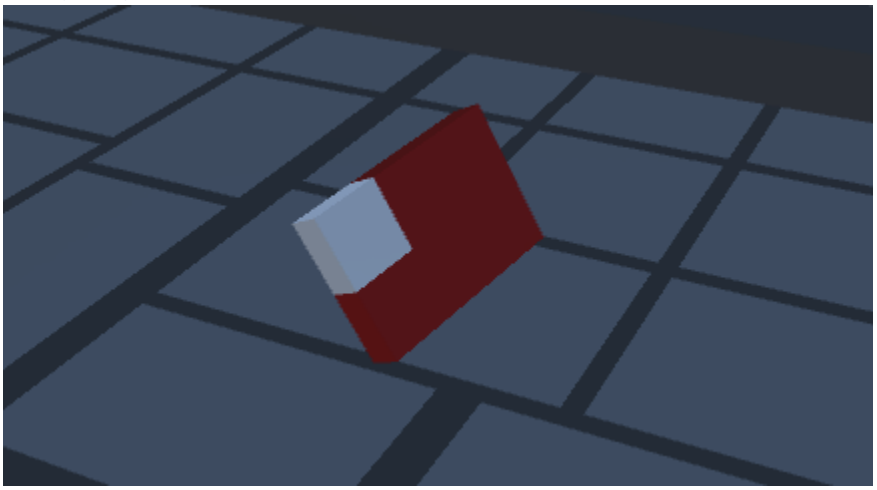
End screen

Dialogue

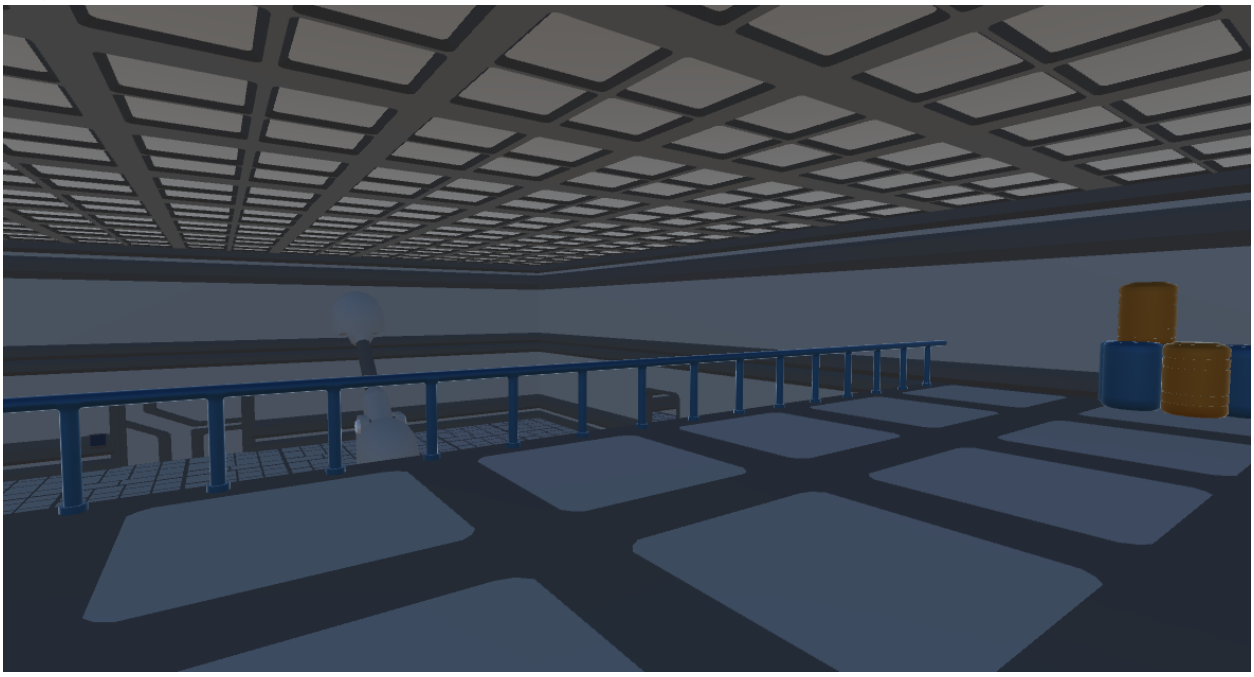
Animations – Jump



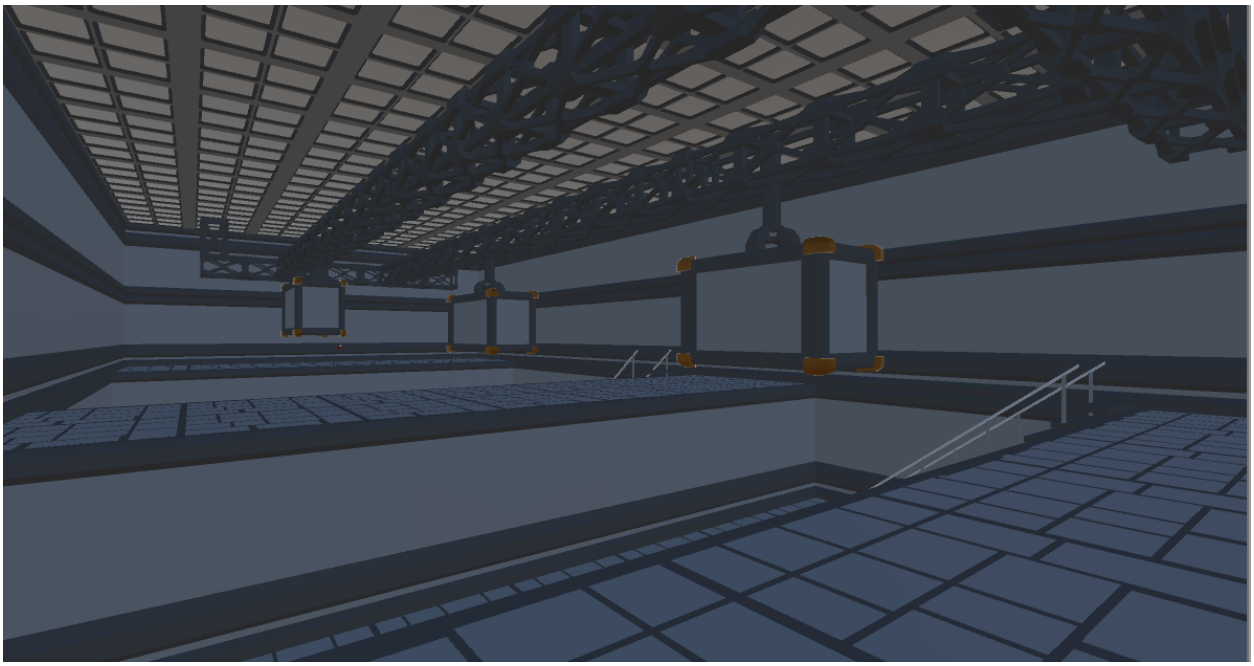
Item pickups



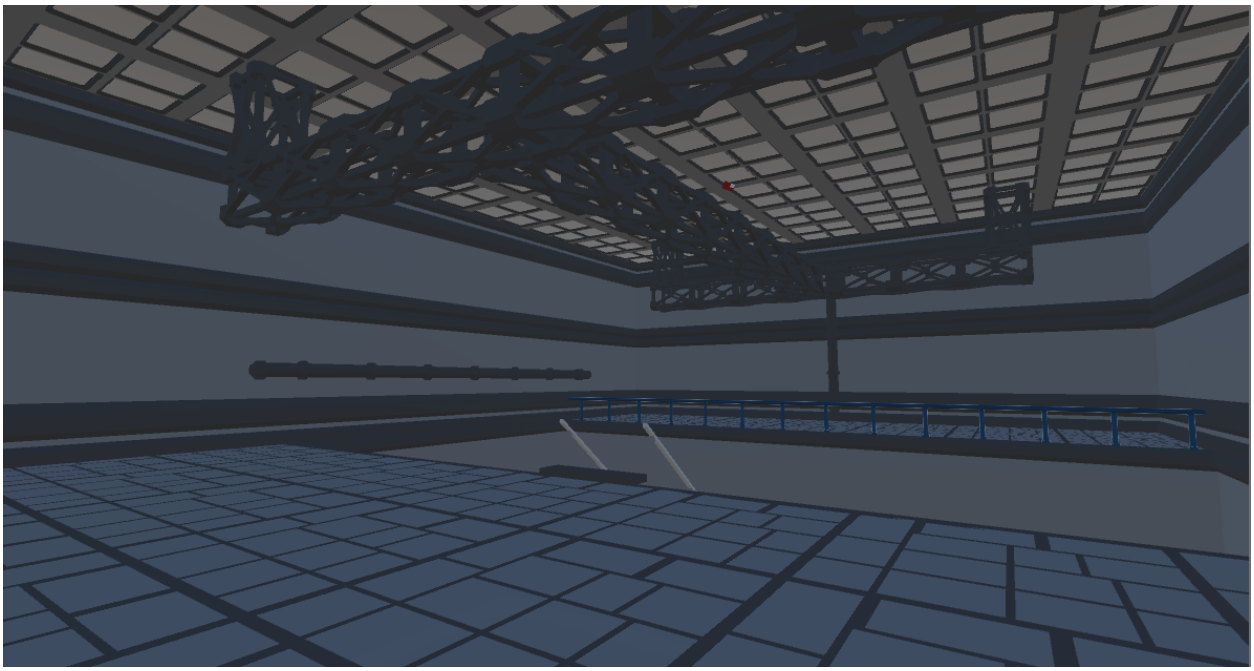
Map – Start



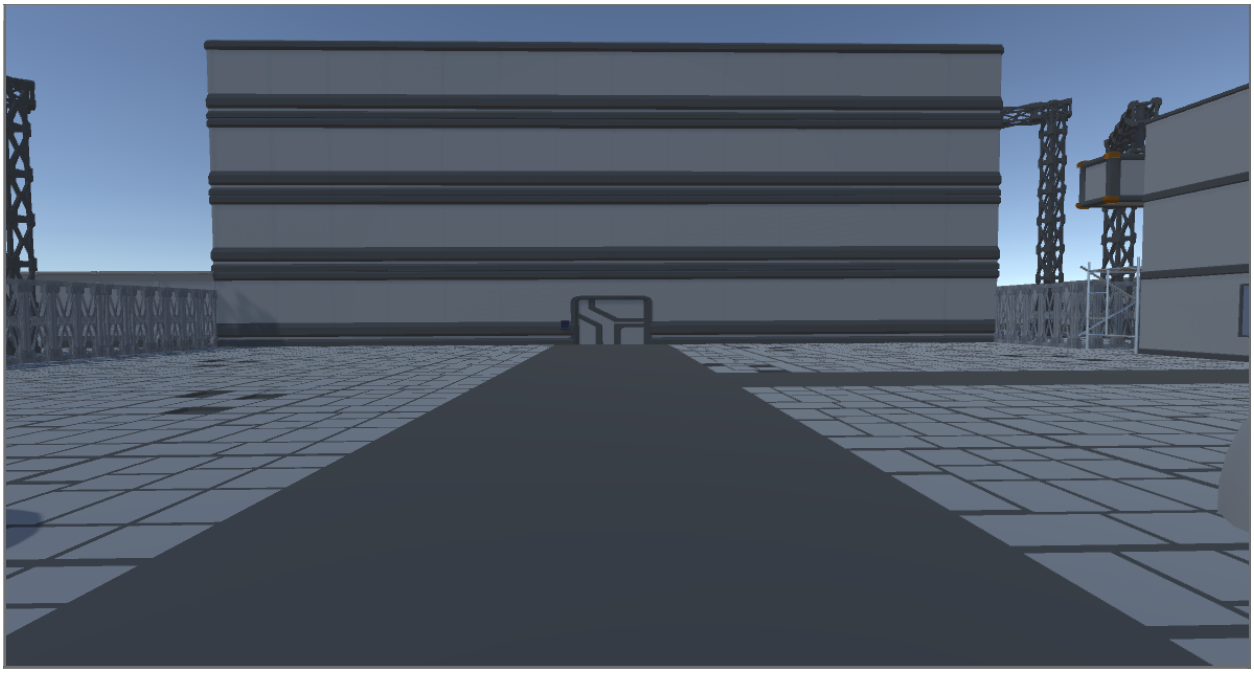
Map – Wallrun tutorial



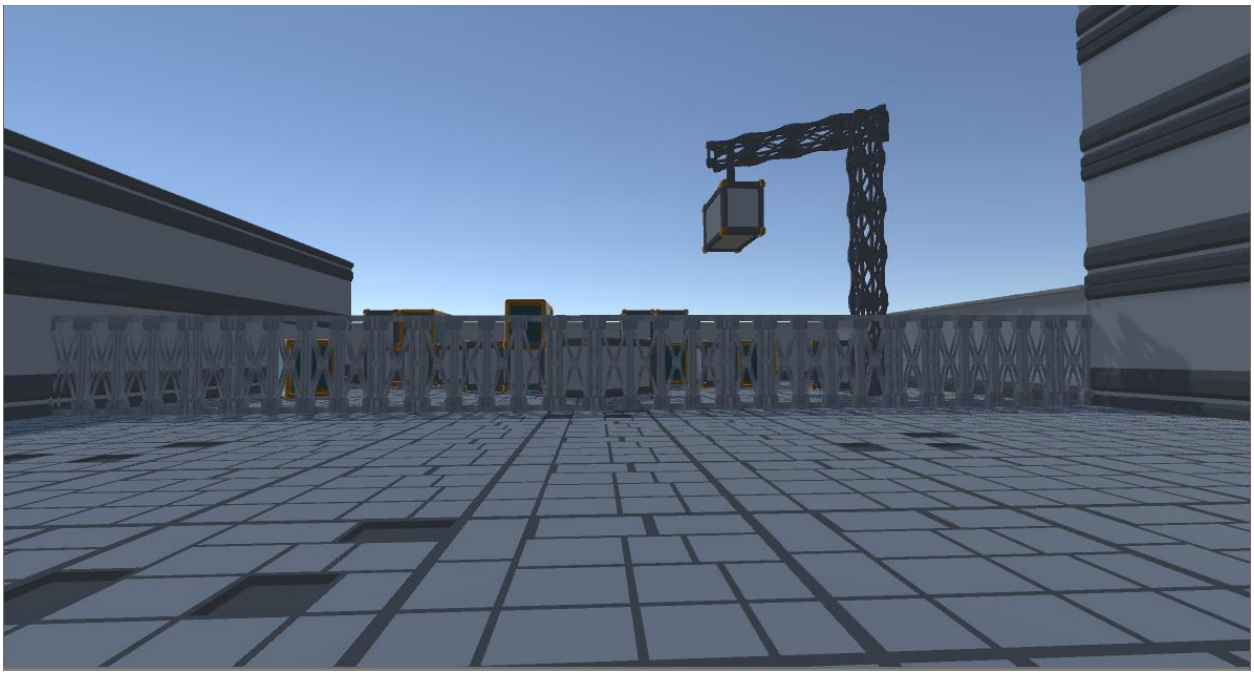
Map – Climbing tutorial



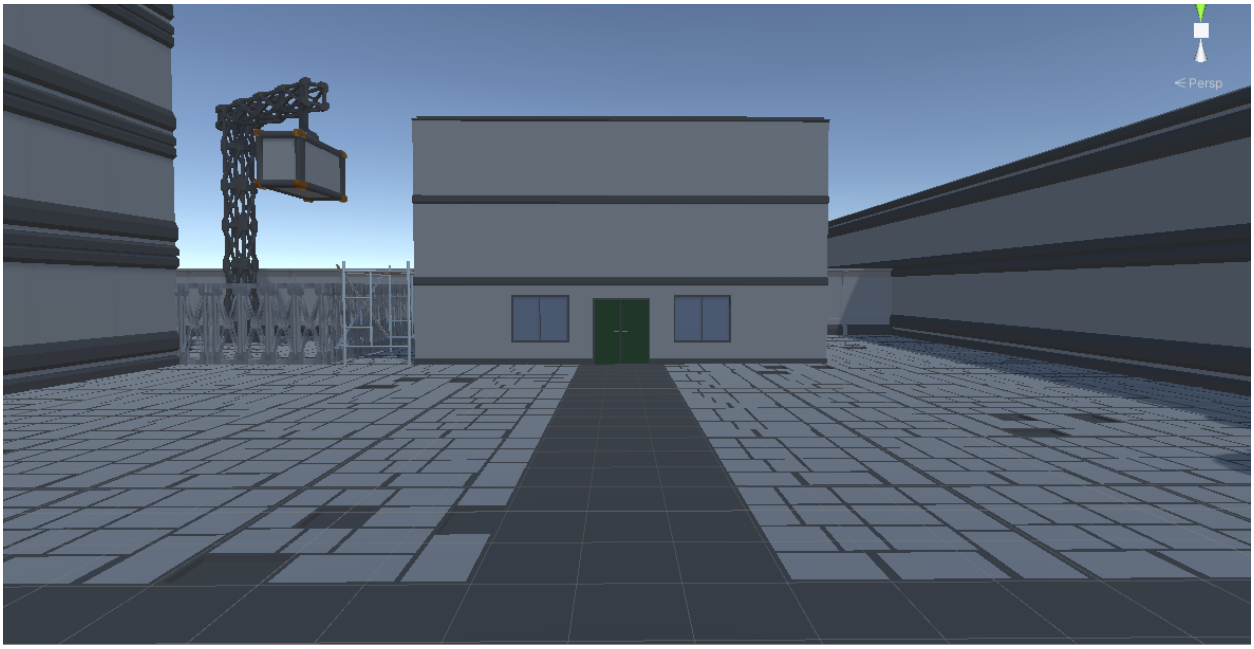
Map – Outside tutorial



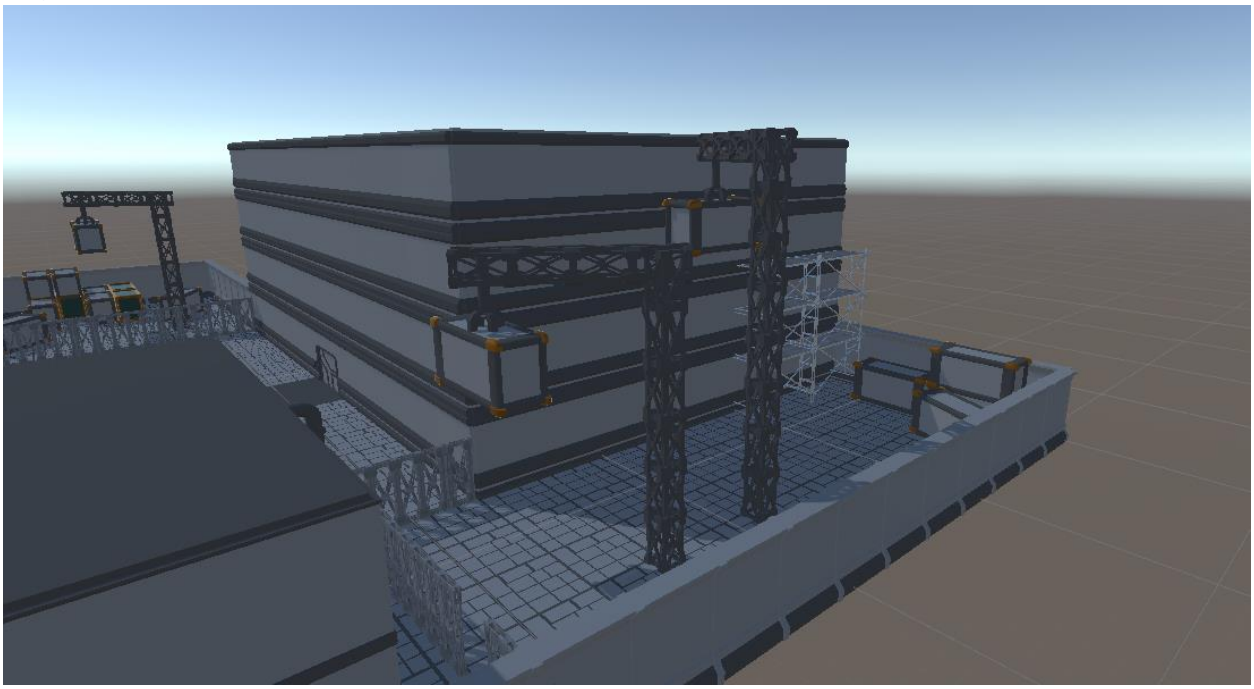
Map – Outside containers



Map – Outside Storage

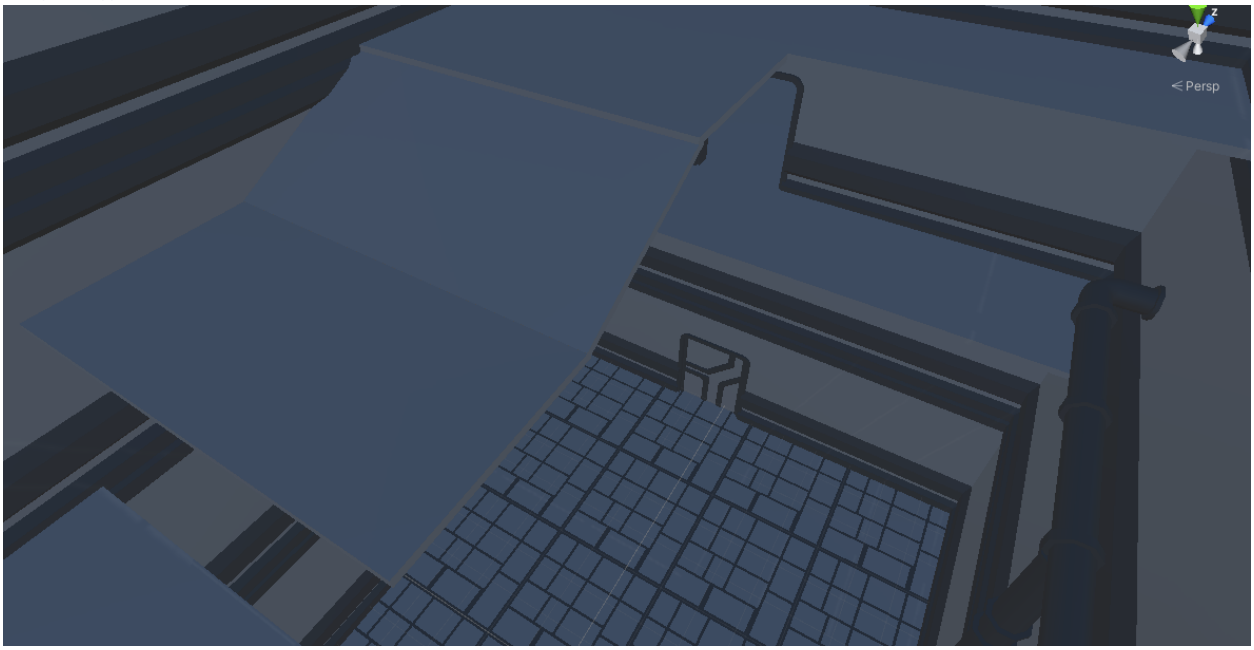


Map – Outside construction site



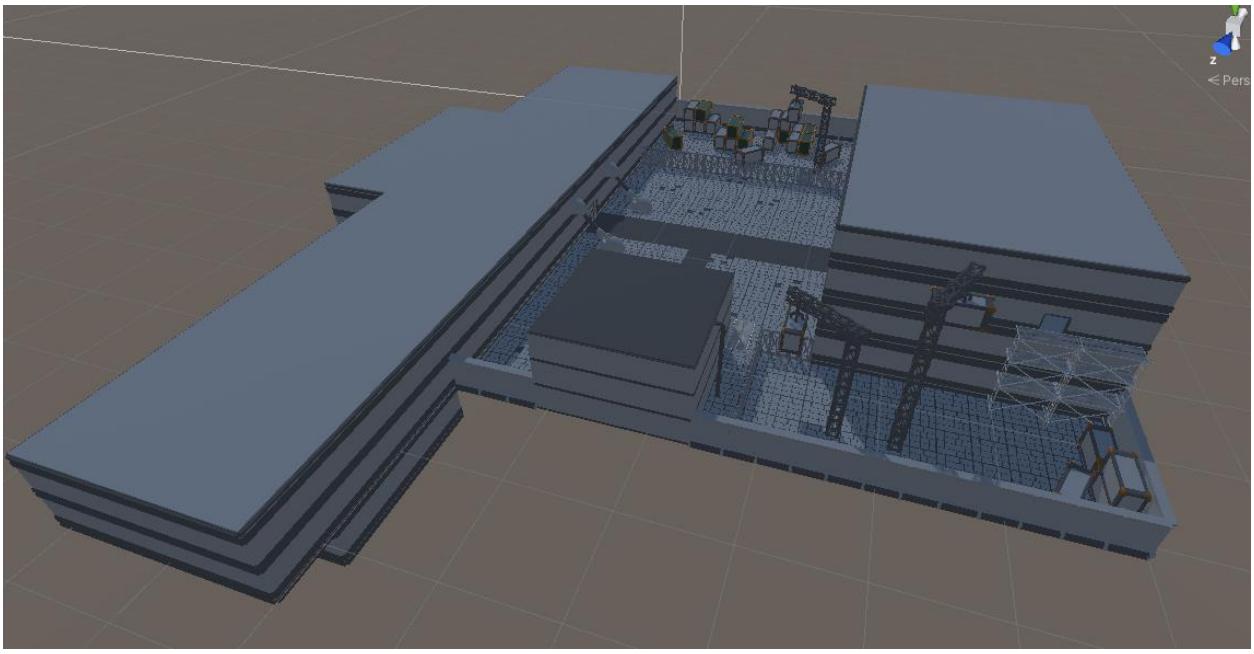
Map – Office room1

Map – Office room3

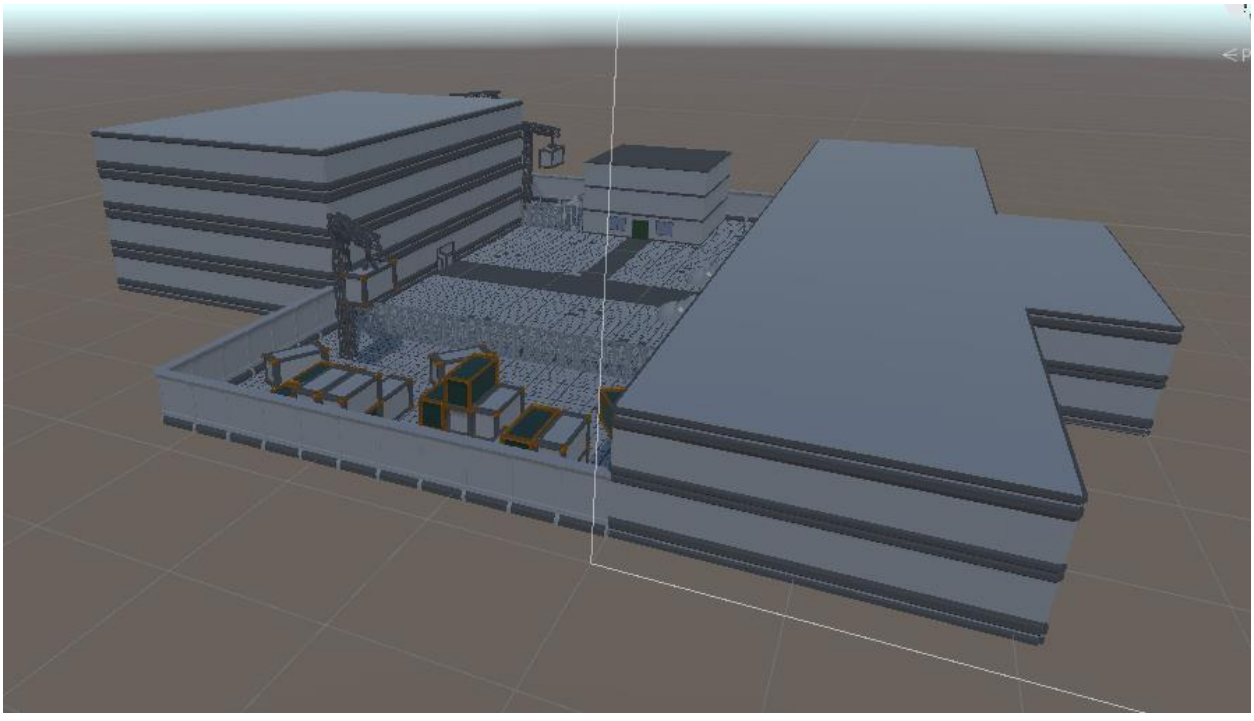


Map – Skyview

Map – Skyview left



Map – Skyview right



Tutorials

