TOUW MECHANIC

Matthias van Herk

Inhoudsopgave

Inleiding

Er is gewerkt aan een touw mechanic. De speler kan op een knop drukken tijdens het klimmen om zich te laten vallen en aan een touw te hangen. Het doel van deze mechanic is om de speler een optie te geven om snel een aanval te ontwijken. Er wordt verwacht dat dit bijdraagt aan de “challenge” in het spel.

Methodes

1. Definiëren: Vaststellen van de benodigdheden
   - Type: eisen analyseren
   - Omschrijving: Vaststellen wat de feature omvat en wat niet.
   - Bron: user-story, delegate product owner
2. Analyseren: brainstormen voor toepassingsmethodes
   - Type: brainstormen/voor- en nadelen bedenken.
   - Omschrijving: verschillende toepassingsmethodes bedenken en overwegen welke het beste is in de situatie.
3. Implementatie: feature programmeren
   - Type: game implementatie
   - Omschrijving: Het programmeren van het touw zodat de speler een stukje valt tijdens het klimmen en daarna blijft hangen.

Uitvoering

1. Definiëren

Voor het vaststellen van de eisen voor de feature is in eerste instantie gekeken naar de eisen die gesteld zijn in de user-story. Verder is er bij sommige eisen om verduidelijking gevraagd bij de delegate product owner. Hieruit komen de volgende eisen:

1. **Rechtermuisknop input:** De speler kan op rechtermuisknop drukken tijdens het klimmen om aan een touw te hangen.
2. **Touw object:** Er is een touw dat de speler verbindt aan het originele punt van de actie. Dit touw beweegt mee met de positie van de speler.
3. **Zwaartekracht:** De zwaartekracht van de speler wordt aangezet.
4. **Touw krachten:** De speler is gelimiteerd in hoever deze valt door de illusie van een touw te creëren. Het touw veert mee met de speler.
5. **Slingeren:** De speler kan slingeren terwijl deze aan het touw hangt.
6. **Actie beëindigen:** De speler laat los van het touw wanneer deze op rechtermuisknop drukt of springt.

2. Brainstormen

Voor de implementatie van het touw zijn verschillende methodes overwogen.

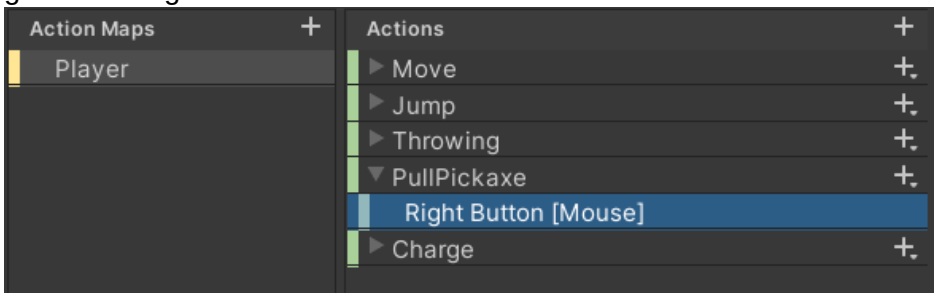
1. Als eerste is er overwogen om het touw een vering te geven. Dit zodat het levendiger en natuurlijker aan zou voelen. Het nadeel kan zijn dat het moeilijk te implementeren is en goed gepolijst moet worden om te voorkomen dat het frustrerend werkt.
2. De tweede optie was om het touw geen vering te geven. Hierbij zou het touw wel buigen als de speler dichter bij het aanmaakpunt kwam, maar de speler zou direct tot stilstand komen wanneer deze voorbij de lengte van het touw probeert te bewegen. Het nadeel zou zijn dat het touw erg statisch kan voelen, wat minder interessant is voor gameplay.

Beide opties zijn geprobeerd en er is uiteindelijk besloten om de eerste optie toe te voegen aan de game, omdat deze zoals verwacht natuurlijker aanvoelt. Verder bleek dit ook makkelijker te implementeren te zijn zonder problemen te creëren. Er zijn hierbij verschillende waardes voor variabelen geprobeerd en er is nu gekozen om de huidige waardes te behouden en te gaan testen. Beide deze methodes zijn zelf bedacht.

3. Implementatie

Stap 1: Action map

1.1 Als eerste moet er in de action map een interactie voor de rechtermuisknop worden toegevoegd. In dit geval was deze al aanwezig, maar nog niet gebruikt. Hier hebben we gebruik van gemaakt.



