Village Farm Game Kit

The game manager handles data, characters, and their various behaviors, where characters can embody different roles with behaviors related to diverse elements. Moreover, the game has prepared examples demonstrating management of different components such as markets, wholesale trade, retail trade, and even farm management companies. Additionally, it includes well-designed UX/UI that is beautifully presented.

Controllers

- Camera Controller
- Game Manager

Utilities

- Prices and Timer

Entities

- Market and Products Data

Characters

- Farmer
- Customer
- Villager
- Truck

Units

- Shelf and livestock

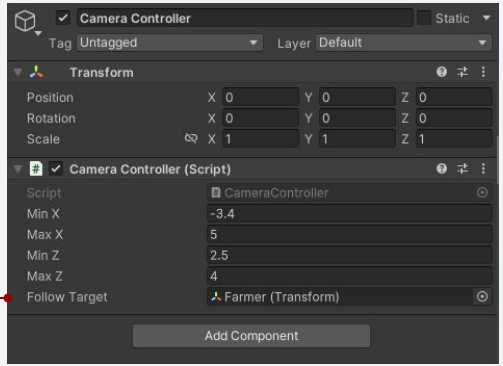
User Interfaces

- Dialogs and Templates

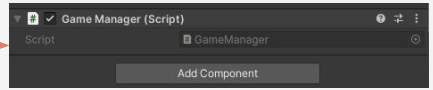
Controllers

Camera Controller

The camera controller follows the designated character. Furthermore, it can also limit the boundaries of the camera's tracking movement.



Controllers



Game Manager

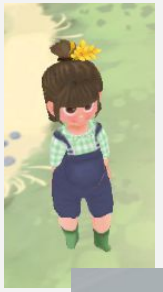
The game manager is designed with easily understandable scripts, and it utilizes the principle of independence so that developers can freely modify and add features as needed. It integrates various elements such as character behaviors with different roles, item management, and UI display management seamlessly.

```
using Clicknext.StylizedLocalVillage.Characters;
using Clicknext.StylizedLocalVillage.Entities;
using Clicknext.StylizedLocalVillage.UI;
using Clicknext.StylizedLocalVillage.UI.Dialogs;
using Clicknext.StylizedLocalVillage.Units;
using UnityEngine;

namespace Clicknext.StylizedLocalVillage.Controllers
{
    Unity Script (1 asset reference) | 0 references
    public class GameManager : MonoBehaviour
    {
        private Farmer _player;
        private Customer[] _customers;
        private Area[] _allArea;
        private Shelf[] _allShelves;

        private Area _currentArea;
        private OrderPopup _OrderDialog;
        private ShelfLabels _shelfLabels;
        private FloatingAmounts _floatingAmounts;
        private ThinkingBalloons _thinkingBalloons;
        private TransactionDialog _transactionDialog;
        private LevelDialog _levelDialog;

        Unity Message | 0 references
        private void Awake()...

        1 reference
        private void OnCustomerBought(Customer customer, Product product, int amount)...

        1 reference
        private void OnEnterShelf(Collider collider, Shelf shelf, Item targetType)...

        1 reference
        private void OnEnterArea(Collider collider, Area area, Product[] products, Product product)...

        1 reference
        private void OnExistArea(Collider collider)...

        1 reference
        private void OnGatherProduct()...

        1 reference
        private void OnChangedProduct(Product product)...

        2 references
        private void OnFarmerPickItem(Item item, int amount)...

        Unity Message | 0 references
        private void FixedUpdate()...
    }
}
```

For example, checking access to the farm's area to order agricultural product production.

Characters

This project implements the NavMesh Agent to allow different characters to navigate freely and independently find paths. Additionally, it incorporates distinct behaviors for each character. The farm area has been baked to enable navigation, and destination points have been placed for certain types of characters.