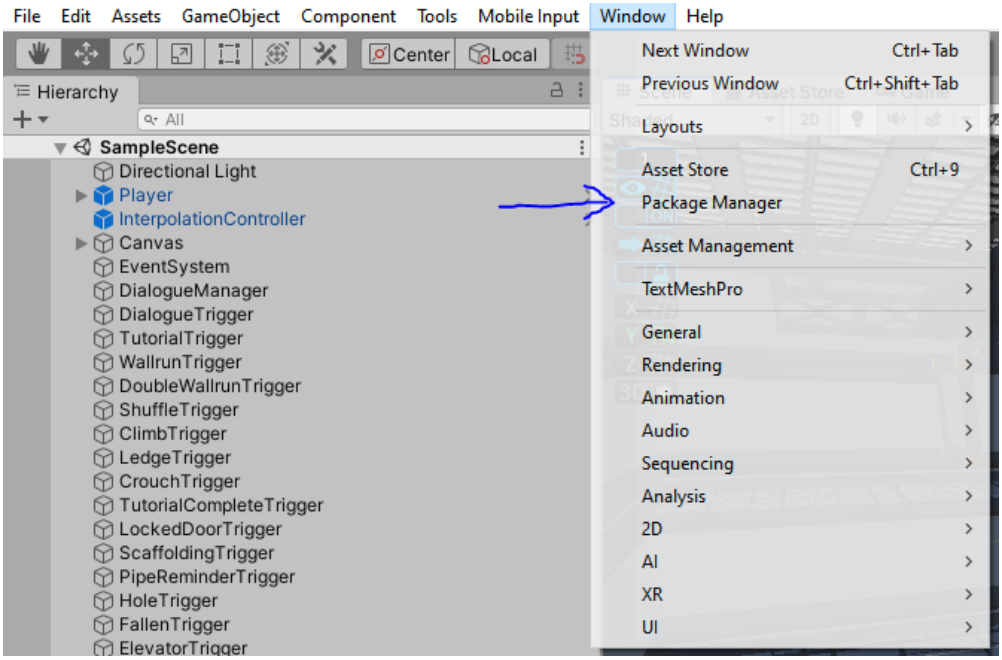
SNAPS

Algemene uitleg

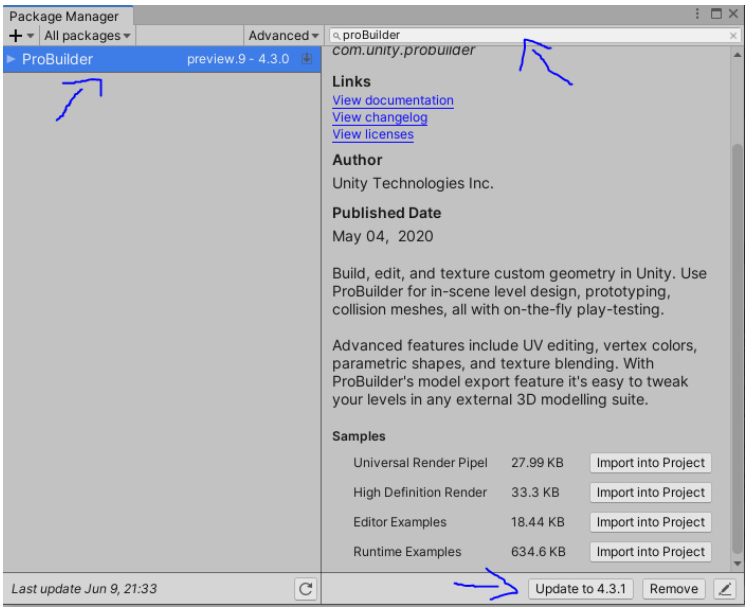
SNAPS is een asset package van unity die je kan gebruiken om gemakkelijk een level te designen in Unity.

Voorwaarden

Vooraleer je aan de slag kan met SNAPS moet je eerst ProBuilder en ProGrids importeren in je project. Dit doe je door bij het tabje “Window” “package Manager” te openen.



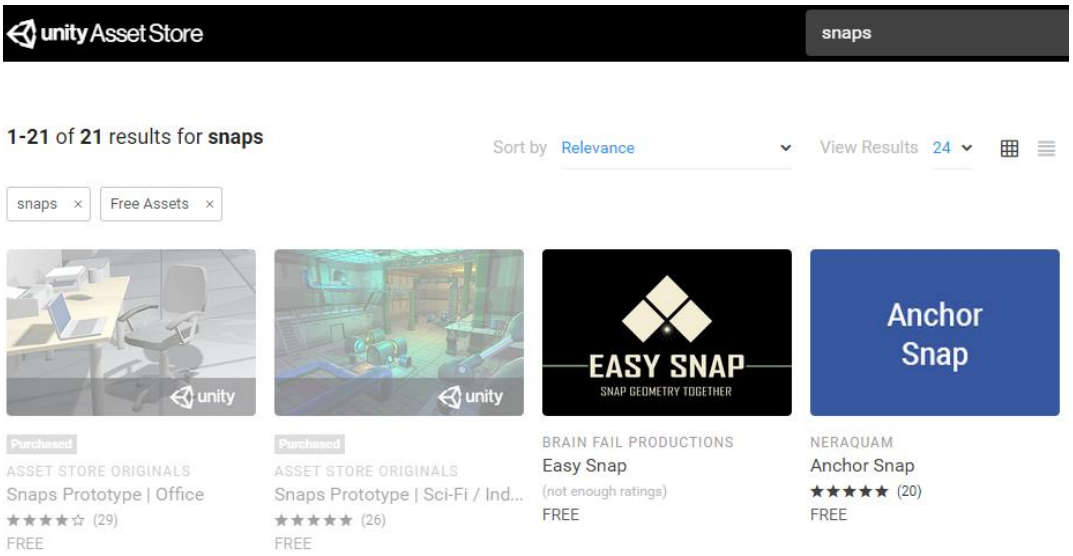
Hier zoek je dan naar “ProBuilder” en “ProGrids” en installeer je deze door op de knop rechts onder te klikken



SNAPS importeren

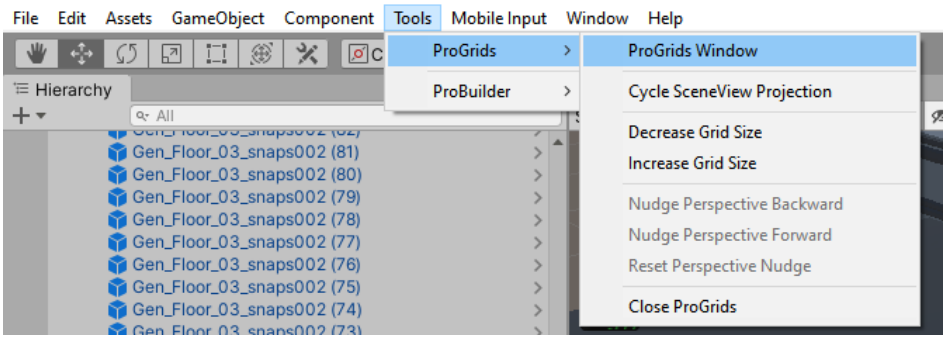
Als je ProBuilder en ProGrids hebt geimporteerd kan je nu aan de slag met SNAPS. Om SNAPS te importeren in je project ga je naar de asset store en zoek je SNAPS. Er zijn heel wat SNAPS assets