FIGUUR 1 ACTION MAP

Stap 2: activatie van actie

2.1 Voor het hangen aan een touw is gebruik gemaakt van het eerder geïmplementeerde statensysteem. Er is een staat: “HANGING” toegevoegd (Technische documentatie van Rens).

2.2 Wanneer de speler op de rechtermuisknop drukt wordt “void OnPullPickaxe()” aangeroepen. Deze methode maakt het touw actief en zet de begin positie van het touw naar waar de speler zich bevindt. Hierbij wordt de helft van de speler opgeteld om het touw tegen de muur te laten beginnen.

2.3 Verder wordt het hangpunt van het touw gezet. Dit is het punt wat gebruikt wordt om de krachten van het touw op de speler te laten werken. Dit punt is juist een stuk van de muur afgehaald om te voorkomen dat de speler constant langs de muur schaaft terwijl deze slingert.

2.4 Vervolgens wordt de zwaartekracht aangezet en wordt er een neerwaartse beweging gegeven zodat de speler direct valt en de mechanic kan gebruiken om aanvallen te ontwijken.

2.5 Als laatste wordt de staat naar de nieuwe “HANGING” staat gezet zodat de “Hang()” methode wordt aangeroepen in de “FixedUpdate”. Als de speler al hangt wordt de methode “StopHanging()” aangeroepen.

Problemen:

Er was een probleem waarbij de speler bij het slingeren steeds tegen de muur aan kwam waardoor een groot deel van het momentum verloren ging. Dit is opgelost door het eigenlijke punt waarvan de speler hangt een stuk van de muur af te zetten.

```
void OnPullPickaxe()
{
    if(state == PlayerStates.CLIMBING)
    {
        rope.SetActive(true);
        ropeAttachment = transform.position + transform.forward/2;
        HangPoint = transform.position - transform.forward;
        rb.useGravity = true;
        rb.velocity = new Vector3(0, -10, 0);
        state = PlayerStates.HANGING;
    }
    else if(state == PlayerStates.HANGING)
    {
        StopHanging();
    }
}
```

FIGUUR 2 ONPULLPICKAXE() METHODE

```
case PlayerStates.HANGING:
    Hang();
    break;
```

FIGUUR 3 SWITCH CASE VOOR DE HANGING STAAT IN DE FIXEDUPDATE

Stap 3: code van touw

3.1 In de “Hang()” methode worden verschillende dingen gedaan:

1. “UpdateRope()” wordt aangeroepen. Hierover is in Stap 4 meer te lezen.

2. Er wordt een kracht relatief aan de speler gegeven op basis van de bewegingsinput. Dit geeft de speler de mogelijkheid om te slingeren.
3. De rest van de code zorgt voor de vering en limitaties op het touw.

3.2 Als eerste wordt er gecheckt of de afstand tussen het beginpunt en de speler groter is dan de lengte van het touw.

3.3 Daarna wordt een richting van de kracht gezet. Dit is de richting van de speler naar het beginpunt.

3.4 Vervolgens wordt de veerkracht van het touw gezet. Dit wordt gedaan door de minimumwaarde 1 te geven en hierbij de afstand van de speler tot het beginpunt min de lengte van het touw op te tellen. Dit betekent dat de kracht groter wordt gebaseerd op hoe ver de speler voorbij de lengte van het touw is, wat dus een soort vering creëert die groter is als de speler verder weg is.

3.5 Uiteindelijk wordt deze kracht nog maal de “baseRopeStretch” gedaan zodat dit makkelijk aan te passen is in de editor.

3.6 Hierna wordt de richting keer de kracht meegegeven als force aan de rigidbody. Als laatste wordt de speler in de richting van de camera gedraaid. Dit zodat het slingeren relatief is aan waar de speler naartoe kijkt.

Problemen:

