Wholesome Gardening Behaviour overview

Player controls

Camera

* The player camera angle is fixed but dragging the cursor can change the position of the camera.

Clicking

* Clicking on a button in the corner will open the catalogue. Having the catalogue open will enable your active item.
* When the catalogue is open an item can be selected and change the active item. Once there is an active item the spaces on the grid will glow red or blue to indicate that a tile can be placed. Once clicked on an empty tile the active item will be placed.
* When you click on an occupied tile a context menu will pop up asking if you would like to discard the current item.
* Clicking on the watering can will enable you to click on a tile with a plant to water it. After clicking it will play a short animation where the can waters the plant in world space.

Wholesome Gardening Class overview

GardeningData

-A class dedicated to various structs and enums that we might to store in the data table

Enums

-ESoilType (sandy, silty, clay, peaty, chalky, loamy)

-EItemType (plant, object)

-EItemCategory (Fruit, Vegetable, Seeds, Soil, Decorations)

-ESunlightType (Sunny, Shade)

-EGrowthStage (Sapling, Medium, Grown)

Structs

-FSoilData (Inherits from FTableRowBase, contains a static mesh for the soil, ESoilType, ESunlightType)

-FItemData (inherits from FTableRowBase, contains an item name, an EItemType, an EItemCategory, and a UMaterialInstance\* for the inventory catalogue)

-FInventoryItem (contains an FItemData and int quantity)

-FPlaceableObjData (inherits from FTableRowBase, contains an FString name, a tile size, a static mesh

-FPlantData (Inherits from FTableRowBase, contains an FString name, a tile size, a TArray of static meshes, an ESoilType, an ESunlightType, an int number\_of\_stages and a float plant\_duration,

IPlaceable

-An interface specifically for the player raycasts/interactions with tiles that determine whether a tile is occupied and can be placed.

Functions

-bool IsOccupied(int size) { returns whether the tile is occupied }

-AItemBase\* GetCurrentItem() { returns pointer to current item }

-ESoilType GetSoilType() { returns soil type on the current tile }

-FItemData GetTileData() { returns data from the current tile }

-void PlaceItem(FItemData new\_item) { places the new item on the tile }

IWaterExtractable

-An interface specifically for the player raycasts with watercans and wells.

Functions

-bool IsEmpty() { returns whether the container is empty }

-void ExtractOneCharge() { takes one charge from the container and the class calling the function will arbitrarily add a charge themselves.

GardeningGameMode

-main method of retrieving information from other objects.

GardeningPawn

-Contains an inventory catalogue component, a pointer to an active item

Functions

-bool IsSoilCompatible(ESoilType plant\_soil, ESoilType tile\_soil, FString output)

-FPlaceableObject GetObjData(FString name) gets the object data from the active item’s id

-FPlantData GetPlantData(FString name) gets the plant data from the active item’s id

Grid

-Spawns a grid of 1x1 tile objects

-For each tile we’re going to raycast down from onto the floor to find the z position of the normal on the landscape and place the tile there.

Tile

-overrides from iplaceable, has an index, a pointer to a static mesh actor and a bool determining whether it is occupied as well as the soil data struct.

ItemBase

-The base class for all of our placeable objects. Contains an EItemType.

Plant

-Inherits from ItemBase, contains an FPlantData, an EGrowthStage for the current growth stage, a bool for whether the plant is watered, a float for current duration and two delegates

Functions

-void InitPlant(FPlantData new\_data) gives the plant class all the data it needs.

-bool IsWatered() returns is\_watered

-void StartGrowth() sets is\_watered to true

-void GrowPlant() increments the duration by delta time, once it reaches the duration it increments the growth stage and resets the current duration to 0.

-void StopGrowth() sets is\_watered to false

-void ChangeGrowthStage() changes the current growth stage and updates the mesh.

-void ReachNewStage() called every time a new state is reached

WaterContainer

-Inherits from the IWaterExtractable interface, contains a static mesh component, a bool for whether a container is infinite, an int max\_number\_charges and int current\_number\_charges.

InventoryCatalogue

-contains a TArray<FInventoryItem> filled with all of the current inventory items’ catalogue data.

Functions

-FInventoryItem GetItem(int id) { returns the item at the current index }

-bool ContainsItem(FString name) { returns whether an item exists in the inventory }

-void RemoveItem(int id) { removes an item or one quantity of the item from the inventory }

-void RemoveQuantity(int id, int amount) { removes a specific amount of the item from the inventory }

-void AddItem(FItemData new\_data) { adds new item to the inventory catalogue }

AmbientSoundClass

-Contains an audio component, a float volume modifier and a sound cue.

Functions

-void StartLoop() { starts the component’s loop }

Watering can and plant behaviour

Step 1

-Click on watering can

-Cast to item base, if this fails then we try and cast to watercontainer

Step 2

-If the cast to watercontainer is successful, get the item\_data and check the name is “Watering Can”

-Set the active item to the item\_data from the watering can

Step 3 – Click on tile

-if active item is watering can check the function IsEmpty, if not

-check the bool is\_extracting and is\_refilling, if both are false

-check the plant’s is watered function, if it’s false then continue

-call teleportopoint and pass in the position of the tile

-call settargetplant and pass in the plant from the tile

-call extractonecharge