beschikbaar dus kies er maar 1 uit (de meeste zijn betalend). In dit project heb ik gebruik gemaakt van de office en Sci-Fi industrial assets.



Beginnen met SNAPS

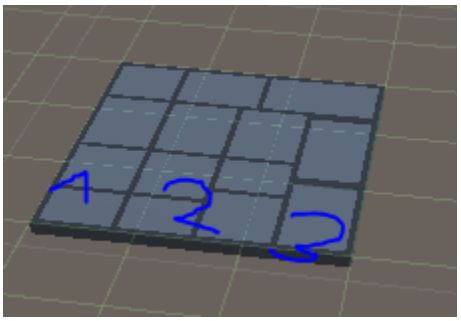
Nu dat we SNAPS assets hebben geimporteerd kunnen we eindelijk aan de slag. Als het ProGrids venster niet zichtbaar is open je deze via Tools>ProGrids>ProGrids Window.



Je zal een klein venster zien in de linker bovenhoek van je scene. Hieronder meer uitleg.

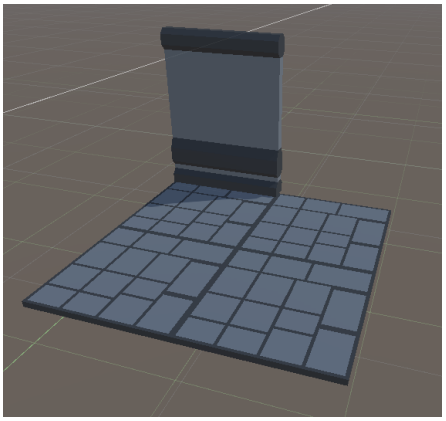
1. Hier stel je in met hoeveel grid vakjes je per keer wilt werken (bijvoorbeeld veplaats met 3 grid vakjes)
2. Hier zet je grid venster aan of uit
3. Hier zet je snapping aan of uit, als je deze uit zet dan heb je de standaard unity werking
4. Hier zet je de geselecteerde objecten op de grid
5. Dit locked de center van de grid