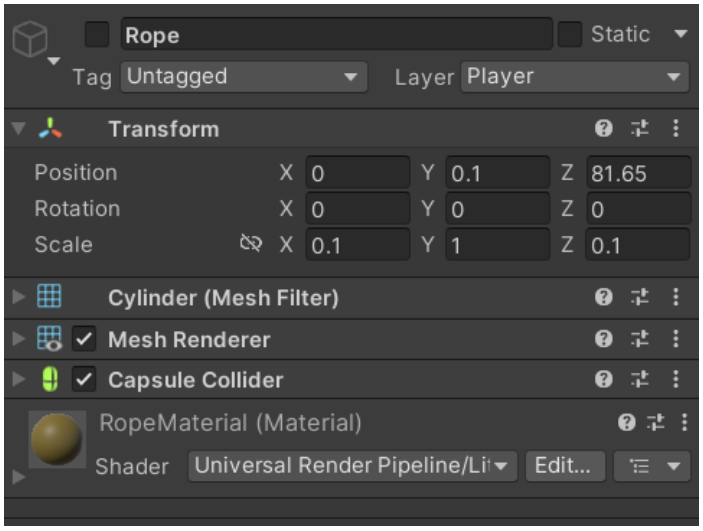
Er is geprobeerd om te testen hoe het zou zijn als het touw statisch was en niet langer kon worden dan de lengte van het touw, maar dit zorgde voor problemen bij het slingeren. Als de afstand van de speler tot het touwbeginpunt groter was dan de lengte van het touw dan werd de positie van de speler veranderd naar de positie op de lengte van het touw. Hierbij onstond het probleem dat de speler heel erg begon te schudden. Hier is uiteindelijk geen oplossing voor gevonden en dit heeft meegewogen met het uiteindelijke besluit om het touw vering te geven.

```
private void Hang()
{
    UpdateRope();

    rb.AddRelativeForce(new Vector3(movement.x, 0, movement.y) * swingStrength);
    if (Vector3.Distance(HangPoint, transform.position) >= ropeLength)
    {
        Vector3 forceDirection = HangPoint - transform.position;
        forceDirection.Normalize();

        ropeStretch = baseRopeStretch * (1 + (Vector3.Distance(HangPoint, transform.position) - ropeLength));
        rb.AddForce(forceDirection * ropeStretch);
    }
    transform.LookAt(transform.position + cam.forward);
}
```

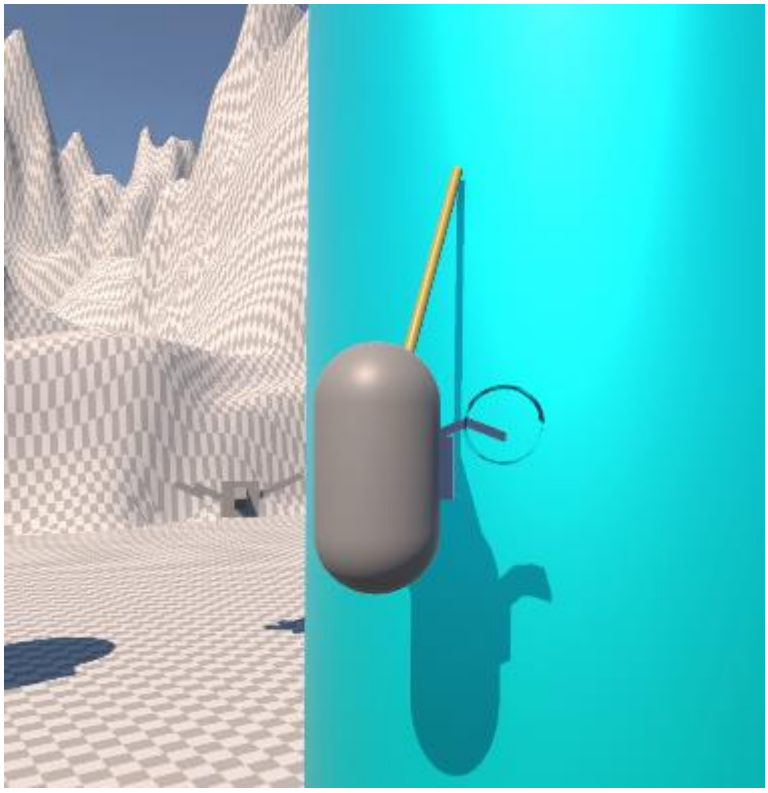
FIGUUR 4 DE HANG() METHODE

Stap 4: Het touw visueel

4.1 Het touw is voor nu een verdund cilinder object dat deel uitmaakt van de speler. Het touw heeft een eigen materiaal en is standaard niet actief.



FIGUUR 5 HET TOUW OBJECT



FIGUUR 6 HET TOUW IN DE GAME

4.2 Het touw wordt geüpdatet door de “UpdateRope()” methode. Hierin wordt het touw als eerste gedraaid. Dit wordt gedaan door gebruik te maken van “Quatenion.FromToRotation” waarbij de bovenkant van het touw wordt gedraaid naar de richting van de speler naar het beginpunt van het touw dat in stap 2.4 is uitgelegd (Unity Documentation Quaternion.FromToRotation).

4.3 Als volgende wordt het touw verplaatst. Hierbij wordt de positie van het touw gezet naar de helft van de afstand tussen het touwbeginpunt en de speler. Dit wordt gedaan omdat het middelpunt van het touw in het midden van de cilinder zit, dus wanneer we de schaal aanpassen wordt het touw aan beide uiteinden langer.