Farmer (Player)

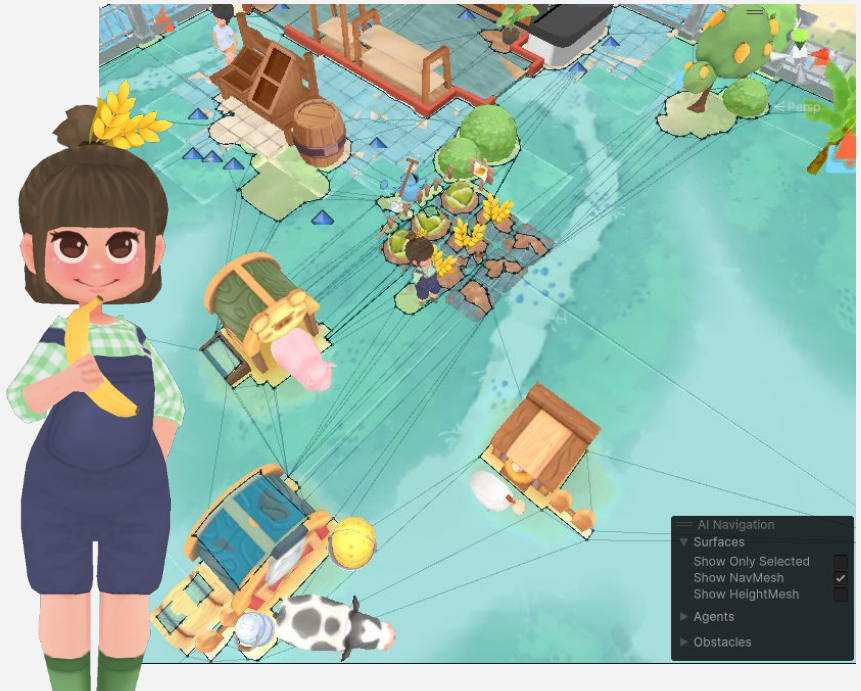
Farmers can store their produce in the produce warehouse, and players can control this character.

Customer

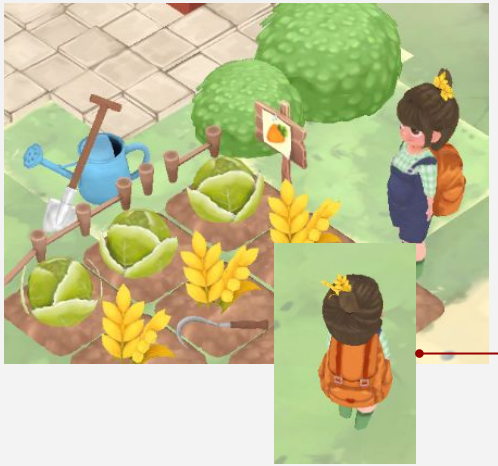
Every day, customers will come into the store section to buy either raw produce or cooked food. This is part of the retail trading aspect.

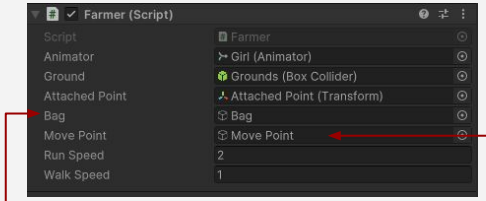
Villager

To create a beautiful scene, we've prepared scripts for the villagers' behaviors, allowing them to walk around and fill the gameplay scene. These scripts are also designed to be easily extendable and customizable according to your needs.



Characters



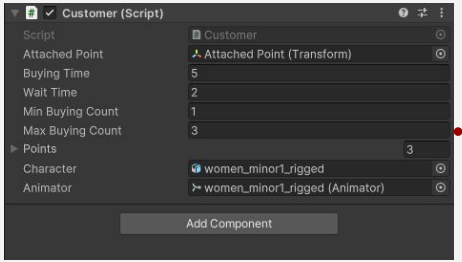


Farmer (Player)

Farmers will gather their produce into their own bags and take it to the sales shelves at each point to sell or use for food preparation.