6. Hier laat je de grid op de X as tonen
7. Hier laat je de grid op de Y as tonen
8. Hier laat je de grid op de Z as tonen
9. Hier laat je de grid op alle assen tonen

Nu je weet hoe ProGrid werkt zullen we eens iets maken. Ga naar de SNAPS assets en sleep een vloer in je scene. Als je de vloer hebt geplaatst en je verplaatst deze hierna dan zul je zien dat deze beweegt volgens de grid. Als we nu onze vloer willen uitbereiden kijken we even naar de grid, hier zie je dat de vloer 3 vakjes breed is. Dus stel in dat je met 3 vakjes moet aanpassen (1 hier boven).

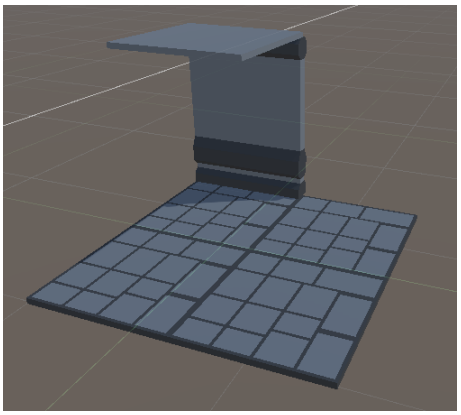


Als je deze vloer nu duplicate (ctrl + d) en deze verplaatst zal je zien dat deze mooi naast de andere vloer komt te staan. Je kan ook meerdere objecten tegelijk selecteren en dupliceren.