4.4 Als laatste wordt de schaal van het touw verandert. Dit wordt gedaan door de y waarde van de schaal de afstand tussen de speler en het touwbeginpunt te maken en dit te delen door 2.

```
private void UpdateRope()
{
    rope.transform.rotation = Quaternion.FromToRotation(rope.transform.up, (ropeAttachment - transform.position).normalized) * rope.transform.rotation;
    rope.transform.position = ropeAttachment - (ropeAttachment - transform.position) / 2;
    rope.transform.localScale = new Vector3(rope.transform.localScale.x, Vector3.Distance(HangPoint, transform.position) / 2, rope.transform.localScale.z);
}
```

FIGUUR 7 UPDATEROPE() METHODE

Stap 5: Het beëindigen van de actie

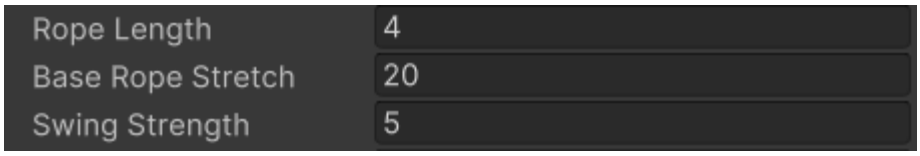
5.1 Wanneer de speler al in de “HANGING” staat is en daarna nog een keer op rechtermuisknop drukt of springt wordt de “StopHanging()” methode aangeroepen.

5.2 Als eerste wordt het touw weer inactief gezet en wordt de staat naar “WALKING” gezet.

5.3 Verder wordt de speler een beetje gelanceerd wanneer de speler omhoog aan het bewegen is. Dit wordt gedaan door eenmalig een kracht toe te voegen in de richting waarin de speler al beweegt. Er is voor gekozen om dit toe te voegen zodat de speler beter kan slingeren.

```
private void StopHanging()
{
    rope.SetActive(false);
    state = PlayerStates.WALKING;
    if (rb.velocity.y > 0)
    {
        rb.AddForce(new Vector3(Mathf.Abs(rb.velocity.x), Mathf.Abs(rb.velocity.y), Mathf.Abs(rb.velocity.z)) * 10);
    }
}
```

FIGUUR 8 STOPHANGING() METHODE



Stap 6: Aanpassen van variabelen

De volgende variabelen zijn in de Unity editor aan te passen.

6.1 “Rope Length” zorgt voor de afstand tussen het touwbeginpunt en de speler voordat de krachten beginnen te werken.

6.2 “Base Rope Stretch” is de kracht van het touw. Hoe hoger deze waarde des te groter de kracht, wanneer de speler verder is van het touwbeginpunt, die de speler richting het touwbeginpunt duwt.

6.2 “Swing Strength” geeft de speler meer horizontale kracht wanneer deze aan het touw hangt.

Validatie

Teamgenoten hebben de code gecontroleerd en hebben getest of de functie werkte zoals bedoeld was. Hierbij kwam naar voren dat de speler sneller moest vallen, waarna een van de teamgenoten de neerwaartse snelheid heeft toegevoegd die beschreven wordt in 2.4.

Verder is er getest naar de ervaringen, controles en mechanics. Hierbij is naar voren gekomen dat de mechanic op het moment niet het gewenste effect heeft. Er zou niet genoeg reden zijn om de mechanic te gebruiken. Er gaat verder gekeken worden naar of de feature nog een plaats heeft in de game en, als dat het geval is, hoe deze feature aangepast kan worden om hem meer toevoeging te geven.

Reflectie

Volgende keer wil ik meer kijken naar de toevoeging van bepaalde features en zorgen dat vanaf het begin duidelijk is wat precies de eisen en verwachtingen van een feature zijn, aangezien dat nu eerst gecontroleerd moest worden. Verder wil ik van tevoren meer onderzoek doen naar de verschillende mogelijkheden voor implementaties, omdat er online mogelijk meer manieren te vinden zijn dan ik zelf kan bedenken.

Ik heb tijdens het werken aan deze feature meer geleerd over het werken met rigidbodies en de mogelijke verschillen tussen posities aanpassen met de positie tegenover de rigidbody velocity (unity discussions: verschil tussen transform.position en rigidbody.velocity).

Bronnen

Unity documentatie: Quaternion.FromToRotation
https://docs.unity3d.com/ScriptReference/Quaternion.FromToRotation.html

Unity discussions: verschil tussen transform.position en rigidbody.velocity
https://discussions.unity.com/t/can-someone-help-me-understand-the-difference-between-transform-position-and-rigidbody-velocity/781113