Uitbreidingsdocument voor SailAway voor pre set solutions

Dit document is gemaakt met het doel om de eerstvolgende persoon die werkt aan SailAway te helpen met het implementeren van vaste (handgemaakte) puzzels met behulp van de kaarten van het spel rush hour van thinkfun (ben je hier niet bekend mee, vraag dan voor het spel aan Carla Scholten). Houd tijdens het aanpassen van het spel in de gaten dat dit gemaakt is door een game designer met als doel een prototype voor dit spel (wat uiteindelijk toch wat verder is uitgewerkt). Comments die in de code staan zijn geschreven in het engels, dus het is enigszins vereist om engels te kunnen om dit project aan te pakken.

Aanpak om pre set solutions te maken (met de kaarten van het autospel)

Pre set solutions is nu een van de modes van het spel. Als je in de unity editor naar het gameobject “GameValues” gaat dan zie je ergens onderaan een checkbox met de naam Pre Set Solution. Als je deze aan zet, activeer je de pre set solution mode. Staat deze uit, dan gaat het spel terug naar de random solutions mode die al is uitgewerkt.

In de startup.cs file staat een speciale region waar alle functies in staan die te maken hebben met pre set solutions. Op het moment zijn dit de functies:

- `SpawnPreSetPlayerBoat()`
- `CreatePreSetField()`
- `CreatePreSetPuzzle()`

Van al deze functies moet je voornamelijk bezig zijn met CreatePreSetPuzzle. De andere twee functies zijn al ingesteld op de pre set solutions. SpawnPreSetPlayerBoat en CreatePreSetField zijn al geïmplementeerd en aangeroepen op de juiste plekken, dus hier zou je in principe niets extra mee hoeven doen.

In dit geval gaat de code ervan uit dat de exacte oplossingen van de fysieke kaarten in het spel gezet worden. Het grid van deze kaartjes is x = 6 en y = 6. Aangezien het script alleen werkt met oneven getallen, wordt x = 7 en y = 7 en is het de bedoeling dat je hier de meest bovenste rij (van y = 3) en de meest rechtse kolom (van x = 3) niet gebruikt om de boten te spawnen. Dit zorgt dus uiteindelijk voor een 6 bij 6 grid, zonder dat de code compleet omgeschreven moet worden.

In de functie CreatePreSetPuzzle() staat via comments aangegeven wat er in hoofdlijnen gedaan moet worden om de andere boten te plaatsen op de juiste manier. Daarnaast heb je waarschijnlijk ook een blauwe boot van 3 breed nodig en de art hiervoor is te vinden op de drive in de folder Smartwall development/Art, in de folder games, in de folder Boat Game.

Werken met grid tiles

De grid tiles in het spel worden met hele nummers aangegeven. In het geval van de pre set solutions zal de grootte van het grid altijd van x = -3 tot x = 3 gaan en van y = -3 tot y = 3. Op deze manier zijn alle waardes een stuk makkelijker te berekenen, maar de uiteindelijke positie van een boot komt niet volledig overeen met de waardes van FrontPos en BackPos (en uiteindelijk ook MidPos) aangezien de boten dan verkeerd zouden spawnen. Dit is belangrijke informatie om in je achterhoofd te houden wanneer je bezig bent met dit spel.

Andere functies waar je nog iets mee moet doen

Er zijn een paar andere functies die aandacht nodig zullen hebben om het spel compleet te laten werken. Het spel gaat op het moment nog altijd ervan uit dat boten een breedte hebben van 2, maar op de kaarten staan ook boten van 3 groot. Dit betekent dat er een extra check gedaan moet worden tijdens het selecteren van grid tegels en het bewegen van boten. Hier zou je gewoon een

eigen functie van kunnen maken en deze vervolgens aan het einde van de functie neerzetten waar de FrontPos en BackPos van een boot al gecheckt worden.

De volgende functies hebben zeker een vorm van extra check nodig voor boten die een MidPos hebben en dus 3 tiles breed zijn (dit kan door een specifieke functie daarvoor te maken en deze aan te roepen in de functies die hieronder staat of voeg het toe binnen de functie die hieronder staat):

- `Startup.cs: FindHitBoat()`
- `Startup.cs: FindPossibleMoveTiles() (aan te raden om hier een eigen functie aan het einde van deze functie bij te zetten aangezien deze functie vrij complex is op zichzelf)`
- `Startup.cs: FindPossibleMoveTilesPlayer() (ook hier is het aan te raden om een eigen functie aan te roepen aan het einde van deze functie)`
- `BoatControl.cs: voeg ergens een Vector2 MidPos variabele toe om boten met 3 tiles grootte mogelijk te maken`
- `BoatControl.cs: MoveBoat() aan het einde van deze functie worden op het moment alleen FrontPos en BackPos aangepast op het bewegen. Zet hierbij midPos.x += difference.x; en midPos.y += difference.y; om dit op te lossen.`

Als je al deze stappen juist hebt gevolgd en gebruikt, dan zou het spel nu moeten werken met pre set solutions van de fysieke kaartjes. Verder zou je alleen nog moeten zorgen voor een manier om alle verschillende oplossingen op een fatsoenlijke manier te kunnen aanroepen en selecteren. Hopelijk heeft dit document geholpen met het aanpassen en begrijpen van de code in SailAway.