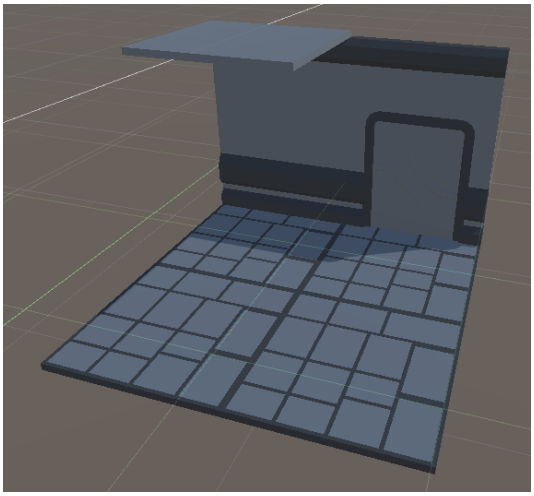
Als je nu een muur wilt plaatsen doe je het zelfde proces. Je plaatst je muur, je kijkt hoeveel blokjes deze in beslag neemt (hier is dat 3) en verplaatst deze naar waar je wilt



Als je nu een dak wilt maken zal je zien dat deze niet mooi uit komt als je deze naar boven plaatst ookal tel je 3 vakjes op de grid. Dit komt omdat we nu in de hoogte werken en dit op een andere grid moeten aflezen. Dus stel de grid in op de X-as (6 hier boven) en klik op de muur, hier zal je zien dat deze 2 vakjes groot is. Stel het in op 2 vakjes en verplaats het dak naar boven, deze zal nu mooi samenkomen met de muur.

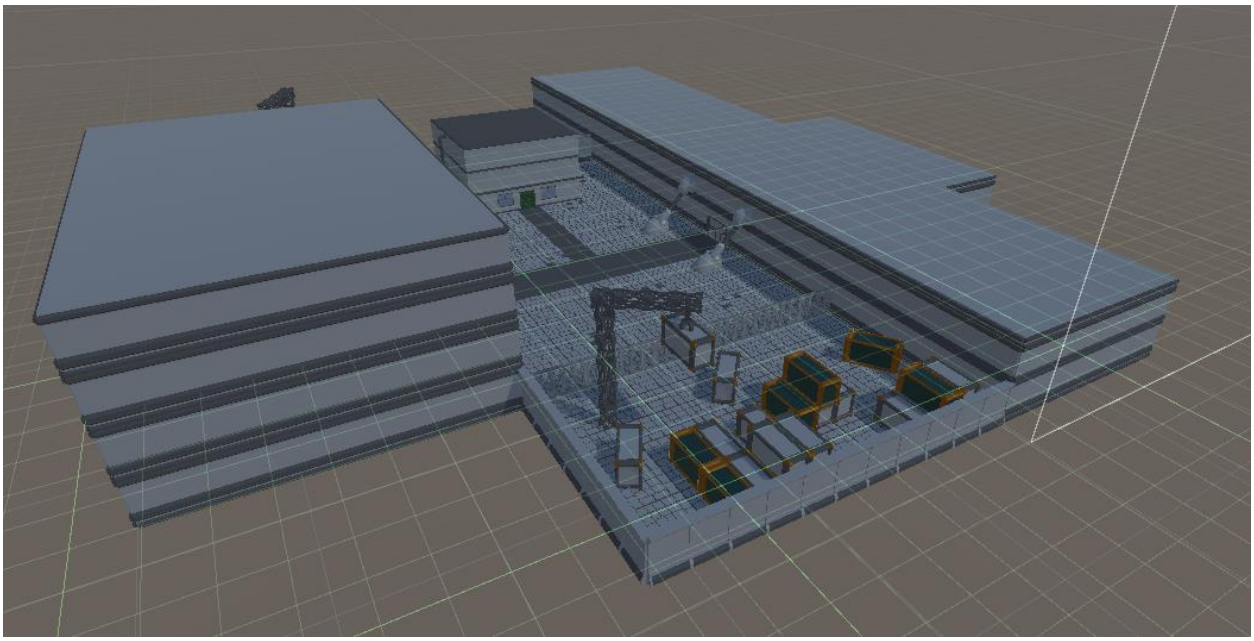


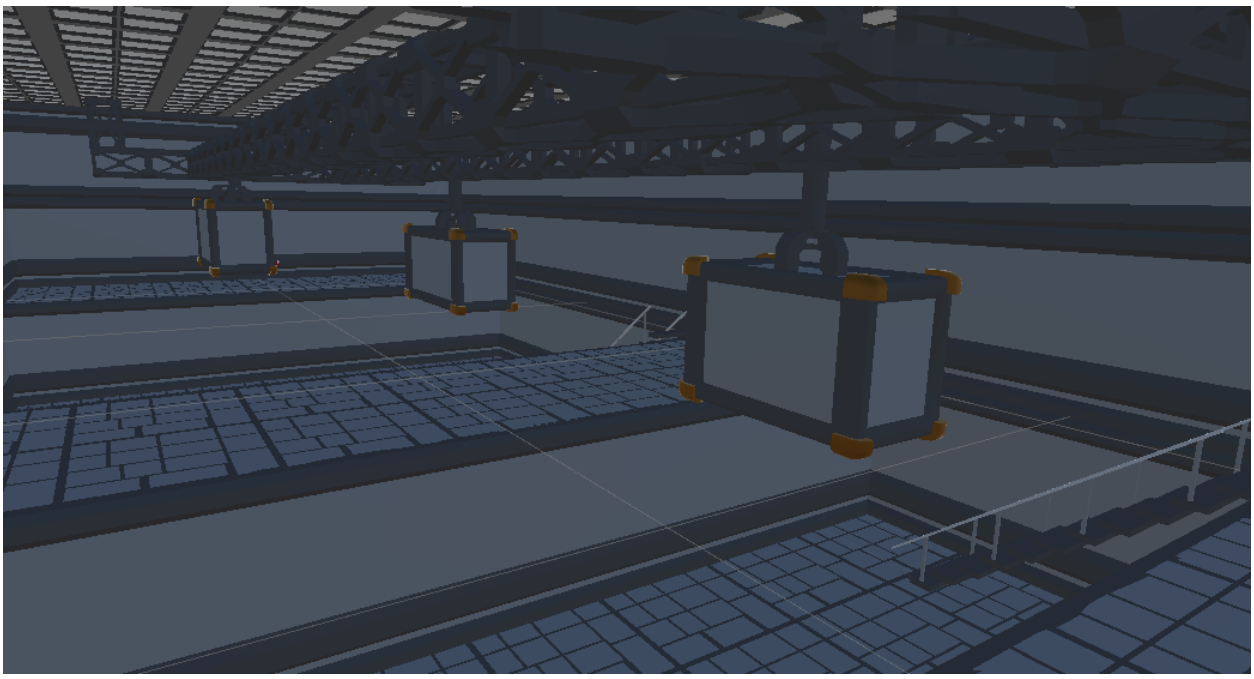
Soms kan het zijn dat je een object niet mooi op zijn plek krijgt met behulp van deze tools. Een voorbeeld hiervan is het plaatsen van een deur. Plaats een muur met een gat voor een deur op dezelfde manier als je een gewone muur plaatst. Dit lukt nog perfect, als je nu de deur in het gat wilt plaatsen zal je zien dat je wat problemen tegen komt. Dit zal je manueel moeten oplossen, maar wees gerust ook hier is een hulpmiddel voor. Eerst zet je de snapping van objecten uit (3 hier boven) om terug de standaard Unity controle te hebben. Nu houd je “v” ingedrukt en sleep je een rand van je deur naar de rand van het gaat, deze zal dan hieraan gesnapt worden. Als dit is gelukt zou het als volgt uit moeten zien.



Soms kan het moeilijk zijn om deze methode te gebruiken, als dit het geval is kan je soms beter even zelf spelen op de ouderwetse manier.

Heel mijn project is gemaakt met behulp van SNAPS





Animations (Mixamo)

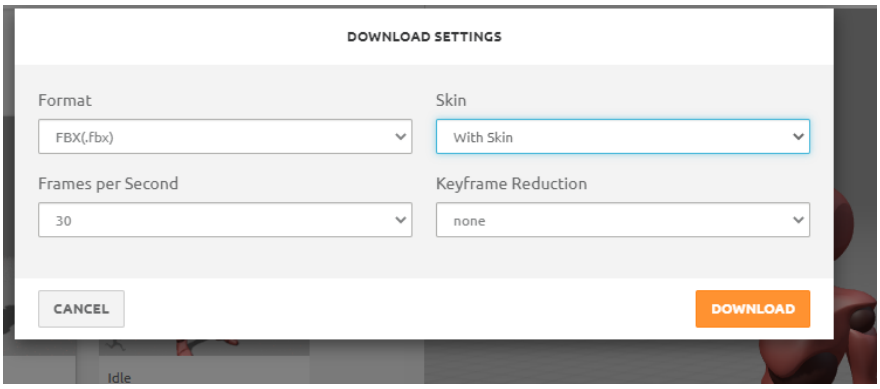
Algemene informatie

Met animations kan je verschillende handelingen van je characters laten animeren. Mixamo is een website waar je animaties kan downloaden. In deze tutorial laat ik zien hoe je deze animaties gebruikt en linkt met elkaar.

Mixamo

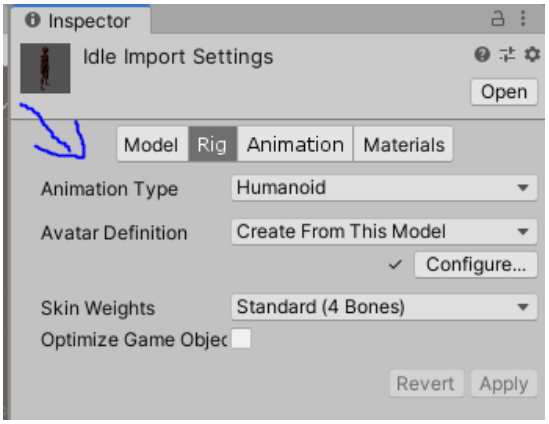
In deze tutorial zal ik 2 animaties downloaden, importeren en linken met elkaar. Ga naar www.Mixamo.com en login, dit is nodig om animaties te kunnen downloaden. Als je ingelogt bent kies je

2 animaties, ik gebruik een idle en een walking animation. Als je een animatie hebt gevonden zal deze rechts spelen met een paar instellingen, standaard zijn instellingen goed maar je kan hier mee spelen naar je eigen verlangen. Als je op download klikt krijg je een venster, laat alles op default staan buiten “Skin” zet deze op “without skin”. Als je zelf nog geen character hebt kan je bij je eerste animatie de “with skin” optie aanhouden, dan krijg je ineens een character mee met je animaties.