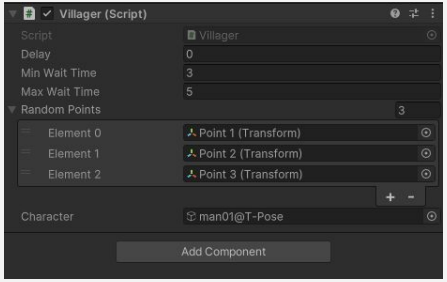
We implemented a raycast system to control the character's movement to different locations by clicking with the mouse or tapping with a finger on a mobile screen onto the colliders on the floor.

```
1 reference
private void CheckRaycast(Vector3 position, Action<Vector3> OnRaycast)
{
    Ray ray = Camera.main.ScreenPointToRay(position);
    if (Physics.Raycast(ray, out RaycastHit hit, Mathf.Infinity, mLayer))
        OnRaycast(hit.point);
}
```



Customer

Customers will enter the store and randomly choose to buy raw materials or food. They will walk from a designated point (Points) to the product display shelves that we have set up.



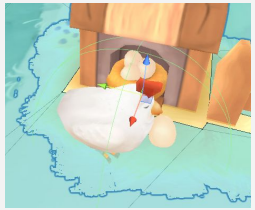
Villager (Supporting Actor)

The villagers will move from one point to another randomly.

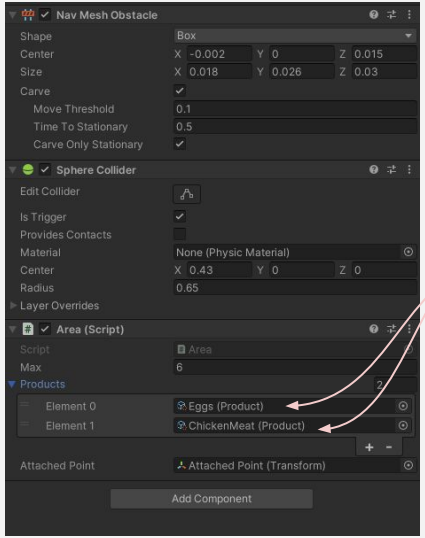
Units

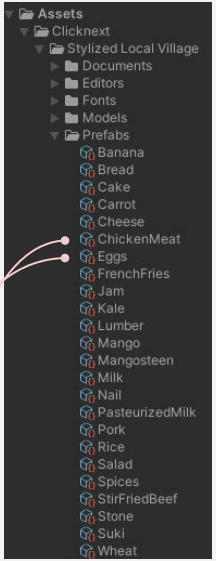
Area

In the farm unit area, there is a manager that specifies what players can produce as products. The system utilizes a sphere collider to check for entry into the area, which is the management area for product production.



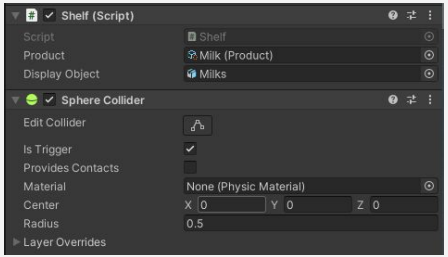
The added list of products will be displayed for players to choose what to produce. There will be a production time frame that players can set and adjust themselves.



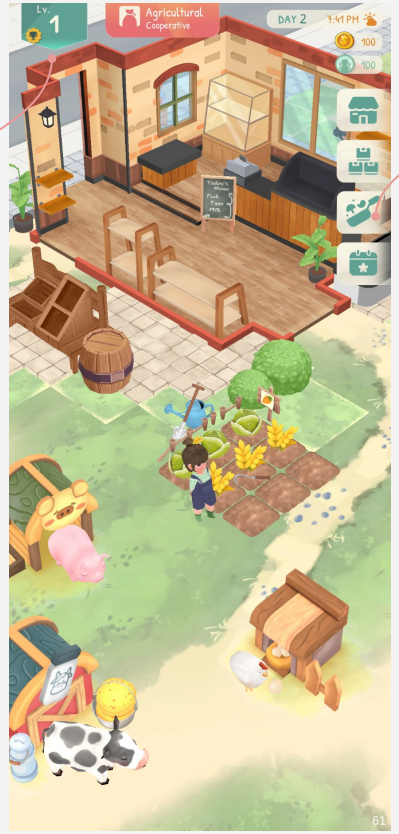


Shelf

At the same time, the product display shelves are units designed for storing items and serving as selling points for customers.

