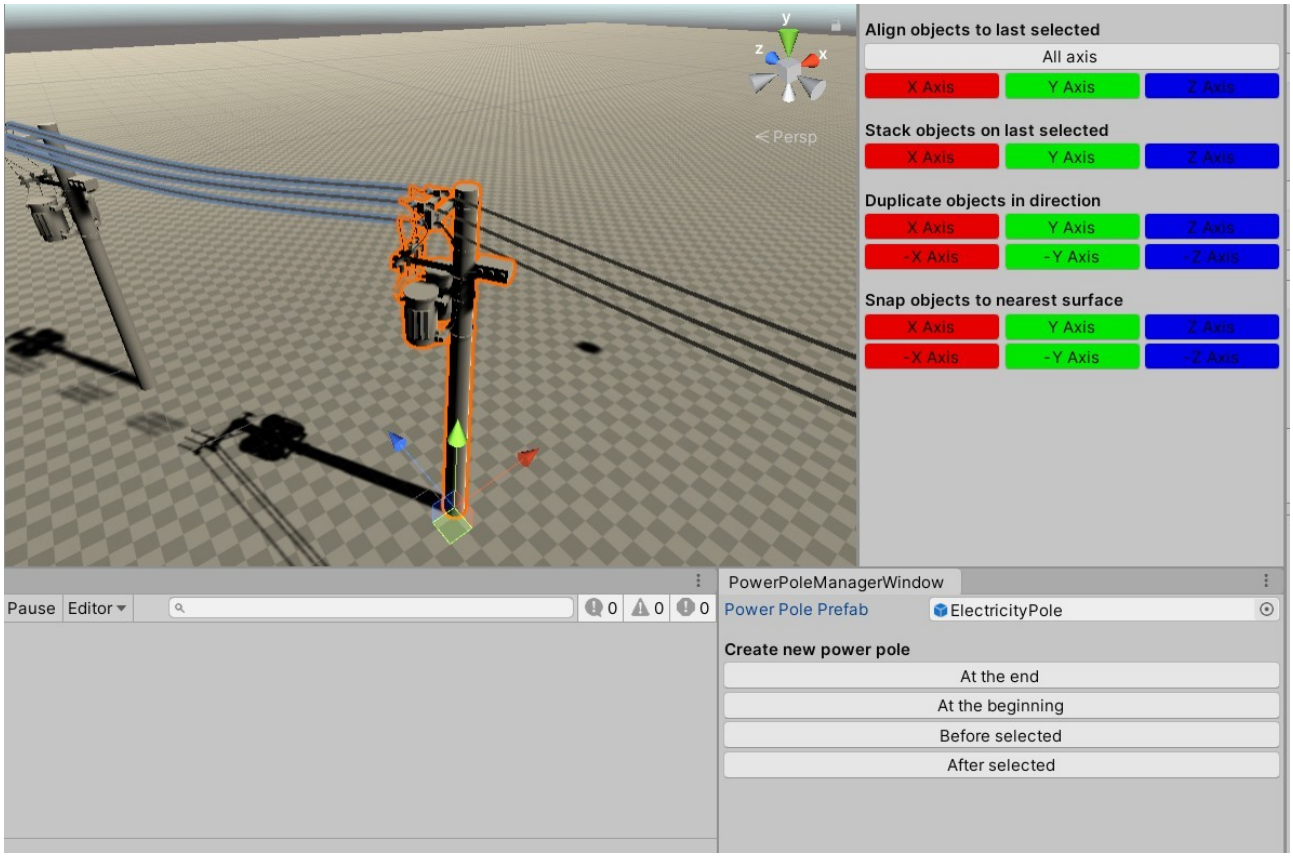
Herkansing Kernmodule 2
Game Development