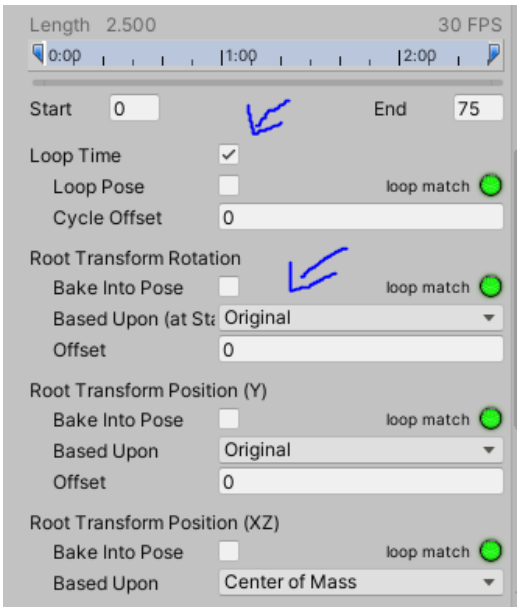


Mixamo animaties importeren

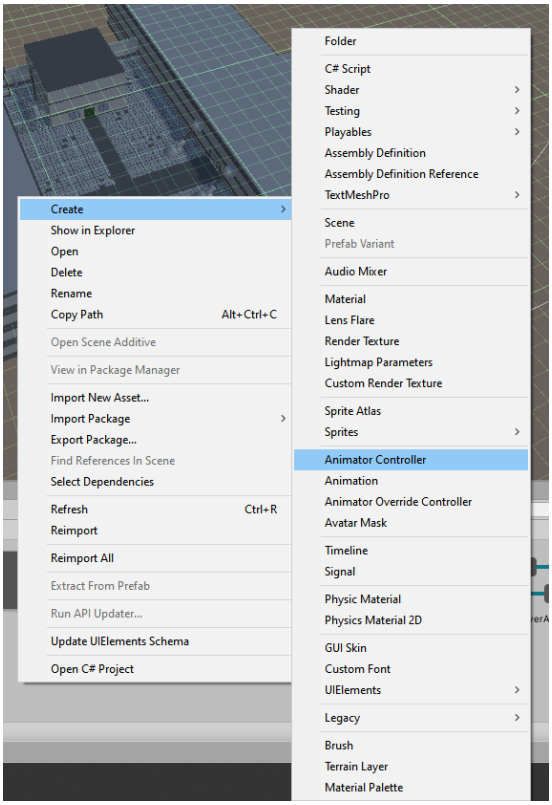
Maak een map aan in je project en sleep je animaties hier in. Je animaties kunnen nog niet gebruikt worden, maar dat is snel opgelost. Klik op 1 van je animaties en in de inspector ga je naar het tabje “Rig” hier verander je “Animation Type” naar “Humanoid” en klik je op apply.



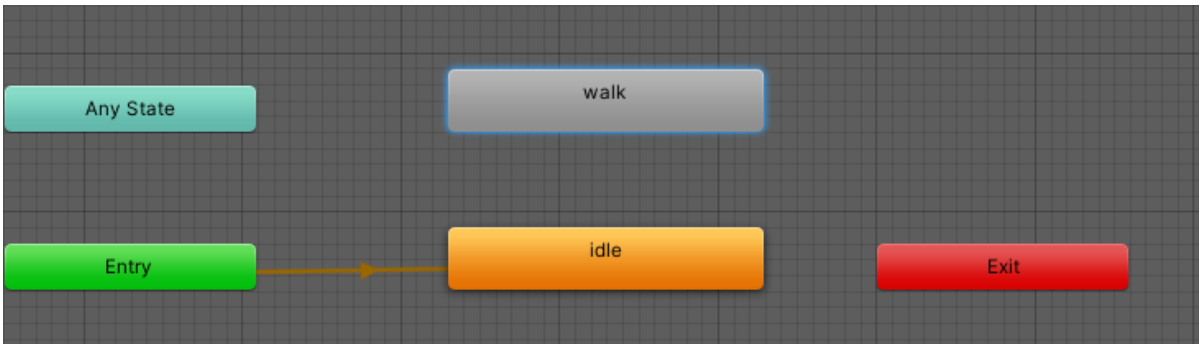
Hierna ga je naar het tabje “Animation” en scroll je naar onder. Hier zet je bij “Root Transform Rotation” de “Based upon (at start)” op “Original”. Hier kan je ook aanduiden ofdat de animatie moet loopen, voor beide gevallen is dit het geval dus duid deze aan.



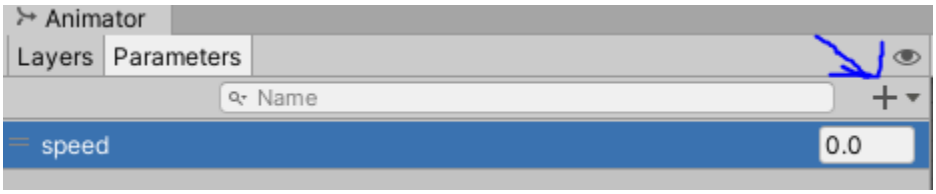
Doe dit voor beide animaties. Als dit gebeurd is maak je in deze map ook een Animator Controller, deze staat bij rechter klik>Create>Animator Controller.



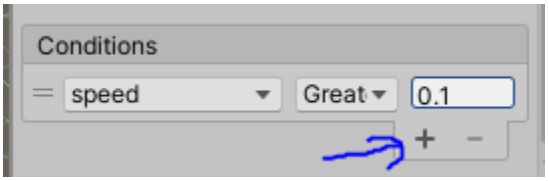
Als je deze opent zul je 3 states zien, hier moet je niets mee doen. Sleep je animaties die je net hebt gedownload in deze window. Stel nu de default animatie in (bijvoorbeeld idle) door hier op te rechter klikken en te klikken op “set layer as default state”.



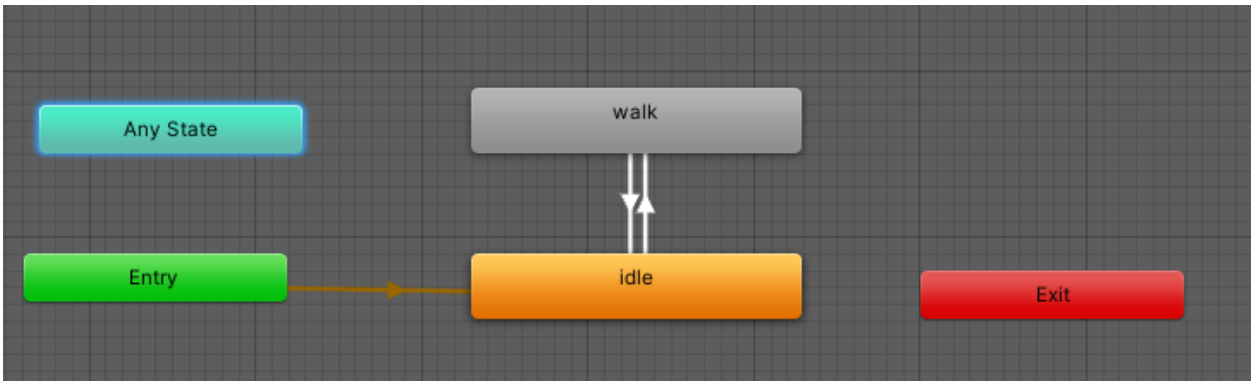
Technisch gezien zijn onze animaties klaar, maar walk wordt nooit aangesproken omdat deze niet gelinkt is. Door dit te doen maak je links eerst een nieuwe variable dit kan een boolean zijn of een float (een float kan handig zijn als je later ook nog een animatie voor rennen wilt toevoegen).



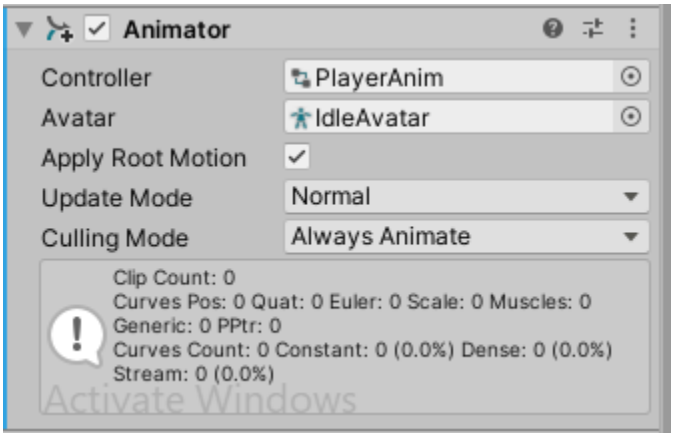
Als je je variable hebt aangemaakt klik je met je rechter muisknop op de idle animatie en kies je “Make transition” en klik je op de walk animatie. Je zal een pijl te voorschijn zien komen, klik hier op. Hier kan je een preview zien van de 2 gelinkte animaties. Maar dat is niet het enige dat we hier kunnen doen, je kan ook nog een conditie meegeven voor wanneer de volgende animatie afgespeeld moet worden. Voeg bij “conditions” een nieuwe condition toe, dit zal speed zijn. Zet de waarde van deze conditie op “greater” en “0.1”.



Doe nu het zelfde in de tegengestelde richting maar inplaats van “greater” “0.1” bij de conditie geef je “lesser” “0.1” in. Je bent nu klaar met je animator, deze zou er als volgt moeten uit zien.