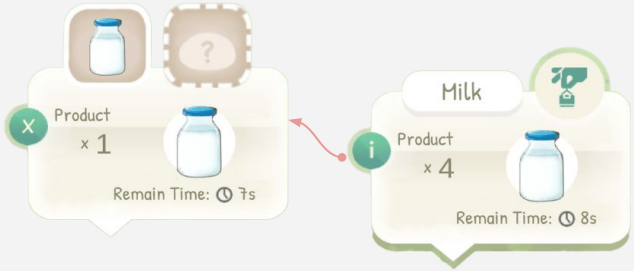


User Interfaces



Menu Canvas

The menu canvas is a UI element that uses the overlay technique to stay fixed across different parts of the screen. We've designed it to be cute, stylized, and suitable for the various model sets as much as possible.

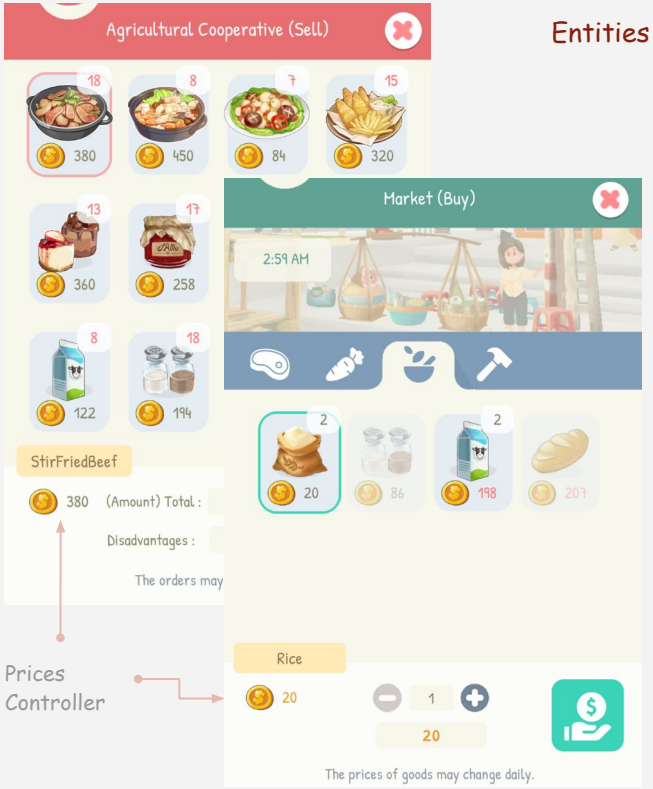


Floating Canvas

The floating canvas is a type of UI that we've designed to display the coordinates of 3D objects, and it has a relationship with the position of the screen. When players move their position, this UI box will show or hide according to conditions set for when players enter an area or meet certain criteria.

```
Unity Message | 0 references
private void Update()
{
    transform.position = Camera.main.WorldToScreenPoint(_targetPosition);
}
```

Entities and Utilities



Prices Controller

Entities

We organize a group of data sets separately from other parts and have them in the format of a static class. This includes markets, warehouses, wholesale stores, items, and even data sets in terms of value.

- Item
- Market
- Product
- Storage
- Vault

```
namespace Clicknext.StylizedLocalVillage.Entities
{
    14 references
    public static class Market
    {
        static readonly int minPrice = 10;
        static readonly int maxPrice = 100;
        static readonly int minAmount = 1;
        static readonly int maxAmount = 10;

        private static int[] prices;
        private static int[] amounts;

        0 references
        static Market() => ReGenerate();
```

```
namespace Clicknext.StylizedLocalVillage.Entities
{
    20 references
    public class Storage
    {
        public static event Action OnUpdated = delegate { };

        private static readonly int[] mStorage = new int[Enum.GetValues(typeof(Item)).Length];

        4 references
        public static void Add(Item type, int value)...

        4 references
        public static int Remove(Item type, int value)...

        10 references
        public static int Get(Item type) => mStorage[(int)type];

        1 reference
        public static bool IsEmpty()...
    }
}
```

(field) static readonly int[] Storage.mStorage

Timer

Utilities

Our system has a time flow that acts as a controller for product prices each day. These prices will change randomly to make managing harvested materials enjoyable for players.