Tjaard van Verseveld – 3028886



Juli 2020

Introductie

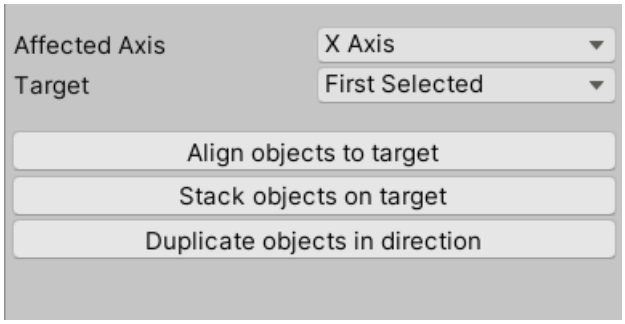
Voor deze opdracht moesten we een eigen tool maken die gebruikt kan worden om eigen gekozen handelingen te vergemakkelijken in de Unity editor. Hiervoor had ik aanvankelijk besloten om twee tools te maken die het voor designers eenvoudiger maakt om levels of omgevingen te ontwerpen. De eerste tool biedt de mogelijkheid om een selectie van objecten uit te lijnen, op elkaar te stapelen, of ze aan een dichtstbijzijnde oppervlak vast te maken (denk hierbij bijvoorbeeld aan een uithangbord aan een gebouw of een zwevend object dat op de grond gezet kan worden). De tweede tool is een waypointsysteem waarmee je objecten in serie kunt plaatsen en kunt passen. Hoewel er momenteel met beide veel mogelijk is is enkel het waypointsysteem volledig uitgewerkt, omdat ik tijdens het testte merkte dat deze tool de meeste belangstelling had.

Repertoire onderzoek

Op de assetstore kon ik niet heel veel tools vinden voor het positioneren van objecten. Een van de tools die ik vond was Align Tools. Het was makkelijk om hiermee te werken, omdat de functies duidelijk werden weergegeven met pictogrammen op de verschillende knoppen. Een nadeel was wel dat er niet enorm veel mogelijkheden aanwezig waren. In tegenstelling tot de positiesystemen waren er wel veel tools vindbaar om waypoints op te zetten. Echter was het merendeel hiervan bedoeld voor pathfinding en niet voor het plaatsen van objecten of in mijn geval elektriciteitskabels. Het was dan ook toen dat ik me besefte dat hetgeen wat ik zocht misschien een andere benaming had. De termen path en curve leverde wat interessantere resultaten, zoals deze tool https://assetstore.unity.com/packages/tools/utilities/b-zier-path-creator-136082

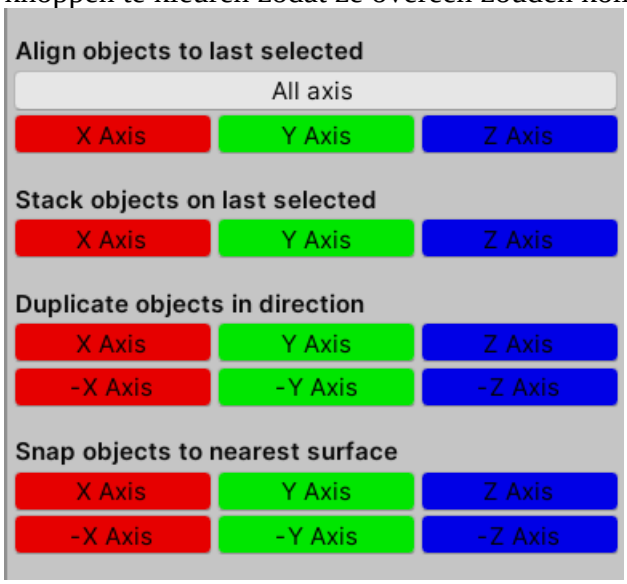
Proces

Ik begon met de tool om objecten te positioneren. De meeste functies hiervan waren niet extreem ingewikkeld en het duurde ook niet lang voordat ik veel opties toe had gevoegd. Wat langer duurde was het maken van het venster waarmee je al deze positioneringsfuncties met aanklikbare knoppen uit kon voeren, simpelweg omdat ik hier voor deze opdracht nog nooit eerder wat mee gedaan had. Wanneer ik dit voor elkaar dacht te hebben kwam ik door te testen met enkele mensen tot de ontdekking dat het ontwerp van dit venster niet bepaald goed was.