Je animaties zijn nu volledig klaar, maar je hebt nog geen geen manier om deze aan te spreken. Om dit te doen klik je op speler en voeg je een nieuw Animator component toe. Hierin sleep je jouw Animator Controller en avatar.



Als je de avatar wilt gebruiken die je net hebt gedownload ga je naar je animatie en klik je op het pijltje naast de afbeelding. Hier zal je je avatar zien, sleep deze naar je Animator.

Nu willen we nog dat de animaties afspelen wanneer ze moeten afspelen, hiervoor open je het script waarin je de beweging van je character afhandelt. Hier maak je een nieuwe variable aan van het type Animator en geef je in je start methode de Animator mee aan deze variable.

```
Animator animator;
```

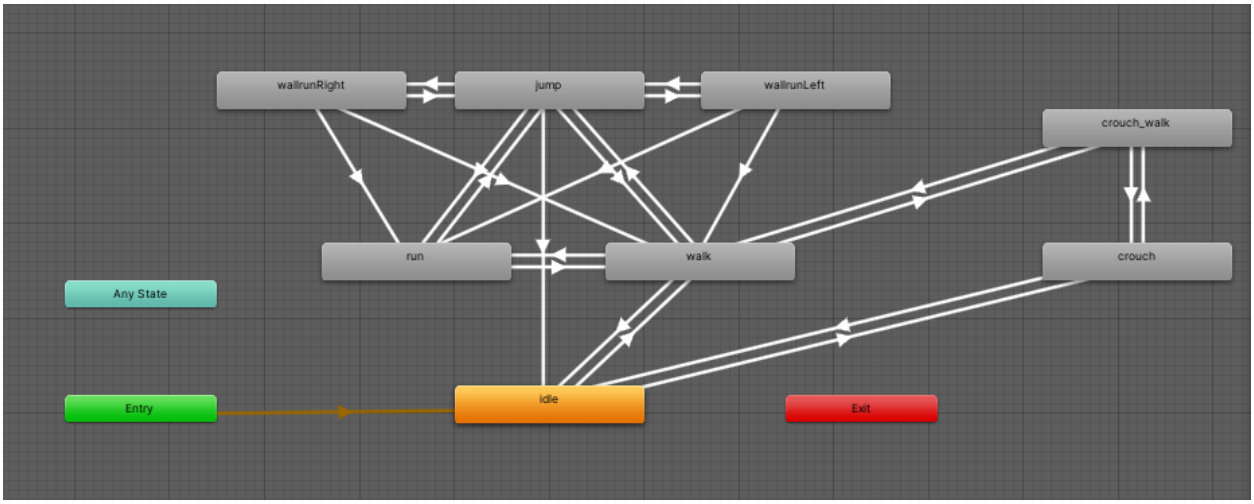
```
private void Start()
{
    CreateVaultHelper();
    playerInput = GetComponent<PlayerInput>();
    animator = gameObject.GetComponent<Animator>();
```

Nu ga je naar de code die er voor zorgt die je character laat lopen. Als je character aan het lopen is moet je de variable van de animator aanpassen naar 1 met behulp van .SetFloat() en als deze stil staat moet deze 0 worden. In mijn geval ziet dit er zo uit.

```
if ((int)status <= 1 || isSprinting())
{
    if (playerInput.input.magnitude > 0.02f)
    {
        animator.SetFloat("speed", 1);
        ChangeStatus((shouldSprint()) ? Status.sprinting : Status.walking);
    }
    else
    {
        animator.SetFloat("speed", 0);
        ChangeStatus(Status.idle);
    }
}
```

Als je alles hebt gevolgd zou alles nu klaar moeten zijn, start je spel en beweeg. Je zal zien dat je animaties aan het spelen zijn.

In mijn project ziet mijn Animator Controller er zo uit.



Dialogue

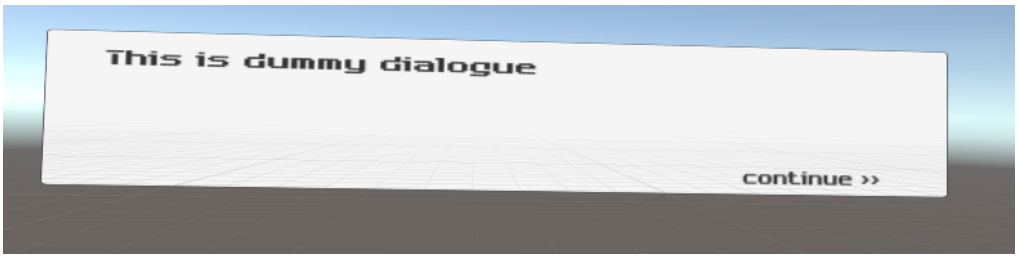
Algemene informatie

Dialogue boxes zijn dozen die gebruikt worden om characters te laten praten in tekstvorm. In mijn project worden Dialogue boxes gebruikt voor de narrator en de speler.

Je kan op veel verschillende manieren dialogue boxes maken en hiermee interacten. In deze tutorial laat ik een Dialogue box zien die wordt opgeroepen als de speler in een bepaalde area gaat en kan deze terug weg doen door op “enter” te klikken.

Het maken van een dialogue box

Eerst en vooral maak je een canvas, in deze canvas maak je hoe je de dialogue box er uit moet zien. Ik gebruik een Image met daarin 2 textboxes, 1 textbox voor de tekst die gezegd zal worden en 1 textbox die de tekst “continue >>” afprint.



Dialogue

Ga nu naar je map met al je scripts en maak een nieuw script aan genaamd “Dialogue”. Doe alles binnen deze class weg en maak een array van strings waarin je zinnen zullen komen. Ook zet de je deze class op Serializable.

```
[System.Serializable]
2 references
public class Dialogue
{
    public string[] sentences;
}
```

DialogueManager

Maak een nieuw script aan en noem deze “DialogueManager”. Voeg volgende variables toe.

```
public Text dialogueText;

public Animator animator;

private Queue<string> sentences;

private GameObject trigger;
```

In je start methode geef je de variable sentences volgende waarde. Een queue is een soort list die via een FIFO manier werkt.

```
void Start()
{
    sentences = new Queue<string>();
}
```

In de update methode check je ofdat de speler de “enter” key heeft ingedrukt, als dit zo is zal de volgende zin getoond worden.

```
private void Update()
{
    if (Input.GetKeyDown(KeyCode.Return))
    {
        DisplayNextSentence();
    }
}
```

Schrijf volgende methode. Deze methode zorgt er voor dat de dialogue gestart zal worden. Het zal de meegegeven zinnen in de queue steken en ook de zinnen die er al inzaten verwijderen. Een GameObject wordt hier gevraagd, hier komt later meer uitleg over.

```
public void StartDialogue(Dialogue dialogue, GameObject gameObject)
{
    trigger = gameObject;

    animator.SetBool("isOpen", true);

    sentences.Clear();

    foreach (string sentence in dialogue.sentences)
    {
        sentences.Enqueue(sentence);
    }

    DisplayNextSentence();
}
```

Voeg de volgende methode toe. Hierin wordt gezien of er nog zinnen zijn en worden deze dan toegevoegd aan de Dialogue box.

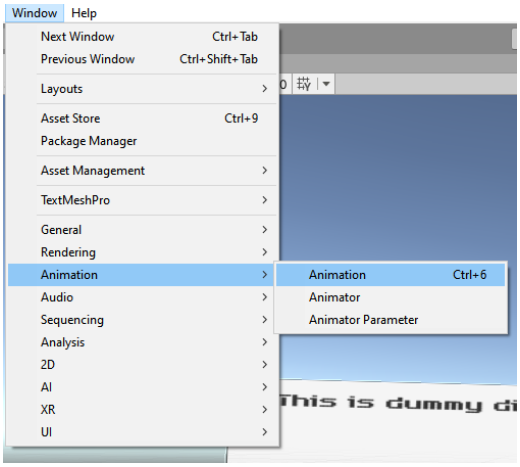
```
2 references
public void DisplayNextSentence()
{
    if (sentences.Count == 0)
    {
        EndDialogue();
        return;
    }

    string sentence = sentences.Dequeue();
    dialogueText.text = sentence;
}
```

Tot slot voeg je deze methode nog toe. Deze zal het gesprek eindigen en het GameObject verwijderen die we daarstraks hadden meegegeven . Dit doen we zodat een dialogue niet meerdere keren opgeroepen kan worden.

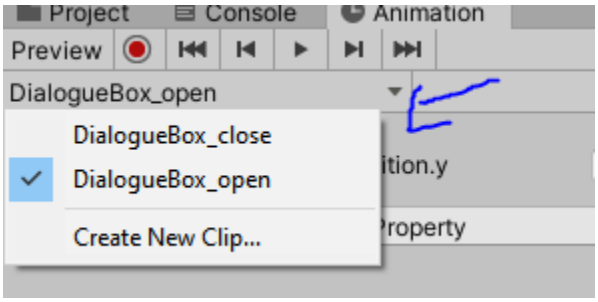
```
void EndDialogue()
{
    animator.SetBool("isOpen", false);
    Destroy(trigger);
}
```

Animaties

Zoals je kon zien gebruiken we hier ook een Animator, deze gebruiken we om de dialogue box te openen en te sluiten. Als we dit niet doen blijft deze altijd open staan. Ga naar de MessageBox en open de Animation tab deze kan je vinden onder “Window”.