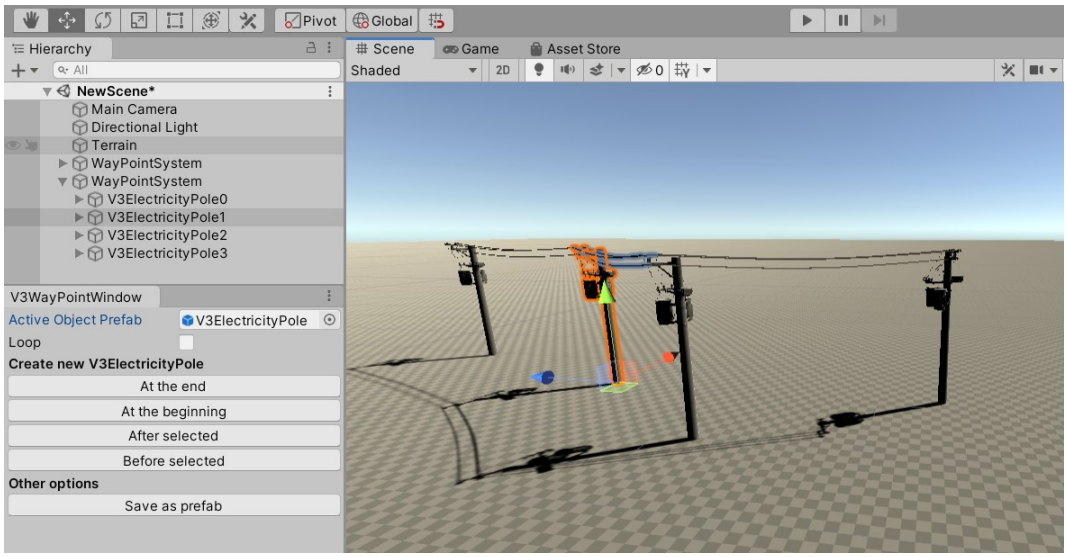
Ik had namelijk de meeste functies onder een knop gegroepeerd, omdat ze zich enkel van elkaar onderscheidde doordat ze de functie in een andere richting uitvoerde. Vandaar dat je bovenaan deze richting aan moest duiden voordat je de functieknop aanklikte. Bij het testen zag ik dat de meesten mensen eerst alle knoppen aanklikte en daarna pas ontdekte dat je bovenaan de richting in moest stellen. Een persoon stelde daarom ook voor of er niet voor elke richting een aparte knop gemaakt kon worden. Hoewel ik in het begin hier mijn twijfels over had leek het me geen slecht idee om het

alsnog uit te proberen. Na deze wijziging was het venster een stuk voller en leek het me slim om de knoppen te kleuren zodat ze overeen zouden komen met de kleur van de as die zij beïnvloeden.



Deze verandering in het venster zorgde er inderdaad voor dat mensen beter begrepen hoe ze met het venster konden werken, en hier meer mee gedaan konden krijgen. Echter waren een aantal mensen niet enorm onder de indruk van de mogelijkheden.

Vandaar dat ik hen nog een andere tool liet testen die ik gemaakt had om te gebruiken voor project vrij II. Met deze tool kon je stroompalen in serie plaatsen en aanpassen, zonder dat je daarvoor het spel moest draaien (wat eerst het geval was). Hierop was de reactie veel positiever en ik kreeg gelijk al een hoop suggesties voor wat ik hieraan toe kon voegen. Een van de eerste verandering die ik hieraan heb gemaakt is de mogelijkheid om elementen met delete te verwijderen, zoals dat gebruikelijk is in Unity in plaats van een knop die in het venster aanwezig was. Om dit voor elkaar te krijgen was niet bepaald makkelijk, omdat de opzet van de code niet heel goed was en ik daardoor vreemde uitzonderingsgevallen in een paar functies moest maken. Ik wilde het niet alleen mogelijk maken dat een waypointsysteem zichzelf juist bijwerkt wanneer een object met delete verwijderd werd, maar ook als een object teruggehaald zou worden als je vervolgens op control z drukt. Aangezien er niet heel veel tijd meer was toen ik hier klaar mee was besloot ik om zoveel mogelijk bugs op te lossen zodat ik de opdracht tenminste tijdig en in een zo goed mogelijk werkende staat in kon leveren en ik feedback kon ontvangen voor een eventuele herkansing.

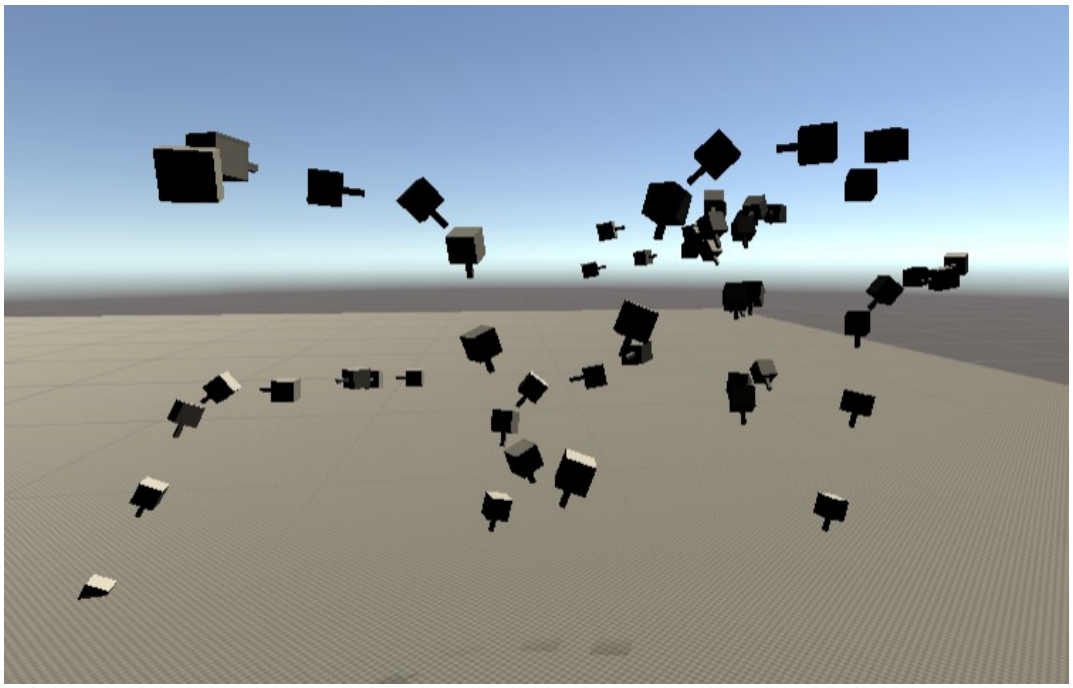


De feedback die ik ontving heeft me aardig op weg geholpen voor deze herkansing. Het eerste punt waar ik werk van heb gemaakt is dat het interessant zou zijn als het mogelijk is om iedere objectsoort te kunnen gebruiken voor het waypointsysteem in plaats van alleen stroompalen. Om dit voor elkaar te krijgen heb ik er eerst een ander punt uitgewerkt, namelijk dat het slim zou zijn om de code die verantwoordelijk is voor het waypointsysteem te scheiden van het venster waarmee deze systemen aangemaakt en aangepast worden. Dat zorgde er niet alleen maar voor dat de code in zijn algemeenheid overzichtelijker werd maar ook dat het makkelijker werd om de code aan te passen.