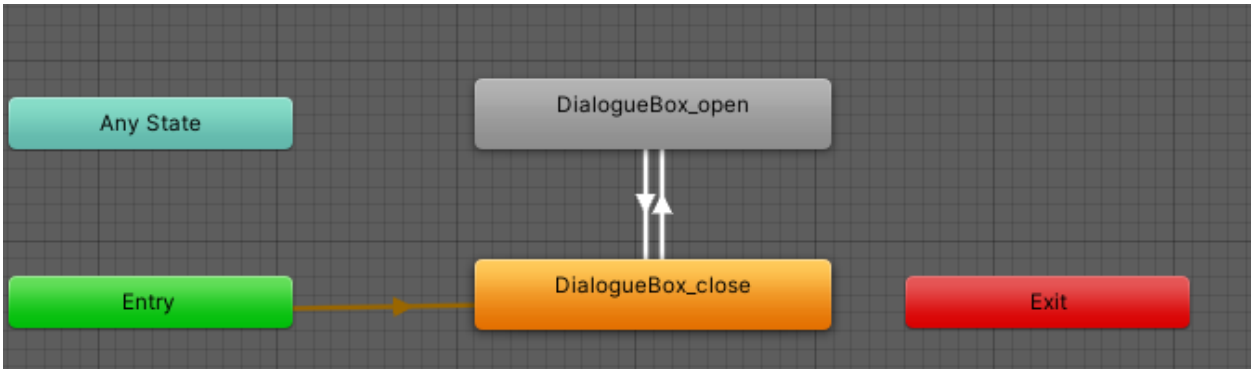


In de Animation maak je een nieuwe clip en noem je deze “DialogueBox_open”. Klik op record en kopieer de Y-waarde van de message box en sleep deze waarde zodat er een keyframe gemaakt word, stop hierna de recording. Plak de originele Y-waarde weer op zijn plaats.

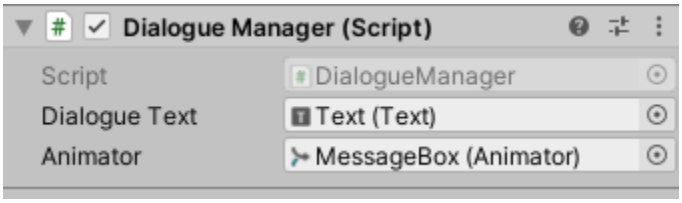


Maak nu een nieuwe clip en noem deze “DialogueBox_close”. Klik op record en sleep de MessageBox in een rechte lijn naar beneden zodat deze nu buiten het canvas valt en stop dan de recording. Sleep de MessageBox terug op zijn originele positie als dit niet vanzelf gebeurd.

Open nu de Animator tab, je zal een bekend venster te zien krijgen. Omdat we animaties in een andere tutorial bespreken ga ik hier niet lang bij stil staan. Zet de close animatie op default door hierop te rechterklikken en “Set as Layer Default state” aan te duiden. Maak nu links nieuwe variable, namelijk een boolean genaamd isOpen. Rechterklik op de close animatie en selecteer “make State” en klik dan op de open animatie. Klik dan op de pijl en maak een nieuwe conditie, namelijk “isOpen” en “true”. Doe nu hetzelfde in de tegengestelde richting maar zet de conditie op false.



Nu maken we in ons project een nieuw leeg object. En noem deze DialogueManager. Hierin sleep je het script dat je net hebt gemaakt en geef je de variables hun waardes. “Dialogue text” is de textbox waarin je tekst moet komen en “Animator” is de animator die op de MessageBox staat, deze is automatisch aangemaakt toen we onze animaties hebben gemaakt. Sleep dus gewoon de MessageBox hier in.



DialogueTrigger

Omdat we in dit voorbeeld willen dat een dialogue box wordt opgeroepen wanneer de speler in een bepaalde area staat maken we een nieuw script aan genaamd “DialogueTrigger”.

Hier maak je volgende variable.

```
public Dialogue dialogue;
```

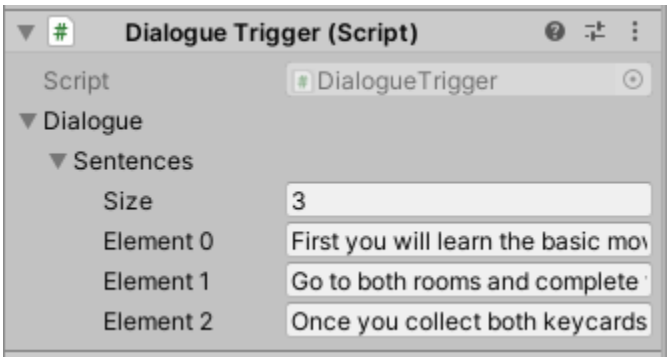
Schrijf volgende methode. Hier wordt er gecheckt ofdat de speler in bepaalde area stapt.

```
private void OnTriggerEnter(Collider player)
{
    if (player.gameObject.tag == "Player")
    {
        TriggerDialogue();
    }
}
```

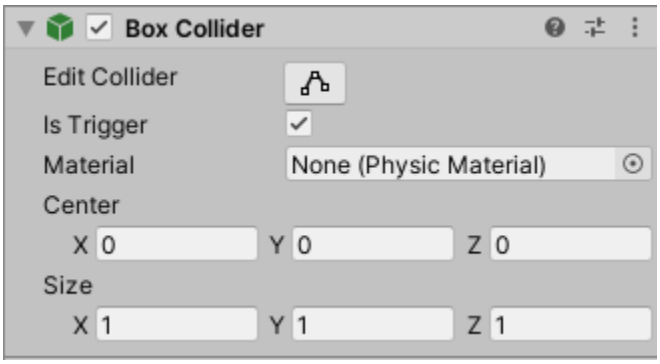
Als laatste methode schrijfe je deze methode. Hier zal de dialogue gestart worden door de dialogue en GameObject mee te geven aan de StartDialogue methode van de DialogueManager.

```
1 reference
public void TriggerDialogue()
{
    FindObjectOfType<DialogueManager>().StartDialogue(dialogue, this.gameObject);
}
```

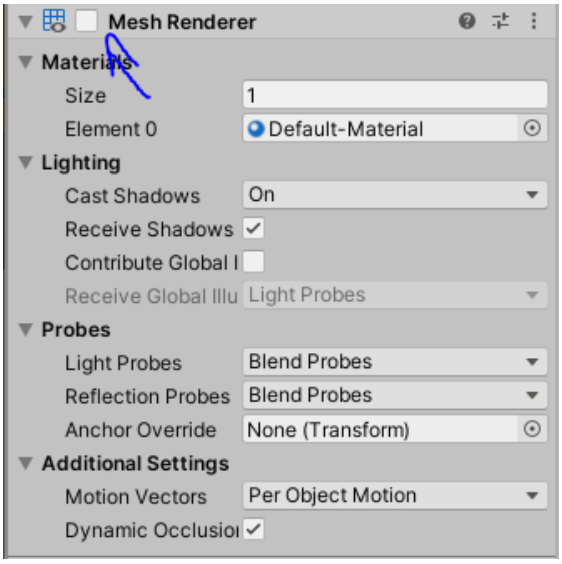
Nu we alle klassen hebben gemaakt moeten we nog de area maken waarin een Dialogue getriggered kan worden. Maak in je project een nieuwe Cube en noem deze naar de actie die lijdt naar het triggeren van de dialogue om makkelijk de trigger terug te vinden. Ik noem de mijne TutorialTrigger. Aan deze Cube geef je het script “DialogueTrigger”. Hier geef je in hoeveel zinnen er getoond zullen worden en welke dit zijn.



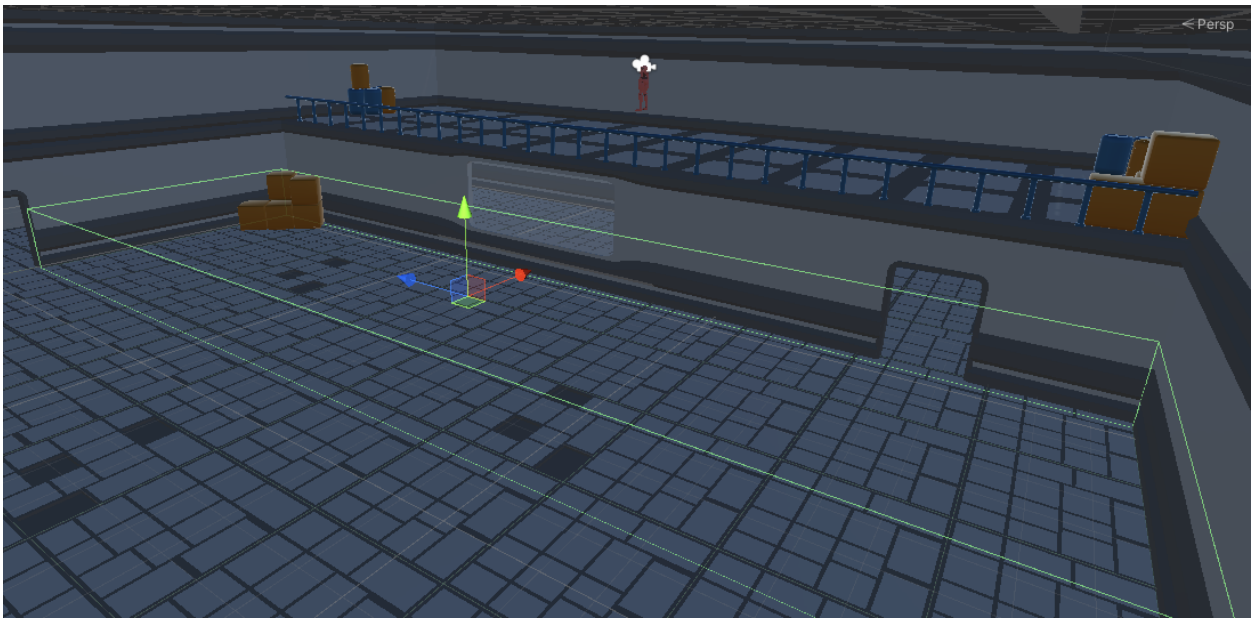
Nu zet je de Box Collider als Trigger zodat de speler hier door kan lopen en het script effectief geactiveerd word.



Als volgt zet je de Mesh Renderer uit zodat de speler deze Cube niet kan zien.



Als laatste stap pas je de grootte van de Cube aan zodat de dialogue zeker getriggerd wordt. In mijn voorbeeld zal de dialogue getriggered worden als de speler naar beneden springt.



Als je alles hebt gevolgd zou je Dialogue box moeten werken.