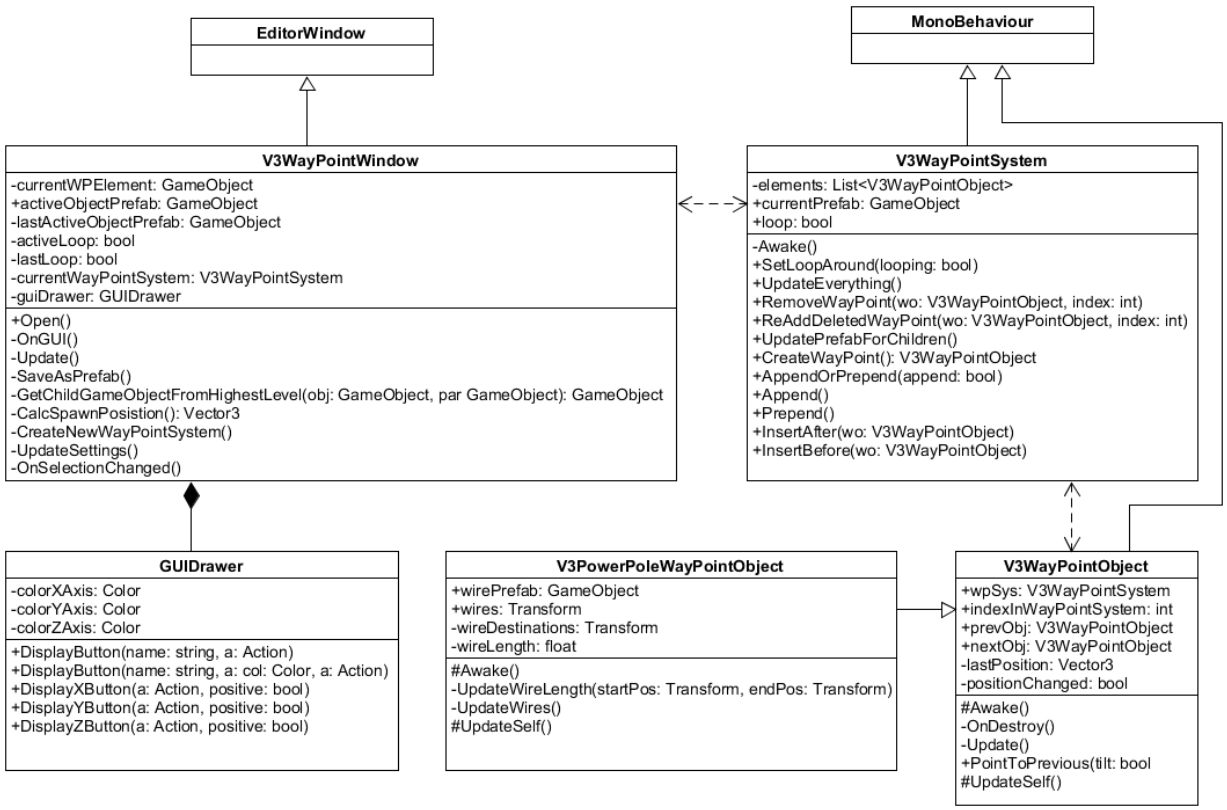
Om er daadwerkelijk voor te zorgen dat de waypoints met ieder type object om konden gaan heb ik na het fatsoeneren van de codestructuur het een en ander geprobeerd voordat ik uiteindelijk met een oplossing kwam. Deze oplossing was om een basisclass met een simpele updatefunctie te maken die bij ieder element in een waypointsysteem toegevoegd werd als deze niet reeds aanwezig was. Deze functie wordt bij ieder object in het waypoint aangeroepen als er een verandering aan het geheel opgemerkt wordt. Standaard zorgt het ervoor dat ieder object naar de vorige wijst, maar deze functie kan net zo goed uitgebreid worden door een class te maken die hiervan afstamt en deze toe te voegen op het object of prefab dat in het waypointsysteem is ingesteld. Zo heb ik bijvoorbeeld de oude class die ik gebruikte voor de stroompalen hiermee kunnen vervangen en allen de functies om de kabels naar andere palen te verbinden over moeten zetten.

Naast het generiek maken van het waypointsysteem heb ik er ook voor gezorgd dat je in kunt stellen of het geheel rond gaat of niet, wat handig is als je bijvoorbeeld een racecircuit met het systeem wilt maken en de weg volledig verbonden moet zijn.

Het laatste punt wat ik verwerkt heb is de optie om een waypointsysteem als prefab op te slaan. Deze prefabs kunnen zoals dat gewoonlijk in Unity is terug in een scene geplaatst worden. Het enige verschil met gewone prefabs is dat waypointprefabs gelijk uitgepakt worden, zodat ze in de scene aan kunt passen. Daarnaast is het zelfs mogelijk om een waypointprefab als prefab te gebruiken in een ander waypointsysteem, wat voor bijzondere resultaten kan zorgen.



UML Diagram


EditorWindow
MonoBehaviour
V3WayPointWindow
-currentWPElement: GameObject
+activeObjectPrefab: GameObject
-lastActiveObjectPrefab: GameObject
-activeLoop: bool
-lastLoop: bool
-currentWayPointSystem: V3WayPointSystem
-guiDrawer: GUIDrawer
+Open()
-OnGUI()
-Update()
-SaveAsPrefab()
-GetChildGameObjectFromHighestLevel(obj: GameObject, par GameObject): GameObject
-CalcSpawnPosistion(): Vector3
-CreateNewWayPointSystem()
-UpdateSettings()
-OnSelectionChanged()
V3WayPointSystem
-elements: List<V3WayPointObject>
+currentPrefab: GameObject
+loop: bool
-Awake()
+SetLoopAround(looping: bool)
+UpdateEverything()
+RemoveWayPoint(wo: V3WayPointObject, index: int)
+ReAddDeletedWayPoint(wo: V3WayPointObject, index: int)
+UpdatePrefabForChildren()
+CreateWayPoint(): V3WayPointObject
+AppendOrPrepend(append: bool)
+Append()
+Prepend()
+InsertAfter(wo: V3WayPointObject)
+InsertBefore(wo: V3WayPointObject)
GUIDrawer
-colorXAxis: Color
-colorYAxis: Color
-colorZAxis: Color
+DisplayButton(name: string, a: Action)
+DisplayButton(name: string, a: col: Color, a: Action)
+DisplayXButton(a: Action, positive: bool)
+DisplayYButton(a: Action, positive: bool)
+DisplayZButton(a: Action, positive: bool)
V3PowerPoleWayPointObject
+wirePrefab: GameObject
+wires: Transform
-wireDestinations: Transform
-wireLength: float
#Awake()
-UpdateWireLength(startPos: Transform, endPos: Transform)
-UpdateWires()
#UpdateSelf()
V3WayPointObject
+wpSys: V3WayPointSystem
+indexInWayPointSystem: int
+prevObj: V3WayPointObject
+nextObj: V3WayPointObject
-lastPosition: Vector3
-positionChanged: bool
#Awake()
-OnDestroy()
-Update()
+PointToPrevious(tilt: bool
#UpdateSelf()

Reflectie PMI

Positief

Ik heb kennis opgedaan hoe je eigen tools kunt maken, waardoor ik in de toekomst minder afhankelijk zal zijn van tools van anderen, of de portemonnee minder vaak hoef te trekken.

Minpunten

De tools hadden eerder getest kunnen worden, zodat ik in had kunnen zien dat er bepaalde eigenschappen aangepast moesten worden.

Interessant

Ik had me niet eerder beseft hoe extreem handig en tijdsbesparend het kan zijn als je met de juiste tools aan een project werkt, en ik ga dan ook voor toekomstige projecten hier gretig gebruik van maken.

Externe links

Broncode
Demonstratievideo